

# **Security**

Sommersemester 2022  
(LV 4120, 7240)

dienstags, 08:15 bis 09:45

Prof. Dr. Bernhard Geib

### Worum geht es in der Einführung?

- Womit beschäftigt sich die Informationssicherheit?
  - ✓ Ziele und Merkmale
  - ✓ Themenbereiche und Abgrenzung
- Vorlesungsübersicht
  - ✓ Gliederung und Inhalte
  - ✓ Anwendungsbeispiele
- Organisation der Lehrveranstaltung
  - ✓ Vorlesung, Übungen und Klausur
  - ✓ Ablauf und Vereinbarung zur Leistungsbewertung
  - ✓ Hilfsmittel und Unterrichtsmaterial
  - ✓ Quellen- und Literaturangabe

### Die vier wesentlichen Sicherheitsbereiche:

Funktionale Sicherheit (Safety)		
<b>Computersicherheit IT-Sicherheit</b>	<b>Cyber-Sicherheit</b>	<b>Informationssicherheit (Security)</b>

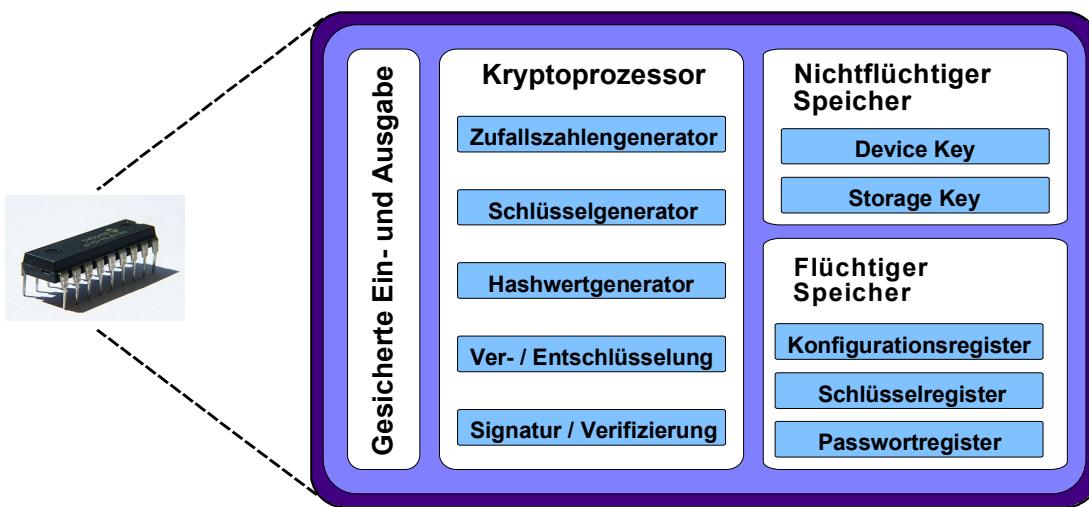
- Sicherheitsthemen und -fragestellungen sind seit Jahrzehnten allgegenwärtig.
- Täglich erreichen uns Meldungen über Identitätsdiebstahl und Datenklau durch Cyber-Kriminelle und andere Angreifer.
  - ✓ Methoden der Informationsgeheimhaltung
  - ✓ Datenunversehrtheit
  - ✓ Systemverfügbarkeit

## Kap. 1: Einführung und Motivation

### Einleitung:

- Worum geht es in dieser Lehrveranstaltung?
  - Was verstehen wir unter Security?
  - Wozu brauchen wir Informationssicherheit?
  - Welche Rolle spielt die Kryptologie?
  - Angestrebte Lernergebnisse (Zielsetzung)
  - Inhalte der Vorlesung und Gliederung
  - Organisation, Konzeption und Leistungsnachweis
  - Literatur und Hilfsmittel
-

## Worum geht es in dieser Lehrveranstaltung?



Trusted Platform Module  
(TPM 2.0)

Infineon SLB9665TT20

Quelle: *Bundesamt für Sicherheit in  
der Informationstechnik*

## 1. Kennenlernen von Krypto-HW

Sichere Infrastruktur für Trusted Computing

- Zertifizierungs- und Signierungsinfrastrukturen
- Schlüsselverwaltung für kritische Infrastrukturen
- Offene, vertrauenswürdige Datenverarbeitung

## Worum geht es in dieser Lehrveranstaltung?



ISDN - Bus- / Port- Schlüsselgerät ElcroDat 6-2

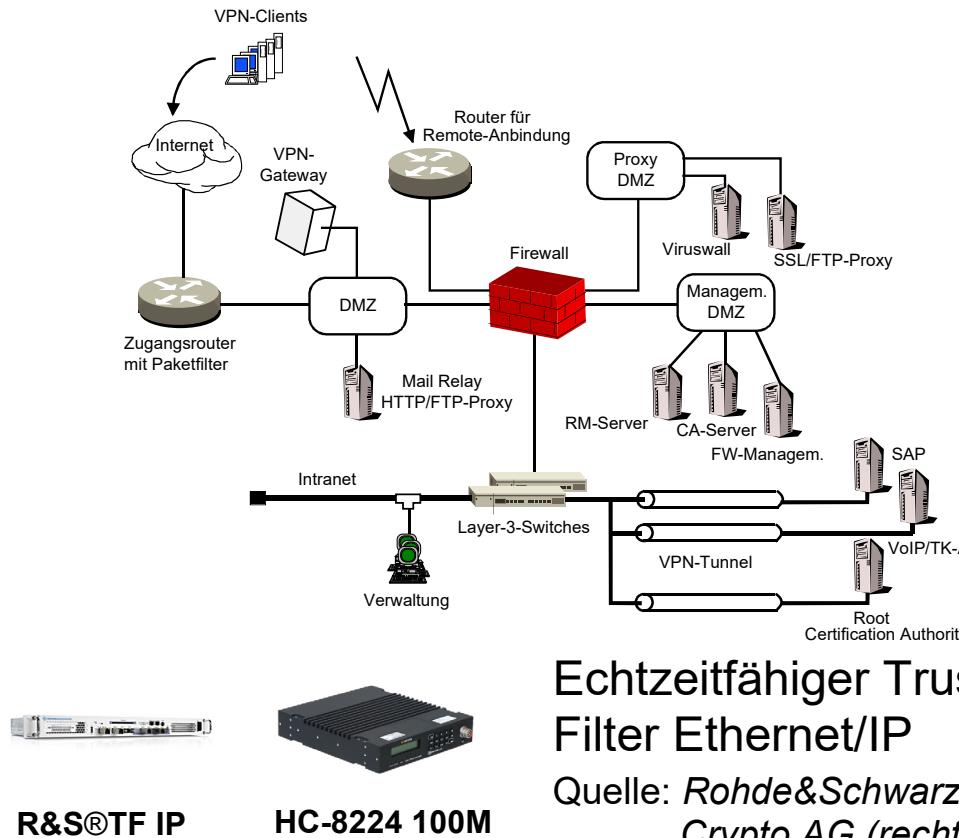
Quelle: *Bundesamt für Sicherheit in der Informationstechnik*

## 2. Anwendung von Krypto-Devices

Sichere Kommunikation  
(Verwaltung, Militär,  
Sicherheitsbehörden)

- Chiffrier- und Dechiffrierung
- Zufällige Schlüsselgenerierung
- Sprache, Daten, Video

## Worum geht es in dieser Lehrveranstaltung?



## 3. Absicherung einer IT&TK-Infrastruktur

Sicherer Übergang zwischen Sicherheitsdomänen

- Separierung von Ethernet- und IP-Netzwerken
- Zustandslose Protokollfilterung
- Netzwerkverschlüsselungsplattform

## Einteilung und Abgrenzung?

### Funktionssicherheit (engl. *safety*):

zielt auf Übereinstimmung der Ist-Funktionalität der Komponenten mit der spezifizierten Soll-Funktionalität ab (Gefahrenabweitung, Ausfallsicherheit, Schutz von Leib und Leben).

### IT-Sicherheit (engl. *IT-security*):

umfasst alle Vorkehrungen zum Schutz von elektronisch gespeicherten Informationen sowie informationstechnischen Systemen (SW & HW).

### Datenschutz (engl. *privacy*):

regelt die Verwendung und Weitergabe personenbezogener Daten (informationelles Selbstbestimmungsrecht gemäß BDSG & DSGVO).

---

## Was verstehen wir unter Informationssicherheit?

- ist gemeinhin ein Oberbegriff und beinhaltet sowohl die Cyber- und Internet-Sicherheit als auch die IT-Sicherheit.
- erweitert insbesondere die IT-Sicherheit um die Sicherheit von nicht technisch gespeicherten und nicht elektronisch verarbeiteten Daten.
- sicherstellen, dass es zu keiner unerlaubten Informationsveränderung, zu keinem unerlaubten Informationsgewinnung oder zu keiner unerlaubten Informationsvorenthalaltung kommt.
- verwendet den vom BSI entwickelten IT-Grundschutz als eine bewährte Methodik für den systematischen Aufbau eines dem individuellen Schutzbedarf angemessenen und wirksam angepassten IT-Systems.

### Was verstehen wir unter Kryptologie?

#### Kryptologie:

Wissenschaft der Verfahren zur Geheimhaltung von Nachrichten, aber auch zu deren Brechung. Kryptologie vereinigt Kryptographie und Kryptanalyse.

#### Kryptographie:

Geheimschriftkunde – offen versendete Nachrichten sollen durch Verschlüsselung bzw. Chiffrierung für Unbefugte nicht lesbar sein.

#### Kryptanalyse:

Meist mathematische und statistische Methoden zur Entzifferung von Geheimtexten, d.h. Informationen unbefugt erlangen.

---

### Wozu brauchen wir Kryptologie?

- Kryptologie ist als mathematische Disziplin wissenschaftlich fundiert und anerkannt.
- Mathematik liefert – jedenfalls im Prinzip – Rechtfertigung für die „Stärke“ einer Sicherheitsmaßnahme.
- Im Idealfall lässt sich beweisen, dass ein kryptographischer Algorithmus ein gewisses Sicherheitsniveau hat (oder halt nicht).

Damit kann der **Nachweis** erbracht werden, dass für eine bestimmte Anwendung der beanspruchte **Sicherheitswert** tatsächlich erreicht wird.

## **Angestrebte Lernergebnisse (Zielsetzung):**

Nach Absolvieren dieser Kurseinheit sollten Sie

- Verfahren zur Authentifizierung von Teilnehmern verstanden haben und auswählen können,
- Methoden der Informationsverschlüsselung einordnen, in ihrer Wirkung analysieren und in der Praxis anwenden können,
- Vorkehrungen zur Datenintegrität und Geheimhaltung sensibler Dateninhalte beurteilen und sicherstellen können,
- Konzept für Einweg- und Hashfunktionen verstanden haben sowie Probleme beim Schlüsselaustausch behandeln können.

## Typische Fragestellungen:

Aus Sicht eines Anwenders ergeben sich die Fragen

- Warum ist Sicherheit nötig (IT-Sicherheitsgesetz, kritische Infrastrukturen) und wie ist sie erreichbar?
  - Mit welchen Kosten ist Sicherheit verbunden?
  - Was ist für ein erfolgreiches E-Business (IT-gestützter Arbeitsablauf) nötig?
  - Wie ist die Risikolage (Gefahrenlage, Angreifer und Täter, Konsequenzen)?
-

## Inhalte der Vorlesung und Gliederung:

1. Einführung in die Informationssicherheit
  2. Algebraische Strukturen und elementare Zahlentheorie
  3. Monoalphabetische Chiffren und deren Analyse
  4. Symmetrische Verfahren und moderne Blockchiffren
  5. Einwegfunktionen
  6. Asymmetrische Kryptosysteme
  7. Schlüsselmittelmanagement und Zufallszahlen
  8. Kryptographische Protokolle und Anwendungen
-

## Organisation und Leistungsnachweis:

- Lehrform: Vorlesung und Praktikum / Übung
  - ECTS / SWS: **5 cp / 4**
    - 2 SWS Vorlesung
    - 2 SWS Praktikum / Übung
  - Gesamtaufwand: **150 h** (etwa 8 h pro Woche)
    - Anwesenheit Vorlesung und Praktikum 60 h
    - Vorbereitung und Nachbereitung Vorlesung 30 h
    - Bearbeitung der Praktikumsaufgaben 60 h
  - Leistungsnachweis: **Klausur** (90-minütig mit Formelsammlung)
-

## Konzeption der Lehrveranstaltungen:

### Vorlesungen

- Vorlesungen werden im Präsenz oder in Corona-Zeiten online jeweils für alle Studierenden des Semesters gemeinsam abgehalten.
- Die Vorlesung findet jeweils dienstags von 08:15 bis 09:45 Uhr statt.
- Anwesenheitspflicht besteht nicht.
- Die Lehrveranstaltung wird am Semesterende mit einer schriftlichen Prüfung (Klausur) abgeschlossen.
- Formale Voraussetzung für das Antreten zur Vorlesungsprüfung ist die rechtzeitig erfolgte Prüfungsanmeldung.

## Übungen und integriertes Praktikum

- Übungen dienen der praktischen Vertiefung und Ergänzung des Vorlesungsstoffs bzw. der Verständnisbildung.
  - Sie werden in Teilgruppen (25 bis 30 Teilnehmer) ca. und wöchentlich in Einheiten zu jeweils 90 Minuten durchgeführt.
  - Die Teilnahme am Übungsbetrieb bereitet die Studierenden gezielt auf die theoretischen und praktischen Anforderungen der Klausur vor (typische Sicherheitsthemen / sicherheitstechnische Fragestellungen).
  - Die Gruppeneinteilung erfolgt jeweils zu Beginn eines Semesters im Rahmen der Belegung.
  - Eine Beurteilung der Übungsteilnahme erfolgt nicht.
-

## Konzeption der Lehrveranstaltungen:

### Übungen und integriertes Praktikum (Fortsetzung)

- Die Teilnahme an den Übungen ist in Corona-Zeiten nicht verpflichtend, wird allerdings empfohlen.
- Integriert in theoretische Übungen sind Praktikumsaufgaben betreffend die Anwendung kryptographischer Algorithmen wie Verschlüsselung, Signaturen, Authentifizierung und Schlüsselmittelherstellung).
- Als Programmiersprache kommt C zur Anwendung.

### Sprechstunde:

**mittwochs zwischen 10:30 und 11:30 Uhr im Raum C 210  
oder nach Vereinbarung**

---

## Literatur und Hilfsmittel:

- Albrecht Beuelspacher: Kryptographie, Vieweg
- Wolfgang Ertel, Angewandte Kryptographie, Fachbuchverlag
- Johannes Buchmann: Einführung in die Kryptographie, Springer
- Claudia Eckert: IT-Sicherheit, Oldenbourg Verlag
- Ditmar Wütjen: Kryptographie, Spektrum Akademischer Verlag
- Bruce Schneier: Applied Cryptography, John Wiley & Sons

## Papers und Dokumentation zur Lehrveranstaltung:

[www.cs.hs-rm.de/~rnlab/LVaktuell/Security/](http://www.cs.hs-rm.de/~rnlab/LVaktuell/Security/)

## Zur Verfügung gestelltes Material:

- Vorlesungsfolien (Kapitel 1 bis 8) als PDF
- Grundlagen zur Zahlentheorie (Skript, 52 S.) als PDF
- Aufgabensammlung (passend zum Skript Zahlentheorie)
- Übungsunterlagen (Aufgabenblätter 1 bis 12) als PDF
- Übersicht der verwendeten kryptographischen Funktionen (Kryptolibrary mit 76 Modulen)
- Formelsammlung (abgestimmt auf Vorlesungsschwerpunkte)

Alles Weitere ist zu finden unter:

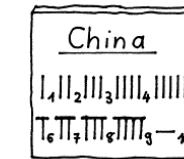
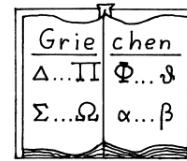
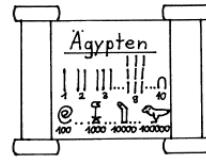
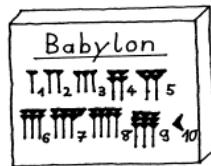
**[www.cs.hs-rm.de/~rnlab/LVaktuell/ Security/Material/](http://www.cs.hs-rm.de/~rnlab/LVaktuell/ Security/Material/)**

---

# Einleitung in die Vorlesung

Security

## Zur Verfügung gestelltes Material:

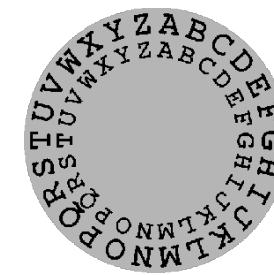
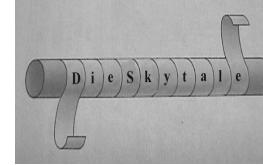


- = ≡ ≈ ≈   6 7 8 9 >
Indisch, um 300 v. Chr.
7 2 5 8 4   6 7 8 9 +
Indisch, um 800 n. Chr., mit der ersten Null der Welt

1 2 3 4 5   6 7 8 9
Arabisch, aus dem heutigen Spanien, um 1000 n. Chr.

1 2 3 4 5   6 7 8 9 +
Deutsch, 16. Jahrhundert

### Begriffe und Grundlagen der Zahlentheorie



Security

Sommersemester 2022  
(LV 4120 und 7240)

Formelsammlung

### Aufgaben- sammlung mit Beispiel- Lösungen

Security

Sommersemester 2022  
(LV 4120 und 7240)

Aufgabenblätter

## Kap. 1: Einführung in die Informationssicherheit

### Teil 1: Begrifflichkeiten

- IT-Systeme
- Sicherheitsbegiffe
- Aktuelle Sicherheitslage
- Effiziente Angriffstechniken

### Was ist ein IT-System?

Unter dem Begriff Informationstechnisches System (IT-System) versteht man jegliche Art elektronischer datenverarbeitender Systeme.

Kurz: Ein IT-System ist ein dynamisches technisches System mit der Fähigkeit zur Speicherung, Übertragung und Verarbeitung von Daten.

- Computer, Großrechner, Serversysteme, Datenbanksysteme
- Prozessrechner, digitale Messsysteme, Microcontroller-Systeme
- Informationssysteme, Kommunikationssysteme, Verteilte Systeme
- Betriebssysteme, eingebettete Systeme
- Mobiltelefone, Handhelds, digitale Anrufbeantworter, u.v.a.m.

### **Sicherheitsbegriffe:**

#### **Schwachstelle oder Sicherheitslücke:**

Fehler in einem IT-System, durch die ein Angreifer in ein Computersystem eindringen oder im IT-System Schaden verursachen kann.

#### **Bedrohung:**

Eine Bedrohung ist eine potentielle Gefahr mit zeitlichem, räumlichem oder personellem Bezug zu einem Schutzziel bzw. Schutzobjekt.

#### **Gefährdung:**

Trifft eine Bedrohung auf eine Schwachstelle (z. B. technische oder organisatorische Mängel), so entsteht eine Gefährdung.

### Sicherheitsbegriffe:

#### Risiko:

Ein Risiko ist das Produkt aus Eintrittswahrscheinlichkeit eines Ereignisses und dessen Konsequenz (Schadenshöhe) bezogen auf ein konkretes Schutzziel (Vertraulichkeit, Integrität, Verfügbarkeit).

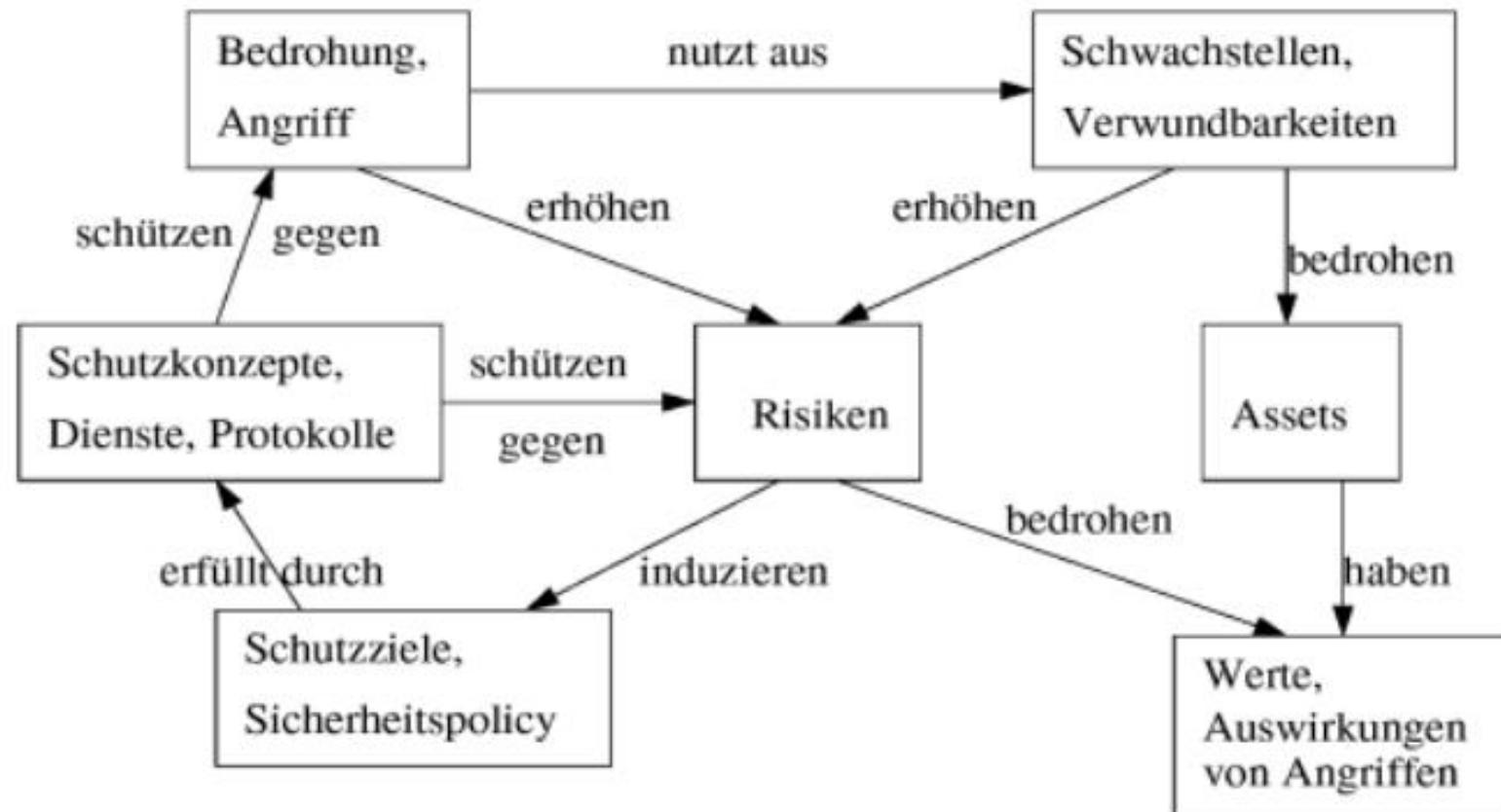
#### Eintrittswahrscheinlichkeit:

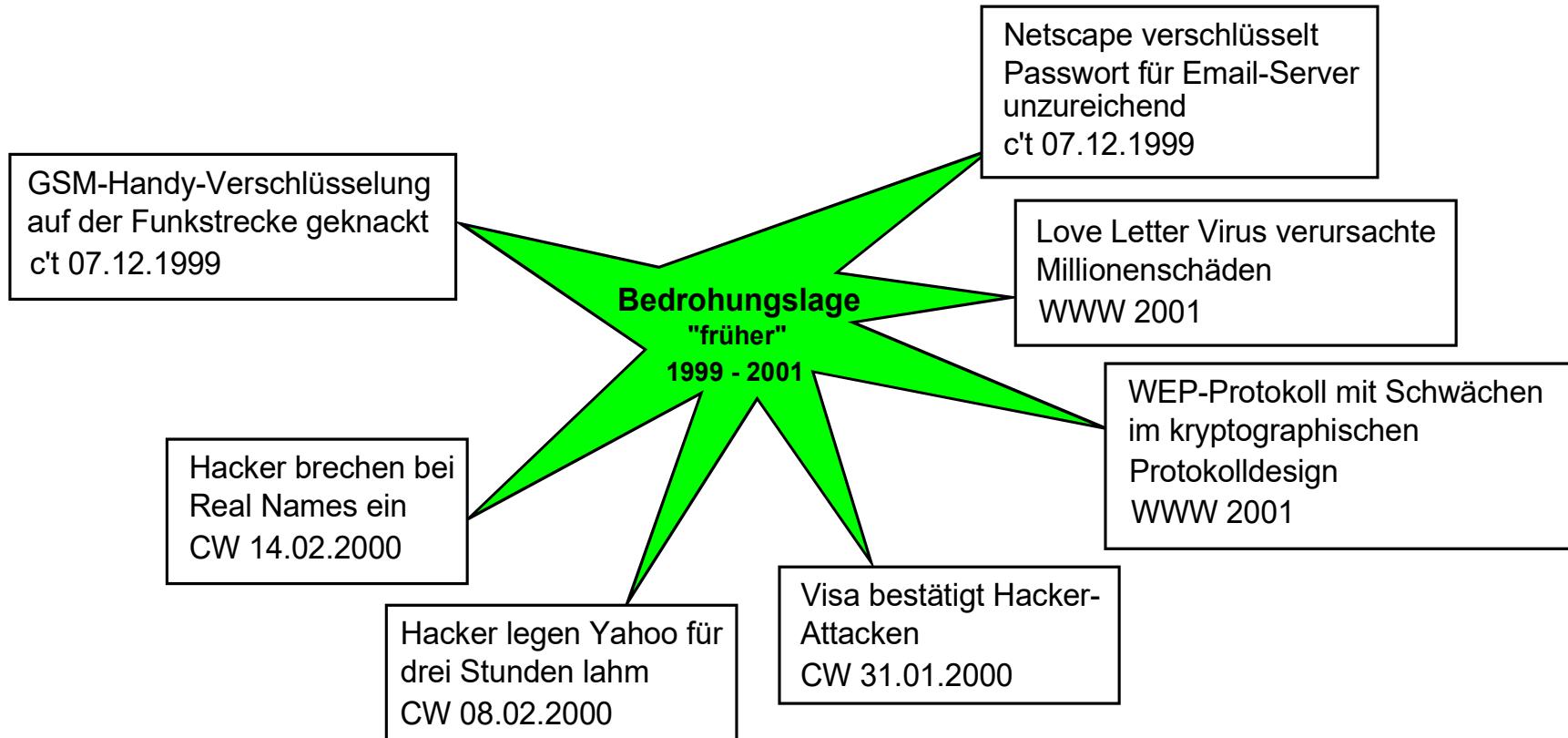
Ist die Wahrscheinlichkeit dafür, dass ein Schutzziel gebrochen wird.

#### Schadenshöhe:

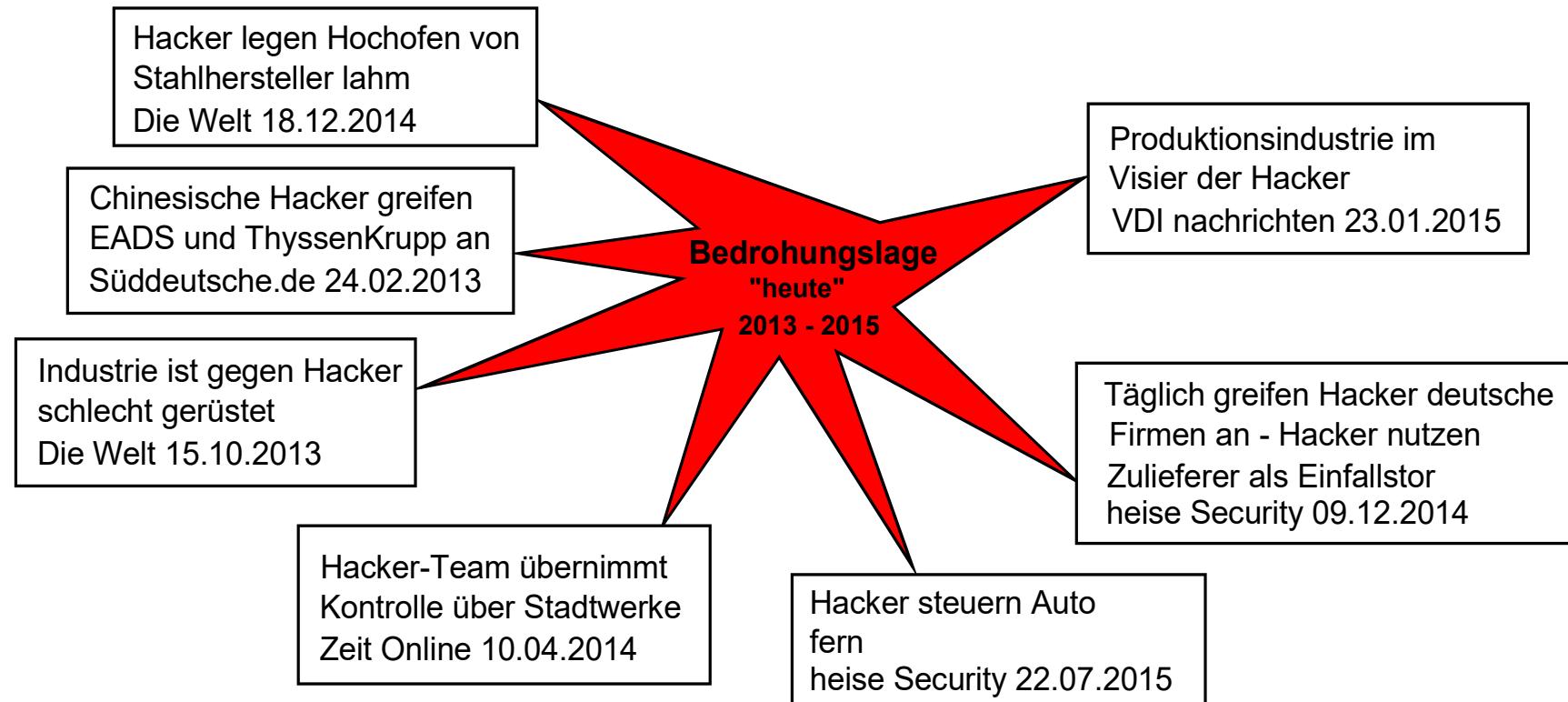
Höhe des Schadens (monetär oder nicht monetär), der sich aus einem Schadensszenario (erfolgreicher Angriff auf ein IT-System durch Ausnutzen einer Schwachstelle) ergibt.

---





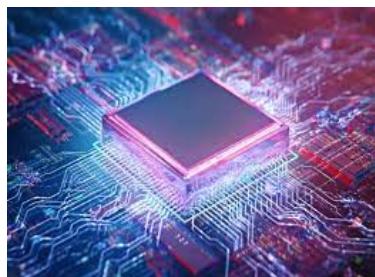
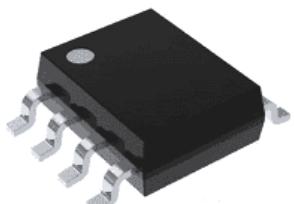
⇒ Lokal beschränktes Schadensausmaß



⇒ Angriffe mit existenzbedrohendem Schadensausmaß

### Fehlerinjektion:

Ziel der Angriffsmethode ist es, Kryptoprozessoren zu kompromittieren, indem man die Ausführung von Maschinenbefehlen unterbricht bzw. stört.



Dadurch können Befehle

- falsch geladen,
  - ungültige Daten weitergereicht oder
  - Instruktionen übersprungen
- werden.

Die Technik der Fehlerinjektion stammt aus den **1970er Jahren** als sie zum ersten Mal verwendet wurde, um Fehler auf Hardwareebene zu induzieren.

### Fehlerinjektion:

- Power/Clock Glitch Fault Injection (**PGFI**); Angriffspunkt: Pin-Zugriff; kurzfristiges Abschalten der Spannungsversorgung (voltage glitch) oder Einfügen eines Taktpulses (clock glitch)
  - Light/Laser Fault Injection (**LFI**); Angriffspunkt: Chip-Oberfläche; z. B. Optischer Bereich: 3 THz – 300 PHz (Peta); 100 µm – 1 nm
  - Elektromagnetische Fault Injection (**EMFI**); Angriffspunkt: Chip-Oberfläche; z. B. Mikrowellen (Hochfrequenz): 300 MHz – 300 GHz; 10 cm – 1 mm
  - Body Biased Fault Injection (**BBFI**); Angriffspunkt: Chip-Oberfläche; Hochspannungsimpuls auf das Schaltungssubstrat
  - Softwareimplementierte Fault Injection (**SWIFI**); Angriffspunkt: Software-system; unterschieden wird zwischen Kompilerzeit- und Laufzeit-Injektion
-

## Kap. 1: Einführung in die Informationssicherheit

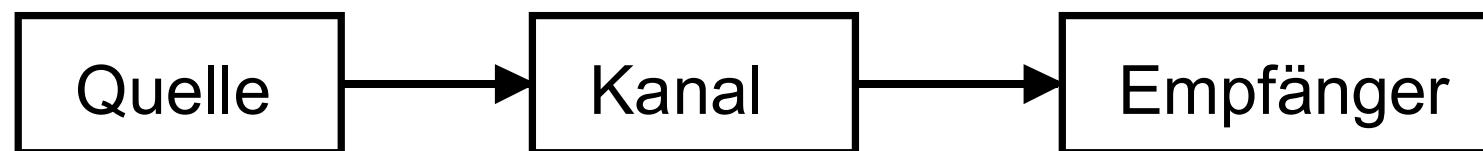
### Teil 2: Daten, Nachrichten und Informationen

- Terminologie
- Nachrichten- und Informationsmodelle
- Kryptosysteme

### Codierungstheorie (US-amer. Mathematiker Claude Shannon)

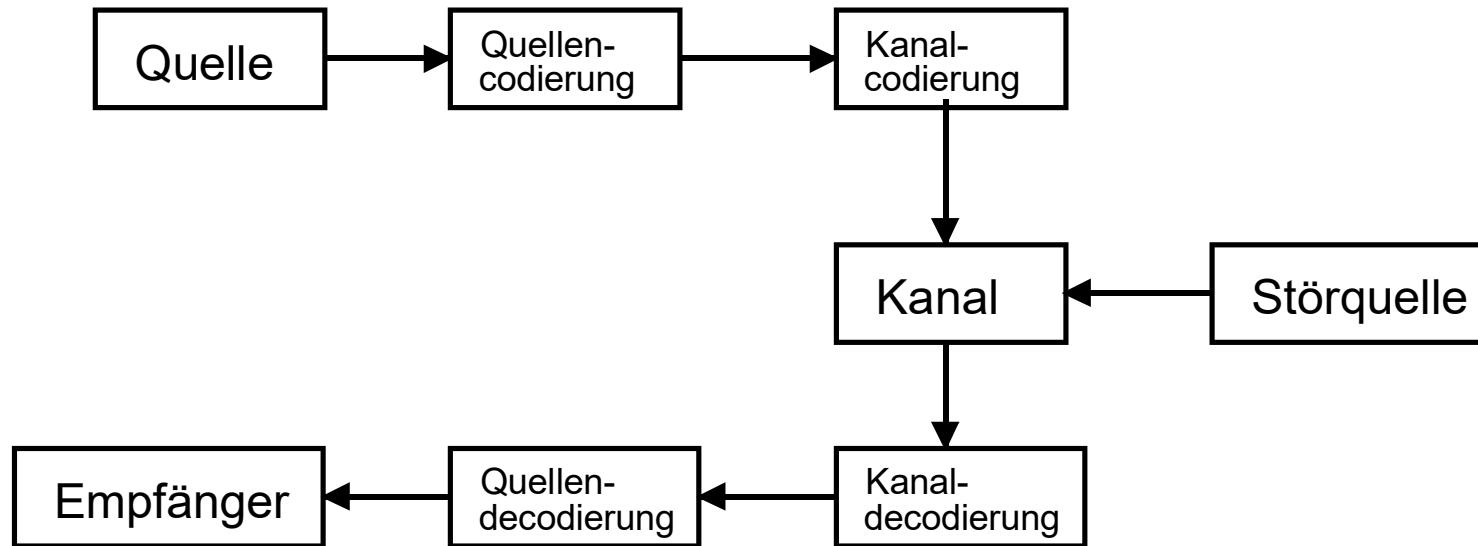
Nachrichten möglichst effizient und möglichst fehlerfrei übertragen bzw. speichern (z. B. Rundfunk, Fernsehen, Telefon, Datenspeichersysteme, Rechnernetze etc.)

Einfachstes Modell:



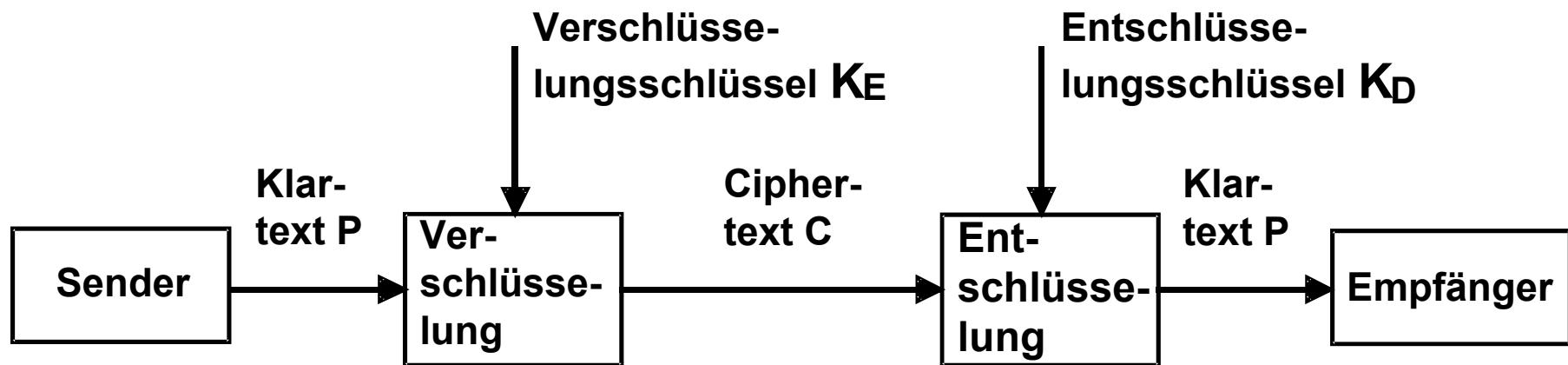
- Viele Quellen besitzen Redundanz (Weitschweifigkeit, Überbestimmtheit)
  - Fast alle Kanäle unterliegen Störungen (Rauschen)
-

Verfeinertes Modell:



- Eliminierung der Redundanz (Datenkompression oder Quellencodierung)
  - Gezieltes Hinzufügen von Redundanz (Kanalcodierung)
-

Kryptosystem:



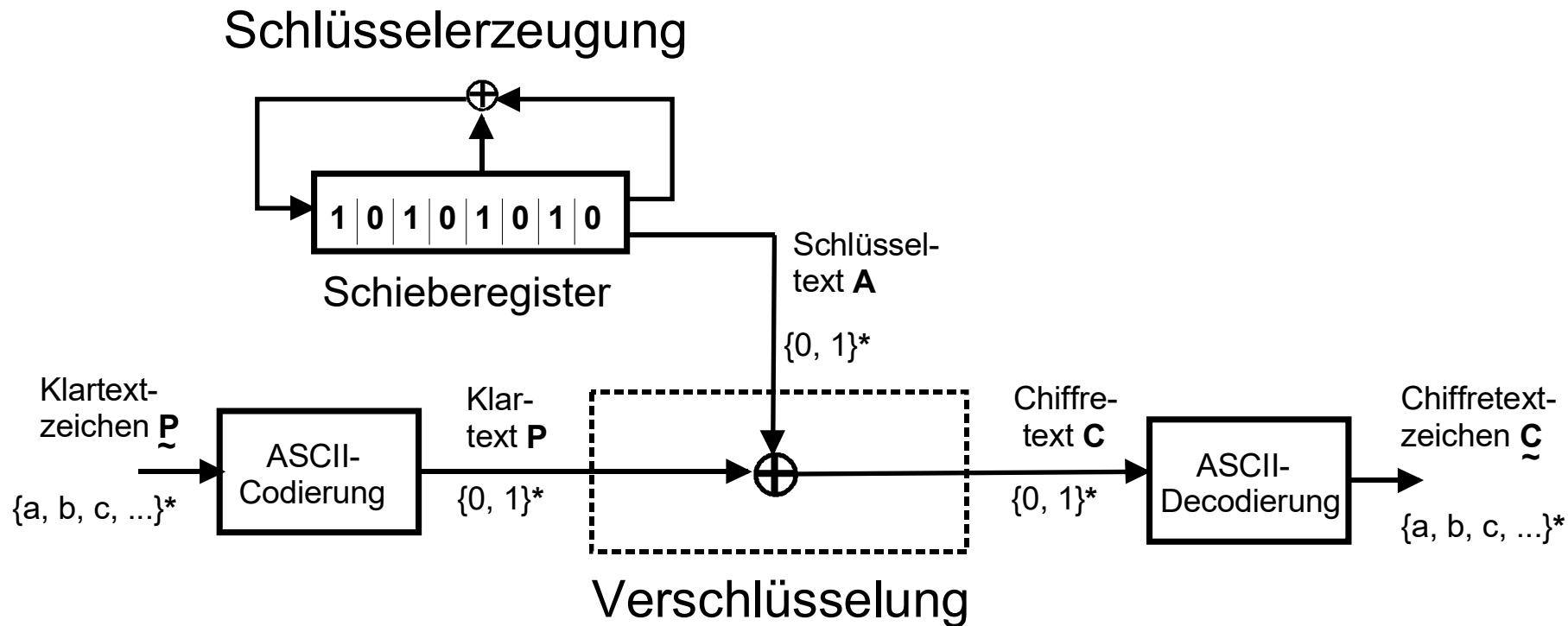
- Schlüsselgesteuerte Transformation (asymmetrisch)
- Formale Beschreibung durch das Quintupel: (  $P$ ,  $C$ ,  $K$ ,  $E$ ,  $D$  )  
( $P$  = Plaintext,  $C$  = Ciphertext,  $K$  = Key,  $E$  = Encryption,  
 $D$  = Decryption)

### Namensgebung:

Klartext <b>P</b> (plaintext)	lesbarer Text einer Nachricht (message), z. B. Buchstaben, Zahlenfolge, Zeichenkette etc., welche man vertraulich übermitteln möchte
Geheimtext <b>C</b> (ciphertext)	verschlüsselte, tatsächlich übermittelte Nachricht (Zeichenkette über dem gleichen Alphabet A oder einem anderen Alphabet B)
Schlüssel <b>K</b>	Geheimnis (Parameter, der in der Rechenvorschrift zur Anwendung kommt, Sicherheit → Kerckhoffs)
Chiffrieralgorithmus <b>E</b> bzw. <b>D</b>	Rechenvorschrift zum Ver- bzw. Entschlüsseln mit $\mathbf{C} := E(\mathbf{P}, \mathbf{K})$ und $\mathbf{P} := D(\mathbf{C}, \mathbf{K}^{-1}) = D(E(\mathbf{P}, \mathbf{K}), \mathbf{K}^{-1})$
Vorgang	chiffrieren = verschlüsseln → encryption (enc E) dechiffrieren = entschlüsseln → decryption (dec D)

---

## Prinzip einer Stromchiffre



Die 26 möglichen Verschiebechiffren des Alphabets:

Klartext:      a b c d e f g h i j k l m n o p q r s t u v w x y z

Chiffretexte:    0 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

                      1 B C D E F G H I J K L M N O P Q R S T U V W X Y Z A

**Schlüssel**        2 C D E F G H I J K L M N O P Q R S T U V W X Y Z A B

**k**                  3 D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

↓                    4 E F G H I J K L M N O P Q R S T U V W X Y Z A B C D

                      5 F G H I J K L M N O P Q R S T U V W X Y Z A B C D E

                      6 G H I J K L M N O P Q R S T U V W X Y Z A B C D E F

                      7 H I J K L M N O P Q R S T U V W X Y Z A B C D E F G

                      8 I J K L M N O P Q R S T U V W X Y Z A B C D E F G H

...

**25** Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

### Beispiel (Vigenère-Chiffre)

Plaintext:

- P = schwachstellenanalyse

Ciphertext:

- C = XHMBFHMXYJQQJSFSFQDXJ

Key:

- K = 5

Encryption:

- $E = z \rightarrow (z + k) \bmod n$  ; z = Plaintextzeichen

Decryption:

- $D = z' \rightarrow (z' - k) \bmod n$  ; z' = Ciphertextzeichen

**Prinzip einer Hill-Chiffre** (Lester S. Hill; 1891-1961, US-amer.)  
Mathematiker, Lehrer und Kryptograph

Ausgangslage:

- Restklassenring  $\mathbb{Z}_p = \{0, 1, \dots, p - 1\}$  mit  $p \in \mathbb{P}$  ist ein Körper.
- In einem Körper existiert die **modulare Inverse**.

Algorithmus:

Verschlüsselung:  $C = P \cdot K \pmod{p}$        $C$  = Chiffrat

Entschlüsselung:  $P = C \cdot K^{-1} \pmod{p}$        $P$  = Klartext  
 $K$  = Schlüsselmatrix

$C$ ,  $P$ ,  $K$  sind Matrizen, z. B.  $3 \times 3 \rightarrow 64$  Bit Blocklänge

---



(1912 – 1954)

### Alan Mathison Turing

- britischer Mathematiker und Kryptoanalytiker (Bletchley Park, 1943)
- einflussreichster Theoretiker der Computerentwicklung (Colossus)
- legte die theoretischen Grundlagen der frühen Informatik (Berechen- und Entscheidbarkeit)
- maßgeblich an der Entzifferung von Enigma-verschlüsselten Funksprüchen beteiligt

Von 1945 bis 1948 im National Physical Laboratory, Teddington, tätig am Design der Automatic Computing Engine (ACE)

---

### Shannonsche Theorie

Wichtige **Konstruktionsprinzipien** für die kryptographische Sicherheit sind Konfusion und Diffusion.

#### **Konfusion:**

Die Konfusion einer Blockchiffre ist dann groß, wenn die statistische Verteilung der Chiffretexte in Abhängigkeit von der Verteilung der Klartexte für den Angreifer zu groß ist (keine Ausnutzbarkeit).

#### **Diffusion:**

Die Diffusion einer Blockchiffre ist dann groß, wenn jedes einzelne Bit des Klartextes (und des Schlüssels) möglichst viele Bits des Chiffertextes beeinflusst (typisch etwa 50 %).

---

### Komplexität:

Das Entscheidungsproblem **PRIMES** besteht darin, zu entscheiden, ob es sich bei einer gegebenen natürlichen Zahl  $z > 1$  um eine Primzahl handelt. Dabei sei die Zahl  $z$  zur Basis  $b \in \mathbb{N}$  dargestellt.

Die dazugehörige Sprache sei mit  $L_b = L[\text{PRIMES}, b]$  bezeichnet.

Satz:

Sei  $L_1 := L[\text{PRIMES}, 1]$ . Erst 2002<sup>1)</sup> konnte gezeigt werden, dass gilt:

**$L_1$  liegt in P**

d. h. es gibt eine DTM, deren Laufzeit von der Ordnung  $O(n^3)$  und damit polynomial beschränkt ist.

- 1) Drei indische Mathematiker: M. Agrawal, N. Kayal und N. Saxena
-

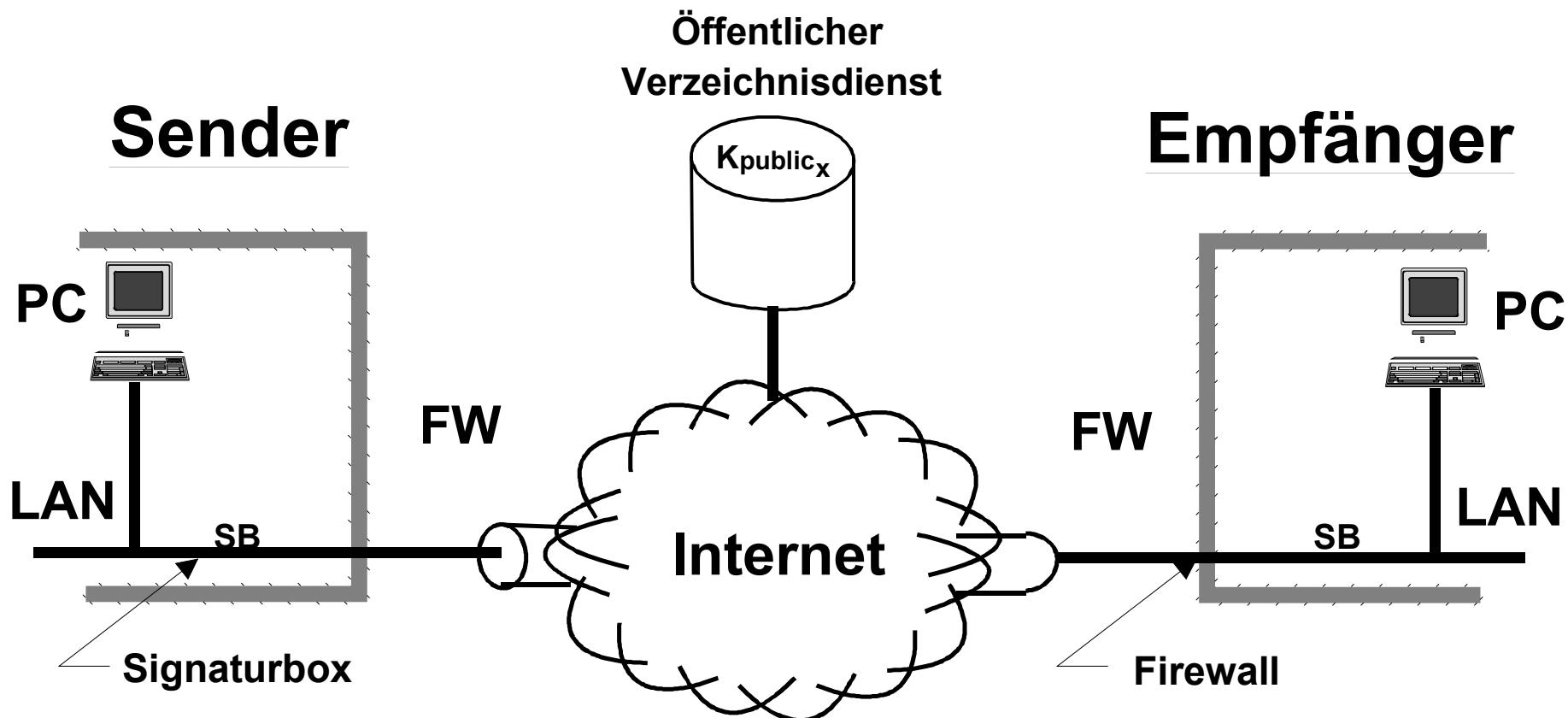
## Kap. 1: Einführung in die Informationssicherheit

### Teil 3: Schutzziele der Datensicherheit

- Begrifflichkeiten im Kontext von Datensicherheit
- Sicherheitsanforderungen und Sicherheitsziele
- Verschlüsselungsfunktionen und -algorithmen
- Kryptographische Hashfunktionen und digitale Signaturen
- Schlüsselmittel

# Datensicherheit

## Anwender-Infrastruktur



**Sicherheitsanforderungen** werden i. a. mit handelnden Subjekten und schützenswerten Objekten verknüpft.

- **was** soll geschützt werden? ⇒ schützenswerte Objekte
- **vor wem** oder was soll geschützt werden ⇒ handelnde Subjekte

Die (positive) Verknüpfung von Subjekten und Objekten wird im folgenden **Schutzziel** oder **Sicherheitsziel** genannt.

Zur Erreichung der Sicherheitsziele (Schutzziele) müssen geeignete **Sicherheitsdienste** bzw. **Sicherheitsfunktionen** und -maßnahmen bereitgestellt werden.

