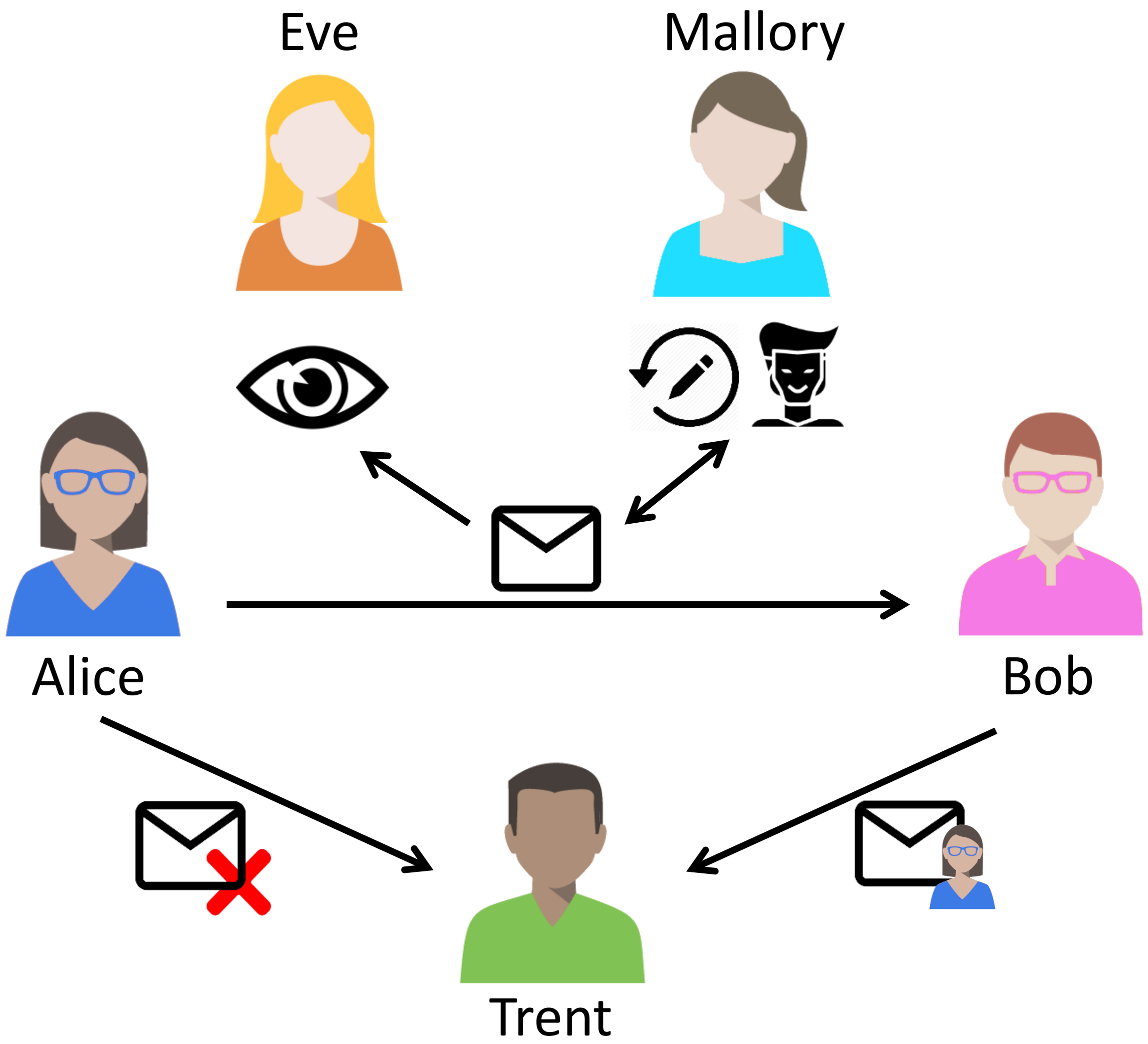
IT-Sicherheit
WiSe 2021/22

Kryptographie

Kryptographie Soap Opera



- **Alice** will Nachricht an **Bob** senden
- **Eve** (Eavesdropper) will Nachricht unbefugt lesen
- **Mallory** (Malicious) will Nachricht unbefugt verändern oder sich als Alice ausgeben
- **Trent** (Trusted Entity) ist eine vertrauenswürdige dritte Instanz, die Meinungsverschiedenheiten zwischen Alice und Bob klärt (z.B. ein Gericht)

Angriffspotential Nachrichtenübertragung

Hochschule Fulda
University of Applied Sciences

Sicherheitsziel	Beschreibung	Werkzeug	
Vertraulichkeit	Eve und Mallory sollen die Nachricht nicht lesen können	Verschlüsselung	Dieses Kapitel
	Bob soll nicht wissen von wem die Nachricht kommt	Anonymisierung	
	Eve und Mallory sollen die Kommunikation nicht sehen	Steganographie	
Integrität	Änderungen der Nachricht von Mallory sollen erkannt werden	Hashfunktionen, Messages Authentication Codes, Digitale Signaturen	Dieses Kapitel
Authentizität	Bob will sichergehen, dass die Nachricht von Alice stammt	Message Authentication Codes, Digitale Signaturen	Dieses Kapitel
Verfügbarkeit	Die Nachricht muss bei Bob ankommen	Redundanz, Content Distribution	
Autorisierung	Andere Nutzende von Alice‘s oder Bob‘s Computer dürfen die Nachricht nicht senden oder sehen	Access Control	
Verbindlichkeit	Alice kann die Nachricht im Nachhinein nicht leugnen	Digitale Signaturen	Dieses Kapitel

Übersicht

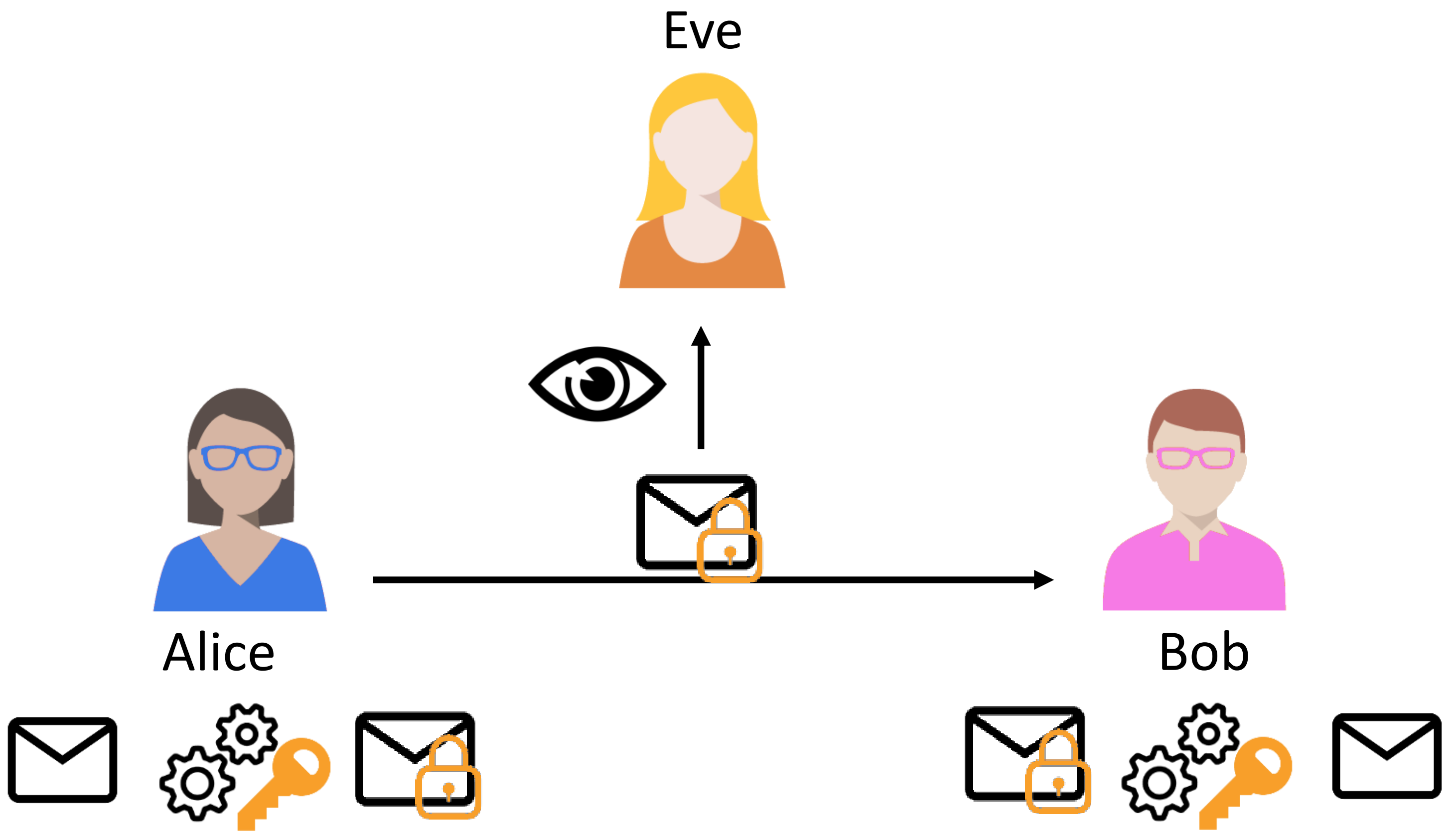
1. Verschlüsselung
    1. Historie
    2. Sicherheit von Kryptographischen Verfahren
    3. Symmetrische Verschlüsselungsverfahren
    4. Asymmetrische Verschlüsselungsverfahren
2. Digitale Signaturen
3. Hashfunktionen
4. Message Authentication Codes (MACs)
5. Zufallszahlengeneration
6. Moderne Themen der Kryptographie

Lernziele Kryptographie

- Verständnis von kryptographischen Verfahren, deren Sicherheitsgarantien und Grenzen
- Grundverständnis des Aufbaus von kryptographischen Verfahren
- Fähigkeit zur praktischen Anwendung von kryptographischen Verfahren

Vertraulichkeit durch Verschlüsselung

- **Bedrohung:** Eve liest die Nachricht mit
- **Ziel:** Personen ohne den entsprechenden Schlüssel können keine Informationen aus verschlüsselter Nachricht gewinnen



Definitionen

Zeichen	Bezeichnung	Erklärung
P	Plaintext	Nachricht im Klartext
C	Ciphertext	Verschlüsselte Nachricht
K_E	Verschlüsselungs-schlüssel	Schlüssel der zum Verschlüsseln der Nachricht verwendet wird.
K_D	Entschlüsselungs-schlüssel	Schlüssel der zum Entschlüsseln der Nachricht verwendet wird. Muss basierend auf K_E berechnet werden ($K_D = f(K_E)$).
K	Ver/Entschlüsselungs-schlüssel	Wird verwendet falls $K_E = K_D$
$C = Enc_K(P)$	Verschlüsselungs-funktion	Verschlüsselt den Plaintext P zum Ciphertext C unter Verwendung des Schlüssels K
$P = Dec_K(C)$	Entschlüsselungs-funktion	Entschlüsselt den Ciphertext C zum Plaintext P unter Verwendung des Schlüssels K. Es gilt: $P = Dec_K(Enc_K(P))$.

Caesar Chiffre [SECENG]

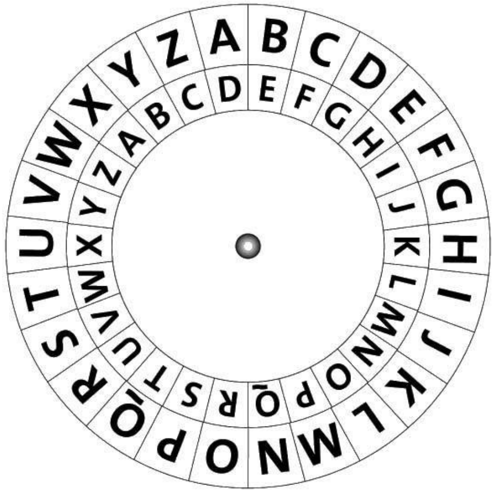
- Wurde von Gaius Julius Caeser genutzt um Nachrichten zu verschlüsseln

- Ersetze jeden Buchstaben mit dem Buchstaben K Positionen weiter hinten im Alphabet

- Beispiel für $K = 3$:
  - $Enc_K(P)$: Ersetze in P jedes $A \rightarrow D, B \rightarrow E, C \rightarrow F, \ldots$
  - $Dec_K(C)$: Ersetze in C jedes $D \rightarrow A, E \rightarrow B, F \rightarrow C\ \ \ldots$

- Gegeben Ciphertext C = NWFA NAVA NAUA
  - Wie lauten der Plaintext P und Schlüssel K $(\neq 3)$?

- Anzahl der Schlüssel: 26



Drehscheibe einer Caeser Chiffre mit $K = 3$

Monoalphabetische Substitution [SECENG]

Hochschule Fulda
University of Applied Sciences

- Ersetze jeden Buchstaben durch einen im Alphabet. Schlüssel ist die Permutation.
  - $Enc_K(P)$: Ersetze in P jedes $A \rightarrow F, B \rightarrow R, C \rightarrow A$, …
  - $Dec_K(C)$: Ersetze in C jedes $F \rightarrow A, R \rightarrow B, A \rightarrow C$ …

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

- Gegeben K wie in der Tabelle oben und $C = FQAT\ NQ, PZIL\ SETL$?
  - Wie lautet der Plaintext P?

