

Systeme II

7. Sicherheit

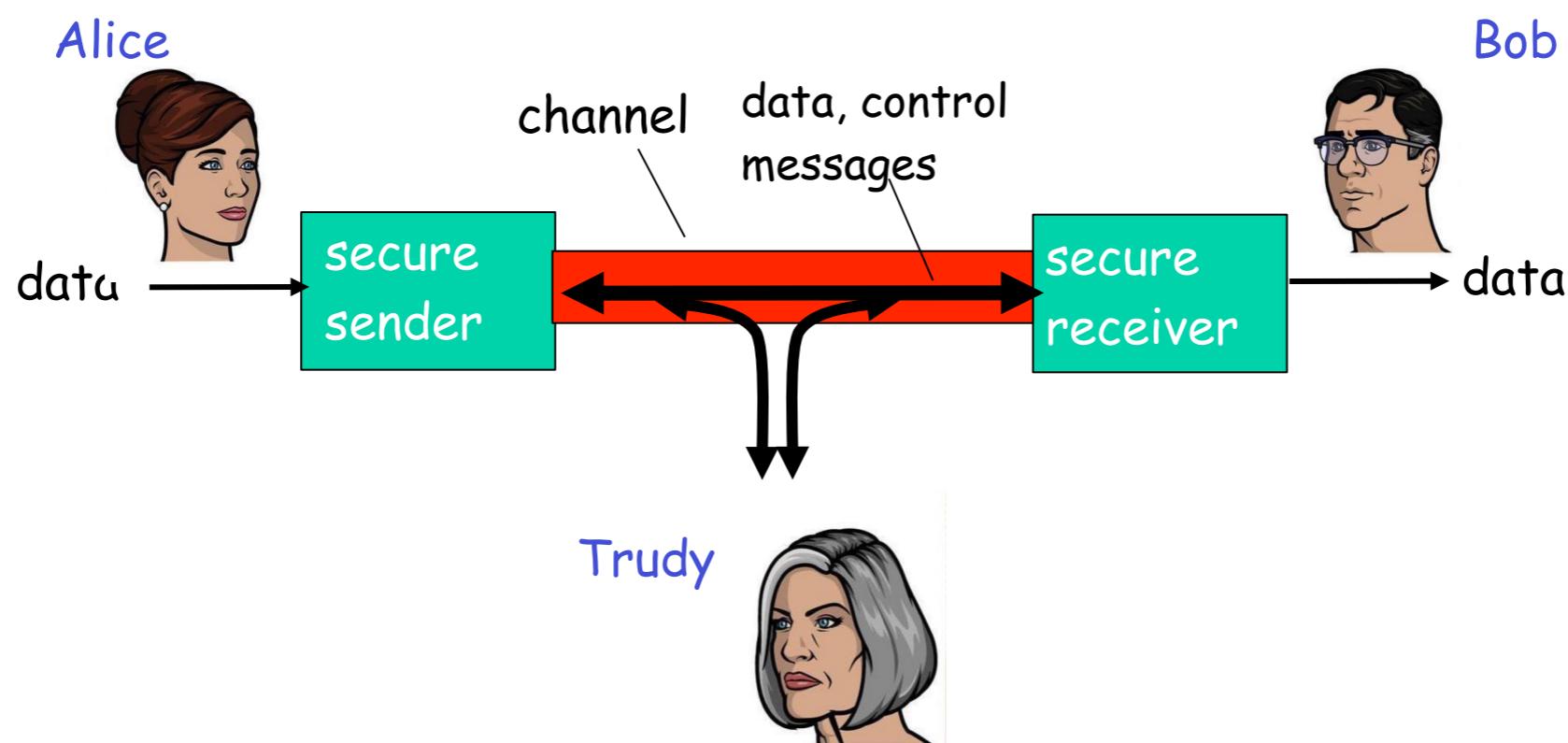
Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
(Version 19.07.2017)

