IT-SICHERHEIT UND DATENSCHUTZ

KAPITEL 2 - KRYPTOGRAPHIE

buchmann@hft-leipzig.de

LERNZIEL UND AUFBAU DIESES KAPITELS

- Lernziele
  - Sie können die verschiedenen Verschlüsselungsverfahren anwenden, und kennen ihre Stärken und Schwächen.
  - Sie können die Ziele der Kryptographie aus Datenschutzsicht erklären, sowie passende Sicherheitskriterien und Verschlüsselungsverfahren erläutern.

- Inhalt
  - Einführung in die Kryptographie
  - Symmetrische Verschlüsselung
  - Asymmetrische Verschlüsselung
  - Secure Multiparty Computation

ZIELE DER IT-SICHERHEIT

- „Security triad“
  - **Vertraulichkeit**: Asset ist nur Autorisierten zugänglich
  - **Integrität:** Asset kann nur von Autorisierten modifiziert werden
  - **Verfügbarkeit**: Asset kann von Autorisierten genutzt werden

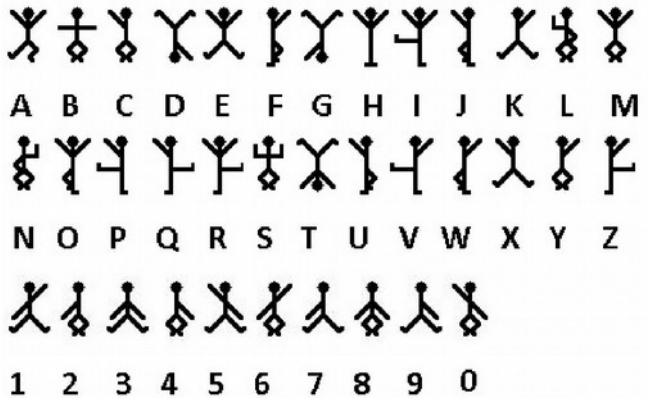
- ISO 7498-2 fügt hinzu
  - **Authentisierung**: Identität eines Senders wird überwacht
  - **Nichtabstreitbarkeit**: Sender kann nicht abstreiten, das eine Nachricht von ihm kam
- US Department of Defense fügt hinzu
  - **Auditierbarkeit**: Alle Aktionen mit dem Asset sind nachvollziehbar

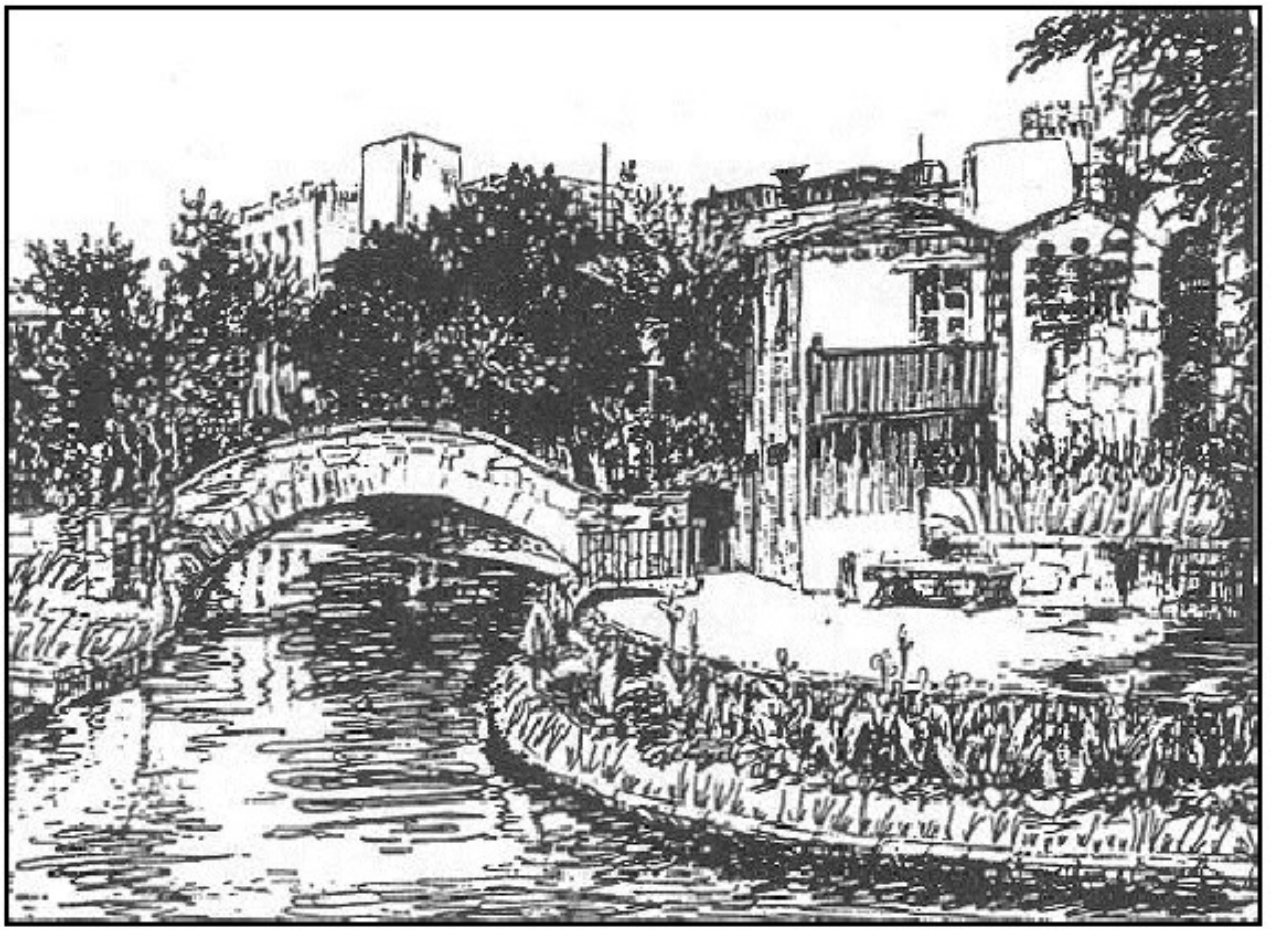
EINFÜHRUNG IN DIE KRYPTOGRAPHIE

WER ETWAS ZU VERBERGEN HAT…

...versucht es zu verstecken.

- Unsichtbare Tinte *(Zitronensaft trocken unsichtbar, bei erhitzen braun)*
- Codebücher
- Steganographie *(rechts: Morsecode in Grashalm-länge codiert)*
- Geheinschrift *(unten: nach Arthur C. Doyle)*





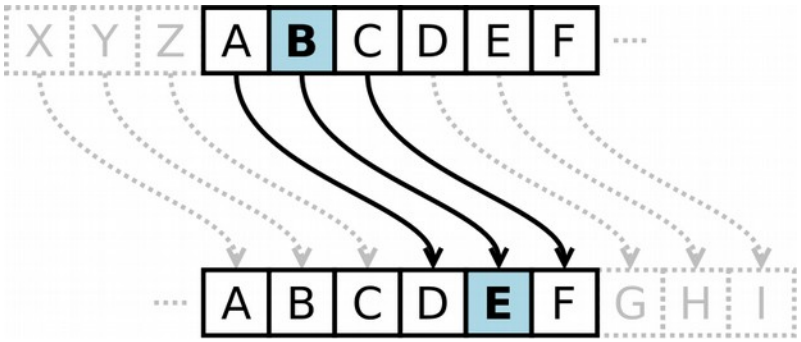
ANFÄNGE DER VERSCHLÜSSELUNG

- Motivation: ein Angreifer soll eine Nachricht nicht lesen können, obwohl er die Methode der Übertragung kennt

- Skytale
  - Mittelmeerraum (Griechische Kleinstaaten, Persien)
  - Um 500 v. Chr.

- Caesar-Chiffre
  - Nach Julius Caesar, um 100 v. Chr.
  - Verschiebechiffre
    $X \rightarrow X + 3 \bmod 26$

  (wer kennt noch ROT-13?)



Bilder: Wikimedia

KLASSISCHE VORGEHENSWEISE

- Security through obscurity
  - Mache das Verfahren möglichst kompliziert und halte es geheim

- break-it fix-it break-it fix-it...
  - Alice entwickelt Verschlüsselung
  - Carol versucht Verfahren zu knacken
  - Nach einiger Zeit kein Angriff: → Verfahren gilt als „sicher“
    Bei erfolgreichem Angriff: → Alice entwickelt nächstes Verfahren

- Beispiel von so entstandenen Verfahren
  - Enigma, 2. WK (drehbare Rotoren mit unregelmäßiger Verzahnung)

STAND DER TECHNIK

- Kerckhoff's Maxime (1883):
  "The security of a cryptosystem must not depend on the secrecy of the algorithm. The security is based only on the secrecy of the key. "

- Etabliertes Prinzip in der modernen Kryptographie, denn
  - Sicherheit von Geheimhaltung des Algorithmus abhängig:
    Algorithmus wird öffentlich → **alle Nutzer des Algorithmus** betroffen
  - Sicherheit von Geheimhaltung des Schlüssels abhängig:
    Schlüssel wird öffentlich → **nur Nutzer des Schlüssels** betroffen

- Ein veröffentlichter Algorithmus kann von Experten geprüft werden

STAND DER TECHNIK (CONT'D.)