- Anzahl Schüssel: 26! = 403,291,461,126,605,635,584,000,000 = 4.03E+26

- Sicherheit → Siehe Übung

Polyalphabetische Substitution
Vigenère Chiffre [SECENG]

- Jeder Buchstabe repräsentiert eine Zahl ($A = 0, B = 1, C = 2, D = 3, \dots, Z = 25$)
  - Addition von Buchstaben als Addition/Subtraktion der Zahlen modulo 26
  - Verschlüsselung Beispiel: $C + Z \; mod \; 26 = 2 + 25 \; mod \; 26 \equiv 1 \; mod \; 26 = B$
  - Entschlüsselung Beispiel: $A - C \; mod \; 26 = 0 - 2 \; mod \; 26 \equiv 24 \; mod \; 26 = Y$

- Vigneré Verschlüsselung mittels Addition mit **expandiertem Schlüsselwort**:

```
P = DERFRUEHEVOGELFAENGTDENWURM
K = SCHLUESSELSCHLUESSELSCHLUES
C = VGYQLYWZIGGILWZEWFKEVGUHOVE
```

- Anzahl der Schlüsselkombinationen abhängig von der Schlüsselwortlänge

- Sicherheit → Siehe Übung

Polyalphabetische Substitution
One-Time Pad [SECENG]

- **One-Time Pad:** Polyalphabetische Substitution wobei P und K gleich lang sind

- One-Time Pad Verfahren bietet **Perfekte Geheimhaltung**
  - Jeder Plaintext kann in jeden Ciphertext verschlüsselt werden (und umgekehrt)
  - Unmöglich nur mit dem Ciphertext Rückschluss auf den Plaintext zu ziehen

```
            C = JUYMVOOL

K1 = DQRYEWOZ           K2 = JATUCOBI
P1 = GEHORSAM           P2 = AUFSTAND
```

- Das Verfahren ist allerdings nicht praxistauglich, da der Schlüssel…
  - … gleich lang sein muss wie die Nachricht (potentiell mehrere Gigabyte)
  - … wirklich zufällig generiert werden muss (mehr zu Zufallszahlengeneration später)

Angriffsmodelle (1/3)
Enigma und Stärkere Angriffe

Hochschule Fulda
University of Applied Sciences

- **Bisher:** Angreifer*in kennt nur Ciphertext C – **Ciphertext-Only Angriff**

- Enigma wurde im 2. Weltkrieg zur Verschlüsselung genutzt
  - Analyse war aufgrund komplexer Rotormechanik sehr schwierig

- Um Ciphertexte zu entschlüsseln nutzen die Alliierten verschiedene Tricks:
  - Teile des Plaintexts waren bekannt (Datum, Absendername,…) – **Known Plaintext Angriff**
  - Inhalt bestimmter Nachrichten konnte frei gewählt werden (Kriegsschiff hat an Position X gehalten) – **Chosen Plaintext Angriff**

- ➔ Moderne Verschlüsselung muss viele Angriffsmöglichkeiten berücksichtigen

Angriffsmodelle (2/3)
Modellierung von Angriffen

Hochschule Fulda
University of Applied Sciences

- Angriffsmodelle formalisieren Möglichkeiten von Angreifer*innen
  - Abstraktion vom Anwendungsfall durch Modellierung
  - Angreifer*in bekommt ein „Orakel“, um ergänzende Informationen zu erhalten

- Beispiel Chosen-Plaintext Modell
  1. Verschlüsselungsorakel wählt einen geheimen Schlüssel K
  2. Eve (Angreifer*in) wählt n Plaintexte P_i und sendet diese an das Orakel
  3. Orakel berechnet $C_i = Enc_K(P_i)$ für $1 \leq i \leq n$ und sendet die n Ciphertexte C_i an Eve
  4. Eve wählt Plaintexte m_0 und m_1 mit $m_0 \neq m_1 \neq P_i$ für $1 \leq i \leq n$ und sendet sie an das Orakel
  5. Das Orakel wählt ein zufälliges Bit $b \in \{0,1\}$ und sendet $C = Enc_K(m_b)$ an Eve
  6. Eve rät welcher Plaintext $m_{b'}$ verschlüsselt wurde und gibt b' aus

- Ein Verfahren gilt als sicher, wenn $b = b'$ nur mit vernachlässigbar höherer Wahrscheinlichkeit als ~50%

Angriffsmodelle (3/3)
Bekannte Modelle und Beispiele

Hochschule Fulda
University of Applied Sciences

- Beispiel: Login auf Seite eines Webmail Providers (z.B. T-Online, GMX)
  - Kontext: Eve ist im selben Raum wie Alice und fängt alle verschlüsselten Pakete ab
  - Ziel von Eve: Abfangen der verschlüsselten Zugangsdaten von Alice

Angriffsmodell	Beschreibung	Beispiel Szenario
Ciphertext-Only	Nur der Ciphertext ist bekannt	Nur verschlüsselte Zugangsdaten sind bekannt
Known-Plaintext	Eve erhält zufällige Plaintext/Ciphertext Paare	Alice surft auf öffentlichem Teil der Webseite und loggt sich dort später auf Konto ein
Chosen-Plaintext	Eve hat Zugriff auf ein Verschlüsselungsorakel, das beliebige Plaintexte verschlüsselt	Eve sendet eMail an Alice. Alice loggt sich ein und ruft Eve's eMail ab.
Chosen-Ciphertext	Eve hat Zugriff auf ein Entschlüsselungsorakel, das beliebige Ciphertexte entschlüsselt	Eve hat für begrenzte Zeit Zugriff auf Alice's Gerät mit verschlüsselter Sitzung (ohne bestehendes Login) und lässt sich gefälschte Pakete entschlüsseln. Alice kommt später wieder und loggt sich auf Webseite ein.

Sicherheit von Krypto Verfahren (1/3)

- Die meisten Krypto Verfahren gelten als ungebrochen solange **Brute-Force** der effizienteste Angriff ist
  - Brute-Force: Testen aller möglichen Schlüsselkombinationen
  - Komplexität steigt exponentiell in der Schlüssellänge

- Verschiedene Stufen des „Brechens" von Krypto Verfahren
  - **Theoretisch Gebrochen**: Ein effizienterer Angriff als Brute-Force wird bekannt
  - **Überholt**: Der Angriffsaufwand fällt unter eine erreichbare Schranke
  - **Praktisch Gebrochen**: Ein Angriff wurde demonstriert

- Damit ein Verfahren als gebrochen gilt muss der Angriff auf das Verfahren zielen, nicht auf die Implementierung

Sicherheit von Krypto Verfahren (2/3)
Rechnerische Sicherheit

Hochschule Fulda
University of Applied Sciences

- **Rechnerische Sicherheit**: Ein Krypto Verfahren ist zwar theoretisch zu brechen, praktisch existieren aber nicht genug Zeit oder Ressourcen

	Anzahl Bits Schlüssel	Anzahl der Schlüssel-kombinationen	Zeitaufwand für Brute-Force in Jahren (Annahme: Pro Kombination eine Operation der gesamten Top500 Supercomputer mit 2.8E^18 Operationen pro Sekunde in Juni 2021)	Speicheraufwand für Kombinationen in Faktor derzeit anfallenden Datenmenge (Annahme: Pro Kombination 16 byte, Datenmengen derzeit sind 79 Zetabytes = 10^21 Bytes)
	8	256	0	0
	32	4,294,967,296	0	0
	64	1.84E+19	0	0
Unsicher	80	1.21E+24	0.01	244
Sicher	128	3.40E+38	3.85E+12	6.89E+16
	192	6.28E+57	7.11E+31	1.27E+36
	256	1.58E+77	1.31E+51	2.35E+55

Sicherheit von Krypto Verfahren (3/3)
Standardisierung und Kryptoanalysen

- Wie kann garantiert werden, dass ein Verfahren auch wirklich sicher ist?
  - Wurden Angriffe beim Design übersehen? [Tews12]
  - Hat Entwickler*in Hintertüren in das Verfahren eingebaut? [BD+21]

- **Kerckhoffs Prinzip:** Die Sicherheit des Verfahrens muss auf der Geheimhaltung des Schlüssels beruhen anstatt der Geheimhaltung des Verfahrens selbst.

- Krypto Verfahren werden heutzutage via öffentlicher Ausschreibung standardisiert:
  - Jede Person darf ein Verfahren einreichen
  - Verfahren müssen Rahmenbedingungen einhalten (z.B., Transparentes Design, Schlüssellänge)
  - Öffentliche Kryptoanalyse konformer Verfahren
  - Gewinner wird aus Menge der übrigen sicheren und effizienten Verfahren ausgewählt

AES - Advanced Encryption Standard
Entstehung

- 1997 wurde die Suche nach einem Nachfolger des Data Encryption Standards (DES) von der US Behörde NIST weltweit öffentlich ausgeschrieben

- 15 Vorschläge wurden bis 1998 eingereicht und von der wissenschaftlichen Community öffentlich analysiert:
  - Prüfung auf Einhaltung der formalen Kriterien (Schlüssellänge, Eingabegrößen, …)
  - Prüfung auf Schwachstellen
  - Prüfung der Effizienz und Umsetzbarkeit

- 5 Vorschläge kamen in die nächste Runde (Rjindael, MARS, RC6, Twofish, Serpent)

- Im Jahr 2000 wurde **Rjindael** als Gewinner gekürt und als AES standardisiert

AES - Advanced Encryption Standard
Aufbau und Varianten

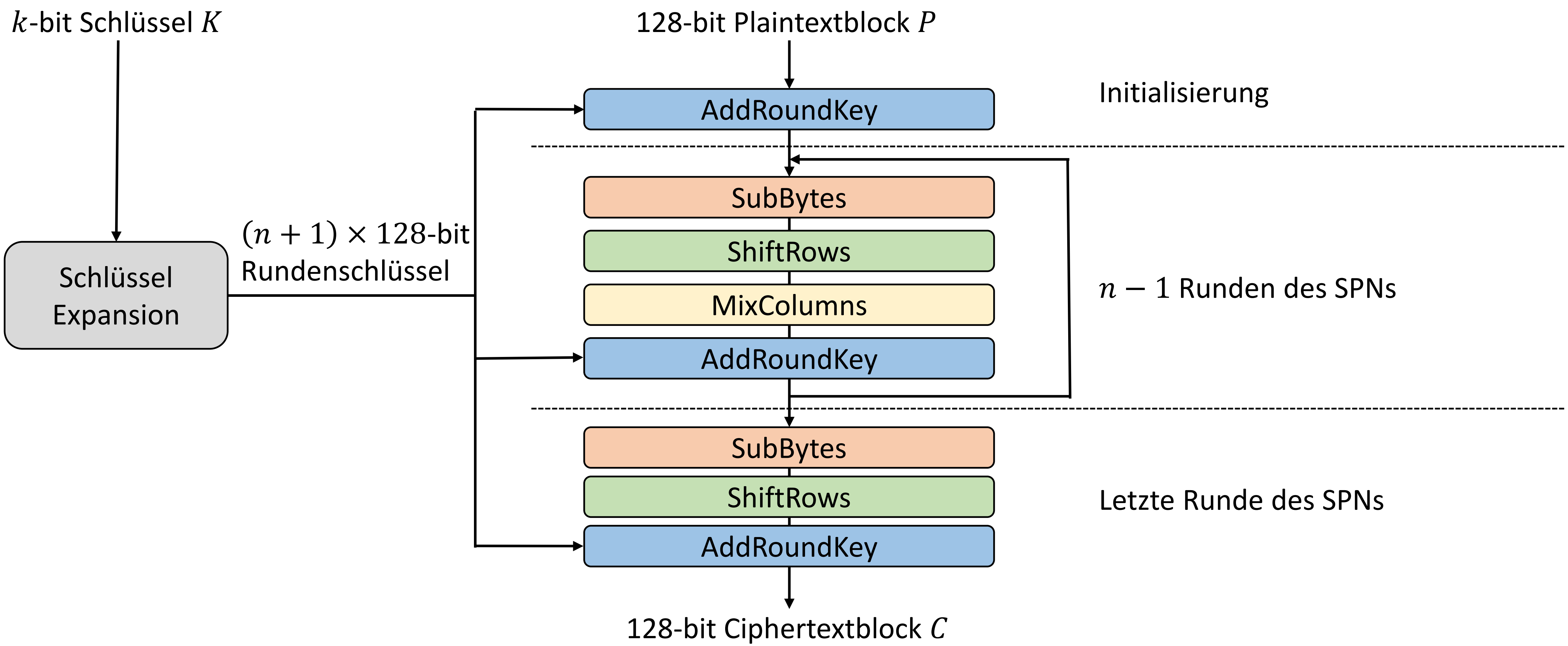
Hochschule Fulda
University of Applied Sciences

- AES ist eine **Blockchiffre**, die auf 128-bit Blöcken arbeitet
  - **Blockchiffre:** Der Plaintext wird in Blöcke eingeteilt und blockweise verarbeitet
  - **Stromchiffre:** Zeichen werden einzeln verarbeitet (z.B., Caesar Chiffre, Vigneré Chiffre)

- Die Blöcke werden in n Runden durch ein Substitutions-Permutations-Netzwerk (SPN) verschlüsselt

- Es existieren drei AES Varianten mit Schlüssellänge k und Rundenanzahl n:
  - AES-128: k = 128-bit Schlüssel mit n = 10 Runden
  - AES-192: k = 192-bit Schlüssel mit n = 12 Runden
  - AES-256: k = 256-bit Schlüssel mit n = 14 Runden

AES - Advanced Encryption Standard
Überblick zur Verschlüsselung



AES - Advanced Encryption Standard
Interne Repräsentation

128-bit (16-byte) Eingabeblock

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte 16

$\downarrow$ Repräsentationswechsel (keine interne Operation)

$$\begin{bmatrix} \text{Byte 1} & \text{Byte 2} & \text{Byte 3} & \text{Byte 4} \\ \text{Byte 5} & \text{Byte 6} & \text{Byte 7} & \text{Byte 8} \\ \text{Byte 9} & \text{Byte 10} & \text{Byte 11} & \text{Byte 12} \\ \text{Byte 13} & \text{Byte 14} & \text{Byte 15} & \text{Byte 16} \end{bmatrix}$$

128-bit Interner Status als 4x4 Matrix

AES - Advanced Encryption Standard
AddRoundKey

Input: 128-bit Interner Status

Input: 128-bit Rundenschlüssel → AddRoundKey

Output: 128-bit Interner Status

Input Status | **Rundenschlüssel** | **Output Status**

$$
\begin{bmatrix} B1 & B2 & B3 & B4 \\ B5 & B6 & B7 & B8 \\ B9 & B10 & B11 & B12 \\ B13 & B14 & B15 & B16 \end{bmatrix}
\oplus
\begin{bmatrix} R1 & R2 & R3 & R4 \\ R5 & R6 & R7 & R8 \\ R9 & R10 & R11 & R12 \\ R13 & R14 & R15 & R16 \end{bmatrix}
=
\begin{bmatrix} B1 \oplus R1 & B2 \oplus R2 & B3 \oplus R3 & B4 \oplus R4 \\ B5 \oplus R5 & B6 \oplus R6 & B7 \oplus R7 & B8 \oplus R8 \\ B9 \oplus R9 & B10 \oplus R10 & B11 \oplus R11 & B12 \oplus R12 \\ B13 \oplus R13 & B14 \oplus R14 & B15 \oplus R15 & B16 \oplus R16 \end{bmatrix}
$$

AES - Advanced Encryption Standard
SubBytes

Input: 128-bit Interner Status

↓

SubBytes

↓

Output: 128-bit Interner Status

Input Status

B1 = 0x01	B2 = 0xFF	B3 = 0x03	B4
B5	B6	B7	B8
B9	B10	B11	B12
B13	B14	B15	B16

Substitution-Box (**S-Box**:
Array mit 256 x 8-bit Werten)

	ID	Wert
	0x00	0x63
B1 →	0x01	0x7c
	0x02	0x77
B3 →	0x03	0x7b
	….	…..
	0xFE	0xbb
B2 →	0xFF	0x16

Output Status

0x7c	0x16	0x7b	…
…	…	…	…
…	…	…	…
…	…	…	…

AES - Advanced Encryption Standard
ShiftRows

Input: 128-bit Interner Status

$\downarrow$

ShiftRows

$\downarrow$

Output: 128-bit Interner Status

Input Status

B1	B2	B3	B4
B5	B6	B7	B8
B9	B10	B11	B12
B13	B14	B15	B16

0 Spalten nach links →

1 Spalte nach links →

2 Spalten nach links →

3 Spalten nach links →

Output Status

B1	B2	B3	B4
B6	B7	B8	B5
B11	B12	B9	B10
B16	B13	B14	B15

AES - Advanced Encryption Standard
MixColumns

Input: 128-bit Interner Status

↓

MixColumns

↓

Output: 128-bit Interner Status

MixColumns Matrix · **Input Status** · **Output Status**

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} B1 & B2 & B3 & B4 \\ B5 & B6 & B7 & B8 \\ B9 & B10 & B11 & B12 \\ B13 & B14 & B15 & B16 \end{bmatrix} = \begin{bmatrix} O1 & O2 & O3 & O4 \\ O5 & O6 & O7 & O8 \\ O9 & O10 & O11 & O12 \\ O13 & O14 & O15 & O16 \end{bmatrix}$$

$$O1 = 2 \times B1 \oplus 3 \times B5 \oplus B9 \oplus B13$$

Operationen in Galios Feld $\mathrm{GF}(2^8)$!