Was ist Netzwerk-Sicherheit

- Vertraulichkeit (Confidentiality)
 - Nur der Sender, gewünschter Empfänger sollte den Nachrichteninhalt „verstehen“
- Authentifizierung
 - Sender und Empfänger möchten sich ihrer Identität versichern
- Integrität (message integrity)
 - Sender und Empfänger wollen, dass eine Nachricht nicht unbemerkt verändert werden
 - bei der Übertragung oder später
- Zugriff und Verfügbarkeit
 - von Diensten

Freunde und Feinde: Alice, Bob und Trudy

- Standardnamen im Sicherheitsbereich
- Alice und Bob möchten „sicher“ kommunizieren
- Trude (In-Trude-r) möchte mithören, löschen, hinzufügen, verändern



Wer steckt hinter Alice und Bob

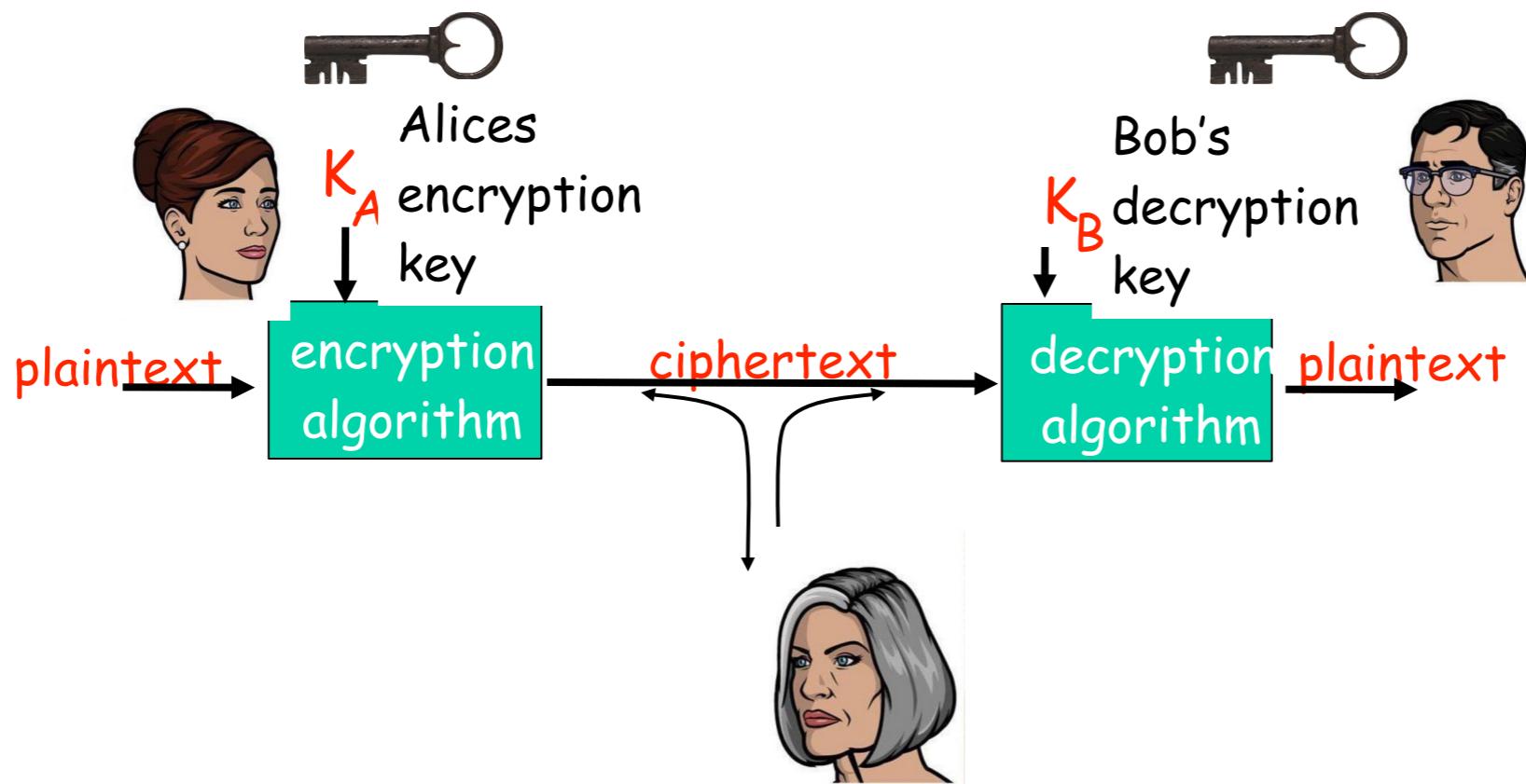
- Echte Menschen
- Web-Browser
- Online-Banking-Clients und Servers
- DNS-Servers
- Routers, die Routing-Tabellen austauschen
- etc.