- Sicherheit bestimmbar durch
  - **Ressourcen des Angreifers**
    - *Polynomial beschränkter Angreifer*
  - **Ziele des Angreifers**
    - den Schlüssel K herausfinden
    - min. ein Klartext-Bit p ϵ P vom cipher C → $enc_K(P)$ herausfinden
    - Metainformationen von C herausfinden, z.B. die Länge von K, die Länge des Texts P, die Sprache des Texts P, den Absender etc.
  - **Angreifermodell** (nächste Folie)

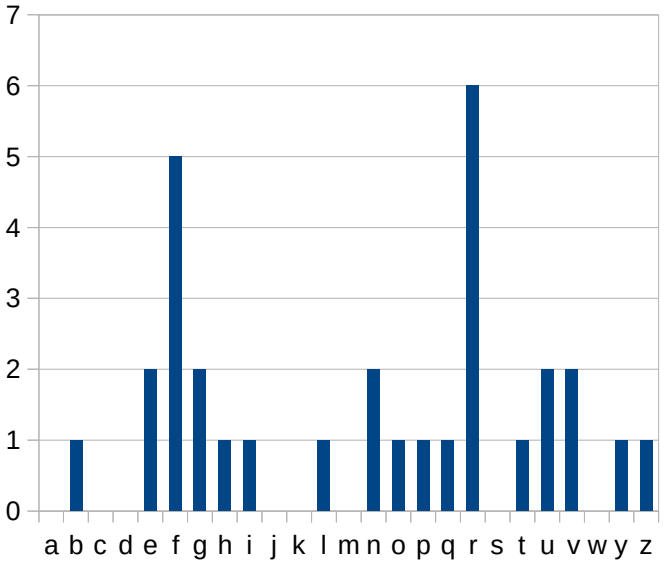
ANGREIFERMODELLE

- Angreifermodelle unterscheiden nach Wissen des Angreifers
- **Ciphertext-Only**
  - Angreifer hat nur Ciphertext zur Verfügung
- **Known-Plaintext**
  - Angreifer kennt Ciphertext-Klartext-Paare
- **Chosen-Plaintext**
  - Angreifer kann aus Klartext selbst Ciphertext erzeugen
    (typisch für Public-Key-Verfahren, öffentlicher Schüssel bekannt)
- **Chosen-Ciphertext**
  - Angreifer kann aus gewählten Ciphertexten Klartext erzeugen

BEISPIEL: BRECHEN DES CAESAR-CHIFFRES

- Beispiel Verschiebechiffre ROT13:
  X → X + 13 mod 26

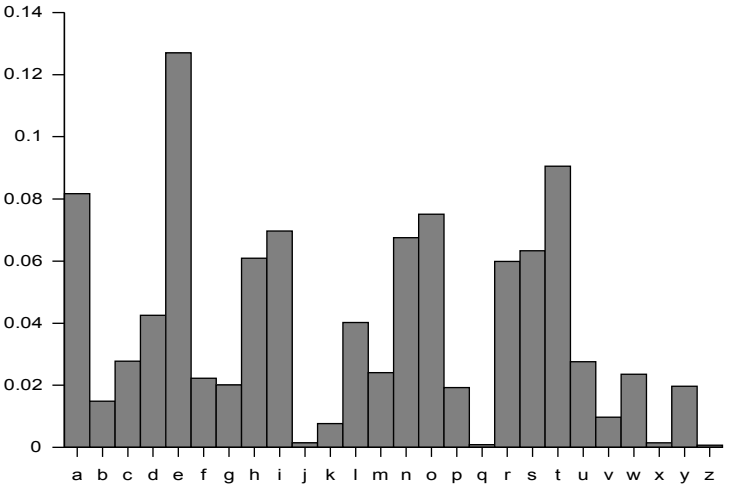
  K: „This should be a very secret message!“
  C: „Guvf fubhyq or n irel frperg zrffntr!“



Häufigkeitsverteilung der Buchstaben im Ciphertext

- **Ziele des Angreifers**
  - Schlüssel herausfinden
  - Klartext erhalten

- **Ciphertext-Only**-Angriff:
  - „e“ ist häufigster Buchstabe, wurde „e“ zu „r“?
  - Bei bekanntem Verfahren ist Verschüsselung gebrochen, „r“→„e“=13



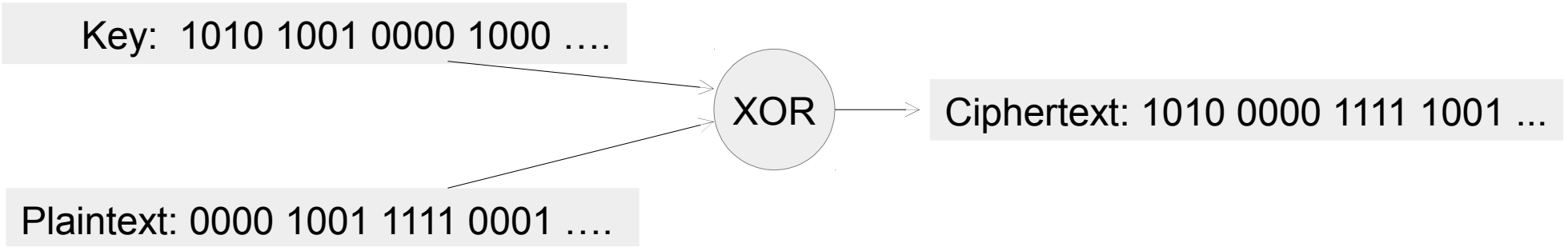
Häufigkeitsverteilung der Buchstaben in eng. Texten

ONE-TIME PAD

- One-time Pad ist das *einzige* Verfahren, das absolute Sicherheit garantiert, wenn Schlüssel folgende Eigenschaften hat:
  - Echt zufälliger Schlüssel
  - Genauso lang wie Nachricht
  - Bits haben Gleichverteilung
  - Schlüssel bleibt geheim
  - Schlüssel wird nur 1x verwendet

```
CIHJT UUHML FRUGC ZIBGD BQPNI PDNJG LPLLP YJYXM
DCXAC JSJUK BIOYT MWQPX DLIRC BEXYK VKIMB TYIPE
UOLYQ OKOXH PIJKY DRDBC GEFZG UACKD RARCD HBYRI
DZJYO YKAIE LIUYW DFOHU IOHZV SRNDD KPSSO JMPQT
MHQHL OHQQD SMHNP HHOHQ GXRPJ XBXIP LLZAA VCMOG
AWSSZ YMFNI ATMON IXPBY FOZLE CVYSJ XZGPU CTFQY
HOVHU OCJGU QMWQV OIGOR BFHIZ TYFDB VBRMN XNLZC
```

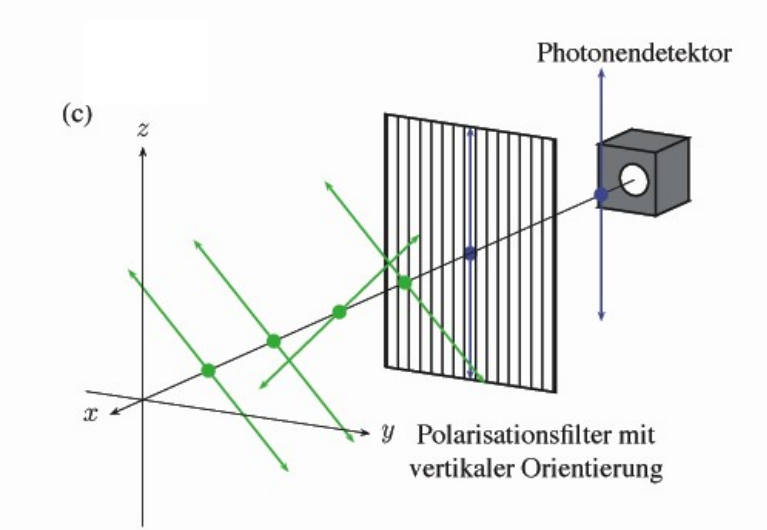
- Encryption: $X_i \rightarrow X_i$ XOR P_i



ONE-TIME PAD & QUANTENPHYSIK

- Beitrag der Quantenkryptographie
  - Laser sendet polarisierte, verschränkte Photonen zum Schlüsselaustausch durch Glasfaser zum Empfänger
  - **Sicherer Kanal:** Messung der Photonen führt dank Quanteneigenschaften zur Veränderung der Photonenverteilung
- Problem: teure Hardware und komplizierte physikalische Prozesse, komplizierte Handhabung

```
CIHJT UUHML FRUGC ZIBGD BQPNI PDNJG LPLLP YJYXM
DCXAC JSJUK BIOYT MWQPX DLIRC BEXYK VKIMB TYIPE
UOLYQ OKOXH PIJKY DRDBC GEFZG UACKD RARCD HBYRI
DZJYO YKAIE LIUYW DFOHU IOHZV SRNDD KPSSO JMPQT
MHQHL OHQQD SMHNP HHOHQ GXRPJ XBXIP LLZAA VCMOG
AWSSZ YMFNI ATMON IXPBY FOZLE CVYSJ XZGPU CTFQY
HOVHU OCJGU QMWQV OIGOR BFHIZ TYFDB VBRMN XNLZC
```



