Karlsruher Institut für Technologie

Datenschutz und Privatheit in vernetzten Informationssystemen

Kapitel 4: Kryptographische Verfahren

Erik Buchmann (buchmann@kit.edu)

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“

Inhalte und Lernziele dieses Kapitels

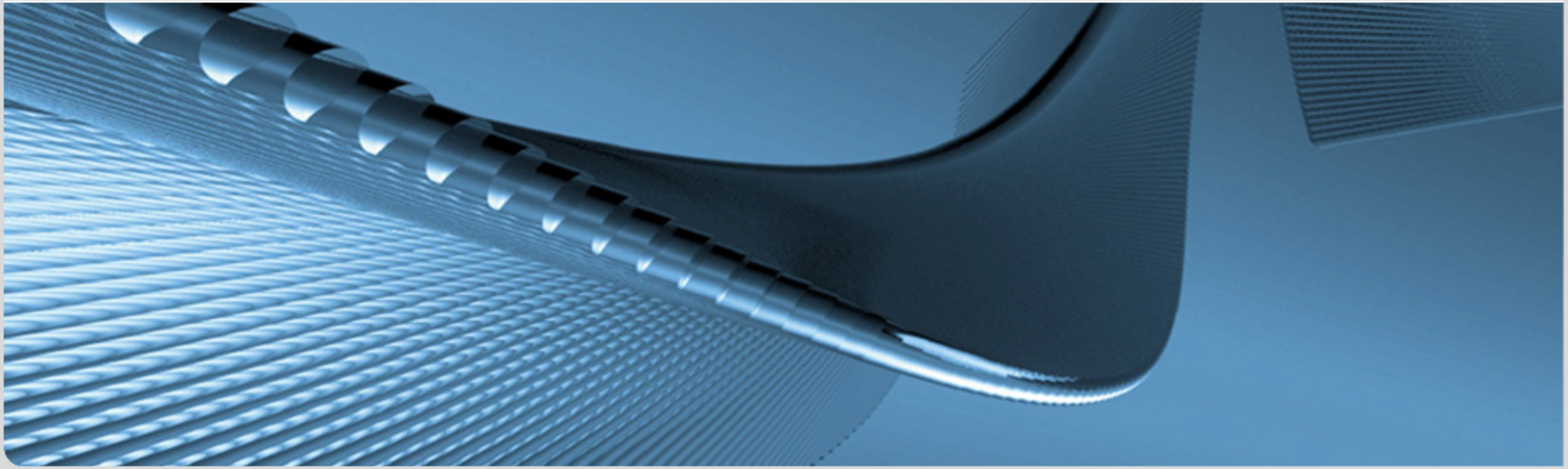
- Inhalt
  - Einführung in die Kryptographie
  - Symmetrische Verschlüsselung
  - Asymmetrische Verschlüsselung
  - Probabilistische Verschlüsselung
  - Secure Multiparty Computation

- Lernziele
  - Sie können die verschiedenen Verschlüsselungsverfahren anwenden, und kennen ihre Stärken und Schwächen.
  - Sie können die Ziele der Kryptographie aus Datenschutzsicht erklären, sowie passende Verschlüsselungsverfahren und zugehörige Sicherheitsbeweise erläutern.
  - Sie können einen Überblick über das Konzept der Secure Multiparty Computation geben.

Karlsruher Institut für Technologie

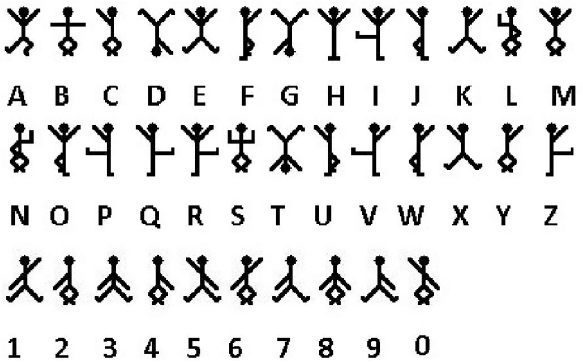
Einführung in die Kryptographie

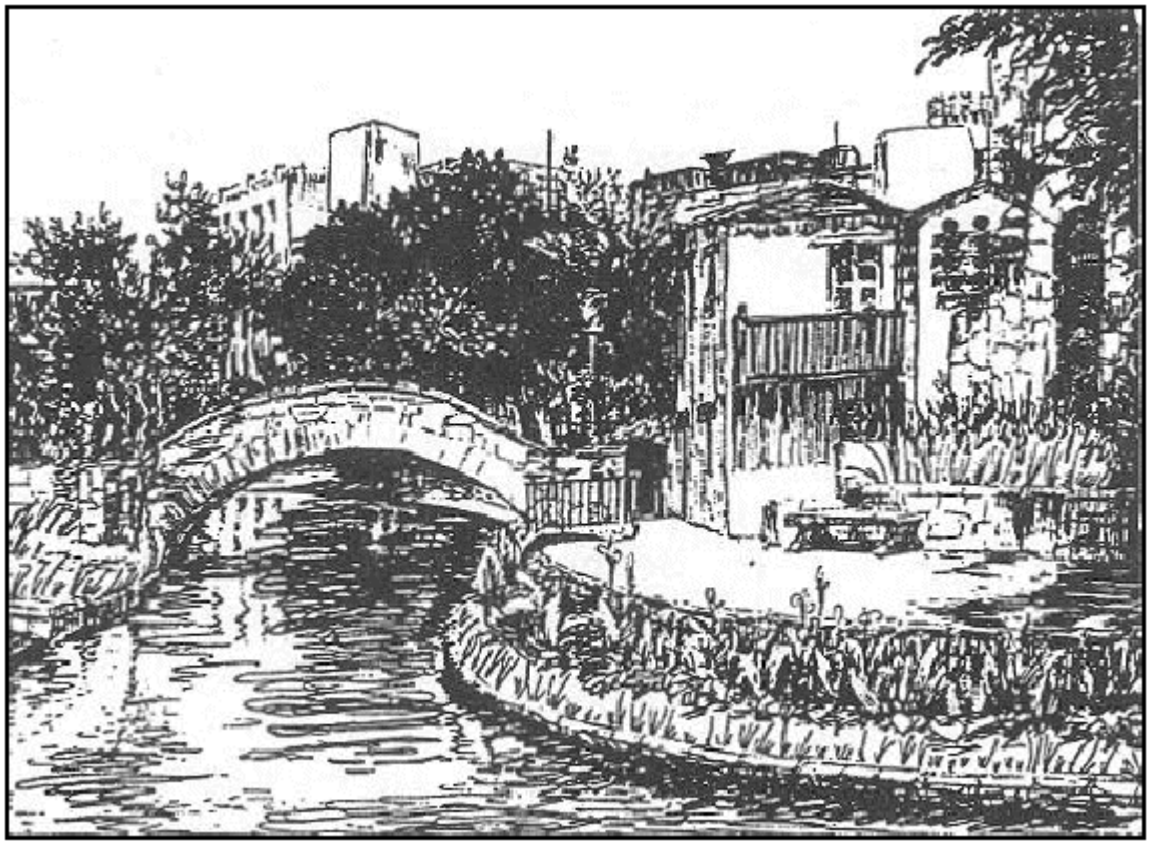
IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Wer etwas zu verbergen hat...