AES - Advanced Encryption Standard
Schlüssel Expansion und Entschlüsselung

- Die Schlüssel Expansion ist ähnlich aufgebaut wie die Verschlüsselung:
  - Substitution via AES S-Box
  - Permutation der Bytes
  - XOR mit vorherigen Spalten bzw. Konstanten

- Die Entschlüsselung funktioniert „rückwärts“ mit invertierten Operationen
  - **AddRoundKey:** Keine Änderung
  - **SubBytes:** Invertierte S-Box
  - **ShiftRows:** Zeilen um gleichen Wert nach rechts schieben
  - **MixColums:** Multiplikation mit invertierter Matrix

AES - Advanced Encryption Standard
Performance

Hochschule Fulda
University of Applied Sciences

- AES Implementierungen in SW und HW sind sehr effizient
  - SubBytes: Array Lookup
  - AddRoundKey, MixColumns: XOR, UND, und Shift
  - ShiftRows: Index Handling
- Moderne Prozessoren enthalten AES-Befehlssätze
  - AES New-Instructions (NI): Ein Befehl pro Runde
- AES Verschlüsselung ist selten das Bottleneck!
  - Puffer- und Pakethandling ist häufig langsamer

Verfahren	Aufrufe pro Sek.
64-bit Mult.	1,401,372,784
AES-128 (SW)	22,413,312
AES-128 (HW)	175,308,800

Single Thread in Ubuntu VM mit Crypto++ und konstantem Schlüssel

AES - Advanced Encryption Standard
Sicherheit

- AES ist bisher nur theoretisch gebrochen [TW15]:
  - AES-128: Angriff mit Komplexität 2^126
  - AES-192: Angriff mit Komplexität 2^190
  - AES-256: Angriff mit Komplexität 2^254
- Angriffe gegen AES mit reduzierten Runden [BR19]:
  - AES mit 5 Runden und 2^32 selbst gewählten Ciphertexten
- Viele Angriffe auf Implementierungen von AES
  - Angriffe setzen (indirekten) Zugriff auf Cache oder Stromverbrauch voraus
- AES gilt weiterhin als sicher

Weitere Symmetrische Chiffren

- Spezielle Verschlüsselungsverfahren für verschiedene Anwendungen

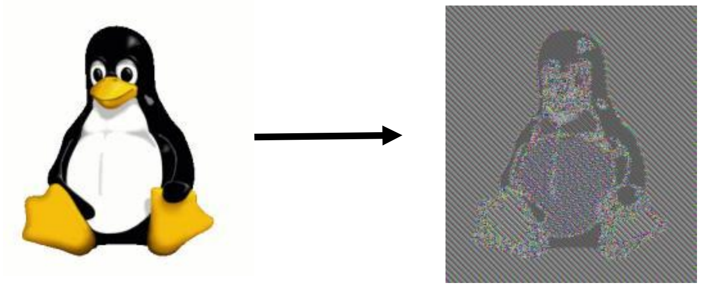
Verfahren	Schlüssellänge [bit]	Eingabeblockgröße [bit]	Kommentar
AES	128/192/256	128	Weiteste Verbreitung
DES/3DES	56/112	64	Vorgänger von AES. Unsicher!
ChaCha20	256	Stromchiffre	Sehr effizient in Software
RC4	Variabel bis 2048	Stromchiffre	Sehr effizient in Software. Unsicher!
Serpent/Twofish	128/192/256	128	Verbreitete Finalisten aus der AES Challenge
PRESENT	80/128	64	Optimiert auf Größe in Hardware
PRINCE	128	64	Optimiert für Echtzeit Verschlüsselung in Hardware
LowMC	128/256	128/192/256	Optimiert für kryptographischen Protokolle

Betriebsmodi von Blockchiffren
Electronic Code Book (ECB)

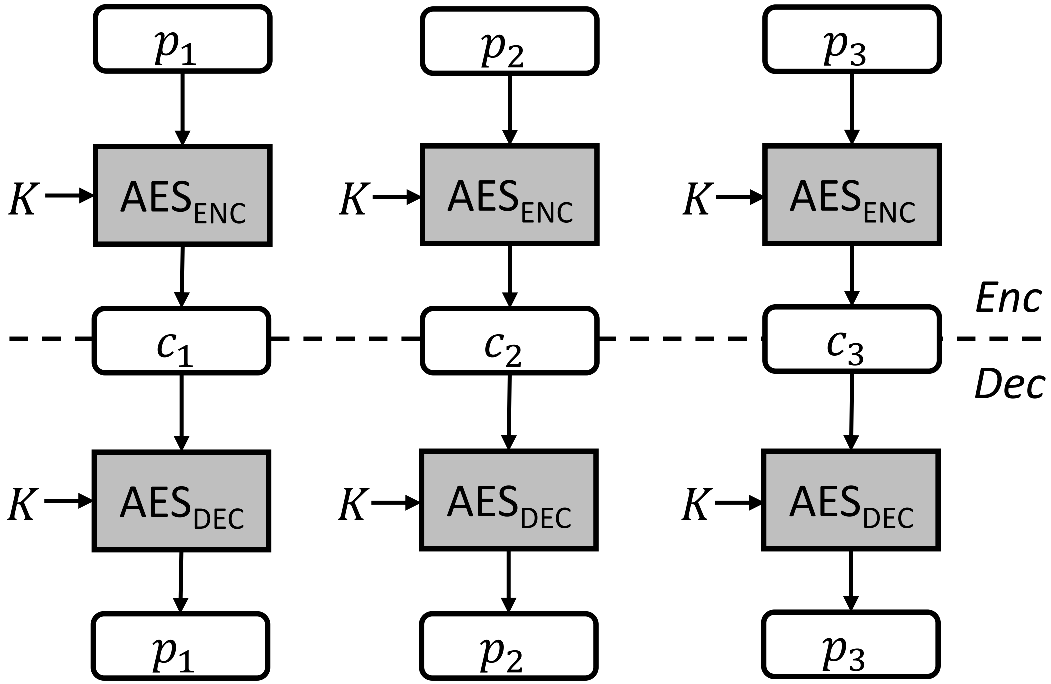
Hochschule Fulda
University of Applied Sciences

- AES verarbeitet Plaintext $P = p_1 p_2 p_3$ in 128-bit Blöcken $p_1 - p_3$

- Ähnlich wie monoalphabetische Chiffren:
  - Gleiche Eingabe → Gleiche Ausgabe



- Spezielle Betriebsmodi sind notwendig!



Ver/Entschlüsselung mit AES im ECB Modus

Betriebsmodi von Blockchiffren

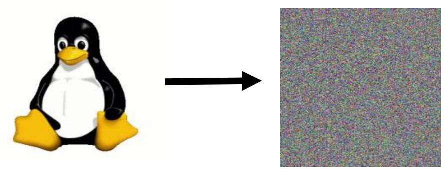
Hochschule Fulda
University of Applied Sciences

- Blockchiffen können in verschiedenen Modi betrieben werden:

Name	Bezeichnung	Einsatzgebiet
ECB	**Electronic Code Book**	Einsatz in Nischen oder wenn nur ein Block verschlüsselt werden muss
CBC	**Cipher Block Chaining**	Verschlüsselung der Datenübertragung
CFB	Cipher Feedback Mode	Verschlüsselung mit Fehlerresistenz bei der Datenübertragung
CTR	**Counter Mode**	Verschlüsselung mit Fehlerresistenz; Macht aus Blockchiffre eine Stromchiffre
XTS	Ciphertext Stealing	Festplattenverschlüsselung; Besonders gesichert gegen Angriffe auf Implementierung
GMAC/CMAC	**Galois/Cipher Message Authentication Mode**	Authentifikation von Daten (Abschnitt „Message Authentication Codes“)
GCM	**Galois-Counter Mode**	Verschlüsselung und Authentifikation von Daten (Abschnitt „Message Authentication Codes“)

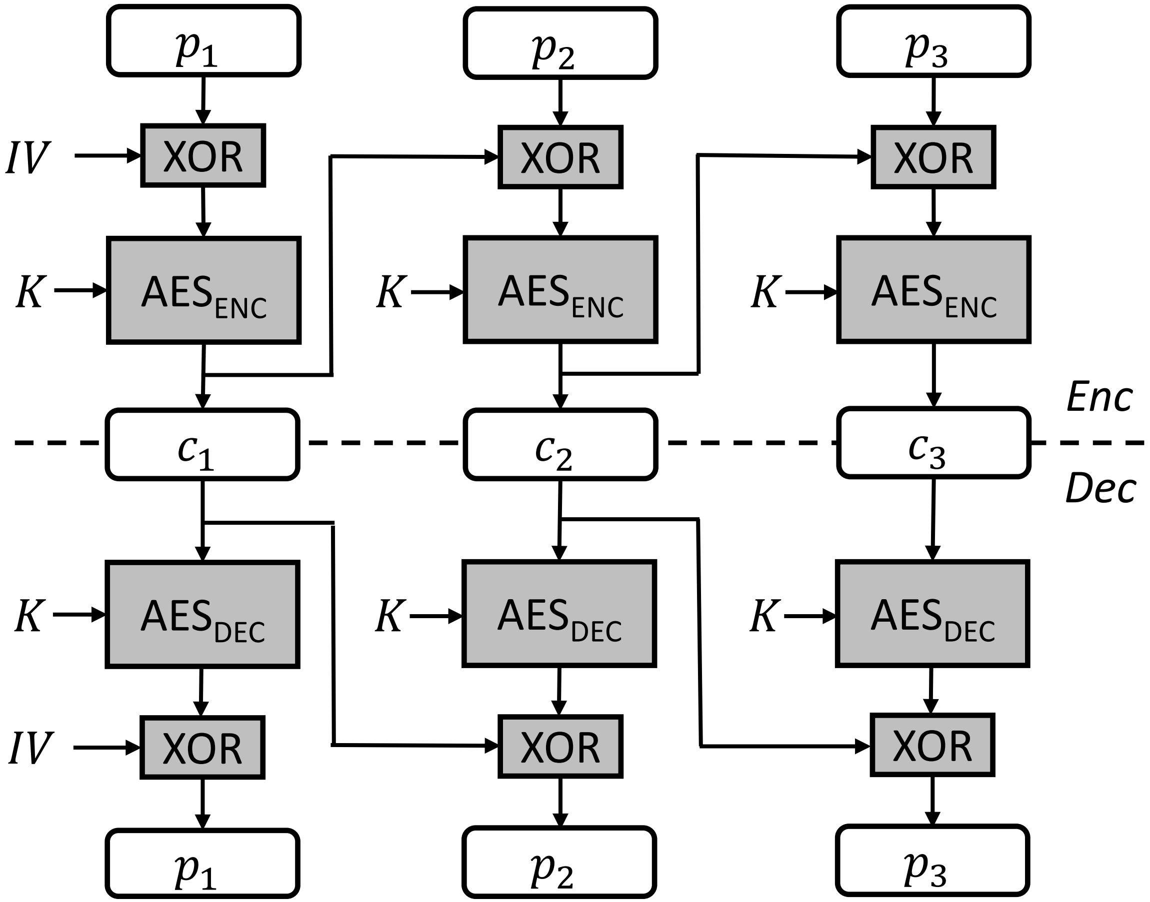
Betriebsmodi von Blockchiffren
Cipher Block Chaining (CBC)

- Ciphertext des vorherigen Blocks fließt in nächsten Block mit ein (via XOR)



- Zufälliger Initialisierungsvector (IV) um gleiche Plaintexte $P_1 = P_2$ zu unterschiedlichen Ciphertexten $C_1 \neq C_2$ zu verschlüsseln

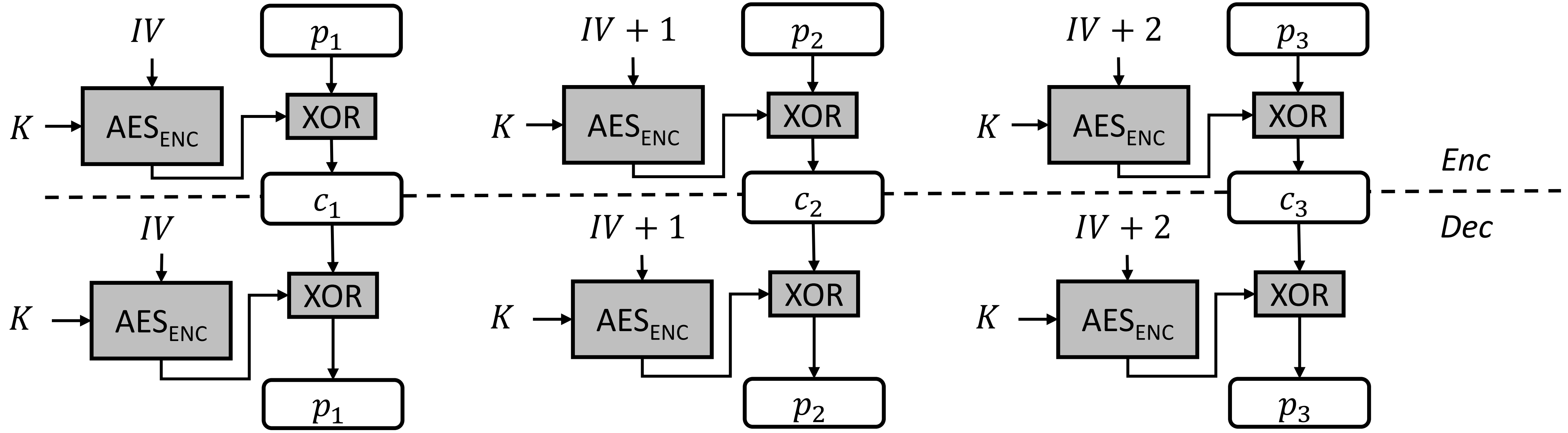
- Nachteile:
  - Nicht parallelisierbar
  - Übertragungsfehler zerstören folgenden Block



Ver-/Entschlüsselung mit AES im CBC Modus

Betriebsmodi von Blockchiffren
Counter (CTR)

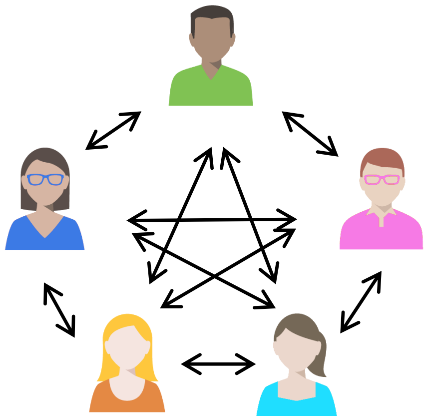
- Zufälliger IV wird verschlüsselt und mit Plaintext ver-XORed
  - Hochgradig parallelisierbar und AES kann vorberechnet werden (Stromchiffre)
  - Übertragungsfehler wirken sich nur auf lokalen Block aus



Ver-/Entschlüsselung mit AES im CTR Modus

Symmetrische vs. Asymmetrische Verschlüsselungsverfahren

- Bei AES benötigen beide Parteien den gleichen, geheimen Schlüssel K
  - AES fällt daher in die Kategorie der „**Symmetrischen**" oder „**Private-Key**" Verschlüsselungsverfahren