**Sicherheitsfunktionen** werden durch ihnen zugrunde liegenden **Sicherheitsmechanismen** (**Sicherheitsalgorithmen**) realisiert.

---

**Sicherheit** ist die Wahrscheinlichkeit, einen bezifferbaren oder nicht bezifferbaren Schaden zu verhindern oder zumindest auf ein erträgliches Restmaß (Restrisiko) einzuschränken ⇒ Schutzziele

### Grundlegende Sicherheitsziele:

Vertraulichkeit → Schutz gegen unautorisierte Kenntnisnahme

Integrität → Schutz gegen unautorisierte Veränderung

Verfügbarkeit → Schutz gegen unautorisierte Vorenthalaltung/  
Verweigerung

Verbindlichkeit → Schutz gegen Verlust der Beweisbarkeit/  
(Authentizität) Zurechenbarkeit und nicht Abstreitbarkeit

---

## Grundlegende Sicherheitsfunktionen und -maßnahmen:

### 1. Vertraulichkeitsschutz

**Kryptographische Algorithmen** sind Berechnungsvorschriften, d. h. mathematisch / logische Funktionen zur Ver- und Entschlüsselung von Nachrichten.

Bei **symmetrischen Algorithmen** wird zum Chiffrieren und zum Dechiffrieren immer der gleiche Schlüssel **K** benutzt.

Bei **asymmetrischen Algorithmen** werden zum Ver- und Entschlüsseln zwei unterschiedliche Schlüssel **K<sub>1</sub>** bzw. **K<sub>2</sub>** benutzt, die allerdings miteinander korrespondieren. Es gilt:

$$C := E(M, K_1) \text{ und } M := D(C, K_2) = D(E(M, K_1), K_2)$$

## Grundlegende Sicherheitsfunktionen und -maßnahmen:

### 1. Vertraulichkeitsschutz (Fortsetzung)

Man unterscheidet bei Kryptoalgorithmen zwischen **Stromchiffren** und **Blockchiffren**.

- Stromchiffren: Zeichen für Zeichen
- Blockchiffren: Nachricht **M** in Blöcke z. B. der Länge  $n = 64$  Bit aufgeteilt

Die Vereinigung von Algorithmus, zugehörigen Schlüsseln und den verschlüsselten Nachrichten (Kryptogramme) wird **Kryptosystem** genannt.

Der **Schlüsselraum**, d. h. die Menge, aus der ein Schlüssel gewählt wird, sollte möglichst groß sein. Er sollte mindestens so groß sein, dass der Aufwand (Kosten, Zeit, Speicherplatz/Datenmenge) für einen Angriff unakzeptabel hoch wird.

---

## Grundlegende Sicherheitsfunktionen und -maßnahmen:

### 2. Integritätsschutz

Man unterscheidet beim Integritätsschutz zwischen **Hashfunktionen** und **digitalen Signaturen**.

Eine Hashfunktion **hash** ist eine Abbildung, die für eine beliebig lange Nachricht **M** einen Funktionswert **H** (den Hashwert) fester Länge liefert.

$$H = \text{hash}(M)$$

Darüber hinaus muß sie gewisse Bedingungen (Einwegeigenschaft, Kompressionseigenschaft, Kollisionsfreiheit) erfüllen.

Eine Besonderheit sind schlüsselabhängig Hashfunktionen, sogenannte **Keyed-Hash Message Authentication Code (HMAC)**.

---

## Grundlegende Sicherheitsfunktionen und -maßnahmen:

### 2. Integritätsschutz (Fortsetzung)

Eine digitale Signatur ist ein Datensatz **Sig<sub>T</sub>**, der zusätzlich zu einem Dokument **M** erzeugt wird und dabei das signierte Dokument eindeutig einem Teilnehmer **T** zuordnet:

$$\mathbf{Sig}_T = \mathbf{sig}(H, Sk_T)$$

Verwendung findet bei der Signaturerstellung der **geheime** Schlüssel des Teilnehmers T.

Bei der Signaturprüfung wird der zugehörige **öffentliche** Schlüssel des Teilnehmers T benötigt.

⇒ Public Key System

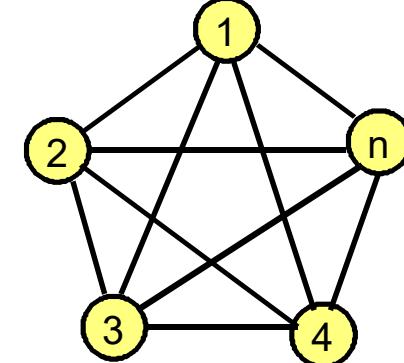
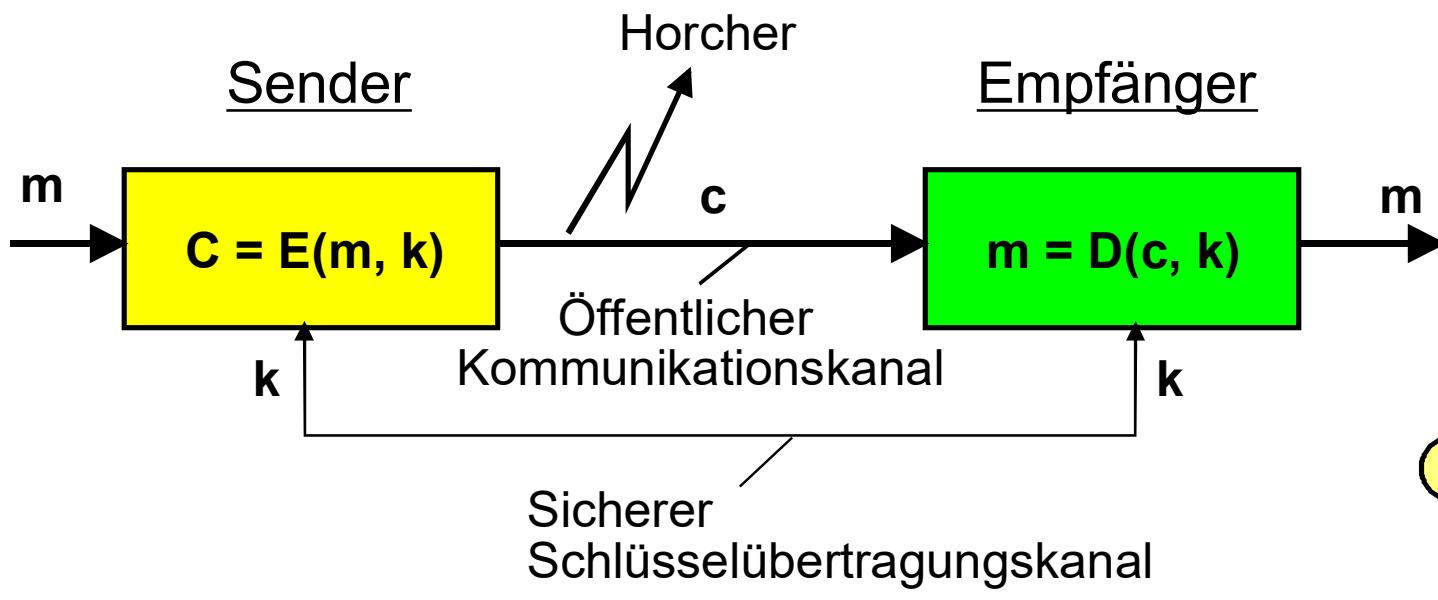
## Kap. 1: Einführung in die Informationssicherheit

### Teil 4: Basismechanismen der Kryptologie (Überblick)

- Symmetrische Ver- und Entschlüsselung
- Asymmetrische Ver- und Entschlüsselung
- Kryptographische Hashfunktionen
- Message Authentication Code
- Digitale Signaturen
- Schlüsselmanagement und Zufallszahlen

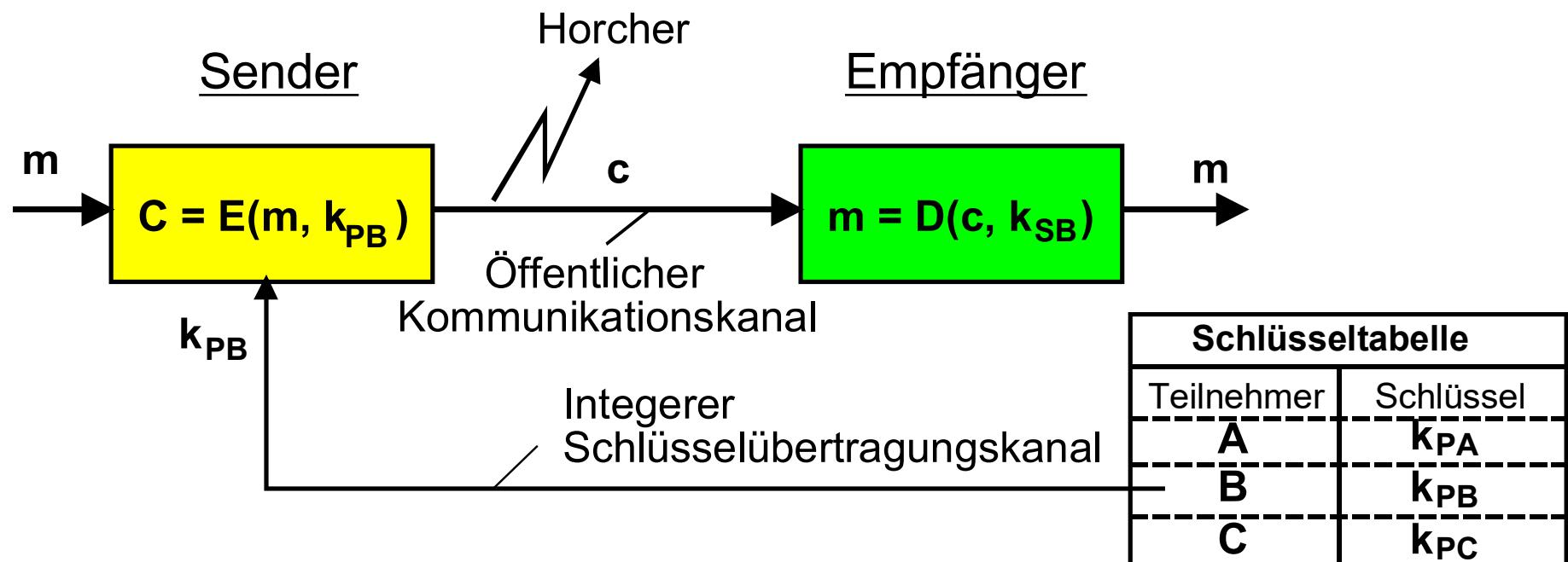
# Basismechanismen im Überblick Symmetrische Verschlüsselung

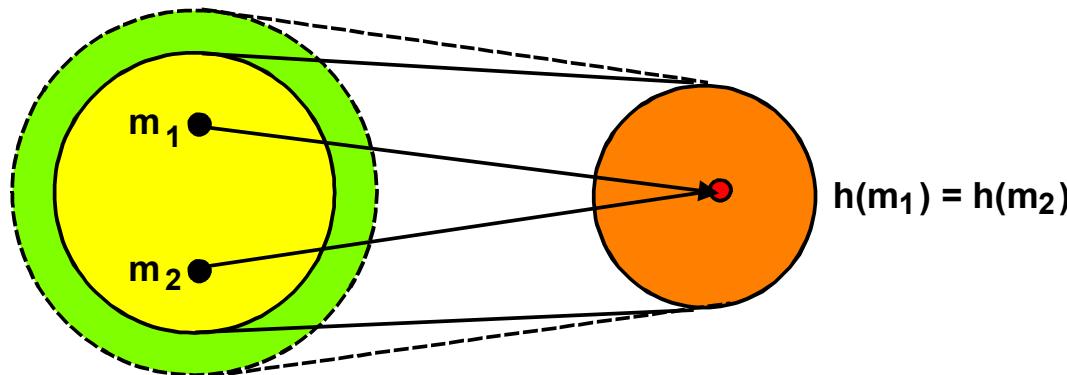
---



# Basismechanismen im Überblick Asymmetrische Verschlüsselung

---





hier:

Prinzip einer Hashfunktion  
**mit Kollision**

### Basisprinzip:

- Nachricht beliebiger und Hashwert fester Länge (typ. 128 Bit)
- Digitaler Fingerprint

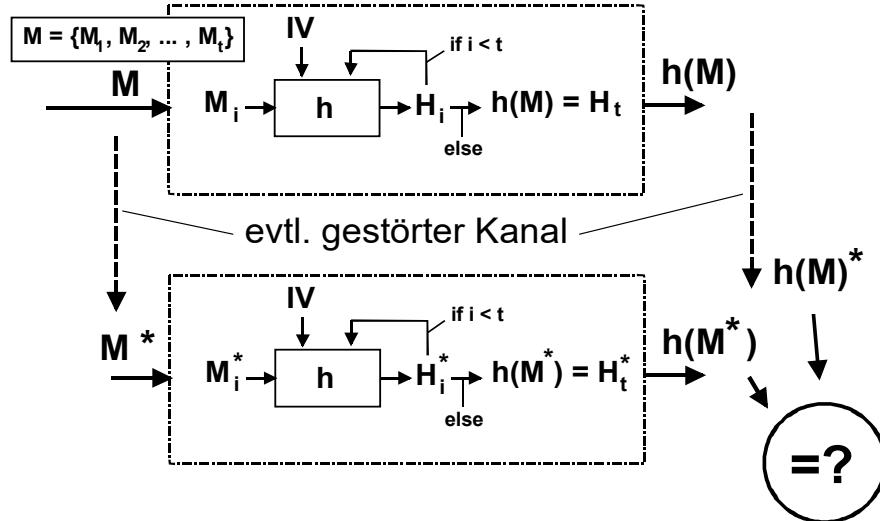
### Eigenschaften:

- kollisionsresistent
- mit und ohne geheimen Schlüssel ( $\rightarrow$  **MD** bzw. **MAC**)

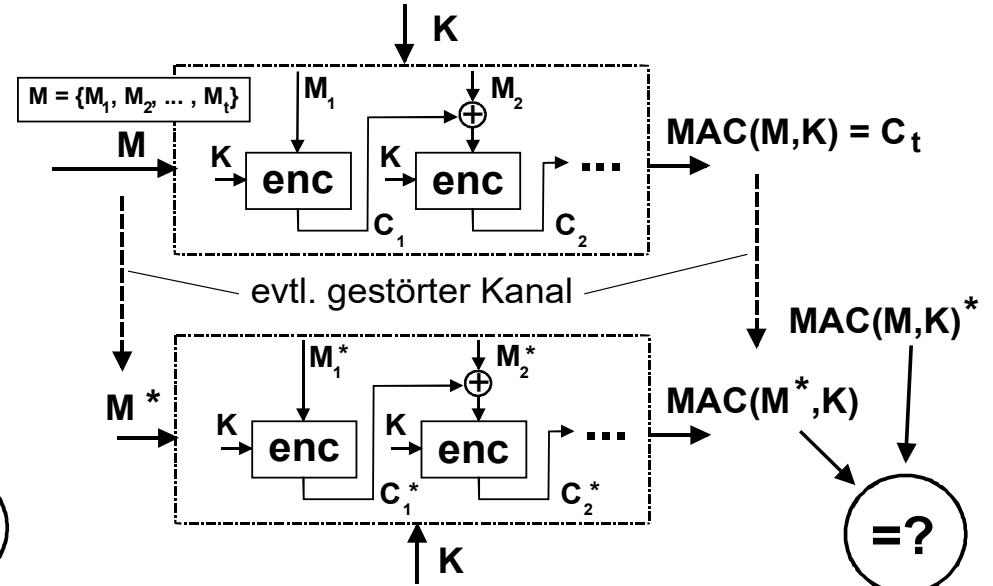
# Basismechanismen im Überblick

MD und MAC

## Message Digest (MD)

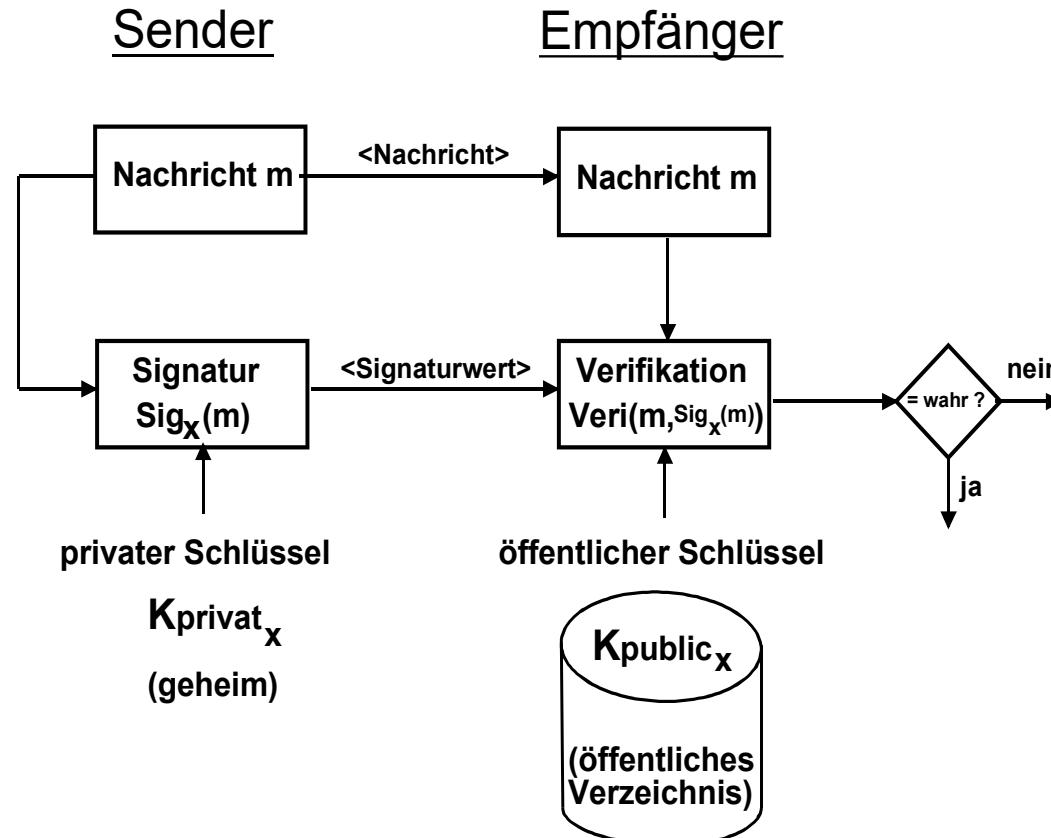


## Message Authentication Code (MAC)



# Basismechanismen im Überblick

## Digitale Signaturen



### Basisprinzip:

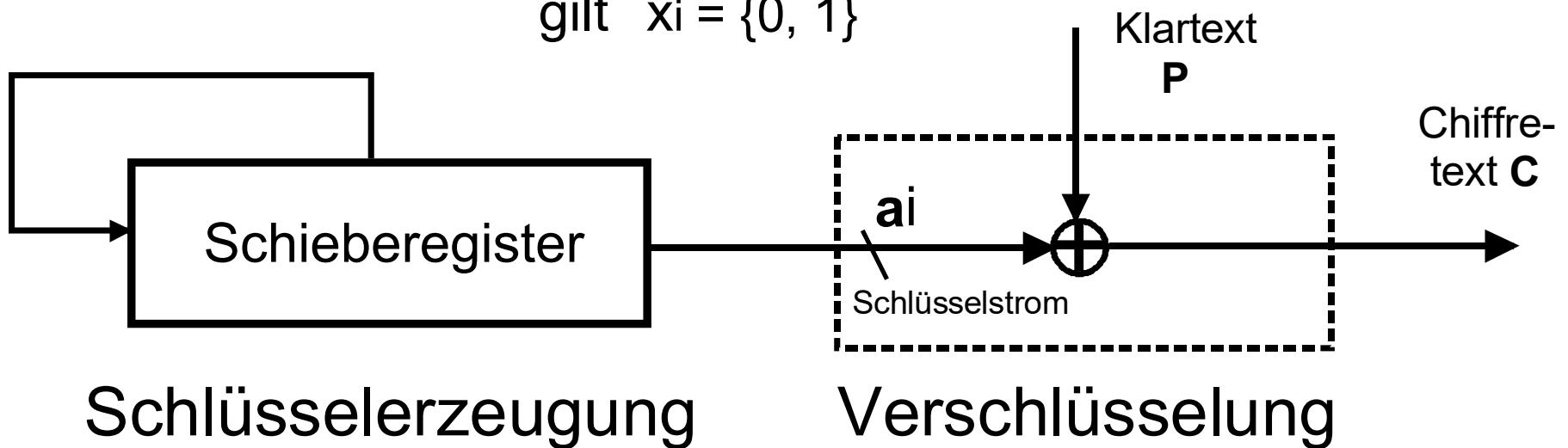
- Privater (geheimer) und öffentlicher Schlüssel
- Signaturwert mit privatem Schlüssel

### Eigenschaften:

- Nachweisbarkeit
- Nicht Abstreitbarkeit
- Authentizität
- Echtheit
- Identitätsnachweis

### Lineare rückgekoppelte Schieberegister

Für  $X = A, P, C$   
gilt  $x_i \in \{0, 1\}$



## Kap. 1: Einführung in die Informationssicherheit

### Teil 5: Kryptanalyse

- Das Prinzip von Kerckhoffs
- Typen von Attacken
- Angriffsstrategien und Analyseverfahren
- Klassifizierung der Sicherheit von Kryptosystemen
- Steganographie

**Alexandre Auguste Kerckhoffs von Nieuwenhof (niederl. Philologe, 1835 – 1903)**

- Klassische Kryptographie ist geprägt vom Wechselspiel zwischen Kryptographie und Krypanalyse (Erkenntnisse → Entwicklungen).
- Die Sicherheit eines Kryptosystems darf nicht von dessen Geheimhaltung, sondern nur von der Schlüssellänge abhängen.

Seien  $P$ ,  $C$ ,  $K$  die Mengen der Plaintexte, Chiffretexte bzw. Schlüssel und  $E : P \times K \rightarrow C$  ein Verschlüsselungssystem. Ist ein Kryptoanalytiker im Besitz eines Plaintext-Chiffertextpaars  $(p, c) \in P \times C$ , so kann der verwendete Schlüssel  $k$  durch **vollständige Suche** ermittelt werden, da  $E(p, k) = c$  gelten muss.

---

- **Ciphertext-only-Attack:**

Es besteht lediglich die Möglichkeit, für die Analyse verschlüsselte Daten (ciphertext) in beliebigem Umfang zu verwenden.

- **Known-Plaintext-Attack:**

Es stehen Klartext-Schlüsseltextrpaare zur Verfügung, wobei bei der Analyse ausgenutzt wird, dass bestimmte Textphrasen häufig verwendet werden.

- **Chosen-Plaintext-Attack:**

Hier verwendet der Kryptoanalytiker beim Angriff die Chiffre zu selbstgewählten Klartexten.

- **Vollständiges Suchen:**

Die gesamte Schlüsselmenge wird durchsucht, um den jeweils verwendeten Schlüssel zu finden (ohne praktische Bedeutung).

- **Trial and Error:**

Im Gegensatz zur vollständigen Suche wird vorausgesetzt, dass eine Strukturanalyse dazu geführt hat, die Schlüsselwahl einzuschränken.

- **Statistische Methoden:**

Hierbei werden statistische Eigenschaften (Verteilungen) verwendet, um Rückschlüsse auf den zugehörigen Klartext zu ermitteln.

- **Strukturanalyse:**

Ausgenutzt werden spezielle Strukturen mit dem Ziel, effiziente Algorithmen zum Brechen des Kryptoverfahrens zu entwerfen.

---

Ein Kryptosystem heißt

- **absolut sicher**,  
wenn nicht genug Information gewonnen werden kann, um hieraus den Klartext oder den Schlüssel zu rekonstruieren.
  - **analytisch sicher**,  
wenn es kein nichttriviales Verfahren gibt, mit dem es systematisch gebrochen werden kann.
  - **komplexitätstheoretisch sicher**,  
wenn es keinen Algorithmus gibt, der das Kryptosystem in Polynomialzeit in Abhängigkeit der Schlüssellänge brechen kann.
  - **praktisch sicher (→ starke Verfahren)**,  
wenn kein Verfahren bekannt ist, welches das Kryptosystem mit vertretbarem Ressourcen-, Kosten- und Zeitaufwand brechen kann.
-

Für die Beurteilung der benötigten Schlüssellänge sind folgende Definitionen sehr hilfreich:

1. Ein **Algorithmus** gilt als **sicher**, wenn

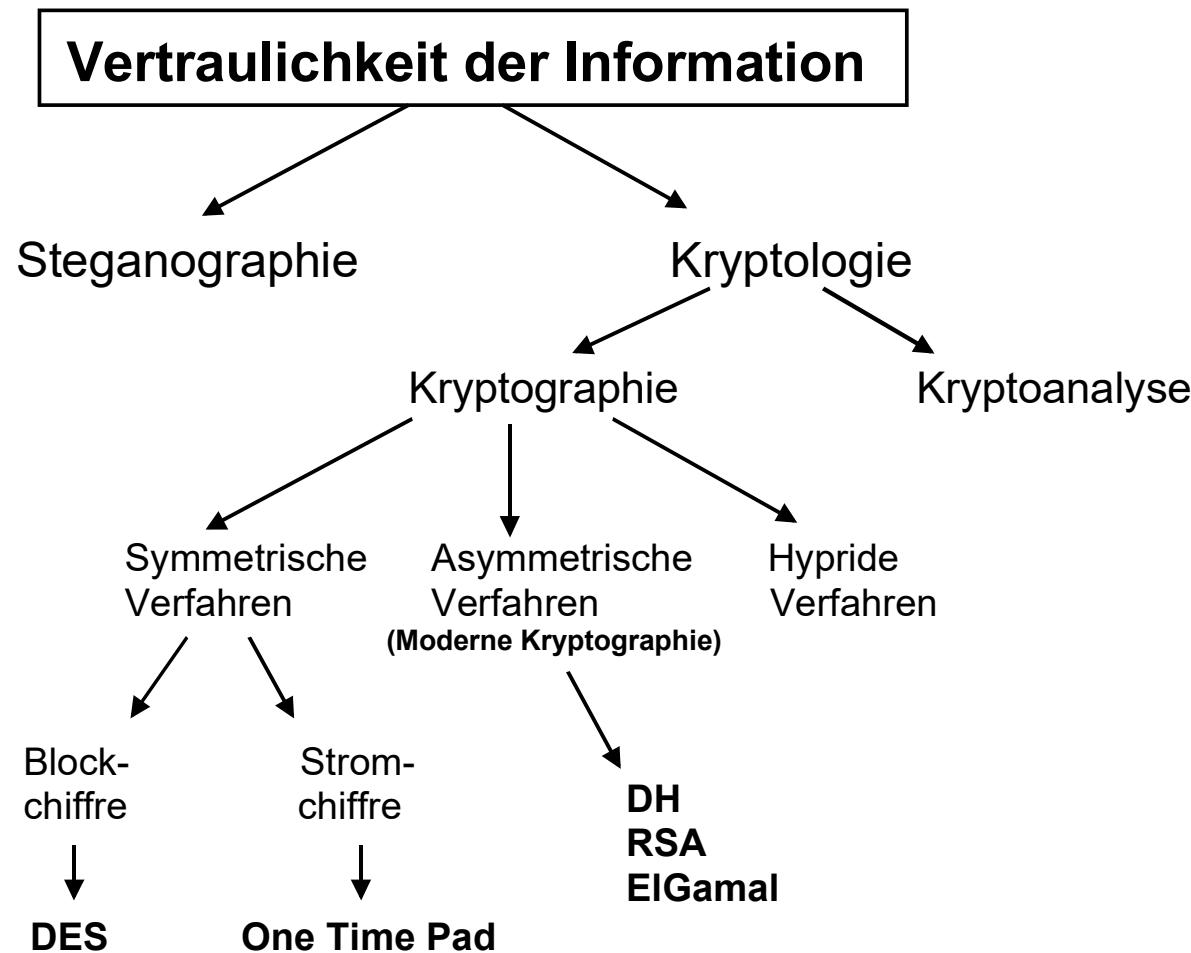
- der zum Aufbrechen nötige Geldaufwand den Wert der verschlüsselten Daten übersteigt oder
- die zum Knacken erforderliche Zeit größer ist als die Zeit, die die Daten geheim bleiben müssen, oder
- das mit einem bestimmten Schlüssel chiffrierte Datenvolumen kleiner ist als die zum Knacken erforderliche Datenmenge.

2. Ein **Algorithmus** gilt als **uneingeschränkt sicher**, wenn der Klartext auch dann nicht ermittelt werden kann, wenn Chiffertext in beliebigem Umfang vorhanden ist ⇒ **starke Kryptographie**.

---

# Wahrung der Vertraulichkeit

Verfahren



D	V	A	B	S	Z
I	H	E	E	S	E
Y	T	E	H	O	T
E	I	Y	T	S	N
I	G	A	E	H	Y
D	O	Y	U	E	I
M	A	N	B	B	L
O	T	I	O	D	S

# Steganographie

---

Klartext

D		A		S	
I				S	
					T
E	I				N
	G		E	H	
				E	I
M		N			
		I			S

## Beispiel für eine verdeckte Botschaft

Was verbirgt sich hinter der folgenden Kleinanzeige?

- Räumung
- Seniorenumzug
- Ankauf

### KLEINTRANSPORTE

Eriko Yamashita

intelligent - sauber - tadellos

Tel: 0126-114719

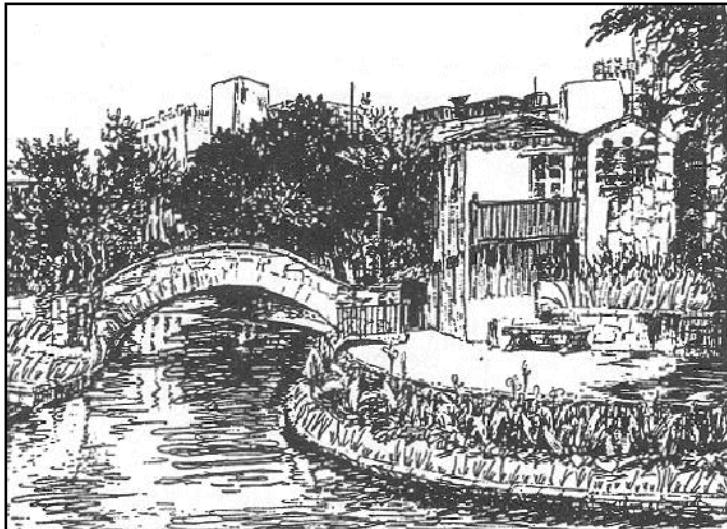


**Hinweis:** Man beachte Anfangsbuchstaben und Tel.-Nr.

---

## Beispiel für eine verdeckte Botschaft

Wo verbirgt sich die Nachricht? Wie lautet diese?



**Hinweis:** Man beachte den Verlauf des San Antonio Rivers (1945)

- **Semagramme:** Nachrichten, die in Details von Skizzen oder Gegenständen verborgen sind.
- Die Botschaft wurde unter Anwendung des Morsealphabets (kurz, lang und Pause/ Leerraum) codiert.
- Zeitschema „ein/aus“ optisch aus der Länge der Grashalmen links von der Brücke auf der kleinen Mauer und rechts entlang des Flusses.

David Kahn: The Codebreakers, Macmillan, 1996, S. 155 ff.

### **Kap. 1: Einführung in die Informationssicherheit**

#### **Zusammenfassung:**

- Aufgrund der gegenwärtigen Gefährdungslage im IT-Bereich sind IT-Sicherheitsmaßnahmen (Funktionen) unerlässlich.
- Die Realisierung von IT-Sicherheitsmaßnahmen und -funktionen erfolgt mit den Mitteln der Kryptologie.
- Wir unterscheiden in klassische und moderne Kryptologie (sog. Public Key Cryptographie).
- Besondere Bedeutung in der Praxis hat nach wie vor die Informationsverschlüsselung (Geheimhaltung).

## Kap. 1: Einführung in die Informationssicherheit

### Zusammenfassung (Fortsetzung):

- Hashwerte und Message Authentication Codes dienen zum Nachweis der Authentizität der Daten, besitzen aber keine Beweiskraft gegenüber Dritten.
  - Eine digitale Signatur wird mittels des geheimen Schlüssels des Urhebers gebildet.
  - Die Überprüfung der Korrektheit einer Signatur findet mittels des zugehörigen öffentlichen Schlüssels statt.
  - Die Urheberschaft kann gegenüber Dritten bewiesen werden.
-

---

# **Security**

**- LV 4120 und 7240 -**

**Algebraische Strukturen und elementare Zahlentheorie**

- Terminologie und Grundsätze der elementaren Zahlentheorie
  - Herausstellung einer algebraischen Struktur
  - Rechenregeln der modularen Arithmetik
  - Zahlentheoretische Funktionen und Algorithmen
  - Primzahlen und Primfaktorzerlegung
  - Vertraulichkeitsschutz mittels einer Stromchiffre
-

### Kap. 4: Algebraische Strukturen und elementare Zahlentheorie

#### Teil 1: Algebraische Strukturen und modulare Arithmetik

- Nomenklatur
  - Algebraische Strukturen
    - Monoid
    - Gruppe
    - Ringe und Körper
  - Modulare Arithmetik
    - Ganzzahlige Division (div und mod)
    - Teilbarkeit
    - Kongruenzen
-

**N** Menge der nichtnegativen ganzen Zahlen

$$\mathbf{N} := \{0, 1, 2, 3, \dots\}$$

**N+1** Menge der natürlichen (positiven ganzen) Zahlen

$$\mathbf{N+1} := \{1, 2, 3, \dots\} = \mathbf{N} \setminus \{0\}$$

**Z** Menge der ganzen Zahlen

$$\mathbf{Z} := \{0, \pm 1, \pm 2, \pm 3, \dots\}$$

**P** Menge der Primzahlen

$$\mathbf{P} := \{2, 3, 5, 7, 11, 13, \dots\}$$

**Q** Menge der rationalen Zahlen (abb. oder period. nicht abb. Dezimalbrüche)

$$\mathbf{Q} := \{a / b \mid a, b \in \mathbf{Z}, b \neq 0, \text{ggT}(a, b) = 1 \text{ (teilerfremd)}\}$$

---

**I** Menge der irrationalen Zahlen (nicht-period. nicht abb. Dezimalbrüche)

$$\mathbf{I} := \{\sqrt[n]{p} \mid p \in \mathbf{P}; n = 2, 3, 4, \dots; \sin(\pi/4); e; \pi; \dots\}$$

**R** Menge der reellen Zahlen

$$\mathbf{R} := \mathbf{Q} \cup \mathbf{I}$$

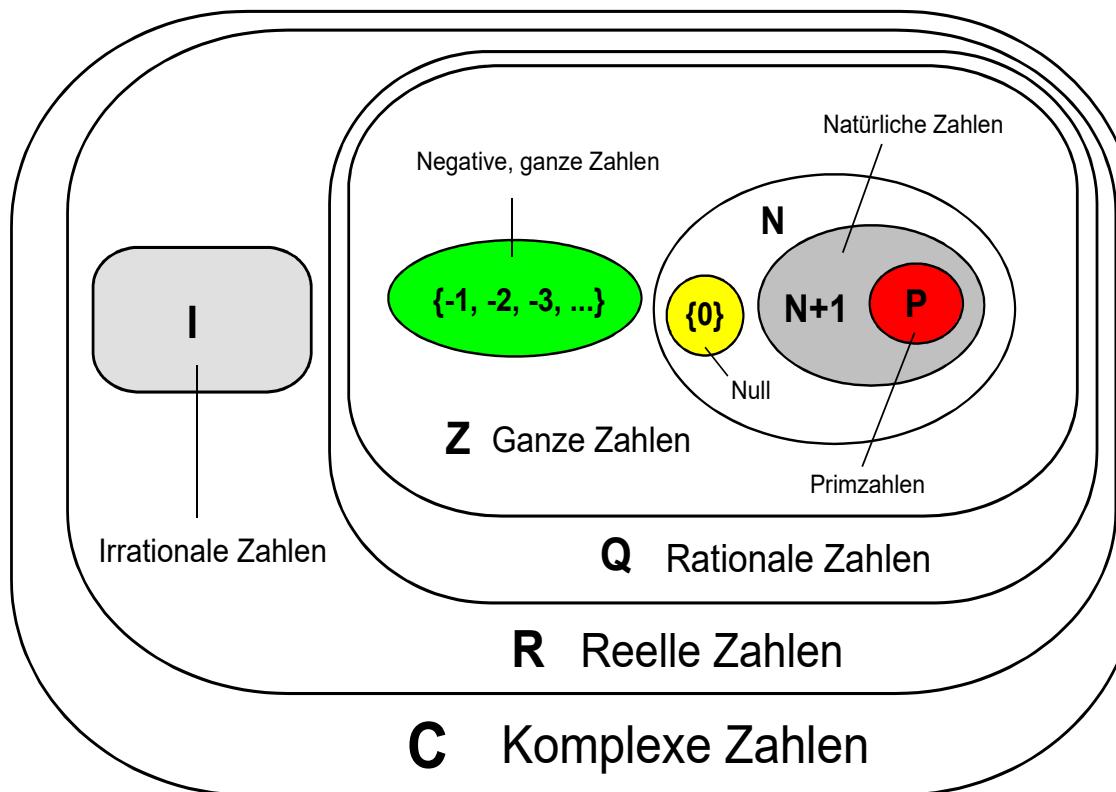
**C** Menge der komplexen Zahlen

$$\mathbf{C} := \{a + jb \mid a, b \in \mathbf{R}; j^2 = -1\}$$

**Z<sub>m</sub>** Restklassenring modulo m

$$\mathbf{Z}_m := \{0, 1, 2, 3, \dots, m-1\}$$

Es gelten die Zusammenhänge:  $\mathbf{N} \subset \mathbf{Z} \subset \mathbf{Q} \subset \mathbf{R} \subset \mathbf{C}$  und  $\mathbf{R} = \mathbf{Q} \cup \mathbf{I}$



$$[k:m] := \begin{cases} 0, & \text{falls } k > m \\ \{z \mid z \in \mathbf{Z}, k \leq z \leq m\}, & \text{sonst} \end{cases} \quad k, m \in \mathbf{Z}$$

$$\begin{aligned} n \mid k &\quad n \text{ teilt } k \\ &\Rightarrow (\exists x \in \mathbf{Z}) \ k = x \cdot n \end{aligned}$$

$$\begin{aligned} n \nmid k &\quad n \text{ teilt } k \text{ nicht} \\ &\Rightarrow (\forall x \in \mathbf{Z}) \ k \neq x \cdot n \end{aligned}$$

**ggT(n, k)** der größte gemeinsame Teiler von n und k

**kgV(n, k)** das kleinste gemeinsame Vielfache von n und k

**n mod d** Divisionsrest, wenn man n durch d teilt

**$\phi(m)$**  EULERsche  $\phi$ -Funktion  
(gibt die Anzahl derjenigen natürlichen Zahlen  $n < m$  an, die teilerfremd zu m sind;  $m, n \in \mathbb{N}$ )

a und b teilerfremd heißt:  $\text{ggT}(a, b) = 1$

### Anmerkungen zum Sprachgebrauch:

- Eine **Algebra** (Plural: Algebren) definiert Elemente und Operatoren sowie Regeln bzw. Eigenschaften (**Axiome**) in bezug auf deren Verknüpfung (abgeschlossene Arithmetik).
- Eigenschaften sind beispielsweise die Existenz eines **neutralen Elementes e** (0 oder 1) oder die Existenz von **inversen Elementen  $a'$**  ( $-a$  bzw.  $a^{-1}$ ).
- Regeln betreffen die **Assoziativität** und die **Kommutativität** bzgl. eines Operators oder die **Distributivität** bzgl. zweier Operatoren.
- Wir benutzen im folgenden die Operatoren  $+$  (**Addition**) und  $\cdot$  (**Multiplikation**) sowie die Quantoren  $\exists$  (**Existenzquantor**) und  $\forall$  (**Allquantor**).

Beispiele:     $\exists x \in B : p(x)$  ist wahr!       $\forall x \in B : p(x)$  ist wahr!

---

$\langle \mathbf{M}, \circ \rangle :=$  algebraisches System (hier: **Monoid**)

$\mathbf{M} =$  Nichtleere Menge  $\{a, b, c, \dots\}$  der Operanden

$\circ =$  Operator (gebildet auf Elementpaare aus  $\mathbf{M}$ )  
hier:  $\circ = \{+, \cdot\}$

mit

$$1) \quad a, b \in \mathbf{M} \Rightarrow a \circ b \in \mathbf{M}$$

$$2) \quad a, b, c \in \mathbf{M} \Rightarrow (a \circ b) \circ c = a \circ (b \circ c) \quad \text{Assoziativ-Gesetz}$$

$$3) \quad a \in \mathbf{M} \Rightarrow \exists e \text{ mit } a \circ e = e \circ a = a$$

d. h. es existiert ein **neutrales Element e** (0 bzw. 1) in bezug auf die Operatoren + und  $\cdot$ .

---

$\langle G, \circ \rangle :=$  algebraisches System (hier: **Gruppe**)

**G** = Nichtleere Menge  $\{a, b, c, \dots\}$  der Operanden

$\circ$  = Operator (gebildet auf Elementpaare aus **G**)  
hier:  $\circ = \{+, \cdot\}$

wenn

1)  $\langle G, \circ \rangle$  ist ein **Monoid**, d. h.  $\exists e = 0$  bzw. 1

2)  $a, b \in G \Rightarrow a \circ b = b \circ a$  **Kommutativ-Gesetz**

3)  $a \in G \Rightarrow \exists a' \text{ mit } a \circ a' = e$  **(vgl. Abelsche Gruppe)**

d. h. es existiert ein **inverses Element**  $a' = -a$  bezüglich der Addition und ein **inverses Element**  $a' = a^{-1}$  bezüglich der Multiplikation.

---

$\langle \mathbf{R}, +, \cdot \rangle :=$  algebraisches System (hier: **Ring**)

$\mathbf{R} =$  Nichtleere Menge  $\{a, b, c, \dots\}$  der Operanden

$+ , \cdot =$  Operatoren (gebildet auf Elementpaare aus  $\mathbf{R}$ )

wenn

1)  $\langle \mathbf{R}, + \rangle$  ist eine **kommutative Gruppe**, d. h.  $\exists e = 0$  &  $\exists a' = -a$

2)  $a, b, c \in \mathbf{R} \Rightarrow a \cdot (b + c) = (a \cdot b) + (a \cdot c)$  **Distributiv-Gesetz**

3)  $\langle \mathbf{R}, \cdot \rangle$  ist ein **Monoid**, d. h. für  $a \in \mathbf{R}$  :  $\exists e = 1$  aber  $\nexists a' = a^{-1}$

d. h. es existiert zwar ein **inverses Element  $a' = -a$**  bezüglich der Addition aber keine multiplikative Inverse.

---

$\langle \mathbf{K}, +, \cdot \rangle :=$  algebraisches System (hier: **Körper**)

$\mathbf{K} =$  Nichtleere Menge  $\{a, b, c, \dots\}$  der Operanden

$+ , \cdot =$  Operatoren (gebildet auf Elementpaare aus  $\mathbf{K}$ )

... ist ein **kommutativer Ring** mit Einselement, in dem jedes von Null verschiedene Element **invertierbar** ist, d. h. für

$$a \in \mathbf{K} \text{ mit } a \neq 0 \quad \Rightarrow \quad \exists a^{-1} \in \mathbf{K} \text{ mit } a \cdot a^{-1} = 1$$

**Satz:**

Jeder Körper  $\langle \mathbf{K}, +, \cdot \rangle$  und jeder Ring  $\langle \mathbf{R}, +, \cdot \rangle$  ist **nullteilerfrei** (Integritätsbereich), d. h. für  $\forall x, y \in \mathbf{K} \setminus \{0\}$  bzw.  $\forall x, y \in \mathbf{R} \setminus \{0\}$  gilt:

$$x \cdot y \neq 0$$

### Satz:

$\mathbf{Z}_m := \{0, 1, \dots, m - 1\}$  ist **genau dann** ein Körper, wenn **m** eine Primzahl ist, d. h.:  $\mathbf{Z}_p$  mit  $p \in \mathbb{P}$  ist ein Körper. (Beweis siehe Übung!)

### Lemma:

Jeder Körper ist ein Ring. (Beweis siehe Übung!)

### Definition:

Eine Algebra  $\langle \mathbf{HG}, \circ \rangle$  heißt **Halbgruppe**, wenn sie in Bezug auf die Operation  $\circ$  dem Assoziativgesetz genügt.

Sie wird **kommutative Halbgruppe** genannt, wenn die Operation  $\circ$  zusätzlich kommutativ ist.

---

# Zahlentheorie

## Algebren-Übersicht

Struktur		Bez.	Formel (Axiom)		A		
Algebra ( $A, +, \cdot$ )	Algebra ( $A, +$ )				N+1 Z Q		
Körper	Ring	abelsche Gruppe	additive Gruppe	HG	assoz.	$a + (b + c) = (a + b) + c$	x x x
					$\exists 0$	$0 + a = a$	- x x
					$\exists -a$	$a + (-a) = 0$	- x x
					komm.	$a + b = b + a$	x x x
					distri.	$a(b + c) = ab + ac$	x x x
Einselem.	Algebra ( $A_0, \cdot$ )	abelsche Gruppe	multipl. Gruppe	HG	assoz.	$a(bc) = (ab)c$	x x x
					$\exists 1$	$1a = a$	x x x
					$\exists a^{-1}$	$a a^{-1} = 1$	- - x
					komm.	$a b = b a$	x x x

Restklassenring ist ein algebraisches System mit einer nichtleeren Menge von Elementen und **zwei** Operatoren ( $\oplus$ ,  $\otimes$ ), die auf die Elemente angewendet werden.

### Anmerkung:

Wir sprechen oft nur von einem Ring  $\mathbf{Z}_m := \{0, 1, 2, 3, \dots, m - 1\}$ , meinen aber den Restklassenring  $\langle \mathbf{Z}_m, \oplus, \otimes \rangle$ , d. h. einen kommutativen Ring – auch „Ring von Integers modulo  $m$ “ genannt. Es ist heute bei weitem das wichtigste algebraische System der modernen Kryptographie.

Beispiel:  $\mathbf{Z}_2 := \{0, 1\}$ ; es gilt das Assoziativ-, Kommutativ- und Distributiv-Gesetz!

$$\begin{array}{ll} 0 \oplus 1 = 1 & 0 \otimes 1 = 0 \\ 1 \oplus 0 = 1 & 1 \otimes 0 = 0 \\ 0 \oplus 0 = 0 & 0 \otimes 0 = 0 \\ 1 \oplus 1 = 0 & 1 \otimes 1 = 1 \end{array}$$

---

## Ganzzahlige Division (Division mit Rest)

- Beim Dividieren einer natürlichen Zahl  $a$  durch eine natürliche Zahl  $b$  bleibt ein Rest  $r \in \{0, 1, \dots, b - 1\}$ .

Beispiel:     $17 : 3 = 5$    Rest **2**   weil    $17 = 5 \cdot 3 + 2$

$12 : 3 = 4$    Rest **0**   weil    $12 = 4 \cdot 3 + 0$

- Der Divisionsrest ist immer eindeutig und es gilt:

Satz:

Seien  $a, b \in \mathbf{N}$  und  $b > 0$ . Dann gibt es eindeutige natürliche Zahlen  $q$  und  $r$  mit

$$a = q \cdot b + r \text{ sowie } 0 \leq r < b.$$

Man nennt  $q$  den Quotient,  $b$  den Divisor und  $r$  den Divisionsrest.

---

## *Mathematica*

- In dem Computeralgebrasystem Mathematica gibt es die Funktionen  $\text{Mod}[a, b]$  und  $\text{Quotient}[a, b]$  mit der Eigenschaft

$$a = \text{Quotient}[a, b] \cdot b + \text{Mod}[a, b].$$

- $\text{Quotient}[a, b] \in \mathbf{Z}$  ist der ganzzahlige Anteil von  $a / b$ .  $a$  und  $b$  dürfen beliebige reelle Zahlen sein mit  $b \neq 0$ .
- Zahlenbeispiel:

$$\text{Quotient}[12345678, 3456] = 3572$$

$$\text{Mod}[12345678, 3456] = 846$$

Definition: Seien  $a, b \in \mathbf{Z}$  und sei  $a = q \cdot b + r$  mit  $0 \leq r < b$ .

Dann schreibt man  $r = a \bmod b$ .

Zwei Zahlen  $a, b \in \mathbf{Z}$  heißen **restgleich**, wenn  $a \bmod n = b \bmod n$ .

Man schreibt:  $a \equiv b \bmod n$

und sagt: **a ist kongruent zu b modulo n.**

Beispiele:

$$19 \equiv 12 \bmod 7 = 5$$

2, 5, 8, 11, ... sind paarweise kongruent modulo 3

Definition: Sei  $n \in \mathbf{N}$ . Dann ist:  $\mathbf{Z}_n := \mathbf{Z} / n\mathbf{Z} := \{0, 1, \dots, n - 1\}$

Man spricht bei  $\mathbf{Z}_n$  von einem **kommutativen Ring**.

---

### Teilbarkeit

Es seien  $n, d \in \mathbf{Z}$ ;  $d \neq 0$ .

- $n$  heißt durch  $d$  teilbar  $\Leftrightarrow (\exists q \in \mathbf{Z}) n = q \cdot d$   
Schreibweise:  $d | n$  (in Worten:  $d$  teilt  $n$ )
- Ist  $n$  nicht durch  $d$  teilbar, so schreiben wir  $d \nmid n$ .  
 $d = 0$  ist als Teiler nie zugelassen.

Satz:

$$\begin{aligned} 0 | a &\Leftrightarrow a = 0 \\ a | b \text{ und } b | a &\Rightarrow a = \pm b \\ (t | a \wedge t | b) &\Rightarrow (\forall x, y \in \mathbf{Z}) t | (a \cdot x + b \cdot y) \end{aligned}$$

---

### Euklidische Divisions-Theorem

Es seien  $n, d \in \mathbf{Z}$ ;  $d \neq 0$ .

$$\Rightarrow (\exists q, r \in \mathbf{Z}) \quad n = q \cdot d + r ; \quad 0 \leq r < |d|$$

q und r sind dabei eindeutig bestimmt.

Man nennt:  $d$  = Divisor;  $n$  = Divident;  $q$  = Quotient;  $r$  = Divisionsrest.

Notation:  $r = R_d(n) = n \bmod d$

Sätze:  $R_{-d}(n) = R_d(n)$

$$R_d(n + i \cdot d) = R_d(n) \quad \text{Fundamental Property of Remainders!} \\ (i \in \mathbf{Z})$$

---

### Größte gemeinsame Teiler

Definition: Der größte gemeinsame Teiler der Integerzahlen  $n_1$  und  $n_2$  (nicht beide gleich Null) ist der größte Integerwert  $t$ , der sowohl  $n_1$  als auch  $n_2$  teilt.

$t$  heißt größter gemeinsamer Teiler von  $n_1$  und  $n_2$ .  
 $\Rightarrow (\exists t = \max\{\tau\}) \mid (\tau \mid n_1 \wedge \tau \mid n_2)$

Notation:  $t := \text{ggT}(n_1, n_2)$

Sätze:  $1 \leq \text{ggT}(n_1, n_2) \leq \min\{|n_1|, |n_2|\}$   
 $\text{ggT}(n_1, 0) = |n_1|$

---

Weitere Sätze:

Ist  $\text{ggT}(a, b) = 1$ , so werden a und b auch teilerfremd oder relativ prim bezeichnet.

$$\text{ggT}(n_1, n_2) = \text{ggT}(n_1 + i \cdot n_2, n_2)$$

**Fundamental Property of Greatest Common Divisors!**

$$\text{ggT}(n_1, n_2) = \text{ggT}(n_2, R_{n_2}(n_1))$$

**Euklid's gcd Recursion!**

Trickreich und nützlich sind noch folgende Sätze, bei denen  $a, b, c \in \mathbb{Z}$  gilt:

- $a | c$  und  $b | c \Rightarrow a \cdot b | c \cdot \text{ggT}(a, b)$
- $a | b \Leftrightarrow \text{ggT}(a, b) = | a |$
- $\text{ggT}(a \cdot c, b \cdot c) = | c | \cdot \text{ggT}(a, b)$
- $a | b \cdot c$  und  $\text{ggT}(a, b) = 1 \Rightarrow a | c$

(Zugehörige Beweise siehe Übungen!)