Was kann ein böser Mensch so tun?

- Abhören (eavesdrop)
 - Nachrichten abfangen und lesen
- Einfügen von Nachrichten
 - Nachrichten werden in die bestehende Verbindung eingefügt
- Sich als jemand anders ausgeben (impersonation)
 - Quell-Adresse kann in einem Paket gefälscht werden
- Hijacking
 - Übernahme einer bestehenden Verbindung durch Ersetzen des Empfängers oder Senders
- Denial of Service
 - Dienst abschalten
 - durch Überlast oder direkten Angriff

Ein kurzer Rundgang durch die Kryptographie

- m : Originalnachricht (message)
- $K_A(m)$: mit Schlüssel K_A verschlüsselte Nachricht
- $m = K_B(K_A(m))$



Einfache Verschlüsselung

- Monoalphabetischer Schlüssel
 - ersetze jeden Buchstaben durch einen anderen
- Beispiel: Edgar Allen Poe „The Gold Bug“
 - 53305))6*;4826)4)4;806*;488¶60))85;1-(:*8-83(88)5*
 - ;46(;88*96*?;8)*(;485);5*2:*(;4956*2(5*-4)8¶8*;40692
 - 85);)68)4;1(9;48081;8:81;4885;4)485528806*81(9;48;
 - (88;4(?34;48)4;161::188;?;
- Jedes Symbol steht für einen Buchstaben:
 - 8 = e
 - ; = h
 - ...

Einfache Verschlüsselung

- Monoalphabetischer Schlüssel
 - ersetze jeden Buchstaben durch einen anderen

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq



E.g.: Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc

- n monoalphabetische Schlüssel, M_1, M_2, \dots, M_n
- Zyklus-Muster
 - e.g., $n=4$, $M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2;$
- Für jeden neuen Buchstaben aus den monoalphabetischen Schlüsseln einer ausgewählt
 - „aus“: a from M_1 , u from M_3 , s from M_4
 - Schlüssel: n Schlüsselverfahren und der Zyklus

Bruch einer Kodierung

■ Cipher-text only Attack

- nur mit verschlüsselten Text
- Zwei Ansätze:
 - Durchsuche alle Schlüssel und teste ob sie einen vernünftigen Text produzieren
 - Statistische Analyse des Schlüssels

■ Known-Plaintext-Attack

- mit der Originalnachricht und dem verschlüsselten Text

■ Chosen Plaintext Attack

- Trudy wählt den Text und lässt Alice ihn verschlüsseln
- Trudy erhält den verschlüsselten Text