- ...versucht es zu verstecken.
  - Unsichtbare Tinte (Zitronensaft trocknet unsichtbar, bei erhitzen braun)
  - Codebücher
  - In Bildern (rechts: Morsecode in Grashalm-länge codiert)
  - Geheinschrift (unten: nach Arthur C. Doyle)





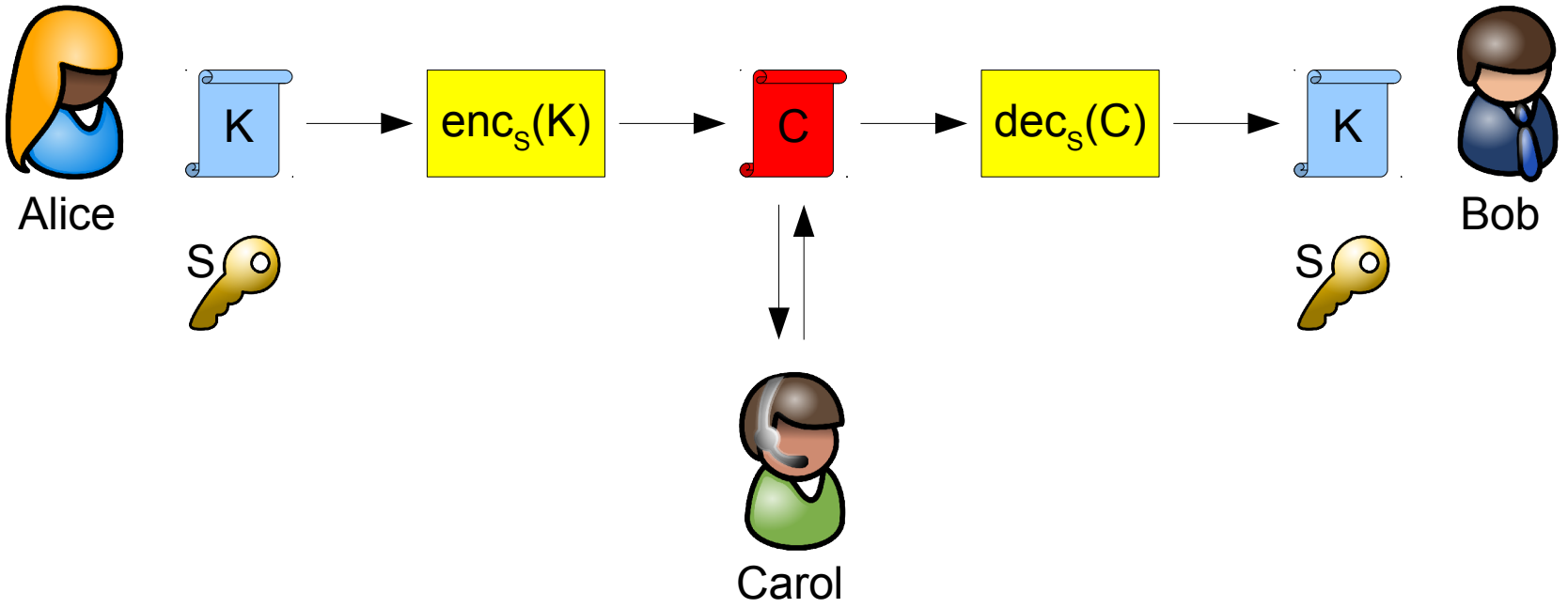
Ziele der Kryptographie

- **Vertraulichkeit**
  - Nur Berechtigte sollen Nachricht lesen,
  - Absender/Empfänger in Erfahrung bringen,
  - Existenz einer Nachricht erfahren
- **Integrität**
  - Daten nachweislich unverfälscht vom Sender zum Empfänger
- **Authentizität**
  - Urheber einer Nachricht eindeutig nachprüfbar
- **Verbindlichkeit**
  - Absender einer Nachricht soll Urheberschaft nicht abstreiten können

- Verfahren müssen nicht unbedingt jedes Ziel erfüllen

Prinzip der Verschlüsselung

- Klartext K, Ciphertext C,
- Schlüssel S
- Verschlüsselung: $C \rightarrow enc_S(K)$
- Entschlüsselung: $K \rightarrow dec_S(C)$



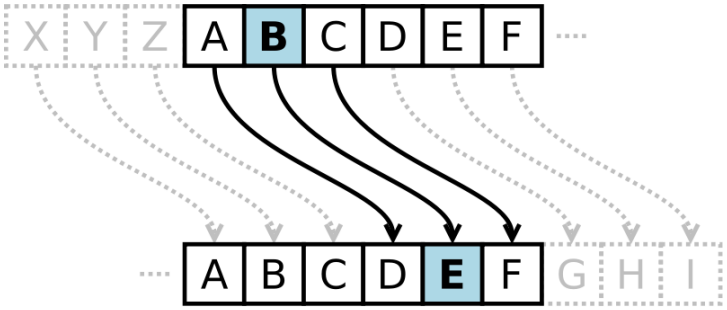
Anfänge der Verschlüsselung

- 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 so kompliziert wie möglich 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)

Aktuelle Verschlüsselungsverfahren

- Verfahren werden veröffentlicht
  - Claude Shannon: „The enemy knows the system“
  - Sicherheit der Verschlüsselung nur durch Geheimhaltung des Schlüssels nicht Geheimhaltung des Verfahrens!

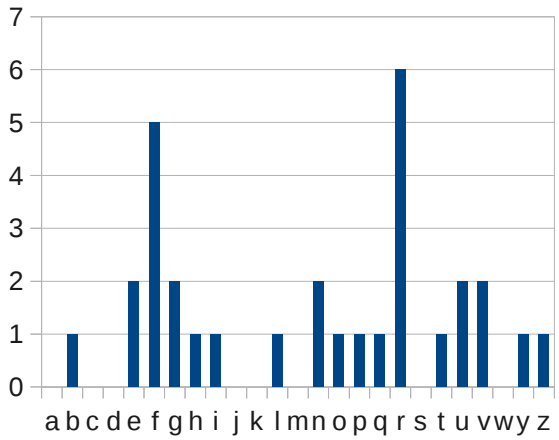
- Sicherheit wird unterschieden nach
  - Ressourcen des Angreifers
    - **polynomial beschränkter Angreifer**
  - Ziele des Angreifers
    - Berechnung des Schlüssels S
    - Mindestens ein Bit Klartext $k \in K$ aus Ciphertext C $\rightarrow$ $enc_S(K)$ extrahieren
    - Bestimmung von Meta-Informationen aus gegebenem Ciphertext C (Länge des Schlüssels, Länge des Textes, Sprache, Urheber etc.)
  - Angreifermodell
    - Ciphertext-Only, Known-Plaintext, Chosen-Plaintext, Chosen-Ciphertext

Angreifermodelle

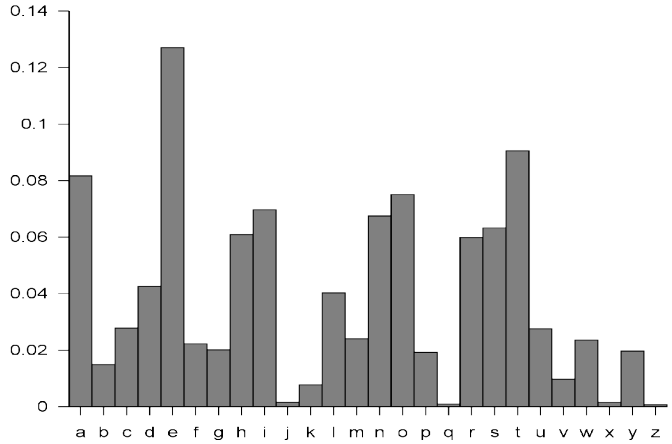
- Angreifermodelle unterscheiden nach Wissen des Angreifers
- **Ciphertext-Only**
  - Angreifer hat nur Ciphertext zur Verfügung
- **Known-Plaintext**
  - Angreifer kennt Ciphertext-Klartext-Paare
- **Chosen-Plaintext**
  - Angreifer kann aus Klartext selbst Ciphertext erzeugen
    (typisch für Public-Key-Verfahren,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

Beispiel: Brechen des Caesar-Chiffres

- Verschiebechiffre ROT13: X → X + 13 mod 26
- K: „`This should be a very secret message!`“
  C: „`Guvf fubhyq or n irel frperg zrffntr!`“
- Ciphertext-Only-Angriff:



Häufigkeitsverteilung der Buchstaben im Ciphertext



Häufigkeitsverteilung der Buchstaben in eng. Texten

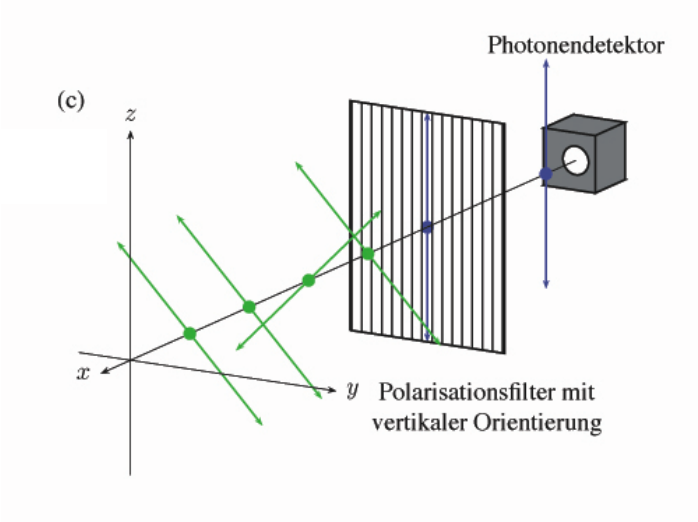
- „e“ ist häufigster Buchstabe, wurde „e“ zu „r“?
- Bei bekanntem Verfahren ist Verschüsselung gebrochen, „r“→„e“=13

Beweisbare Sicherheit bei Ciphertext-Only

- One-time Pad
  - zufälliger Schlüssel K
    - Genauso lang wie die Nachricht
    - Gleichverteilt zufällig gewählt
    - Muss geheim bleiben
    - Darf nur einmal verwendet werden
  - Verschlüsselung: $X_i \rightarrow X_i$ XOR K_i

```
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
```

- Beitrag der Quantenkryptographie
  - Laser sendet polarisierte Photonen für den Schlüsselaustausch beim One-time Pad
  - **Sicherer Kanal**: Messung eines Dritten zerstört Zustand der Photonen



- Problem: Unhandlich, ggf. extra Hardware

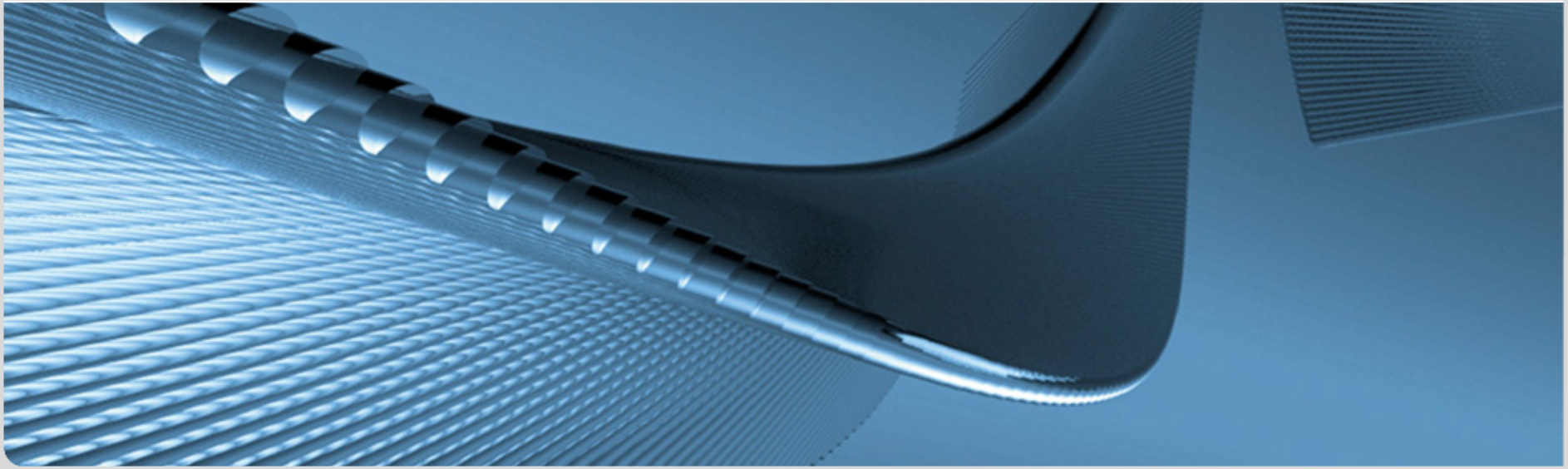
Kryptographie und Datenschutz

- 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 unter Nichtverknüpfbarkeit)
    - Abstrakteres Konzept 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

Karlsruher Institut für Technologie

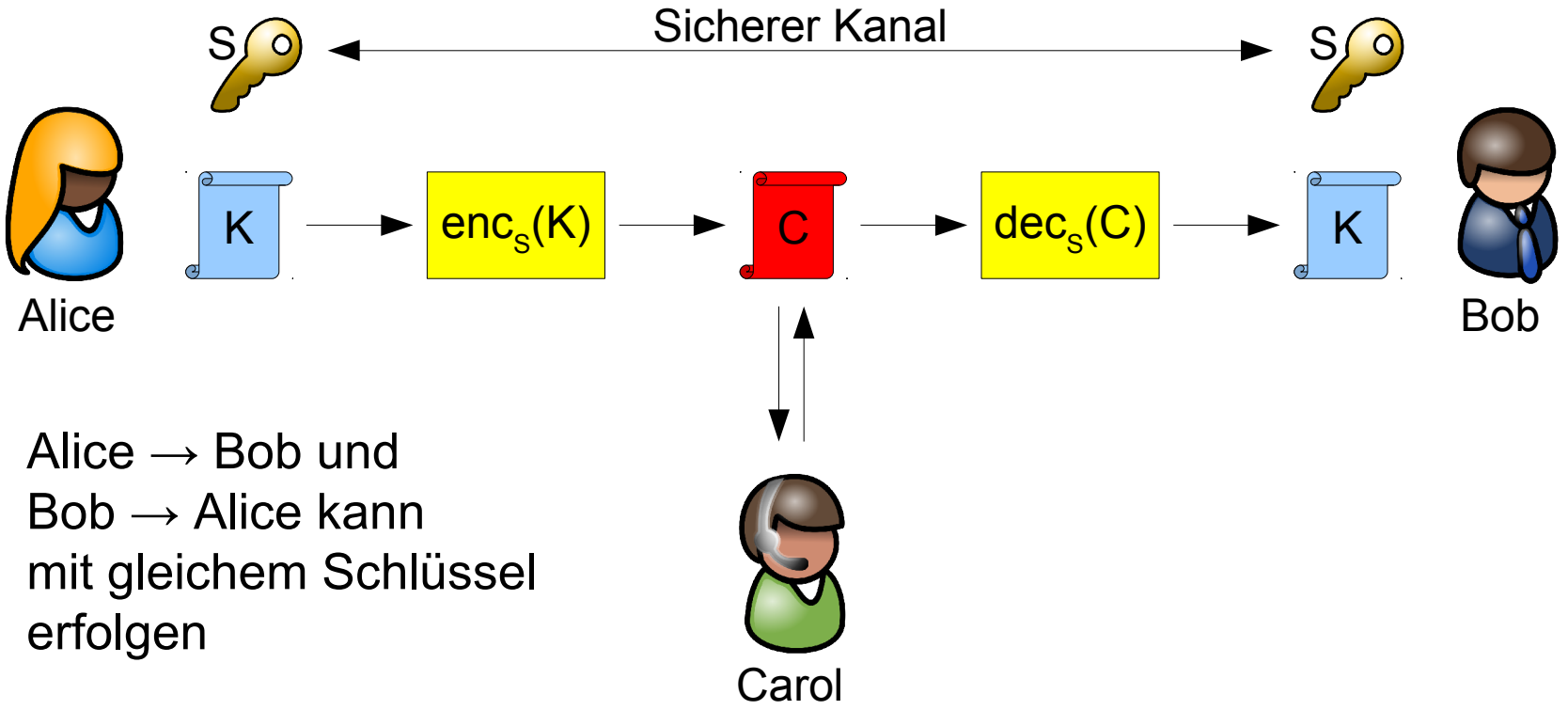
Symmetrische Verschlüsselung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Symmetrische Verschlüsselung