- Meist existiert aber kein geheimer, ausgetauschter Schlüssel
  - Ad-hoc Kommunikation mit unbekannten Parteien im Internet
  - Jedes Paar Parteien benötigt eigenen Schlüssel ($\frac{n(n-1)}{2}$ bei n Parteien)

- Lösung: „**Asymmetrische**" oder „**Public-Key**" Verschlüsselungsverfahren

Asymmetrische Verschlüsselung
Grundprinzip

Hochschule Fulda
University of Applied Sciences

1. Empfänger*in generiert ein **Schlüsselpaar** $(\boldsymbol{K_E}, \boldsymbol{K_D})$
   - $\boldsymbol{K_E}$: **öffentlicher Schlüssel**, der von allen Parteien zum Verschlüsseln genutzt werden kann
   - $\boldsymbol{K_D}$: **geheimer Schlüssel**, mit dem Ciphertexte entschlüsselt werden können
   - K_E und K_D stehen in einer Relation ($K_E = f(K_D)$ und $K_D = g(K_E)$)

2. Sender*in nutzt K_E um Plaintext P zu verschlüsseln als $C = Enc_{K_E}(P)$

3. Nur Empfänger*in kann C entschlüsseln als $P = Dec_{K_D}(C)$

- Asymmetrische Verfahren basieren auf mathematisch schweren Probleme um sicherzustellen, dass nicht von K_E auf K_D geschlossen werden

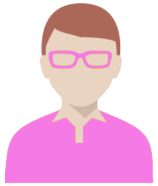
Asymmetrische Verschlüsselung
Rivest-Shamir-Adleman (RSA)

- RSA wurde 1977 entwickelt von R. **R**ivest, A. **S**hamir und L. **A**dleman

- RSA kann zur asymmetrischen Ver/Entschlüsselung genutzt werden

- Die Sicherheit von RSA basiert auf:
  - Dem RSA Problem (e-te Wurzel modulo N)
  - Der Schwierigkeit der Primfaktorzerlegung

RSA Protokoll
Alice möchte Bob Nachricht P senden



Schlüsselgenerierung

- Wähle zufällige Primzahlen p und q
- Berechne $N = p \cdot q$
- Wähle e zufällig mit $\mathrm{ggT}(\varphi(N), e) = 1$
- Berechne d als: $e \cdot d \bmod \varphi(N) = 1$
- Setze $K_E = (N, e)$ und $K_D = d$

$$\xleftarrow{K_E = (N, e)}$$

Verschlüsselung

- Berechne $C = P^e \bmod N$

$$\xrightarrow{C}$$

Entschlüsselung

- Berechne $P = C^d \bmod N$

Beispiel mit Zahlen → Übung

Asymmetrische Verschlüsselung Wieso, Wie, Weshalb?

RSA

1. Wieso funktioniert RSA?

2. Wie berechnet man RSA?

Funktionen:
- Phi-Funktion φ
- Größter Gemeinsamer Teiler (ggT)
- Modulare Invertierung

3. Weshalb ist RSA sicher?

Probleme:
- RSA Problem
- Primfaktorzerlegung

Mathematisches Fundament:
- Restklassen und Modulo
- Algebraische Strukturen: Ring, Gruppe, Halbgruppe

- $\mathbb{N}$: Menge aller positiven ganzen Zahlen ohne 0: $\{1, 2, \dots\}$
- $\mathbb{N}_0$: Menge aller positiven ganzen Zahlen mit 0: $\{0, 1, 2, \dots\}$
- $\mathbb{Z}$: Menge aller ganzen Zahlen: $\{\dots, -2, -1, 0, 1, 2, \dots\}$

- Für $a \in \mathbb{Z}$ und $n \in \mathbb{N}$ gibt es eindeutigen Quotienten $q \in \mathbb{Z}$ und Rest r mit:
  1. $q = a/n$
  2. $r = a \; mod \; n$

- **Beispiele:**
  - $a = 33$ und $n = 4$: Quotient $q = 33 \,/\, 4 = ?$ und Rest $r = 33 \; mod \; 4 = ?$
  - $a = -7$ und $n = 5$: Quotient $q = -7 \,/\, 5 = ?$ und Rest $r = -7 \; mod \; 5 = ?$

- **Rechenregeln für Modulare Arithmetik:**
    1. Addition: $(a \bmod n) + (b \bmod n) \bmod n = a + b \bmod n$
    2. Subtraktion: $(a \bmod n) - (b \bmod n) \bmod n = a - b \bmod n$
    3. Multiplikation: $(a \bmod n) \cdot (b \bmod n) \bmod n \; = a \cdot b \bmod n$
    4. Exponentiation: $(g^a \bmod n)^b \bmod n = (g^a)^b \bmod n = g^{a \cdot b} \bmod n$

- **Beispiele für $\boldsymbol{n = 7, a = 10\ (= 3 \bmod 7)}$ und $\boldsymbol{b = \ 13\ (= 6 \bmod 7)}$:**
    1. Addition: $3 + 6 \bmod 7 = 9 \bmod 7 = 23 \bmod 7 = 10 + 13 \bmod 7$
    2. Subtraktion: $3 - 6 \bmod 7 = -3 \bmod 7 = 10 - 13 \bmod 7$
    3. Multiplikation: $3 \cdot 6 \bmod 7 \; = 18 \bmod 7 = 130 \bmod 7 = 10 \cdot 13 \bmod 7$

Recap Algebra [DHBW]
Definition Restklasse

- Modulo n sind alle Werte $a = i \cdot n + r$ für $i \in \mathbb{Z}$ äquivalent.

- Die Menge $\{i \cdot n + r \mid i \in \mathbb{Z}\}$ wird als **Restklasse** von r bezeichnet, wobei r ein **Repräsentant** der Restklasse ist

- Die Menge aller Restklassen modulo n wird geschrieben als $\mathbb{Z}_n$

- Beispiel: $\mathbb{Z}_3$ (Zahlen aus $\mathbb{Z}$ modulo 3) besteht aus den folgenden Restklassen:
  - Restklasse für $r = 0$: $\{\dots, -9, -6, -3, \mathbf{0}, 3, 6, 9, \dots\}$
  - Restklasse für $r = 1$: $\{\dots, -8, -5, -2, \mathbf{1}, 4, 7, \dots\}$
  - Restklasse für $r = 2$: $\{\dots, -7, -4, -1, \mathbf{2}, 5, 8, \dots\}$

Recap Algebra [DHBW]
Gruppentheorie

Hochschule Fulda
University of Applied Sciences

- Sei G eine Menge und $\circ$ eine Operation bezüglich G mit
  - $\circ: G \times G \mapsto G$ (oder auch geschrieben als $a \circ b = c$ für $a, b, c \in G$).

- Das Tupel $(G,\circ)$ wird als Halbgruppe bezeichnet, wenn:
  1. **Assoziativ**: $\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$

- Das Tupel $(G,\circ)$ wird als abelsche Gruppe bezeichnet, wenn:
  1. **Assoziativ**: $\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$
  2. **Neutrales Element**: $\exists e \in G$ für das gilt: $\forall a \in G$ gilt: $a \circ e = e \circ a = a$
  3. **Inverses Element**: $\forall a \in G$ gilt: $\exists a^{-1} \in G$ mit $a \circ a^{-1} = a^{-1} \circ a = e$
  4. **Kommutativ (abelsch):** $\forall a, b \in G$ gilt: $a \circ b = b \circ a$

Recap Algebra [DHBW]
Gruppenanalyse Beispiele

- Sind die folgenden Tupel Gruppen? (Lösung auf den folgenden Slides):
  1. $(\mathbb{Z}, -)$
  2. $(\mathbb{N}, +)$
  3. $(\mathbb{N}_0, +)$
  4. $(\mathbb{Z}, +)$

Recap Algebra [DHBW]
1. Beispiel $(\mathbb{Z}, -)$

Hochschule Fulda
University of Applied Sciences

- Ist $(\mathbb{Z}, -)$ eine Gruppe?

1. **Assoziativ?** $(\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c))$
   - Nein, $a - (b - c) = a - b + c \neq (a - b) - c$, *für* $a, b, c \in \mathbb{Z}$

- ➔ $(\mathbb{Z}, -)$ ist keine Gruppe!

Recap Algebra [DHBW]
2. Beispiel $(\mathbb{N}, +)$

- Ist $(\mathbb{N}, +)$ eine Gruppe?

1. **Assoziativ?** ($\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$)
   - Ja, da $a + (b + c)$ = $(a + b) + c$ = $a + b + c$, *für* $a, b, c \in \mathbb{N}$
2. **Neutrales Element?** ($\exists e \in G$ für das gilt: $\forall a \in G$ gilt: $a \circ e = e \circ a = a$)
   - Nein, da für alle $a, e \in \mathbb{N}$ gilt: $a + e > a$ (da $0 \notin \mathbb{N}$)

- ➔ $(\mathbb{N}, +)$ ist keine Gruppe!

Recap Algebra [DHBW]
3. Beispiel $(\mathbb{N}_0, +)$

- Ist $(\mathbb{N}_0, +)$ eine Gruppe?

1. **Assoziativ?** ($\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$)
   - Ja, da $a + (b + c)$ = $(a + b) + c$ = $a + b + c$, *für* $a, b, c \in \mathbb{N}_0$
2. **Neutrales Element?** ($\exists e \in G$ für das gilt: $\forall a \in G$ gilt: $a \circ e = e \circ a = a$)
   - Ja, da für alle $a \in \mathbb{N}_0$ gilt: $a + 0 = 0 + a = a$ und $0 \in \mathbb{N}_0$
3. **Inverses Element?** ($\forall a \in G$ gilt: $\exists a^{-1} \in G$ mit $a \circ a^{-1} = a^{-1} \circ a = e$)
   - Nein, da für ein fixes $a \in \mathbb{N}_0$ und $a > 0$ für jedes b $\in \mathbb{N}_0$ gilt: $a + b \geq a > 0$

- ➔ $(\mathbb{N}_0, +)$ ist keine Gruppe!

Recap Algebra [DHBW]
4. Beispiel $(\mathbb{Z}, +)$

- Ist $(\mathbb{Z}, +)$ eine abelsche Gruppe?

1. **Assoziativ?** ($\forall a, b, c \in G$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$)
   - Ja, da $a + (b + c) = (a + b) + c = a + b + c$, *für* $a, b, c \in \mathbb{Z}$
2. **Neutrales Element?** ($\exists e \in G$ für das gilt: $\forall a \in G$ gilt: $a \circ e = e \circ a = a$)
   - Ja, da für alle $a \in \mathbb{Z}$ gilt: $a + 0 = 0 + a = a$ und $0 \in \mathbb{Z}$
3. **Inverses Element?** ($\forall a \in G$ gilt: $\exists a^{-1} \in G$ mit $a \circ a^{-1} = a^{-1} \circ a = e$)
   - Ja, da für jedes $a \in \mathbb{Z}$ gilt: $a + (-a) = 0$ und $-a \in \mathbb{Z}$
4. **Kommutativ (abelsch)?** ($\forall a, b \in G$ gilt: $a \circ b = b \circ a$)
   - Ja, da für alle $a, b \in \mathbb{Z}$ gilt: $a + b = b + a$

- ➔ $(\mathbb{Z}, +)$ ist eine Gruppe

Recap Algebra [DHBW]
Ring und Restklassenring

- Ein Ring $(G, +,\cdot)$ ist eine algebraische Struktur bei der:
  - $(\boldsymbol{G},\cdot)$ eine **Halbgruppe** bildet
  - $(\boldsymbol{G}, +)$ eine **abelsche Gruppe** bildet
  - Die **Distributivgesetze** gelten:
    - Linke Distributivität: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, für $\forall a, b, c \in G$
    - Rechte Distributivität: $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$, für $\forall a, b, c \in G$

- Ein Ring $(\mathbb{Z}_n, +,\cdot)$ über einer **Restklasse** $\mathbb{Z}_{\boldsymbol{n}}$ wird als **Restklassenring** bezeichnet

Recap Algebra
Beispiel Restklassenring $(\mathbb{Z}_6, +, \cdot)$

$(\mathbb{Z}_6, +)$ ist eine Gruppe:

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

$(\mathbb{Z}_6, \cdot)$ ist eine Halbgruppe:

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Zusätzlich zur Halbgruppe:

- $(\mathbb{Z}_6, \cdot)$ hat das neutrale Element 1
- $(\mathbb{Z}_6, \cdot)$ ist für manche Elemente invertierbar

Recap Algebra
Invertierbarkeit der Multiplikation

Hochschule Fulda
University of Applied Sciences

- Für $(\mathbb{Z}_n, \ \cdot)$ sind nicht alle Elemente invertierbar
  - Die Division ist also nicht für alle Elemente möglich

- Aber: Teilerfremde Zahlen $z \in \mathbb{Z}$ zu n sind invertierbar
  - Teilerfremd bedeutet auch: größter gemeinsamer Teiler ist 1 $(\mathrm{ggT}(n, z) = 1)$

Ergebnistabelle $(\mathbb{Z}_6,\cdot)$

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Für $(\mathbb{Z}_6,\cdot)$ sind invertierbar:
- 1 und 5

Nicht invertierbar:
- 2, da $\mathrm{ggT}(6{,}2)$ = 2
- 3, da $\mathrm{ggT}(6{,}3)$ = 3
- 4, da $\mathrm{ggT}(6{,}4)$ = 2

Ergebnistabelle $(\mathbb{Z}_5,\cdot)$

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Für $(\mathbb{Z}_5,\cdot)$ sind invertierbar:
- 1,2,3,4

Grund: 5 ist prim!

Recap Algebra [DHBW]
Anzahl Teilerfremde Zahlen: Phi-Funktion φ

Hochschule Fulda
University of Applied Sciences

- Die Anzahl der teilerfremden Zahlen zu n ist definiert als mittels der **Eulerschen Phi-Funktion** $\boldsymbol{\varphi(n)} = |\{z \in \mathbb{Z}_n : \mathrm{ggT}(n, z) = 1\}|$.

- **Rekursive Berechnung:**
  1. Falls n prim: $\varphi(n) = n - 1$.
     - Beispiel: $\varphi(5) = |\{1,2,3,4\}| = 4$ und $\varphi(3) = |\{1, 2\}| = 2$
  2. Falls $n = p \cdot q$ mit $p \neq q$: $\varphi(n) = \varphi(p) \cdot \varphi(q)$.
     - Beispiel: $\varphi(15) = |\{1, 2, 4, 7, 8, 11, 13, 14\}| = 8 = 4 \cdot 2 = (5 - 1) \cdot (3 - 1) = \varphi(5) \cdot \varphi(3)$

- Beispiel $\varphi(77) = \varphi(7) \cdot \varphi(11) = (7 - 1) \cdot (11 - 1) = 6 \cdot 10 = 60$

Recap Algebra [DHBW]
Allgemeine Potenz-Rechenregel

- Für $a, b, n \in \mathbb{N}$ mit $\mathrm{ggT}(a, n) = 1$ gilt:

$$a^b \ mod \ n = a^{b \ mod \ \varphi(n)} \ mod \ n$$

- **Beispiel** für $a = 3, b = 2, \ n = 5$ und $\varphi(5) = 4$:
  1. $3^2 \ mod \ 5 = 9 \ mod \ 5 = 4$
  2. $3^{2+4} \ mod \ 5 = 3^6 \ mod \ 5 = 729 \ mod \ 5 = 4$
  3. $3^{2+8} \ mod \ 5 = 3^{10} \ mod \ 5 = 59049 \ mod \ 5 = 4$

- Was ist: $3^{422} \ mod \ 5 = 3^{2+420 \ mod \ 4} \ mod \ 5 = 3^2 \ mod \ 5 = 4$

Wieso Funktioniert RSA? Zusammenfassung

- Wichtige Eigenschaften Algebra:

1. **Rechenregel Modulare Exponentiation:**
   - $(g^a \, mod \, n)^b \, mod \, n = g^{a \cdot b} \, mod \, n$
2. **Rekursive Berechnung der Phi-Funktion φ:**
   a. Falls n prim: $\varphi(n) = n - 1.$
   b. Falls $n = p \cdot q$ und $p \neq q$: $\varphi(n) = \varphi(p) \cdot \varphi(q)$
3. **Allgemeine Potenz-Rechenregel:**
   - $a^b \, mod \, n = a^{b \, mod \, \varphi(n)} \, mod \, n$

- Die Regeln werden auf den nächsten Folie benötigt!