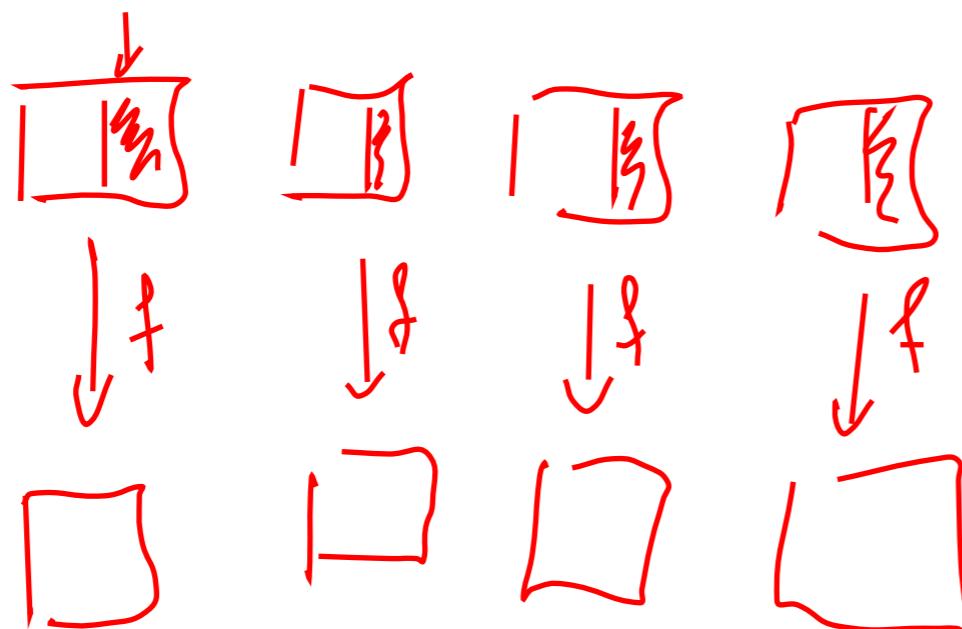
Methoden der Kryptographie

- Geheime Schlüssel sind die Sicherheitsgrundlage
 - Der Algorithmus ist bekannt
 - außer bei „security by obscurity“
- Public-Key-Cryptography
 - verwendet zwei Schlüssel
 - ein geheimer und ein öffentlicher Schlüssel
- Symmetrische Kryptographie
 - beide Seiten verwenden den selben geheimen Schlüssel
- Hash-Funktion
 - Ohne Schlüssel und ohne Geheimnis

Chiffrierungstypen

01010100 → 01100100

- Stromchiffrierer (stream cipher)
 - verschlüsselt bitweise
- Blockchiffre, Blockverschlüsselung (block ciphers)
 - Originaltext wird in gleichgroße Blöcke unterteilt
 - Jeder Block wird einzeln kodiert



Stream Ciphers

- Kombiniere jedes Bit eines Schlüsselstroms (key stream) mit dem Original bit

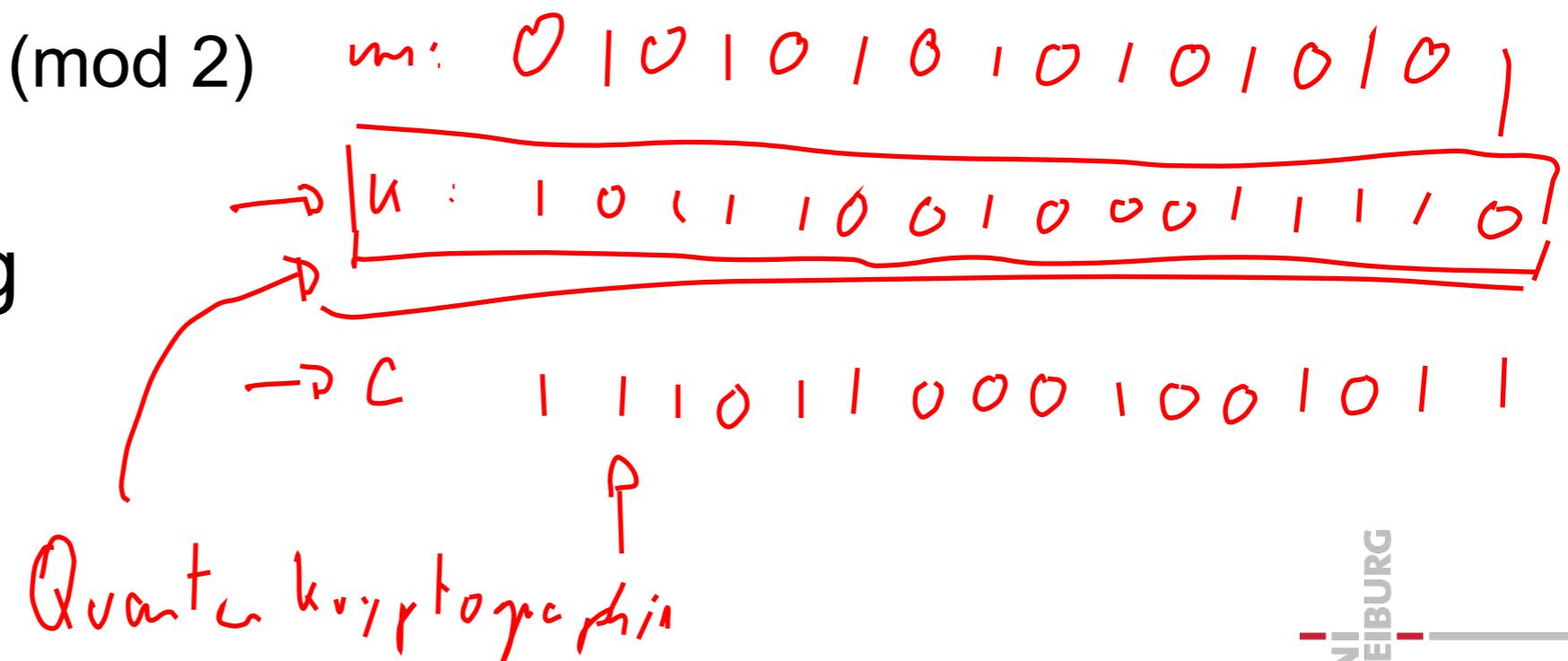
- $m(i)$ = i-tes Bit der Nachricht
- $ks(i)$ = i-tes Bit des Key Streams
- $c(i)$ = i-tes bit des verschlüsselten Texts