- Alice und Bob haben gleichen Schlüssel
- Ein Schlüssel zur Ver- und Entschlüsselung
- Schlüssel ist geheim, muss über gesicherten Kanal übertragen werden

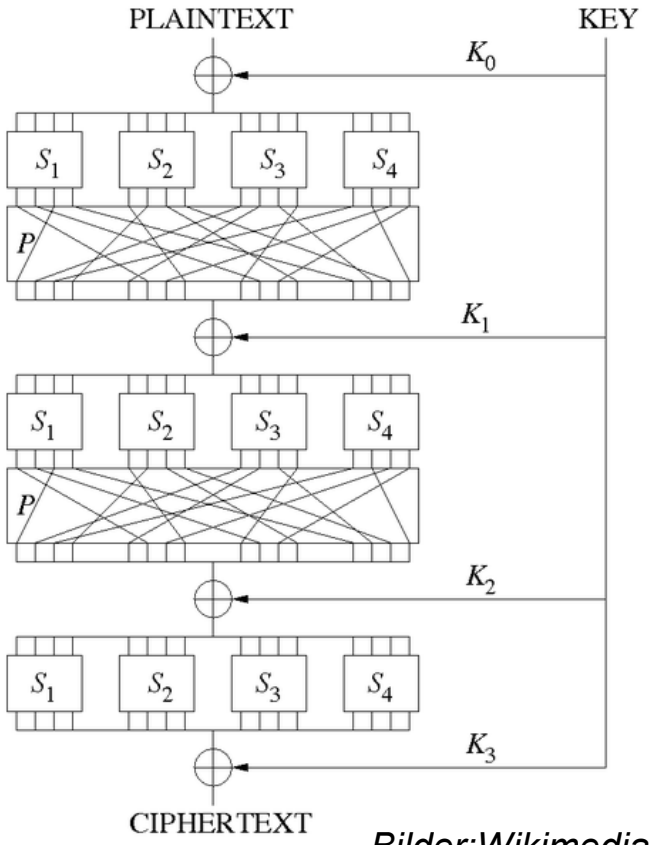


- Alice → Bob und Bob → Alice kann mit gleichem Schlüssel erfolgen

Blockchiffrierung

- Ziel: Klartexte beliebiger Länger mit kürzerem Schlüssel verschlüsseln

- Designprinzip:
  Substitutions-Permutations-Netzwerke
  - Aus Schlüssel mehrere Rundenschlüssel generieren
  - Mehrere gleiche Verschlüsselungsrunden
    - Rundenschlüssel auf Eingabe addieren
    - Ergebnis in Blöcke aufteilen
    - Substitutionsbox (S_1...S_4): 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

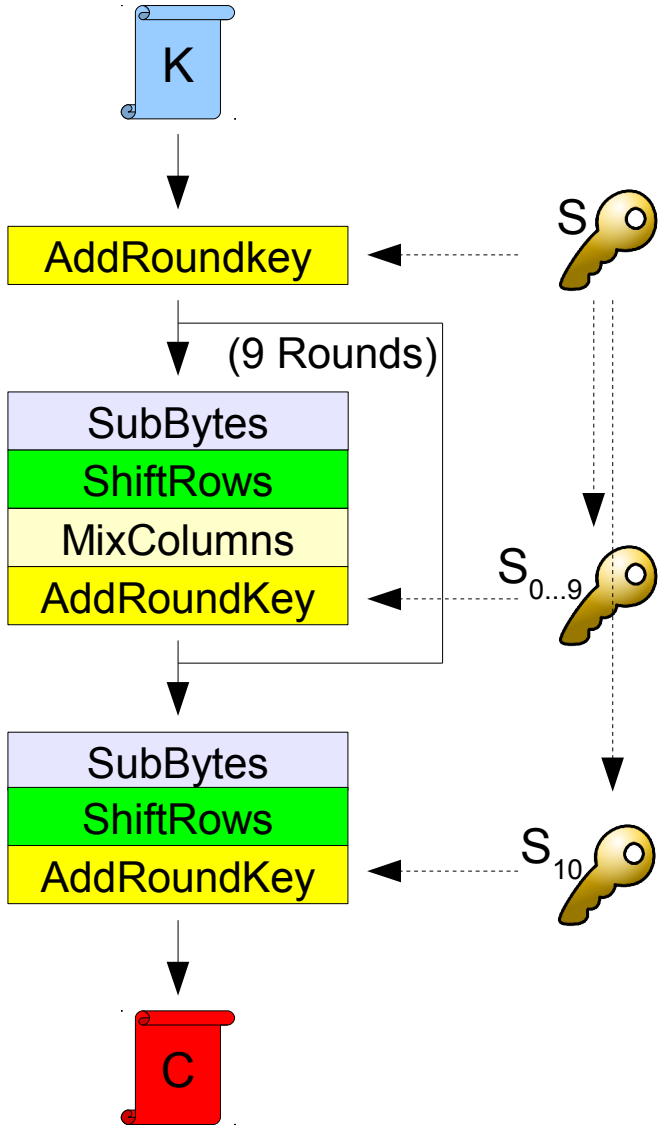


Bilder:Wikimedia

Advanced Encryption Standard (AES)

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

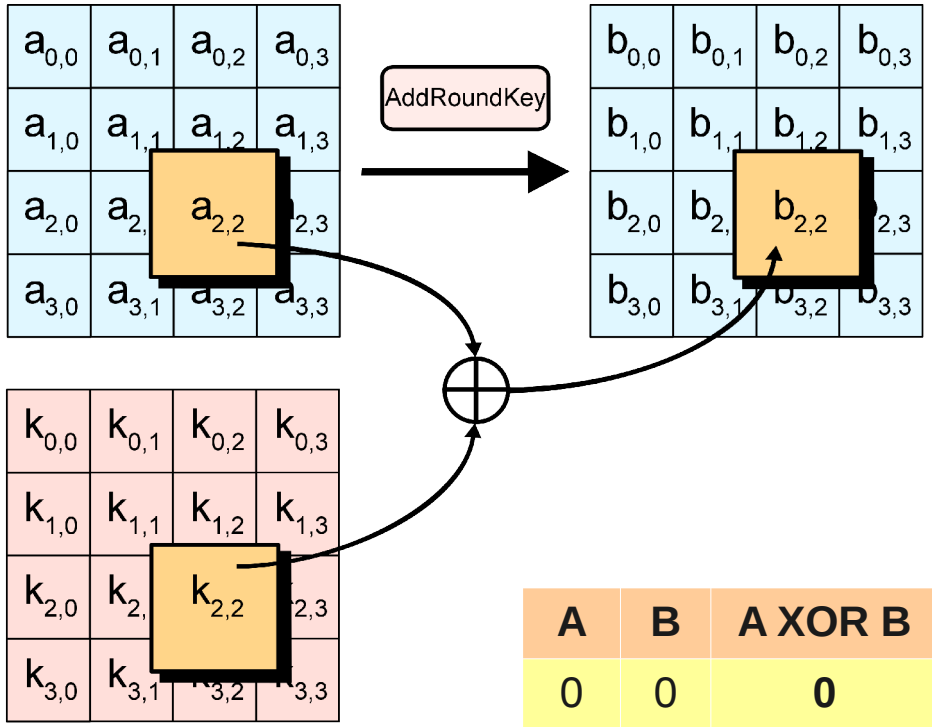
- AES ist ein Blockchiffre
  - Ein Block ist
    - Tabelle mit 4 Zeilen, 4 Spalten
    - 1 Byte pro Zelle
    - 128 Bit insgesamt
  - S-Boxen: SubBytes
  - P-Boxen: ShiftRows, MixColumns