### Kleinste gemeinsame Vielfache

#### Lösungsverfahren:

Das kleinste gemeinsame Vielfache von zwei ganzen Zahlen a und b können wir vereinfacht so bestimmen:

1. Wähle die größere der beiden Zahlen aus.
2. Bilde das 1fache, 2fache, 3fache, ... dieser Zahl. Prüfe jeweils, ob es auch ein Vielfaches der anderen Zahl ist.
3. Das erste gemeinsame Vielfache ist das sogenannte kleinste gemeinsame Vielfache **kgV** der beiden Zahlen.

Satz:

$$\text{kgV}(a, b) \cdot \text{ggT}(a, b) = a \cdot b$$

---

### Beispiel:

Berechnung von  $\text{kgV}(10, 8)$

1. Wähle 10.
2. Prüfe:

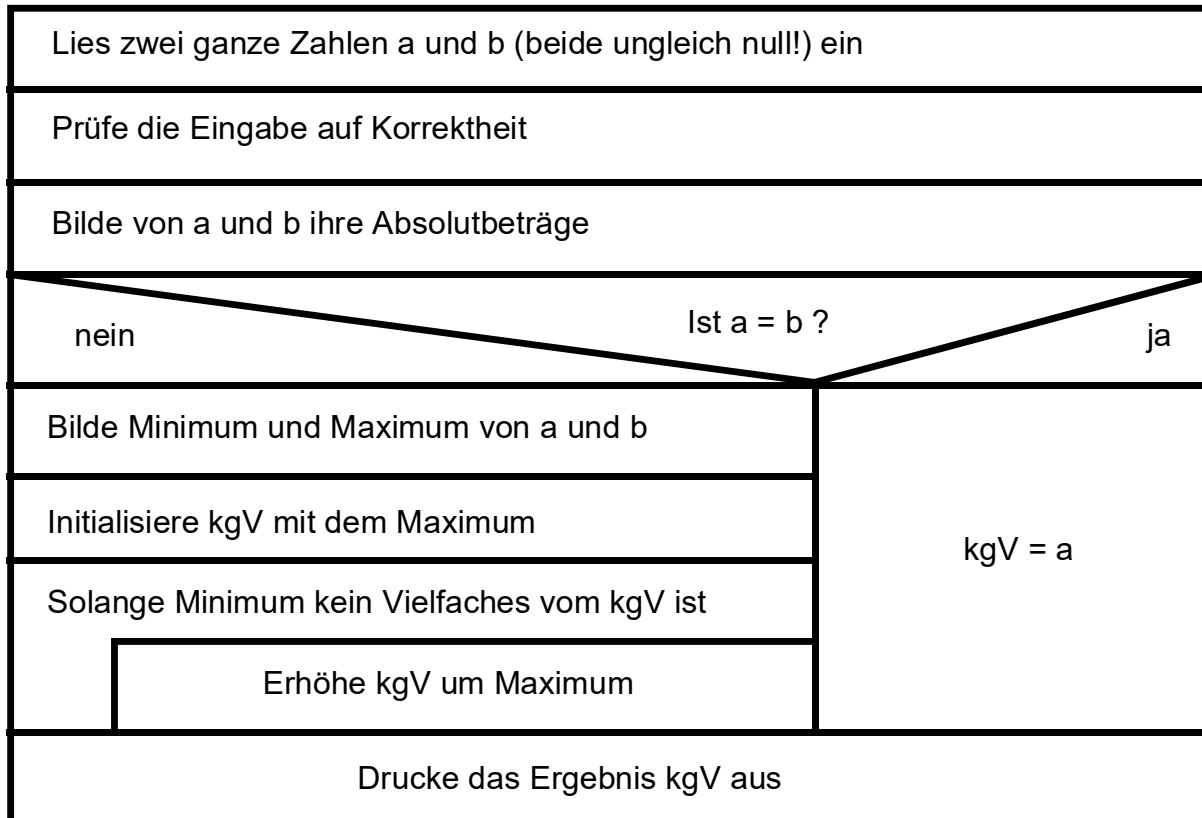
Ist 10 Vielfaches von 8?	→	nein
Ist 20 Vielfaches von 8?	→	nein
Ist 30 Vielfaches von 8?	→	nein
Ist 40 Vielfaches von 8?	→	ja

3. Ergebnis:  **$\text{kgV}(10, 8) = 40$**

Überprüfung: Weil  $\text{ggT}(10, 8) = 2 \Rightarrow \text{kgV}(a, b) \cdot \text{ggT}(a, b) = 80 = a \cdot b$

---

### Struktogramm:



### Kongruenzen

Es seien  $n, r, d \in \mathbf{Z}$ ;  $d \neq 0$ .

Dann heißen  $n$  und  $r$  kongruent modulo  $d \Leftrightarrow d \mid (n - r)$

$$\Leftrightarrow (\exists x \in \mathbf{Z}) n = x \cdot d + r$$

Schreibweise:  $n \equiv r \pmod{d}$

Ist  $n - r$  nicht durch  $d$  teilbar, so heißen  $n$  und  $r$  inkongruent modulo  $d$ .

Schreibweise:  $n \not\equiv r \pmod{d}$

Sätze:  $a \equiv b \pmod{m} \Leftrightarrow b \equiv a \pmod{m} \Leftrightarrow a - b \equiv 0 \pmod{m}$   
 $a \equiv b \pmod{m} \Rightarrow \text{ggT}(a, m) = \text{ggT}(b, m)$

---

### Restsystem

Definition:  $y$  heißt ein Rest von  $x$  modulo  $m \Leftrightarrow x \equiv y \pmod{m}$ .

Eine Menge von Zahlen  $x_1, x_2, \dots, x_m$  heißt vollständiges Restsystem modulo  $m$ , falls für alle  $y \in \mathbf{Z}$  genau ein  $x_j$  ( $j \in [1:m]$ ) mit

$$y \equiv x_j \pmod{m}$$

existiert.

Satz: Es gibt unendlich viele vollständige Restsysteme modulo  $m$ .

$\mathbf{Z}_m := \{0, 1, 2, \dots, m - 1\}$  bilden ein vollständiges Restsystem.

### Sätze von Fermat und Euler

Es sei  $p \in \mathbf{P}$  und  $a \in \mathbf{Z}$ .

$$\text{ggT}(a, p) = 1 \quad \Rightarrow \quad a^{p-1} \equiv 1 \pmod{p}$$

Für  $\forall b \in \mathbf{Z}$  gilt:  $b^p \equiv b \pmod{p}$

$$\phi(p) = p - 1$$

$$\phi(p \cdot q) = (p - 1)(q - 1)$$

für  $p \neq q$  und  $p, q \in \mathbf{P}$

### Eulersche Verallgemeinerung:

Es sei  $a \in \mathbf{Z} \setminus \{0\}$  und  $m \in \mathbf{N} + 1$ .

$$\text{ggT}(a, m) = 1 \quad \Rightarrow \quad a^{\phi(m)} \equiv 1 \pmod{m}$$

---

### Quadratische Reste

Es sei  $a, m \in \mathbf{N} + 1$  mit  $\text{ggT}(a, m) = 1$ .

$a$  heißt quadratischer Rest modulo  $m$ , falls:

$$x^2 \equiv a \pmod{m} \quad \text{lösbar ist.}$$

$a$  heißt quadratischer Nichtrest modulo  $m$ , falls:

$$x^2 \equiv a \pmod{m} \quad \text{nicht lösbar ist.}$$

Satz: Es sei  $p \in \mathbf{P}$ ;  $a \in \mathbf{Z}$  mit  $\text{ggT}(a, p) = 1$  und

$$x^2 \equiv a \pmod{p} \quad (*)$$

Wenn  $a^{(p-1)/2} \equiv 1 \pmod{p}$   $\Leftrightarrow (*)$  hat 2 Lösungen  $x_1$  und  $x_2$ .

$a^{(p-1)/2} \equiv -1 \pmod{p}$   $\Leftrightarrow (*)$  hat keine Lösung.

---

### Der diskrete Logarithmus

Gegeben seien  $g, \alpha \in \mathbf{Z}$  und  $p \in \mathbf{P}$ .

Gesucht ist  $x$ , so daß

$$g^x = \alpha \bmod p$$

- Das Problem ist äußerst schwierig zu lösen, falls  $p$  mindestens 100 Dezimalstellen hat.
- Der Sicherheitswert vieler Algorithmen beruht auf der Schwierigkeit des diskreten Logarithmus.

### Anwendungen des diskreten Logarithmus

- Schlüsselaustausch (Diffie-Hellmann)
- Verschlüsselung (El Gamal, Massey-Omura) und
- Digitale Signatur (El Gamal, DSS)

**Nicht nur diskreter Logarithmus „modulo p“, sondern auch**

- In beliebigen endlichen Körpern  
(mit  $p$  oder  $2^n$  Elementen)
- Auf „elliptischen Kurven“
- Allgemein in „Gruppen“

### Satz 1:

Die „Diskrete Exponentialfunktion“ ist eine Einwegfunktion.

### Satz 2:

Mit den besten bekannten Algorithmen braucht man i. a. exponentiellen Aufwand, um aus  $y$  ein  $x$  mit

$$y = g^x \bmod p$$

zu berechnen.

### Beispiel:

$$g = 2$$

$$p = 37$$

$x$	5	10	20	30
$y$	32	25	33	11

Wir suchen eine Lösung  $x$  der **quadratischen Gleichung**

$$x^2 \equiv a \pmod{p} \quad (1)$$

in der Ringstruktur  $\mathbf{Z}_p$  mit  $p \in \mathbf{P}$ .

Eine solche Lösung bezeichnen wir als **modulare Quadratwurzel** und schreiben sie als:

$$x \equiv \sqrt{a} \pmod{p} \quad (2)$$

Zahlen  $a \in \mathbf{Z}_p$ , welche eine modulare Quadratwurzel besitzen, nennen wir quadratische Reste (sonst quadratische Nichtreste).

Nach heutigem Kenntnisstand ist die Berechnung der modularen Quadratwurzel  $(\text{mod } p)$  ein **recht schwieriges** mathematisches Problem.

---

Satz:

Modulare Quadratwurzeln (mod p) treten immer **paarweise** auf.

Beweis hierzu siehe im folgenden.

Satz:

Damit Gl(1) eine Lösung  $x \in \mathbf{Z}_p$  besitzt, muss gelten:

$$p \equiv 3 \pmod{4} \quad (3)$$

oder

$$a^{\frac{p-1}{2}} \pmod{p} = +1 \quad (4)$$

Ansonsten existiert keine Lösung.

---

### Satz:

Unter der Voraussetzung von  $\text{Gl}(3)$  bzw.  $\text{Gl}(4)$  ergibt sich für  $\text{Gl}(1)$  als eine Quadratwurzel die Lösung:

$$x_1 = a^{\frac{p+1}{4}} \pmod{p} \quad (5)$$

Eine weitere Lösung (zweite Quadratwurzel) von  $\text{Gl}(1)$  ist dann auch gegeben durch:

$$x_2 = p - x_1 \quad (6)$$

Vorstehende Sätze sollen im folgenden bewiesen werden. Dazu benötigen wir lediglich den Satz von Fermat.

---

Beweis für  $x_1$ :

Aus dem Satz von Fermat

$$a^{p-1} \bmod p = 1$$

folgt nach Multiplikation mit  $a^2$ :

$$a^{p+1} \bmod p = a^2 \quad | \quad ()^{1/4}$$

$$\Rightarrow a^{\frac{p+1}{4}} \bmod p = a^{\frac{1}{2}} = \sqrt{a}$$

bzw.

$$x_1 := \sqrt{a} = a^{\frac{p+1}{4}} \bmod p \qquad q.e.d.$$

Beweis für  $x_2$ :

Aus Gl(1) ergibt sich:

$$x \equiv \pm \sqrt{a} \pmod{p}$$

Ist  $\textcolor{brown}{x} = x_1$  eine Lösung von Gl(1), d. h.  $\textcolor{brown}{x}_1 = +\sqrt{a} \pmod{p}$ , dann ist auch  $\textcolor{brown}{x}_2 = -\sqrt{a} \pmod{p} = -x_1$  eine Lösung.

Und somit:

$$x_2 \equiv -x_1 \pmod{p} = (p - x_1) \pmod{p} \quad q.e.d.$$

Beweis für die  $\exists$  von  $x_{1,2}$ :

Eine Lösung nach Gl(5) ist nur **berechenbar**, falls der Exponent  $(p + 1)/4$  eine ganze Zahl ist.

$$\begin{aligned}\Rightarrow & \quad p + 1 \text{ ist ein Vielfaches von } 4 \\ \Rightarrow & \quad 4 \mid (p + 1) \\ \Rightarrow & \quad p + 1 = 4 \cdot k \quad (k = 1, 2, \dots) \\ \Rightarrow & \quad p = 4 \cdot k - 1 \\ \Rightarrow & \quad p = \{3, 7, 11, 19, 23, 31, \dots\} \\ \Rightarrow & \quad p \equiv 3 \pmod{4} \qquad \qquad \qquad q.e.d.\end{aligned}$$

---

Wir suchen eine Lösung  $x$  für die modulare Quadratwurzel

$$x^2 \equiv a \pmod{n} \quad (7)$$

in der Ringstruktur  $\mathbf{Z}_n$  mit  $n = p \cdot q$  und  $p, q \in \mathbf{P}$ .

Satz:

Genügen  $p$  und  $q$  den Bedingungen

$$p \equiv 3 \pmod{4} \quad \text{bzw.} \quad q \equiv 3 \pmod{4},$$

so besitzt die Gl(7) dann insgesamt **vier** Quadratwurzeln der Form  $\pm r$  und  $\pm s$  in der Menge  $\{0, 1, 2, \dots, n-1\}$ , also mod n.

Beweis siehe Übungsaufgaben!

---

## Kap. 2: Algebraische Strukturen und elementare Zahlentheorie

### Teil 2: Zahlentheorie und kryptologische Anwendungen

- Zahlentheoretische Funktionen und Algorithmen
    - Euklidischer und erweiterter Euklidischer Algorithmus
    - Sätze von Fermat und Euler
    - Modulare Inverse und modulare Exponentiation
  - Generierung von Schlüsselparametern
    - Sieb von Eratosthenes
    - Rückgekoppelte Schieberegister
    - Blum-Micali-Generator
  - Bitstromverschlüsselung
-

Definition: Für  $a, b \in \mathbf{N}$  sei  $g = \text{ggT}(a, b)$  der größte gemeinsame Teiler von  $a$  und  $b$ , d. h. die größte ganze Zahl, die  $a$  und  $b$  ohne Rest teilt.

Pseudocode:

```
repeat
    r := a mod b
    if r == 0 then
        g := b
    else
        {   a = b
            b = r   }
    until r == 0
```

Satz: Zwei natürliche Zahlen  $a$  und  $b$  heißen **relativ prim** oder **teilerfremd**, wenn  $\text{ggT}(a, b) = 1$  ist.

---

### Berechnungsbeispiel:

Der ggT von 531 und 93 lässt sich wie folgt durch wiederholte Division berechnen:

$$531 = 5 \cdot 93 + 66$$

$$93 = 1 \cdot 66 + 27$$

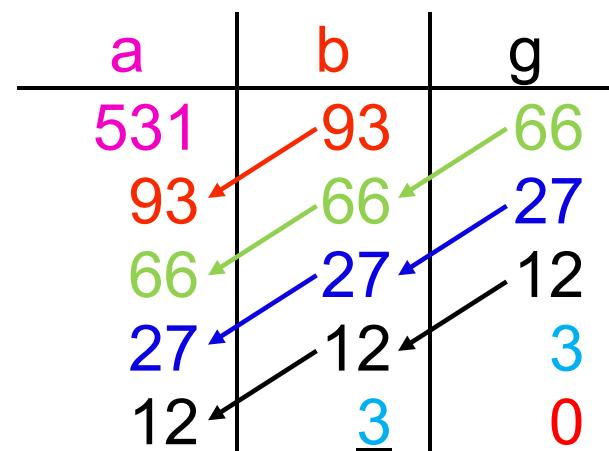
$$66 = 2 \cdot 27 + 12$$

$$27 = 2 \cdot 12 + 3$$

$$12 = 4 \cdot 3$$

$$\Rightarrow \underline{\underline{\text{ggt}(531, 93) = 3}}$$

Kurz:



Satz (Lineare diophantische Gleichung):

Sei  $a, b, d \in \mathbf{Z}$  und  $d = \text{ggT}(a, b) > 0$ . Dann gibt es ganze Zahlen  $x$  und  $y$  mit:

$$d = a \cdot x + b \cdot y,$$

wobei:  $|x| \leq b / (2 \cdot d)$  und  $|y| \leq a / (2 \cdot d)$ .

Pseudocode (Berlekamp-Algorithmus):

```
ErwEuklid(a, b)
  if b == 0 then return (a, 1, 0)
  (d, x, y) = ErwEuklid(b, Mod(a, b))
  return (d, y, x - Div(a, b) * y)
```

Hilfsfunktionen:

$\text{Mod}(a, n)$

**return**  $(a \% n)$

$\text{Div}(a, n)$

**return**  $((a - \text{Mod}(a, n)) / n)$

---

# Zahlentheorie

## Erweiterter Euklidischer Algorithmus

Berechnungsbeispiel:

Gesucht sei die Lösung der Gleichung

$$3 = 531 \cdot x + 93 \cdot y \quad (*)$$

Durch schrittweises Rückwärts-einsetzen ergibt sich aus

$$531 = 5 \cdot 93 + 66 \quad (1)$$

$$93 = 1 \cdot 66 + 27 \quad (2)$$

$$66 = 2 \cdot 27 + 12 \quad (3)$$

$$27 = 2 \cdot 12 + 3 \quad (4)$$

folgende Gleichungskette:

$$3 \stackrel{(4)}{=} 27 - 2 \cdot 12$$

$$\stackrel{(3)}{=} 27 - 2 \cdot (66 - 2 \cdot 27) = -2 \cdot 66 + 5 \cdot 27$$

$$\stackrel{(2)}{=} -2 \cdot 66 + 5 \cdot (93 - 1 \cdot 66) = 5 \cdot 93 - 7 \cdot 66$$

$$\stackrel{(1)}{=} 5 \cdot 93 - 7 \cdot (531 - 5 \cdot 93) = -7 \cdot 531 + 40 \cdot 93$$

→ Zahl 3 darstellbar als Linear-kombination von 531 und 93.

Durch Vergleich mit (\*) erhält man

$$\underline{\underline{x = -7}} \quad \text{und} \quad \underline{\underline{y = 40}}$$

### Modulare Inversion:

Wir nehmen an, dass  $\text{ggT}(a, n) = 1$  ist. Dann gibt es ganze Zahlen  $x$  und  $y$  mit  $1 = a \cdot x + n \cdot y$  und es folgt:

$$x = a^{-1} \bmod n .$$

Mit dem erweiterten Euklidischen Algorithmus (Berlekamp-Algorithmus) haben wir also ein Verfahren zur Berechnung der Inversen  $a^{-1} \bmod n$ .

### Beispiel:

Gesucht ist die Inverse von 11 in  $\mathbf{Z}_{26}$ .

$$\Rightarrow 11^{-1} \bmod 26 = 19 \text{ weil } 19 \cdot 11 \bmod 26 = 1$$

Herleitung:

$$\text{ggT}(a,b) = a \cdot x + b \cdot y \Rightarrow x, y$$

1. Setze:  $b = n$

2. Annahme:  $\text{ggt}(a,n) = 1$

$$\Rightarrow 1 = a \cdot x + n \cdot y$$

3. Bilde: mod n auf beiden Seiten

$$1 \bmod n = a \cdot x \bmod n + n \cdot y \bmod n$$

$\underbrace{\phantom{0}}_{=1}$ 
 $\underbrace{\phantom{0}}_{=0}$

Also  $1 = a \cdot x \bmod n$

$\Leftrightarrow x = a^{-1} \bmod n$

---



---

Berechnungsbeispiel:

gesucht : Inverse von 11 in  $\mathbb{Z}_{26}$  = ?

(falls sie existiert)

a	b	g	$a = 11 \ b = n = 26$
11	26	11	
26	11	4	$26 = 2 \cdot 11 + 4 \quad (1)$
11	4	3	$11 = 2 \cdot 4 + 3 \quad (2)$
4	3	1	$4 = 1 \cdot 3 + 1 \quad (3)$

Da  $\text{gg}(11,26) = 1$  ist 11 invertierbar.

Nun wird schrittweise rückwärts eingesetzt:

$$1 \stackrel{(3)}{=} 4 - 1 \cdot 3$$

$$\stackrel{(2)}{=} 4 - 1 \cdot (11 - 2 \cdot 4) = -11 + 3 \cdot 4$$

$$\stackrel{(1)}{=} -11 + 3 \cdot (26 - 2 \cdot 11)$$

$$= 3 \cdot 26 - 7 \cdot 11$$

$$\underbrace{-} \qquad \underbrace{-}$$

$y \qquad x$

Also ist

$$a^{-1} \bmod n = 11^{-1} \bmod 26$$

$$= -7 \bmod 26$$

$$= 19$$

Probe:  $11 \cdot 19 \bmod 26$   
 $= 209 \bmod 26 = 1$

Definition: Für jede natürliche Zahl  $n$  gibt die Eulersche  $\phi$ -Funktion  $\phi(n)$  die Anzahl der natürlichen Zahlen kleiner als  $n$  an, die zu  $n$  **teilerfremd** sind, d. h.

$$\phi(n) = \left| \{ k \leq n \mid \text{ggT}(k, n) = 1 \} \right|$$

## Beispiel:

$\phi(15) = 8$ , denn die Zahlen 1, 2, 4, 7, 8, 11, 13 und 14 sind teilerfremd zu 15.

## Spezialfälle:

- $p \in P \Rightarrow \phi(p) = p - 1$
  - $k \in N+1 \Rightarrow \phi(p^k) = p^{k-1} \cdot (p - 1)$
  - $p, q \in P \Rightarrow \phi(p \cdot q) = (p - 1) \cdot (q - 1)$   
 $(p \neq q) \qquad \qquad \qquad = \phi(p) + \phi(q)$

Berechnungsbeispiel:

gesucht :  $\Phi(15) = ?$

Berechne :	$ggT(0,15)$	$\neq 1$	# ↓	abzählen :
	$ggT(1,15)$	$= 1$	1	
	$ggT(2,15)$	$= 1$	2	
	$ggT(3,15)$	$\neq 1$		
	$ggT(4,15)$	$= 1$	3	
	$ggT(5,15)$	$\neq 1$		
	$ggT(6,15)$	$\neq 1$		
	$ggT(7,15)$	$= 1$	4	
	$ggT(8,15)$	$= 1$	5	

$ggT(9,15)$	$\neq 1$
$ggT(10,15)$	$\neq 1$
$ggT(11,15)$	$= 1$
$ggT(12,15)$	$\neq 1$
$ggT(13,15)$	$= 1$
$ggT(14,15)$	$= 1$

$$\begin{aligned} & \# ggt = 1 \text{ ist } 8 \\ \Rightarrow & \underline{\underline{\Phi(15) = 8}} \end{aligned}$$

$$\begin{aligned} \text{Spezialfall : } & \Phi(p) = p-1 \\ p \in P & \end{aligned}$$

Satz: Wenn  $\text{ggT}(a, n) = 1$  ist, dann gilt:

$$a^{\phi(n)} \bmod n = 1$$

Ist  $n = p \in \mathbf{P}$  eine Primzahl, so ergibt sich der Fermatsche Satz:

$$a^{p-1} \bmod p = 1 ; (a \neq 0)$$

und damit auch für die modulare Inverse:

$$a^{-1} \bmod p = a^{p-2} \bmod p ; (a \neq 0)$$

Beispiel:  $6^{-1} \bmod 23 = 6^{21} \bmod 23 = \underline{4}$

---

Berechnungsbeispiel:

gesucht :  $6^{-1} = ? \pmod{23}$

Satz von Fermat:

$$a^{p-1} \pmod{p} = 1 ; (a \neq 0 \quad p \in \mathbb{P})$$

$$\underbrace{a \cdot a^{-1} \cdot a^{p-1}}_{1 \quad a^{p-2}} \pmod{p} = 1$$

also

$$a \cdot a^{p-2} \pmod{p} = 1$$

Auf der anderen Seite gilt :

$$a \cdot a^{-1} \pmod{p} = 1$$

also

$$\underline{\underline{a^{-1} \pmod{p} = a^{p-2} \pmod{p}}}$$

d.h.

$$6^{-1} \pmod{23} = \underbrace{6^{21} \pmod{23}}_{=4}$$

$$\underline{\underline{6^{-1} \pmod{23} = 4}}$$

Probe:  $6 \cdot 4 = 24 \equiv 1 \pmod{23}$

q.e.d.

Satz: Wenn  $\text{ggT}(a, n) = 1$  ist, dann gilt:

$$a^b \bmod n = a^{b \bmod \phi(n)} \bmod n$$

Beweis:

Entwickle b als Quotient von  $\phi(n)$ , d. h.

$$b = q \cdot \phi(n) + r \quad \text{mit} \quad r = b \bmod \phi(n) < \phi(n).$$

Also

$$\begin{aligned} a^b \bmod n &= a^{q \cdot \phi(n)} \bmod n \cdot a^r \bmod n \\ &= a^{\phi(n)} \bmod n \cdot a^{\phi(n)} \bmod n \cdot \dots \cdot a^{\phi(n)} \bmod n \cdot a^r \bmod n \\ &\quad \underbrace{\qquad\qquad\qquad}_{q \text{ mal}} \\ &= 1 \cdot 1 \cdot \dots \cdot 1 \cdot a^r \bmod n \\ &= a^r \bmod n = \underline{a^{b \bmod \phi(n)} \bmod n} \end{aligned}$$

q.e.d.

---

### Anwendungsbeispiel:

$$a^{1234} \bmod 31 = a^4 \bmod 31$$

#### Nebenrechnung:

$$\begin{aligned} & 1234 \bmod \Phi(31) \\ &= 1234 \bmod 30 = 4 \end{aligned}$$

Berechnungsbeispiel:

$$a^b \bmod n = a^{b \bmod \Phi(n)} \bmod n$$

$$a = 1234 \quad n = p = 163$$

$$b = 5678$$

$$n = 163$$

$$1234^{5678} \bmod 163 = ?$$

$$\Phi(n) = \Phi(163) = 162$$

$$b \bmod \Phi(n) = 5678 \bmod 162 = 8$$

$$\rightarrow 1234^{5678} \bmod 163 = \\ 1234^8 \bmod 163 = \\ ((1234^2)^2)^2 \bmod 163 = 57$$

$$1234 \bmod 163 = 93$$

$$93^2 \bmod 163 = 10$$

$$10^2 \bmod 163 = 100$$

$$100^2 \bmod 163 = 57$$

Beispiel: Wir betrachten  $(\mathbf{Z}_4, +, \cdot)$  und erstellen die Verknüpfungstabelle für die Multiplikation:

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	<b>0</b>	2
3	0	3	2	1

- $\mathbf{Z}_4$  ist nicht nullteilerfrei und es gibt zu 2 kein multiplikatives Inverses.
  - Die Zahl 3 ist zu sich selbst invers, denn  $3 \cdot 3 = 1$ .
-

Beispiel: Wir betrachten  $(\mathbf{Z}_3, +, \cdot)$  und erstellen ebenfalls die Verknüpfungstabelle für die Multiplikation:

$\cdot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

- Die Zahl 2 ist invers zu sich selbst, denn  $2 \cdot 2 = 1$ , d. h.  $2^{-1} = 2$ .
  - Wegen  $2 \cdot \frac{1}{2} = 1$  folgt in  $\mathbf{Z}_3$  auch:  $\frac{1}{2} = 2$   
(mit  $\frac{1}{2}$  ist nicht die Zahl 0.5 gemeint, die es in  $\mathbf{Z}_3$  nicht gibt!)
-

Zur effizienten Berechnung von  $a^b \bmod n$  wird bei binärer Repräsentation des  $(k+1)$ -Bit-Exponenten  $b = (b_k, \dots, b_1, b_0)$  die Funktion ModExpo( $a, b, n$ ) benutzt.

Pseudocode (Square-and-Multiply-Algorithmus):

```
d := 1
for i := k to 0 do
{   d := (d * d) mod n
    if bi == 1 then
        d := (d * a) mod n
return d
```

Ausgabe:  $d = a^b \bmod n$

**(maximal size(Exponent) Quadrierungen und Multiplikationen!)**

---

Berechnungsbeispiel:

gesucht :  $6^{21} \text{ mod } 23 = ?$

$$\begin{aligned} \Rightarrow a &= 6 \text{ und } b = 21 = 16 + 4 + 1 \\ &= (1 \ 0 \ 1 \ 0 \ 1)_d \\ &\quad // \quad // \quad // \\ &\quad b_4 \quad b_2 \quad b_0 \end{aligned}$$

$\Rightarrow$  8 Multiplikationen  
 $d \cdot d$  oder  $d \cdot a$

```

d = 1
for i = 4 to 0
    d = d · d mod n = 1
    if b4 == 1 then d:= d·a = a (weil b4 = 1)
    d = d · d mod n = a2 mod n
    if b3 == 1 ⇒ nein
    d = d · d mod n = a2 · a2 mod n = a4 mod
n
    if b2 == 1 then d:= d·a = a5 (weil b2 = 1)
    d = d · d mod n = a5 · a5 mod n = a10
    if b1 == 1 ⇒ nein
    d = d · d mod n = a10 · a10 mod n = a20
    if b0 == 1 then d:= d·a = a21 (weil b0 = 1)
```

---

Berechnungsbeispiel:

gesucht :  $6^{21} \text{ mod } 23 = ?$

$$\begin{aligned} \Rightarrow a &= 6 \text{ und } b = 21 = 16 + 4 + 1 \\ &= (1 \ 0 \ 1 \ 0 \ 1)_d \\ &\quad // \quad // \quad // \\ &\quad b_4 \quad b_2 \quad b_0 \end{aligned}$$

$$\Rightarrow \underline{\underline{6^{21} \equiv 4 \text{ mod } 23}}$$

(erzielt nach 8 Multiplikationen)

```

d = 1
for i = 4 to 0
    d = 1 · 1 mod 23 = 1
    if b4 == 1 then d:= 1 · 6 = 6 (weil b4 = 1)
    d = 6 · 6 mod 23 = 62 mod 23 = 13
    if b3 == 1 ⇒ nein
    d = 13 · 13 mod n = 132 mod 23 = 8
    if b2 == 1 then d:= 8 · 6 = 2 (weil b2 = 1)
    d = 2 · 2 mod 23 = 4
    if b1 == 1 ⇒ nein
    d = 4 · 4 mod 23 = 16
    if b0 == 1 then d:= 16 · 6 = 4 (weil b0 = 1)

```

Problemstellung:

$$x \equiv a_i \pmod{m_i} ; i = 1, 2, \dots$$

Beispiel :

$$x \pmod{4} = 2$$

$$x \pmod{3} = 1$$

$$x \pmod{5} = 0$$

gesucht :  $x = ?$

(kleinste natürliche Zahl)

Antwort / Lösung :  $x = 10$

Fragestellungen :

1. Wie findet man  $x$  ?
2. Lässt sich  $x$  eindeutig benennen ?
3. Effiziente Berechnung möglich ?

⇒ Lösungsalgorithmus

### Lösungsmethode:

Seien  $m_1, m_2, \dots, m_n \in \mathbf{N}+1$  und paarweise teilerfremd sowie  $a_1, a_2, \dots, a_n \in \mathbf{Z}$ .

### Simultane Kongruenz:

$$x \equiv a_1 \pmod{m_1}; \quad x \equiv a_2 \pmod{m_2}; \quad \dots; \quad x \equiv a_n \pmod{m_n} \tag{1}$$

Lösung:

1. Setze:  $m = \prod_{i=1}^n m_i$ ;  $\textcolor{red}{M}_i = m / m_i \quad (1 \leq i \leq n)$   
d. h., in  $\textcolor{red}{M}_i$  ist kein  $m_i$  enthalten!

2. Weil die  $m_i$  paarweise teilerfremd sind, gilt:

$$\text{ggT}(m_i, \textcolor{red}{M}_i) = 1 \quad \text{für } 1 \leq i \leq n.$$

3. Nutze den erweiterten Euklidischen Algorithmus, um Zahlen  $y_i \in \mathbf{Z}$  für  $1 \leq i \leq n$  zu berechnen:

$$y_i \cdot M_i \equiv 1 \pmod{m_i} \Leftrightarrow y_i \cdot M_i + n_i \cdot m_i = 1 \quad \text{für } (1 \leq i \leq n) \quad (2)$$

Behauptung: Eindeutige Lösung der simultanen Kongruenz (1) ist:

$$x = \left( \sum_{i=1}^n a_i \cdot y_i \cdot M_i \right) \pmod{m} \quad (3)$$

Beweis: Aus (2) folgt:

$$a_i \cdot y_i \cdot M_i \equiv a_i \pmod{m_i} \quad \text{für } (1 \leq i \leq n) \quad (4)$$

und weil für  $j \neq i$  die Zahl  $m_j$  ein Teiler von  $M_j$  ist, gilt:

$$(a_j \cdot y_j \cdot m_1 \cdot m_2 \cdot \dots \cdot m_i \cdot m_j \cdot m_n / m_j) \pmod{m_i} = 0 \quad \Leftrightarrow$$

$$a_j \cdot y_j \cdot M_j \equiv 0 \pmod{m_i} \quad \text{für } (1 \leq \{i, j\} \leq n \text{ sowie } i \neq j) \quad (5)$$

Aus dem Lösungsansatz gemäß (3) folgt:

$$x = \left( \sum_{i=1}^n a_i \cdot y_i \cdot M_i \right) \pmod{m}$$

$$= (a_i \cdot y_i \cdot M_i) \pmod{m} + \left( \sum_{\substack{j=1 \\ j \neq i}}^n a_j \cdot y_j \cdot M_j \right) \pmod{m},$$

wobei wir die Summe in zwei Terme geschrieben haben!

Nun wenden wir auf der rechten und linken Seite die Operation **mod m<sub>i</sub>** an:

---

$$x \bmod m_i = ((\sum_{i=1}^n a_i \cdot y_i \cdot M_i) \bmod m) \bmod m_i$$

$$= (\sum_{i=1}^n a_i \cdot y_i \cdot M_i) \bmod m_i$$

$$= (a_i \cdot y_i \cdot M_i) \bmod m_i + (\sum_{j=1}^{n-i} a_j \cdot y_j \cdot M_j) \bmod m_i$$

$$\downarrow \qquad \qquad j \neq i \qquad \qquad \downarrow$$

$$= a_i \text{ wegen (4)} \qquad \qquad = 0 \text{ wegen (5)}$$

---

→

$$x \equiv a_i (\bmod m_i) \text{ für } (1 \leq i \leq n) \quad \text{q.e.d.}$$

Also löst  $x$  nach (3) die Kongruenz (1)!

### Rechenbeispiel:

$$x \equiv 2 \bmod 4; \quad x \equiv 1 \bmod 3 \quad \text{und} \quad x \equiv 0 \bmod 5; \quad \text{gesucht: } x = ?$$

$$a_1 = 2 \quad m_1 = 4$$

$$\leftrightarrow \quad a_2 = 1 \quad \text{und} \quad m_2 = 3 \quad \rightarrow \quad m = m_1 \cdot m_2 \cdot m_3 = 60 \quad \rightarrow \\ a_3 = 0 \quad m_3 = 5$$

$$M_1 = 60 / 4 = 15;$$

$$M_2 = 60 / 3 = 20 \quad \text{und}$$

$$M_3 = 60 / 5 = 12.$$

Wir lösen:

- 1)  $y_1 \cdot M_1 \equiv 1 \pmod{m_1} \rightarrow y_1 \cdot 15 \equiv 1 \pmod{4} \rightarrow y_1 = 3$
- 2)  $y_2 \cdot M_2 \equiv 1 \pmod{m_2} \rightarrow y_2 \cdot 20 \equiv 1 \pmod{3} \rightarrow y_2 = 2$
- 3)  $y_3 \cdot M_3 \equiv 1 \pmod{m_3} \rightarrow y_3 \cdot 12 \equiv 1 \pmod{5} \rightarrow y_3 = 3$

Mit (3) erhalten wir die Lösung:

$$\begin{aligned}\underline{\underline{x}} &= \left( \sum_{i=1}^3 a_i \cdot y_i \cdot M_i \right) \pmod{m} \\ &= (2 \cdot 3 \cdot 15 + 1 \cdot 2 \cdot 20 + 0 \cdot 3 \cdot 12) \pmod{60} = \underline{\underline{10}}\end{aligned}$$

---

### Algorithmus:

Pseudocode (Chinesischer Restalgorismus):

```
CRT(int koeff[], int moduli[], int number)
{    int multi[number], result = 0, modulus, i
    CRTPre(moduli[], number, modulus, multi[])
    for i := 1 to number do
        result = (result + koeff[i] * multi[i]) % modulus;
    return result
}
```

Benutzt wird die Hilfsfunktion CRTPre, die mittels ErwEuklid die Werte  $y_i$  und anschließend die Produkte  $y_i \cdot M_i$  berechnet und in der Variable  $multi_i$  übergibt.

---

Pseudocode (Hilfsfunktion):

```
CRTPre(moduli[], number, modulus, multi[])
{
    int i, m, M, x, y, ggT, modulus = 1
    for i := 1 to number do
        modulus = modulus * moduli[i]
    for i := 1 to number do
        {
            m = moduli[i]
            M = modulus / m
            (ggT, x, y) = ErwEuklid(M, m)
            multi[i] = (x * M) % modulus
        }
    return (modulus, multi[])
}
```

Für die Implementierung des erweiterten Euklidischen Algorithmus ErwEuklid (Berlekamp-Algorithmus) siehe Kap. IV Seite 38.

---

Seien  $m_1, m_2, \dots, m_n \in \mathbf{N}+1$  und paarweise teilerfremd sowie  $a_1, a_2, \dots, a_n \in \mathbf{Z}$ .

Behauptung:

Dann hat die simultane Kongruenz gemäß (1)

$$x \equiv a_i \pmod{m_i} \quad (1 \leq i \leq n)$$

eine Lösung gemäß (3), die eindeutig ist mod  $m$  mit:  $m = \prod_{i=1}^n m_i$ .

**Satz:**

Der Algorithmus zur Lösung der simultanen Kongruenz erfordert mit  $k := \text{size}(m)$  folgende Aufwände:

Zeit-Aufwand:  $O(k^2)$

Speicherplatz-Aufwand:  $O(k)$

- Gegenstand der elementaren Zahlentheorie sind die natürlichen Zahlen **N** und deren Erweiterungen **Z** (ganze Zahlen), sowie der Körper **Q** der rationalen Zahlen.
  - Der Gebrauch der Zahlentheorie setzt an manchen Stellen auch grundlegende Kenntnisse der Algebra (Gruppen- und Ringtheorie) sowie der linearen Algebra voraus.
  - Das Material zur Lehrveranstaltung ist so ausführlich wie nötig und so knapp wie möglich zusammengestellt, um hier sämtliche Grundlagen der folgenden Praktikumsaufgaben zu legen und doch das erste Verständnis nicht zu erschweren.
  - In einer Krypto-Library wurden die für unsere Zwecke effizientesten Algorithmen (erweiterter Euklidischer Algorithmus, modulare Inverse, Square & Multiply Algorithmus, Chinesischer Restalgorismus etc.) implementiert.
  - Damit werden die Teilnehmer in die Lage versetzt, praktische Kryptoverfahren nachprogrammieren und den Aufwand von insbesondere asymmetrischen Verschlüsselungs-, Signatur- und Authentifikationssystemen abschätzen zu können.
-

---

# **Security**

## **- LV 4120 und 7240 -**

**Monoalphabetische Chiffren und deren Analyse**

- Terminologie und Grundsätze der Kryptographie
  - Transpositions- und Substitutionsschiffen
  - Verschiebechiffre (Caesar-Verschlüsselung)
  - Multiplikative Chiffre
  - Tauschchiffre (Affine Chiffre)
  - Häufigkeitsanalyse
  - Realisierung
-

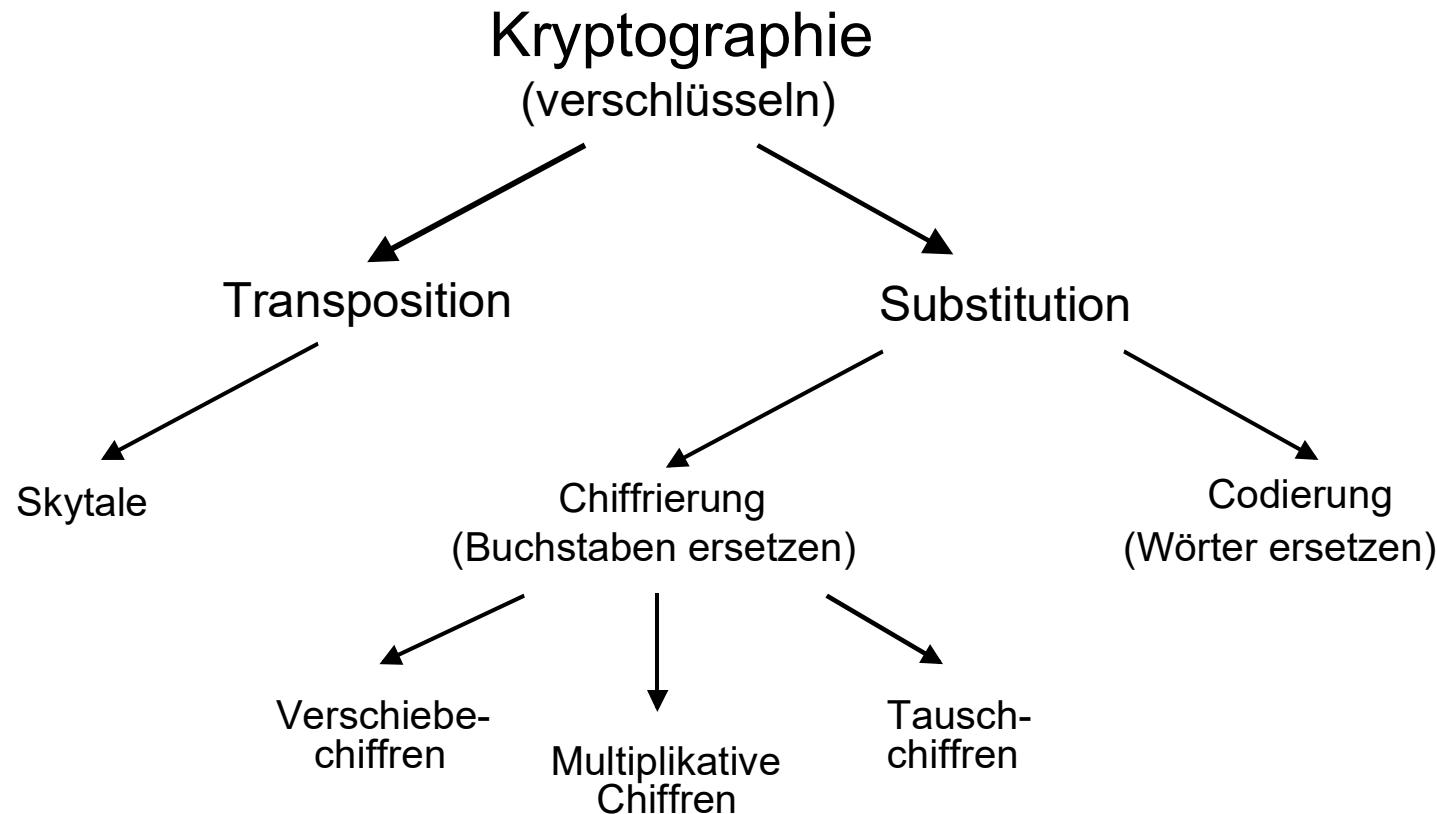
### **Kap. 3: Monoalphabetische Chiffren und deren Analyse**

#### **Teil 1: Einteilung der kryptographischen Chiffrierverfahren**

- Transpositionschiffren
- Substitutionschiffren

# Kryptographische Algorithmen

Einteilung



- Bei einer **Transpositionschiffre** wird der Geheimtext durch eine Permutation der Klartextzeichen erzeugt.

⇒ Die Zeichen bleiben gleich, tauschen aber ihre Plätze.

- Bei einer **Substitutionschiffre** wird jedes Zeichen des Klartextes durch ein anderes ersetzt.

⇒ Die Position bleibt jedoch erhalten.

- Substitutionschiffren sind demnach invertierbare Abbildungen eines endlichen Alphabets A auf ein (evtl. anderes) endliches Alphabet.
- Eine Substitutionschiffre heißt **monoalphabetisch**, wenn jedes Klartextzeichen immer auf das gleiche Geheimtextzeichen abgebildet wird.
- Ansonsten heißt die Substitutionschiffre **polyalphabetisch**.

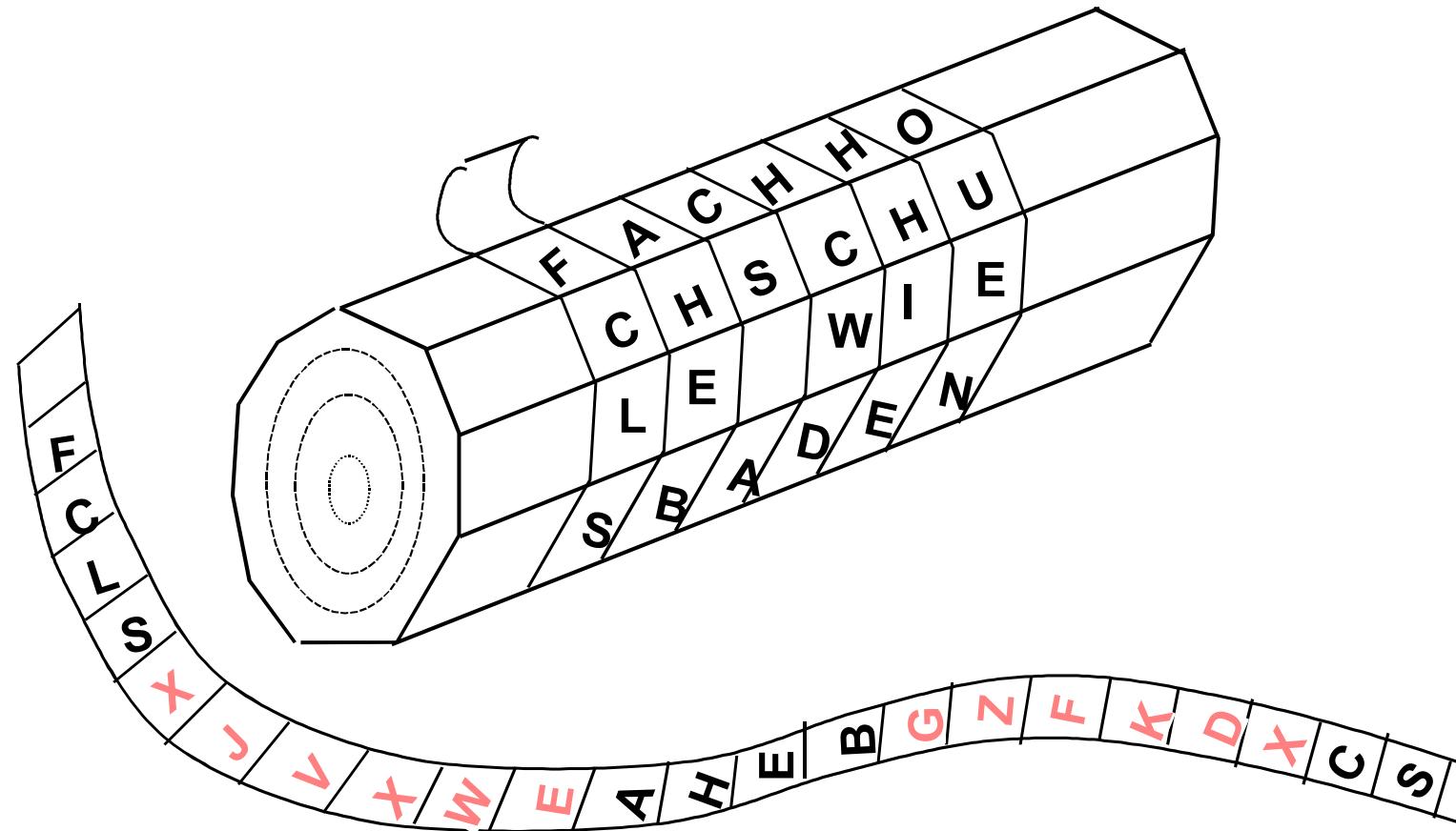
### **Kap. 3: Monoalphabetische Chiffren und deren Analyse**

#### **Teil 2: Einfache Chiffriermaschinen**

- Skytale
- Alberti-Scheibe

# Kryptographische Algorithmen

## Transpositionschiffren



- Für die rechnergestützte Realisierung einer Substitutionschiffre benötigen wir Rechenregeln für das Addieren und Multiplizieren von Zahlen in  $\{0, 1, 2, \dots, n-1\}$ , deren Resultat ebenfalls in  $\{0, 1, 2, \dots, n-1\}$  liegt.
  - Ferner müssen für das erzielte Resultat die zuvor aufgestellten Rechenregeln weiterhin gelten.
  - Wir erreichen dies, indem wir Resultate größer als  $n-1$  durch  $n$  dividieren und den Divisionsrest als neues Ergebnis benutzen.
  - Zum Rechnen mit Resten benötigen wir des weiteren einige grundlegende Sätze aus der elementaren Zahlentheorie (vgl. Kap. II), insbesondere zum Rechnen mit Zahlen **modulo**  $n$ .
-

- Julius Caesar (100 bis 44 v. Chr.)
- Jedes Klartextzeichen wird um **drei** Positionen verschoben.

<u>Klartext:</u>	a	b	c	d	e	f	g	...	z
<u>Chiffertext:</u>	D	E	F	G	H	I	J	...	C

- Verallgemeinerung: Bei einer Verschiebechiffre wird jedes Klartextzeichen **z** durch ein um **k** Zeichen im Alphabet verschobenes Zeichen ersetzt.

Es sei A ein Alphabet mit n Zeichen, die von 0 bis n-1 durchnumeriert sind.