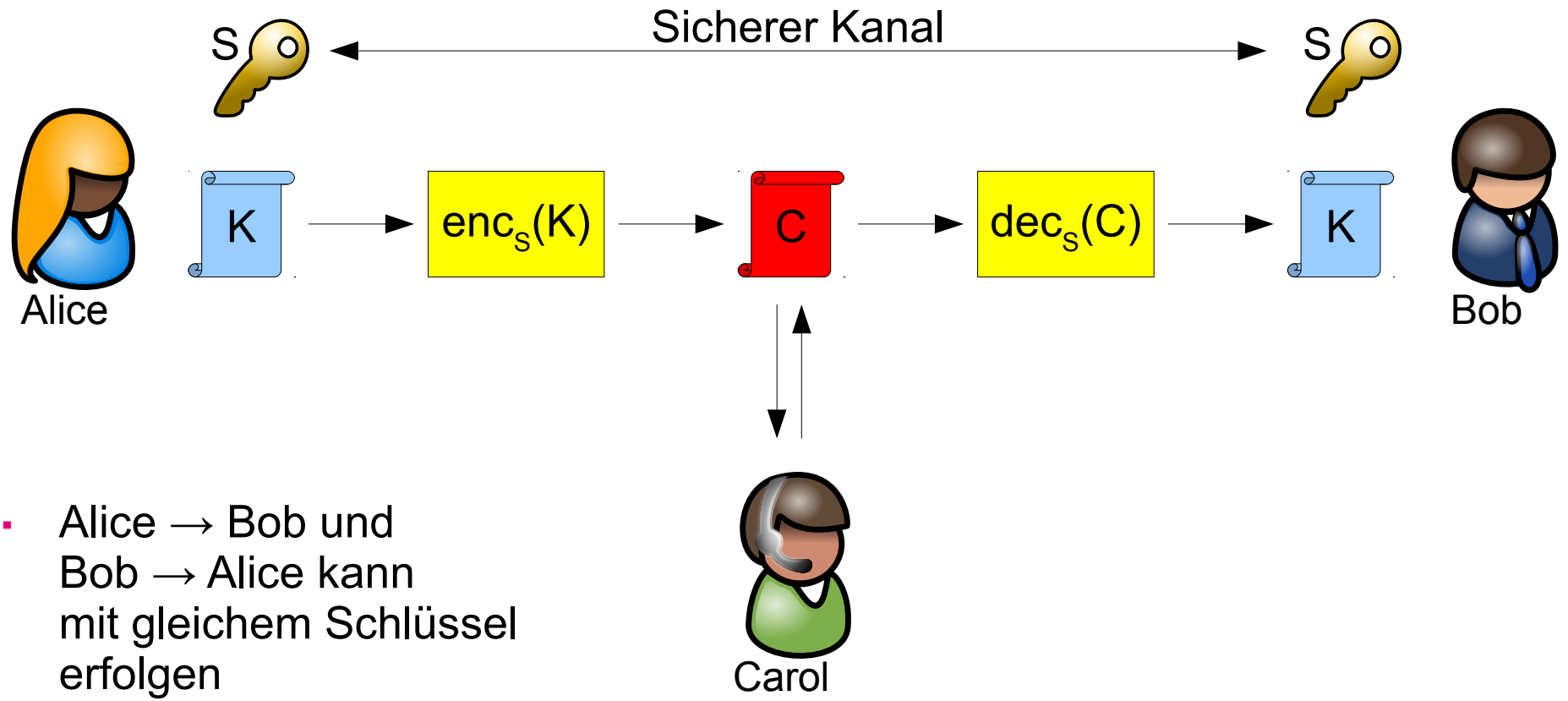
KRYPTOGRAPHIE UND DATENSCHUTZ

- Allgemein: Zugriffsschutz
  - Symmetrische, Asymmetrische Verschlüsselung, z.B. für HTTPS/SSL
- Neue Ziele
  - Plausible Deniability (Glaubhafte Abstreitbarkeit)
    - Gegenkonzept zur Verbindlichkeit
    - Urheberschaft einer Nachricht plausibel leugnen können
  - Separation of Duties (Funktionstrennung bei Nichtverknüpfbarkeit)
    - Abstrakter als Anonymisierung; hier geht es um Daten allgemein
- Neuartige Anwendungen
  - Homomorphe Verschlüsselung: Berechnung von Operationen auf verschlüsselten Daten
  - Secure Multiparty Computation: Verteilte Berechnung, ohne Rohdaten oder Zwischenergebnisse öffentlich zu machen

SYMMETRISCHE VERSCHLÜSSELUNG

SYMMETRISCHE VERSCHLÜSSELUNG

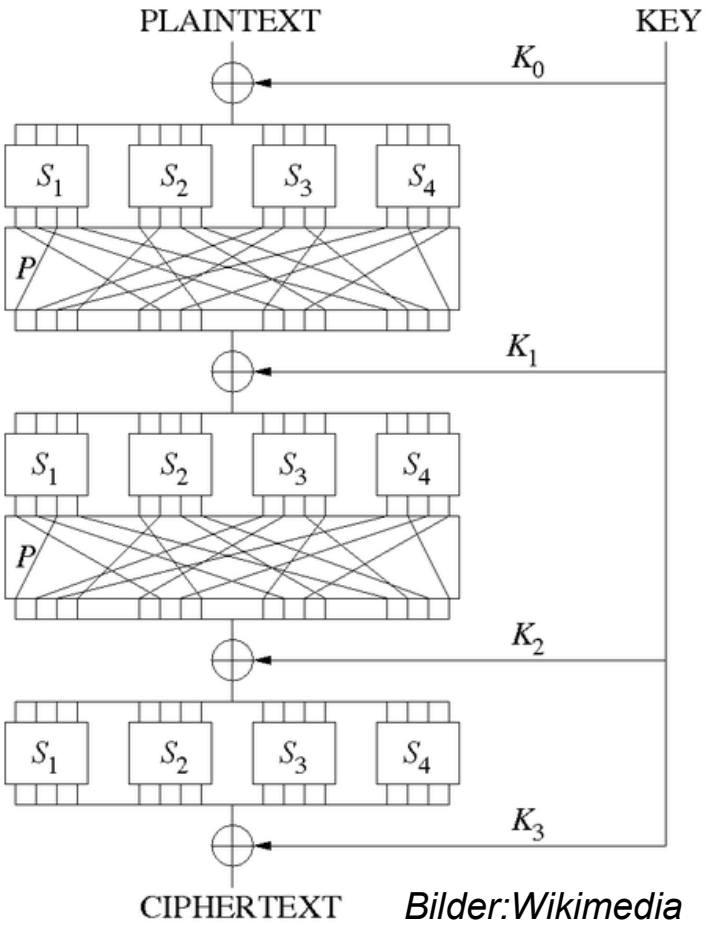
- Alice und Bob haben den gleichen Schlüssel
  - Der selbe Schlüssel zur Ver- und Entschlüsselung
  - Schlüssel ist geheim, über gesicherten Kanal übertragen



- Alice → Bob und
  Bob → Alice kann
  mit gleichem Schlüssel
  erfolgen

BLOCKCHIFFRIERUNG

- Ziel: Klartexte *beliebiger* Länge mit *kurzem* Schlüssel verschlüsseln
- Designprinzip: Substitutions-Permutations-Netzwerke
  - Aus Schlüssel *Rundenschlüssel* generieren
  - Mehrere *Verschlüsselungsrunden*
    - Rundenschlüssel auf Eingabe addieren
    - Ergebnis in Blöcke aufteilen
    - Substitutionsbox (S1...S4): Einen Block durch einen anderen substituieren
    - Permutationsbox (P): Entstehende Blöcke durchmischen
  - In der letzten Verschlüsselungsrunde nochmals Rundenschlüssel addieren
  - Entschlüsselung erfolgt genauso wie Verschlüsselung, nur rückwärts

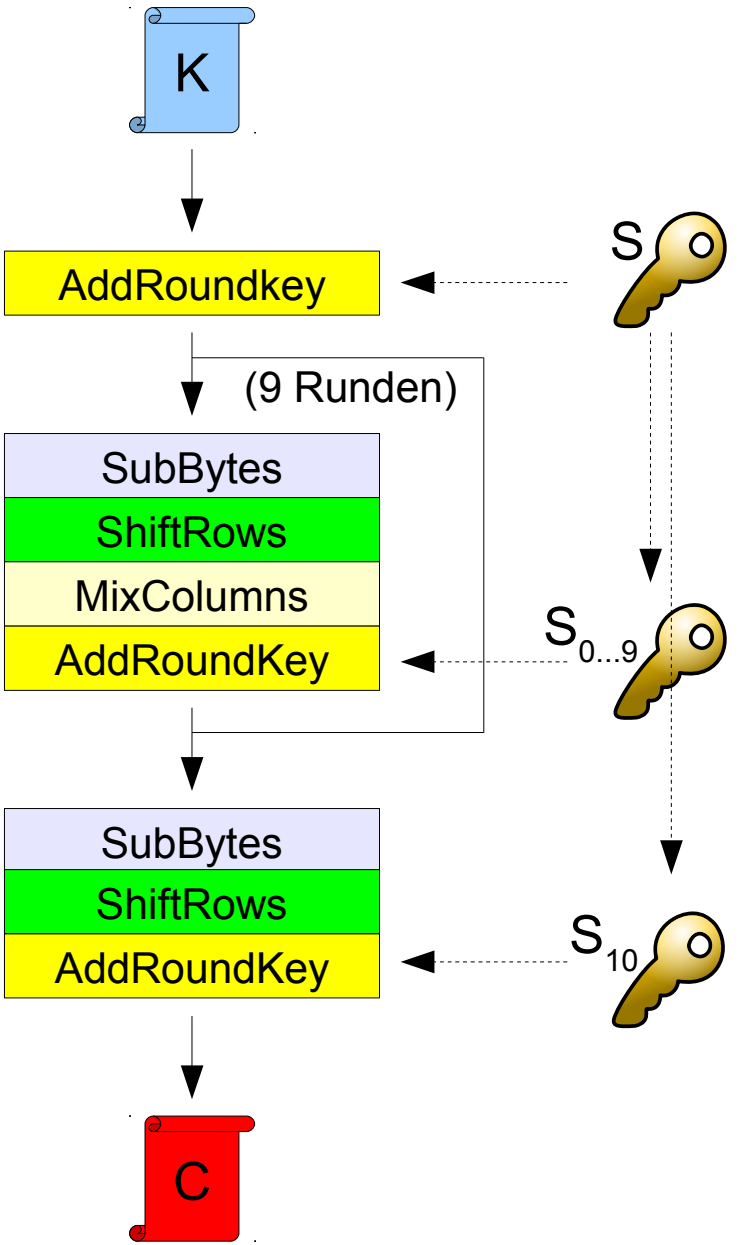


Bilder:Wikimedia

ADVANCED ENCRYPTION STANDARD (AES)

- Symmetrische Verschlüsselung
- Schlüssel 128, 192 oder 256 Bit
- Anwendungen in IPSec, WLAN (WPA2), SSH, HTTPS, etc.