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

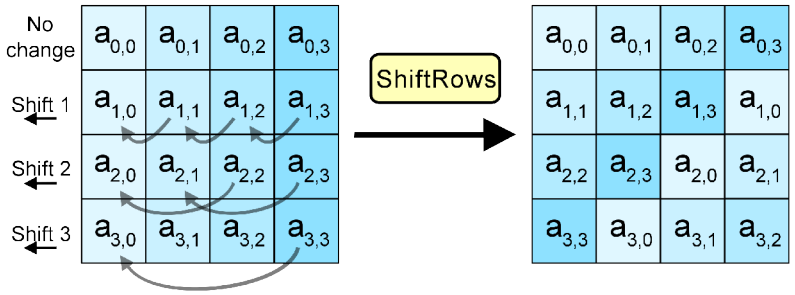
S-Boxen in AES

- 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

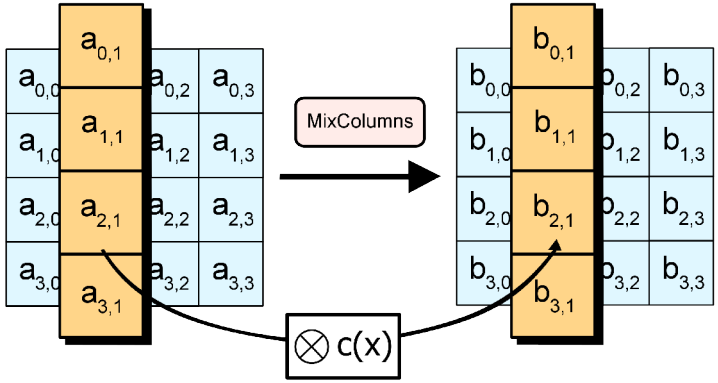
- Methode **ShiftRows**
  - Jede Zelle um die Nummer seiner Zeile nach Links schieben
  - Links herausfallende Zellen rechts einfügen



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

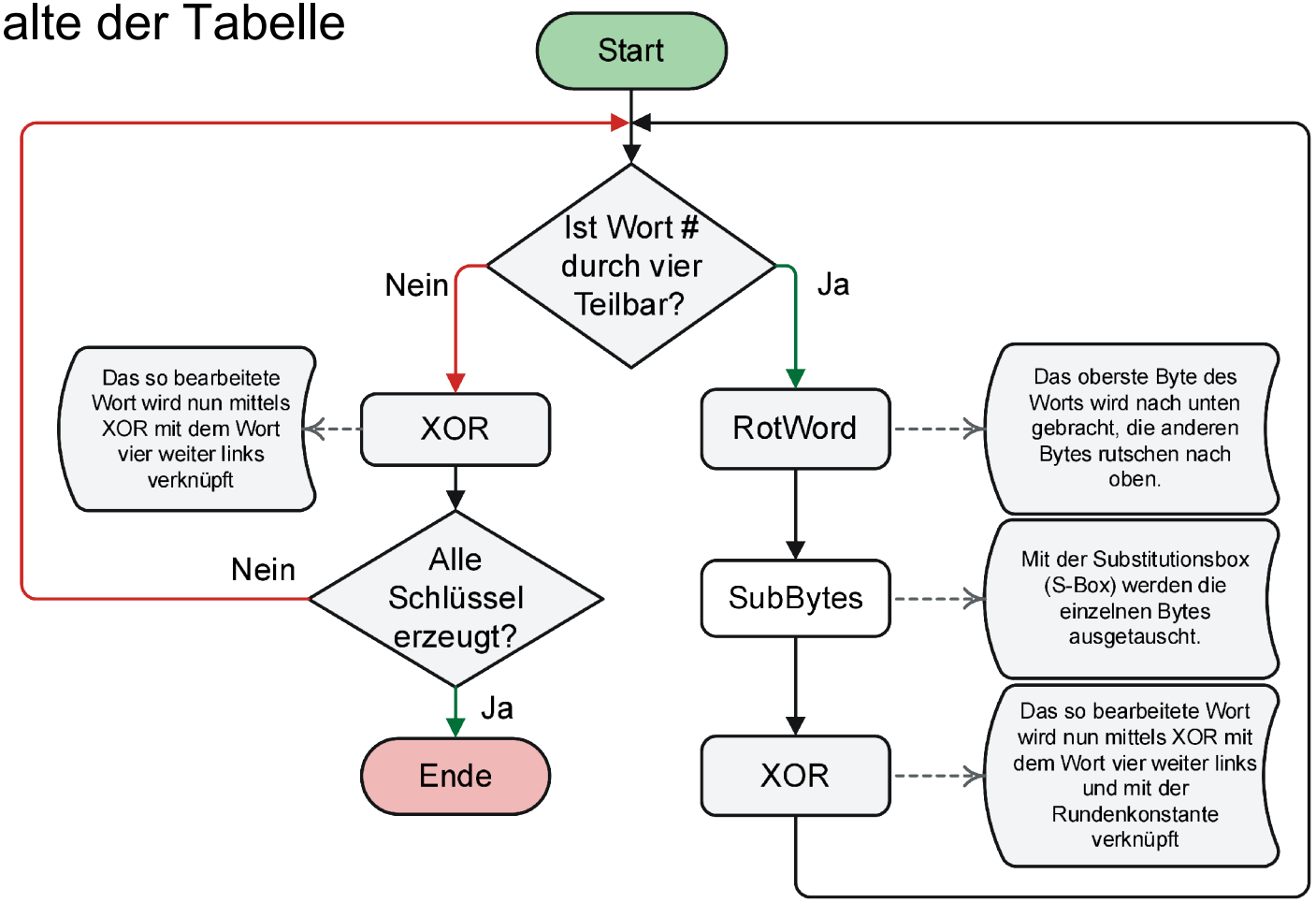


  - alle 4 Eingabebytes beeinflussen jedes Ausgabebyte

Bilder:Wikimedia

Rundenschlüssel in AES

- Schlüssel als 4x4-Tabelle, Wort: eine Spalte der Tabelle
- Rekursive Berechnung des Runden-schlüssels
- 10 Runden-schlüssel werden ge-braucht



Bilder:Wikimedia

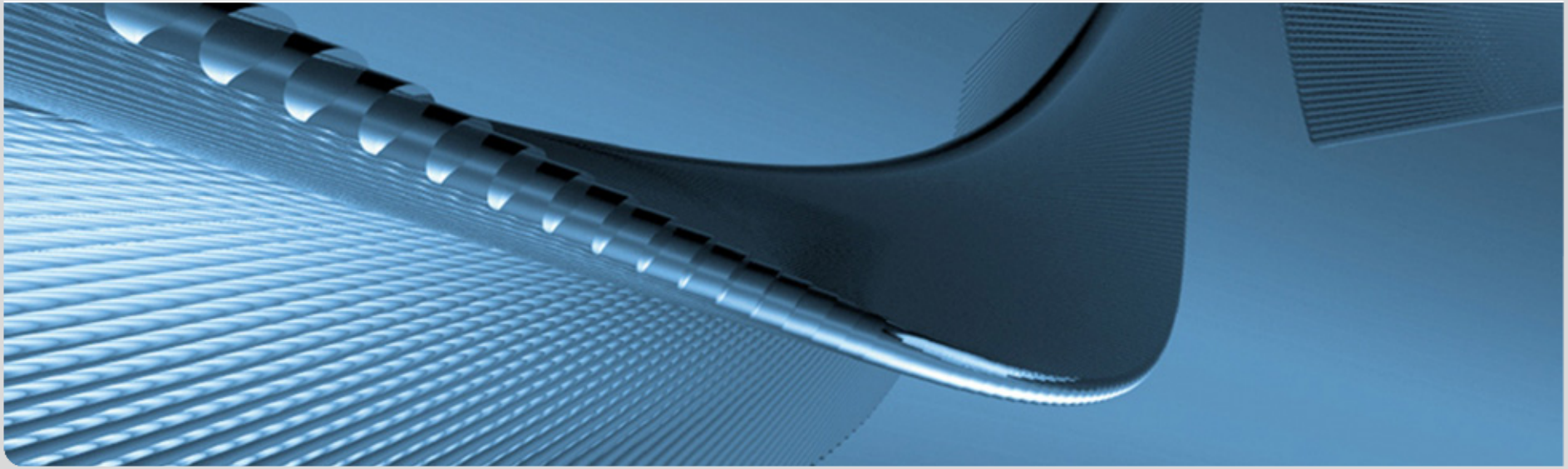
Diskussion AES

- AES ist performant
  - Nur „billige“ Operationen
  - 128 Bit Blockgröße passt sehr gut in die CPU-Register
  - Hardware-Implementierung möglich, z.B. auf FPGA
- AES ist sicher
  - 2011 erste Lösung, aber nur um Faktor 4 schneller als vollständige Suche → kein ernsthafter Angriff, vollst. Suche hat 2^{128} - 2^{256} Möglichkeiten
- Nachteile
  - Sicherer Kanal für die Schlüsselübertragung nötig