Dann gilt für eine Verschiebechiffre allgemein: **E : z → (z + k) mod n**

- Eigenschaften:
  - Durch Probieren leicht zu knacken
  - Durchführung von Häufigkeitsanalysen möglich

Die 26 möglichen Verschiebechiffren:

Klartext:      a b c d e f g h i j k l m n o p q r s t u v w x y z

Chiffretexte:    0 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

                      1 B C D E F G H I J K L M N O P Q R S T U V W X Y Z A

Schlüssel          2 C D E F G H I J K L M N O P Q R S T U V W X Y Z A B

k                    3 D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

                      4 E F G H I J K L M N O P Q R S T U V W X Y Z A B C D

                      5 F G H I J K L M N O P Q R S T U V W X Y Z A B C D E

                      6 G H I J K L M N O P Q R S T U V W X Y Z A B C D E F

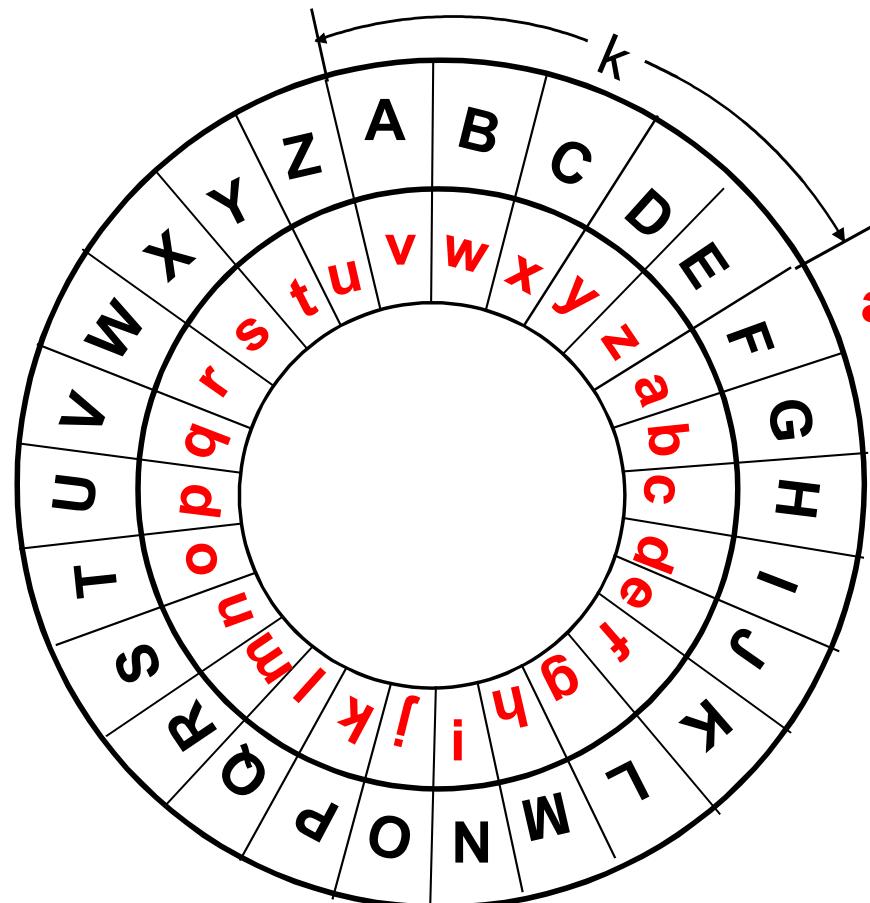
                      7 H I J K L M N O P Q R S T U V W X Y Z A B C D E F G

                      8 I J K L M N O P Q R S T U V W X Y Z A B C D E F G H

...

25 Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

---



a b c ... z = Klartextzeichen  
A B C ... Z = Chiffertextzeichen

Schlüssel

hier:  $k = 5$

## Kap. 3: Monoalphabetische Chiffren und deren Analyse

### Teil 3: Komplexe Verschiebechiffren

- Multiplikative Chiffren
- Affine Tauschchiffren

- Bei einer multiplikativen Chiffre über dem Alphabet A wird jedes Klartextzeichen  $z$  mit einer Zahl  $t \in \{0, 1, \dots, n\}$  multipliziert.
- $t$  und  $n = |A|$  (Mächtigkeit) müssen teilerfremd sein, d. h. es muss gelten:  $\text{ggT}(t, n) = 1$
- Die Chiffrevorschrift lautet:

$$E : z \rightarrow (z \cdot t) \bmod n \quad \text{mit} \quad t \in \mathbb{Z}_n \setminus \{0\} = \{1, \dots, n-1\}$$

- Zu jeder multiplikativen Chiffre  $E$  mit  $\text{ggT}(t, n) = 1$  gibt es eine multiplikative Dechiffrierfunktion  $D$  mit  $D(E(z)) = z$  für  $\forall z \in A$ .
- Es gilt:

$$D : z' \rightarrow (b \cdot z') \bmod n ,$$

wobei  $b \in \mathbb{Z}_n$  mit  $t \cdot b \equiv 1 \pmod{n}$  ist.

---

- Sei  $\text{ggT}(t, n) = 1$ . Dann wird jede Chiffre  
 $E : z \rightarrow (z \cdot t + k) \bmod n$  mit  $t \in \mathbb{Z}_n \setminus \{0\} = \{1, \dots, n-1\}$   
eine *affine Chiffre* oder Tauschchiffre genannt.
  - Um aus Chiffrezeichen  $z'$  wieder Klartextzeichen berechnen zu können, wendet man die Dechiffrierfunktion  $D$  wie folgt an:  
$$D : z' \rightarrow (b \cdot z' + l) \bmod n,$$
wobei  $b, l \in \mathbb{Z}_n$  mit  $t \cdot b \equiv 1 \bmod n$  und  $l \cdot t \equiv (n - k) \bmod n$  gilt.  
Ferner besteht der Zusammenhang:  
$$l = b(n - k) \bmod n$$
  - Beispiel:  $t = 5; k = 7; n = 26$   
 $\Rightarrow E : z' = (5 \cdot z + 7) \bmod 26$  mit der Dechiffrierfunktion  
 $D : z = (21 \cdot z' + 9) \bmod 26$ , um aus  $z'$  wieder  $z$  berechnen zu können  $\rightarrow b = 21; l = 9$  und  $t \cdot b = 105 \equiv 1 \bmod 26$ .
-

$z$	$z' = (5 \cdot z + 7) \bmod 26$	$z = (21 \cdot z' + 9) \bmod 26$
1	12	$(21 \cdot 12 + 9) \bmod 26 = 1$
2	17	$(21 \cdot 17 + 9) \bmod 26 = 2$
3	22	$(21 \cdot 22 + 9) \bmod 26 = 3$
4	1	$(21 \cdot 1 + 9) \bmod 26 = 4$
...	...	...
12	15	$(21 \cdot 15 + 9) \bmod 26 = 12$

### **Kap. 3: Monoalphabetische Chiffren und deren Analyse**

#### **Teil 4: Häufigkeitsanalyse**

- Buchstabenverteilungen
- Bi- und Trigramme

# Alphabet

## Häufigkeiten (1)

Buchstabe	Häufigkeit [%]	Buchstabe	Häufigkeit [%]
a	6,51	n	9,78
b	1,89	o	2,51
c	3,06	p	0,79
d	5,08	q	0,02
e	17,40	r	7,00
f	1,66	s	7,27
g	3,01	t	6,15
h	4,76	u	4,35
i	7,55	v	0,67
j	0,27	w	1,89
k	1,21	x	0,03
l	3,44	y	0,04
m	2,53	z	1,13

### Gruppenhäufigkeiten und Bigramme der deutschen Sprache:

Gruppe	Anteil der Buchstaben dieser Gruppe an einem Text in [%]
e, n	27,18
i, s, r, a, t	34,48
d, h, u, l, c, g, m, o, b, w, f, k, z	36,52
p, v, j, y, x, q	1,82

Buchstabenpaar	Häufigkeit [%]	Buchstabenpaar	Häufigkeit [%]
en	3,88	nd	1,99
er	3,75	ei	1,88
ch	2,75	ie	1,79
te	2,26	in	1,67
de	2,00	es	1,52

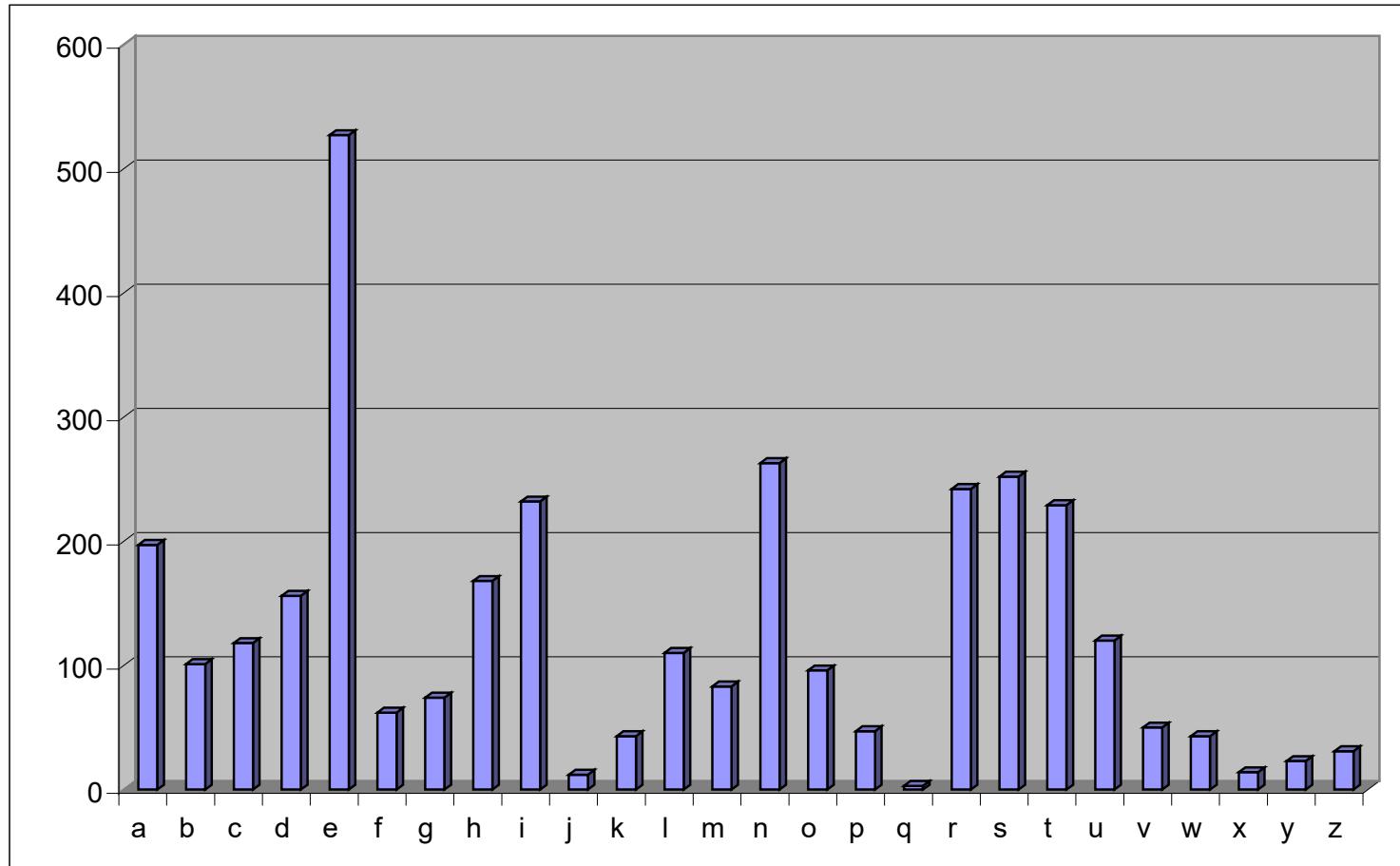
# Alphabet

## Häufigkeiten (3)

Buchstabe	Häufigkeit [%]	Buchstabe	Häufigkeit [%]
a	8,2	n	6,7
b	1,5	o	7,5
c	2,8	p	1,9
d	4,3	q	0,1
e	12,7	r	6,0
f	2,2	s	6,3
g	2,0	t	9,1
h	6,1	u	2,8
i	7,0	v	1,0
j	0,2	w	2,4
k	0,8	x	0,2
l	4,0	y	2,0
m	2,4	z	0,1

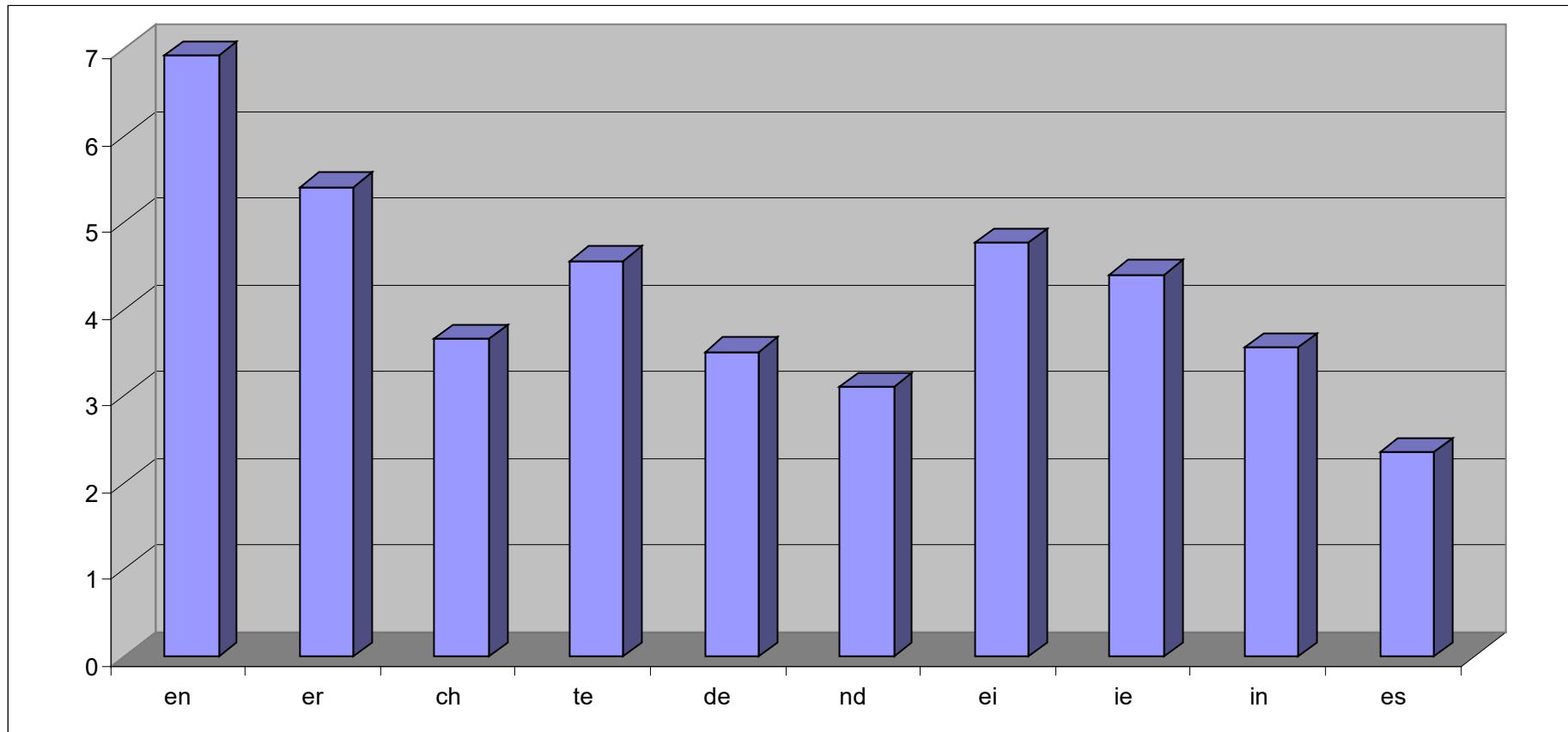
# Alphabet

## Häufigkeiten (4)



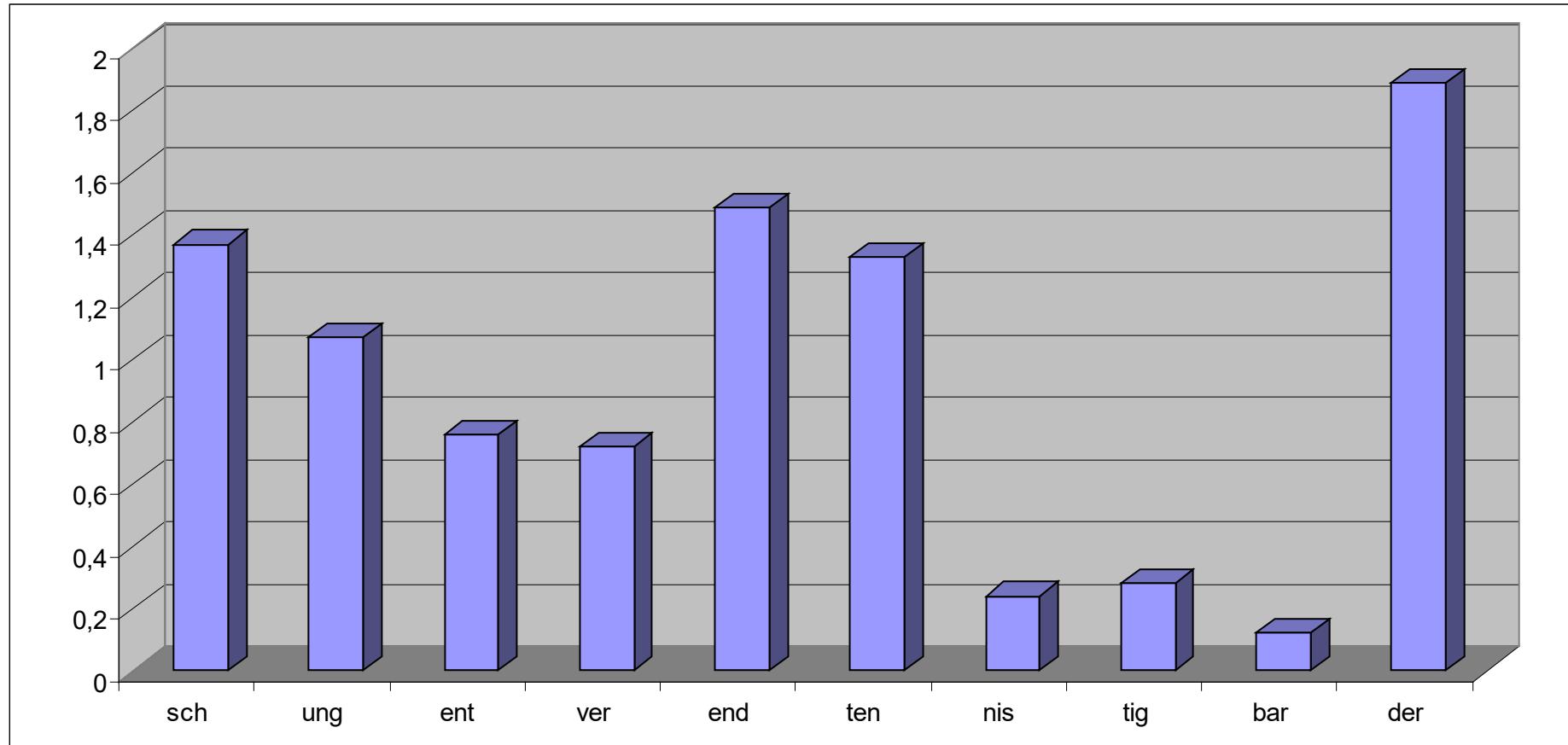
# Alphabet

## Häufigkeiten (5)



# Alphabet

## Häufigkeiten (6)



```
/* Datum: 19.07.2002 */  
/* Autor: Bernhard Geib */  
/* Funktion: Verschluesselung mit einer affinen Tauschchiffre */  
  
#include <stdio.h>  
  
int main (void)  
{  int c;  
  
    c = getchar();  
    while (c != EOF)  
    {  
        if (c != '\n')  
        {  
            c = (17 * c + 4) % 256;  
        }  
        printf ("%c", c);  
        c = getchar();  
    }  
    return 0;  
}
```

```
/* Datum: 19.07.2002 */  
/* Autor: Bernhard Geib */  
/* Funktion: Entschluesselung mit einer affinen Tauschchiffre */  
  
#include <stdio.h>  
  
int main (void)  
{  int c;  
  
    c = getchar();  
    while (c != EOF)  
    {  
        if (c != '\n')  
        {  
            c = (241 * c + 60) % 256;  
        }  
        printf ("%c", c);  
        c = getchar();  
    }  
    return 0;  
}
```

---

# **Security**

## **- LV 4121 und 4241 -**

**Symmetrische Verfahren und moderne Blockchiffren**

- Gegenüberstellung symmetrische und asymmetrische Kryptoverfahren
  - Symmetrische Blockverschlüsselung und **DES-Algorithmus**
  - Advanced Encryption Standard (**AES**)
  - **Betriebsarten** für blockorientierte Verschlüsselungsalgorithmen
  - Symmetrische Bitstromverschlüsselung (one time pad)
  - Grundlegende Aspekte des Schlüssel- und Sicherheitsmanagements
  - **DH-Schlüsselaustausch** (gegenseitige Schlüsselabsprache)
  - Schlüsselhierarchie und Schlüsselklassen
-

## Kap. 4: Symmetrische Verfahren und moderne Blockchiffren

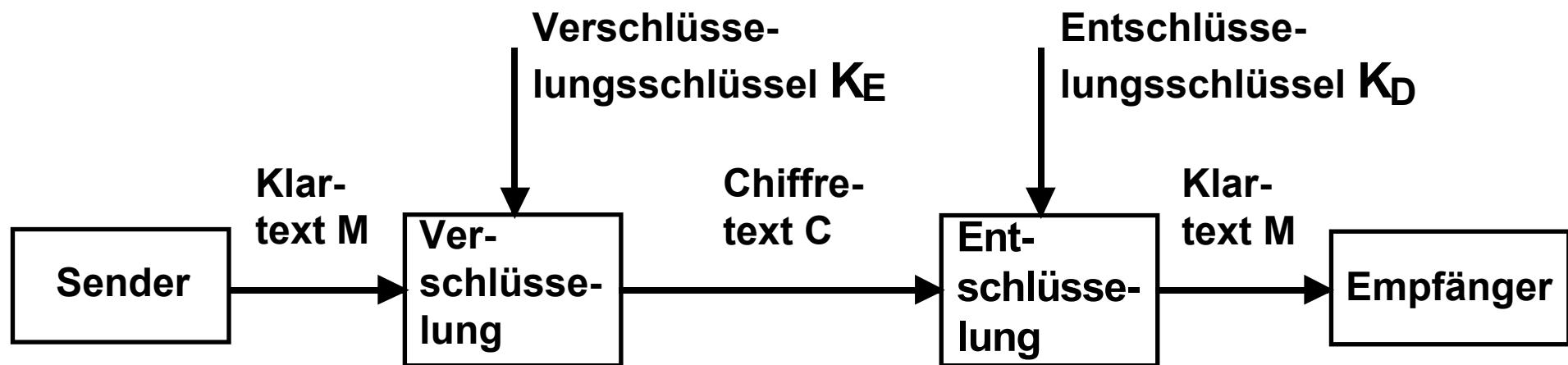
### Teil 1: Symmetrische Blockverschlüsselung

- Schlüsselgesteuerte Transformation
- Gegenüberstellung der Chiffrierverfahren
- Data Encryption Standard (DES-Algorithmus)
- Advanced Encryption Standard (AES-Algorithmus)

# Verschlüsselung

Schlüsselgesteuerte Transformation

---



### Symmetrische vs. asymmetrische Chiffrierverfahren:

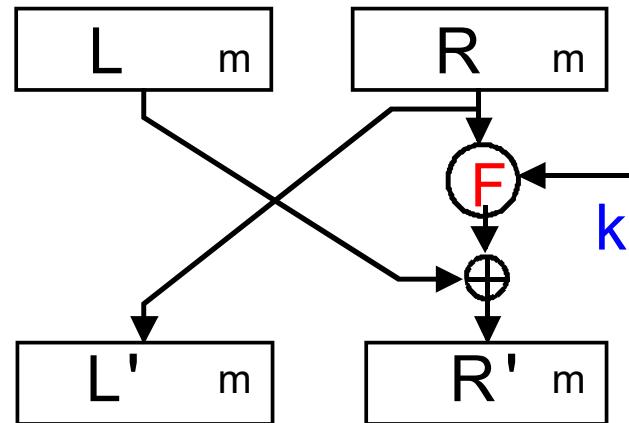
Symmetrische Verfahren	Asymmetrische Verfahren
<p><b>Vorteile:</b></p> <ul style="list-style-type: none"><li>• Sie sind <b>schnell</b>, d. h. sie haben einen <b>hohen Datendurchsatz</b>.</li><li>• Die Sicherheit ist im wesentlichen durch die Schlüssellänge festgelegt.</li></ul> <p><b>Nachteile:</b></p> <ul style="list-style-type: none"><li>• Jeder Teilnehmer muß sämtliche Schlüssel seiner Kommunikationspartner geheimhalten.</li><li>• Es ist ein komplexeres Schlüsselmanagement erforderlich.</li></ul>	<p><b>Vorteile:</b></p> <ul style="list-style-type: none"><li>• Jeder Teilnehmer muß nur seinen eigenen <b>privaten Schlüssel</b> geheimhalten.</li><li>• Sie bieten elegante Lösungen für die Schlüsselverteilung in Netzen.</li></ul> <p><b>Nachteile:</b></p> <ul style="list-style-type: none"><li>• Sie sind langsam, d. h. sie haben im allgemeinen einen deutlich geringeren Daten-durchsatz als symmetrische Verfahren.</li><li>• Es gibt wesentlich bessere Attacken als das Durchprobieren aller Schlüssel.</li></ul>

# Idee einer Feistel-Chiffre

Konstruktionsprinzip

## Transformation und Rekonstruktion:

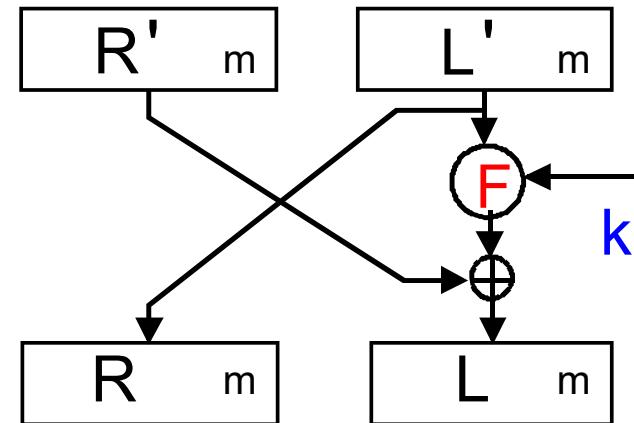
Verschlüsselung



**Transformation** der Eingangsblöcke  $L$  und  $R$  in Ausgangsblöcke  $L'$  und  $R'$ , wobei  $k \in K$  der Schlüssel ist.

$$L' = R \text{ und } R' = F(R, k) \oplus L$$

Entschlüsselung



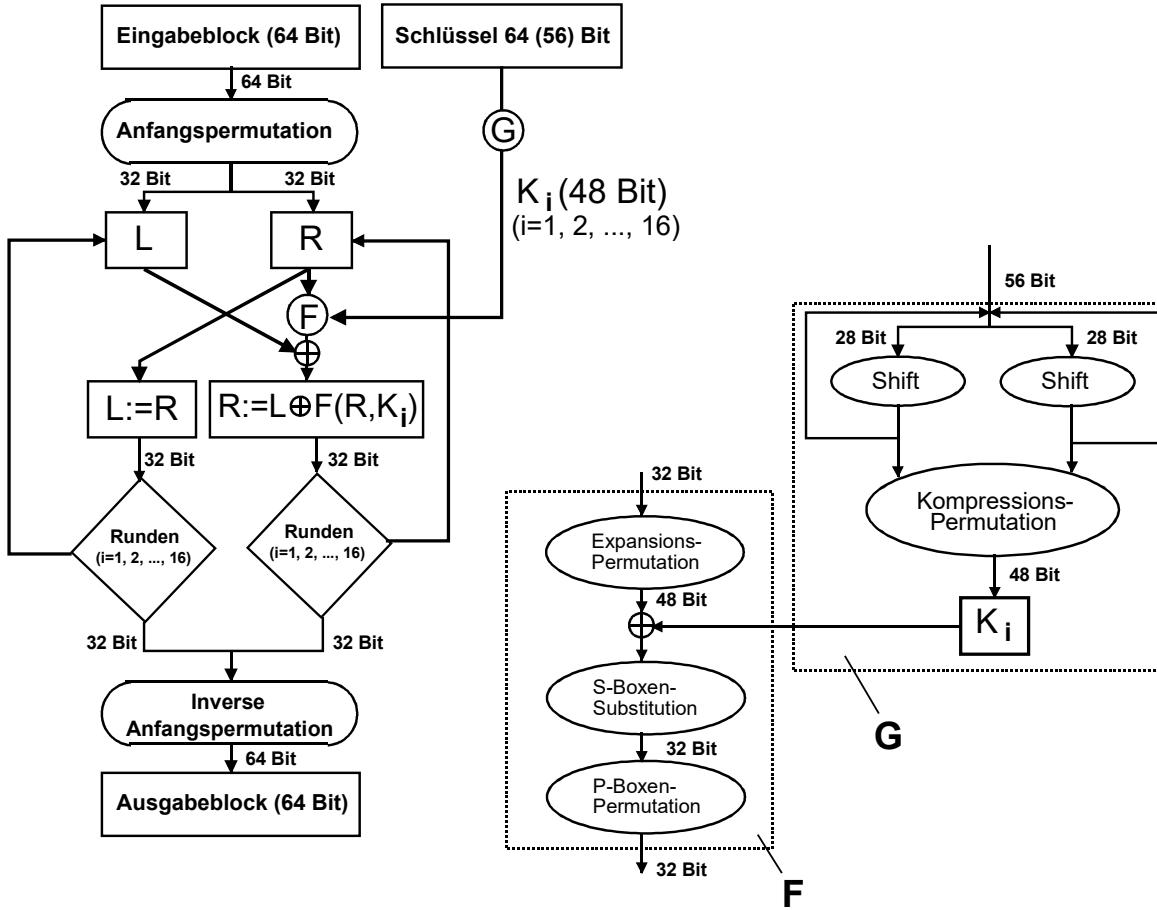
**Rekonstruktion** der Eingangsblöcke  $L$  und  $R$  aus den Ausgangsblöcken  $L'$  und  $R'$ , wobei  $k \in K$  der Schlüssel ist.

$$R = L' \text{ und } L = F(L', k) \oplus R'$$

- Eines der bedeutesten Hilfsmittel für den Entwurf heutiger Blockchiffren ist das von Horst Feistel entwickelte Konstruktionsprinzip einer **Feistel-Chiffre**.
- Der wesentliche Aspekt besteht darin, dass eine beliebige Funktion  $F : \{0, 1\}^m \times k \rightarrow \{0, 1\}^m$  zum Einsatz kommen kann. Die Funktion  $F$  muss nicht einmal umkehrbar sein.
- Soll eine Feistel-Chiffre sowohl zur Ver- als auch Entschlüsselung verwendet werden, so ist es notwendig, die beiden Ausgabeblöcke zu vertauschen (vgl. DES).
- Wendet man das Konstruktionsprinzip wiederholt auf die sich ergebenden Ausgangsblöcke an, so können **sichere Verschlüsselungssysteme** konstruiert werden.

# Data Encryption Standard

## Prinzipieller Aufbau

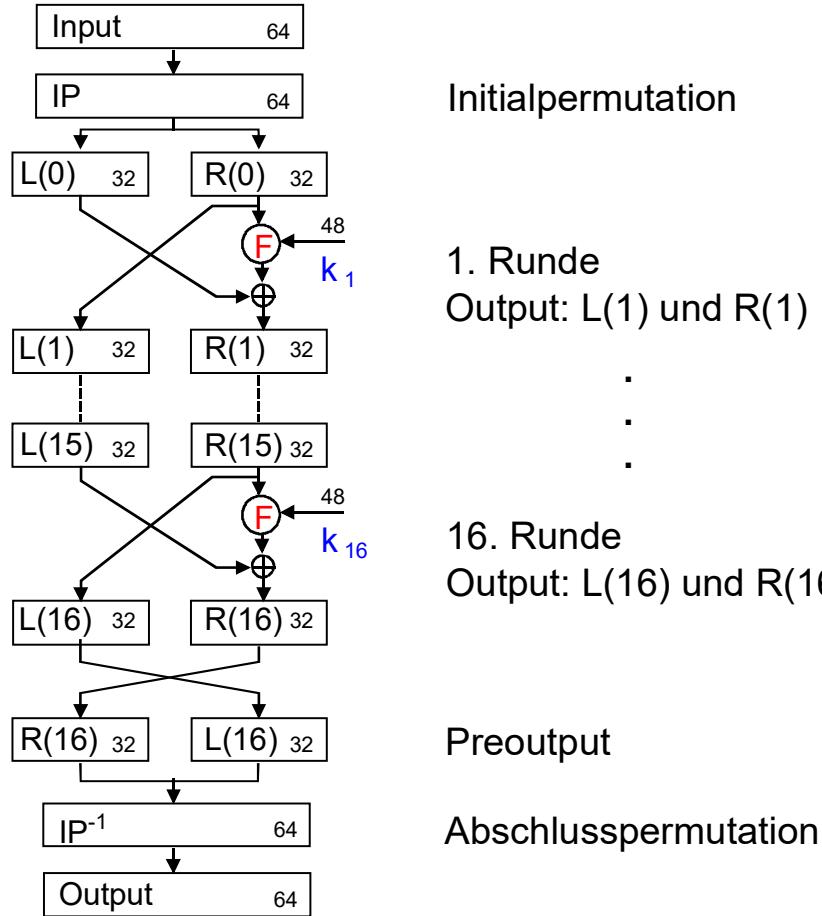


### DES:

- 1974 veröffentlicht
- ANSI-Standard (USA)
- Blocklänge 64 Bit
- Schlüssellänge 56 Bit
- ca.  $7,2 \cdot 10^{16}$  Schlüssel
- Anfangspermutation
- Zwei Hälften L und R
- 16 Runden Iteration
- Rundenschlüssel 48 Bit
- **Substitution (nichtlinear)**
- **Transposition**
- Ver-/ Entschlüsselung
- Abschlusspermutation

# Data Encryption Standard

## Verschlüsselungsalgorismus



### DES:

- 16 Teilschlüssel  $k_1$  bis  $k_{16}$  aus 56-Bit-Schlüssel mittels **Schlüsselauswahlfunktion**
- Algorithmus so ausgelegt, dass er für Ver- und Entschlüsselung identisch ist
- Verschlüsselung  $k_1, k_2, \dots, k_{16}$  und Entschlüsselung  $k_{16}, k_{15}, \dots, k_1$
- $IP^{-1}$  zu IP inverse Permutation sind beide ohne Sicherheitsbedeutung
- $IP(X) = (x_{58}, x_{50}, x_{42}, \dots, x_{15}, x_7)$
- $IP^{-1}(Y) = (y_{40}, y_8, y_{48}, \dots, y_{57}, y_{25})$

# Data Encryption Standard

## Ein-/ Ausgangspermutation

### Initialpermutation IP

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	35	47	39	31	23	15	7

### Abschlusspermutation IP <sup>-1</sup>

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

heißt:

schreibe 1. Bit an Position 58 und  
Bit 2 an Position 50.

entsprechend:

schreibe 58. Bit an Position 1 bzw.  
Bit 50 an Position 2 zurück.

### Iteration der Verschlüsselung:

Im Anschluss an die **Initialpermutation IP** wird der **Block IP(X)** in einen linken **Block L** und einen rechten **Block R** zerlegt ( $L \parallel R = IP(X)$ ). Beide Blöcke sind 32 Bit lang. Auf  $L = (x_{58}, x_{50}, \dots, x_8)$  und  $R = (x_{57}, x_{49}, \dots, x_7)$  wird folgende Operation angewandt:

```
L(0) := L ;
R(0) := R ;
for i := 1 to 16 do
    begin
        L(i) := R(i - 1) ;
        R(i) := L(i - 1)  $\oplus$  F(R(i - 1), ki) ;
    end ;
```

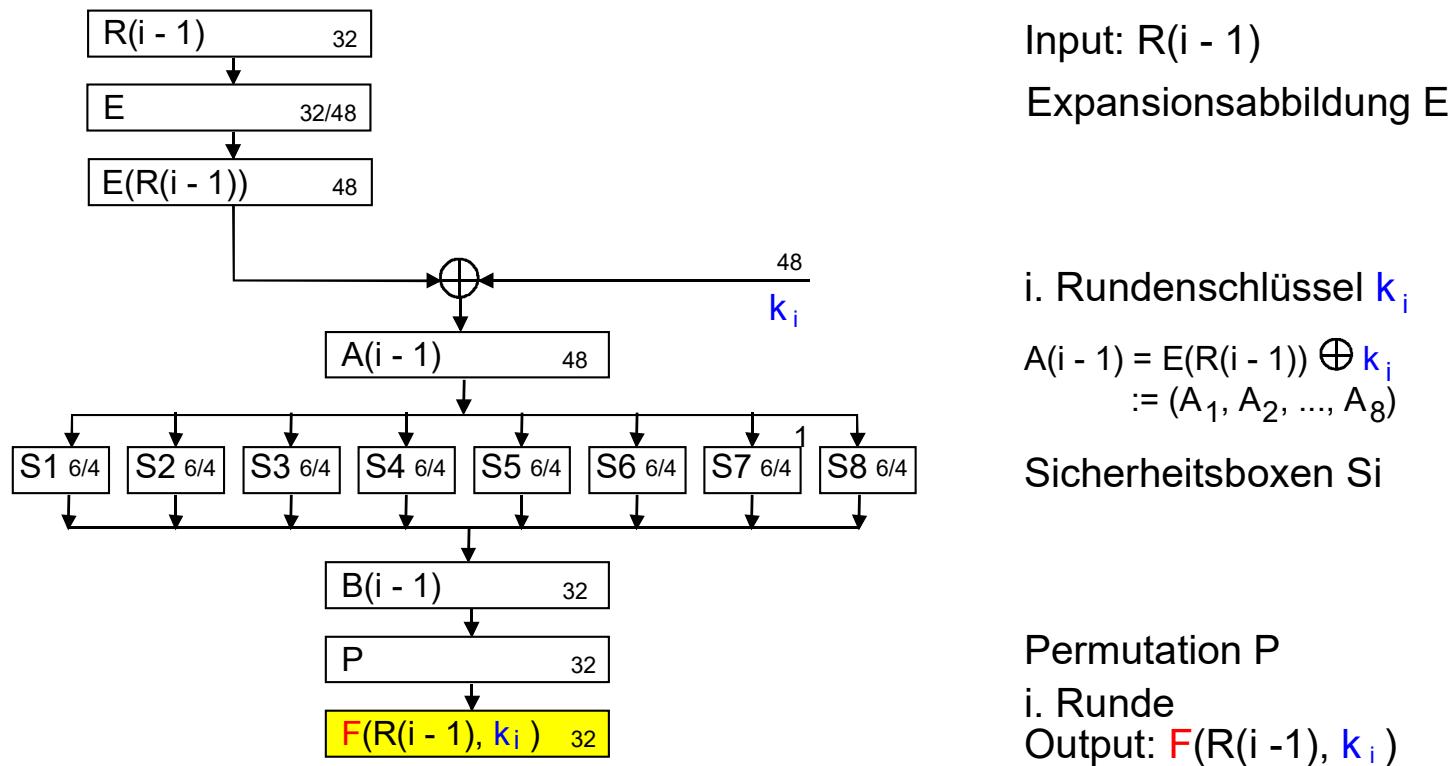
---

# Data Encryption Standard

## Die Rundenverschlüsselung

### Die Funktion $F(R(i - 1), k_i)$ :

Kern des gesamten Verfahrens  $F : \{0, 1\}^{32} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{32}$ .



# Data Encryption Standard

## Die Rundenverschlüsselung

### Expansionsabbildung E

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

### Indextabelle der Permutation P

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

heißt:

$$R(i-1) = (r_1, r_2, r_3, \dots, r_{31}, r_{32}) \Rightarrow$$

$$E(R(i-1)) = (r_{32}, r_1, r_2, \dots, r_{32}, r_1)$$

entsprechend:

$$B(i-1) = (b_1, b_2, b_3, \dots, b_{32}) \Rightarrow$$

$$P(B(i-1)) = (b_{16}, b_7, b_{20}, \dots, b_{25})$$

### Die S-Box S1:

- Das Ergebnis der bitweisen XOR-Bildung **A** bildet den Input für die **S-Boxen** S1 bis S8, wobei  $A_j$  der zu  $S_j$  gehörende Input ist.
- Jede S-Box kann vier unterschiedliche Substitutionen realisieren, wobei die einzelnen S-Boxen durch eine  $4 \times 16$ -Matrix festgelegt sind.
- Die Zeilen werden mit **0, 1, 2, 3** und die Spalten mit **0, 1, ..., 14, 15** bezeichnet (im Beispiel ist die S-Box S1 wiedergegeben).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

### Die S-Box S1:

- Jede Zeile der S-Box  $S_j$  stellt eine Substitution  $S_{j,k} : \{0, 1\}^4 \rightarrow \{0, 1\}^4$  dar, wobei  $k$  die Zeilennummer bezeichnet.
  - Auf diese Weise können mit acht S-Boxen insgesamt 32 verschiedene Substitutionen realisiert werden.
  - Ist  $A_j = (a_{j1}, a_{j2}, a_{j3}, a_{j4}, a_{j5}, a_{j6})$  ein 6-Bit-Block.  
Dann bestimmen die Bits  $a_{j1}a_{j6}$  – als Binärzahl gelesen – die **Zeilennummer** und  $a_{j2}a_{j3}a_{j4}a_{j5}$  – ebenfalls als Binärzahl aufgefasst – die **Spaltennummer** der S-Box  $S_j$ .
  - Der zugehörige Matrixeintrag  $S_{j,a_{j1}a_{j6}} (a_{j2}a_{j3}a_{j4}a_{j5})$  legt das Substitutionsergebnis eindeutig fest, wobei jeder Eintrag als **Dezimalzahl** eine **Folge von 4 Bit** ergibt ( $\rightarrow$  Output der S-Box  $S_j$ ).
-

# Data Encryption Standard

Die S-Boxen

---

## Die DES Substitutionsboxen:

S1:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

# Data Encryption Standard

Die S-Boxen

---

## Die DES Substitutionsboxen:

S3:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

# Data Encryption Standard

Die S-Boxen

---

## Die DES Substitutionsboxen:

S5:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

# Data Encryption Standard

Die S-Boxen

---

## Die DES Substitutionsboxen:

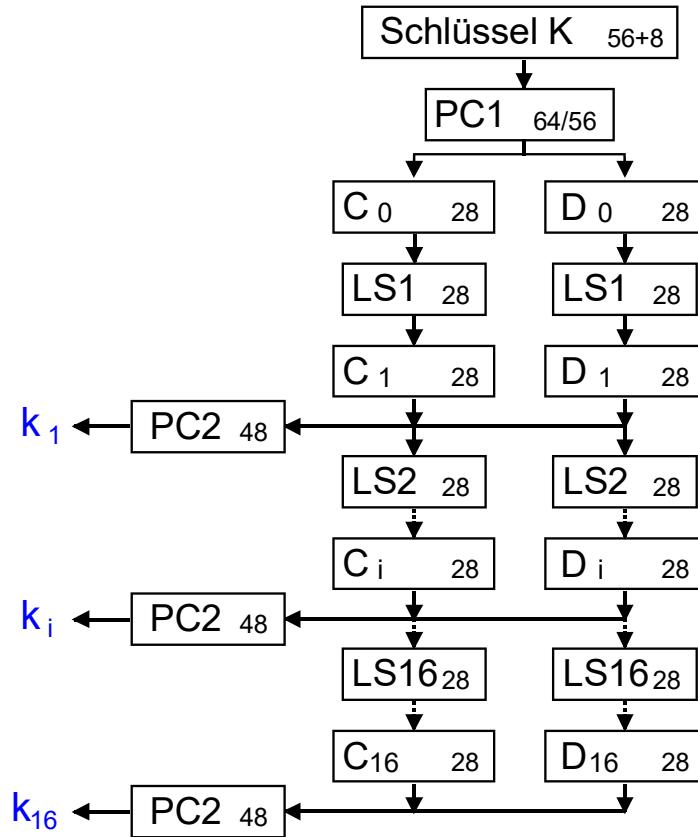
S7:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	4	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

### Die Schlüsselauswahlfunktion:



Gegeben:  $K = (k_1, k_2, \dots, k_{64})$

Permuted Choice 1  
(Schlüsselreduktion und Permutation)

Zirkuläre Linksshifts

Permuted Choice 2

1. Rundenschlüssel (Konkatenation)

.

.

i. Rundenschlüssel (Konkatenation)

.

.

16. Rundenschlüssel (Konkatenation)

### Die Schlüsselauswahlfunktion:

- Die bei jeder Iteration benutzten **Rundenschlüssel  $k_i$**  werden aus dem gegebenen **Schlüssel  $K$**  ermittelt.
  - Aus Sicherheitsgründen sollten alle Rundenschlüssel  $k_i$  verschieden sein (unterschiedliche Teilmengen aus  $K$ ).
  - Gegeben sei der vorgegebene Schlüssel  $K = (k_1, k_2, \dots, k_{64})$  mit den **Paritätsbits  $k_i$**  an den Stellen  $i = 8(8)64$ .
  - PC1 entfernt alle Paritätsbits und reduziert den Schlüssel  $K$  auf 56 **aktive** Schlüsselbits, die zudem permutiert werden.
  - Der resultierende Wert  $PC1(K)$  ergibt sich zu  $(k_{57}, k_{49}, \dots, k_{12}, k_4)$ .
-

### Permutation PC1

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

### Permutation PC2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

heißt:

$$PC1(K) = (k_{57}, k_{49}, k_{41}, \dots, k_{12}, k_4)$$

$$\Rightarrow C_0 = (k_{57}, k_{49}, \dots, k_{36}) \text{ und}$$

$$\Rightarrow D_0 = (k_{63}, k_{55}, \dots, k_4)$$

entsprechend:

$$C_1 = (k_{49}, k_{41}, \dots, k_{44}, k_{36}, k_{57}) \text{ und}$$

$$D_1 = (k_{55}, k_{47}, \dots, k_{12}, k_4, k_{63}) \Rightarrow$$

$$k_1 = (k_{10}, k_{51}, \dots, k_{13}, k_{62}, k_{55}, k_{31})$$

- Der permutierte Wert  $PC1(K)$  wird in eine linke Hälfte  $C_0 = (k_{57}, k_{49}, \dots, k_{36})$  und eine rechte Hälfte  $D_0 = (k_{63}, k_{55}, \dots, k_4)$  aufgeteilt.
- Die Vektoren  $C_i \parallel D_i$  (Konkatenation) werden für  $i = 1, 2, \dots, 16$  rekursiv aus  $C_{i-1} \parallel D_{i-1}$  durch **zirkuläres Linksshiften**  $LS_i$  der Hälften  $C_{i-1}$  und  $D_{i-1}$  um eine oder zwei Bitpositionen abgeleitet.
- Die Hälften werden dabei getrennt geshiftet.  $C_1 = LS_1(C_0)$  bzw.  $D_1 = LS_1(D_0)$  um eine Position zirkulär nach links.

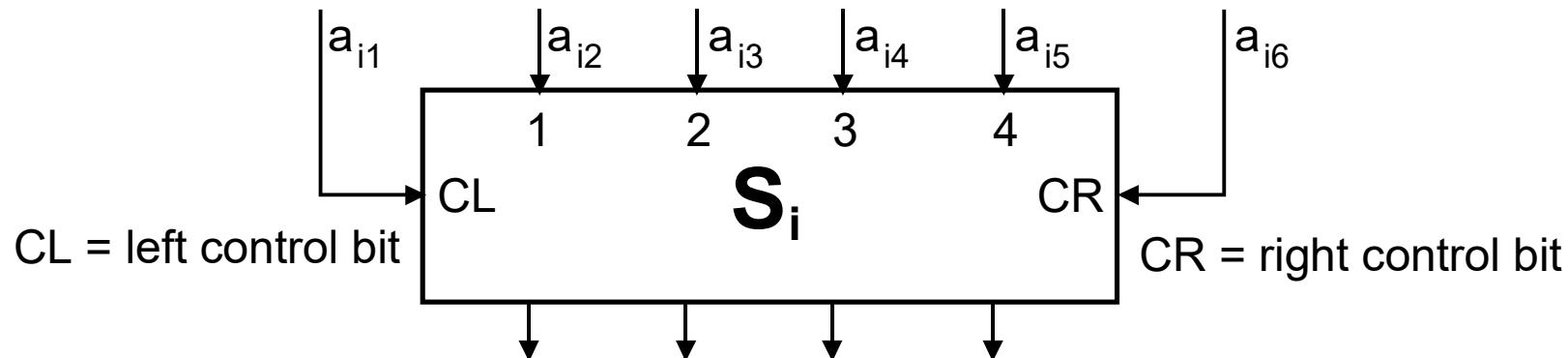
Nummer der Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Anzahl der Linksshifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

---

- Damit ergeben sich beispielsweise die beiden Hälften:  
 $C_1 = (k_{49}, k_{41}, \dots, k_{36}, k_{57})$  und  $D_1 = (k_{55}, k_{47}, \dots, k_4, k_{63})$  aufgeteilt.
- PC2 bestimmt schließlich für  $i = 1, 2, \dots, 16$  aus den Konkatenationen  $C_i \| D_i$  den Rundenschlüssel  $k_i$ .
- Hierzu werden zuerst die Bits von  $C_i \| D_i$  auf den Positionen 9, 18, 22, 25, 35, 38, 43 und 54 entfernt.
- Die verbleibenden 48 Bits werden abschließend der Permutation PC2 unterworfen.

- Der **DES** wurde von der Firma IBM entwickelt und auf Empfehlung des National Bureau of Standards, Washington D. C., 1977 genormt.
  - Seine offizielle Beschreibung erfährt der DES in **FIPS PUB 46** (Federal Information Processing Standards Publication).
  - DES wurde explizit in Übereinstimmung mit den Shannonschen Design-Prinzipien bezüglich Konfusion und Diffusion entwickelt.
  - Lokale Diffusion und Konfusion wird durch die im hohen Grad **nicht-lineare** Funktion  $F_i = F(R(i - 1), k_i)$  innerhalb jeder Runde  $i$  erzeugt, wobei die tatsächliche Nichtlinearität in den **S-Boxen** verankert ist.
  - Weitere Diffusion wird durch Transposition bzw. Swapping in zwei Hälften L bzw. R innerhalb jeder Runde (mit Ausnahme der letzten) erzeugt.
-

### Die Substitutionsboxen (S-Boxen) des DES:



- Für jede der vier Kombinationen der beiden Steuerbits  $CL$  und  $CR$  liefert die S-Box  $S_i$  eine unterschiedliche Permutation in Abhängigkeit der vier Input-Bits  $a_{i2} a_{i3} a_{i4} a_{i5}$  (4-Tuple).
  - In dieser **Unterschiedlichkeit** ist die **Nichtlinearität** der S-Boxen und letzten Endes die hohe Sicherheit des DES begründet.
-

## Design-Regeln für die S-Boxen (Empfehlung):

- (1) Für jede Kombination der beiden Steuerbits CL und CR sollten die S-Boxen eine über GF(2) **nichtlineare Transformation** von dem Input-4-Tuple in das Output-4-Tuple darstellen.
- (2) Eine jegliche Änderung der 6 Input-Bits sollte **mindestens zwei** Output-Bits ändern.
- (3) Wenn ein der 6 Input-Bits konstant gehalten wird, dann sollten die sich für die Output-4-Tuples ergebenden  $2^5 = 32$  Möglichkeiten eine gute Balance von 0 und 1 erzielen, falls die übrigen 5 Input-Bits variiert werden.

Jedoch sind die tatsächlichen Design-Prinzipien für die S-Boxen des U.S.-Government **nie** veröffentlicht worden (U.S. „classified“ information).

---

### Schwache und semi-schwache Schlüssel:

Definition:

**K** ist ein schwacher (oder sogenannter self-dualer) Schlüssel, wenn:

$$\text{DES}_K(\cdot) = \text{DES}^{-1}K(\cdot),$$

d. h. wenn die Verschlüsselungsfunktion für **K** mit der Entschlüsselungsfunktion übereinstimmt, da in diesem Fall die abgeleiteten Teilschlüssel **k<sub>i</sub>** nicht alle voneinander verschieden wären.