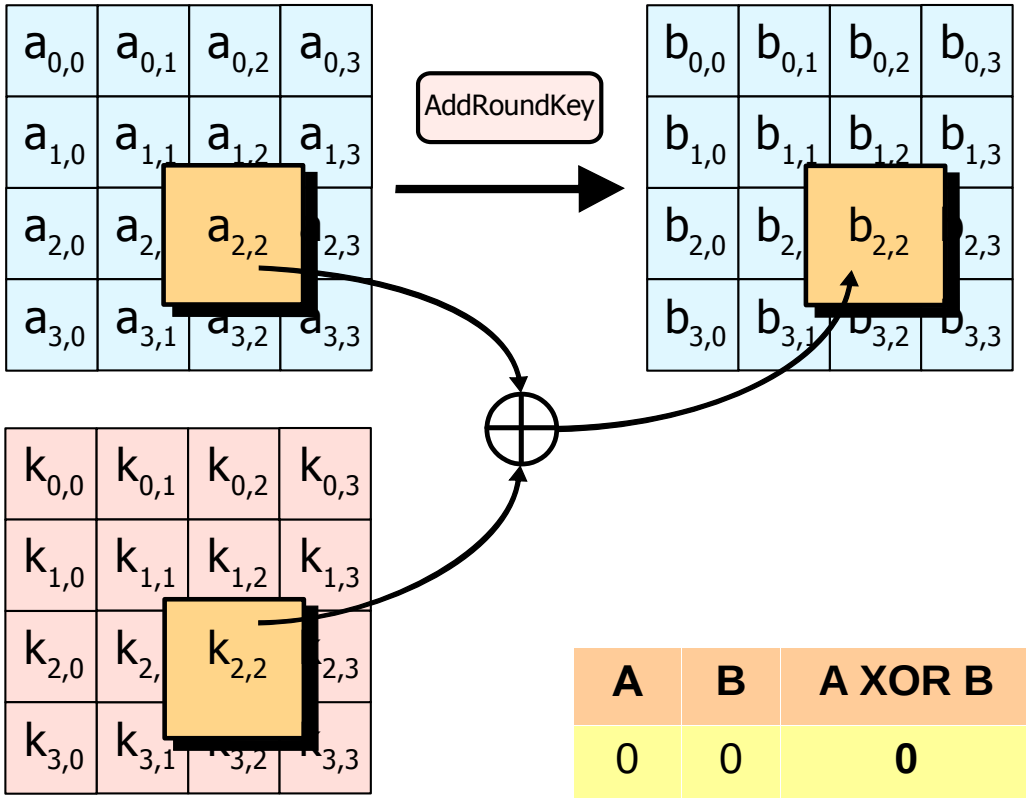
- AES ist ein Blockchiffre
  - Ein Block ist
    - Tabelle mit 4 Zeilen, 4 Spalten
    - 1 Byte pro Zelle
    - 128 Bit insgesamt
  - S-Boxen: SubBytes
  - P-Boxen: ShiftRows, MixColumns



Frage: Warum ist die letzte Operation AddRoundKey?

ADDITION DES RUNDENSCHLÜSSELS IN AES

- Methode **AddRoundKey**
- Bitweise XOR-Verknüpfung von einem Block und dem Rundenschlüssel
- Rundenschlüssel ebenfalls 4x4-Tabelle
- *Anmerkung: Schlüssel wird nur hier angewendet, alle anderen Operationen sind von den Eingabedaten abhängig*



A	B	A XOR B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Bilder:Wikimedia

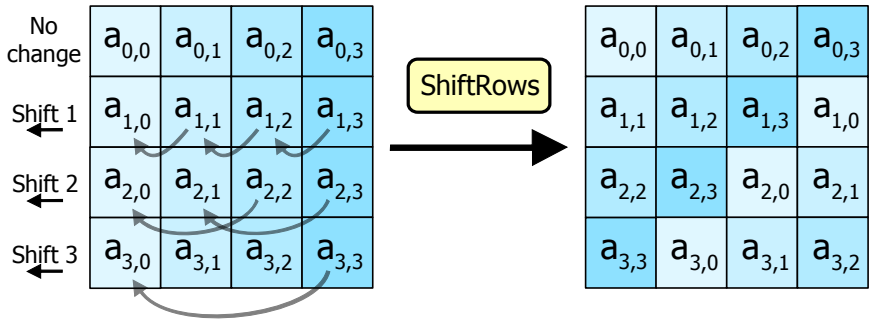
S-BOXEN IN AES

- Methode **SubBytes**
- Jedes Byte der Tabelle wird durch ein anderes in der S-Box ersetzt
  → Monoalphabetische Kodierung
- S-Box entsteht als Kehrwert einer Matrize, die auch in der P-Box genutzt wird
  - *Galois-Körper GF(2^8),*
    → fragen Sie einen Mathematiker ;-)
  - Tabelle in Hex-Code
  - Bsp.: aus 12 wird c9

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

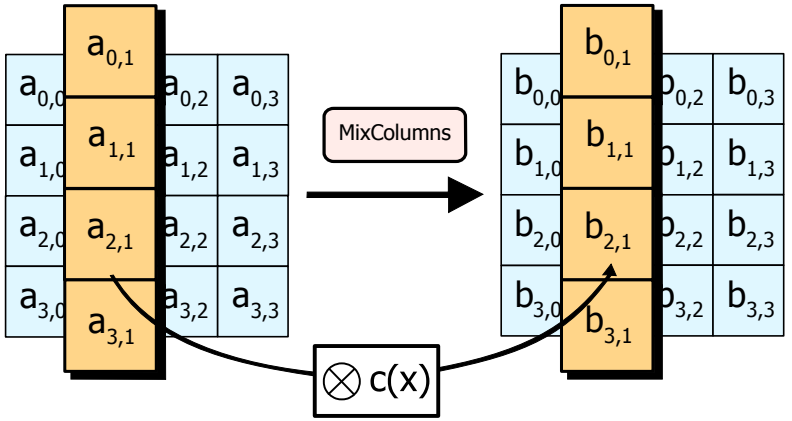
P-BOXEN IN AES

- Methode **ShiftRows**
  - Jede Zelle um Nummer seiner Zeile nach Links schieben
  - Links herausfallende Zellen rechts einfügen



- Methode **MixColumns**
  - Matritzenmultiplikation

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{pmatrix} = \begin{pmatrix} b_{0,i} \\ b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{pmatrix}$$



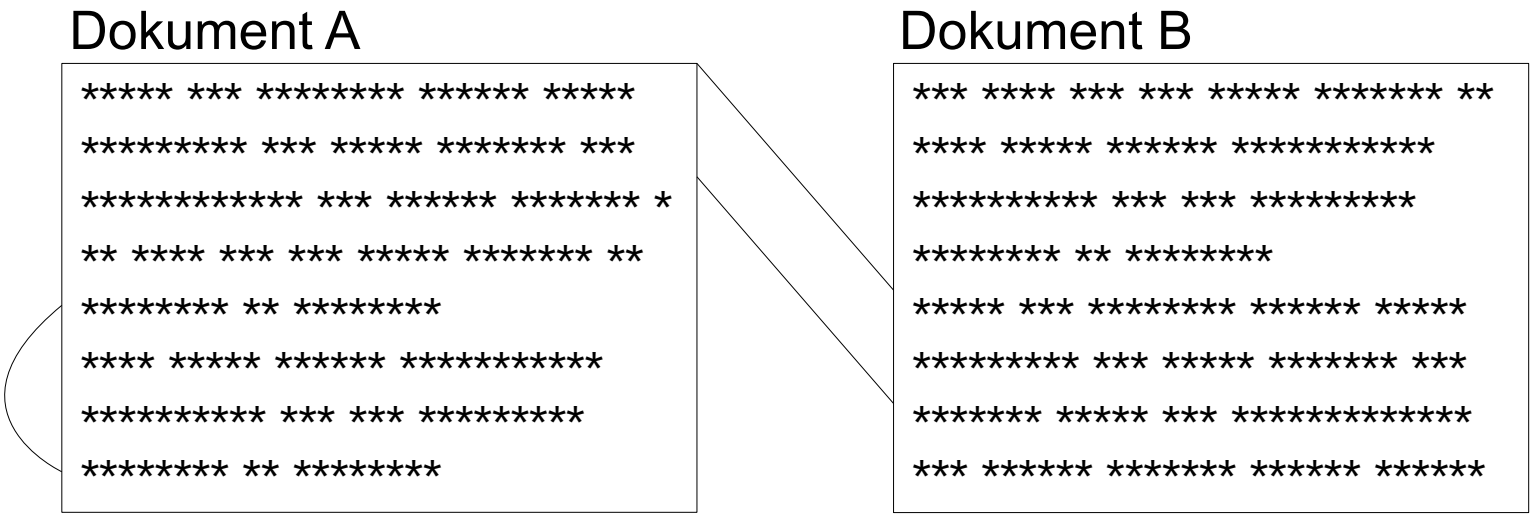
Bsp.: $b_{01}=2*a_{01}+3*a_{11}+1*a_{21}+1*a_{31}$

  - alle 4 Eingabebytes beeinflussen jedes Ausgabebyte

Bilder:Wikimedia

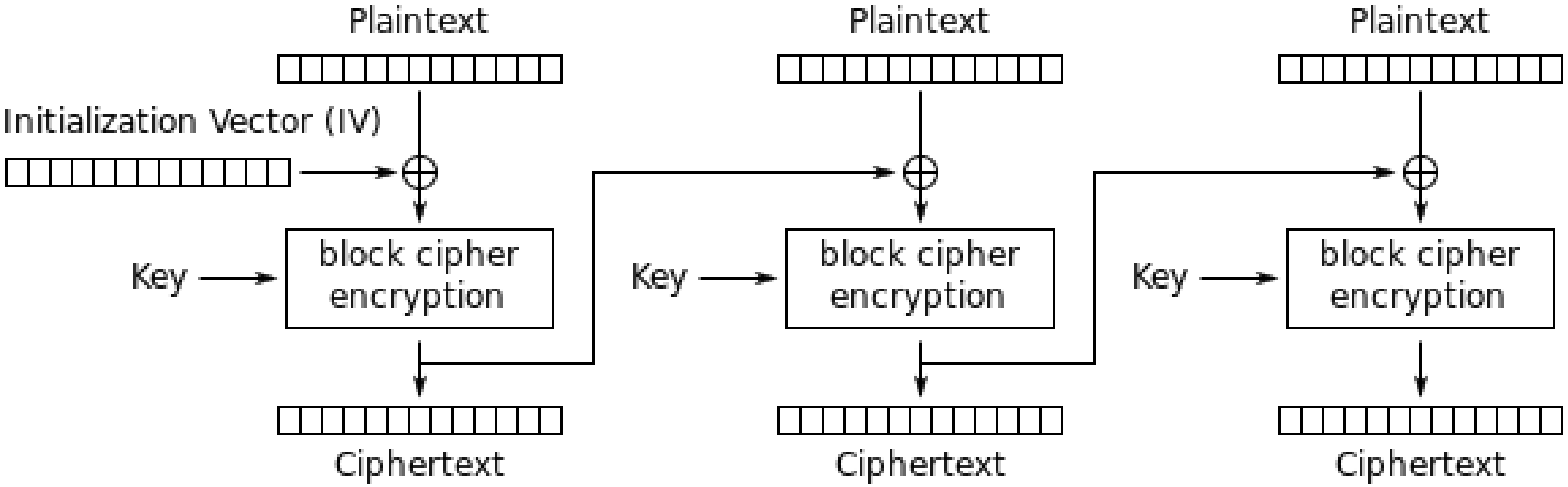
WIE SICHER IST DAS, WAS SIE BISHER GESEHEN HABEN?