Asymmetrische Verschlüsselung
Wieso, Wie, Weshalb?

RSA

1. Wieso funktioniert RSA?

2. Wie berechnet man RSA?

3. Weshalb ist RSA sicher?

Funktionen:
- Phi-Funktion φ
- Größter Gemeinsamer Teiler (ggT)
- Modulare Invertierung

Probleme:
- RSA Problem
- Primfaktorzerlegung

Mathematisches Fundament:
- Restklassen und Modulo
- Algebraische Strukturen: Ring, Gruppe, Halbgruppe

Wieso Funktioniert RSA?

Eigenschaft	RSA
K_E	1. $N = p \cdot q$ mit Primzahlen p und q 2. e mit $\mathrm{ggT}(\varphi(N), e) = 1$
K_D	d mit $e \cdot d \ mod \ \varphi(N) = 1$
$Enc_{K_E}(P)$	$C = P^e \ mod \ N$
$Dec_{K_D}(C)$	$P = C^d \ mod \ N$

Wie → Größter Gemeinsamer Teiler

Wie → Modulare Invertierung

- RSA Entschlüsselung funktioniert, da $P = Dec_{K_D}(C) = C^d \ mod \ N$

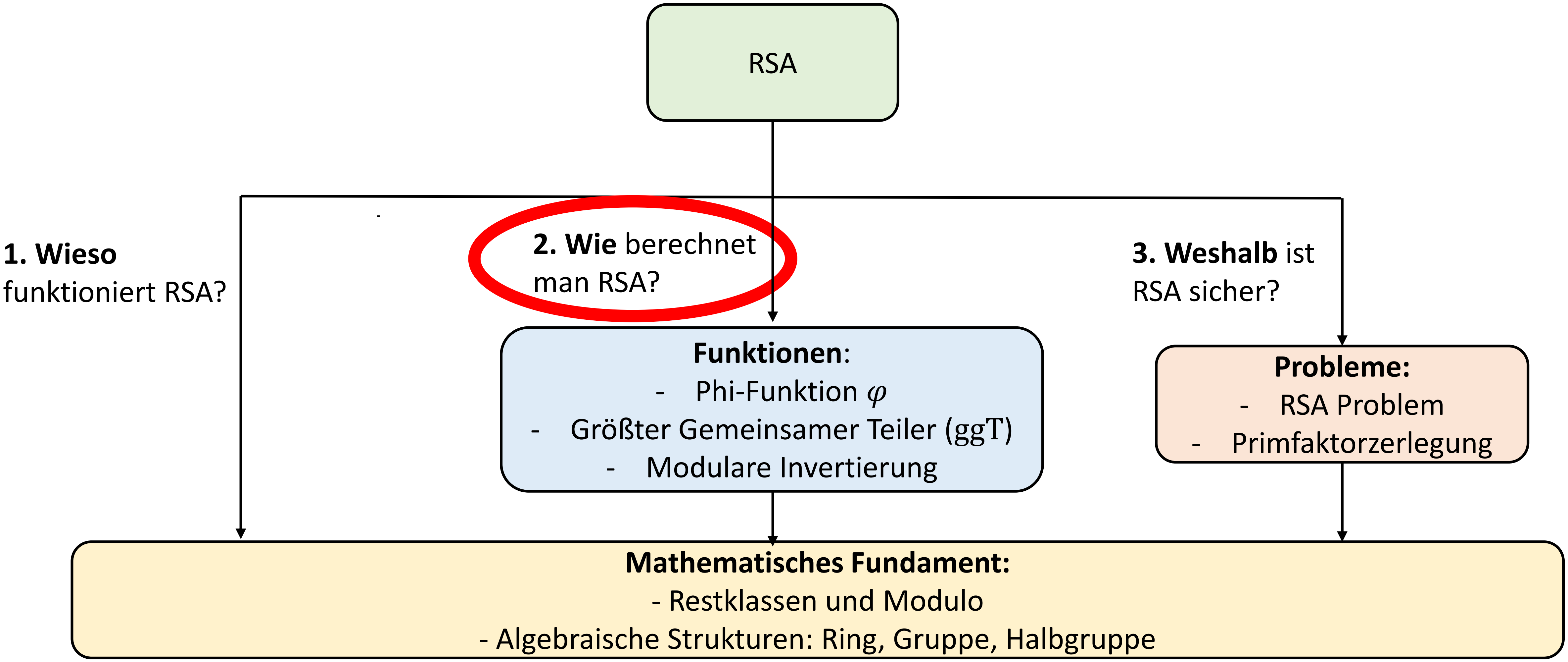
$= (P^e \ mod \ N)^d \ mod \ N = P^{e \cdot d} \ mod \ N$ (Rechenregel Modulare Exp.)

$= P^{e \cdot d \ mod \ \varphi(N)} \ mod \ N$ (Allgemeine Potenz-Rechenregel)

$= P^1 \ mod \ N = P$ (Wahl d als $e \cdot d \ mod \ \varphi(N) = 1$)

Asymmetrische Verschlüsselung
Wieso, Wie, Weshalb?

Hochschule Fulda
University of Applied Sciences



Wie Berechnet Man RSA?
Größter Gemeinsamer Teiler (ggT)

- **Definition:** Für $a, b \in \mathbb{Z}$ ist $c = \mathrm{ggT}(a, b)$ die größte Zahl die sowohl a als auch b ohne Rest teilt. Beispiele:
  - $\mathrm{ggT}(18, 12) = 6$
  - $\mathrm{ggT}(43, 7) = 1$

- Der ggT kann mittels des euklidschen Algorithmus berechnet werden → Übung

```
int ggT(a,b) {
  while(b != 0) {
    c = a % b;
    a = b;
    b = c;
  }
  return a; }
```

Beispiel: $\mathrm{ggT}(270, 192)$

$270 = 1 \cdot 192 + 78$	Iteration 1 ($a = 270, b = 192$)
$192 = 2 \cdot 78 + 36$	Iteration 2 ($a = 192, b = 78$)
$78 = 2 \cdot 36 + 6$	Iteration 3 ($a = 78, b = 36$)
$36 = 6 \cdot 6 + 0$	Iteration 4 ($a = 36, b = 6$)
Ende, da $b = 0$	Iteration 5 ($a = 6, b = 0$)

→ Somit ist $\mathrm{ggT}(270, 192) = 6$.

Wie Berechnet Man RSA? Modulare Invertierung (1/2)

- Für RSA muss d als das **modular Inverse Element** von e zu $\varphi(n)$ gewählt werden, so dass gilt: $e \cdot d \ mod \ \varphi(N) = 1$

- **Modular Inverses:** Für $a, n \in \mathbb{Z}$ finde $a^{-1} \in \mathbb{Z}$ mit $a \cdot a^{-1} \ mod \ n = 1$.

- **Alternative Form:** Für $a, n \in \mathbb{Z}$ finde $a^{-1}, b \in \mathbb{Z}$ mit $n \cdot b + a \cdot a^{-1} = 1$.
  - Beispiel für $a = 5,\ n = 7$: $7 \cdot (-2) + 5 \cdot 3 = 1$ ➔ $5 \cdot 3 \ mod \ 7 = 1$

- Die Modulare Invertierung kann mittels des erweiterten euklidschen Algorithmus berechnet werden → Übung

Wie Berechnet Man RSA? Modulare Invertierung (2/2)

Hochschule Fulda
University of Applied Sciences

Pseudocode erweiterter euklidscher Algo.

```
def Mod_Inv(n, a):
    b,aI = 1,0 # #Ergebnisse
    u,v  = 0,1 # Temporäre Werte
    while a != 0:
        z1,z2,z3 = a,u,v #Zwischenspeicher
        q = n/a
        u = b-q*u
        v = aI -q*v
        b,aI = z2,z3 #Zwischenspeicher
        #Haltepunkt Beispielrechnung
        a = n%a
        n = z1
    return b,aI #n*b+a*aI=ggT(n,a)
```

Beispiel: $\mathrm{Mod\_Inv}(270, 192)$

Iteration	n	a	q	u	b	v	a^{-1}
Init	270	192	-	0	1	1	0
Halt 1	270	192	1	1	0	-1	1
Halt 2	192	78	2	-2	1	3	-1
Halt 3	78	36	2	5	-2	-7	3
Halt 4	36	6	6	-32	5	45	-7
Schluss	6	0	-	-	-	-	-

$n \cdot b + a \cdot a^{-1} = 270 \cdot 5 + 192 \cdot (-7) = 6,$
da $ggT(270{,}192) = 6$

Asymmetrische Verschlüsselung
Wieso, Wie, Weshalb?

RSA

1. Wieso funktioniert RSA?

2. Wie berechnet man RSA?

3. Weshalb ist RSA sicher?

Funktionen:
- Phi-Funktion φ
- Größter Gemeinsamer Teiler (ggT)
- Modulare Invertierung

Probleme:
- RSA Problem
- Primfaktorzerlegung

Mathematisches Fundament:
- Restklassen und Modulo
- Algebraische Strukturen: Ring, Gruppe, Halbgruppe

Weshalb ist RSA sicher?

- **Öffentlich ist**: $K_E = (N, e)$
- **Geheim sind:** $K_D = d$ sowie p und q.

- **Sicherheit:** wie kann aus C mit $C = P^e \; mod \; N$ der Wert P berechnet werden?
  1. Bilden der e-ten Wurzel aus C modulo N: $P = \sqrt[e]{C} \; mod \; N$ (ϟ RSA Problem)
  2. Berechnen des geheimen Schlüssels $e \cdot d \; mod \; \varphi(N) = 1$ (ϟ Primfaktorzerlegung)

Weshalb ist RSA sicher?
Bilden der e-ten Wurzel mod N

Operation	Invertierung für a	Methode
$c = \ a + b \ mod \ n$	$a = c - b \ mod \ n$	Einfache Subtraktion Modulo n
$c = a \cdot b \ mod \ n$	1. $a = c/b \ mod \ n$ 2. $a = c \cdot b^{-1} \ mod \ n$ mit $b \cdot b^{-1} \ mod \ n = 1$	1. Division schwierig. Stattdessen: 2. Multiplikation mit modular Inversem Element
$c = a^b \ mod \ n$	$a = \sqrt[b]{c} \ mod \ n$	Schwierig

- Beispiel ($b = 3, n = 7$). Gesucht ist a mit den folgenden Bedingungen:
  - $3/4 \bmod 7 = a = ?$
  - $\sqrt[3]{6} \ mod \ 7 = a = ?$

Weshalb ist RSA sicher?
Berechnen des Geheimen Schlüssels K_D

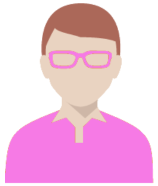
Hochschule Fulda
University of Applied Sciences

- **Öffentlich ist**: $K_E = (N, e)$
- **Geheim sind:** $K_D = d$ sowie p und q.

- $K_D = d$ als modular Inverses: $e \cdot d \ mod \ \varphi(N) = 1$. Strategie
  1. Berechne $\varphi(N)$ mittels N
  2. Berechne d via erweitertem euklidschen Algorithmus auf e und $\varphi(N)$

- Berechnung von $\varphi(N)$:
  - Wenn Primfaktorzerlegung $N = p \cdot q$ bekannt: $\varphi(N) = (p-1) \cdot (q-1)$
  - Ansonsten: Schwierig!

RSA Protokoll (Recap)
Alice möchte Bob Nachricht P senden



Schlüsselgenerierung

- Wähle zufällige Primzahlen p und q
- Berechne $N = p \cdot q$
- Wähle e zufällig mit $\mathrm{ggT}(\varphi(N), e) = 1$
- Berechne d als: $e \cdot d \bmod \varphi(N) = 1$
- Setze $K_E = (N, e)$ und $K_D = d$

$$\xleftarrow{\quad K_E = (N, e) \quad}$$

Verschlüsselung

- Berechne $C = P^e \bmod N$

$$\xrightarrow{\quad C \quad}$$

Entschlüsselung

- Berechne $P = C^d \bmod N$

RSA – Sicherheit in der Anwendung

- RSA ist als asymmetrisches Verfahren bereits im Chosen-Plaintext Modell
  - Angreifer*in kann Plaintexte mit öffentlichem Schlüssel $\mathrm{K_E} = (N, e)$ verschlüsseln
  - Chosen-Ciphertext Angriffe gegen Textbuch RSA sind möglich

- Kurze Plaintexte können via Brute-Force gebrochen werden:
  - Telefonnr. (~32 bit): Verschlüsseln aller Nummern mit K_E und Vergleich mit Ciphertext

- ➔ Textbuch RSA benötigt weitere Paddingverfahren um Chosen-Ciphertext und Brute-Force Angriffe auszuschließen
  - **RSA-OAEP:** Nachricht wird um Zufallszahl und Prüfsumme erweitert

Wahl der Primzahlgröße für RSA

Hochschule Fulda
University of Applied Sciences

- Die Sicherheit von RSA basiert auf Komplexität der Primfaktorzerlegung
- Die RSA Factoring Challenge wurde ausgerufen um einen Überblick über die Komplexität der Primfaktorzerlegung zu erhalten
- Moderne Empfehlungen für Größen [KeyLen - ECRYPT]
  - 1024: Nicht mehr nutzen
  - 3072: Sicher 2021 - 2028
  - 15360: Sicher 2021 - 2068

Bit	Preisgeld	Datum	Kommentar
330	1.000 $	April 1991	
430	14.527 $	April 1996	
512	9.383 $	August 1999	
576	10.000 $	Dezember 2003	
640	20.000 $	November 2005	
768	-	Dezember 2009	Ab 2007 kein Preisgeld mehr
795	-	Dezember 2019	4000 CPU Kern Jahre Aufwand
829	-	Februar 2020	2700 CPU Kern Jahre Aufwand

Asymmetrische Verschlüsselung Performance

- RSA Ver/Entschlüsselung mit Modulus $|N| = n$ hat Komplexität $O(n^3)$
  - Multiplikation zweier n-bit Werte hat $O(n^2)$
  - Exponentiation mit n-bit Exponent hat $O(n^3)$
- Exponent e für Verschlüsselung wird kurz gewählt
  - e = {3, 65537}
- Schlüsselgenerierung ist sehr rechenintensiv
  - Finden und verifizieren von Primzahlen