- DES hat mindestens vier schwache Schlüssel, die es unbedingt zu vermeiden gilt.
- Es ist sehr wahrscheinlich, dass es außer diesen vier Schlüsseln keine weiteren schwache Schlüssel gibt.

Begründung:

0000 0001	...	0000 0001	0000 0001	...	0000 0001
1111 1110	...	1111 1110	1111 1110	...	1111 1110
1110 0000	...	1110 0000	1111 0001	...	1111 0001
<u>0001 1111</u>	...	<u>0001 1111</u>	<u>0000 1110</u>	...	<u>0000 1111</u>

Byte 1                  Byte 4                  Byte 5                  Byte 8

- Aufgrund der Permutation **PC1** machen diese 4 Schlüssel  $C_0$  entweder zu  $(00 \dots 0)$  oder  $(11 \dots 1)$  und  $C_1$  entweder zu  $(00 \dots 0)$  oder  $(11 \dots 1)$ , so dass folgt:  $k_1 = k_2 = \dots = k_{16}$ .
- Hieraus folgt  $(k_1, k_2, \dots, k_{16}) = (k_{16}, k_{15}, \dots, k_1)$ , so dass

$$\text{DES}_{\mathbf{K}}(\cdot) = \text{DES}^{-1}_{\mathbf{K}}(\cdot).$$

### Schwache und semi-schwache Schlüssel:

Definition:

**K** ist ein **semi-schwacher** (oder sogenannter dualer) Schlüssel, wenn:

$$\text{DES}\mathbf{K}'(\cdot) = \text{DES}^{-1}\mathbf{K}(\cdot),$$

d. h. wenn die Verschlüsselungsfunktion für unterschiedliche Schlüssel **K'** und **K** mit der Entschlüsselungsfunktion übereinstimmt, da in diesem Fall die abgeleiteten Teilschlüssel **k<sub>i</sub>** nicht alle voneinander verschieden wären.

- DES hat mindestens 12 semi-schwache Schlüssel.
  - Semi-schwache Schlüssel erscheinen immer paarweise.
  - Es ist sehr wahrscheinlich, dass es außer diesen 12 Schlüsseln keine weiteren semi-schwache Schlüssel gibt.
-

Begründung:

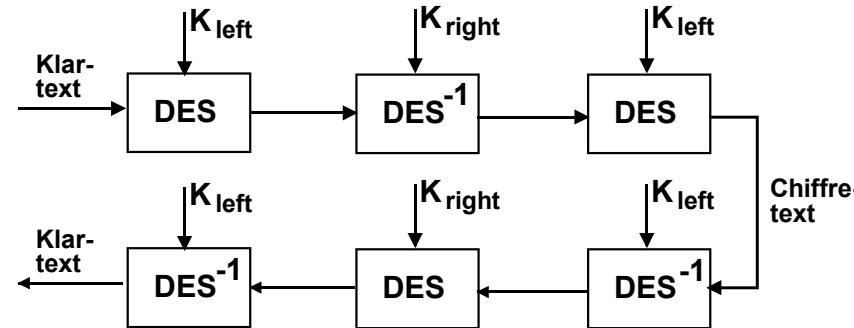
- Ein Schlüssel **K'**, der  $C_0 = (1010 \dots 10)$  und  $D_0$  entweder zu  $(00 \dots 0)$  oder  $(11 \dots 1)$  oder  $(1010 \dots 10)$  oder  $(0101 \dots 01)$  liefert, ist dual zum Schlüssel **K**, der  $C_0 = (0101 \dots 01)$  und  $D_0$  entweder zu  $(00 \dots 0)$  oder  $(11 \dots 1)$  oder  $(0101 \dots 01)$  oder  $(1010 \dots 10)$  ergibt.
  - Eine ähnliche Situation ergibt sich für  $C_0 = (0101 \dots 01)$  und  $D_0 = (1010 \dots 10)$  oder  $D_0 = (0101 \dots 01)$ .
  - Hieraus resultieren insgesamt 12 unterschiedliche Fälle.
  - Die Hauptschwäche des DES (im sogenannten ECB-Mode betrieben) besteht jedoch in der zu kurzen Schlüssellänge von lediglich 56 Bit.
-

# DES-Algorithmus

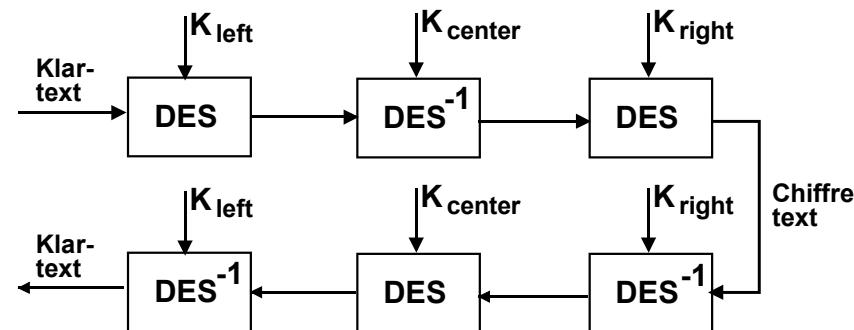
Varianten

---

Triple DES mit **doppelter** Länge  $K := K_{\text{left}} // K_{\text{right}}$



Triple DES mit **dreifacher** Länge  $K := K_{\text{left}} // K_{\text{center}} // K_{\text{right}}$



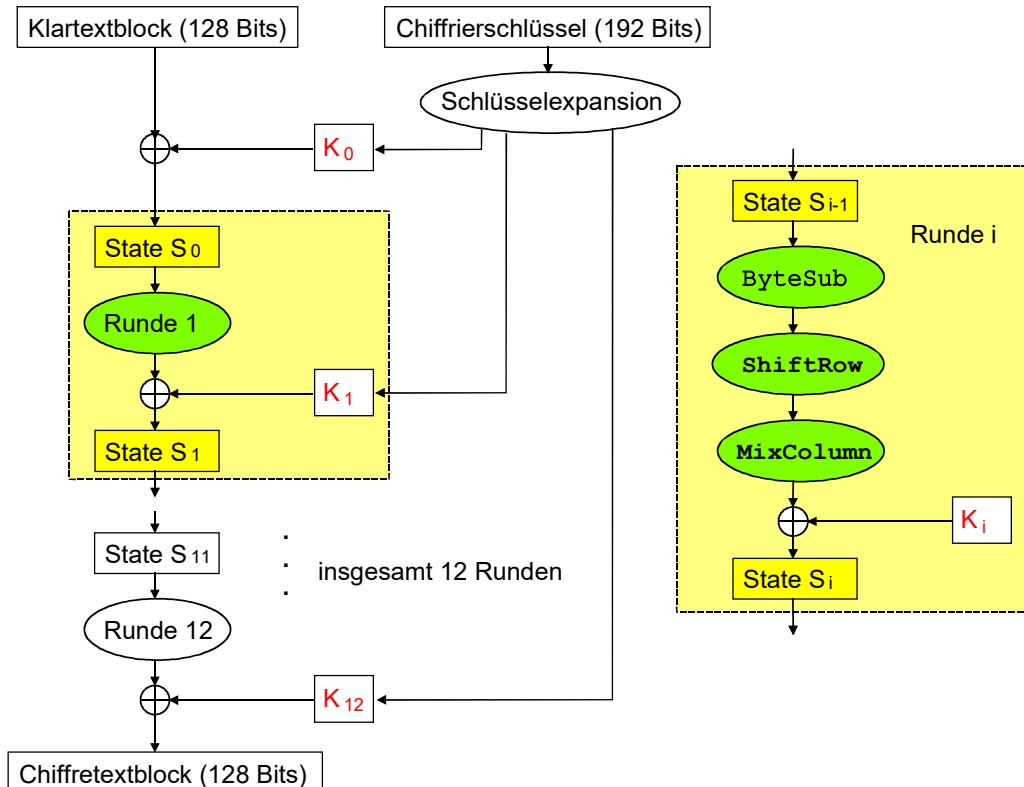
## Anforderungen, Funktionalitäten und Designkriterien:

- Im Jahr 1997 vom U.S.-amerikanischen NIST (National Institut of Standards and Technology) als Nachfolger für DES initiiert.
- Der Algorithmus von AES heißt **Rijndael** und wurde in Belgien von den Kryptologen **Joan Daemen** und **Vincent Rijmen** entwickelt.
- AES arbeitet auf einer Blöckgröße von 128 Bit mit Schlüssellängen von 128, 192 und 256 Bit.
- Spezielle Anforderungen an den Standard betrafen die Sicherheit, Einfachheit, Flexibilität, Effizienz und die Implementierung.
- Im Dezember 2001 wurde AES offiziell zum **FIPS 197** (Federal Information Processing Standards) erklärt.

# Advanced Encryption Standard

Prinzipien

## Schematischer Ablauf:



## AES (Rijndael):

- Symmetrische Blockchiffre
- Blocklänge **b** (hier: 128 Bit)
- Schlüssellänge **k** (128, 192 oder 256 Bit)
- Variable Rundenzahl **r** (zwischen 10 und 14)
- Schlüsselexpansion erzeugt  $r + 1$  **Rundenschlüssel  $K_0, K_1, \dots, K_r$**
- Zwischenergebnisse des Verschlüsselungsprozesses werden **Zustand  $S_i$**  genannt
- **Rundenschlüssel** haben **gleiche Länge** wie der jeweilige **Zustand**

### Zusammenhang zwischen r, b und k:

Rundenzahl r	Blocklänge		
Schlüssellänge	<b>b = 128</b>	<b>b = 192</b>	<b>b = 256</b>
<b>k = 128</b>	<b>10</b>	<b>12</b>	<b>14</b>
<b>k = 192</b>	<b>12</b>	<b>12</b>	<b>14</b>
<b>k = 256</b>	<b>14</b>	<b>14</b>	<b>14</b>

**Zustandsmenge und Schlüssel:** Jedes Element  $a_{m,n}$  bzw.  $k_{m,n}$  **1 Byte**

Zustände bzw. Klartext ( $b = 128$ )

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Schlüssel ( $k = 192$ )

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

## Verschlüsselungsprozedur:

- Vor der ersten und nach jeder Runde  $i$  wird der Rundenschlüssel (hier  $K_i = 128$  Bit) **XOR-verknüpft** mit dem aktuellen Zustand  $S_i$ .
  - Das Ergebnis dient als Eingabe für die nächste Runde  $i + 1$  bzw. als Chiffertext nach der letzten Runde.
  - Jede Runde (mit Ausnahme der letzten) besteht aus den Funktionen:
    - ByteSub** → nichtlineare S-Boxen (Substitutionsschritt)
    - ShiftRow** → zyklisches Verschieben der Zustandsmatrix
    - MixColumn** → invertierbare Matrixmultiplikation
  - Alle vorgenannten Transformationen (außer XOR-Verknüpfung) sind schlüsselunabhängig.
-

### Die **ByteSub**-Transformation:

- Diese Transformation stellt die **nichtlineare S-Box** von AES dar.
  - Sie wird auf jedes Byte  $a_{m,n}$  des Zustands angewandt und wird als „table-lookup“ implementiert (siehe u. a. Folie Nr. 42).
  - Sie entspricht dem Berechnen der **multiplikativen Inversen** in  **$GF(2^8)$**  bzw. **mod m(x)** gefolgt von der nachfolgenden affinen Transformation.
  - Die einzelnen Multiplikationen und Additionen der Komponenten sind **modulo zwei** zu berechnen.
  - Die Umkehrung von **ByteSub** erfolgt durch Anwendung der inversen affinen Transformation gefolgt von der multiplikativen Inversen in  $GF(2^8)$ .
-

### Die **ByteSub**-Transformation (Fortsetzung):

→ affinen Transformation (kommt genau **16 mal** zur Anwendung!)

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

y und x jeweils **1 Byte** lang!

---

Mathematische Beschreibung:

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b}$$

→

$$\mathbf{y} - \mathbf{b} = \mathbf{A} \cdot \mathbf{x}$$

$$\mathbf{A}^{-1}(\mathbf{y} - \mathbf{b}) = \mathbf{x}$$

→ Umkehrfunktion

$$\mathbf{x} = \mathbf{B} \cdot \mathbf{y} + \mathbf{c}$$

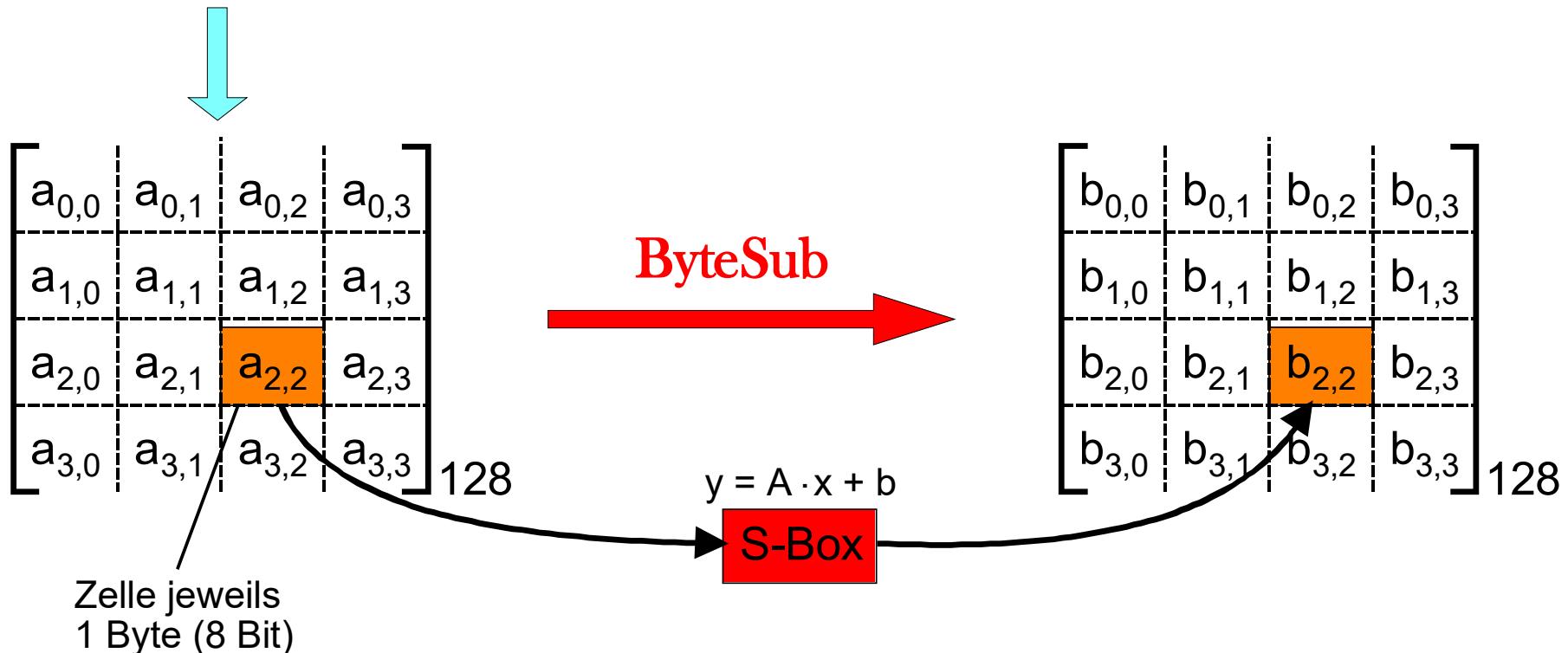
mit

$$\mathbf{B} = \mathbf{A}^{-1} \pmod{2} \quad \text{und} \quad \mathbf{c} = \mathbf{B} \cdot \mathbf{b} \pmod{2}$$

# Advanced Encryption Standard

Substitution

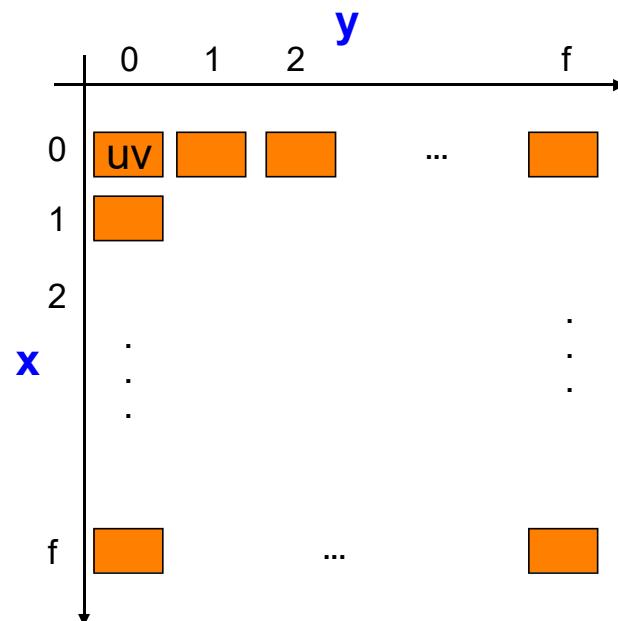
Klartext 128 Bit



# Advanced Encryption Standard

## Substitution

$$y = A \cdot x + b$$



$x = x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$  ist  
8 Bit lang  $\rightarrow$  256 Möglichkeiten

- Jedes Byte  $x$  (8 Bit) wird in Hexadezimaldarstellung als  $x = xy_{\text{hex}}$  geschrieben.
- Damit insgesamt 256 Werte  $y = UV_{\text{hex}}$ , die ebenfalls hexadezimal interpretiert werden.

→ **S-Box**, Substitutionswerte  $uv_{\text{hex}}$  für das Byte  $xy_{\text{hex}}$

# Advanced Encryption Standard

Substitution

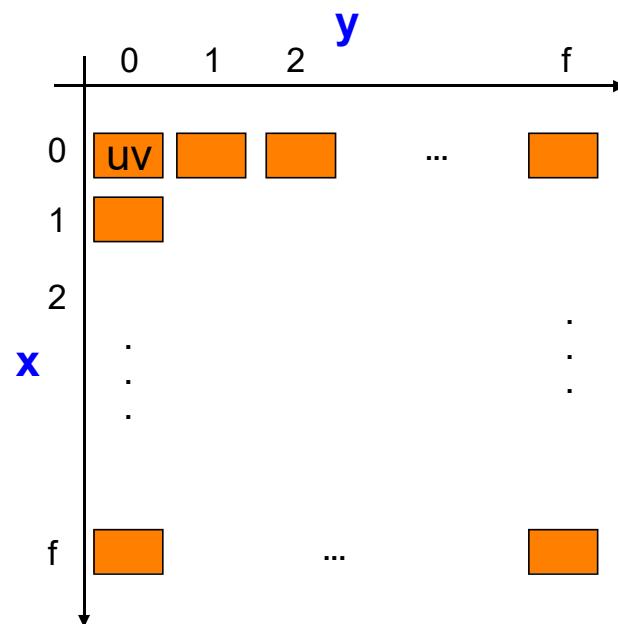
Substitutionswerte:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X		63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
		ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
0	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15	
1	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75	
2	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84	
3	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf	
4	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8	
5	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2	
6	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73	
7	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db	
8	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79	
9	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08	
a	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a	
b	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e	
c	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df	
d	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16	

# Advanced Encryption Standard

Substitution

$$x = B \cdot y + c$$



$y = y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0$  ist  
8 Bit lang  $\rightarrow$  256 Möglichkeiten

- Jedes Byte  $y$  (8 Bit) wird in Hexadezimaldarstellung als  $y = xy_{\text{hex}}$  geschrieben.
- Damit insgesamt 256 Werte  $x = UV_{\text{hex}}$ , die ebenfalls hexadezimal interpretiert werden.

→ **Inverse S-Box**, Substitutionswerte  $uv_{\text{hex}}$  für das Byte  $xy_{\text{hex}}$

# Advanced Encryption Standard

Substitution

Substitutionswerte:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X		52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
		7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
0	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e	
1	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25	
2	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92	
3	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84	
4	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06	
5	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b	
6	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73	
7	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e	
8	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b	
9	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4	
a	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f	
b	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef	
c	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61	
d	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d	
e	f																

### Die **ShiftRow**-Transformation:

- Diese Transformation verschiebt die Zeilen 1 bis 3 der Zustandsmatrix **zyklisch nach links**.
  - Die Verschiebung hängt von der Blockgröße **b** ab.
  - Zeile 0 wird nicht verändert.
  - Zeile 1 wird im allgemeinen um  $c_1$  Bytes, Zeile 2 und  $c_2$  Bytes und Zeile 3 um  $c_3$  Bytes verschoben.
  - Zum Dechiffrieren erhält man die **inverse** Transformation durch Ausführen der zyklischen Verschiebung nach **rechts**.
-

# Advanced Encryption Standard

Verschiebung

---

Die **ShiftRow**-Transformation (Fortsetzung):

Verschiebungen	Blocklänge		
	<b>b = 128</b>	<b>b = 192</b>	<b>b = 256</b>
C1	1	1	1
C2	2	2	3
C3	3	3	4

hier: Im Falle von **AES** gilt **b = 128**.

---

### Die **ShiftRow**-Transformation (Fortsetzung):

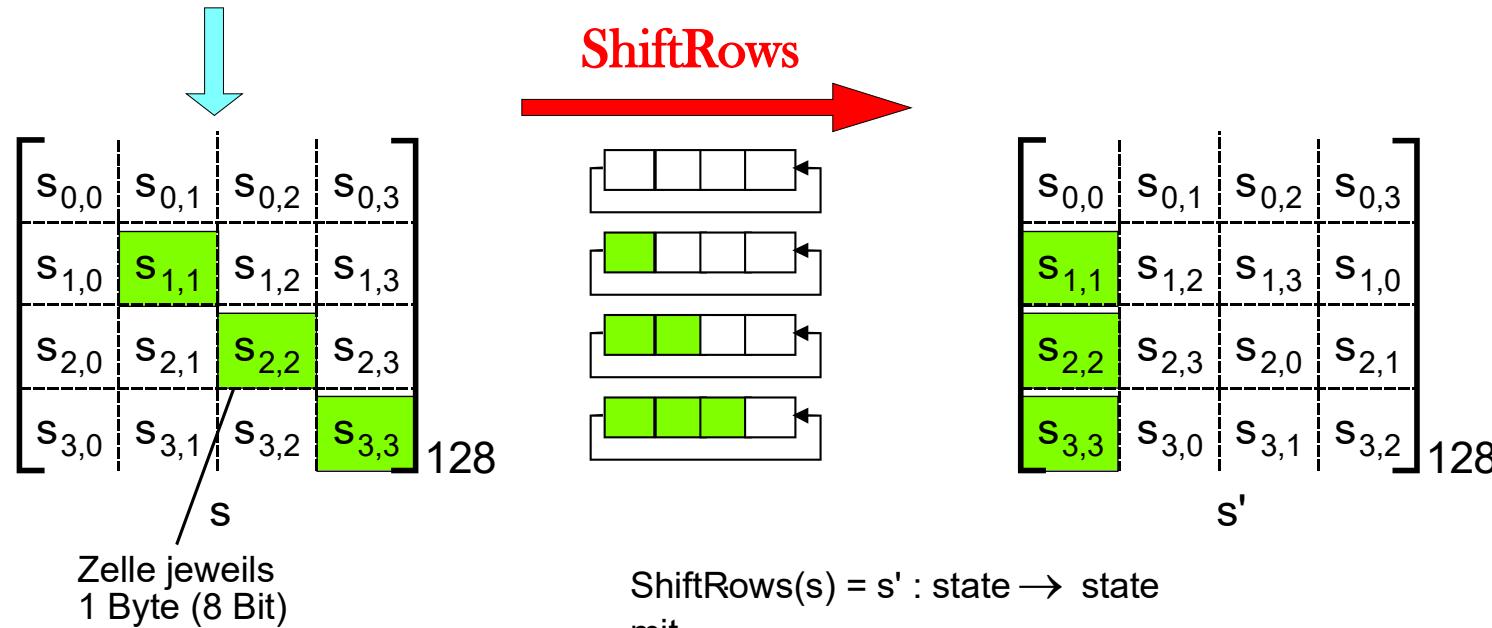
- Sei **s** ein **state**, also nach vorangegangener Substitution ein **teiltransformierter Klartext**.
- Schreibe **s** als **Zustandsmatrix** mit **4 Zeilen** und **4 Spalten**. Die Matrixeinträge sind jeweils Bytes.
- Verschiebe die letzten drei Zeilen der Zustandsmatrix **s** zyklisch nach links.

$$\text{LeftShift}(\text{Zeile } i) = i \text{ für } i \in \{0, 1, 2, 3\}$$

- Es ergibt sich eine Abbildung **state → state**, die bei Anwendung in mehreren Runden für eine **hohe Diffusion** sorgt.
-

### Die Wirkung der **ShiftRow**-Transformation:

Teiltransformierter Klartext 128 Bit



## Die **MixColumn**-Transformation:

- Diese Transformation wirkt auf verschiedene **Spalten**  $k \in \{0, 1, 2, 3\}$  der **Zustandmatrix**  $s$  und sorgt dort jeweils für eine **Vermischung**.
  - Die **Elemente** der vier Spaltenvektoren  $s_k = (s_{0,k}, s_{1,k}, s_{2,k}, s_{3,k})$  sind **1 Byte** lang und werden als Hexadezimalzahl  $(xy)_{hex}$  interpretiert.
  - Ferner werden die Elemente  $s_{0,k}, s_{1,k}, \dots, s_{3,k}$  einer jeden Spalte  $s_k$  als Koeffizienten eines Polynoms in  $\text{GF}(2^8)[x] / (x^4 + 1)$  aufgefasst:  
$$s_k(x) = s_{3,k} \cdot x^3 + s_{2,k} \cdot x^2 + s_{1,k} \cdot x + s_{0,k} \in \text{GF}(2^8)[x] / (x^4 + 1)$$
  - Die Transformation **MixColumn** setzt nun  
$$s_k(x) \leftarrow (s_k(x) \cdot a(x)) \bmod (x^4 + 1), \quad 0 \leq k \leq 3,$$
wobei  $a(x)$  das feste Polynom  $(03) \cdot x^3 + (01) \cdot x^2 + (01) \cdot x + (02)$  ist.
-

### Grundlagen:

- Speziell für den **AES** wählen wir einen **endlichen Körper  $GF(2^8)$**  der **Charakteristik 2** mit dem zugehörigen Polynom  $m(x)$  vom **Grad 8**.  
$$m(x) = x^8 + x^4 + x^3 + x + 1$$
- Es ist also:

$$GF(2^8) = \{a(x) \mid a_7 \cdot x^7 + \dots + a_1 \cdot x + a_0 \\ \text{für } a_i \in \mathbb{Z}_2 = \{0, 1\}, i = 0, 1, \dots, 7\}$$

- Die Elemente werden auch als Binärstrings  $a_7 a_6 \dots a_0$  (bzw. als Bytes) oder in hexadezimaler Notation  $xy_{hex} = (xy)$  geschrieben.

**Beispiel:**  $x^7 + x^6 + 1 = 1100\ 0001 = c1_{hex} = (c1)$

### Grundlagen (Fortsetzung):

- Die Addition  $\oplus$  in  $\text{GF}(2^8)$  erfolgt komponentenweise (XOR) und die Multiplikation  $\bullet$  wird in  $\text{GF}(2^8)$  modulo  $m(x)$  durchgeführt.

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

**Beispiel:**  $(57) \bullet (83) = 0101\ 0111 \bullet 1000\ 0011 = ?$

→

$$\begin{aligned} & (x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1) \\ &= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) \\ &= (x^7 + x^6 + 1) \end{aligned}$$

also

$$= \boxed{(c1)}$$

### Die **MixColumn**-Transformation (Fortsetzung):

- Dies kann wiederum als **lineare Transformation** ( $\rightarrow$  Matrixmultiplikation  $\mathbf{s}' = \mathbf{A} \cdot \mathbf{s}$ ) in  $(\text{GF}(2^8))^4$  über dem Körper  $\text{GF}(2^8)$  für  $k \in \{0, 1, 2, 3\}$  beschrieben werden:

$$\begin{pmatrix} s_{0,k} \\ s_{1,k} \\ s_{2,k} \\ s_{3,k} \end{pmatrix} \leftarrow \begin{pmatrix} (02) & (03) & (01) & (01) \\ (01) & (02) & (03) & (01) \\ (01) & (01) & (02) & (03) \\ (03) & (01) & (01) & (02) \end{pmatrix} \bullet \begin{pmatrix} s_{0,k} \\ s_{1,k} \\ s_{2,k} \\ s_{3,k} \end{pmatrix}$$

- Diese Transformation sorgt somit für eine **gute Diffusion** innerhalb der Spalten von **state**.
  - Auch die **MixColumn**-Transformation ist **invertierbar**.
-

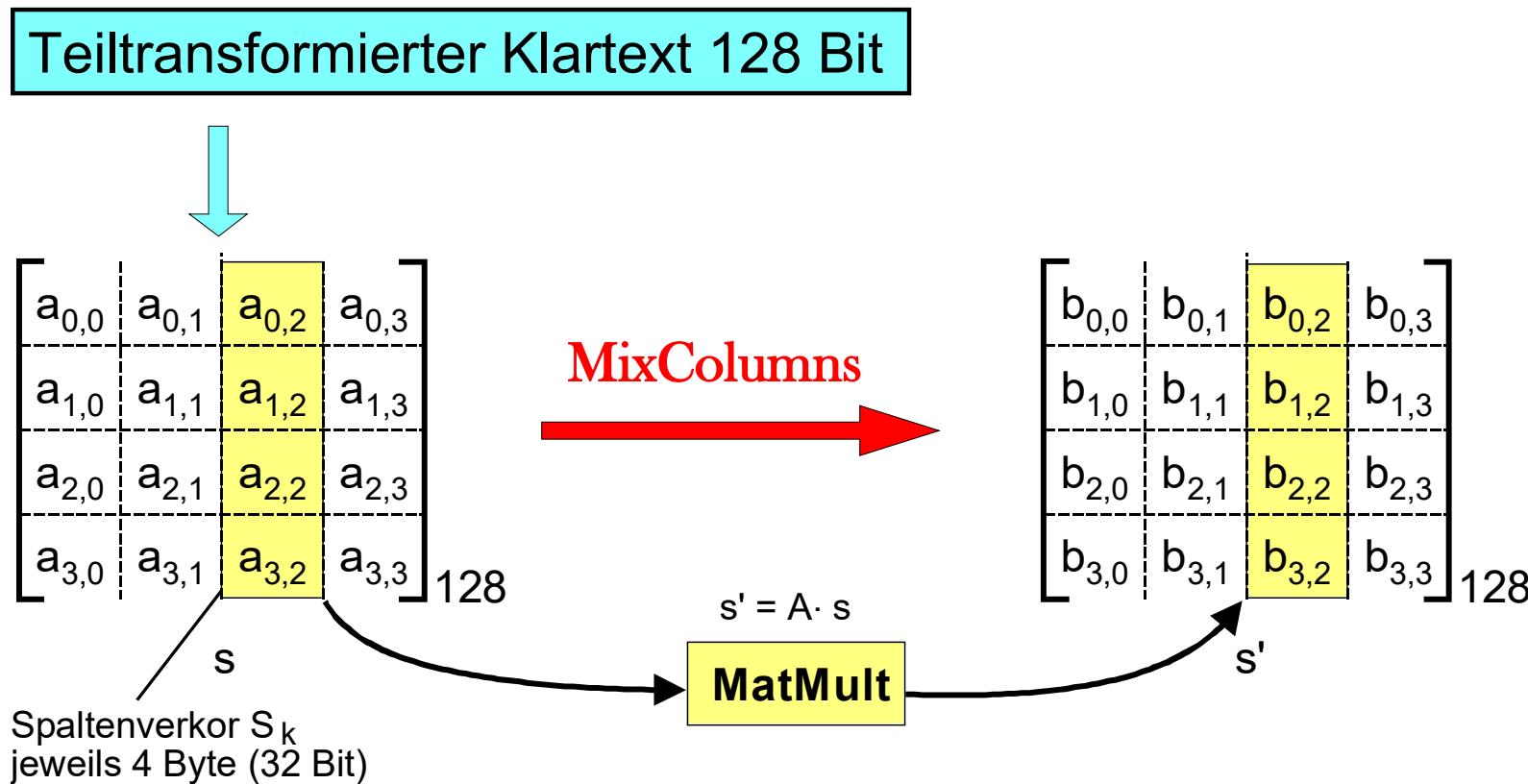
### Anmerkung zur Invertierbarkeit:

- AES verwendet Polynome über dem Körper  $\text{GF}(2^8)$ , aber nur solche der Form  $a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$  für  $a_i \in \text{GF}(2^8)$ ,  $i = 0, 1, 2, 3$ .
  - Daher müssen Reduktionen **modulo** einem Polynom über  $\text{GF}(2^8)$  vom **Grad 4** durchgeführt werden. Es wird  $x^4 + 1 \in \text{GF}(2^8)$  gewählt.
  - Wir bilden also den Ring (und keinen Körper!!!)  $\text{GF}(2^8)[x] / (x^4 + 1)$ .
  - Somit muss ein Element dieses so gebildeten Rings nicht unbedingt eine **Inverse** besitzen.
  - Durch die spezielle Wahl  $a(x) = (03) \cdot x^3 + (01) \cdot x^2 + (01) \cdot x + (02)$  existiert jedoch das **Inverse**  $a^{-1}(x) = (0b) \cdot x^3 + (0d) \cdot x^2 + (09) \cdot x + (0e)$ .
-

# Advanced Encryption Standard

Multiplikation

## Die Wirkung der **MixColumn**-Transformation:



## AES-Funktionen Cipher und KeyExpansion:

### Der **KeyExpansion**-Algorithmus:

- Der Chiffrierschlüssel **K** wird beim AES durch Schlüsselexpansion so aufgeweitet, dass sich **r + 1** Teilschlüssel mit je **b** Bits bilden.
- Bei einer Blocklänge von **b = 128 Bit** und **zwölf Runden** werden somit insgesamt  $128 \cdot 13 = 1664$  Schlüsselbits generiert.

### Der **Cipher**-Algorithmus:

- Eingabe ist der **Klartextblock (128 Bit)** und der expandierte Schlüssel (1664 Bit).
  - Ausgabe ist nach **zwölf Runden** der **Chiffertextblock (128 Bit)**.
-

## AES-Dechiffrierfunktion:

### Der **InvCipher**-Algorithmus:

- Die Entschlüsselung des AES wird von der Funktion **InvCipher** besorgt.
  - Man erhält die Dechiffrierfunktion dadurch, dass wir die Reihenfolge der zuvor betrachteten Transformationen umdrehen und dabei – außen für Berechnung der Teilschlüssel – die jeweiligen inversen Transformationen betrachten.
  - Des weiteren werden die Teilschlüssel in der umgekehrten Reihenfolge benutzt.
    - Ausgabe ist nach **zwölf Runden** der Klartextblock (128 Bit).
-

### Grundlegende Konstruktionsprinzipien

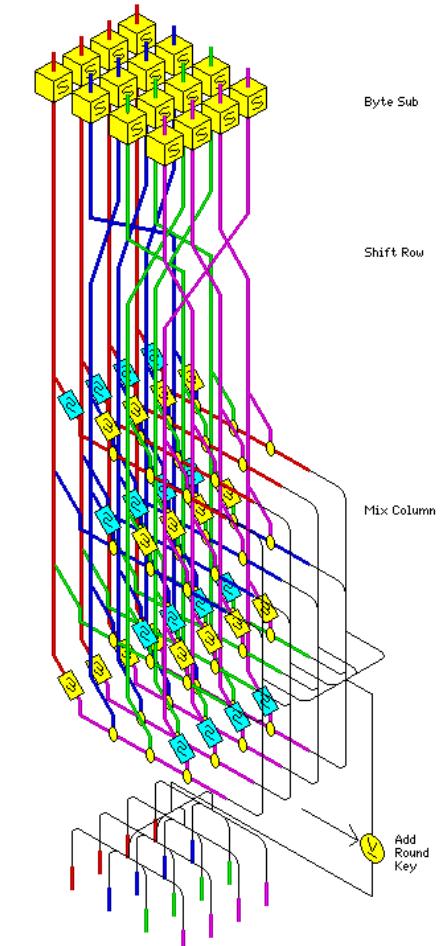
(Claude Elwood Shannon, Begründer der Informationstheorie)

#### Konfusion:

Auflösen von statistischen Strukturen (z. B. Buchstabenhäufigkeiten) eines Klartextes beim Verschlüsseln, d. h. jedes Ciphertextzeichen sollte von möglichst vielen Klartextzeichen abhängig sein.

#### Diffusion:

Verschleierung des Zusammenhangs zwischen Klartext und Geheimtext, d. h. bei einer Änderung von **einem** Schlüsselbit oder **einem** Klartextbit sollte sich 50 % des Geheimtextes ändern.



### Kap. 4: Symmetrische Verfahren und moderne Blockchiffren

#### Teil 2: Betriebsmodi

- ECB
- CBC
- CFB
- OFB

## ECB – Electronic Code Book

Wie bei einem Wörterbuch gibt es zu  $2^N$  möglichen Klartextstrings  $2^N$  Schlüsselstrings und umgekehrt.

### Eigenschaften:

- Jeweils ein Block von N Bit wird **unabhängig** von anderen Blöcken verschlüsselt.
- Reihenfolge kann verändert werden, ohne daß die Entschlüsselung davon beeinflußt wird.
- Gleicher Klartext ergibt gleichen Schlüsseltext (sicherheitskritisch → keine identische Blöcke!).

### Fehlerfortpflanzung:

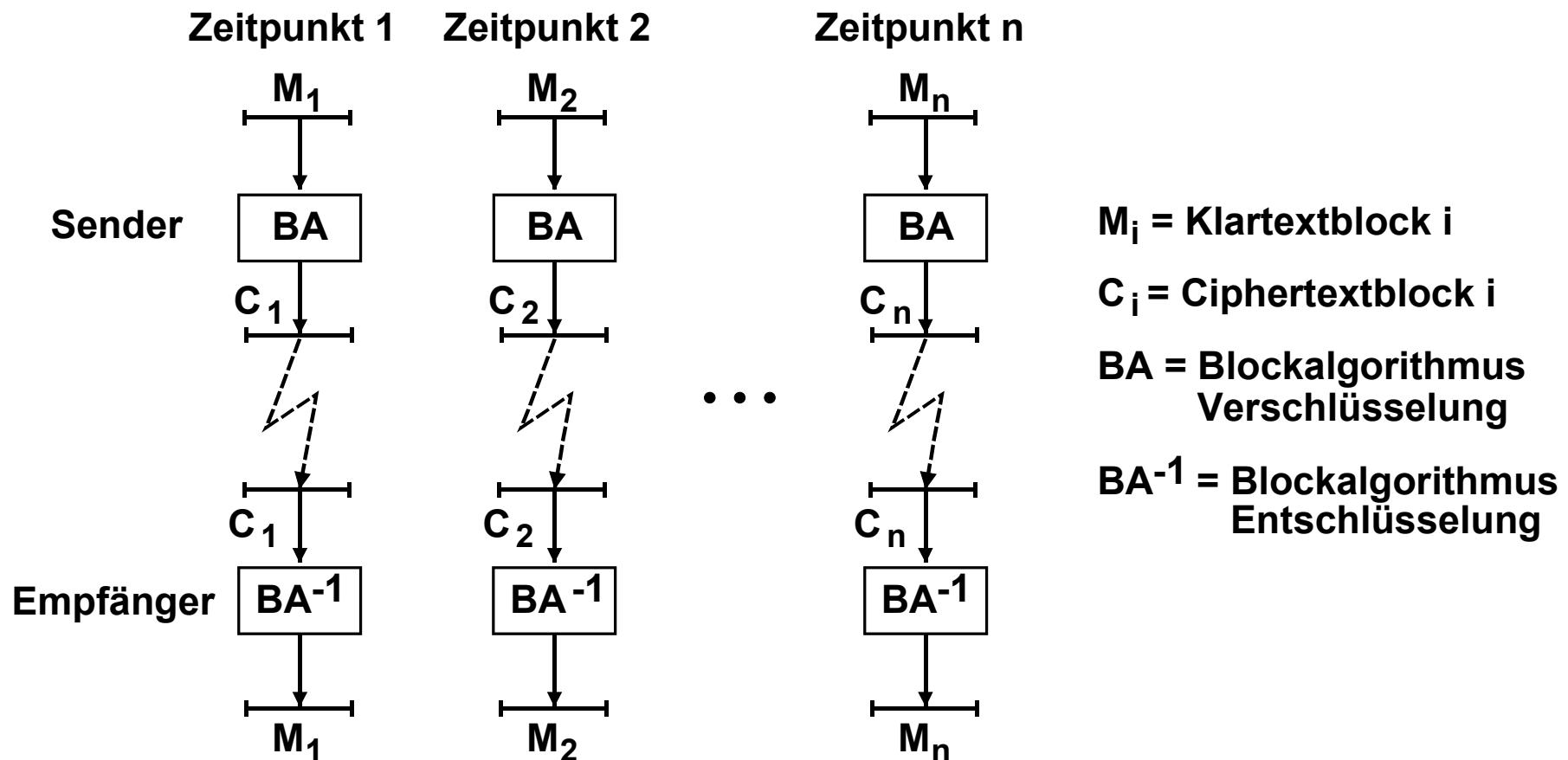
- Bitfehler oder Bitgruppenfehler eines Schlüsseltextblockes verursachen einen fehlerhaften Klartextblock (mindestens 50 % aller Bits im Outputblock betroffen).

### Synchronisation:

- Wenn Blockgrenzen während der Übertragung verlorengehen (z. B. Bitschlupf), geht die Synchronisation zwischen Ver- und Entschlüsselung verloren (d. h. alle Folgeblöcke werden nicht mehr korrekt entschlüsselt).
-

# Betriebsmodus

ECB (2)



## CBC – Cipher Block Chaining

Im Gegensatz zum EBC-Modus erfolgt nun eine **Verkettung der Blöcke.**

Eigenschaften:

- Verkettung bewirkt, daß der Chiffretext von dem ganzen vorangegangenen Klartext und dem IV abhängt.
- Die Blöcke können daher nicht umgeordnet werden.
- Der IV verhindert, daß gleicher Klartext gleichen Chiffretext ergibt.

Fehlerfortpflanzung:

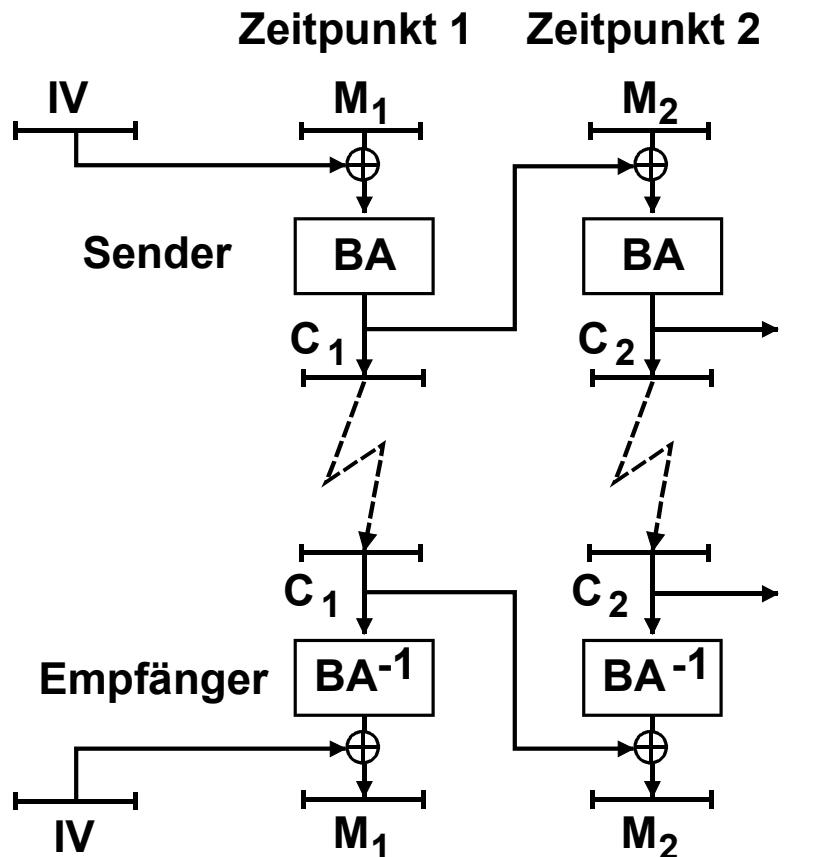
- Wenn in einem Block des Chiffretextes ein Bit- oder Bitgruppenfehler auftritt, wird die Entschlüsselung des betreffenden und des nachfolgenden Blockes gestört (→ Fehlerfortpflanzung).

Synchronisation:

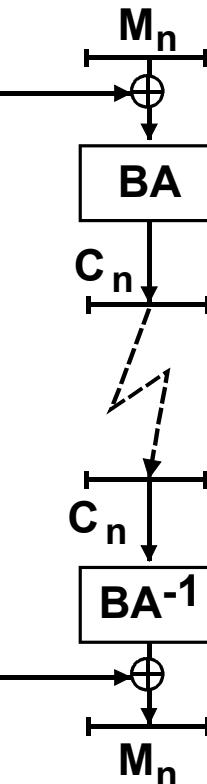
- Wenn die Bitgrenzen z. B. durch Bitschlupf verlorengehen, geht auch die Synchronisation zwischen Ver- und Entschlüsselung verloren (→ Neuinitialisierung notwendig).
-

# Betriebsmodus

CBC (2)



Zeitpunkt n



$M_i$  = Klartextblock i

$C_i$  = Ciphertextblock i

BA = Blockalgorithmus  
Verschlüsselung

$BA^{-1}$  = Blockalgorithmus  
Entschlüsselung

IV = Initialisierungsvektor

### CFB – Cipher FeedBack

Sowohl sender- als auch empfängerseitig arbeitet die Blockverschlüsselung im Verschlüsselungsmodus und erzeugt eine pseudozufällige Bitfolge E, die modulo 2 (XOR) zu den Klartextzeichen bzw. Schlüsseltextzeichen addiert wird (if  $j = 1 \Rightarrow$  Bitstromverschlüsselung; if  $j = N \Rightarrow$  verkettete Blockverschlüsselung).

#### Eigenschaften:

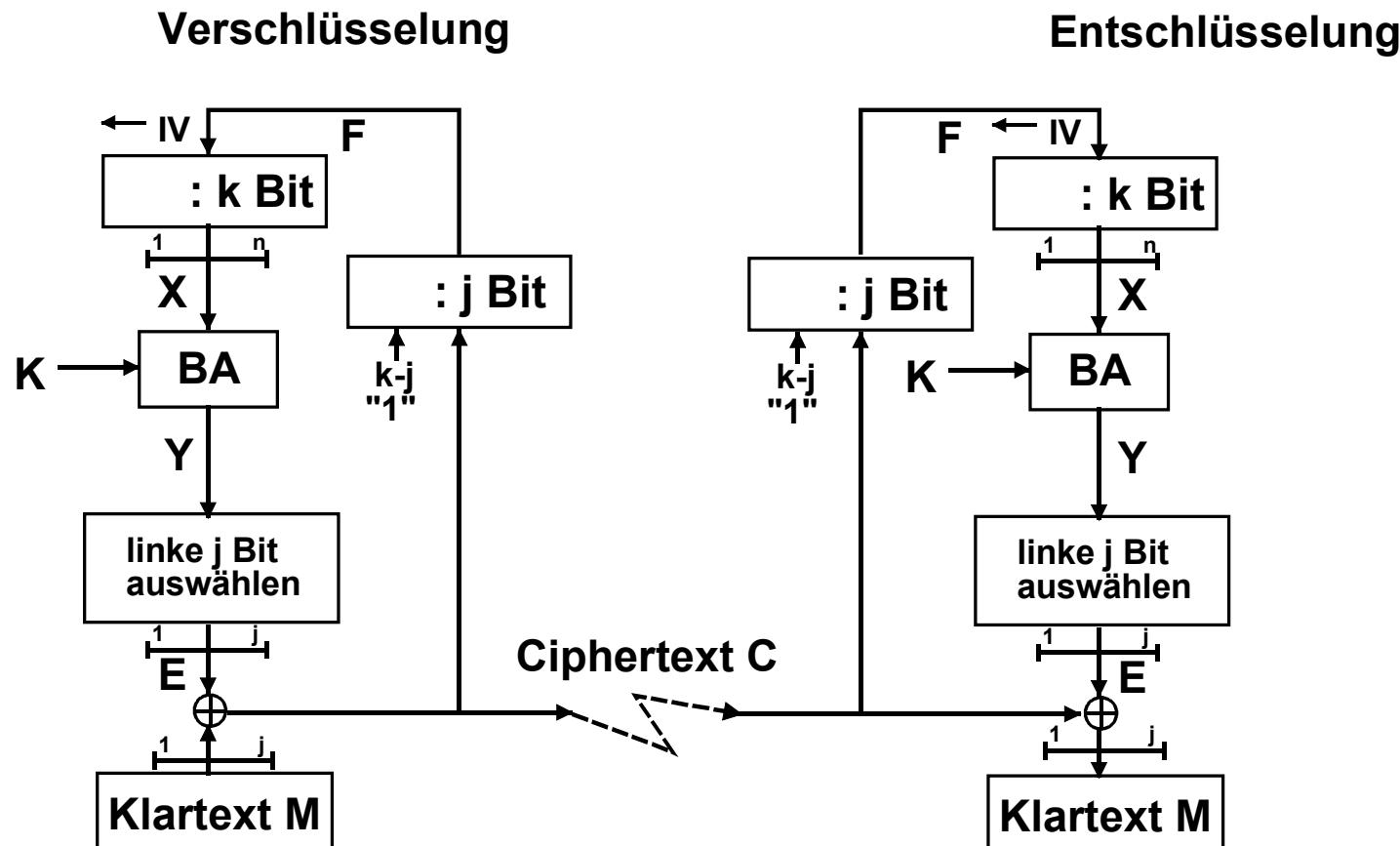
- Wenn implementierter Blockalgorithmus BA einen Durchsatz von T Bit/s bietet, so leistet der CFB-Modus effektiv nur noch  $T * j/N$  Bit/s ( $j$  = Länge der Klartextvariablen).
- Falls gleicher Schlüssel und IV verwendet wird, produziert CFB-Modus bei gleichem Klartext gleichen Chiffretext.

#### Fehlerfortpflanzung:

- Falls im CFB-Modus eine Chiffretextvariable C gestört wird, so werden solange falsche Klartextzeichen generiert, bis fehlerhafte Bits beim Empfänger herausgefiltert wurden.

#### Synchronisation:

- Wenn Variablengrenze verloren geht, sind Sender und Empfänger solange außer Synchronisation, bis Blockgrenzen wieder erreicht, d. h. CBF-Modus ist **selbstsynchronisierend**.
-



## OFB – Output FeedBack

Im Gegensatz zum CFB-Modus werden beim OFB-Modus nicht die Chiffretextvariablen C, sondern der Output der Blockverschlüsselung BA als Input für die nächste Verschlüsselungsoperation zurückgeführt.

Eigenschaften:

- Der erzeugte Schlüsselstrom hängt nicht vom Klartext ab.
- Da keine Verkettung erfolgt, ist der OFB durch spezifische Angriffe gefährdet.
- Gleicher Klartext ergibt gleichen Chiffretext, falls gleicher Schlüssel und IV verwendet wird.