- Bis hierher: **Electronic Codebook Mode**
  - Jeder Block einzeln für sich verschlüsselt
- Was passiert, wenn
  - ein Bit in einem Cipher-Block verändert wird?
  - ein Cipher-Block aus dem verschlüsselten Dokument gelöscht wird?
  - identische Klartextabschnitte existieren?
  - nur einen bestimmten Block entschlüsseln möchten?



CIPHER BLOCK CHAINING MODE

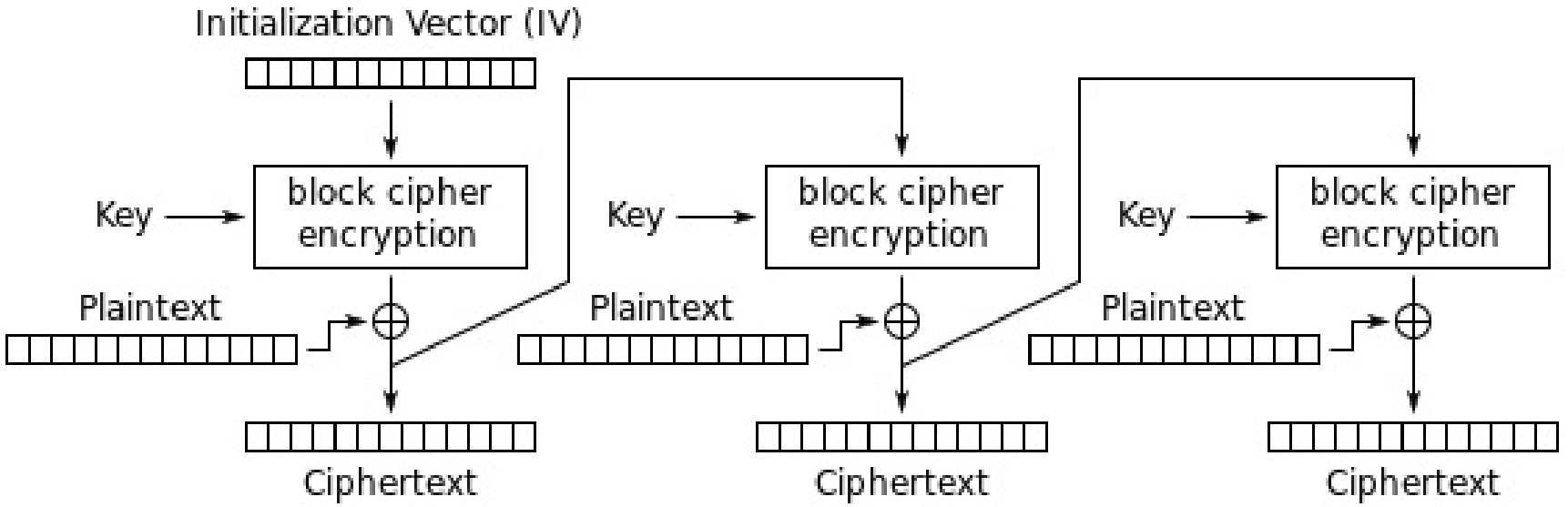
- Erster Block XOR Initialisierungsvektor mit Zufallszahlen
- Ergebnis wird verschlüsselt und ergibt Cipher-Block
- Jeder Cipher-Block per XOR mit dem nächsten Klartext-Block verkettet
  - identische Klartextblöcke → unterschiedliche Cipher-Texte
  - Bitfehler in einem Block zerstört diesen und den nächsten Block



(Image: Wikimedia)

CIPHER FEEDBACK MODE

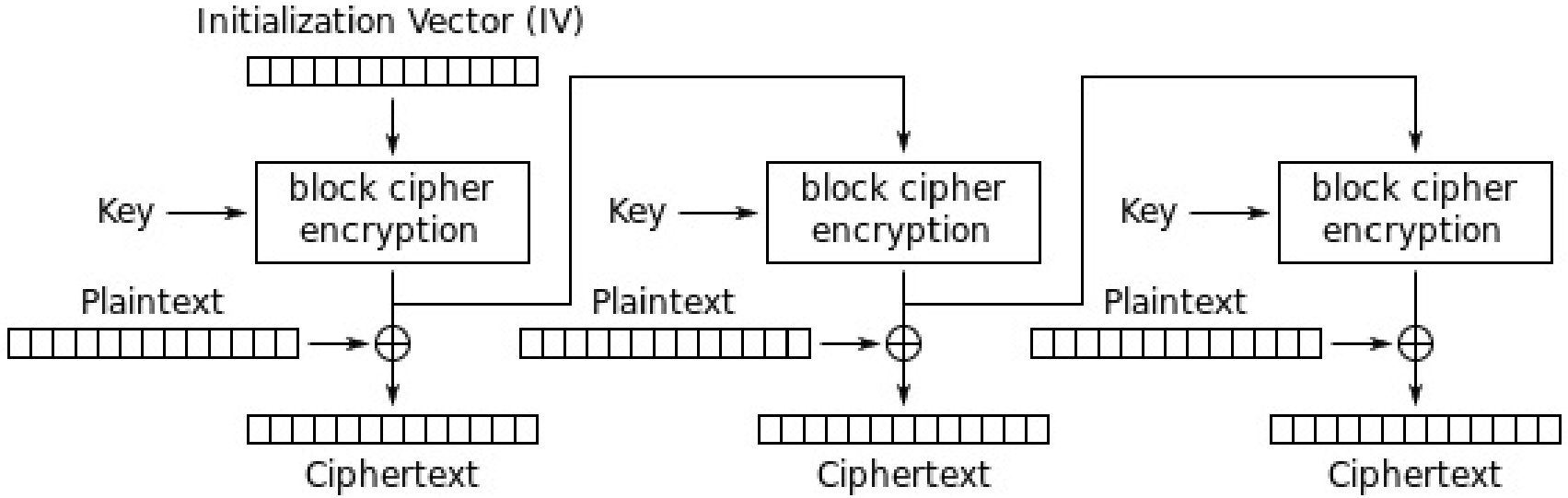
- Initialisierungsvektor wird verschlüsselt
- Jeder Klartext-Block per XOR mit dem Cipher-Block verkettet
- Ergebnis wird verschlüsselt
  - Bitfehler zerstört den betreffenden Block
  - Jeder Block ist der Initialisierungsvektor des Folgeblocks



(Image: Wikimedia)

OUTPUT FEEDBACK MODE

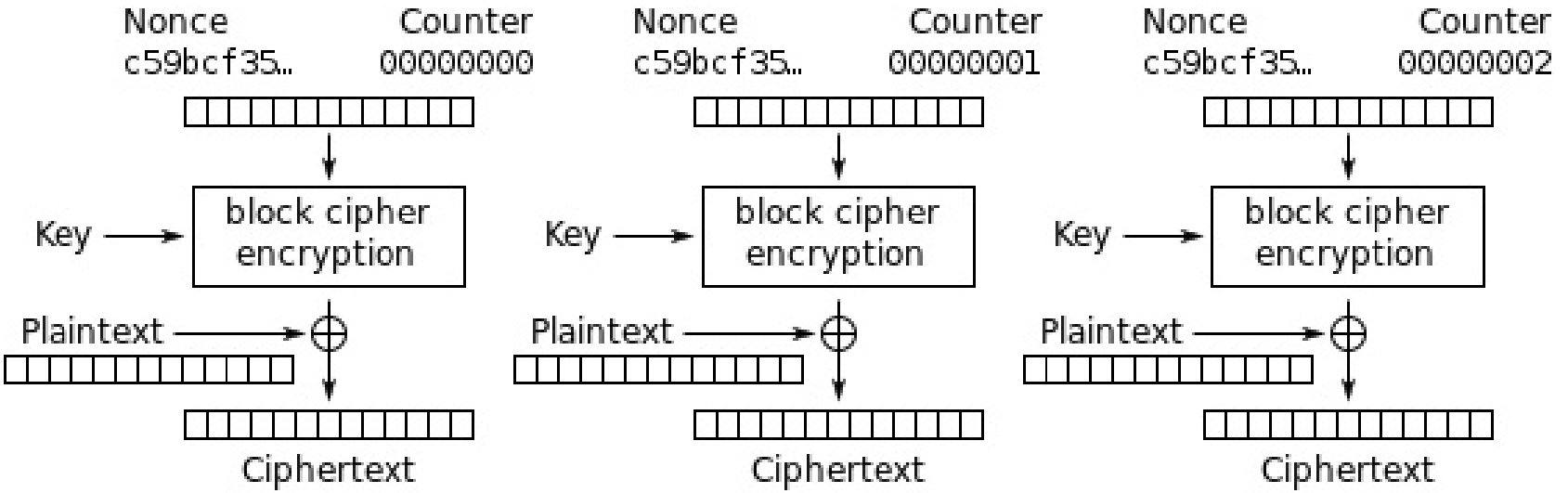
- Initialisierungsvektor wird verschlüsselt
- Verschlüsselter Initialisierungsvektor wird an die weiteren Blöcke weitergereicht
- Jeder Klartext-Block per XOR mit dem Ciphertext-Block verkettet
  - Bitfehler zerstört nur das betreffende Bit im Klartext-Block
  - Bitfolge für XOR kann ohne Klartext vorausberechnet werden



(Image: Wikimedia)

COUNTER MODE

- Vereinfachung des Output Feedback Mode
- Counter wird auf Initialisierungsvektor aufaddiert, anstelle verschlüsselten Initialisierungsvektor an den nächsten Block weiterzureichen
  - Bitfehler zerstört nur das betreffende Bit im Klartext-Block
  - Bitfolge für das XOR kann für beliebigen Block vorausberechnet werden



(Image: Wikimedia)

DISKUSSION AES

- AES ist performant
  - Nur „billige“ Operationen
  - 128 Bit Blockgröße passt sehr gut in CPU- oder GPU-Register
  - Hardware-Implementierung möglich, z.B. auf FPGA

- AES ist sicher
  - Bester Angriff (2011) nur um Faktor 4 schneller als Brute Force