Verfahren	Aufrufe pro Sek.
64-bit Mult.	1,401,372,784
AES-128 (SW)	22,413,312
AES-128 (HW)	175,308,800
RSA-2048 (SW)	Enc: 33,483 Dec: 822 KeyGen: 2

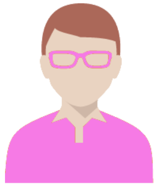
Single Thread in Ubuntu VM mit Crypto++ und konstantem Schlüssel

Hybride Verschlüsselung (1/2)

Aspekt	Symmetrische Verschlüsselung	Asymmetrische Verschlüsselung
Vorteile	Sehr schnell (~Gigabyte/Sekunde)	Es muss kein geheimer Schlüssel ausgetauscht sein
Nachteile	Geheimer Schlüssel muss ausgetauscht sein	Langsam (~Hunderte Kilobyte/Sekunde)

- Hybride Verschlüsselung kombiniert die Vorteile beider Verfahren:
  1. **Asymmetrische Verfahren** um einen **symmetrischen Schlüssel** zu übertragen
  2. **Symmetrische Verfahren** um die Daten zu übertragen

Hybride Verschlüsselung (2/2)



Schlüsselpaar (K_E, K_D)

K_E ←

Wähle zufälligen sym. Schlüssel K

$C_K = Enc_{K_E}(K)$ →

Entschlüssele sym. Schlüssel $K = Dec_{K_D}(C_K)$

Asymmetrische Verschlüsselung

Symmetrische Verschlüsselung

$C = Enc_K(P)$ ←

Berechne $P = Dec_K(C)$

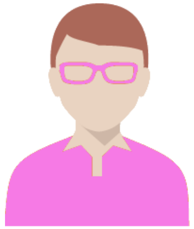
$Enc_K(\ldots)$ ↔

Diffie-Hellman Verfahren (DH)

- Asymmetrisches Verfahren zur Schlüsselvereinbarung, entwickelt in 1976
- Basiert auf diskreten Logarithmusproblem in primen Restklassenringen
- Voraussetzung: Alice und Bob kennen öffentliche Primzahl p und Basis g
  - Mögliche Primzahlen und Basen sind in Standards definiert [DHP]
- DH kann nicht für Verschlüsselung, nur für Schlüsselvereinbarung
  - DH Benötigt weiteres Verschlüsselungsverfahren (z.B. symmetrisches Verfahren)

Diffie-Hellman Protokoll

Bekannt: Öffentliche Primzahl p und Basis g



1. Wähle zufälligen Wert a
2. Berechne $A = g^a \; mod \; p$
3. Berechne $K = B^a \; mod \; p = g^{a \cdot b} \; mod \; p$

$$\xrightarrow{\quad A \quad}$$

$$\xleftarrow{\quad B \quad}$$

1. Wähle zufälligen Wert b
2. Berechne $B = g^b \; mod \; p$
3. Berechne $K = A^b \; mod \; p = g^{a \cdot b} \; mod \; p$

(K kann in Verschlüsselungsverfahren genutzt werden)

Wieso ist DH sicher?

- Eve möchte den Schlüssel K berechnen
  - Eve kennt: $g, p, A = g^a \; mod \; p, B = g^b mod \; p$.
  - Eve kennt nicht: a, b und $K = g^{a \cdot b}$.

- Um a (oder b) zu finden muss Eve den diskreten Logarithmus berechnen:
  - $a = \log_g A \;\; mod \; p$ oder
  - $b = \log_g B \;\; mod \; p$

- Aber: Bester bekannter Algorithmus zur Berechnung des diskreten Logarithmus hat Komplexität $\mathrm{O}(\sqrt{p})$.

DH Schlüssel Behalten oder Löschen?

- Originales DH Protokoll: a und b werden für jeden Austausch neu generiert
  - **Vorteil:** Falls a oder b einer Sitzung veröffentlich werden, ist nur die aktuelle Sitzung korrumpiert (sog. **Forward Secrecy**). ➔ Standard in vielen Protokollen
  - **Nachteil:** Mallory kann Schlüsselaustausch abfangen, da Alice und Bob sich nicht anhand von A und B authentifizieren können (sog. **Attacker-in-the-Middle Angriff**)

$A = g^a \bmod p$

$B = g^b \bmod p$

$F = g^f \bmod p$

$F = g^f \bmod p$

$K = g^{af} \bmod p$

$K' = g^{bf} \bmod p$

- Eine Alternative zu primen Restklassenringen sind **elliptische Kurven (ECC)**
  - **Elliptische Kurve:** Menge an Punkten die eine Gleichung erfüllen, z.B.: $y^2 = x^3 + ax + b$

- Seien A, P zwei Punkte auf einer Kurve mit $a \cdot P = A$
  - **Einfach:** Aus P und a den Punkt A zu berechnen ($A = a \cdot P$)
  - **Schwer:** Aus P und A den Wert a zu berechnen ($a = P/A$)

Punktaddition Kurve $y^2 = x^3 + ax + b$

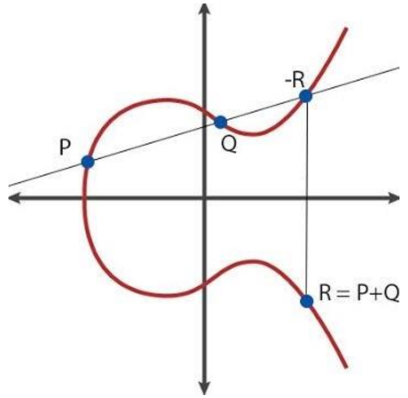


Bild Quelle: https://paulmillr.com/posts/noble-secp256k1-fast-ecc/

Punktmultiplikation Kurve $y^2 = x^3 + ax + b$

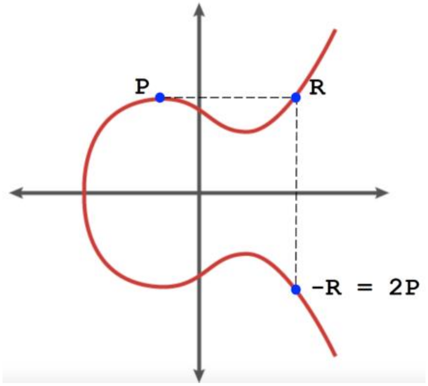


Bild Quelle: https://blog.intothesymmetry.com/2019/07/on-isogenies-verifiable-delay-functions.html

Asymmetrische Verschlüsselung
Elliptische Kurven (2/2)

Hochschule Fulda
University of Applied Sciences

- Elliptische Kurven können in Verfahren genutzt werden, die auf dem diskreten Logarithmusproblem basieren:
  - Elliptische Kurven Diffie-Hellmann (ECDH)
- Vorteil von elliptischen Kurven ist, dass die Kurven kleinere Bit-werte besitzen

Bitlänge sym. Schlüssel	Bitlänge Primzahl	Bitlänge ECC	Ratio Bitlänge Primzahl / ECC
80	1024	160	6.4
128	3072	256	12.0
256	15360	512	30

Vergleich Schlüsselgrößen [KeyLen – ECRYPT]

Verfahren	Aufrufe pro Sek.
64-bit Mult.	1,401,372,784
AES-128 (SW)	22,413,312
AES-128 (HW)	175,308,800
RSA-2048 (SW)	Enc: 33,483 Dec: 822 KeyGen: 2
DH (SW)	**Prime-2048: 1,302** **ECC-256: 1,773**

Single Thread in Ubuntu VM mit Crypto++ und konstantem Schlüssel

Implementierung Asymmetrische Verschlüsselung

- Asymmetrische Verfahren nur sehr schwer sicher zu implementieren [Trail, B99]:
  - Primzahlen in RSA dürfen weltweit nicht doppelt vorkommen [ND+12]
  - Bestimmte Primzahlen müssen vermieden werden [C96]
  - Bestimmte Werte für d und e müssen vermieden werden [W90]
  - Fehler im Paddingverfahren können zur Kompromittierung des Schlüssels führen [B98]

- Etliche Tricks können asymmetrische Verfahren beschleunigen
  - Chinesischer Restsatz
  - Wahl einer Basis aus einer Restklassengruppe mit kleinerer Ordnung

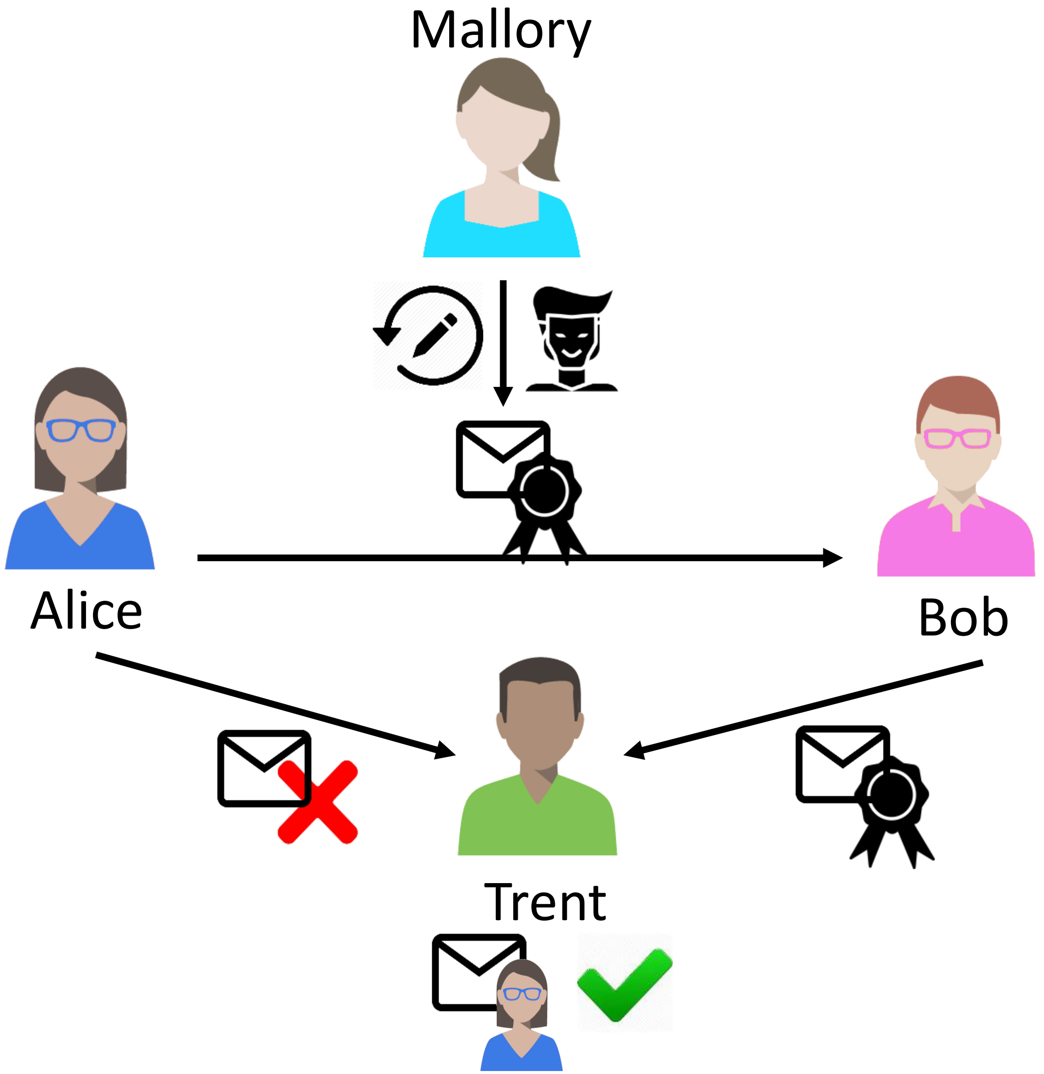
- ➔ Implementieren Sie asymmetrische Verfahren nicht selbst, sondern nutzen Sie bestehende Bibliotheken!

Integrität, Authentizität und Verbindlichkeit durch Digitale Signaturen

Hochschule Fulda
University of Applied Sciences

- **Bedrohungen:**
  - **Integrität:** Mallory ändert die Nachricht
  - **Authentizität:** Mallory fälscht eine Nachricht
  - **Verbindlichkeit:** Alice bestreitet eine Nachricht an Bob gesendet zu haben

- **Ziele:**
  - **Integrität:** Bob kann prüfen ob die Nachricht verändert wurde
  - **Authentizität:** Bob kann prüfen ob die Nachricht von Alice stammt
  - **Verbindlichkeit:** Bob kann gegenüber einer vertrauenswürdigen, dritten Instanz nachweisen, dass Alice eine Nachricht von Alice stammt

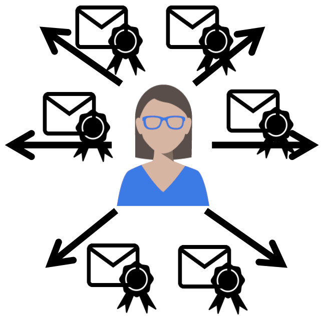


Digitale Signaturen

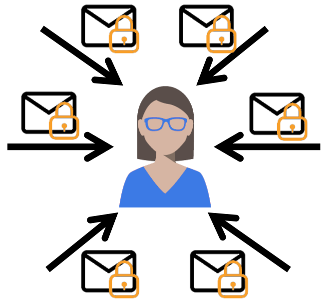
Hochschule Fulda
University of Applied Sciences

- Digitales Pendant zur handgeschriebenen Unterschrift
  - Zu einer öffentlichen Nachricht M soll es eine digitale Signatur S geben

- Anforderungen an ein digitales Signaturverfahren:
  1. Nur Alice darf eine gültige Signatur S zur Nachricht M erzeugen
  2. Jede*r muss die Signatur von S zu M verifizieren können



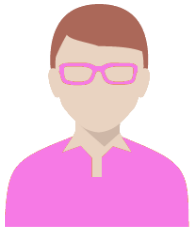
- Ähnlichkeit zur asymmetrischen Verschlüsselung:
  - Jede*r darf eine Nachricht an Alice verschlüsseln
  - Nur Alice darf den Ciphertext entschlüsseln



Recap: Verschlüsseln via RSA
Bob Verschlüsselt Nachricht an Alice

Hochschule Fulda
University of Applied Sciences

Alice sendet $K_E = (N, e)$ vorher an Bob über anderen Kanal



$K_E = (N, e)$ und $K_D = d$

C

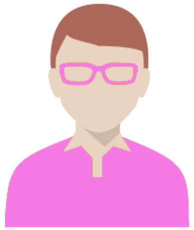
Verschlüssele $C = P^e \; mod \; N$

Entschlüssele $P = C^d \; mod \; N$

- Entschlüsselung funktioniert, da $C^d \; mod \; N = P^{ed} \; mod \; N = P$
  - Jede Partei darf $K_E = (N, e)$ kennen und Nachrichten verschlüsseln
  - Nur Alice kennt $K_D = d$ und kann somit Nachrichten entschlüsseln

Signieren via RSA
Alice Signiert Nachricht an Bob

Alice sendet $K_E = (N, e)$ vorher an Bob über anderen Kanal



$K_E = (N, e)$ und $K_D = d$

Signiere $\boldsymbol{S = M^d \bmod N}$

$\boldsymbol{M, S}$

Berechne $\boldsymbol{M' = S^e \bmod N}$

Prüfe ob $\boldsymbol{M' = M}$

- **Signaturprüfung funktioniert, da $\boldsymbol{S^e \bmod N = M^{ed} \bmod N = M}$**
  - Jede Partei darf $K_E = (N, e)$ kennen und **Signaturen prüfen**
  - Nur Alice kennt $K_D = d$ und kann somit **Nachrichten signieren**

Digitale Signaturverfahren in der Praxis

- Weitere Verfahren zur Signaturberechnung existieren:
  - **Digital Signature Algorithm (DSA)**
  - Elliptic Curve DSA (ECDSA)
  - Elgamal Signatur
  - Merkle Signatur

- Für Verbindlichkeit müssen weitere Informationen an den öffentlichen Schlüssel gebunden werden (➔ Zertifikate und PKI im Kapitel Protokolle)

Digital Signature Algorithm (DSA)
Entstehung und Verwendung

- Digital Signature Algorithm (DSA) wurde 1994 standardisiert [DSA].
  - Von der Benutzung von DSA wird mittlerweile abgeraten!