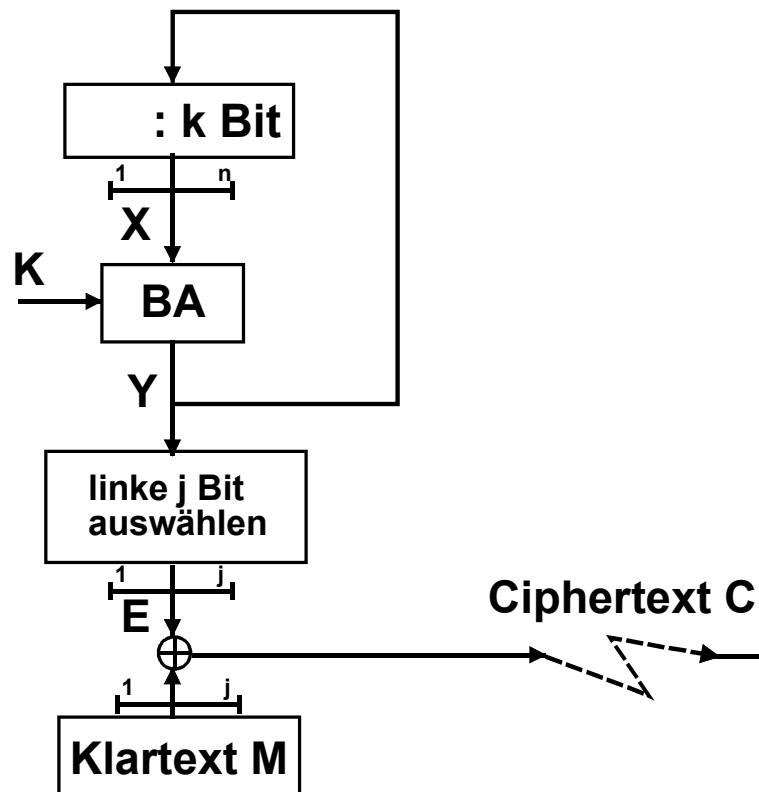
Fehlerfortpflanzung:

- Es gibt keine Fehlerfortpflanzung, solange die Ver- und Entschlüsselung synchron erfolgt.
- Jedes fehlerhafte Bit im Schlüsseltext ergibt ein fehlerhaftes Bit im Klartext.

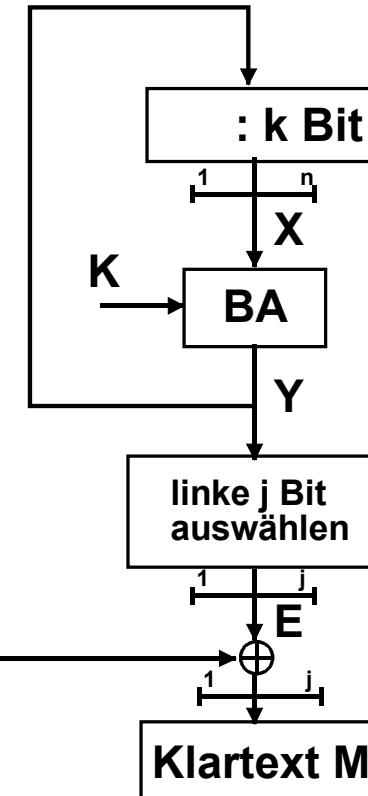
Synchronisation:

- Der OFB-Modus ist nicht selbstsynchronisierend.
  - Tritt z. B. Bitschlupf auf, muß System neu initialisiert werden.
-

### Verschlüsselung



### Entschlüsselung



### Kap. 4: Symmetrische Verfahren und moderne Blockchiffren

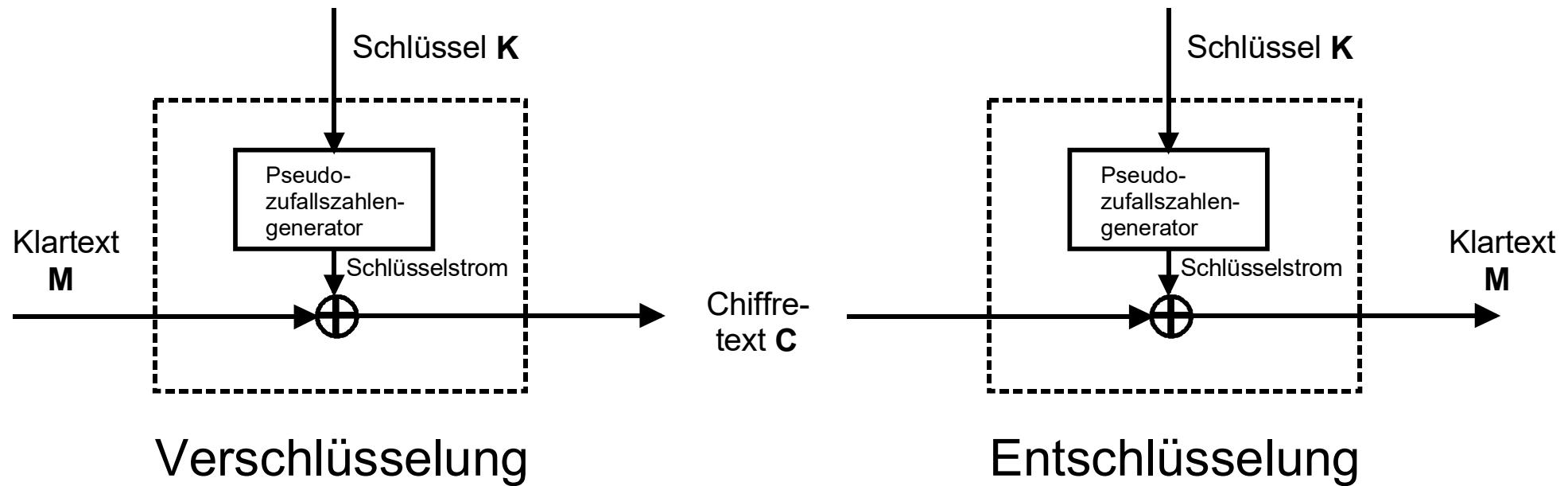
#### Teil 3: Symmetrische Bitstromverschlüsselung

- Bitstromverschlüsselung mittels XOR-Algorithmus
- One-Time-Pad und perfekte Sicherheit

- Symmetrische Verschlüsselungsverfahren können in Blockalgorithmen und Bitstromverschlüsselung unterschieden werden.
  - Bei der Bitstromverschlüsselung wird der Klartext Bit-für-Bit verschlüsselt.
  - Der angewandte geheime Schlüssel(strom) wird in einem Pseudo-Zufallszahlgenerator eingegeben oder der Anfangszustand des Pseudo-Zufallszahlengenerators wird mit dem geheimen Schlüssel vorbelegt (Initialisierung).
  - Die Periode der vom Pseudo-Zufallszahlengenerator erzeugten Folge muß größer als die zu verschlüsselnde Nachricht sein (sog. one time pad).
  - Die Output-Bits werden auf der Verschlüsselungsseite mit dem Klartext durch XOR verknüpft.
  - Auf der Entschlüsselungsseite wird der Pseudo-Zufallszahlengenerator mit demselben geheimen Schlüssel vorbelegt, und die Output-Folge mit dem Schlüsseltext wiederum durch XOR verknüpft, so daß man wieder den ursprünglichen Klartext erhält.
  - Ein besonderes Problem stellt der Austausch des geheimen Schlüssels dar.
-

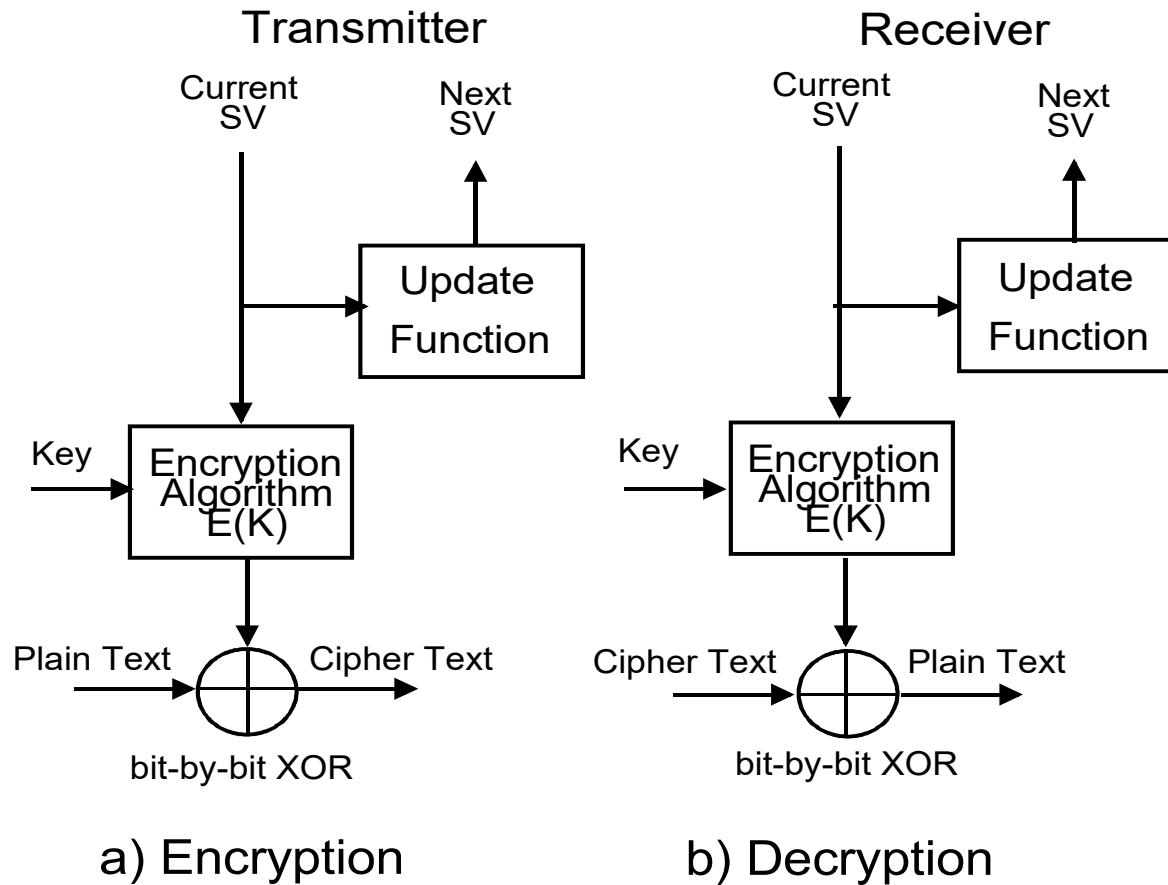
# Bitstromverschlüsselung

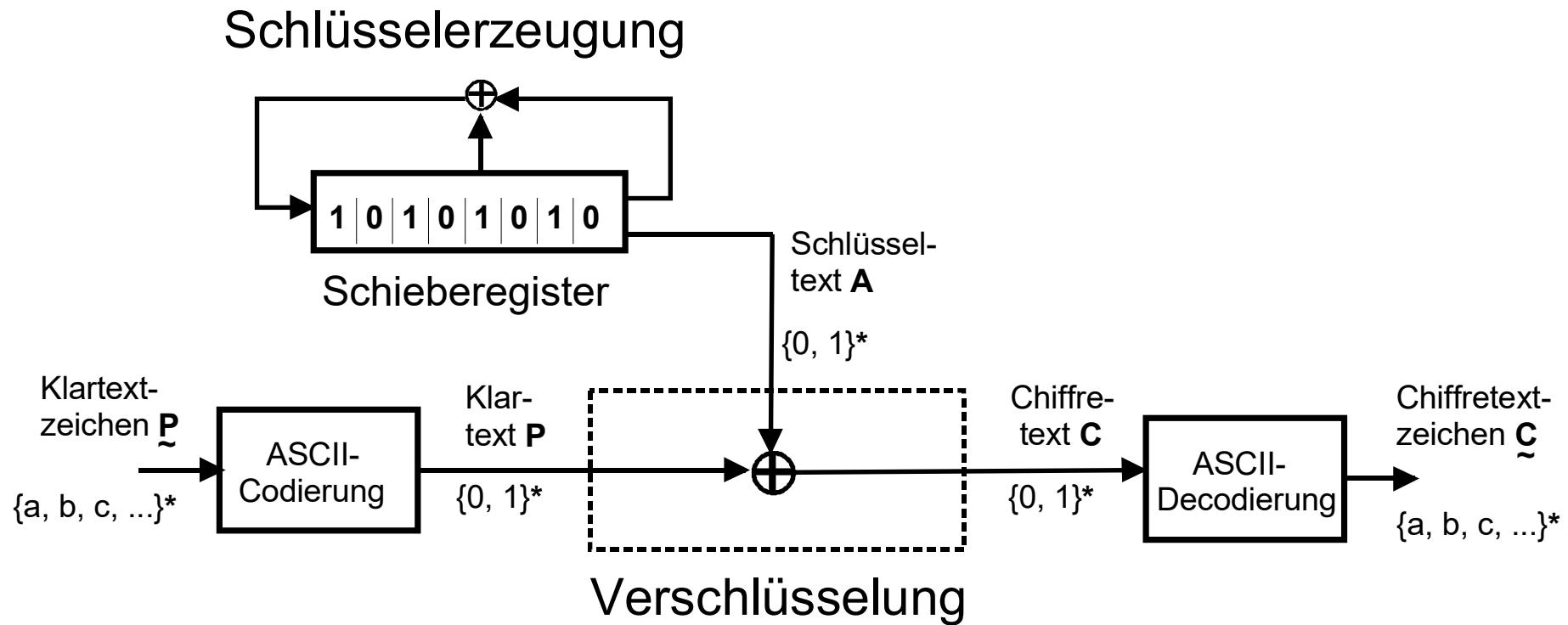
Realisierung (1)



# Bitstromverschlüsselung

Realisierung (2)





# Bitstromverschlüsselung

## XOR-Algorithmus

---

Klartext:  $a_1, a_2, \dots, a_n$        $a_i, k_i, c_i \in \{0, 1\}$   
Schlüssel:  $k_1, k_2, \dots, k_n$       mit  
Geheimtext:  $c_1, c_2, \dots, c_n$        $i = 1, 2, \dots, n$

### Rechenvorschrift: Binäre Addition modulo 2

$$1 \oplus 1 = 0$$

$$a_i \oplus k_i := c_i$$

$$0 \oplus 0 = 0$$

$$1 \oplus 0 = 1$$

Entschlüsselung:

$$0 \oplus 1 = 1$$

$$c_i \oplus k_i := a_i \oplus k_i \oplus k_i$$

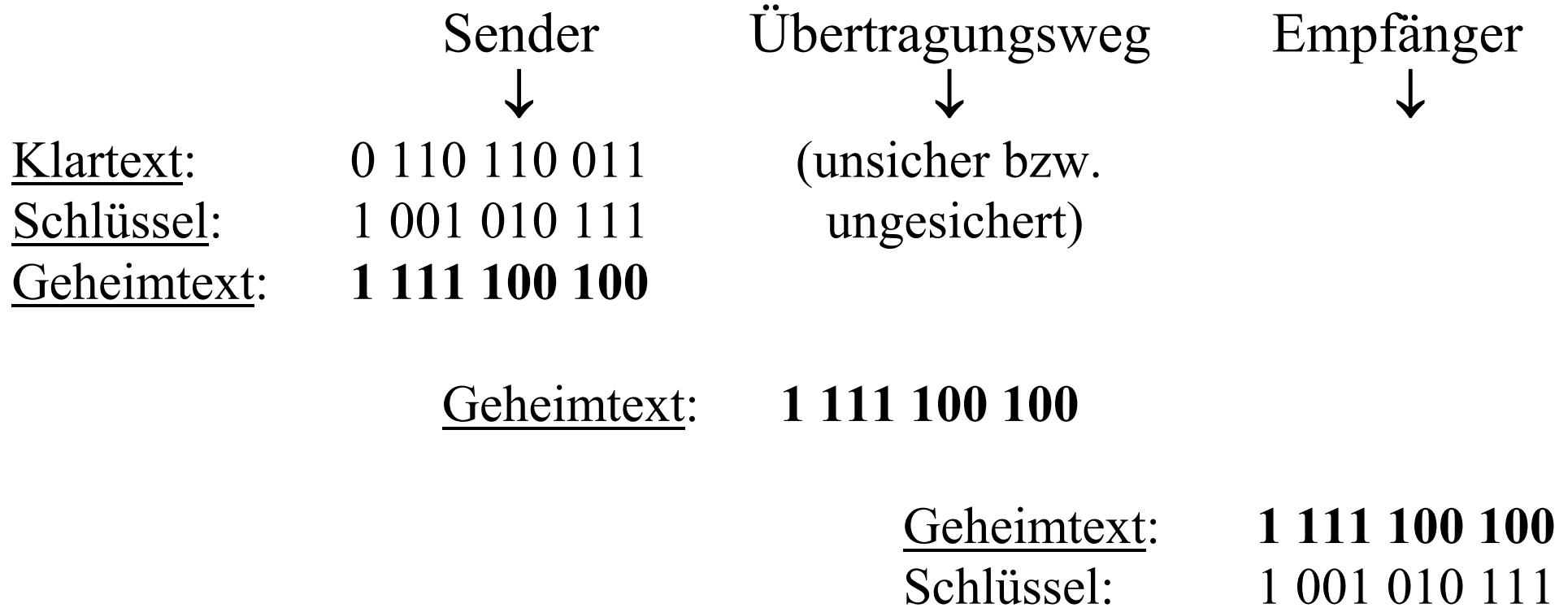
### Eigenschaften:

- alle Folgen der Länge  $n$  mit derselben Wahrscheinlichkeit  $:= a_i$
  - ohne Kenntnis von  $k_i$  lässt sich nicht auf  $a_i$  schließen
  - mit Kenntnis von  $k_i$  lässt sich  $a_i$  aus  $c_i$  rekonstruieren
  - gleicher Schlüssel auf beiden Seiten  $\rightarrow$  (geheimer) Schlüsselaustausch notwendig!
-

# Bitstromverschlüsselung

---

Zahlenbeispiel



- **56 bit DES-Schlüssel geknackt**
  - bis zu 7 Milliarden Schlüssel pro Sekunde ausprobiert
  - nach etwa 25 % der möglichen 72 Billionen Schlüssel wurde der richtige gefunden
  - gerechnet wurde während der Wartezeiten gewöhnlicher Rechner von tausenden Leuten, die sich nie gesehen haben
- **RSA: "sicher" heißt nur "relativ sicher"**
  - 100 Pentium-Prozessoren mit 100 MHz Taktfrequenz brauchen rund 1 Jahr an Rechenzeit, um einen 428-Bit-RSA-Schlüssel zu knacken
  - generell kann kein bekanntes Chiffrierverfahren<sup>1</sup> als (mathe. beweisbar) vollkommen sicher erachtet werden
  - Sicherheit abhängig von Rechnerleistung, Wissen, Gelegenheit etc.

---

<sup>1</sup> Einzige Ausnahme: sog. One-time-pad

- Das One-Time-Pad wurde 1917 von Major J. Mauborgne und G. Vernam von AT&T erfunden.
- Sei  $A = \{0, 1\}$  ein Alphabet und  $z, r \in A$ .  
Ein Klartextbit  $z$  wird mit dem Zufallbit  $r$  des Schlüssels chiffriert durch die Vorschrift (Vigenere-Chiffre):

$$z \rightarrow (z + r) \bmod 2 = z \text{ XOR } r = z \oplus r$$

- Das One-Time-Pad gehört damit eindeutig zur Klasse der **Stromchiffren** (bitweise XOR-Verknüpfung).
  - Als eines der wenigen **perfekten** Chiffriersysteme ist es gleichzeitig gemäß der vorangestellten Definition **uneingeschränkt sicher**.
-

### Definition:

Ein Chiffriersystem heißt **perfekt**, wenn bei beliebigem Klartext  $\mathbf{M}$  und beliebigem Chiffretext  $\mathbf{C}$  die a-priori-Wahrscheinlichkeit  $P(\mathbf{M})$  gleich der bedingten Wahrscheinlichkeit (a-posteriori-Wahrscheinlichkeit)  $P(\mathbf{M} | \mathbf{C})$  ist, d. h.

$$P(\mathbf{M} | \mathbf{C}) = P(\mathbf{M}) \Leftrightarrow \text{perfektes Chiffriersystem} \quad (1)$$

Mit der Definition der bedingten Wahrscheinlichkeit

$$P(\mathbf{M} | \mathbf{C}) = P(\mathbf{M} \wedge \mathbf{C}) / P(\mathbf{C}) \quad (2)$$

folgt aus (1):

$$P(\mathbf{M} \wedge \mathbf{C}) = P(\mathbf{M}) \cdot P(\mathbf{C}) \Leftrightarrow \begin{array}{l} \text{statistisch unabhängig} \\ \text{sog. Produktregel} \end{array} \quad (3)$$

d. h. um **perfekte Sicherheit** zu garantieren, müssen ein beliebiger Klartext  $\mathbf{M}$  und der zugehörige Chiffretext  $\mathbf{C}$  **statistisch unabhängig** sein.

---

Sei  $\mathbf{M}$  die Menge  $\forall$  Klartexte  $\mathbf{M}$  der Länge  $n$  ,  
 $\mathbf{C}$  die Menge  $\forall$  Chiffretexte  $\mathbf{C}$  der Länge  $n$  und  
 $\mathbf{K}$  die Menge  $\forall$  Schlüsseltexte  $\mathbf{K}$  der Länge  $n$  .

Bei einem One-Time-Pad gilt:

$$m := |\mathbf{M}| = |\mathbf{C}| = |\mathbf{K}| = 2^n , \quad (4)$$

denn alle drei Mengen bestehen aus Texten der Länge  $n$  über einem vorgegebenen Alphabet  $A = \{0, 1\}$ .

Für einen beliebig vorgegebenen Klartext  $\mathbf{M}$  wird jeder Chiffertext  $\mathbf{C}$  mit der gleichen Wahrscheinlichkeit erzeugt:

$$P(\mathbf{C}) = 1 / m = 2^{-n} \quad (5)$$

---

Da es genau  $m$  Schlüssel gibt, gibt es auch genau  $m$  unterschiedliche Chiffretexte  $C$  zu jedem Klartext  $M$ , also

$$P(C | M) = 1 / m = 2^{-n} \quad (6)$$

Wendet man (2) auch auf  $P(C | M)$  an, so erhält man die sog. **Bayes`che Formel**:

$$P(M | C) \cdot P(C) = P(C | M) \cdot P(M) \quad (7)$$

Einsetzen von (5) und (6) in (7) ergibt:

$$P(M | C) \cdot 1 / m = 1 / m \cdot P(M)$$

Hieraus folgt:

$$P(M | C) = P(M) \Leftrightarrow \text{One-Time-Pad} = \text{perfektes Chiffriersystem}, \quad (8)$$

gleichzeitig **uneingeschränkt sicher**, da es keine Möglichkeit gibt, aus einem beliebig langen Chiffretext  $C$  – ohne Kenntnis des Schlüssels – auf den Klartext  $M$  zu schließen!

---

# Bitstromverschlüsselung

mittels Coset-Muster

---

```
/* Datum: 19.07.2002 */  
/* Autor: Bernhard Geib */  
/* Funktion: Verschluesselung mit Coset-Muster 01010101 */  
  
#include <stdio.h>  
  
int main (void)  
{  int c;  
  
    c = getchar();  
    while (c != EOF) {  
        if (c != '\n') {  
            c = c ^ 85; /* hier eine Zufallszahl verwenden */  
            printf ("%c", c);  
            c = getchar();  
        }  
    }  
    return 0;  
}
```

# Bitstromverschlüsselung

mittels selbstinverser Chiffre (1)

---

```
/* Datum: 17.07.2002 */  
/* Autor: Bernhard Geib */  
/* Funktion: Bijektive, selbstinverse Chiffre */  
  
#include <stdio.h>  
#include <string.h>  
  
void verschluessler(char *text, const char *geheimnis)  
{  
    /* Annahme: text und geheimnis zeigen jeweils auf */  
    /* string mit Zeichen aus dem Bereich 32 .. 95 */  
  
    int i ;  
    int lt = strlen(text) ;  
    int lg = strlen(geheimnis) ;
```

# Bitstromverschlüsselung

mittels selbstinverser Chiffre (2)

---

```
// Fortsetzung Verschluessler

for (i=0; i < lt; ++i)
{
    char c = text[i] - ' ';
    char key = geheimnis[i % lg] - ' ';

    c = c ^ key ;
    text[i] = c + ' ';
}

} // Ende Verschluessler
```

# Bitstromverschlüsselung

mittels selbstinverser Chiffre (3)

---

```
// Zugehöriges Hauptprogramm

int main( void )
{
    char text[] = "Dies ist der gegebene Klartext" ;
    char geheimnis[] = "GEHEIM" ;

    printf("Vor der Verschlüsselung: %s\n", text) ;
    verschluessler(text, geheimnis) ;
    printf("Nach der Verschlüsselung: %s\n", text) ;
    verschluessler(text, geheimnis) ;
    printf("Nach der Entschlüsselung: %s\n", text) ;

    return 0 ;
}
```

---

# **Security**

## **- LV 4120 und 7240 -**

**Einwegfunktionen**

- Idee und Konzept der Einwegfunktion
- Einfache Realisierungsmöglichkeit für eine Einwegfunktion
- Message Digest (MD) und Message Authentication Code (MAC)
- Hashfunktionen mit blockorientierten Verschlüsselungsalgorithmen
- Secure Hash Algorithm SHA-1

### Definition:

Es seien  $X$  und  $Y$  zwei endliche, nichtleere Mengen. Eine Funktion  $f : x \rightarrow y$  heißt Einwegfunktion, wenn für alle  $x \in X$  der **Funktionswert**  $f(x)$  leicht berechnet werden kann, es aber de facto unmöglich ist, für ein gegebenes  $y \in Y$  ein  $x \in X$  mit  $f(x) = y$  zu finden. Der Wert  $x$  muss dabei nicht eindeutig bestimmt sein.

- $X$  und  $Y$  müssen eine genügend große Kardinalität besitzen, damit die Attacke der vollständigen Suche ausgeschlossen werden kann.
  - In der Praxis werden die Mengen  $X$  und  $Y$  durch Binärzeichen repräsentiert, deren Länge zwischen 64 und 2048 Bit liegen.
  - Eine Verschlüsselungsfunktion  $E_k : x \rightarrow y$  kann als Einwegfunktion angesehen werden, wenn der Schlüssel  $k$  geheimgehalten wird.
-

### Trapdoor one-way functions:

Eine Funktion  $f$  heißt Falltür-Einwegfunktion, wenn bei Kenntnis gewisser geheim zu haltender **Zusatzinformationen** für alle  $y \in Y$  ein  $x \in X$  mit  $f(x) = y$  existiert und leicht ermittelt werden kann.

- In zahlreichen Anwendungen soll die **Invertierbarkeit** der Einwegfunktion aber gerade vermieden werden, z. B. Passwortablage  $f(P)$ .
  - Bei der Passwortablage in einem Verzeichnis reicht es aus, dass das Passwort  $P$  mittels der Einwegfunktion  $f$  in den Wert  $w = f(P)$  transformiert und anstelle des Passwortes gespeichert wird.
  - Dadurch ist gewährleistet, dass aus dem gespeicherten Wert  $w$  das Passwort  $P$  auch dann nicht rekonstruiert werden kann, wenn  $f$  öffentlich bekannt ist – ein Vergleich jedoch damit ermöglicht wird.
-

### Einfache Realisierungsmöglichkeiten:

Sind die Mengen  $X = Y = Z_n$  gegeben, wobei  $n$  eine große natürliche Zahl ist.

Dann kann die folgende **Funktion  $f(x)$**  für jede natürliche Zahl  $k$  als eine der bedeutesten Einwegfunktionen der Kryptologie angesehen werden:

$$y = f(x) = x^k \bmod n$$

- Der Funktionswert  $y = f(x)$  kann schnell berechnet werden.
- Praxisrelevante Größenordnungen von  $n$  und  $k$  sind:

$$2^{1024}$$

Für  $x = f^{-1}(y)$  existiert kein effizienter Algorithmus

### Kap. 5: Einwegfunktionen

#### Teil 1: Einwegfunktionen und Hashfunktionen

- Integritätsschutz
- Hashfunktion

### Zielsetzung:

Das Ziel des Integritätsschutzes ist es, daß ein Empfänger einer Nachricht feststellen kann, ob er diese Nachricht unverfälscht erhalten hat. Das Grundprinzip des Integritätsschutzes besteht darin, die Nachricht unverschlüsselt und unverändert zu übersenden, gleichzeitig aber bestimmte Kontrollinformationen mitzuschicken, die die Kontrolle auf Unverfälschtheit der eigentlichen Nachricht ermöglicht. Für diese Kontrolldaten stellen sich damit folgende Bedingungen:

### Bedingungen:

- Der Umfang der Kontrollinformationen muß möglichst gering sein, um die zusätzlich zu übertragenden Informationen zu minimieren.
  - Praktisch jede Manipulation, auch nur eines einzelnen Bits der Nachricht muß anhand der Kontrollinformation feststellbar sein.
  - Die Kontrollinformationen müssen unmanipulierbar übertragen werden können.
-

### Hashfunktion:

Eine Hashfunktion ist eine Datentransformation mit folgenden Eigenschaften:

- **Kompressionseigenschaft:** Beliebig lange Bitfolgen werden auf Bitfolgen fester, im allgemeinen kürzerer Länge abgebildet (typischerweise 128 - 512 Bit)
- **Einwegeigenschaft:** Es muß praktisch unmöglich sein, zu einem vorgegebenen Hashwert eine Nachricht zu finden, deren Hashwert der vorgegebene Hashwert ist.
- **Kollisionsresistenz:** Es muß praktisch unmöglich sein, zwei Nachrichten zu finden, die zum gleichen Hashwert führen.

### Prüfvorgang:

Mit Hilfe einer beiden Kommunikationspartnern bekannten Hashfunktion können A und B die Integrität einer Nachricht überprüfen: Alice hasht ihre Nachricht, und übermittelt diese und den Hashwert so an Bob, daß die Unverfälschtheit des Hashwertes gewährleistet ist. Bob hasht die empfangene Nachricht ebenfalls und vergleicht sein Ergebnis mit dem von Alice gelieferten Hashwert.

---

### Kap. 5: Einwegfunktionen

#### Teil 2: Kryptographische Prüfsummen

- Message Digest (MD)
- Message Authentication Code (MAC)
- Kryptographische Hashfunktionen

### Message Authentication Code (MAC):

Ein Message Authentication Code ist eine kryptographische Checksumme zur Nachrichtensicherung, also eine Datentransformation, bei der zusätzlich ein geheimer Schlüssel in die Berechnung mit folgenden Eigenschaften eingeht:

- **Kompressionseigenschaft:** Beliebig lange Bitfolgen werden auf Bitfolgen fester im allgemeinen kürzerer Länge abgebildet.
- **Fälschungssicherheit:** Für jeden, der nicht im Besitz des Schlüssels ist, muß es praktisch unmöglich sein, den MAC-Wert einer neuen Nachricht zu berechnen, selbst wenn er im Besitz einiger alter Nachrichten mit den zugehörigen MAC-Werten gelangt ist.

### Prüfvorgang:

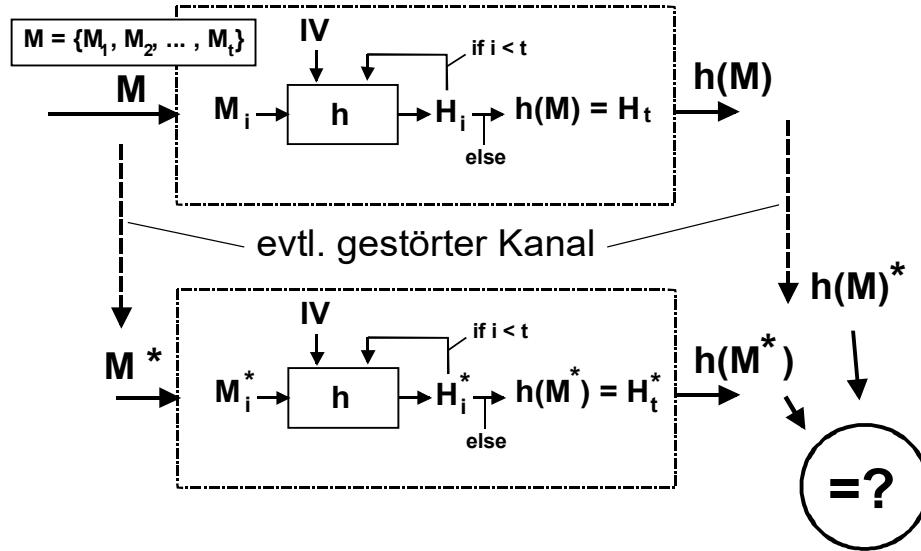
Besitzen Alice und Bob einen MAC und einen gemeinsamen, geheimen MAC-Schlüssel, so authentisiert Alice ihre Nachricht einfach dadurch, daß sie den MAC-Wert der Nachricht berechnet und zusammen mit der Nachricht an Bob schickt. Bob berechnet seinerseits den MAC-Wert der empfangenen Nachricht mit dem auch ihm bekannten MAC-Schlüssel.

---

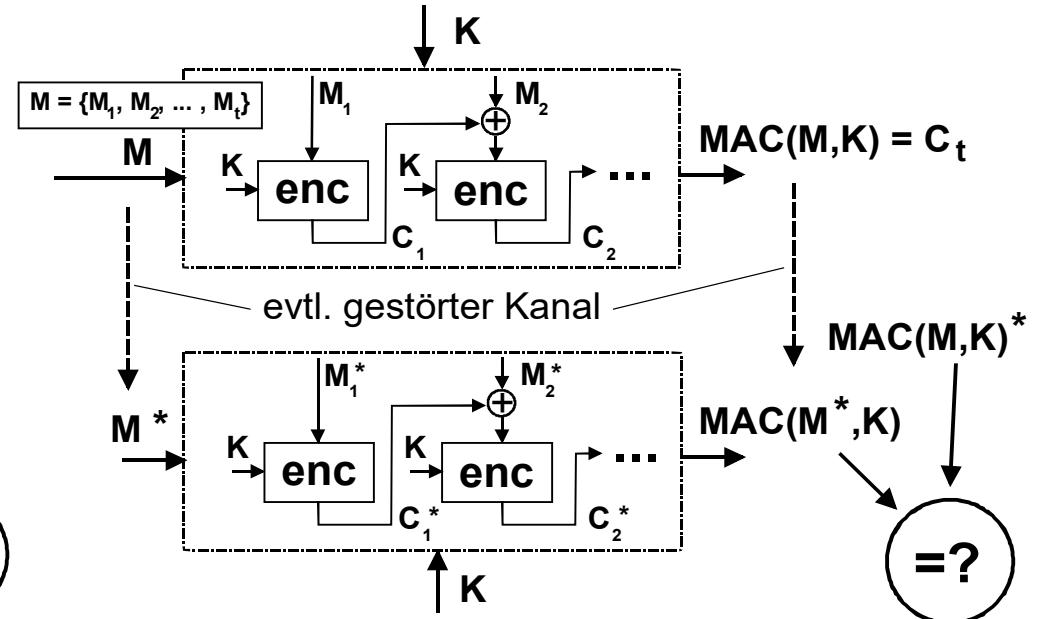
# Hashfunktion und MAC

Realisierung

## Message Digest (MD)



## Message Authentication Code (MAC)



## „Schlüsselgesteuerte“ Hashfunktion:

Im Anhang D zum **Standard X.509** ist noch folgende Möglichkeit zur Realisierung einer Hashfunktion aufgezeigt:

- Zunächst werde die Nachricht **m** in geeignet lange Blöcke  $m_1, m_2, \dots, m_r$  unterteilt und ggf. mit Einsen aufgefüllt.
- Der Hashwert  $h = H(m)$  wird dann für  $i = 1, 2, \dots, r$  durch die nach-folgende Vorschrift ermittelt:

$$h_i = (h_{i-1} + m_i)^2 \bmod n,$$

wobei als Startwert  $h_0 = 0$  und als Hashwert  $H(m) = h_r$  gesetzt werden.

- Durch die Modulofunktion wird das Vertauschen der Blöcke wirkungs-voll verhindert.
-

### Kap. 5: Einwegfunktionen

#### Teil 3: Secure Hash Algorithm

- Beschreibung des Verfahrens
- Praktische Implementierung

**SHA-1 ist heute eine sehr häufig benutzte Hashfunktion und wird beispielsweise im **Digital Signature Standard (DSS)** verwendet.**

### **SHA-1:**

- entwickelt vom U.S.-Government (NIST zusammen mit NSA)
- veröffentlicht 1993 unter der Bezeichnung FIPS PUB 180
- der Hashwert  **$h(M)$**  ist **160 Bit** lang (feste)
- die Original-MESSAGE  **$M$**  ist bis zu  **$(2^{64} - 1)$  Bit** lang (variabel)
- Die Blockgröße  **$M_i$**  beträgt **512 Bit**
- Die Rundenzahl ist 80

### **Anmerkung:**

Wir gehen im folgenden davon aus, dass uns die Original-MESSAGE  **$M$**  in Binärform  $x \in \{0, 1\}^*$  vorliegt.

---

### Anpassung der erforderlichen Länge:

Sei  $x \in \{0, 1\}^*$  die originäre Message und  $|x| < 2^{64}$ . Dann wird  $x$  so ergänzt (expandiert), dass  $x$  ein Vielfaches von 512 Bit ist.

1. Zunächst wird an  $x$  eine 1 angehängt:  $x \leftarrow x \circ 1$
2. Dann werden an  $x$  so viele Nullen angehängt, dass  $x = k \cdot 512 - 64$  bzw.  $x \equiv 448 \pmod{512}$ .
3. Schließlich wird die originäre  $x$ -Länge in Bits als 64-Bit-Integerzahl geschrieben und ebenfalls an  $x$  angehängt.  
⇒ Das **erweiterte**  $x$  hat somit eine Gesamtlänge von  $x = k \cdot 512$  Bit.  
vgl. Pseudocode S. 17

### Anpassung der erforderlichen Länge: (Pseudocode)

```
// Vorbereitung der Nachricht 'message':  
var int message_laenge := bit_length(message)  
erweitere message um bit "1"  
erweitere message um bits "0" bis Länge von message  
in bits  $\equiv 448 \pmod{512}$   
erweitere message um message_laenge als 64-Bit big-endian Integer
```

# Secure Hash Algorithm

## Beschreibung des Verfahrens

### Berechnung des Hashwertes:

Bei der Berechnung von **SHA-1(x)** werden

- die Funktionen

$$f_t : \{0, 1\}^{32} \times \{0, 1\}^{32} \times \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$$

sowie

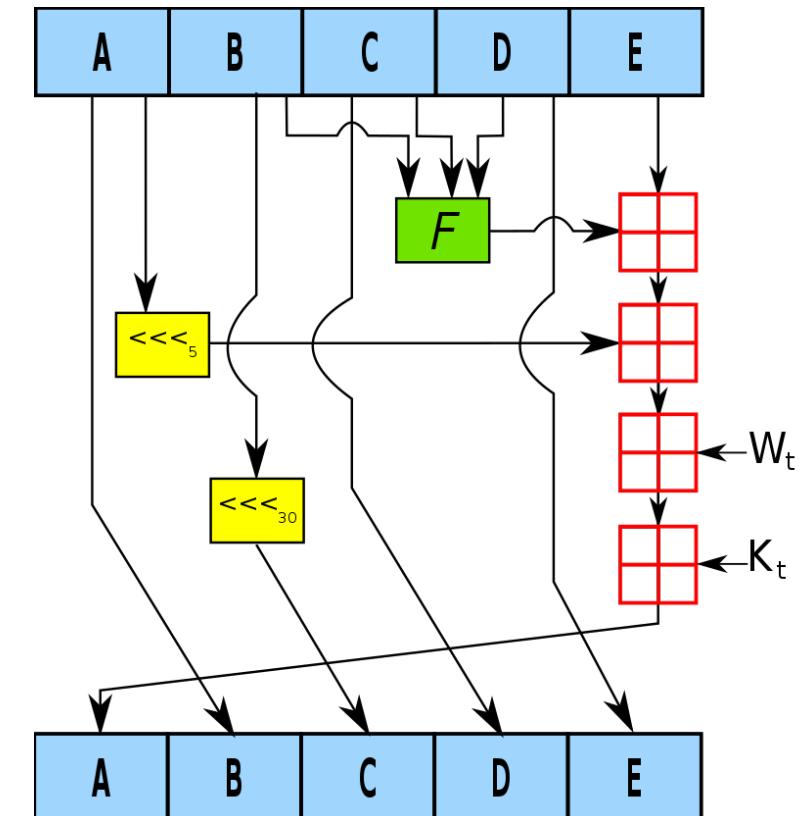
- die Konstanten

$K_t$

gemäß nebenstehender Abbildung benutzt.

$W_t$  = 32 Bit Wort des **erweiterten x**

$t$  = Rundenzähler [0; 79]



# Secure Hash Algorithm

## Beschreibung des Verfahrens

---

### Definitionen:

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \leq t \leq 19 \\ B \oplus C \oplus D & 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{für } 40 \leq t \leq 59 \\ B \oplus C \oplus D & 60 \leq t \leq 79 \end{cases}$$

$$K_t = \begin{cases} 5A827999 & 0 \leq t \leq 19 \\ 6ED9EBA1 & 20 \leq t \leq 39 \\ 8F1BBCDC & \text{für } 40 \leq t \leq 59 \\ CA62C1D6 & 60 \leq t \leq 79 \end{cases}$$

### Hashwertprozedur:

- A. Zerlege den **erweiterten Bitstring x** im Anschluss an den vorangegangenen Schritt 3 in 512 Bit Blöcke, d. h.  $x = M_1 M_2 M_3 \dots M_k$ .
  - B. Initialisiere  $H_0 = 67452301$ ,  $H_1 = EFCDAB89$ ,  $H_2 = 98BADCFE$ ,  
 $H_3 = 10325476$ ,  $H_4 = C3D2E1F0$ .
  - C. **for**  $i = 1$  **to**  $k$  **do** // Wiederhole für alle Blöcke  $M_i$ 
    1. Schreibe jedes  $M_i$  als Folge von **sechzehn** 32-Bit-Wörtern  $W_n$ ,  
d. h.  $M_i = W_0 W_1 \dots W_{15}$ ;
    2. Erweitere die **sechzehn** 32-Bit-Wörter um 64 weitere 32-Bit-Wörter,  
so dass sich insgesamt 80 Wörter je 32 Bit ergeben, d. h.  
Für  $t = 16, 17, \dots, 79$  setze  $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ ;
-

3. Setze  $A = H_0$ ,  $B = H_1$ ,  $C = H_2$ ,  $D = H_3$ ,  $E = H_4$ ;

4. **Hauptschleife**: Für  $t = 0, 1, \dots, 79$  setze nunmehr

$$T = S^5(A) + f_t(B, C, D) + E + W_t + K_t$$

$$E = D, D = C, C = S^{30}(B), B = A, A = T;$$

5. Setze im aktuellen Schritt i

$$H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C, H_3 = H_3 + D, H_4 = H_4 + E;$$

**next i** // Ende des i-ten Durchlaufs

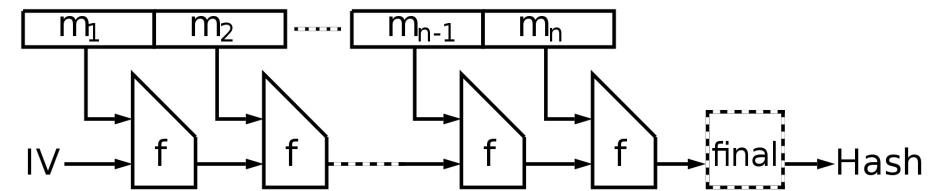
Im Anschluss an den **k-ten** Durchlauf obiger Prozedur ist der Hashwert von x dann gegeben durch

$$\text{SHA-1}(x) = H_0 H_1 H_2 H_3 H_4$$

Der so berechnete Hashwert **SHA-1(x)** wird nun mit dem nächsten Nachrichtenblock gehasht bzw. nach Abarbeitung des letzten Nachrichtenblocks als **Hashwert der Gesamt-Nachricht** ausgegeben.

### Anmerkung:

- In der vorangestellten Prozedur bedeutet  $S^k(w)$  einen zirkulären Links-Shift eines 32-Bit-Wortes w um k Bits.
- Außerdem bedeutet + die Addition zweier durch 32-Bit-Wörter dargestellter Zahlen mod  $2^{32}$ .
- Mit  $\oplus$  ist die bitweise XOR-Verknüpfung bezeichnet.



## SHA-1:

- Ein kritisches Angriffsszenario setzt voraus, dass der Angreifer eine (zumindest in Teilen) sinnvolle Variante eines Dokuments erzeugen kann, die den gleichen SHA-1-Wert besitzt.
  - 2017 wurde die erste Kollision von SHA-1 anhand zweier unterschiedlicher PDF-Dateien veröffentlicht.
  - Der Rechenaufwand war dabei enorm. Eine einzelne CPU hätte etwa 6500 Jahre dafür benötigt.
  - Derzeit werden Hashfunktionen der SHA-2-Familie (SHA-224, SHA-256, SHA-384 und SHA-512) empfohlen.
  - Langfristig sollen diese durch den neuen Standard **SHA-3** ersetzt werden.
-

## Berechnungen:

Nachricht	Hashwert der Nachricht
4711	e8fed7c5621fcc32f5db606fefee7c98f36cc2fa
4712	5ee217943f0d94ebbbdc7825adfd41fea2268f05
""	da39a3ee5e6b4b0d3255bfef95601890af80709
“Franz”	cb7ec4b22a9ba1e588e7f76247c201792d82e262
“Ganz”	e24176bf5cce5c6630792c6f2eb63144678f3ed5

### Kap. 5: Einwegfunktionen

#### Zusammenfassung:

- In diesem Kapitel wurde das Konzept der Einweg- und Falltür-Einwegfunktionen erläutert.
  - Was deren Klassifikation anbelangt, so wird zwischen Manipulation Detection Codes und Message Authentication Codes unterschieden.
  - Daneben wurden Realisierungen mit Hilfe der modularen Potenzfunktion sowie eines geeigneten Blockverschlüsslers vorgestellt.
  - Des Weiteren wurden Einwegfunktionen auf der Basis des Faktorisierungsproblems und diskreten Logarithmusproblems besprochen.
-

### Kap. 5: Einwegfunktionen

#### Zusammenfassung (Fortsetzung):

- Der für den Einsatz mit dem Digital Signature Standard (DSS) vom NIST zusammen mit der NSA entwickelte SHA-1 produziert einen 160 Bit langen Hashwert.
- Dadurch bietet SHA-1 einen besseren Schutz vor einem Brute-Force-Angriff (einschließlich Geburtstagsangriff) als die Vorgänger-algorithmen MD4 und MD5.
- SHA-1 kann heutzutage nicht mehr als ein sehr sicherer Hashalgorithmus bezeichnet werden.

---

# **Security**

## **- LV 4120 und 7240 -**

**Asymmetrische Kryptosysteme**

- El-Gamal Kryptosystem
- Asymmetrische kryptographische Verfahren sowie RSA-Algorithmus
- Das Rabin-Verschlüsselungsverfahren
- El-Gamal-Signaturverfahren
- Das Drei-Wege-protokoll nach X.509

### Kap. 6: Asymmetrische Kryptosysteme

#### Teil 1: Asymmetrische Verschlüsselung

- Das ElGamal-Kryptosystem
- Wahl der Systemparameter
- Ver- und Entschlüsselungsalgorithmus
- Sicherheit des Verfahrens

# ElGamal-Verschlüsselung

---

Beschreibung

- Das ElGamal-Verschlüsselungsverfahren wurde 1985 von dem Kryptologen **Taher Elgamal** entwickelt.
  - Es zählt zu den Public-Key-Verschlüsselungsverfahren und verwendet für jeden **Teilnehmer T** einen **öffentlichen Schlüssel  $PK_T$**  sowie einen **geheimen Schlüssel  $SK_T$** .
  - Der öffentliche Schlüssel kann veröffentlicht werden und dient der Verschlüsselung, während der geheime Schlüssel (nur dem Empfänger der Nachricht bekannt) bei der Entschlüsselung angewandt wird.
  - Im folgenden wird davon ausgegangen, dass ein **Sender A** eine **Nachricht m** an einen **Empfänger B** senden möchte.
  - Sender A und Empfänger B verfügen jeweils über ein **Schlüsselpaar**, welches einmalig erzeugt werden muss.
-

### Systemparameter:

1. eine Primzahl  $p$ .
2. eine multiplikative Einheitengruppe (Körper)  $\mathbf{G}$  über  $\mathbb{Z}_p^*$  mit:

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} \mid \text{ggT}(a, n) = 1\}$$

3. einen Erzeuger  $g$ .

### Anmerkung:

Die Parameter  $(\mathbf{G}, g)$  werden öffentlich gemacht und die Sicherheit des Kryptosystems verlangt eine **möglichst große Ordnung** der Gruppe  $\mathbf{G}$ . Daher wird  $p$  so gewählt, dass

$$p - 1 = 2 q ,$$

wobei  $q$  wiederum eine Primzahl ist (vgl. Sophie-Germain-Primzahlen).

---

### Schlüsselerzeugung:

Das Schlüsselpaar  $(PK_B, SK_B)$  des **Empfängers B** wird folgendermaßen erzeugt:

1. **B** wählt zufällig eine Zahl  $b \in \{1, \dots, p - 1\}$  mit  $\text{ggT}(b, p) = 1$ .
2. **B** berechnet das Gruppenelement:

$$B = g^b \in G(\mathbb{Z}_{p^*})$$

3. **B** setzt den Entschlüsselungsschlüssel (**geheim**)  $SK_B$ :

$$SK_B := b$$

4. **B** setzt den Verschlüsselungsschlüssel (**öffentlich**)  $PK_B$ :

$$PK_B := B$$

### Verschlüsselungsvorschrift:

Um die **Nachricht  $m \in G(\mathbb{Z}_p^*)$**  zu versenden, verfährt der **Sender A** folgendermaßen:

1. **A** wählt zufällig eine Zahl  $r \in \{1, \dots, p - 1\}$  mit  $\text{ggT}(r, p) = 1$ .
2. **A** berechnet das Gruppenelement:

$$R = g^r \in G(\mathbb{Z}_p^*)$$

3. **A** berechnet das Chiffrat (mit dem öffentlichen Schlüssel des Empfängers **B**):

$$c := P_{KB}^r \cdot m \in G(\mathbb{Z}_p^*)$$

4. **A** versendet:

$$(R, c)$$

als verschlüsselte **Nachricht  $m$**  an den Empfänger **B**.

---

### Entschlüsselungsvorschrift:

Um die verschlüsselte Nachricht  $(R, c)$  zu entschlüsseln, verfährt der **Empfänger B** folgendermaßen:

1. **B** berechnet das Gruppenelement (mit seinem **geheimen** Schlüssel  $SK_B$ ):

$$R^{-SK_B} \cdot c \equiv R^{p-1-SK_B} \cdot c \in G(\mathbb{Z}_{p^*})$$

bzw. unter Verwendung des Satzes von Fermat<sup>1)</sup>:

$$R^{p-1-SK_B} \cdot c \in G(\mathbb{Z}_{p^*})$$

---

1)

$$a^{p-1} \bmod p = 1 ; (a \neq 0)$$

### Entschlüsselungsvorschrift (Fortsetzung):

2. **B** verwendet das zuvor berechnete Gruppenelement als **Nachricht m**, denn es gilt:

$$\begin{aligned} R^{-SK_B} \cdot c &= (g^r)^{-SK_B} \cdot PK_B^r \cdot m = (g)^{-r \cdot SK_B} \cdot (g^b)^r \cdot m \\ &= (g)^{-r \cdot b} \cdot (g^b)^r \cdot m \\ &\quad \underbrace{\qquad\qquad\qquad}_{=1} \\ &= m \end{aligned}$$

□

---

### Anmerkung:

$$R := g^r ; c := PK_B^r \cdot m ; PK_B := B ; B = g^b ; SK_B := b$$

---

### Aufgabenstellung:

Wir betrachten das ElGamal-Verschlüsselungsverfahren über der Gruppe  $\mathbf{G}(\mathbb{Z}_{29}^*)$  mit dem Erzeuger  $g = 2$  und verschlüsseln die Nachricht  $m = 10$  mit dem öffentlichen Schlüssel  $\text{PK}_B = 5$  des Empfängers B sowie der Zufallszahl  $r = 8$ .