    → kein ernsthafter Angriff, vollst. Suche 2^{128} – 2^{256} Möglichkeiten, nun Suchraum reduziert auf 2^{126} - 2^{254} Möglichkeiten

- Nachteile
  - Sicherer Kanal für die Schlüsselübertragung
  - Viele Kommunikationspartner = viele Schlüssel (s. nächste Folie)

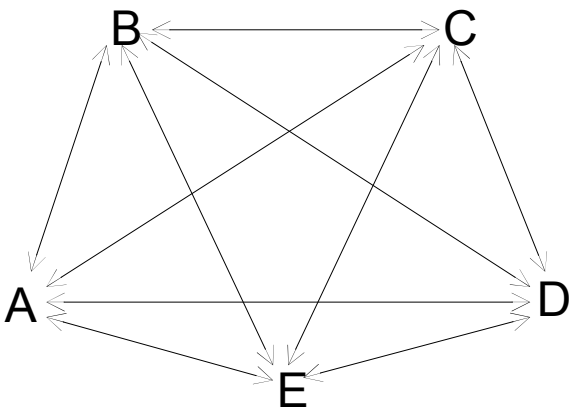
ANZAHL DER SCHLÜSSELPAARE BEI N TEILNEHMERN

- N Nutzer wollen miteinander privat kommunizieren
  - Jeder der N Nutzer benötigt N-1 Schlüssel um mit den anderen zu kommunizieren
  - Gesamtanzahl Schlüssel im System:

$$\binom{N}{2} = \frac{\left(N\left(N-1\right)\right)}{2}$$

- Beispiel für 5 Nutzer:
  10 Schlüssel

- Beispiel für 500 Nutzer:
  124.750 Schlüssel



STROMVERSCHLÜSSELUNG

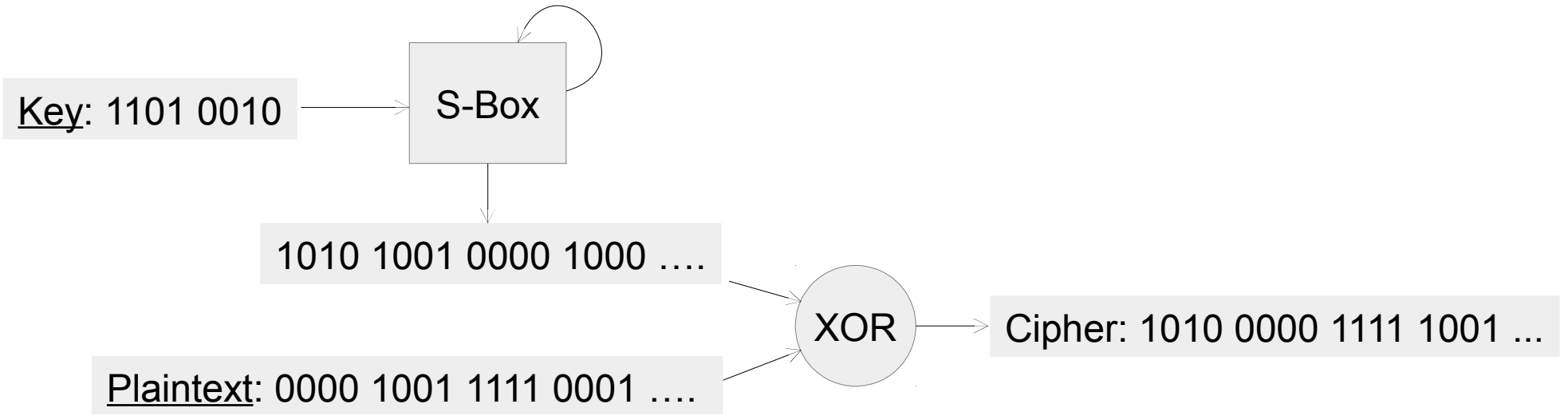
STROMVERSCHLÜSSELUNG VS. BLOCKVERSCHLÜSSELUNG

- **Blockverschlüsselung**
  - Eingabedaten werden *Block für Block* verschlüsselt
  - Für die Verschlüsselung von im Ganzen vorliegenden Daten z.B. Dateien, Bilder, aber auch größere Datenströme
  - Ansätze: DES, AES, RC2, Blowfish, Twofish

- **Stromverschlüsselung**
  - Eingabedaten werden *sequenziell Bit für Bit* verschlüsselt
  - Für die Verschlüsselung von Echtzeitdaten z.B. Voice over IP, Video, Audio
  - Üblicherweise weniger sicher wie Blockverschlüsselung
  - Ansätze: RC4, E0 (Bluetooth-Standard), WEP, WPA2, WPA3

RC4

- 1987 von Ron Rivest (das R in RSA) entwickelt
- In HTTPS, SSH, WEP/WPA (wegen Sicherheitslücken werden heute Nachfolger mit ähnlicher S-Box verwendet,)
- Funktionsweise:
  - Eine rekursive S-Box ähnlich derjenigen in AES erzeugt einen endlosen Bitstrom $X_{i=0...\infty}$ aus dem Schlüssel K
  - Verschlüsselung: $X_i \rightarrow X_i$ XOR P_i



DISKUSSION RC4

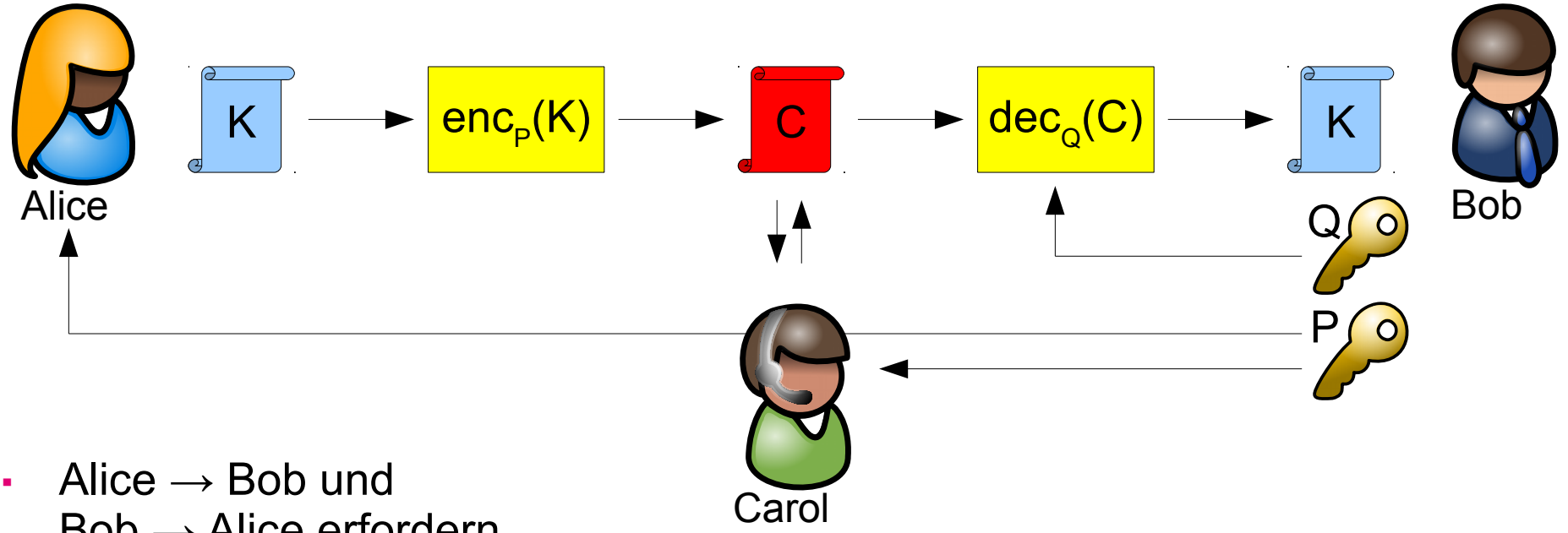
- Keine Garantie der Integrität (wie bei jedem Stromcipher)
  - Wenn ein Bit im Cipher-Strom verändert wird, verändert sich dasselbe Bit im Klartext
    (vgl. Output Feedback Mode, Coutnter Mode von AES)

- Eine Chosen-Plaintext-Attacke enthüllt den Schlüssel
  - Genau genommen, den Schlüsselstrom aus der S-Box
  - Nie, nie niemals nicht denselben Schlüssel mehrmals verwenden
    (in AES auch ein Problem, wenn Initialisierungsvektor oder Klartext einiger Blöcke erraten werden kann)

ASYMMETRISCHE VERSCHLÜSSELUNG

ASYMMETRISCHE VERSCHLÜSSELUNG

- auch: Public-Key-Verschlüsselung
- Öffentlicher Schlüssel P zur Verschlüsselung, Übertragung ungesichert
- Privater Schlüssel Q zur Entschlüsselung,
  wird vom Empfänger geheimgehalten



- Alice → Bob und
  Bob → Alice erfordern
  unterschliedliche Schlüssel

RSA-CHIFFRE

- 1977 von Ron Rivest, Adi Shamir, Leonard Adleman am MIT entwickelt
- Erstes veröffentlichtes Public-Key-Verschlüsselungsverfahren

- Idee: Multiplikation von zwei Primzahlen ist „billiger“ als Primfaktorzerlegung

- Public Key
  - Für die Verschlüsselung, kann veröffentlicht werden
  - Schlüssel: {RSA-Modul, Verschlüsselungskomponente}
    (Produkt zweier Primzahlen, Hilfswert)
- Private Key
  - Für die Entschlüsselung, muss geheim bleiben
  - Schlüssel: {RSA-Modul, Entschlüsselungsexponent}
    (Produkt zweier Primzahlen, anderer Hilfswert)

SCHLÜSSELERZEUGUNG FÜR RSA

Beispiel

- Wähle zwei große Primzahlen p,q
- Berechne den RSA-Modul
  $N = p*q$
- Berechne die Eulersche φ-Funktion des RSA-Moduls
  $\varphi(p,q) = (p-1) * (q-1)$
- Wähle zufällig eine zu φ(p,q) teilerfremde Zahl e mit
  $1 < e < \varphi(p,q)$
- Berechne den Entschlüsselungsexponenten d, so dass gilt
  $d * e \equiv 1 \text{ MOD } \varphi(p,q)$
- Die Parameter p,q und φ(p,q) können nun gelöscht werden