Karlsruher Institut für Technologie

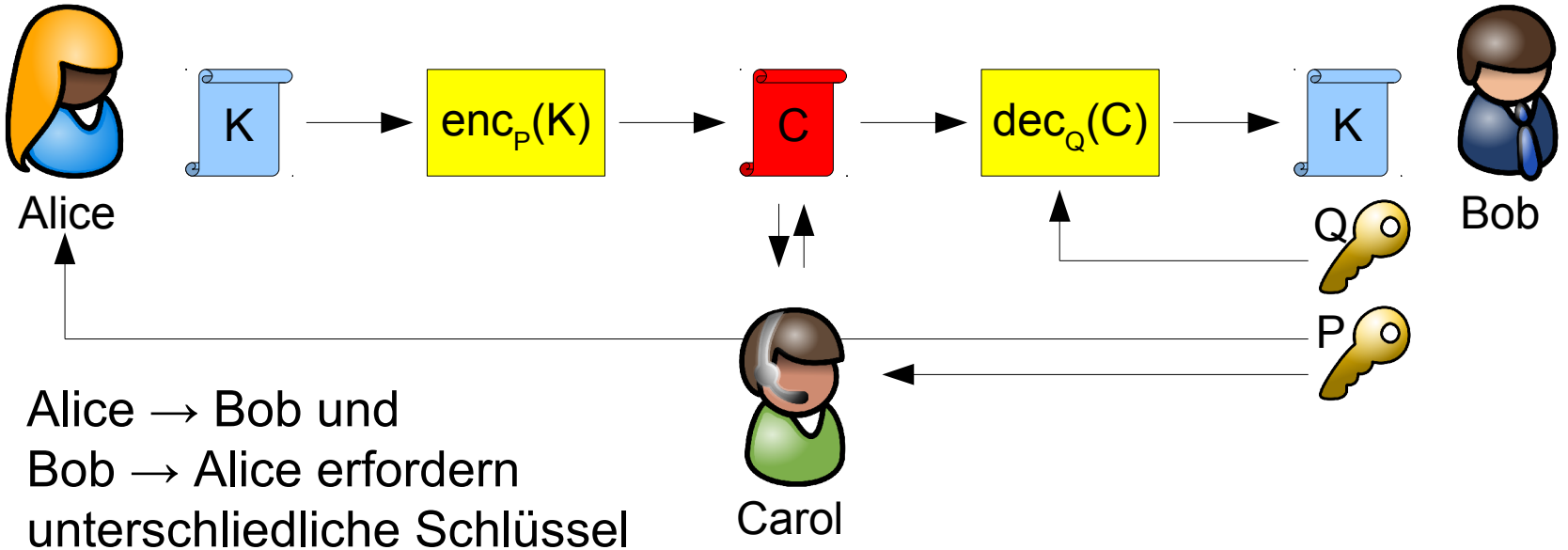
Asymmetrische Verschlüsselung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



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, wird geheim gehalten
  - Schlüssel: {RSA-Modul, Entschlüsselungsexponent} (Produkt zweier Primzahlen, anderer Hilfswert)

Schlüsselerzeugung für RSA

- 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}

Ver- und Entschlüsselung mit RSA

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

- Verschlüsseln: $C = K^e \text{ MOD } N$
- Entschlüsseln: $K = C^d \text{ MOD } N$

- Rechenbeispiel
  - Verschlüssele 7
    gewählt: p = 11, q = 13
    berechnet: e = 23, d = 47
    enc(7) = (7^23) MOD (11*13) = 2
  - Entschlüssele 2
    dec(2) = (2^47) MOD (11*13) = 7

Hilfsvariablen:

$$N = p*q$$
$$\varphi(p,q) = (p-1)*(q-1)$$
$$e = \text{zu } \varphi(p,q) \text{ teilerfremde Zahl}$$
$$d * e \equiv 1 \text{ MOD } \varphi(p,q)$$

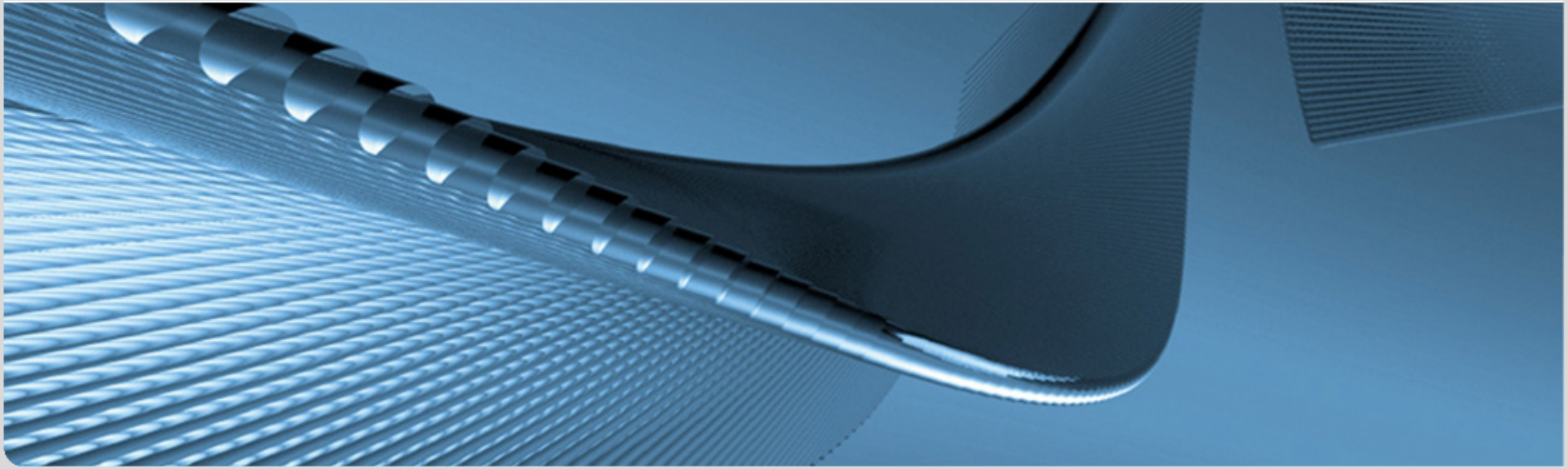
Diskussion Public-Key-Verfahren

- Asymmetrische Verschlüsselung ist langsam
  - Viele Modulo-Divisionen
  - Sehr große Primzahlen mit wenigstens 1024 Bit
    → in der Praxis immer mit symmetrischer Verschlüsselung kombiniert
- Zwei verschiedene Schlüssel für Ver- und Entschlüsselung
  - Öffentlichen Schlüssel wie authentifizieren/publizieren/zurückziehen?
    → komplexe Lösungen zum Schlüsselmanagement erforderlich
  - Es sind immer Chosen-Plaintext-Angriffe möglich
    → Angreifer kennt den öffentlichen Schlüssel, kann beliebig chiffrieren
- Sicherheit hängt davon ab, dass es keine Möglichkeit zur schnellen Primzahlzerlegung gibt
  - DNA-Computing?
  - Quantencomputing?