- Die Sicherheit von DSA beruht auf dem diskreten Logarithmen Problem

DSA Algorithmus	Input	Output	Durchgeführt von
Parametergenerierung	-	Parameter (p, q, g)	Vertrauenswürdige Partei
Schlüsselgenerierung	(p, q, g)	Schlüssel $K_E = y, K_D = x$	Alice (Sender*in)
Signieren	$(p, q, g), x, M$	Signatur (r, s) zu M	Alice (Sender*in)
Verifizieren	$(p, q, g), y, M, (r, s)$	Wurde (r, s) für M von Besitzer*in von K_E erzeugt?	Bob (Empfänger*in)

Digital Signature Algorithm (DSA)
Parameter- und Schlüsselgenerierung

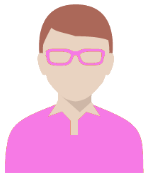
Hochschule Fulda
University of Applied Sciences

Parametergenerierung:

- Wähle eine Primzahl p zufällig
- Wähle eine Primzahl q die $p-1$ teilt
- Berechne $g = h^{(p-1)/q} \ mod\ p$ für zufälliges h

$\xleftarrow{(p, q, g)}$

$\xrightarrow{(p, q, g)}$

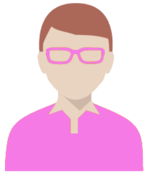


Schlüsselgenerierung:

- Wähle x mit $1 \leq x \leq q$ zufällig
- Berechne $y = g^x \ mod\ p$
- Setze $K_E = y$ und $K_D = x$

$\xrightarrow{K_E = y}$

Digital Signature Algorithm (DSA)
Signieren und Verifizieren



Signieren einer Nachricht $\boldsymbol{M}$:

- Wähle k zufällig mit $1 < k \leq q$
- Berechne $\underline{r = \left(g^k \; mod \; p\right) \; mod \; q \neq 0}$
- Berechne $\underline{s = k^{-1} \cdot (M + r \cdot x) \; mod \; q \neq 0}$

$$\xrightarrow{(r,s), M}$$

Verifizieren der Signatur $(\boldsymbol{r}, \boldsymbol{s})$ zur Nachricht $\boldsymbol{M}$:

- Berechne $w = s^{-1} \; mod \; q$
- Berechne $u_1 = M \cdot w \; mod \; q$
- Berechne $u_2 = r \cdot w \; mod \; q$
- Berechne $\underline{v = (g^{u_1} \cdot y^{u_2} \; mod \; p) \; mod \; q}$
- Signatur ist valide falls $\underline{v == r}$

Wieso funktioniert DSA? ➔ Siehe Übung

Angriffe auf Signaturverfahren

Hochschule Fulda
University of Applied Sciences

- Die Sicherheit von DSA hängt stark vom Zufallswert k ab:
  - Falls k bekannt wird, kann $K_D = x$ berechnet werden
  - Falls k wiederverwendet wird, kann x berechnet werden [PS3] → Siehe Übung
  - Falls k aus einem schlechten Zufallszahlengenerator stammt, kann k geraten werden

- RSA Verschlüsselung und Signaturen niemals mit dem gleichen Schlüssel
  - Verschlüsselte Nachricht könnte entschlüsselt werden via Anfrage zur Signatur
  - Unbeabsichtigte Signatur könnte erzeugt werden via Anfrage zur Entschlüsselung

- Textbuch RSA benötigt wieder weitere Mechanismen um als sicheres Signaturverfahren verwendet zu werden
  - Paddingverfahren: RSA-PSS

Digitale Signaturen Performance

- RSA Signieren analog zu RSA Verschlüsseln:
  - Verifizieren = Verschlüsseln
  - Signieren = Entschlüsseln

- Signieren in DSA ist effizienter als Verifizieren, da weniger Exponentiationen benötigt werden

Verfahren	Aufrufe pro Sek.
64-bit Mult.	1,401,372,784
AES-128 (SW)	22,413,312
AES-128 (HW)	175,308,800
RSA-2048 (SW)	Enc/**Verify**: **33,483** Dec/**Sign**: **822** **KeyGen: 2**
DH (SW)	Prime-2048: 1,302 ECC-256: 1,773
ECDSA-256 (SW)	**Verify: 586** **Sign: 1,736**

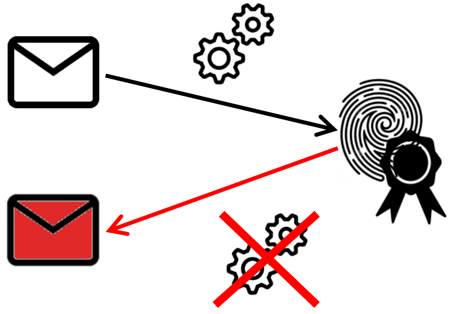
Single Thread in Ubuntu VM mit Crypto++ und konstantem Schlüssel

Asymmetrische Signaturverfahren

- Direktes Signieren und Verifizieren von großen Nachrichten ist sehr ineffizient
  - Signieren: $s = \boldsymbol{k}^{-1} \cdot (\boldsymbol{M} + \boldsymbol{r} \cdot \boldsymbol{x})\ mod\ q$
  - Verifizieren: $u_1 = \boldsymbol{M} \cdot \boldsymbol{w}\ mod\ q$

- Analog zur hybrider Verschlüsselung: Große Nachricht mit **Hilfsfunktion** in einen kleinen, eindeutigen **Fingerabdruck** umwandeln, der dann signiert wird

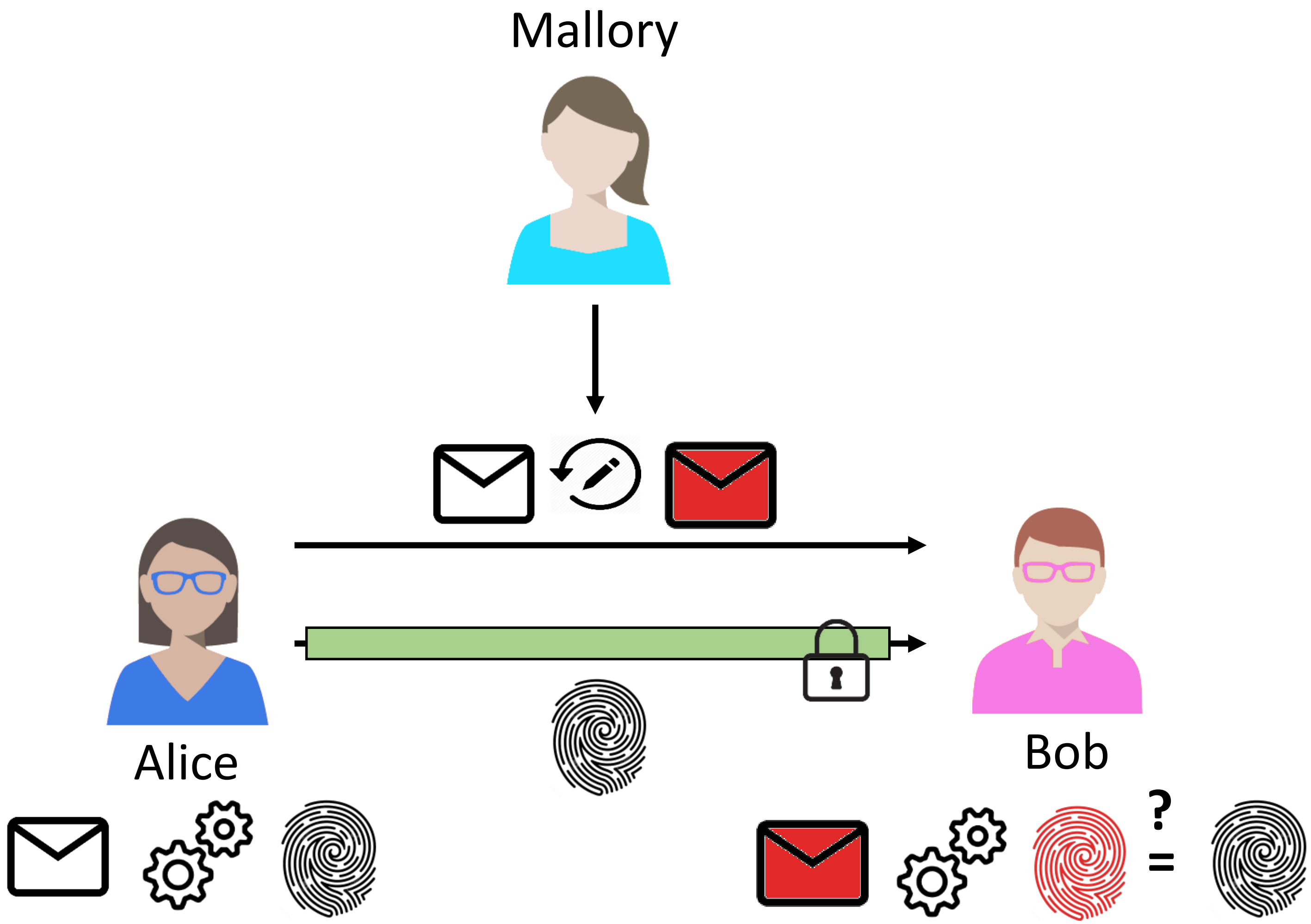
- Anforderung an Hilfsfunktion und Fingerabdruck:
  - Jede Person sollte die Hilfsfunktion berechnen können
  - Es sollte nicht möglich vom Fingerabdruck auf eine Nachricht zurückzurechnen



Integrität durch Hashfunktionen

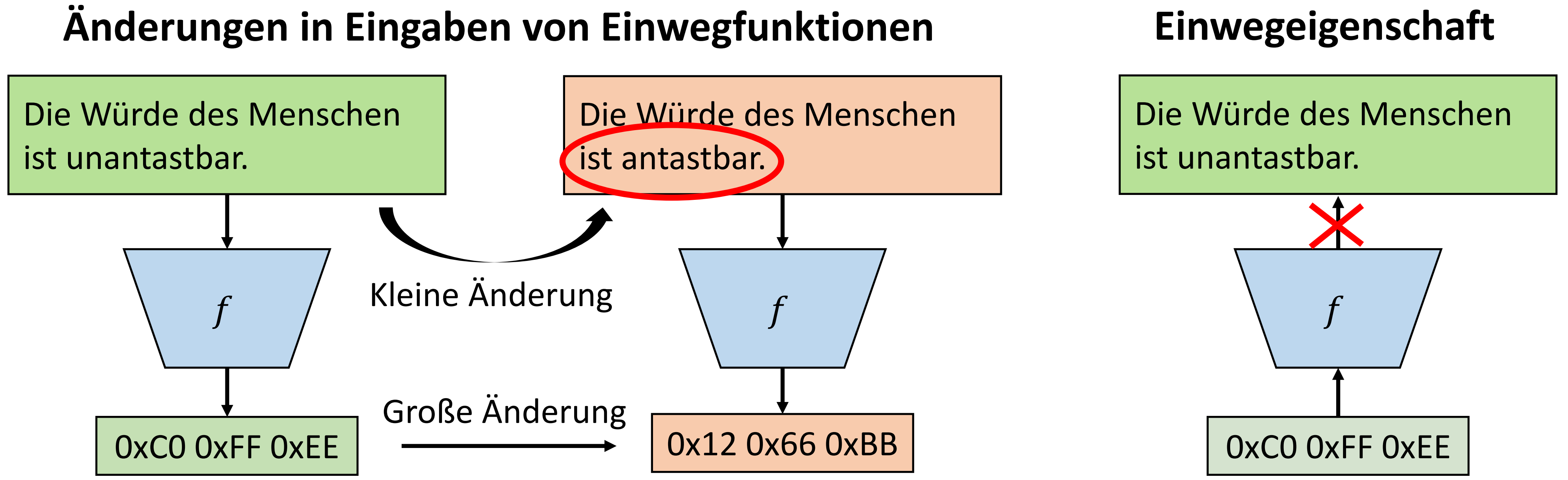
- **Bedrohung:** Mallory verändert die Nachricht

- **Ziel:** Eindeutiger Fingerabdruck mit dem unerlaubte Änderungen an der Nachricht erkannt werden können



Integrität durch Einwegfunktionen

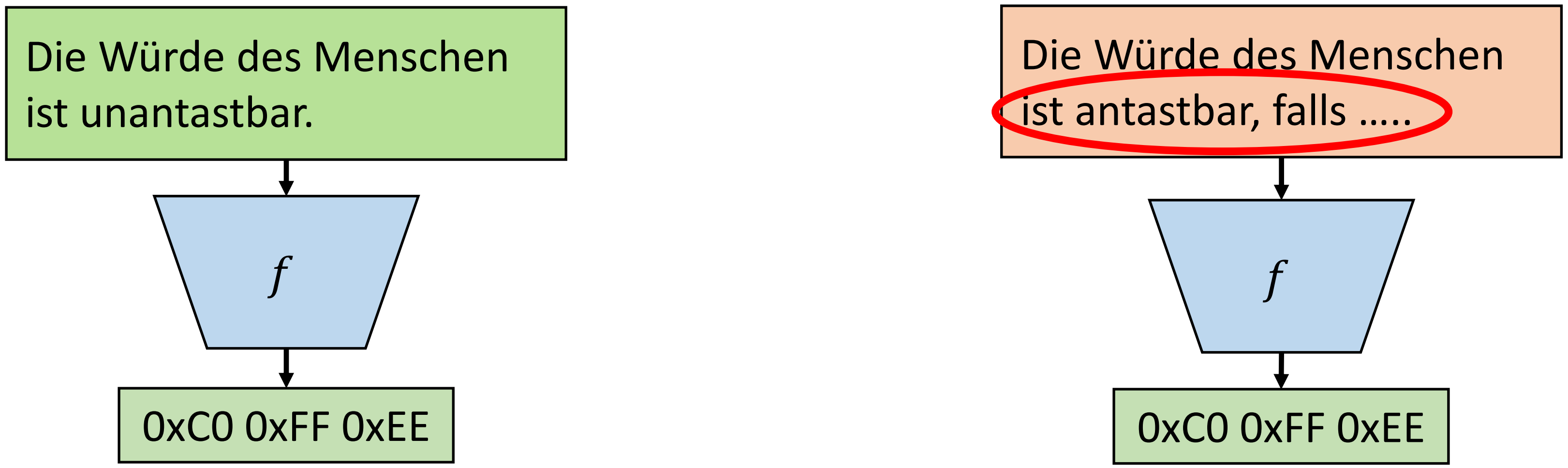
- Eine Einwegfunktion f bildet einen **beliebig langen** Wert auf einen **nicht-invertierbaren** Wert fixer Länge ab



Kollisionsresistenz von Einwegfunktionen

Hochschule Fulda
University of Applied Sciences

- Kollisionen nicht vermeidbar, da die Länge der Nachricht reduziert wird
  - Möglicher Angriff: Ausprobieren von Nachrichten bis Kollision gefunden wird



- **Hashfunktionen** erweitern die Einwegfunktionen um Kollisionsresistenz

Eigenschaften von Hashfunktionen H

Eine Hashfunktion H bildet eine beliebig lange Nachricht m auf einen Hashwert fixer länge $H(m)$ und erfüllt die folgenden Eigenschaften:

1. **Einwegeigenschaft:**
   - Die Funktion $H(m)$ muss effizient berechenbar sein
   - Es darf nicht möglich sein von $H(m)$ auf m zu schließen
2. **Schwache Kollisionsresistenz:**
   - Es darf nicht möglich sein zu m ein anderes m' zu finden mit $m \neq m'$ und $H(m) = H(m')$
3. **Starke Kollisionsresistenz:**
   - Es darf nicht möglich sein zwei beliebige m und m' zu finden mit $m \neq m'$ und $H(m) = H(m')$

Schwache vs. Starke Kollisionsresistenz
Geburtstagsparadox (1/2)

Hochschule Fulda
University of Applied Sciences

- Beispiel: Geburtstag am gleichen Tag im Jahr

- **Schwache Kollisionsresistenz:** Wie viele Personen müssen im Raum sein, damit mit $\geq$ 50% Wahrscheinlichkeit eine Person am gleichen Tag **wie Sie** Geburtstag hat?