- Public Key: {N,e}
- Private Key: {N,d}

VERSCHLÜSSELUNG MIT RSA

Beispiel

Hilfsvariablen:

$N = p*q$

$\varphi(p,q) = (p-1)*(q-1)$

e = *zu* $\varphi(p,q)$ *teilerfremde Zahl*

$d * e \equiv 1 \text{ MOD } \varphi(p,q)$

- Public Key: {N,e}
- Private Key: {N,d}
- Verschüsselung: $C = K^e \text{ MOD } N$
- Entschlüssselung: $K = C^d \text{ MOD } N$

- <u>Beispiel: Verschlüssele den Wert 7</u>

festgelegt	berechnet		
p = 11, q = 13	N = 11*13	= 143	← *Public/Private Key*
	φ(p,q) = (11-1)*(13-1)	= 120	
Klartext	1< e < 120	e = 23	← *Public Key*
K = 7	d * 23 MOD 120 =1	d = 47	← *Private Key*

enc(7) = (7^23) MOD (11*13) = 2

ENTSCHLÜSSELUNG MIT RSA

Beispiel

Hilfsvariablen:

$N = p*q$

$\varphi(p,q) = (p-1)*(q-1)$

e = *zu* $\varphi(p,q)$ *teilerfremde Zahl*

$d * e \equiv 1 \text{ MOD } \varphi(p,q)$

- Public Key: {N,e}
- Private Key: {N,d}
- Verschüsselung: $C = K^e \text{ MOD } N$
- Entschlüssselung: $K = C^d \text{ MOD } N$

- <u>Beispiel: Entschlüssele den Ciphertext 2</u>

Private Key	Ciphertext
N = 143	C = 2
d = 47	

dec(2) = (2^47) MOD (143) = 7

DISKUSSION PUBLIC-KEY-VERFAHREN

- Asymmetrische Verschlüsselung ist langsam
  - Viele Modulo-Divisionen
  - Sehr große Primzahlen mit wenigstens 1024 Bit
    → in der Praxis mit symmetrischer Verschlüsselung kombiniert

- Zwei verschiedene Schlüssel für Ver- und Entschlüsselung
  - Öffentlichen Schlüssel wie authentifizieren / zurückziehen?
    → komplexe Lösungen zum Schlüsselmanagement erforderlich
  - Es sind immer Chosen-Plaintext-Angriffe möglich
    → Angreifer kennt öffentlichen Schlüssel, kann beliebig chiffrieren

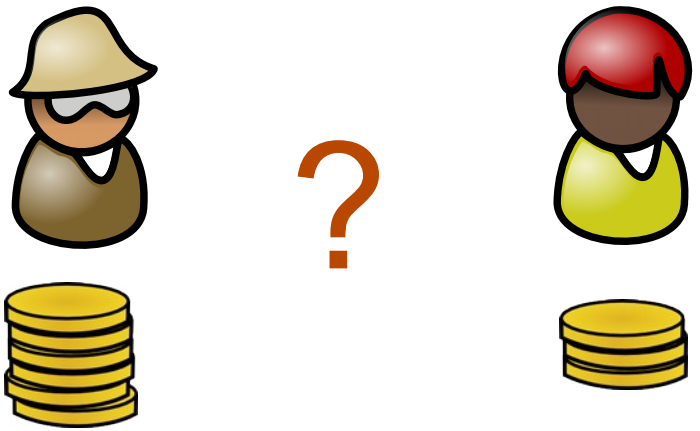
- Asymmetrische Verschlüsselung ist sicher
  - Aber: hängt davon ab, dass es keine Möglichkeit gibt, bestimmte Rechnungen effizient durchzuführen (RSA: Primzahlzerlegung)

SECURE MULTIPARTY COMPUTATION

PRIVATE BERECHNUNGEN AUF VERTEILTEN DATEN

- Verschlüsselung schön und gut, aber was, wenn man keine Daten weitergeben will?

- Beispiel: Yao's Millionärsproblem
  - Zwei Millionäre wollen wissen, wer von ihnen der reichere ist
  - Kein Millionär will sein Vermögen offenbaren



  - *Formal: löse $a \leq b$ ohne a und b zu offenbaren*

YAO'S MILLIONÄRSPROBLEM

- Bob nutzt den RSA-Algorithmus, um eine Reihe von Zufallszahlen zu verschlüsseln; eine davon ist Zufallszahl + echter Wert b
- Alice verändert alle Zahlen, die an einer Stelle in der Folge stehen, die größer ist als a, kodiert sie und sendet sie an Bob
- Kodierung erfolgt so, dass Bob nur seinen Zufallswert b wieder entschlüsseln kann, falls a ≥ b

RECHENBEISPIEL (1/2)

Beispiel

- Alice: **a = 5**, Public Key {3337, 79}, Private Key {3337, 1019}
- Bob: **b = 6**, Zufallszahl x = 1234
  - verschlüssele x mit Public Key von Alice
    also Cipher C = 1234^{79} MOD 3337 = 901
  - sende C – b + 1= 896 an Alice
- Alice:
  - Erzeuge Zahlenreihe im fraglichen Wertebereich, hier von 3<z<8
  - entschlüssele y = 896 + z, d.h., rechne $(896+z)^{1019}$ MOD 3337
    z=4: dec(896+4) = 2918
    z=5: dec(896+5) = 358
    z=6: dec(896+6) = **1234** ← Zufallszahl von Bob, Alice weiß es nicht
    z=7: dec(896+7) = 296

Beispiel von [2]

RECHENBEISPIEL (2/2)

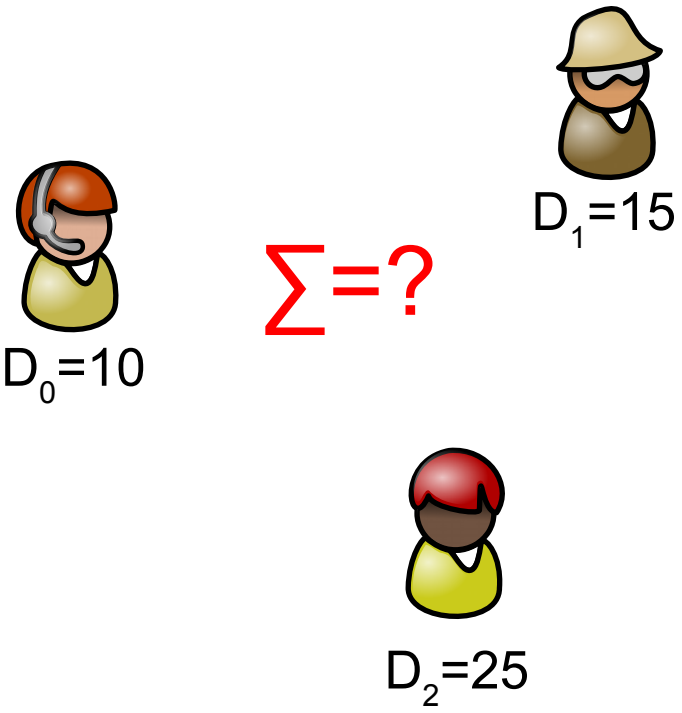
Beispiel

- Alice: Wähle eine Primzahl p = 107
  - Für z ≤ a: berechne y MOD p
    Für z > a: berechne (y MOD p) + 1
    z=4: 2918 MOD 107 = 29
    z=5: 358 MOD 107 = 64
    z=6: **1234 MOD 107 + 1 = 58**
    z=7: 296 MOD 107 + 1 = 83
  - Sende diese Zahlenreihe und Primzahl p an Bob
- Bob:
  - Zufallszahl x = 1234 MOD Primzahl p = 107: Ergebnis = 57
  - Steht in der Zahlenreihe von Alice für z = 6 die 57?
    JA: a ≥ b
    NEIN: a < b ← Bob ist reicher
  - *Bob erfährt nur, ob a < b, aber nicht um wieviel kleiner!*

SECURE SUM

- Erweiterung von Yao's Millionärsproblem:
  Berechne die Summe aus k verteilt vorliegenden Werten, ohne die Summanden D_p zu enthüllen

  Formal: berechne $\sum_{p=0...(k-1)} (D_p)$ ohne D_p zu offenbaren



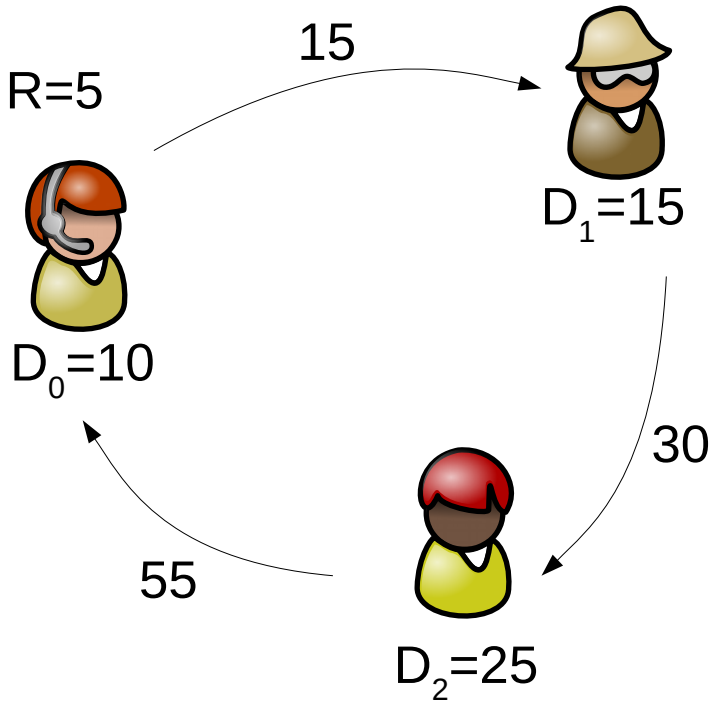
EINFACHSTER ANSATZ (CLIFTON ET AL.)