#### 1. Verschlüsselung:

$$\Rightarrow p = 29 \ (\forall \text{ mod } p!)$$

- Gruppenelement  $R = g^r = 2^8 \text{ mod } 29 = 24$
  - Chiffrat  $c := \text{PK}_B^r \cdot m = (5^8 \cdot 10) \text{ mod } 29 = 8$
  - **Sender A** versendet  $(R, c) = (24, 8)$  als **verschlüsselte** Nachricht m an den **Empfänger B**.
-

2. Entschlüsselung:

$$\Rightarrow p = 29 \ (\forall \text{ mod } p!)$$

Der Zusammenhang zwischen dem öffentlichen und dem geheimen Schlüssel ergibt sich beim ElGamal-Verfahren aus:

$$PK_B := B ; B = g^b ; SK_B := b \Rightarrow PK_B := g^b = g^{SK_B} \Rightarrow PK_B \equiv g^{SK_B} \text{ mod } p$$

$\Rightarrow$

- Geheimer Schlüssel aus  $5 \equiv 2^{SK_B} \text{ mod } 29 \Rightarrow SK_B = 22 \text{ (DL-Problem!)}$
- Gruppenelement  $R^{p-1-SK_B} \cdot c = (24^{29-1-22} \cdot 8) \text{ mod } 29$   
 $= (24^6 \cdot 8) \text{ mod } 29 = 10$
- D. h. die gesendete Nachricht ist  $m = 10$ .  $\square$

## Sicherheit des Verfahrens:

- Das ElGamal-Verschlüsselungsverfahren baut auf der Idee des Diffie-Hellman-Verfahrens auf.
- Es ist beweisbar sicher unter der Annahme, dass das sogenannte Diskret-Log-Problem in der zugrundeliegenden Gruppe schwierig ist.
- Durch schlechte Parameterwahl oder Implementierungsfehler können jedoch Spezialfälle konstruiert werden, die unsicher sind.
- In jedem Fall ist bei der Wahl der Gruppenstruktur  $G$  deren Zerfall in kleine Untergruppen zu vermeiden.
- Durch Mehrfachnutzung der gleichen Zufallszahl  $r$  sind Known-Plaintext-Angriffe möglich.

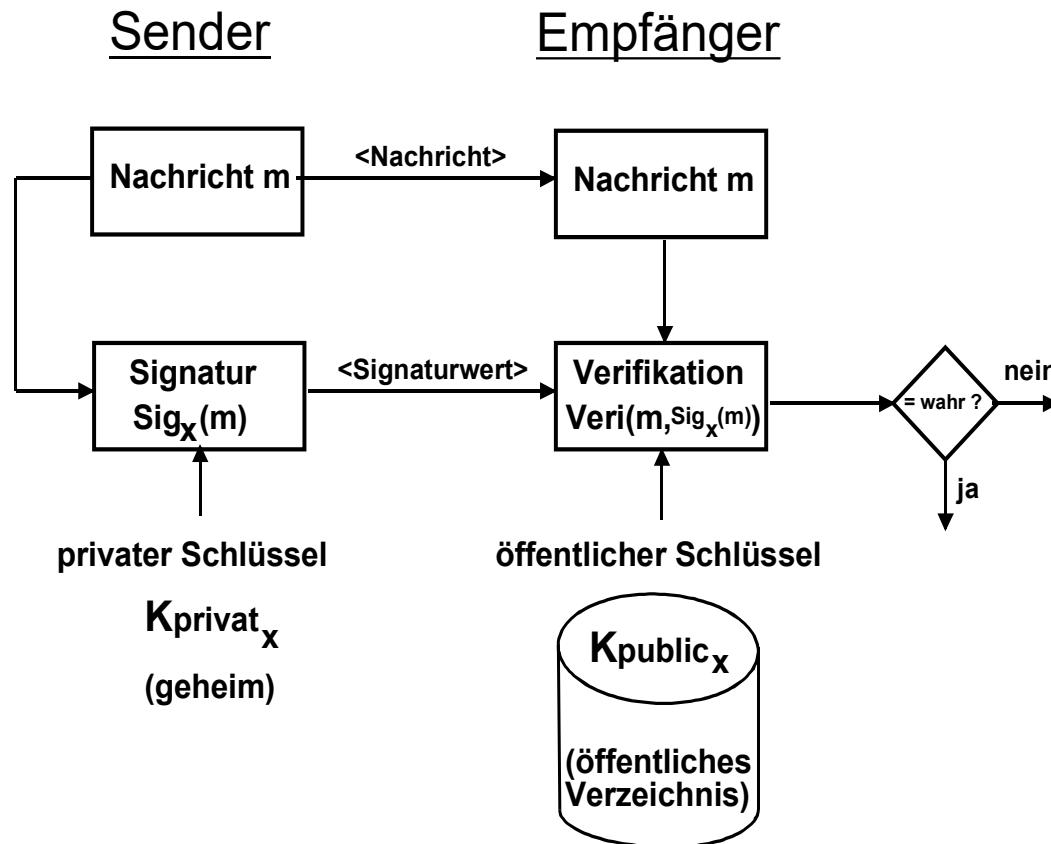
## Kap. 6: Asymmetrische Kryptosysteme

### Teil 2: Digitale Signaturen

- Ablaufskizze
- RSA-Signaturen
- Angriff auf RSA-Signatursysteme

# Digitale Signatur

Ablaufskizze



## Basisprinzip:

- Privater (geheimer) und öffentlicher Schlüssel
- Signaturwert mit privatem Schlüssel

## Eigenschaften:

- Nachweisbarkeit
- Nicht Abstreitbarkeit
- Authentizität
- Echtheit
- Identitätsnachweis

## RSA-Verfahren

# Digitale Signatur

## Signiervorschrift:

$$\text{sig}(m) := h(m)^{Sk} \bmod n$$

wobei  $n = p \cdot q$ ; (Modul)  
 $h$  = Hashfunktion  
 $Sk$  = geheimer Signaturschlüssel

## Schlüsselerzeugung:

$$\text{Sk} \cdot \text{Pk} \bmod \phi(n) = 1 \quad \text{sowie} \quad \text{ggT}(\text{Sk}, \phi(n)) = 1$$

$$\phi(n) = (p - 1)(q - 1)$$

## Verifizievorschrift:

$$h(m) \stackrel{?}{=} \text{sig}(m)^{pk} \bmod n$$

wobei  $P_k$  = öffentlicher Verifizierschlüssel

## Sicherheit:

$\text{Sk} = \text{Pk}^{-1} \bmod \phi(n)$  schwierig zu berechnen!  $\Rightarrow n = \{2^{768} \dots 2^{4096}\}$

# RSA-Verfahren

Zahlenbeispiel

---

<i>Vorgang</i>	<i>Erklärung</i>
	↓
Wähle:	$p = 13$ $q = 17$
Berechne:	$n = 13 \cdot 17 = 221$ $\phi(n) = (13 - 1)(17 - 1) = 192$
Wähle (zufällig):	
	$P_k = 101$
Nun ermittle <b>Sk</b> aus	
	$Sk \cdot 101 \bmod 192 = 1$
	$Sk \cdot P_k \bmod \phi(n) = 1$ d. h.
	$Sk \cdot 101 = z \cdot 192 + 1$
	$\Rightarrow Sk = 173$ (für $z = 91$ ) (geheimer Signaturschlüssel)

# RSA-Verfahren

Zahlenbeispiel

---

*Vorgang*



Annahme:  $h(m) := \mathbf{50}$

*Erklärung*



zu signierende Nachricht (Text)  
 $50 = 110010_2$

signieren:

$$\text{sig}(m) = 50^{173} \bmod 221 = \underline{\underline{33}}$$

$\text{sig}(m) := h(m)^{Sk} \bmod n$   
Die berechnete Signatur ist  
 $33 = 100001_2$

verifizieren:

$$33^{101} \bmod 221 = \underline{\underline{50}} := \underline{\underline{h(m)}}$$

$\text{sig}(m)^{Pk} \bmod n ?=? h(m)$   
Die berechnete Signatur ist  
korrekt!

# RSA-Signatur

Angriff

Signatur:  $\text{sig}(m) := m^{sk} \bmod n = \mathbf{a} \underbrace{(m^{sk} \bmod p)}_{\mathbf{a} \bmod q(p) = 0(1)} + \mathbf{b} \underbrace{(m^{sk} \bmod q)}_{\mathbf{b} \bmod q(p) = 1(0)}$  wobei  
 $\mathbf{sig}_1$   $\mathbf{sig}_2$

$$(m^{sk} \bmod n) \bmod p = \mathbf{a} \bmod p (m^{sk} \bmod p) + \mathbf{b} \bmod p (m^{sk} \bmod q)$$
$$m^{sk} \bmod p = \mathbf{1} (m^{sk} \bmod p)$$

Analog  $\bmod q$ :  $\Rightarrow m^{sk} \bmod q = \mathbf{1} (m^{sk} \bmod q)$

Fehlerbetrachtung (Differential Fault Analysis):

Voraussetzung: Fehler bei der Berechnung von  $\mathbf{sig}_1$  oder  $\mathbf{sig}_2$

Annahme: Fehler bei der Berechnung  $\mathbf{sig}_1$ ; Berechnung  $\mathbf{sig}_2$  sei o.k.!

$\mathbf{sig}_{\text{err}}$  ist das Ergebnis der fehlerhaften Berechnung von  $\text{sig}(m)$

$$\text{sig}(m) = \mathbf{a} \mathbf{sig}_1(m) + \mathbf{b} \mathbf{sig}_2(m) \Rightarrow \text{korrekte Signatur}$$

$$\mathbf{sig}_{\text{err}}(m) = \mathbf{a} \mathbf{sig}_{\text{err}}(m) + \mathbf{b} \mathbf{sig}_2(m) \Rightarrow \text{fehlerhafte Signatur}$$

---

$$\text{sig}(m) - \mathbf{sig}_{\text{err}}(m) = \mathbf{a} (\mathbf{sig}_1(m) - \mathbf{sig}_{\text{err}}(m)) \quad \text{Differential Fault Analysis (DFA)}$$

$$(\text{sig}(m) - \text{sig}_{\text{err}}(m)) = a (\text{sig}_1(m) - \text{sig}_{\text{err}}(m)) \quad | \text{ mod } q$$

$$(\text{sig}(m) - \text{sig}_{\text{err}}(m)) \text{ mod } q = 0, \text{ weil } a \text{ mod } q = 0$$

dann gilt:  $\text{sig}(m) \equiv \text{sig}_{\text{err}} \text{ mod } q$  aber  $\text{sig}(m) \neq \text{sig}_{\text{err}} \text{ mod } p$

also ist:  $q$  ein Teiler von  $(m - \text{sig}_{\text{err}}^{pk})$  und  $p$  kein Teiler von  $(m - \text{sig}_{\text{err}}^{pk})$

Ein Faktor von  $n = p \cdot q$  kann dann ermittelt werden aus:

$$\text{ggT}(\text{sig}(m)^{pk} - \text{sig}_{\text{err}}^{pk}, n) = \text{ggT}(m - \text{sig}_{\text{err}}^{pk}, n) = q$$

Und der zweite Faktor aus:

$$p = n / q$$

Geheimer Signaturschlüssel aus:  $S_k \cdot P_k \text{ mod } \phi(n) = 1$  mit  $\phi(n) = (p - 1)(q - 1)$

$$\Rightarrow S_k = P_k^{-1} \text{ mod } ((p - 1)(q - 1))$$

# RSA-Signatur

---

Zahlenbeispiel

Öffentlich bekannt:  $Pk = 101$  (öffentlicher Verifizierschlüssel)  
 $n = 221$  (Modul)

Korrekte Signaturergebnis:

$$\text{sig}(m) = [a \text{ sig}_1 + b \text{ sig}_2] \mod n = 33$$

Fehlerhafte Signaturberechnung:

$$\text{sig}_{\text{err}} = [a \text{ sig}_{\text{err}}_1 + b \text{ sig}_2] \mod n = 84$$

Analyse:

$$\rightarrow \text{sig}(m) - \text{sig}_{\text{err}} = -51$$

$$\begin{aligned} \rightarrow \text{ggT}(-51, n = 221) &= 17 =: q \quad \Rightarrow \quad p = 221 / 17 = 13 \\ &\quad \downarrow \quad \downarrow \\ -3 \text{ mal } 17 &\quad 13 \text{ mal } 17 \end{aligned}$$

$$\rightarrow \phi(n) = (13 - 1)(17 - 1) = 192$$

Geheimer Signaturschlüssel aus:

$$\Rightarrow \text{Sk} = 101^{-1} \mod 192 = \underline{\underline{173}} \rightarrow \text{Signatursystem gebrochen!}$$

---

### Kap. 6: Asymmetrische Kryptosysteme

#### Teil 3: Das Rabin-Verschlüsselungsverfahren

- Rabin-Modul und quadratische Reste
- Ver- und Entschlüsselung
- Sicherheit des Verfahrens

### Asymmetrische Kryptographie:

#### Erfinder:

Michael O. Rabin

- 1979 von Michael O. Rabin (isr. Inform.) veröffentlicht
- Asymmetrisches Kryptosystem
- Öffentlicher und privater Schlüssel
- Verwandt mit RSA-Verfahren



Michael O. Rabin

### Ablauf:

Sei  $T$  der Klartext,  $G$  der verschlüsselte Geheimtext und  $n = p \cdot q$  (Rabin-Modul) sowie  $p \equiv q \equiv 3 \pmod{4}$  mit  $p, q \in P$ .

- Wähle zwei Primzahlen  $p$  und  $q$  mit  $p \equiv q \equiv 3 \pmod{4}$ . Das Paar  $(p, q)$  ergibt den **geheimen** Schlüssel.
  - Als Produkt der beiden Primzahlen ergibt sich der Rabin-Modul  $n = p \cdot q$ , welcher den **öffentlichen** Schlüssel darstellt.
  - Der Sender verschlüsselt seine Nachricht mithilfe des öffentlichen Schlüssels  $n$  wie folgt:  $G = T^2 \pmod{n}$ .
  - Der Empfänger entschlüsselt die Nachricht, indem er mithilfe des geheimen Schlüssels  $(p, q)$  die **vier** Zahlen  $\pm r$  und  $\pm s$  berechnet durch
-

$$r = (y_p \cdot p \cdot T_q + y_q \cdot q \cdot T_p) \bmod n, \quad s = (y_p \cdot p \cdot T_q - y_q \cdot q \cdot T_p) \bmod n$$

mit

$$y_p \cdot p + y_q \cdot q = 1, \text{ sodass gilt: } T \in \{\pm r \bmod n, \pm s \bmod n\}.$$

Die Sicherheit des Verfahrens bemisst sich aus kryptologischer Sicht an:

$$G = T^2 \bmod n \quad \text{schwierig nach } T \text{ aufzulösen!}$$

$$\Rightarrow n = \{2^{400} \dots 2^{1024}\}$$

Anmerkung:

Man nennt **y** in der Gleichung des Typs  $y = x^2 \bmod n$  auch als **quadratischen Rest** bezüglich des Moduls n, falls Lösungen für x bzw. Quadratwurzeln von y mod n existieren. Sonst quadratischer Nichtrest.

---

### Faktorisierungsalgorithmus:

Bestimme Zufallszahl  $r \in \mathbb{Z}_n^*$

Berechne  $x = \text{RABINDECRYPT}(r^2 \bmod n)$ , d. h.  $x = r^2 \bmod n$

**if** ( $x \equiv +r \pmod{n}$  ||  $x \equiv -r \pmod{n}$ ) // ODER-Beziehung

**dann** Fehlschlag (wg. trivialer Quadratwurzel)

→ neues  $r$  bestimmen

**sonst** faktorisieren:  $p = \text{ggT}(x + r, n)$  sowie  $q = n / p$ .

Der Algorithmus faktorisiert  $n$  mit einer Erfolgswahrscheinlichkeit von  $1/2$ .

---

(Anm.:  $-r = n - r$ )

---

### Kap. 6: Asymmetrische Kryptosysteme

#### Teil 4: Signaturen und Authentifizierung

- ElGamal-Signaturen über  $Z_p^*$
- Das Drei-Wege-Protokoll nach X.509

Der ElGamal-Algorithmus ist eine Verallgemeinerung des Diffie-Hellman-Verfahrens und beruht auch auf der Basis des **diskreten Logarithmusproblems**. Mit den gleichen Bezeichnungen wie auf Folie 4 ergibt sich die gleiche Prozedur bis zum Austausch der beiden öffentlichen Schlüssel.

- A und B einigen sich auf eine große **Primzahl p** und eine geeignete **Primitivwurzel g**. Die beiden Zahlen dürfen öffentlich bekannt sein.
- A wählt eine Zufallszahl **SK<sub>A</sub>** und sendet  $\mathbf{PK_A} = g^{SK_A} \text{ mod } p$  an B.  
( $\mathbf{PK_A}$ , g, p) ist der **öffentliche**, ( $\mathbf{SK_A}$ , g, p) der **geheime** Schlüssel von A.
- B wählt eine Zufallszahl **SK<sub>B</sub>** und sendet  $\mathbf{PK_B} = g^{SK_B} \text{ mod } p$  an A.  
( $\mathbf{PK_B}$ , g, p) ist der öffentliche, ( $\mathbf{SK_B}$ , g, p) der geheime Schlüssel von B.

### Signaturerstellung:

Wir gehen davon aus, dass der Teilnehmer **A** dem Teilnehmer **B** eine signierte Nachricht  $h(m)$  übermitteln will. Die zu  $h(m)$  gehörige **digitale Signatur** werde durch ein Paar  $\text{sig}(h(m)) = (r, s)$  repräsentiert.

- **A** wählt eine zu  $p - 1$  teilerfremde Zahl  $k$ . // Wegen Voraus.  $\exists k^{-1}$
  - **A** berechnet  $r = g^k \mod p$ .
  - **A** löst die Kongruenz  $h(m) = (SK_A \cdot r + k \cdot s) \mod (p - 1)$ . Der unbekannte Wert  $s$  ergibt sich durch  $s = k^{-1} \cdot (h(m) - SK_A \cdot r) \mod (p - 1)$ .
  - **A** schickt  $h(m)$  sowie  $\text{sig}(h(m))$  d. h.  $h(m)$  und  $(r, s)$  an **B**.
-

### Signaturprüfung:

Es ergibt sich mit

$$h(m) = (SK_A \cdot r + k \cdot s) \bmod (p - 1)$$

die Beziehung:

$$g^{h(m)} = g^{SK_A \cdot r + k \cdot s} = g^{SK_A \cdot r} \cdot g^{k \cdot s} = PK_A^r \cdot r^s \bmod p.$$

Damit ist jeder Teilnehmer, insbesondere der Empfänger **B**, in der Lage zu verifizieren, ob  $h(m)$  tatsächlich von **A** signiert wurde.

$$\text{Verify}(h(m), (r, s), PK_A) = \text{true} \iff g^{h(m)} \bmod p = PK_A^r \cdot r^s \bmod p$$

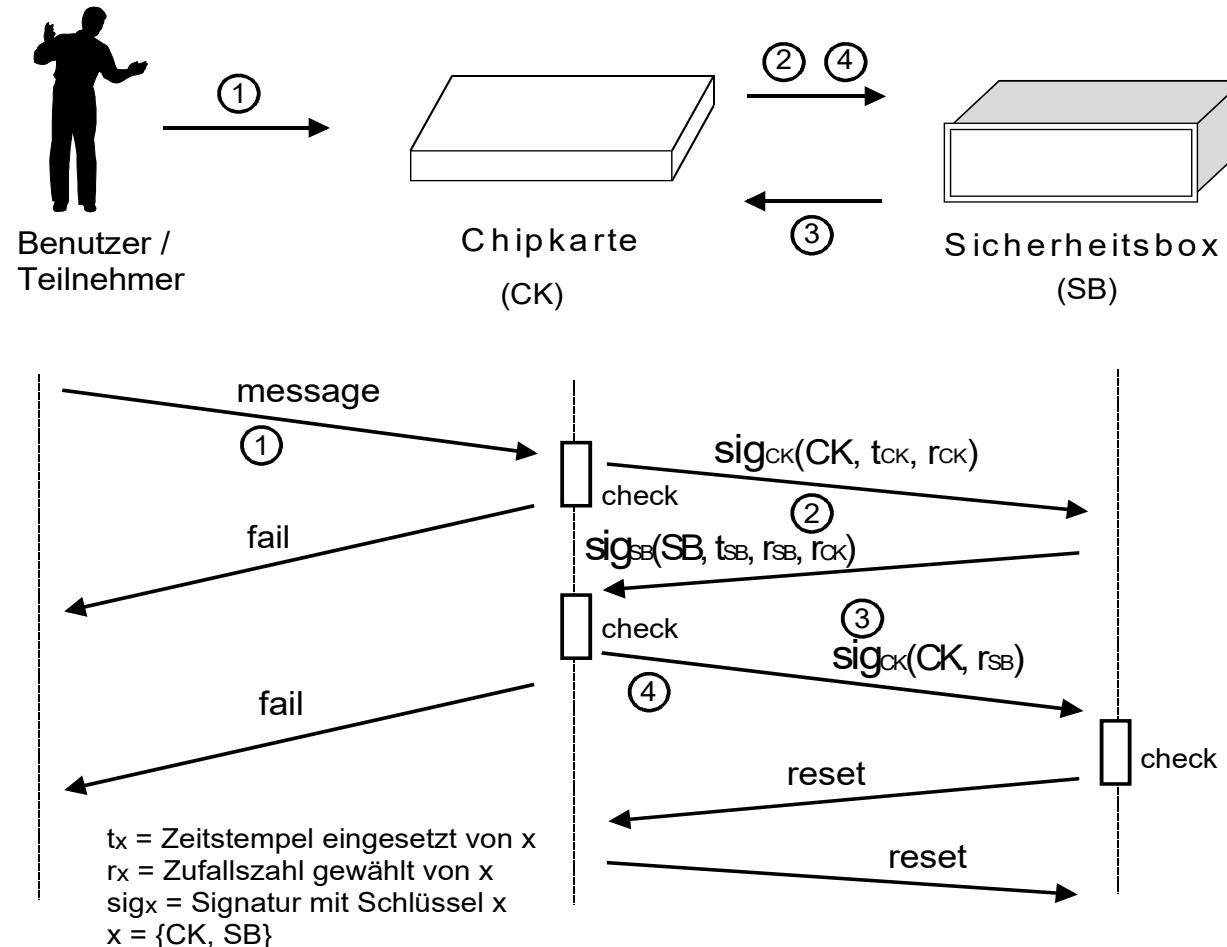
Im Gegensatz zu RSA ist hier eine Signatur  $(r, s)$  etwa doppelt so lang.

## Sicherheit des Verfahrens:

- In der Praxis wählt man auch hier die (sichere) Primzahlen  $p \in \mathbb{P}$  (sogenannte *Sophie-Germain-Primzahl*) gemäß  $p = 2q + 1$ , wobei  $q \in \mathbb{P}$  ebenfalls eine Primzahl ist.
- Dann besitzen alle Untergruppen die **Ordnung**  $q$ .
- Die Größe von  $q$  bestimmt die Sicherheit des Verfahrens.
- Die Parameterwahl  $p$ ,  $q$  und  $g$  kann für alle Teilnehmer gemeinsam getroffen werden.
- Ferner wird eine **kollisionsresistente** Hashfunktion  $h$  benötigt.
- Alle  $SK_T$  der Teilnehmer  $T$  müssen selbstverständlich **geheim** gehalten werden.

# Authentizität

## Gegenseitiger Nachweis



Initiator A

Responder B

$A, \{B\}, R_a, \text{SecNeg}_a, \{\text{Cert}_a\}$

Flow1-3WE

$A, B, \text{SecNeg}_b, \{\text{Cert}_b\}, \{R_a, R_b, \{\text{Enc}_{K_a}(\text{ConfPar}_b)\}, \text{Sig}_{K_b}(\text{hash}(A, B, R_a, R_b, \text{SecNeg}_a, \text{SecNeg}_b, \{\text{ConfPar}_b\}))\}$

<

Flow2-3WE

$\{A, B, R_b, \{\text{Enc}_{K_b}(\text{ConfPar}_a)\}, \text{Sig}_{K_a}(\text{hash}(A, B, R_b, \{\text{ConfPar}_a\}))\}$

Flow3-3WE

**SecNeg\_** = Security Negotiation

**Cert\_** = Zertifikat / CRL

**Enc\_** = Verschlüsselung

**ConfPara\_** = Confidential Parameters

**Sig\_** = Signatur

**hash** = Hashfunktion

## Kap. 6: Asymmetrische Kryptosysteme

### Zusammenfassung:

- In diesem Kapitel wurden asymmetrische Kryptoverfahren, die auf dem Faktorisierungsproblem, dem modularen Wurzelziehen und dem diskreten Logarithmusproblem beruhen, vorgestellt.
  - Dabei wurden zunächst das auf dem Faktorisierungsproblem basierende Signaturverfahren von Rivest, Shamir und Adleman (**RSA**-Verfahren) und anschließend das auf quadratischen Resten beruhende Verschlüsselungsverfahren von **Rabin** präsentiert.
  - Dann folgte das auf dem DLP basierende Schlüsselaustauschprotokoll von von **EIGamal** (**EG**-Verfahren).
-

---

# **Security**

## **- LV 4121 und 4241 -**

**Schlüsselmittelmanagement und Zufallszahlen**

## Kap. 7: Schlüsselmittelmanagement und Zufallszahlen

### Teil 1: Erzeugung von Zufallszahlen

- Allgemeine Aspekte und Zielsetzung
- Zufallszahlengeneratoren
- Neumann-Filter

### Allgemeine Aspekte

- Im Zusammenhang mit **kryptographischen Schlüsselpараметern** (Initialisierungsvektoren, Startwerte etc.) spielt das Erzeugen von Zufallszahlen (möglichst zufällig, hinreichend groß, unvorhersehbar, besondere Eigenschaften uvm.) eine zentrale Rolle.
  - So basiert die **Sicherheit** aller kryptographischen Verfahren auf der Schwierigkeit, einen geheimen Schlüsselparameter zu erraten oder anderweitig zu beschaffen.
  - Zufallszahlen und deren Nachbehandlung sind daher ein wesentliches kryptographisches Grundelement.
  - Da die Erzeugung echter Zufallszahlen nicht unproblematisch ist, werden vielerorts Pseudozufallszahlen verwendet.
-

## Zielsetzung:

- Es sollte sehr schwierig sein, die Output-Bits eines Pseudo-Zufallszahlengenerators oder eines echten Zufallszahlengenerators vorherzusagen zu können.
- Selbst wenn eine Folge von  $n$  aufeinanderfolgenden Output-Bits  $a_1, a_2, \dots, a_n$  eines Generators vorliegen, sollte es rechnerisch unmöglich sein, das  $(n+1)$ -te Bit  $a_{n+1}$  mit einer Wahrscheinlichkeit, die größer als 0.5 ist, vorherzusagen.

### Pseudo-Zufallszahlengenerator:

#### Definition:

- Ein **Pseudozufallszahlengenerator** ist ein Algorithmus, der nach Eingabe von gewissen Initialisierungsdaten (sogenannten **seed numbers**) eine Zufallsfolge deterministisch erzeugt.
- Einen solchen randomisierten Algorithmus, der in Form eines Simulations- oder Rechenprogramms lediglich eine pseudozufällige Bitfolge liefert, nennt man **pseudo random number generator (PRNG)**.

### Der echte Zufallszahlengenerator:

#### Definition:

A **random bit generator** is a device that is designed to output a sequence of statistically independent and symmetrically distributed **binary random variables**, i. e., that is designed to be the implementation of a so-called **binary symmetric source (BSS)**.

- Das Wissen der ersten n Bits einer zufälligen Folge liefert keine Information über das n + 1-te Bit.
  - Eine gute Zufallsquelle stützt sich auf physikalische Zufallsereignisse wie zum Beispiel thermisches Rauschen oder radioaktiver Zerfall ab.
  - Den zugehörigen Prozess nennt man **real random number generator (RRNG)**.
-

### Nachbehandlung echter Zufallsfolgen:

Auch wenn die Zufallszahlen aus einem physikalischen Prozess stammen, muss untersucht werden, ob der zugrunde liegende physikalische Prozess **echt** zufällig ist und im Falle einer **statistisch unabhängigen** Zahlenfolge diese eine symmetrische Verteilung bezüglich der Werte „0“ und „1“ aufweist.

### Der Neumann-Filter:

Der Informatikpionier **John von Neumann** schlug 1951 eine sehr effektive Funktion **f** zur Beseitigung der Asymmetrie in einer Bitfolge vor:

$$f : \{0, 1\}^m \rightarrow \{0, 1\}^n \text{ mit } 00 \rightarrow \varepsilon, 11 \rightarrow \varepsilon, 01 \rightarrow 0, 10 \rightarrow 1,$$

wobei sich **f** auf zwei aufeinanderfolgende Bits (nicht überlappende Bit-Paare) bezieht und  $\varepsilon$  für die leere Zeichenkette steht.

---

### Eigenschaften nach Anwendung des Neumann-Filters:

- Wenn in einer Bitfolge  $a_i \rightarrow \{0,1\}^n$  aufeinanderfolgende Bits **statisch unabhängig** sind und den Wert „1“ mit der Wahrscheinlichkeit  $p$  annehmen, so verkürzt sich die Länge der Bit-Folge durch die Filterung um den Faktor  $p(1 - p)$ .
- Im Falle  $p = 1/2$  gehen dann etwa  $3/4$  aller ursprünglichen Bits verloren und für alle anderen Werte von  $p$  ist der **Verlust** noch höher (dies ist der Preis für die Verbesserung der Zufälligkeit).
- Da die Wahrscheinlichkeit für ein Paar „01“ bzw. „10“ in der ursprünglichen Bitfolge gleich  $p(1 - p)$  ist, ergibt sich für die Wahrscheinlichkeit  $p_0$  und  $p_1$  für den Wert 1 bzw. 0 nach der Filterung der Wert  $1/2$ .

## Kap. 7: Schlüsselmittelmanagement und Zufallszahlen

### Teil 2: Primzahlen

- Primzahlen und Fundamentalsatz der Arithmetik
- Sieb von Eratosthenes
- Primzahltests und Primteilerzerlegung
- Primzahlhäufigkeit und Primzahldichtefunktion

Definition: Es sei  $p \in \mathbf{Z}$  und  $p > 1$ .  $p$  heißt **Primzahl** (auch unzerlegbar oder irreduzierbar)  $\Leftrightarrow$  Es gibt keine Teiler  $a$  von  $p$  mit  $1 < a < p$ .

Eine Zahl  $a \in \mathbf{Z}$  mit  $|a| \geq 2$  heißt **zerlegbar**, wenn  $|a|$  keine Primzahl ist.

Satz:

1. Es sei  $p \in \mathbf{Z}$  und  $p > 1$ .  $p$  ist genau dann eine Primzahl, wenn gilt:

$$\text{Aus } p \mid (a \cdot b) \Rightarrow p \mid a \text{ oder } p \mid b .$$

2.  $\forall a \in \mathbf{N}$  mit  $a \geq 2 \Rightarrow \exists$  Primzahl  $p$  mit  $p \mid a$ .

3. Jede Zahl  $a \in \mathbf{N}$  mit  $a \geq 2 \Rightarrow a$  ist das Produkt endlich vieler Primzahlen, d. h. es gilt die **Primfaktorzerlegung**:

$$a = p_1 \cdot p_2 \cdot \dots \cdot p_k$$

### Fundamentalsatz der Arithmetik

Jede natürliche Zahl  $a > 1$  besitzt eine eindeutige Primfaktorzerlegung der Form:

$$a = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_k^{a_k}$$

wobei  $p_1, \dots, p_k \in \mathbf{P}$  und  $a_1, \dots, a_k \in \mathbf{N}$ .

Satz:

Sei  $n$  eine ungerade Zahl für die auch  $(n - 1) / 2$  ungerade ist. Dann gilt:

$$1. \ n \text{ prim} \quad \Rightarrow \quad a^{(n-1)/2} = \pm 1 \text{ für } \forall a \in \mathbf{Z}_n \setminus \{0\}$$

$$2. \ n \text{ nicht prim} \quad \Rightarrow \quad a^{(n-1)/2} = \pm 1 \text{ für höchstens die Hälfte der } a \in \mathbf{Z}_n \setminus \{0\}$$

(Beweis folgt aus dem Fermatschen Satz, vgl. Übung!)

---

- Es existieren **unendlich** viele Primzahlen.
  - Es existiert **kein** bekanntes effizientes Verfahren für die Zerlegung großer Zahlen in ihre Primfaktoren.
  - Die Zahl 1234567891 ist eine Primzahl.
  - Die Zahl 1987654321 ist keine Primzahl, sondern das Produkt von 457 und 4349353.
  - $(2^{77232917} - 1)$  ist die größte bislang bekannte Primzahl. Sie hat 23.249.425 **Dezimalstellen** und wurde 2018 entdeckt (Jonathan Pace, US-amerik. Germantown, Tennessee).
  - Für jede natürliche Zahl  $n$  gibt  $\pi(n)$  die Zahl der Primzahlen  $\leq n$  an:  
$$\pi(n) \approx n / (\ln(n) - 1.08366) \approx n / \ln(n)$$
 für großes  $n$ .
-

Definition: Eine natürliche Zahl  $n > 1$  heißt Primzahl, wenn sie nur durch 1 und sich selbst ohne Rest teilbar ist.

Lösungsalgorithmus:

1. Wir bringen alle Zahlen von 2 bis  $n$  in ein Sieb.
  2. Wir nehmen die erste (kleinste) Zahl aus dem Sieb heraus und heben sie separat auf. Danach lassen wir alle *nicht-trivialen* Vielfachen dieser Zahl durch das Sieb fallen.
  3. Dann nehmen wir die nächst größere der *übriggebliebenen* Zahlen aus dem Sieb heraus und heben auch diese separat auf. Alle Vielfachen dieser Zahl lassen wir wiederum durch das Sieb fallen.
  4. Wir wiederholen den 3. Schritt so oft, bis keine Zahlen mehr im Sieb sind.
  5. Die separat aufgehobenen Zahlen sind die gesuchten Primzahlen.
-

Erzeugt Primzahlen  $p$  unterhalb der Schranke  $\text{maxp}$ .

Pseudocode:

```
for p = 2 bis maxp put Z(p) := TRUE  
for p = 2 bis  $\sqrt{\text{maxp}}$   
  if Z(p) == TRUE then  
    { k = p * p  
      do  
        Z(k) := FALSE  
        k := k + p  
      while k <= maxp }
```

Ausgabe:

$\forall$  Primzahlen  $p = 2$  bis  $\text{maxp}$ , für die  $Z(p) == \text{TRUE}$  ist.

# Zahlentheorie

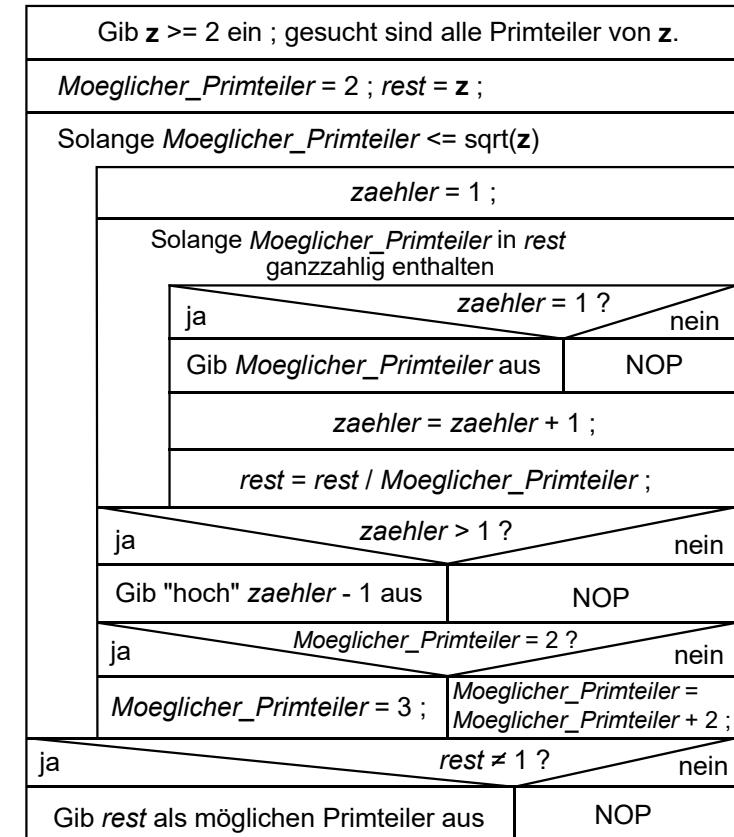
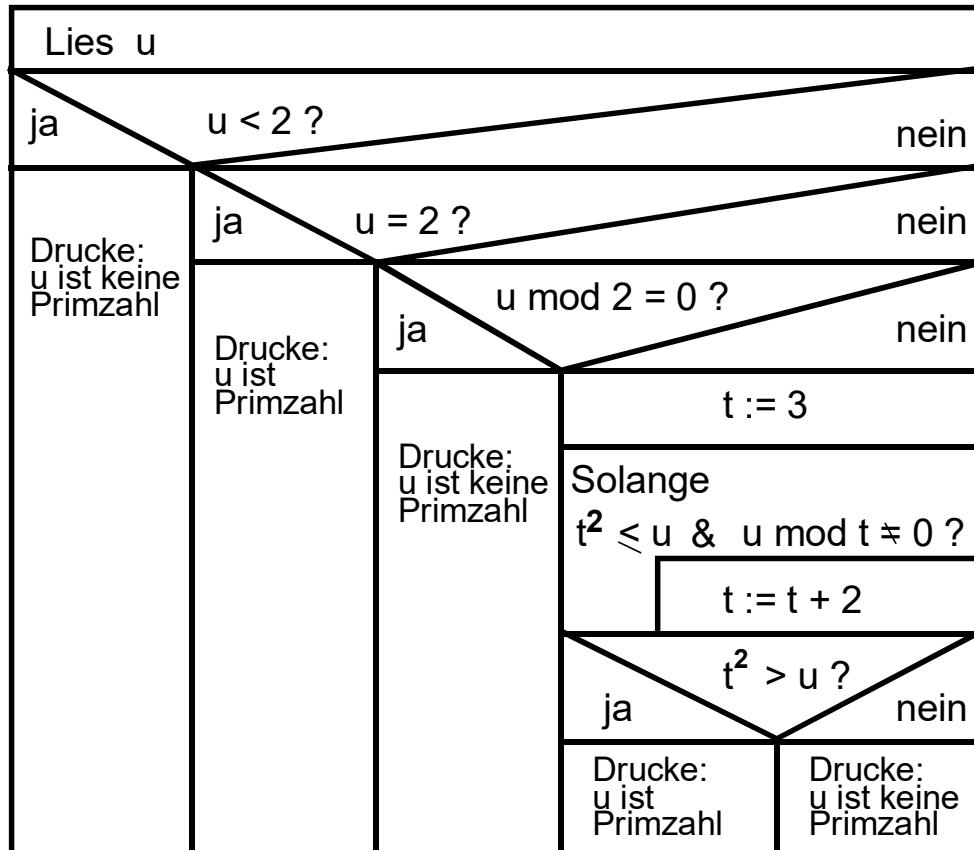
## Primzahlen

Primzahlen zwischen 0 und 300: Rechenbsp.:  $n = 300$   $\ln(300) = 5.7$   
 $\Rightarrow$  Es existieren 62 Primzahlen.  $\Rightarrow \pi(300) \approx \underline{\underline{64}}$

## Abschätzung:

$$\pi(n) \approx n / (\ln(n) - 1.08366) \approx n / \ln(n) \text{ für großes } n.$$

### Primzahltest und Primteilerzerlegung (deterministisch)



## Statistischer Primzahltest von Miller und Rabin (MRT)

Sei  $n$  eine ungerade Zahl für die auch  $(n - 1) / 2$  ungerade ist (sog. *erlaubte Zahlen*):

1. Wähle Zufallszahlen  $a_1, \dots, a_k \in \{2, \dots, n - 2\}$  aus (auch *Zeugen* genannt).
2. Berechne  $a_i^{(n-1)/2}$  in  $\mathbf{Z}_n$  (folgt aus dem kleinen Satz von Fermat).
3. Falls alle  $a_i^{(n-1)/2} = \pm 1$  bzw.  $1$  oder  $n - 1$ ,  
dann entscheide:  $n$  ist **wahrscheinlich** prim.  
sonst entscheide:  $n$  ist **definitiv nicht** prim.

Die Fehlerwahrscheinlichkeit des Tests ist  $\leq (1/4)^k$ , wenn  $k$  die Anzahl der gewählten Zeugen ist.

→ Test liefert nicht immer die korrekte Antwort!

---

## MRT (Fortsetzung)

Der Miller-Rabin-Algorithmus kann deterministisch angewendet werden, indem alle **Basen** in einer bestimmten Menge getestet werden.

Aus  $(\mathbb{Z}/n\mathbb{Z})^*$  ist  $a$  ein Zeuge für das Zusammengesetztsein von  $n$  mit

$$\forall a \in \{2, 3, \dots, \min(n - 1, 2 \cdot \ln^2(n))\}.$$

Allerdings müssen in der Praxis nicht alle  $a$  bis zur Obergrenze  $2 \cdot \ln^2(n)$ , sondern nur eine sehr viel kleinere Menge getestet werden.

$n$	ungefähr	Als Zeuge zu testen sind die Zahlen
$< 2^{16}$	65.536	2, 3
$< 2^{32}$	4.294.967.296	2, 7, 61
$< 2^{64}$	$\approx 1.844 \cdot 10^{19}$	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37
$< 3.317.044.064.679.887.385.961.981$		2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41

---

# Primzahlen

Tests(4)

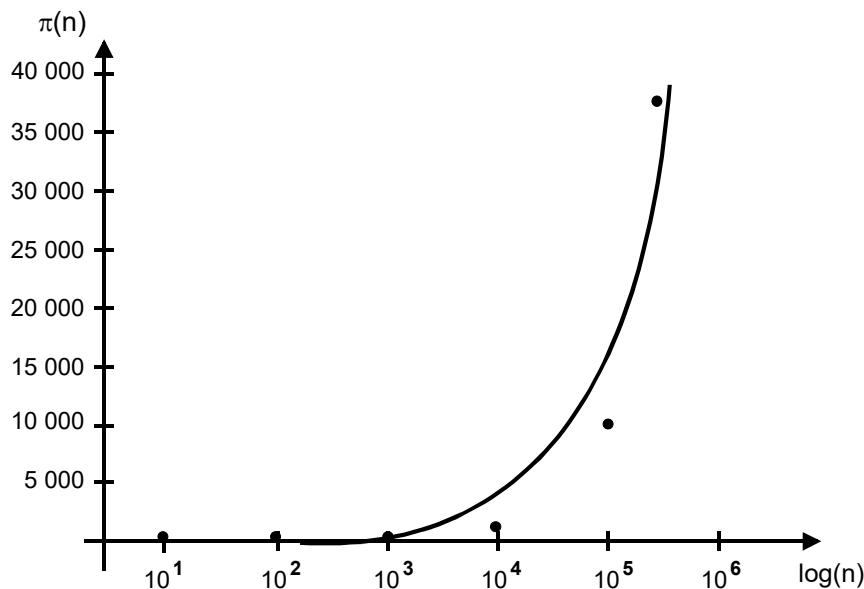
---

Beispiele:

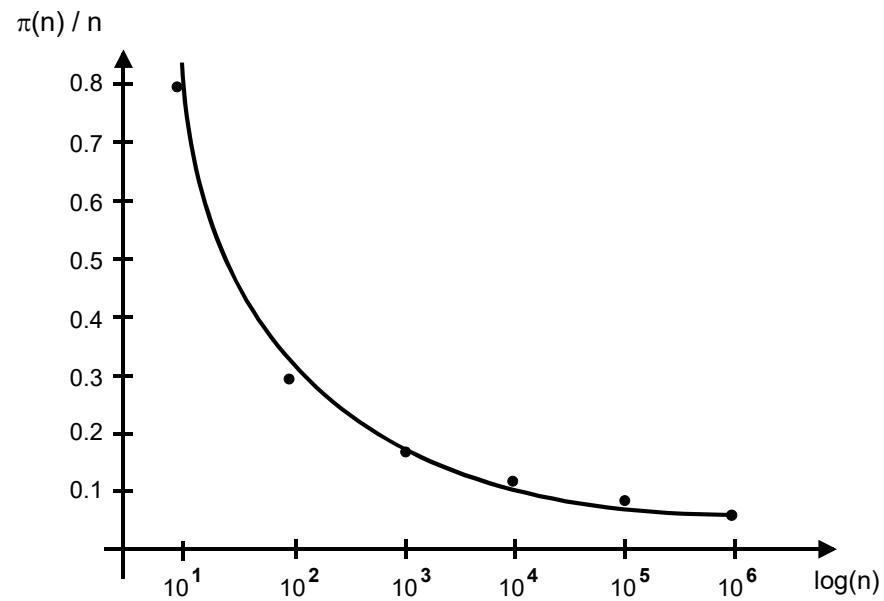
Zahl u	Primzahl ja / nein ?
15.681	nein
194.609	ja
224.711	ja
302.515.449	nein
2.147.483.647	nein

Zahl u	Zerlegung
137.917	$13 \cdot 103^2$
119.394.613	$13^2 \cdot 19^3 \cdot 103$
2 E9	$2^{10} \cdot 5^9$
183.495.637	$13^3 \cdot 17^4$

## Primzahlhäufigkeit



## Primzahldichtefunktion



$$\pi(n) = n / (\ln(n) - a_0) \quad \text{mit} \quad a_0 = 1.08366$$

## Kap. 7: Schlüsselmittelmanagement und Zufallszahlen

### Teil 3: Pseudozufallszahlengeneratoren

- Linearer Kongruenzgenerator und Rauschgenerator
- Blum-Blum-Shub-Generator (BBS)
- Lineare Schieberegister mit Rückkopplung
- Geffe-Generator
- Blum-Micali-Generator

# Pseudozufallszahlengenerator

## Linearer Kongruenzgenerator

---

Formel:

$$x_{n+1} = (a \cdot x_n + b) \bmod n$$

$x_0$  = Startwert (seed number)  $\in \mathbf{N}+1$

$a, b, n$  = Parameter (konstant)  $\in \mathbf{N}+1$

$x_{n+1}$  = Pseudozufallszahlen ( $n = 0, 1, 2, \dots$ )

Parameterwahl:

Parameter	Möglichkeit 1	Möglichkeit 2
a	137153	7141
b	17	54773
n	$2^{19}$	259200

Ausgabe:

$x_0, x_1, x_2, \dots$

Formel:

$$x_{n+1} = a \cdot x_n - \text{int}(a \cdot x_n / b) \cdot b$$

$x_0$  = Startwert (seed number)  $\in \mathbf{N+1}$

$a, b$  = Parameter (konstant)  $\in \mathbf{N+1}$

$\text{int}(\dots)$  = ganzzahliger Anteil Klammerausdruck

$x_{n+1}$  = Pseudozufallszahlen ( $n = 0, 1, 2, \dots$ )

Parameterwahl:

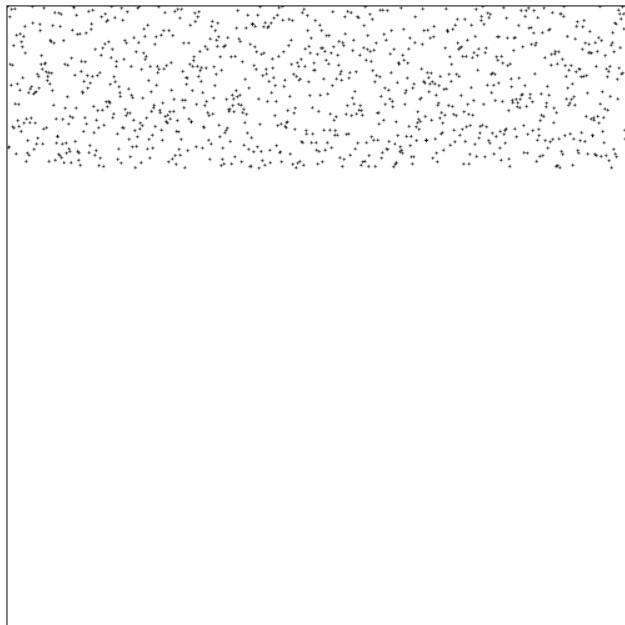
Parameter	Möglichkeit 1	Möglichkeit 2
$a$	23	29
$b$	100 000 001	1 000 001
$x_0$	439 147	691 156

Ausgabe:

$x_1, x_2, x_3, \dots$

### Linearer Kongruenzgenerator

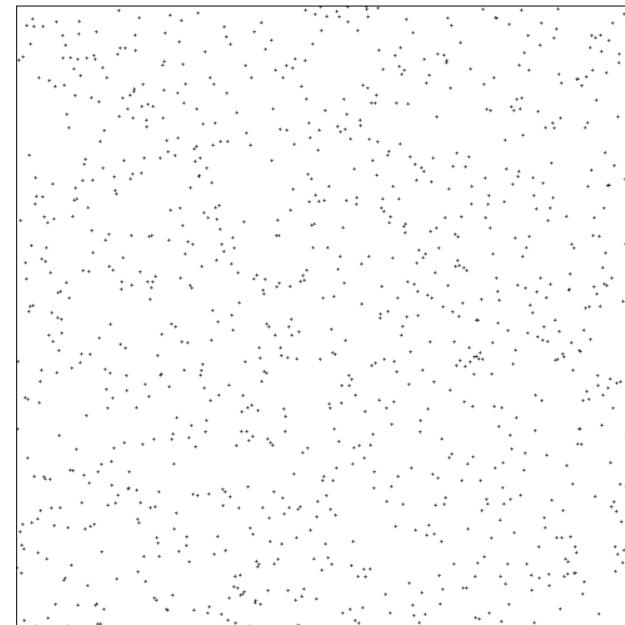
$x_0 = 4711$   $a = 7141$   $b = 54773$   
 $n = 259200$



x-Achse: 1, 2, 3, ..., 1000  
y-Achse:  $x_1, x_2, x_3, \dots, x_{1000}$

### Integerzahlengenerator

$x_0 = 439147$   $a = 23$   $b = 100000001$   
(gerechnet von Marcell Dietl)



x-Achse: 1, 2, 3, ..., 1000  
y-Achse:  $x_1, x_2, x_3, \dots, x_{1000}$

1. Wähle zwei große Primzahlen  $p$  und  $q$ , die beide bei Division durch 4 den Rest 3 ergeben. D. h.

$$p \equiv q \equiv 3 \pmod{4}$$

2. Berechne  $n = p \cdot q$  und wähle eine Zufallszahl  $s$ , die relativ prim zu  $n$  ist. Daraus berechne die Seed-Zahl  $x_0$ :

$$x_0 = s^2 \pmod{n}$$

3. Berechne nun mit  $i = 1$  beginnend wiederholt:

$$x_i = (x_{i-1})^2 \pmod{n}$$

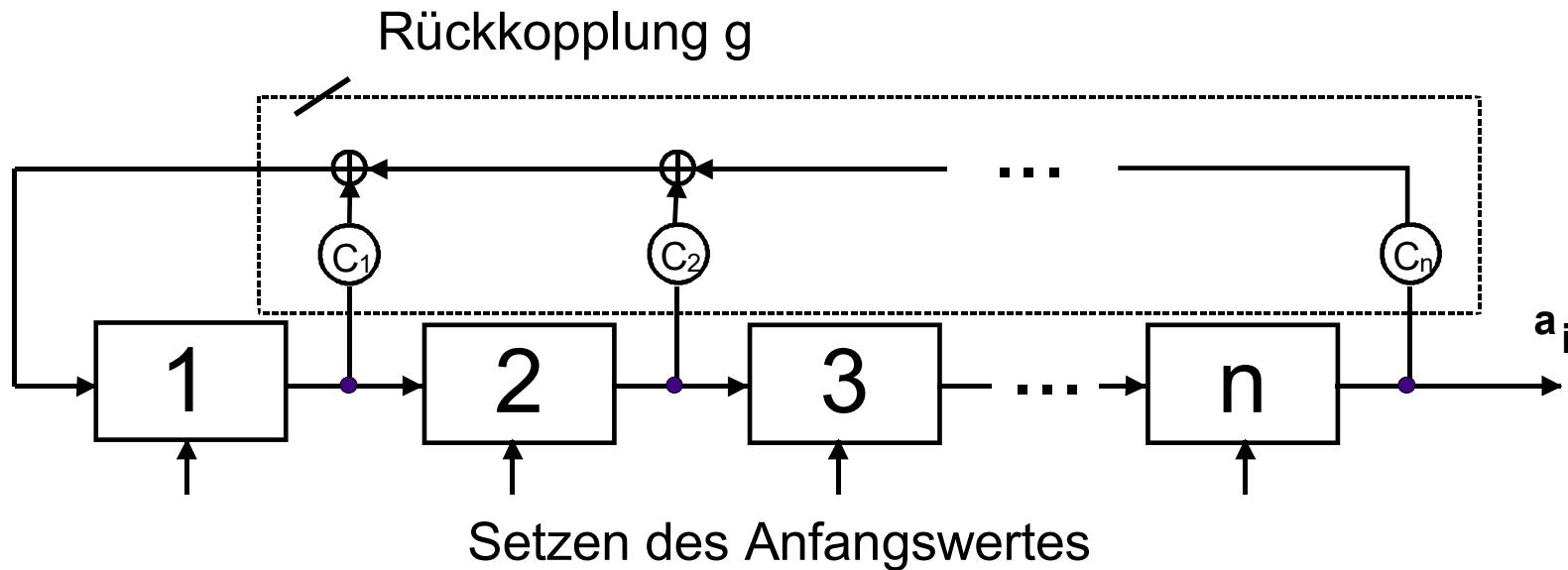
4. Gib das folgende Bit  $b_i$  als  $i$ -tes Zufallsbit aus:

$$b_i = x_i \pmod{2} \in \{0, 1\}$$

- Je nachdem, ob die Rückkopplungsfunktion  $g$  linear oder nichtlinear ist, spricht man von linearen oder nichtlinearen Schieberegistern.
  - Die maximale Periode eines  $n$ -stufigen Schieberegisters ist  $2^n$ .
  - Periodische Zufallsfolgen mit einer möglichst großen Periode bewirken gleichzeitig eine gute statistische Verteilung.
  - Die Länge  $N$  des kürzesten linear-rückgekoppelten Schieberegisters, durch das die Erzeugung einer vorgegebenen Zufallszahlenfolge  $a$  ersetzt werden kann, heißt lineare Komplexität der Folge  $a$ .
  - Wenn  $2 \cdot N$  aufeinanderfolgende Output-Bits eines  $N$ -stufigen linear-rückgekoppelten Schieberegisters bekannt sind, können alle nachfolgenden Output-Bits vorausgesagt werden.
-

# Zufallszahlengenerator

## Linear-rückgekoppeltes Schieberegister



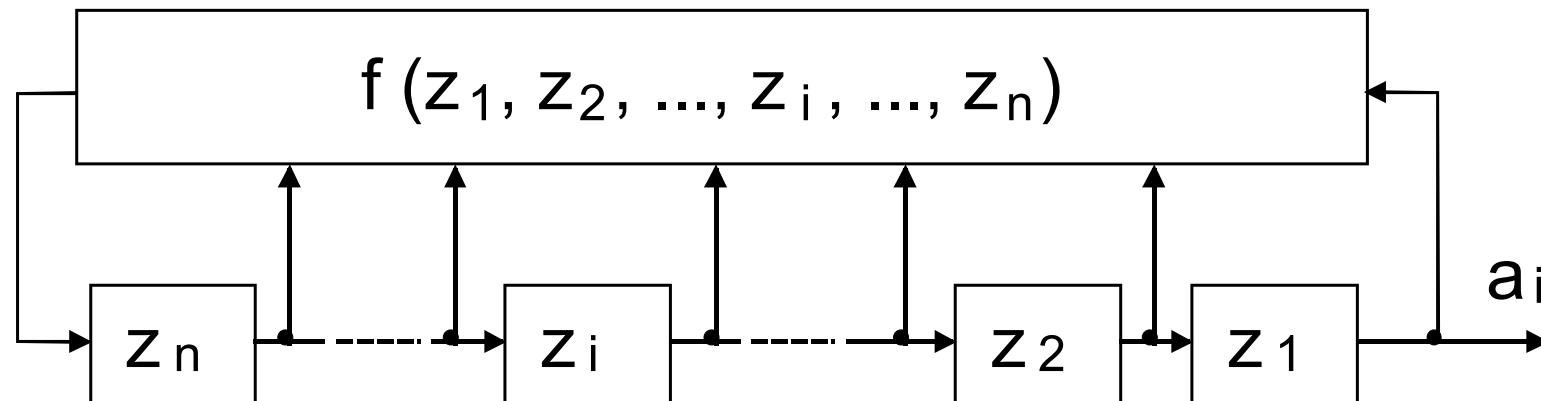
$g$  = Rückkopplungsfunktion ( $g = C_1 \cdot z_1 \oplus C_2 \cdot z_2 \oplus \dots \oplus C_n \cdot z_n$ )

$n$  = Anzahl der Stufen

$C_i \in 0 \vee 1$ ; je nachdem, ob entspr. Rückführung vorhanden

$\oplus$  = XOR-Verknüpfung bzw. modulo-2-Addition

### Prinzip des Linear Feedback Shift Register (LFSR)



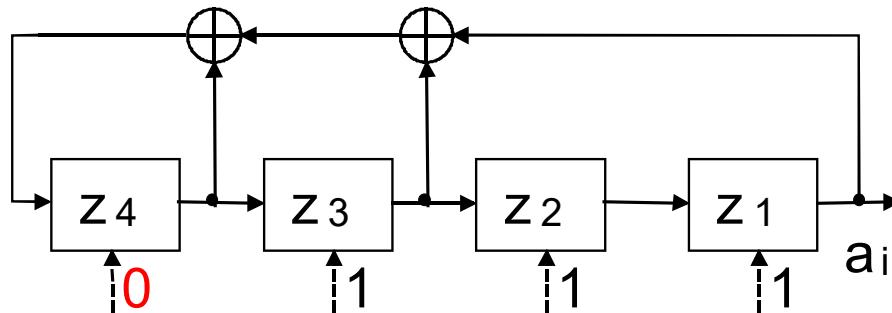
- In den Zellen  $z_1$  bis  $z_n$  des n-stufigen LFSR können die Binärwerte 0 oder 1 gespeichert werden.
- Bei jedem Berechnungsschritt werden die Inhalte der Zellen  $z_n$  bis  $z_2$  nach rechts geschoben.