- **Starke Kollisionsresistenz:** Wie viele Personen müssen im Raum sein, damit mit $\geq$ 50% Wahrscheinlichkeit **zwei beliebige** Personen im Raum am gleichen Tag Geburtstag haben?

Schwache vs. Starke Kollisionsresistenz Geburtstagsparadox (2/2)

- Schwache Kollisionsresistenz (Bestimmter Tag): 253

- Starke Kollisionsresistenz (Beliebiger Tag): 26

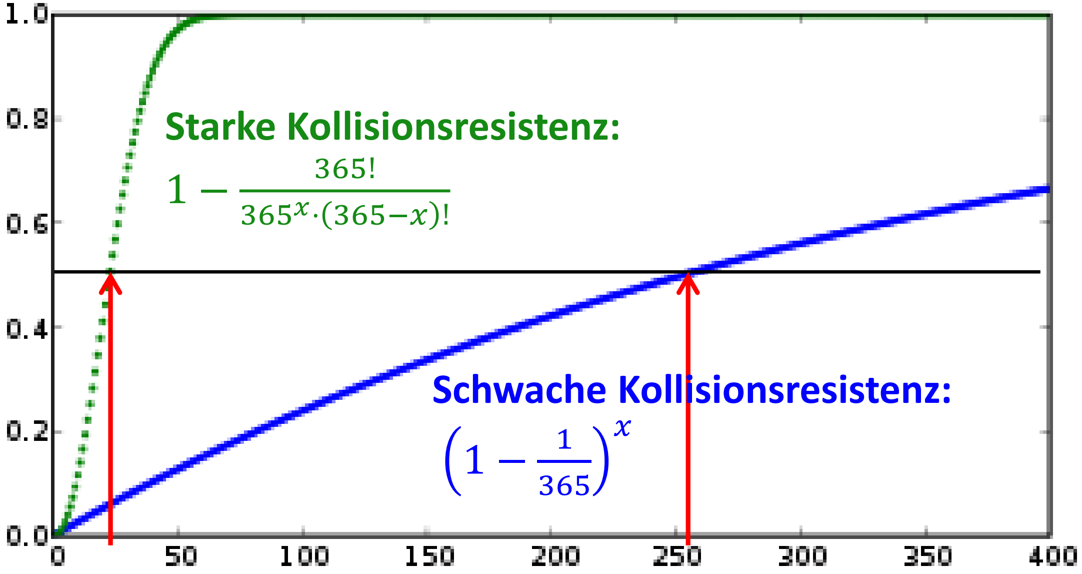


Bild Quelle: Wikipedia, Geburtstagsparadox

Wahl der Hashwertlänge

- Wie lang muss ein Hashwert sein um starke Kollisionsresistenz zu besitzen?

- **Wurzel** als Überapproximation der starken Kollisionsresistenz:
  Bei Elementen mit x-bit Länge sind bei einer Menge von $\sqrt{2^x} = 2^{x/2}$ zufällig gewählten Werten mit $\leq 50\%$ Wahrscheinlichkeit zwei Werte gleich
  - **Beispiel Geburtstagsparadox:** 365 Tage im Jahr, $\sqrt{365} = 19.10 \leq 26$

- ➔ Um **x-bit Berechnungssicherheit** zu erhalten, muss die **Ausgabelänge** einer **Hashfunktion $2x$-bit** sein:
  - 128-bit Sicherheit: 256-bit Ausgabelänge Hashfunktion
  - 256-bit Sicherheit: 512-bit Ausgabelänge Hashfunktion

Hashfunktionen und Sicherheit

Hochschule Fulda
University of Applied Sciences

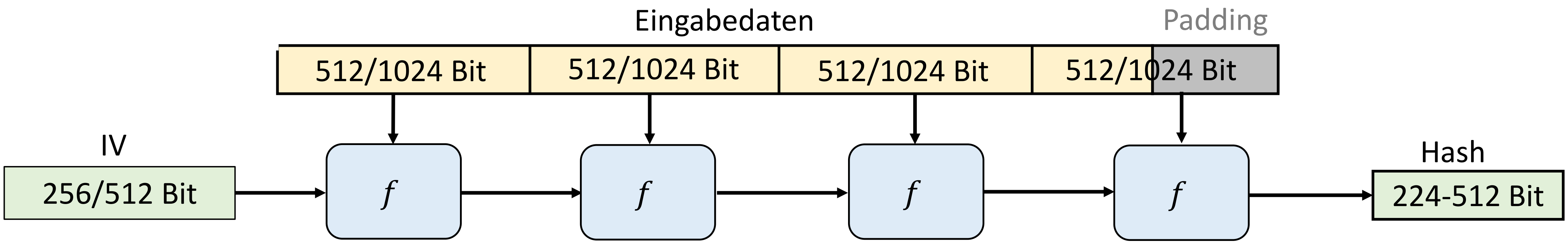
Hash Funktion	Länge Hashwert [bit]	Sicherheit
MD5	128	• Schwache Kollisionsresistenz theoretisch gebrochen (2^123 [SA09]) • Starke Kollisionsresistenz praktisch gebrochen (~35 Minuten Berechnung)
RIPEMD	128/160/256/320	• RIPEME-128 Starke Kollisionsresistenz praktisch gebrochen • RIPEMD-160/256/320 Sicher (Angriffe gegen Versionen mit reduzierten Runden)
SHA1	160	• Schwache Kollisionsresistenz theoretisch gebrochen (2^159.3 [KK12]) • Starke Kollisionsresistenz praktisch gebrochen (110 GPU Jahre Berechnung)
SHA2	224/256/384/512	• Angriffe gegen Versionen mit reduzierten Runden
SHA3	224/256/384/512	• Angriffe gegen Versionen mit reduzierten Runden

Standard Hash Algorithm 2 (SHA2)

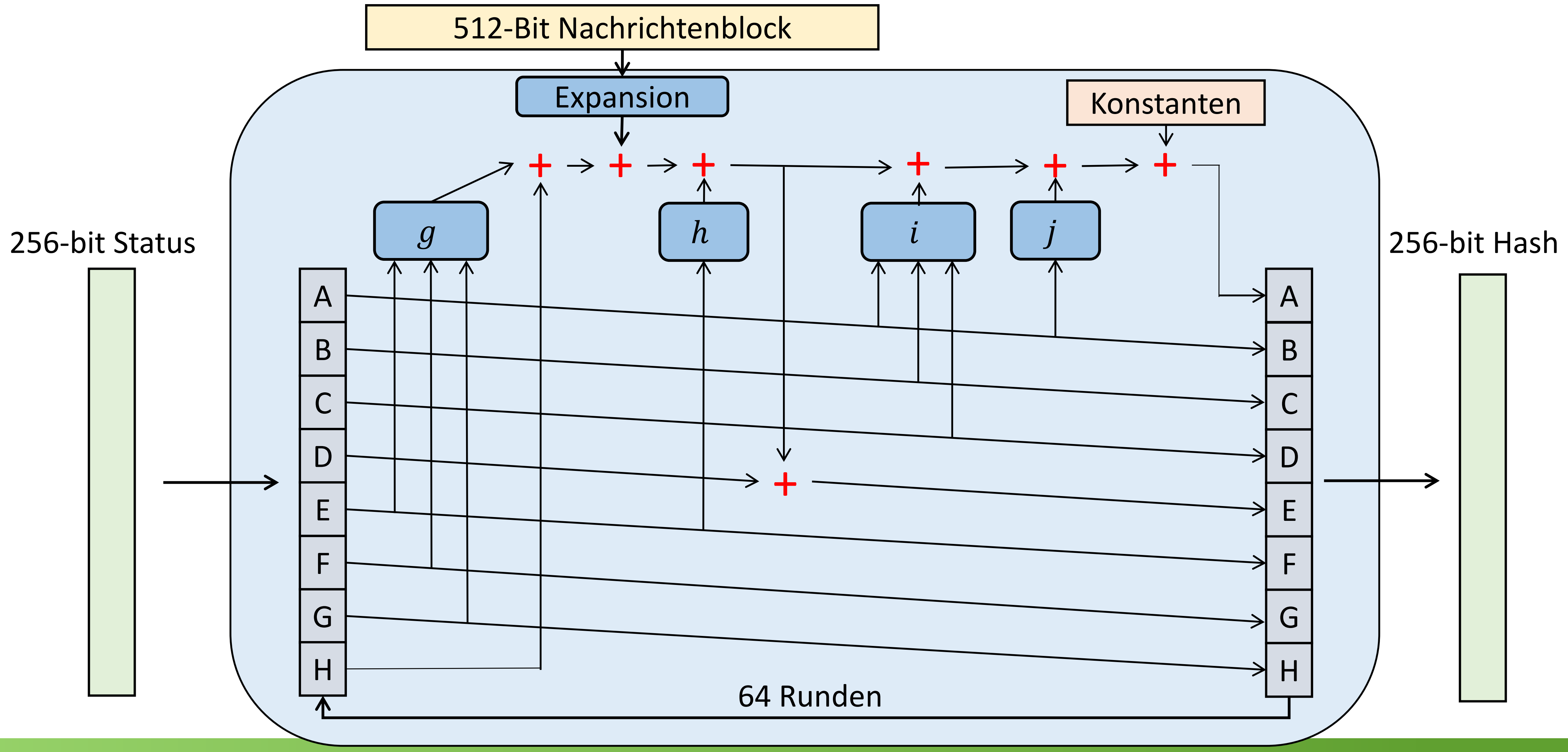
- Standardisiert vom US NIST im Jahr 2002 als Nachfolger des SHA1

- Kommt in den Varianten SHA2-224, SHA2-256, SHA2-384 und SHA2-512
  - Zahl am Ende ist die Länge des Hashwertes
  - SHA224/256 sowie SHA384/512 nutzen die gleiche Funktion mit unterschiedlicher Ausgabelänge

- SHA nutzt eine Kompressionsfunktion f, die 512/1024-Bit (SHA2-224/256 vs. SHA2-384/512) Eingabedatenblöcke mit einem 256/512-bit internen Zwischenstand verarbeitet



Beispiel SHA2-256 Kompressionsfunktion f



Standard Hash Algorithm 2 (SHA-2) Performance

- Hashfunktionen nutzen effiziente Operationen
  - Rotationen und Shifts
  - Logische Operationen (AND und XOR)
  - Additionen

- Allerdings benötigen sie viele Runden
  - 64 Runden für SHA2-224/256
  - 80 Runden für SHA2-384/512

Verfahren	Aufrufe pro Sek.
64-bit Mult.	1,401,372,784
AES-128 (SW)	22,413,312
AES-128 (HW)	175,308,800
RSA-2048 (SW)	Enc/Verify: 33,483 Dec/Sign: 822 KeyGen: 2
DH (SW)	Prime-2048: 1,302 ECC-256: 1,773
ECDSA-256 (SW)	Verify: 586 Sign: 1,736
SHA2-256 (SW)	**4,953,125**

Referenzen

[BD+21]: Cryptanalysis of the GPRS Encryption Algorithms GEA-1 and GEA-2. https://eprint.iacr.org/2021/819

[Tews12]: DECT Security Analysis. https://eprint.iacr.org/2012/321.pdf

[DHP]: Standardisierte Diffie-Hellman Primzahlen: https://datatracker.ietf.org/doc/html/rfc7919

[KeyLen - ECRYPT]: Übersicht zu den empfohlenen Schlüssellängen: https://www.keylength.com/. Asymmetrische Schlüsselgrößen basierend auf ECRYPT-CSA Empfehlungen von 2018

[BSI]:Klassen von Zufallszahlengeneratoren
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf;jsessionid=C84467E1E37C513F1ADBAAF3F9DEBA06.internet482?__blob=publicationFile&v=1

[Crypto_Implementierungsfehler]: https://people.csail.mit.edu/nickolai/papers/lazar-cryptobugs.pdf

[Petya]: https://blog.checkpoint.com/wp-content/uploads/2016/10/GreatCryptoFailuresWhitepaper_Draft2.pdf

[Compiler]: https://wiki.sei.cmu.edu/confluence/display/c/MSC06-C.+Beware+of+compiler+optimizations

[PS3]: http://exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/

Referenzen

[SECENG]: Security Engineering von Ross Andersson. 3te Auflage

[ND+12]: N. Heninger, Z. Durumeric, E. Wustrow, J. Halderman: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. https://factorable.net/weakkeys12.conference.pdf

[Cop96]: D. Coppersmith (1996). Finding a Small Root of a Univariate Modular Equation. Lecture Notes in Computer Science. doi:10.1007/3-540-68339-9_14. ISBN 978-3-540-61186-8.

[W90]: M. WIENER, Cryptanalysis of short RSA secret exponents.

[SA09]: Yu Sasaki and Kazumaro Aoki: Finding Preimages in Full MD5 Faster than Exhaustive Search. https://iacr.org/archive/eurocrypt2009/54790136/54790136.pdf

[KK12] Simon Knellwolf and Dimtry Khvratovich, New Preimage Attacks Against Reduced SHA1, https://eprint.iacr.org/2012/440.pdf

[B99] D. Boneh: RSA Survey. https://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf

[B98] D. Bleichenbacher. Chosen Ciphertext Attacks Against protocols Based on the RSA Encryption Standard PKCS#1. http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf

[DSA] NIST FIPS 186-4, Digital Signature Standard, https://csrc.nist.gov/publications/detail/fips/186/4/final

[DHBW] E. List, J. Wenzel, Folien zur Vorlesung Kryptographie WS16/17

[HMAC]: RFC 2104: https://datatracker.ietf.org/doc/html/rfc2104