- Algorithmus
  - P_0 erzeugt Zufallszahl R, schickt S = R + D_0 an P_1
  - P_1 schickt S = S + D_1 an P_2

    …
  - P_0 erhält S

    Ergebnis = S – R

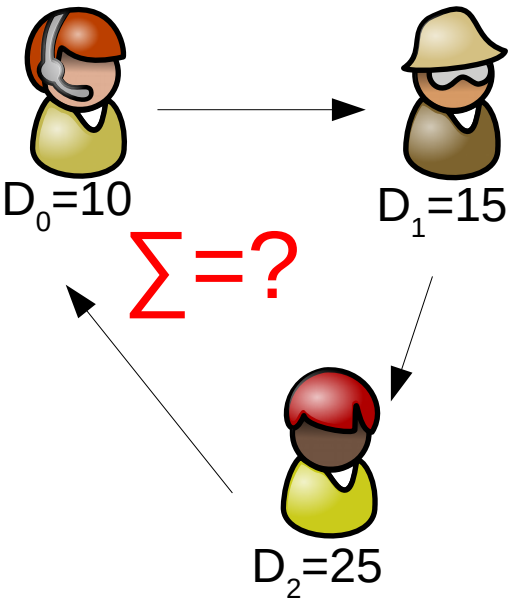
- Problem
  - Wenn P_{l-1} und P_{l+1} zusammenarbeiten, erfahren sie den Wert von P_l



DISTRIBUTED K-SECURE SUM [1]

- k Nutzer: P_0, P_1, … P_k mit geheimen Daten D_0, D_1, … D_k
- Jeder Nutzer p teilt D_p in soviele Segmente $D_{p,q}$ auf wie Nutzer, Summe der Blöcke bleibt erhalten $D_p = \sum_{q=0...(k-1)} D_{p,q}$
- Algorithmus:

```
for p = 0 to k-1
      P_p send k-1 segments randomly to other users
      mix D_p,*
end
S=0
for j = 0 to k-1          //Zeile
    for i = 0 to k-1      //Spalte
        P_i send S = S + D_i,j to user (i+1) MOD k
    end
end
P_0 send S to all other users
```



Anm.: Ubertragungen erfolgen verschlüsselt

RECHENBEISPIEL

Beispiel

- k = 3 Nutzer

1. Runde:
P_0 sendet 0+ 2 an P_1
P_1 sendet 2+10 an P_2
P_2 sendet 12+ 3 an P_0
2. Runde:
P_0 sendet 15+ 1 an P_1
P_1 sendet 16+ 8 an P_2
P_2 sendet 24+ 3 an P_0
3. Runde:
P_0 sendet 27+12 an P_1
P_1 sendet 39+ 5 an P_2
P_2 sendet 44+ 6 an P_0

Ergebnis: S=50


D_0=10


D_1=15


D_2=25

$D_{*,0}$ =	5	8	12
$D_{*,1}$ =	2	6	3
$D_{*,2}$ =	3	1	10

gemischt

$D_{*,0}$ =	2	10	3
$D_{*,1}$ =	1	8	3
$D_{*,2}$ =	12	5	6

SICHERHEIT DES VERFAHRENS

- Fall 1: alle Teilnehmer arbeiten korrekt
  - Protokoll ist sicher
- Fall 2: der Protokoll-Initiator arbeitet unkorrekt
  - Ergebnis wird falsch, aber Privatheit der Daten gewahrt
- Fall 3: zwei Teilnehmer kooperieren
  - Teilnehmer tauschen sich aus, welche Daten sie erhalten haben
  - Simples Protokoll: die Daten eines Teilnehmers werden offenbar
  - Dk-Secure Sum: Da ein Teilnehmer jeweils Datensegmente aller anderen Teilnehmer hat, bleibt die Privatheit der Daten gewahrt
- Fall 4: alle Teilnehmer außer einem kooperieren
  - Daten des einen Teilnehmers werden offenbar

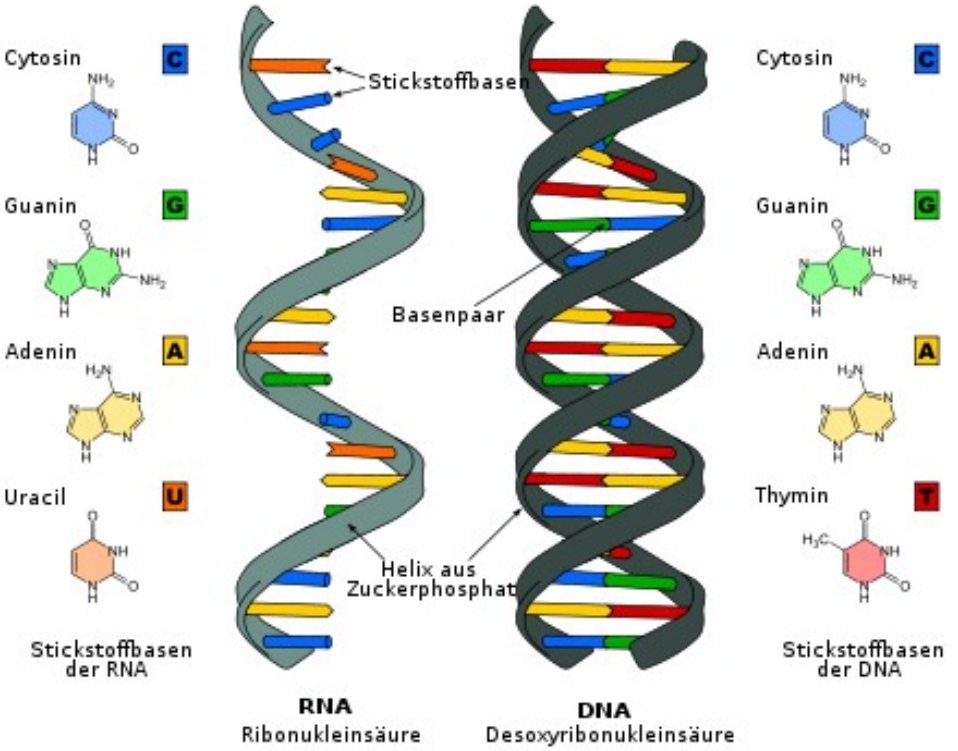
HERAUSFORDERUNGEN FÜR DIE ZUKUNFT

DNA COMPUTING (1/2)

- DNA computing: Ein unglaublicher Parallelrechner

- Schritt 1:
  - Transformation von Daten in RNA-Abschnitte
  - Transformation eines Problems so, dass es sich durch RNA -Rekombination lösen lässt

- Schritt 2:
  - RNA in Reagenzglas bringen
  - Chemisch vervielfachen



DNA COMPUTING (2/2)

- Schritt 3:
  - Zufällige RNA-Abschnitte hinzugeben
  - Reagenzglas schütteln
- Schritt 4:
  - Reagenzglasinhalt auslesen (Sequenzierungsautomat)
- Riesige Datenmengen parallel verarbeiten
  - Alle Daten aller Datacenter weltweit im Kofferraum eines PKW)
    - Haken an der Sache: diese Daten *schnell* in RNA kodieren
  - Immernoch *klassische* Informatik, d.h., keine Überraschungen
    - Richtigen AES-Schlüssel superschnell durch ausprobieren finden (nachdem Sie den vorher monatelang als RNA kodiert haben)

QUANTENKRYPTOGRAPHIE (1/2)

- Seltsame Regeln der Quantenphysik
  - Quantenverschränkung: räumlich getrennte Teilchen tauschen Informationen über ihre Eigenschaften *ohne Zeitverlust* aus
  - Superposition: Teilchen befinden sich *bis zur Messung* des Zustands in *allen* Zuständen *gleichzeitig*

- Daraus kann man Computer mit neuen Eigenschaften bauen
  - Shor‘s Algorithmus faktorisiert in polynomialer Zeit
    → Alle Public-Key-Verfahren sind auf einen Schlag unsicher

QUANTENKRYPTOGRAPHIE (2/2)

- Haken an der Sache: Verschalten von Quantenbits per Supraleiter ist unglaublich störanfällig
  - Größter Quantencomputer weltweit (Stand Februar 2020): Googles Sycamore mit 53 Qubits (54 gebaut, aber eines ist kaputt)
  - RSA: Aktuell 2048 Bit Schlüssellänge sicher, 4096 Bit werden unterstützt

- Post-Quantenkryptographie ist in der Entwicklung
  - Beispiel: XMMS-Signaturen „eXtended Merkle Signature Scheme“
    - Basierend auf Hash-Trees
    - Sicherheit basiert auf Kollisionsresistenz
    - Bereits im Standardisierungsprozess der IETF (RFC 8391)

ABSCHLUSS

ZUSAMMENFASSUNG

- Etablierte Verschlüsselungsverfahren können Daten effektiv gegen ausspähen schützen
  - Aber: keine absolute Sicherheit!
  - „esoterische“ Verfahren wie Quantencomputing, DNA-Computing
- Durch Datenschutz neue Ziele für die Kryptographie
  - Plausible Abstreitbarkeit
  - Berechnungen ohne Zusammenführung von Eingabedaten
- In dieser Vorlesung
  - Symmetrische und asymmetrische Verschlüsselung
  - Probabilistische Verschlüsselung und Secure Multiparty Computation als Mittel gegen Frequenz- und Zuordnungsangriffe

MÖGLICHE PRÜFUNGSFRAGEN

- Warum ist es prinzipiell nicht möglich, einfache Verfahren wie Schiebe-Chiffren oder Ersetzungs-Chiffren sicher zu machen? Also warum benötigt man die Komplexität von Verfahren wie AES?

- Erklären Sie, was alles gegeben sein muss, damit RSA und AES sicher sind. Also welche Technologien, Ressourcen und welches Hintergrundwissen dürfen Angreifer NICHT haben?

- Nennen Sie ein Einsatzszenario, in dem Sie den Cipher Feedback Mode von AES verwenden können, nicht aber den Cipher Block Chaining Mode.

- Was ist der Unterschied zwischen „echter“ Stromverschlüsselung mit RC4 und AES im Output Feedback Mode, das ja sehr ähnlich aussieht?

LITERATUR

[1] Rashid Sheikh et al.: *A Distributed k-Secure Sum Protocol for Secure Multi-Party Computations.* In: Journal of Computing, 2(3), 2010

[2] Rechenbeispiel zu Yao's Millionärsproblem
https://eprint.iacr.org/2017/1129.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8089&rep=rep1&type=pdf

[3] Johannes Buchmann: *Einführung in die Kryptographie*. Springer Verlag, 2003