Karlsruher Institut für Technologie

Probabilistische Verschlüsselung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Zuordnungsangriffe

- Bisher vorgestellte Verschlüsselungsverfahren sind deterministisch
  - Bei jeder Verschlüsselung:
    Identischer Klartext → identischer Ciphertext
  - Public-Key-Verfahren erlauben **Chosen-Plaintext**-Angriffe,
    d.h., man kann ausprobieren, ob man den Klartext erraten kann

- Beispiel: Befindet sich „Alice“ in einer verschlüsselten DB?
  - Angreifer kennt Public Key:
    „Alice“ → „H71A00EF91“

Person	Krankheit	Alter
AA73F9E001	Magersucht	24
H71A00EF91	Diabetes	31
3429A9F64E	Schnupfen	26

Frequenzangriffe

- Bisher vorgestellte Verschlüsselungsverfahren sind deterministisch
  - Bei jeder Verschlüsselung:
    Identischer Klartext → identischer Ciphertext
  - Aus der Verteilung von identischen Ciphertexten kann auf Inhalte geschlossen werden

- Beispiel: Hat „Alice“ in verschlüsselter Datenbank Urlaub genommen?
  - Angreifer weiß, dass am Jahresanfang „30 Tage“ häufigster Eintrag

Person	Urlaubstage
Alice	**A7**
Bob	EF
Carol	**A7**
Dave	**A7**
Eve	21
Frank	FF

Probabilistische Verschlüsselung

- Bei jeder Verschlüsselung desselben Klartexts anderer Ciphertext
- Strategie: Zufall einbauen
  - z.B.: tausche die deterministische Trapdoor-Funktion in RSA gegen eine probabilistsche Trapdoor-Funktion
    (Trapdoor-Funktion: lässt sich nur in einer Richtung effizient berechnen)
- Beispiel: Goldwasser-Micali-Kryptosystem [4]
  - Public-Key-Verschlüsselung

Goldwasser-Micali-Kryptosystem

- Schlüsselerzeugung
  - Wähle zwei Primzahlen p, q so dass (p, q) ≡ 3 MOD 4
  - Berechne n = p * q und x = n – 1
  - Public Key: {x, n} Private Key: {p, q}
- Ein Bit m verschlüsseln
  - Wähle Zufallszahl z mit 1 < z < n
  - $C = x^m * z^2 \text{ MOD } n$
- Ein Bit entschlüsseln
  - Wenn $C^{(p-1)/2} \equiv 1 \text{ MOD } p$ und $C^{(q-1)/2} \equiv 1 \text{ MOD } q$, dann m = 1, sonst 0

Sicherheitsbeweis: IND-CPA

- Indistinguishability under Chosen-Plaintext Attack (IND-CPA)
  - Nachweis, dass Angreifer nicht zwischen der Verschlüsselung zweier unterschiedlicher Klartexte gleicher Länge unterscheiden kann
  - Sicherheit 'klassischer' Verschlüsselungsverfahren gegen Ausspähen
- Input
  - Eine Verschlüsselungsfunktion C = enc(K)
  - Zwei vom Angreifer beliebige gewählte Klartexte K_1, K_2 gleicher Länge
- Angriff
  - Für eine große Zahl von Runden
  - Berechne $C_1 = enc(K_1)$, $C_2 = enc(K_2)$
  - Wähle gleichverteiltes Zufallsbit M, sende C_M an Angreifer
  - Angriff ist erfolgreich, wenn Angreifer mit Wahrscheinlichkeit > 50% + ε das Bit M bestimmen kann
    (ε ist „Sicherheitsmarge“ für polynomiale Beschränktheit des Angreifers)

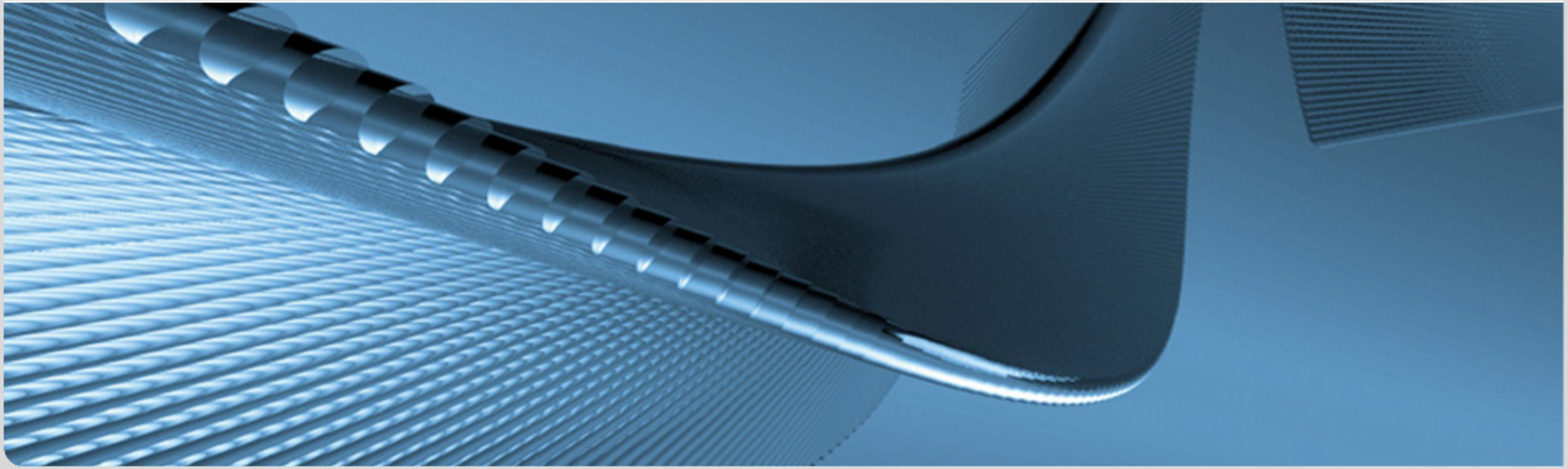
Sicherheitsbeweis: IND-ICP

- Indistinguishability under Independent Column Permutation (IND-ICP)
  - Nachweis, dass Frequenz- und Zuordnungsangriffe unmöglich sind
- Input
  - Datenbank DB
  - Datenbankfunktion f: DB → DB verschlüsselt Datenbank
  - Datenbankfunktion p: DB → DB permutiert alle Spalten in der Datenbank unabhängig voneinander
- Angriff
  - Für eine große Zahl von Runden
  - Zwei Datenbanken $DB_1 = f(DB)$ und $DB_2 = f(p(DB))$
  - Angreifer erhält zufällig mit Gleichverteilung (DB_1, DB_2) oder (DB_2, DB_1)
  - Angriff ist erfolgreich, wenn Angreifer mit Wahrscheinlichkeit > 50% + ε sagen kann, ob die erste DB die originale oder permutierte DB ist
    (ε ist „Sicherheitsmarge“ für polynomiale Beschränktheit des Angreifers)

Karlsruher Institut für Technologie

Secure Multiparty Computation

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



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 nutz 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)

- Alice: a = 5, Public Key {3337, 79}, Private Key {3337, 1019}
- Bob: b = 6, Zufallszahl x = 1234
  - verschlüssele x mit PubKey von Alice
    also Cipher C = 1234^{79} MOD 3337 = 901
  - sende C – b + 1= 896 an Alice
- Alice:
  - Erzeuge Zahlen z im fraglichen Wertebereich, hier von 4 bis 7
  - 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)