- Der Zelleninhalt  $z_n$  wird dabei durch den Wert der binärwertigen Funktion  $f(z_1, z_2, \dots, z_i, \dots, z_n)$  ersetzt.
  - Der Zelleninhalt  $z_1$  geht verloren und kann als binäre Pseudozufallsziffer  $a_i$  betrachtet werden.
  - Damit die maximale Periodenlänge erreicht wird, wird bei LFSR die Rückkopplung durch speziell ausgewählte Zelleninhalte realisiert.
  - Die Verknüpfung der rückgekoppelten Zelleninhalte geschieht durch XOR-Bildung bzw. Addition modulo 2.
  - Liegt bei einem n-stufigen LFSR eine Ausgabefolge von  $2^n$  Bit vor, so lässt sich das Rückkopplungsnetzwerk rekonstruieren.
  - Das Finden eines n-stufigen LFSR mit **maximaler Periode** lässt sich zurückführen auf das Finden eines primitiven Polynoms vom Grad n.
-

# Pseudozufallszahlengeneratoren

## Lineare Schieberegister

Beispiel:



LFSR:

- 1. Spalte von T bestimmt die Positionen der Rückkopplung (hier 4, 3 und 1).
- Restliche Spalten beschreiben Verschiebung um eine Position nach rechts (Einheitsmatrix!).

→ Lineare Transformation

$$T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Startvektor:  $x = (0, 1, 1, 1) \Rightarrow$

Folgevektoren:  $(0, 0, 1, 1), (1, 0, 0, 1), (0, 1, 0, 0), (1, 0, 1, 0), (1, 1, 0, 1), (1, 1, 1, 0), (0, 1, 1, 1), (0, 0, 1, 1), \dots$   
 $\Rightarrow a_i = \{1, 1, 1, 0, 0, 1, 0, \dots\}, d = 7.$

### Diskussion des Beispiels:

- Die Zustandsmenge  $X$  eines LFSR der Länge  $n$  lässt sich durch 0-1-Vektoren  $x$  der Form  $x = (b_n, b_{n-1}, \dots, b_2, b_1)$  darstellen.
- Die Funktion  $f$  kann als lineare Transformation  $f(x) = x \cdot T$  aufgefasst werden, wobei  $T$  ist eine binäre  $n \times n$ -Matrix ist.
- Alle anfallenden Operationen werden modulo 2 ausgeführt – dies entspricht einer binären XOR-Verknüpfung.
- Bezeichnet  $x$  den Initialvektor des LFSR, so wird die Zustandsfolge  $x, x \cdot T, x \cdot T^2, x \cdot T^3, \dots$  generiert.
- Die maximal erreichbare Periodenlänge beträgt  $d = 2^n - 1$ .

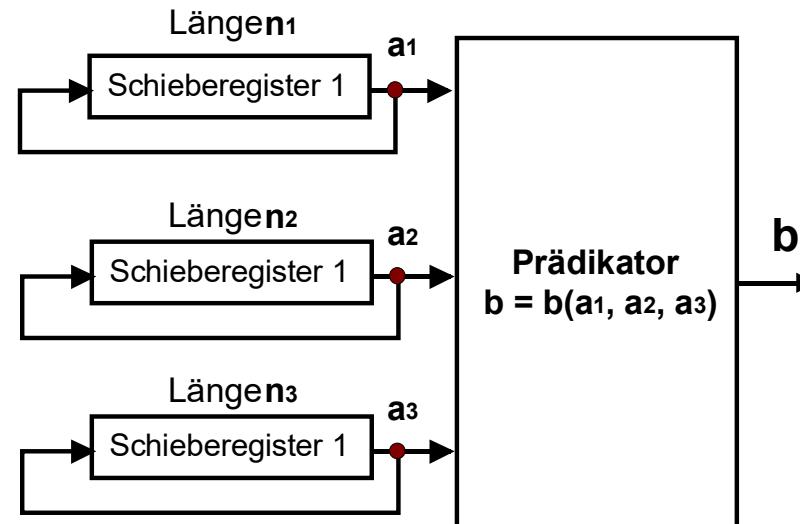
Der Geffe-Generator besteht aus drei zueinander primitiven, nicht-linear-rückgekoppelten Schieberegistern der Länge  $n_1$ ,  $n_2$  und  $n_3$ , d. h. ihre Produktdarstellungen enthalten keine gemeinsamen Faktoren.

Periode P:

$$P = \text{kgV}(2^{n_1} - 1, 2^{n_2} - 1, 2^{n_3} - 1)$$

Lineare Komplexität N:

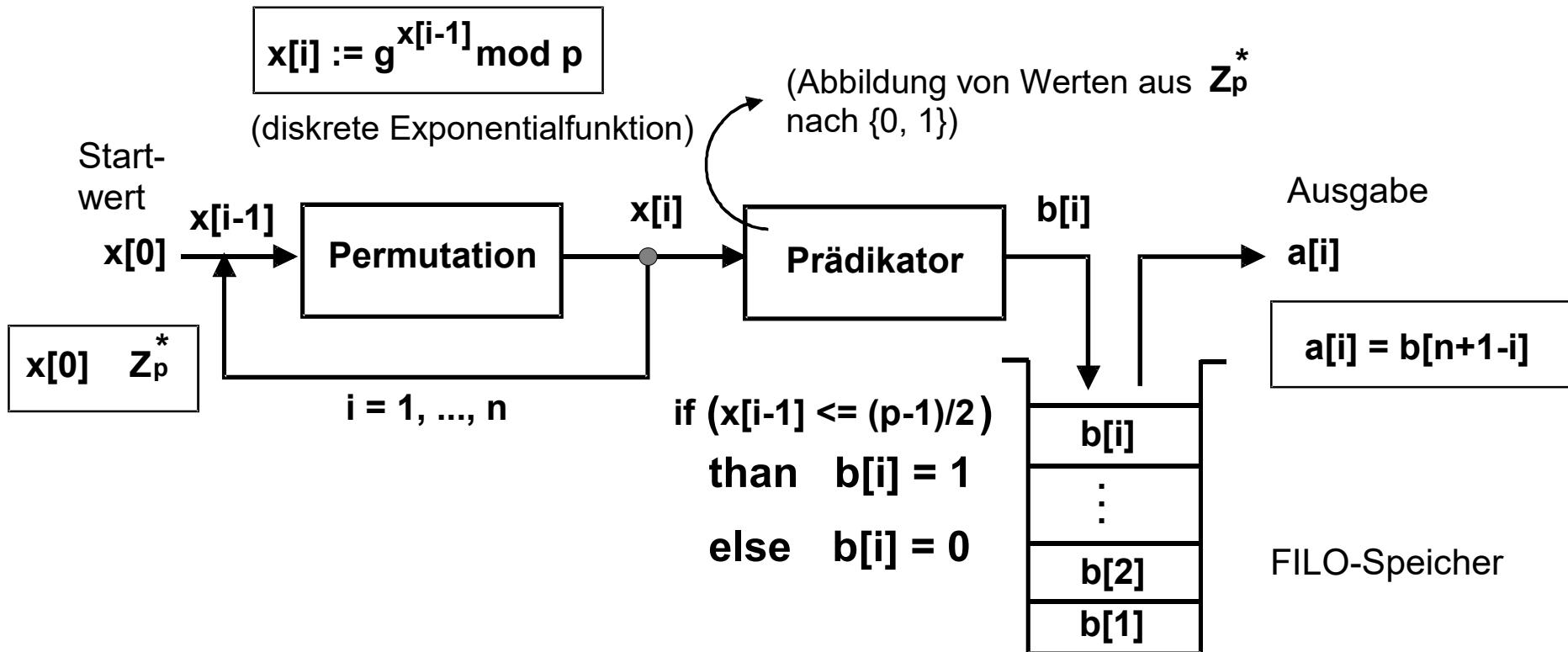
$$N = n_3 * n_1 + (n_1 + 1) * n_2$$



Prädikator:  $b = a_3 \oplus \bar{a}_1 * (a_2 \oplus a_3)$

# Zufallszahlengenerator

## Blum-Micali-Generator



### Notation:

- $p$  (Modulus) sei eine Primzahl;  $p \in \mathbf{P}$
- $n$  ist die Ausgabelänge;  $n \in \mathbf{N}$
- $\mathbf{Z}_n^*$  ist eine **multiplikative Gruppe** (des Restklassenrings  $\mathbf{Z}_n$ ) bestehend aus den Elementen von  $\mathbf{Z}_n$ , die zu  $n$  **teilerfremd** sind.

$$\mathbf{Z}_n^* := \left\{ a \in \mathbf{Z}_n \setminus \{0\} \mid \text{ggT}(a, n) = 1 \right\}$$

- $g$  sei **Primitivwurzel** aus  $\mathbf{Z}_p^*$ ; Zahl  $g$  sollte so gewählt werden, dass sie eine möglichst große Untergruppe von  $\mathbf{Z}_p$  erzeugt (vgl. DL-Problem).
  - $X[0]$  ist Saat aus  $\mathbf{Z}_p^*$
-

# Blum-Micali-Generator

Algorithmus

---

## Ablauf:

```
var x[0:n] : array of integer ;
           b[1:n] : array of {0, 1} ;

for i = 1 to n do
begin
    x[i] = gx[i-1] mod p ;          /* Berechne nächsten Zwischen- */
    if (x[i-1] ≤ (p - 1)/2) then b[i] = 1 /* Berechne das Prädikat mit */
                                  else b[i] = 0 /* Hilfe von x[i - 1] */

end ;
```

## Output:

```
for i = 1 to n do
begin
    a[i] = b[n + 1 - i] ;          /* Ausgabe in umgekehrter Reihenfolge */
    output a[i] ;
end ;
```

---

Typische Beurteilungskriterien sind:

- Periode P
- Lineare Komplexität N
- Statistische Eigenschaften wie
  - Häufigkeitsverteilung von Bits und Bitgruppen
  - Korrelation zwischen Bitfolgen (z. B. zwischen Klar- und Schlüsseltextrn)
  - Mittelwerttests
  - Abstände zwischen dem zweimaligen Auftreten eines Bitmusters
  - uvm.

## Kap. 7: Schlüsselmittelmanagement und Zufallszahlen

### Teil 4: Schlüsselmanagement

- Der Diffie-Hellman-Schlüsselaustausch (DH)
- Schlüsselhierarchie und Schlüsselklassen

### Generelle Anforderungen:

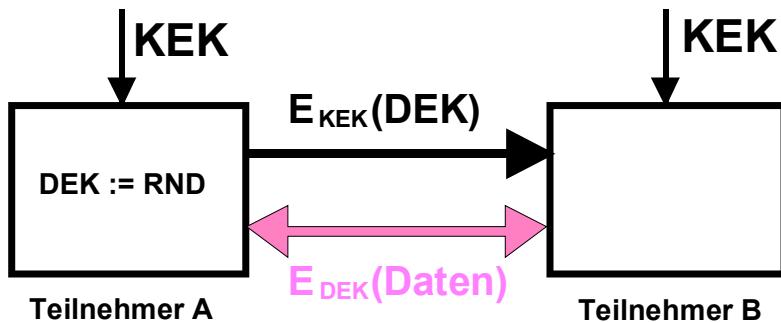
- Es muß gewährleistet sein, daß der ausgetauschte Schlüssel nur den befugten Teilnehmern bzw. Prozessen zugänglich ist.
  - Die auszutauschenden Schlüssel müssen den befugten Teilnehmern unverändert und fehlerfrei zur Verfügung stehen.
  - Bereits benutzte Schlüssel dürfen kein zweites Mal verwendet werden.
  - Schlüsselaustauschprotokolle dürfen den Schlüsselaustausch nicht merklich verzögern.
  - Der Schlüsselabsprache muß eine Authentifikation der Kommunikationspartner vorausgehen.
  - Empfangsbestätigung und Verifikation des abgesprochenen Schlüssels sind in das verwendete Protokoll zu integrieren.
-

- Der für die Nachrichtenverschlüsselung verwendete Schlüssel (sog. **Session Key** oder **Data Encryption Key DEK**) sollte möglichst häufig wechseln, damit keine Analysen oder eingespielte Wiederholungen möglich sind.
  - Der zur Verschlüsselung anderer Schlüssel verwendete Schlüssel heißt **Master Key** oder **Key Encryption Key, kurz KEK**.
  - So wird in der Praxis mit dem KEK zunächst ein DEK verschlüsselt, mit dem anschließend der Datentransfer gesichert wird.
  - Schließlich ist der **Device Key (DK)** ein ausgezeichneter, gerätespezifischer KEK, der im Rahmen der Geräteinitialisierung eingebracht oder hardwaremäßig im Gerät gespeichert ist.
  - Damit verschiedene Angriffsmöglichkeiten unterbunden werden, ist es sinnvoll, den DEK von beiden Seiten gleichberechtigt zu bestimmen.
-

# Gegenseitige Schlüsselabsprache

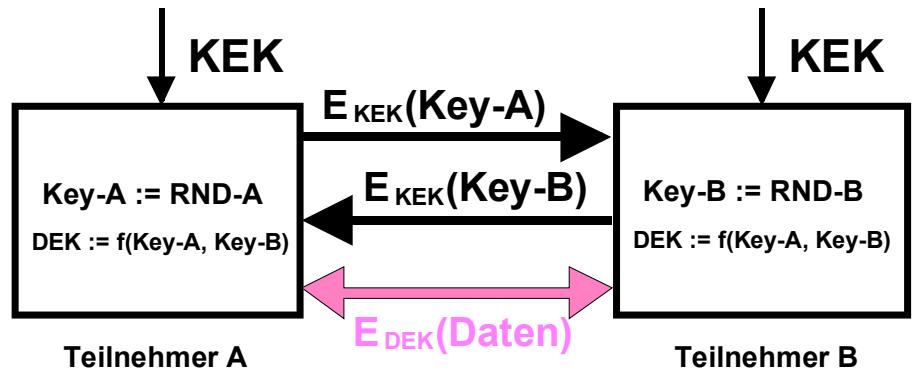
Verfahren

KEK und DEK



$E = \text{Symmetrisches Verfahren}$   
(z. B. DES)

KEK zur Verschlüsselung  
von Teilschlüsseln



$$N = \binom{n}{2} = \frac{n(n - 1)}{2} \sim n^2$$

In ihrer bedeutenden Arbeit haben **W. Diffie** und **M. Hellman** **1976** u. a. ein **asymmetrisches** Verfahren zur Schlüsselabsprache vorgestellt.

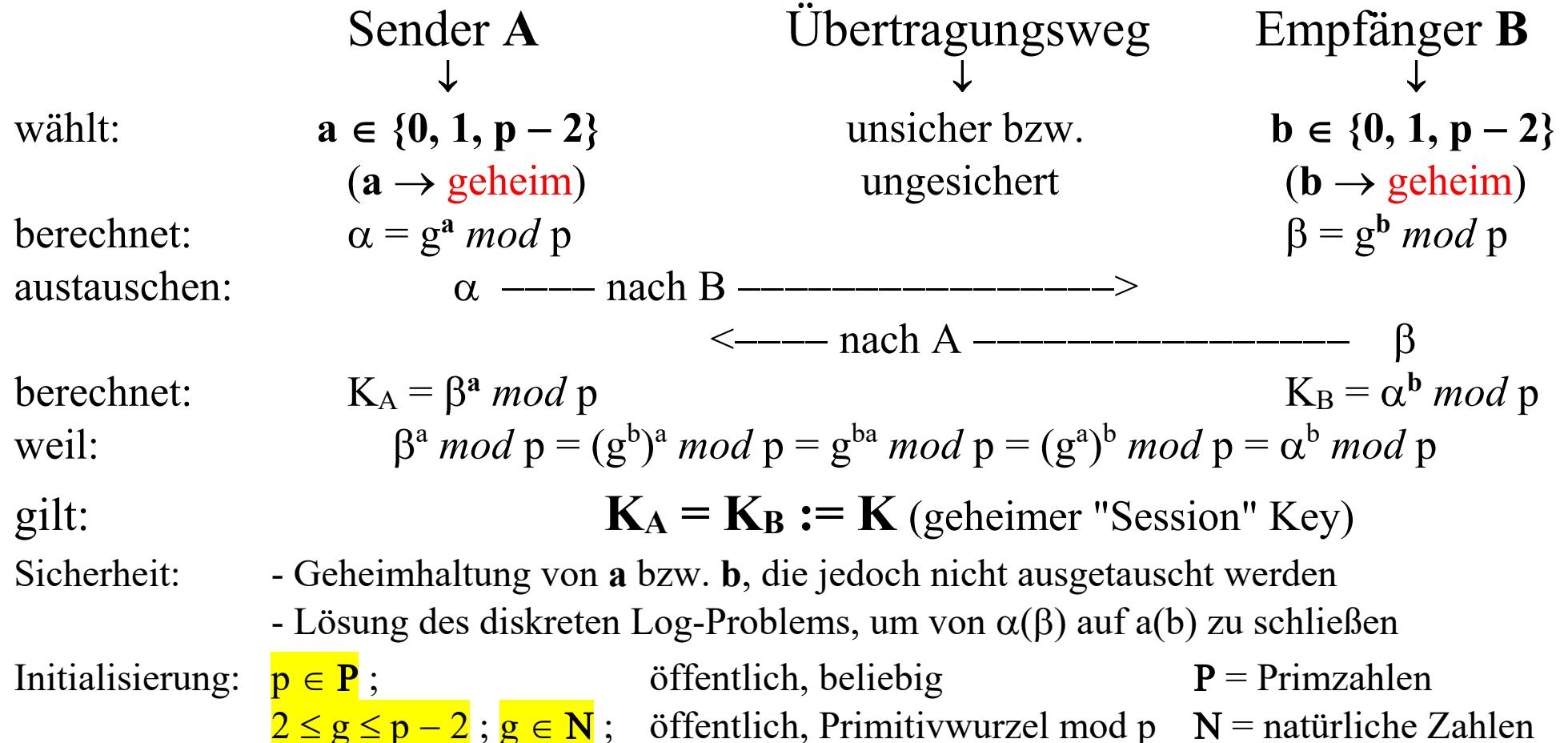
Vorteil: Wie bei asymmetrischen Verfahren üblich, müssen beide Kommunikationspartner von der Schlüsselvereinbarung über keinen gemeinsamen geheimen Schlüssel verfügen.

Nachteil: In der Schlüsselvereinbarung erfolgt keine Authentifikation, d. h. die Kommunikationspartner wissen nicht, mit **wem** sie den Schlüssel vereinbaren.

(Abhilfe schafft hier der Einsatz von Zertifizierungsinstanzen.)

# Schlüsselvereinbarung

## Diffie-Hellman-Verfahren



# Schlüsselklassen

Überblick

---

<b>Schlüsselklasse</b>	<b>Benennung</b>	<b>Schlüssellänge<sup>*)</sup></b>	<b>Lebensdauer<sup>*)</sup></b>
1	Session Key DEK	64 Bit	< 1 Tag
2	Master Key KEK	128 Bit	≈ 1 Monat
3	Device Key	128 Bit	≈ 2 Jahre

<sup>\*)</sup> beispielhaft, abhängig vom konkreten Anwendungsfall

---

---

# **Security**

## **- LV 4120 und 7240 -**

**Kryptographische Protokolle und Anwendungen**

## Kap. 8: Kryptographische Protokolle und Anwendungen

### Teil 1: Authentifikation und digitale Signatur

- Digitale Signaturen in der Praxis
- Authentifikation mit digitaler Signatur

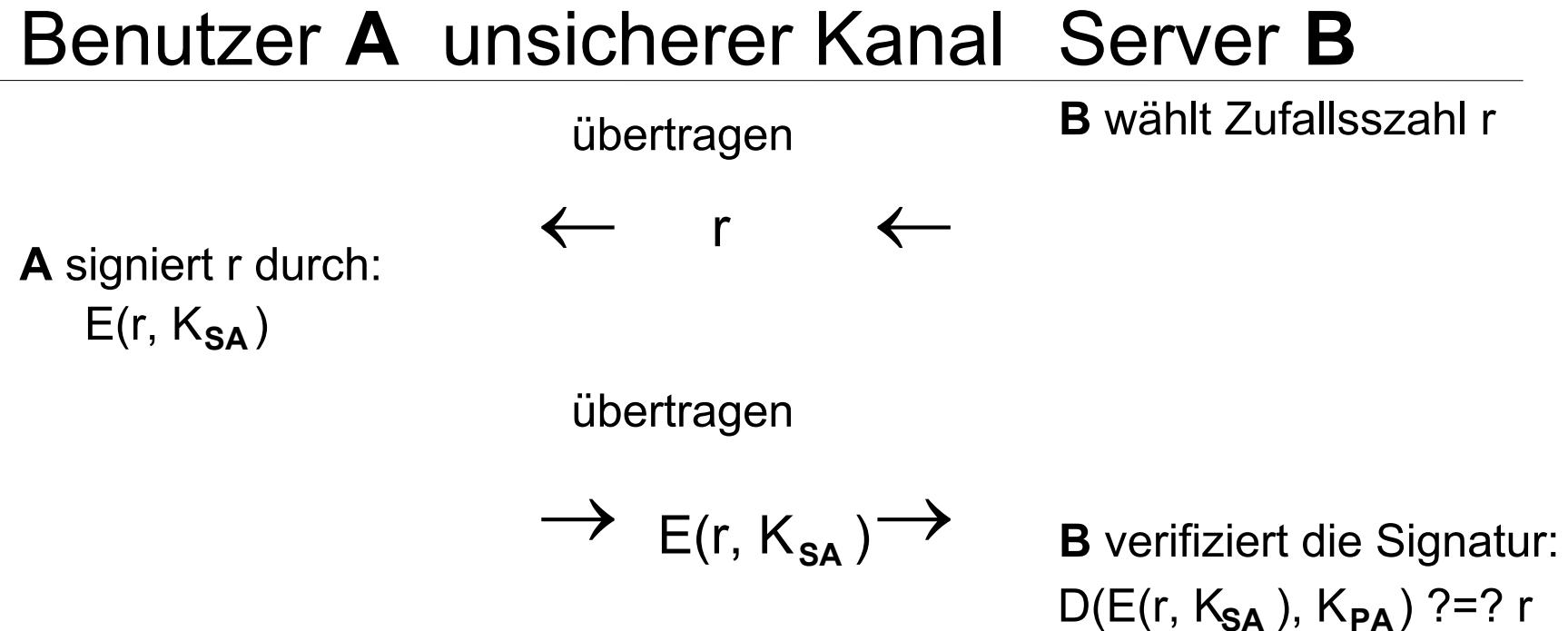
### Digitale Signaturen in der Praxis:

- Absicherung der gesamten Signaturkomponente durch ein möglichst langes Passwort (passphrase).
  - Mit Hilfe dieses Passwortes wird der geheime (private) Schlüssel symmetrisch verschlüsselt und gespeichert.
  - Der öffentliche Schlüssel (eines Kommunikationspartners) wird mittels eines Zertifikats gesichert.
  - Dieses **Zertifikat** trägt die digitale Signatur eines **Trustcenters** oder der sogenannten **Certification Authority (CA)**.
  - Im ersten Schritt gilt es nun mittels öffentlichen CA-Schlüssels das Zertifikat des Kommunikationsteilnehmers zu verifizieren.
-

# Authentifikation und digitale Signatur

# Authentifikation

## **Authentifikation mit digitaler Signatur:**

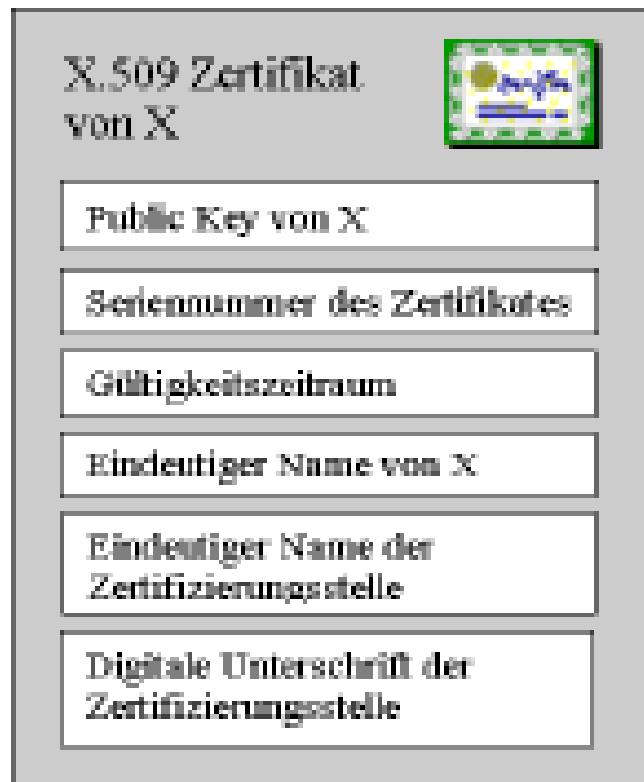


## Kap. 8: Kryptographische Protokolle und Anwendungen

### Teil 2: Public-Key-Infrastruktur

- Prüfung öffentlicher Schlüssel und Trustcenter
- Zertifikatshierarchie

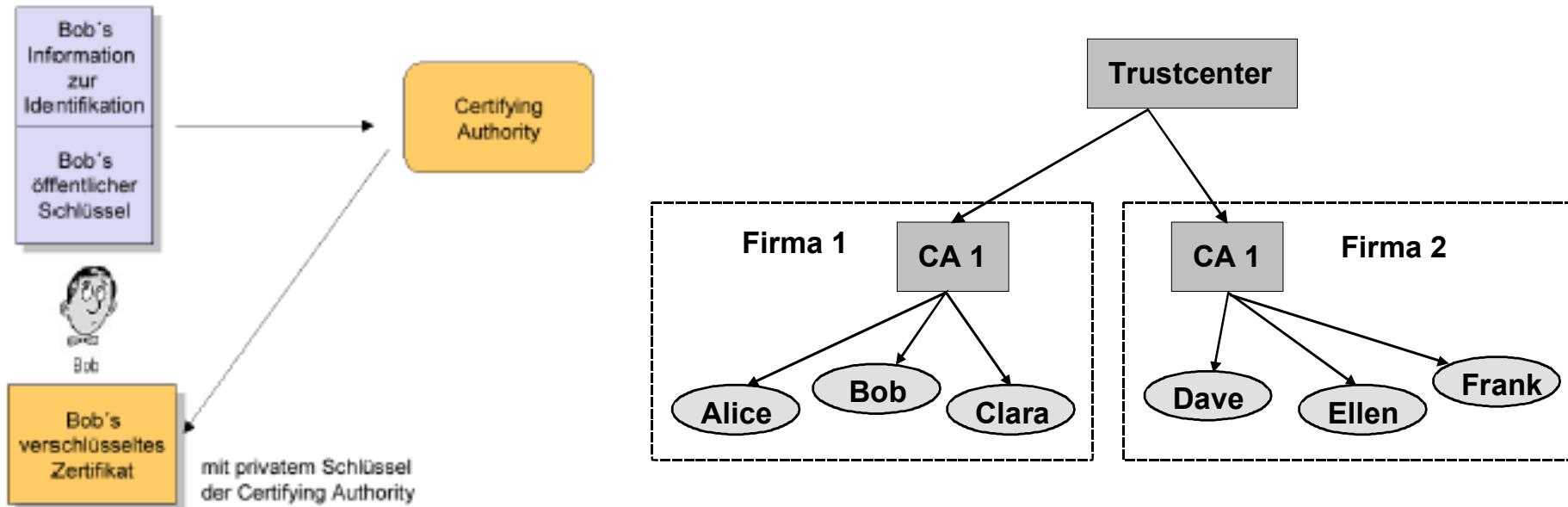
## Prüfung öffentlicher Schlüssel und Trustcenter:



### Bestandteile eines Zertifikats:

- Version
- Seriennummer
- Algorithmus
- Aussteller des Zertifikats
- Geltungsdauer des Zertifikats
- Verwendungszweck
- Öffentlicher Schlüssel
- **Signatur des Ausstellers**

### 128-Bit-Schlüssel und zweistufige Zertifikatshierarchie:



Type	Bits/KeyID	Date	User ID
pub	1024/1C42BD1A	2010/05/29	Berhard Geib <B.Geib_ho@gmx.de>
Key fingerprint	= E4 87 BC 23 A9 77 5D E2 E4 87 BC 23 A9 77 5D E2		

## Kap. 8: Kryptographische Protokolle und Anwendungen

### Teil 3: Secret Sharing und Secret Splitting

- Secret Sharing
- Secret Splitting

- Secret-Sharing-Verfahren wurden bereits 1979 von **Adi Shamir** zur **Aufteilung von geheimen Schlüsseln** eingeführt.
  - Dabei wird ein Geheimnis auf eine Gruppe von  $n$  Personen so aufgeteilt, dass eine beliebige Teilgruppe von  $t$  Personen mit  $t < n$  das Geheimnis rekonstruieren kann,  $t - 1$  oder weniger jedoch nicht.
  - Das Einrichten der sogenannten Shares (Teilgeheimnisse) erfolgt von einer vertraunswürdigen Instanz, die auch Verteiler oder Dealer genannt wird.
  - Die Rekonstruktion wird von einem Zusammensetzer oder Combiner ausgeführt, der im Namen der Teilgruppe das Geheimnis berechnet und allen  $t$  Gruppenmitgliedern mitteilt.
  - Die Gruppe aller Teilnehmer sei durch  $\{P_1, \dots, P_n\}$ ,  $n \in \mathbf{N}$  gegeben.
-

### (t, t)-Schwellenwertverfahren:

Geheimnis  $k \in \mathbf{N}$  soll auf Teilnehmer  $\{P_1, \dots, P_t\}$ ,  $t \in \mathbf{N}$  verteilt werden.

#### Verteiler:

1. Der Verteiler wählt den Modulus  $m \in \mathbf{N}$  mit  $m > k$ .
2. Der Verteiler wählt zufällig  $t - 1$  Elemente  $s_1, \dots, s_{t-1} \in \mathbf{Z}_m$  als Shares für die Teilnehmer  $P_1, \dots, P_{t-1}$ .
3. Der Verteiler berechnet dann das Share für den Teilnehmer  $P_t$  mit Hilfe von

$$s_t = (k - \sum_{i=1}^{t-1} s_i) \bmod m$$

4. Der Verteiler verteilt die Shares sicher an die Teilnehmer  $P_1, \dots, P_t$ .
-

### Combiner:

5. Der Combiner erhält auf sicheren Wege die Shares  $s_1, \dots, s_t$  von den jeweiligen Teilnehmern  $P_1, \dots, P_t$  der Gruppe.
6. Der Combiner berechnet das Geheimnis  $k$  mit der Vorschrift

$$k = (\sum_{i=1}^t s_i) \bmod m$$

7. Der Combiner teilt das Geheimnis  $k$  allen Teilnehmern  $P_1, \dots, P_t$  mit.

Mit  $t - 1$  oder weniger Teilnehmern kann  $k$  nicht berechnet werden, da für die fehlende  $s_i$  jede Zahl aus  $\mathbb{Z}_m$  denkbar ist. Das Verfahren ist somit **perfekt**.

---

- Secret-Splitting ist das Zerteilen einer Bitfolge (Nachricht, Dokument) in zwei oder ggf. mehrere Teile, die alle für sich allein betrachtet wertlos sind und keine Information über die Nachricht  $M$  enthalten.
- Fügt man die einzelnen Teile (sagen wir  $M_1$  und  $M_2$ ) aber zusammen, so ist die Rekonstruktion der Nachricht  $M$  möglich.
- Hat  $M$  die Länge  $n$ , so nimmt man eine  $n$  Bit lange Zufallszahl  $r$  und berechnet:

$$M_1 = r \oplus M \quad \text{und} \quad M_2 = r$$

- Ist  $r$  echt zufällig, so ist die Aufteilung absolut sicher, genau wie das **One-Time-Pad** und es gilt:

$$M = M_1 \oplus M_2$$

- Damit ein Spezialfall des allgemeinen  $(t, n)$ -Schwellenwertproblems.
-

## **Kap. 8: Kryptographische Protokolle und Anwendungen**

### **Teil 4: Zero-Knowledge-Protokolle**

- Challenge-and-Response-Verfahren
- Das Fiat-Shamir-Protokoll

### Die Idee des Challenge-and-Response-Verfahrens:

- Das Protokoll des **Herausforderns und Antwortens** dient der **Benutzerauthentifikation**, die gewöhnlich aus einer **Identifikation** und einer sich anschließenden **Verifikation** besteht.
- Dabei wird eine zufällige **Anfrage** (die Challenge) durch eine zugehörige **Response** beantwortet, welche ein Geheimnis benutzt, ohne jedoch nur ein Bit an Information über das Geheimnis preiszugeben.
- Wir gehen davon aus, dass zwei Benutzer **A** und **B** einen gemeinsamen **geheimen** Schlüssel **k** besitzen und setzen voraus, dass sonst niemand diesen Schlüssel kennt.

Zur **Authentifizierung** von **B** gegenüber **A** dient dann folgendes Protokoll:

---

# Zero-Knowledge-Protokoll

## Challenge-and-Response

### Benutzer A

A wählt zwei gleich lange m-Bit Zufallszahlen  $s_1$  und  $s_2$   
 $s := s_1 \parallel s_2 \in \{0, 1\}^{2m}$   
und verschlüsselt s mit k.  
 $r = E(s, k)$

### unsicherer Kanal

übertragen  
 $\rightarrow ID_A, r \rightarrow$

### Benutzer B

B sucht in Datenbank zu ID A gehörigen Schlüssel k und entschlüsselt r mit k.  
 $s = s_1 \parallel s_2 = D(r, k)$   
B zerlegt s in gleich lange  $s_1$  und  $s_2$ .



A entschlüsselt R mit k und erhält hieraus  $(s_1 \oplus s_2) \parallel t = D(R, k)$ .  
A prüft den Wert  $s_1 \oplus s_2$  auf Korrektheit.

übertragen  
 $\leftarrow R \leftarrow$

B wählt eine m Bit lange Zufallszahl  $t \in \{0, 1\}^m$  und verschlüsselt  $(s_1 \oplus s_2) \parallel t$  mit k.  
 $R = E((s_1 \oplus s_2) \parallel t, k)$

### Das Fiat-Shamir-Authentifikationsprotokoll:

Dieses von **A. Fiat**, **A. Shamir** und **U. Feige** entwickelte **Authentifikationsprotokoll** benutzt, wie bei der Signatur mit einem Public-Key-Verfahren, einen **geheimen** Schlüssel **s**.

Benutzer **A** mit dem **geheimen** Schlüssel **s** möchte **B** seine Identität beweisen, ohne jedoch **s** preisgeben zu müssen.

Vorbereitung:

- Ein vertrauenswürdiger Vermittlungsrechner bestimmt zwei zufällige Primzahlen  $p$  und  $q$ , deren Produkt den Modul  $n$  ergibt.
  - Der **geheime** Schlüssel **s** von **A** wird zufällig gewählt.
  - Der **öffentliche** Schlüssel **v** von **A** wird berechnet nach der Vorschrift  $v = s^2 \text{ mod } n$  und öffentlich bekannt gegeben.
-

Nun läuft folgendes **Protokoll** ab:

1. Benutzer **A** wählt Zufallszahl  $r$  und berechnet  $x = r^2 \bmod n$  und schickt  $x$  an Benutzer **B**.
2. Benutzer **B** wählt zufällig ein Bit  $b \in \{0, 1\}$  und schickt dies zum Benutzer **A**.
3. Falls  $b = 1$  berechnet **A** den Wert  $y = r \cdot s \bmod n$  und falls  $b = 0$  den Wert  $y = r \bmod n$ . **A** sendet nun den Wert  $y$  an **B**.
4. Falls  $b = 1$  verifiziert **B**, ob  $y^2 \bmod n = x \cdot v \bmod n$  und falls  $b = 0$  den Wert  $y^2 \bmod n = x$ .

Die **Sicherheit** des Verfahrens basiert auf der Schwierigkeit der Berechnung **modularer Quadratwurzeln**, was gleich schwierig ist wie die **Primfaktorzerlegung** von  $n$ .

---

## Kap. 8: Kryptographische Protokolle und Anwendungen

### Teil 5: Public-Key-Systeme

- PGP
- IPSec

### Benutzer A unsicherer Kanal Benutzer B

---

A wählt zufälligen  
**Sitzungsschlüssel k,**  
verschlüsselt mit **k** die  
Nachricht M und  
verschlüsselt mit **K<sub>PB</sub>** den  
Sitzungsschlüssel **k**.

$$C = E(M, k) \quad \text{übertragen}$$

$$C_k = E(k, K_{PB}) \rightarrow (C_k, C) \rightarrow$$

B entschlüsselt

$$k = D(C_k, K_{SB})$$

$$M = D(C, k)$$

## Pretty good privacy

- **PGP** wurde 1994 von Phil Zimmermann entwickelt und ist ein verbreitetes Programm zur sicheren Kommunikation per E-Mail.
  - Dateien können mit PGP verschlüsselt oder signiert und das Resultat dann per E-Mail verschickt werden.
  - Die Versionen 2.6.x benutzen **IDEA** zum Verschlüsseln, RSA zum Schlüsseltausch und **MD5** als Einweg-Hash-Funktion.
  - Digitale Signaturen werden mit **RSA** und **MD5** erzeugt.
  - Ab Version 5.X.X wird der Schlüsselaustausch mit dem **Diffie-Hellman**-Verfahren realisiert und der **Digital Signature Algorithm (DSA)** zum Signieren verwendet.
-

## IP Security:

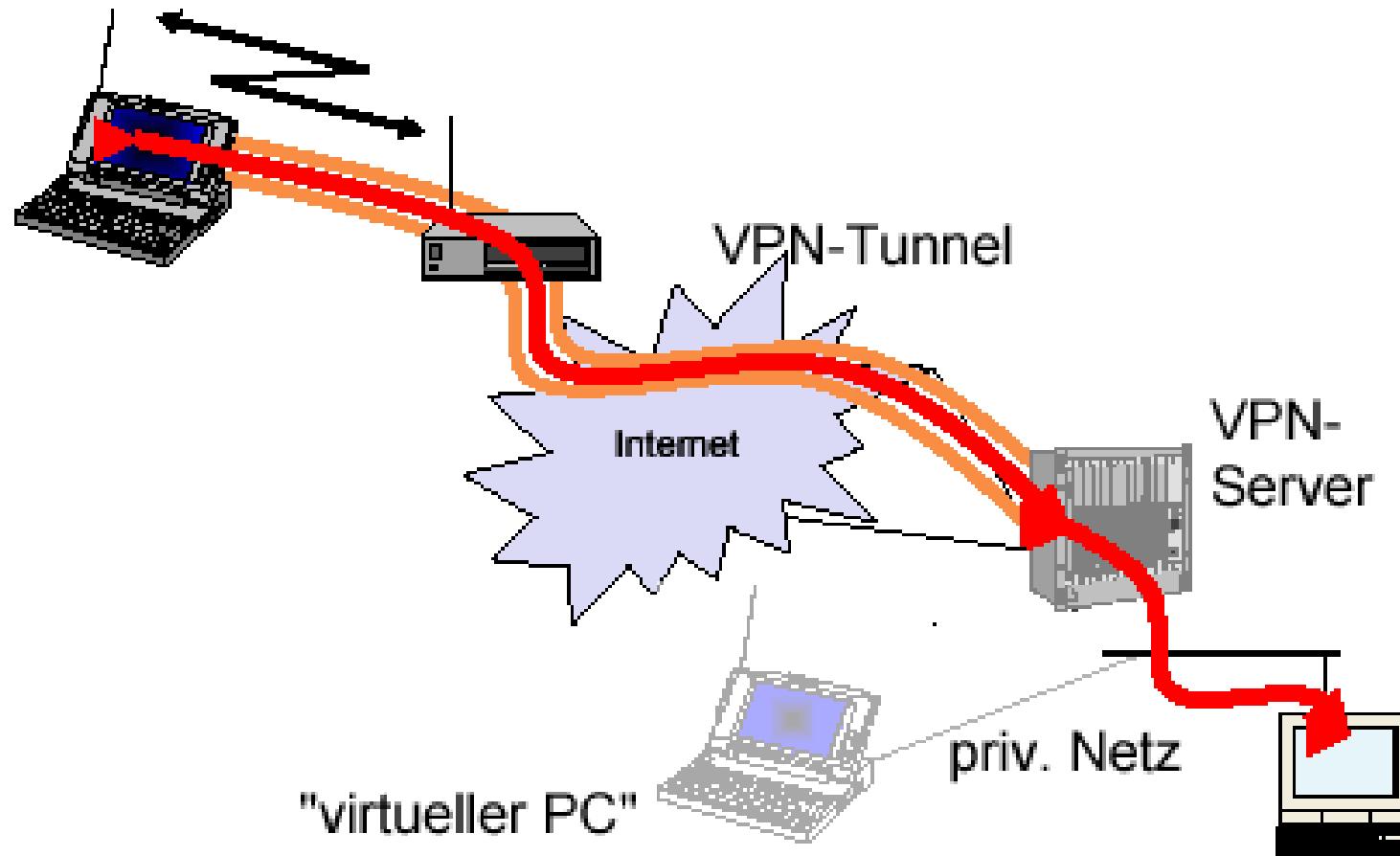
Für die Realisierung von **VPNs** und anderen sicheren Verbindungen wurde 1994 vom **Internet Architecture Board** mit **IP Security (IPSec)** ein mächtiges Protokoll zur Absicherung von **IP-Paketen** initiiert und bis heute weiterentwickelt. Die Verwendung von IPSec auf einer **Firewall** bietet folgende Vorteile:

- Starke kryptographische Sicherheit für den Verkehr nach draußen.
  - IPSec auf einer Firewall kann nicht umgangen werden.
  - Die Anwendersoftware muss nicht angepasst werden.
  - Für den Benutzer ist die Verwendung von IPSec transparent.
  - Verwendete Kryptoverfahren sind: DES und Triple-DES, CBC-Mode, RC5, IDEA, Blowfish, MD5, SHA-1 und Diffie-Hellmann.
-

## Kap. 8: Kryptographische Protokolle und Anwendungen

### Teil 6: Virtual Private Networking

- Realisierung einer Übertragungssicherheit
- SSH und SSL



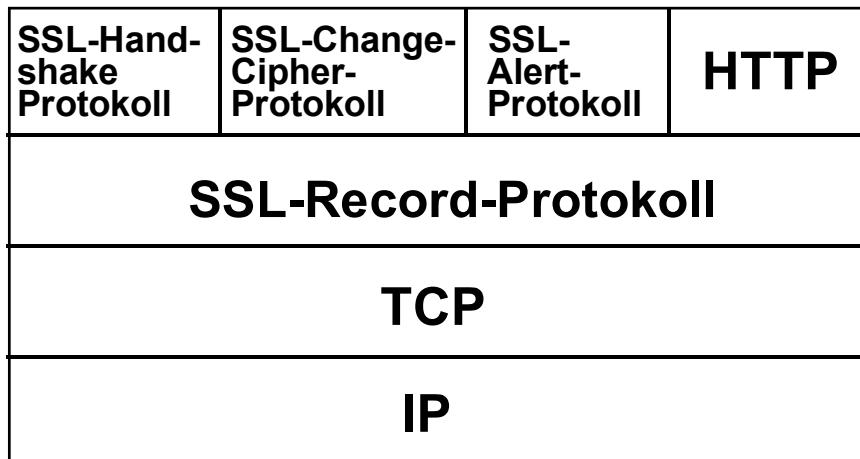
### Secure Shell (SSH):

Die wichtigste Funktionalität ab Version SSH2 ist:

- Sichere gegenseitige Authentifikation von Client und Server.
  - Sicherer Login durch Authentifikation der Benutzer auf dem Server.
  - Mittels RSA oder DSA wird die Identität des Benutzers geprüft.
  - Jede Kommunikation ist verschlüsselt.
  - Mit dem RSA-Algorithmus wird ein Sitzungsschlüssel vereinbart.
  - Mit dem Diffie-Hellmann-Verfahren erfolgt die Schlüsselvereinbarung.
  - Der DSA dient zur Erstellung von Signaturen.
  - Unter Verwendung des Sitzungsschlüssels erfolgt eine symmetrische Verschlüsselung mittels IDEA, Blowfish oder Triple-DES.
-

### Secure socket layer (SSL):

Das **SSL-Protokoll** ist im OSI-Schichtenmodell zwischen der Transportschicht und der Anwendungsschicht eingebettet und dient als Grundlage für die Spezifikation der **Transport Layer Security (TLS)**.



#### SSL-Record-Protokoll:

Kryptoroutinen wie MD5, SHA-1, RSA, DH, IDEA, DES und RC4.

#### SSL-Alert-Protokoll:

Warn- und Fehlermeldungen.

#### SSL-Change-Cipher-Protokoll:

Initialisierung der ausgewählten Routinen.

#### SSL-Handshake-Protokoll:

Schlüsselaustausch und Festlegung der kryptographischen Routinen.