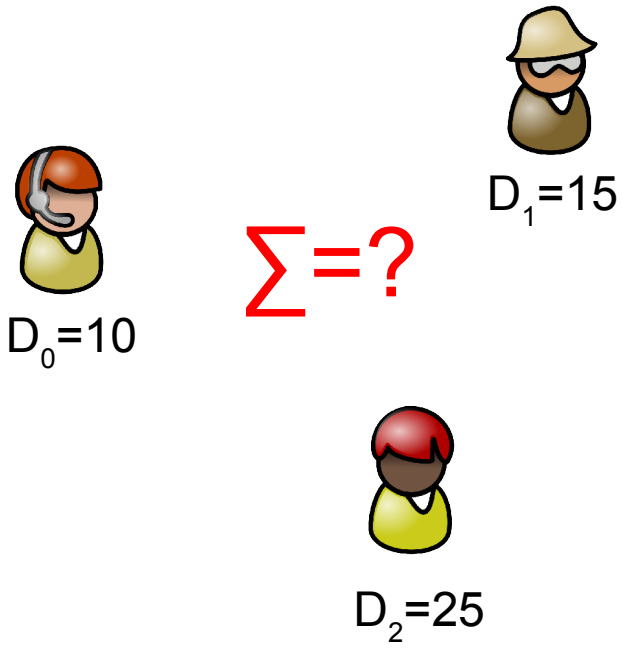
- Alice: Wähle Primzahl p = 107
  - Für z < a: berechne y MOD p
    Für z ≥ a: berechne (y MOD p) + 1
    z=4: 2918 MOD 107 = 29
    z=5: 358 MOD 107 = 64
    z=6: **1234 MOD 107 + 1 = 58**
    z=7: 296 MOD 107 + 1 = 83
  - Sende diese Zahlenreihe und Primzahl p an Bob

- Bob:
  - Berechne Zufallszahl x = 1234 MOD Primzahl p = 107, Ergebnis = 57
  - Schaue nach, ob in der Zahlenreihe von Alice für z = b die 57 steht
    JA: a ≥ b
    NEIN: a < b ← Bob ist reicher
  - *Bob erfährt nur, ob a < b, kan aber nicht sagen, 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



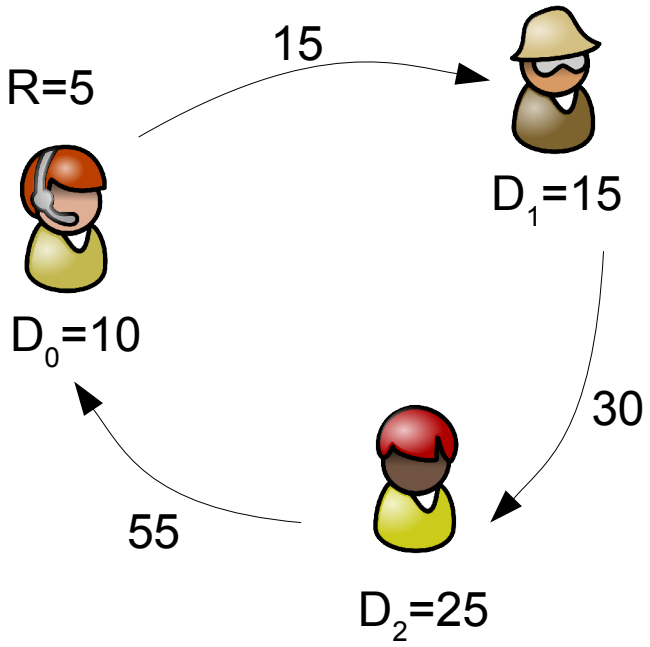
Einfachster Ansatz (Clifton et al.)

- 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

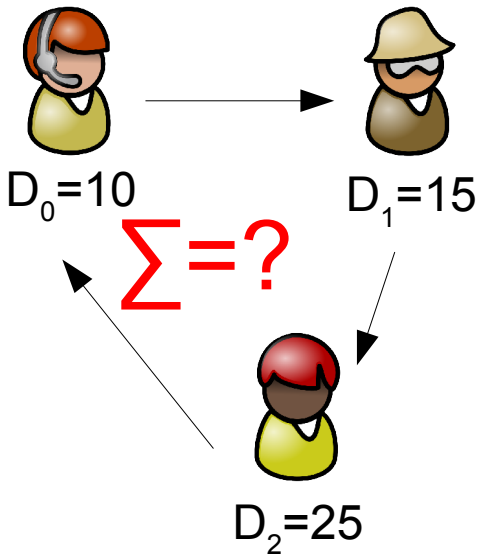
- Problem
  - Wenn P_{l-1} und P_{l+1} zusammenarbeiten, erfahren sie den Wert von P_l



Distributed k-Secure Sum [1]

- k Nutzer: P_1, P_2, … 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 sendet k-1 Datenblöcke zufällig an alle anderen
    mische D_p,*
end
S=0
for j = 0 to k-1
    for i = 0 to k-1
        P_i sendet S = S + D_i,j an Nutzer (i+1) MOD k
    end
end
P_0 sendet S an alle anderen
```



Anm.: Übertragungen erfolgen verschlüsselt

Rechenbeispiel

- 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

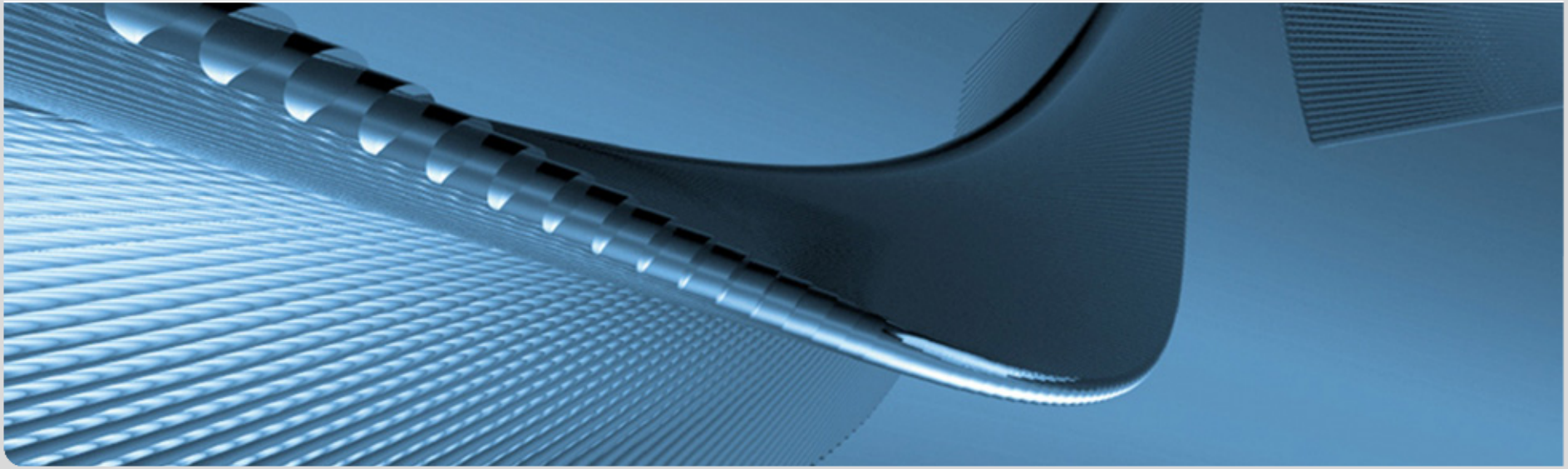
$D_{*,0}$ =	2	10	3
$D_{*,1}$ =	1	8	3
$D_{*,2}$ =	12	5	6

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 darüber 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

Karlsruher Institut für Technologie

Abschluss



Zusammenfassung

- Etablierte Verschlüsselungsverfahren können Daten effektiv gegen ausspähen schützen
- Durch Datenschutz neue Ziele für die Kryptographie
  - Plausible Abstreitbarkeit
  - Berechnungen ohne Zusammenführung von Eingabedaten
  - Passende Sicherheitsbeweise
- In dieser Vorlesung
  - Symmetrische und asymmetrische Verschlüsselung
  - Probabilistische Verschlüsselung und Secure Multiparty Computation als Mittel gegen Frequenz- und Zuordnungsangriffe

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
http://www.proproco.co.uk/million.html

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

[4] S. Goldwasser, S. Micali: *Probabilistic encryption*. Journal of Computer and System Sciences, 28(2), 1984