- Verschlüsselung

- $c(i) = ks(i) + m(i) \pmod{2}$
 $= ks(i) \oplus m(i)$

- Entschlüsselung

- $m(i) = ks(i) \oplus c(i)$



- RC4 ist ein populärer Streamchiffrierer
 - ausführlich analysiert und als sicher angesehen
 - Schlüssellänge: von 1 bis 256 Bytes
 - wird in WEP für 802.11 verwendet
 - kann in SSL verwendet werden

Quell-Code RC4

$$g(f(x)) = x$$

→ `k[]: gegebene Schlüssel-Zeichenfolge der Länge 5 bis 256 Byte`
`L := Länge des Schlüssels in Byte`

→ `s[]: Byte-Vektor der Länge 256`
`Für i = 0 bis 255`
`s[i] := i`
`j := 0`
`Für i = 0 bis 255`
`j := (j + s[i] + k[i mod L]) mod 256`
`vertausche s[i] mit s[j]`

→ `klar[]: gegebene Klartext-Zeichenfolge der Länge X`
`schl[]: Vektor zum Abspeichern des Schlüsseltextes`

`i := 0`
`j := 0`
`Für n = 0 bis X-1`
`i := (i + 1) mod 256`
`j := (j + s[i]) mod 256`
`vertausche s[i] mit s[j]`
~~`zufallszahl := s[(s[i] + s[j]) mod 256]`~~
~~`[schl[n] := zufallszahl XOR klar[n]`~~

■ Aus Wikipedia

- <http://de.wikipedia.org/wiki/Rc4>

Block-Chiffre

- Nachrichten werden in Blöcken von k bits verschlüsselt
 - z.B. 64-bit Blöcke
- Injektive Abbildung um den Quelltext in den k-bit verschlüsselten Text umzuwandeln
- Beispiel k=3:

<u>input</u>	<u>output</u>	<u>input</u>	<u>output</u>
f(000 = 110		100	011
001	111	101	010
010	101	110	000
011	100	111	001

Block-Chiffre

$(2^{64})!$

- Wie viele mögliche Abbildungen gibt es für k-Bit Block-Chiffre?

$$\overbrace{000}^k \rightarrow \overbrace{101}^k$$

$$\vdots$$

$$(2^k)!$$

$$V1$$

$$2(2^k)$$

$$f: A \rightarrow A$$

$$|A| = n$$

$$2^n \leq n! \leq 2^n$$

$$2^{n \cdot \log n + O(n)}$$

$$f(x_1) = n \text{ Mögl.}$$

$$f(x_2) = (n-1) \text{ Mögl.}$$

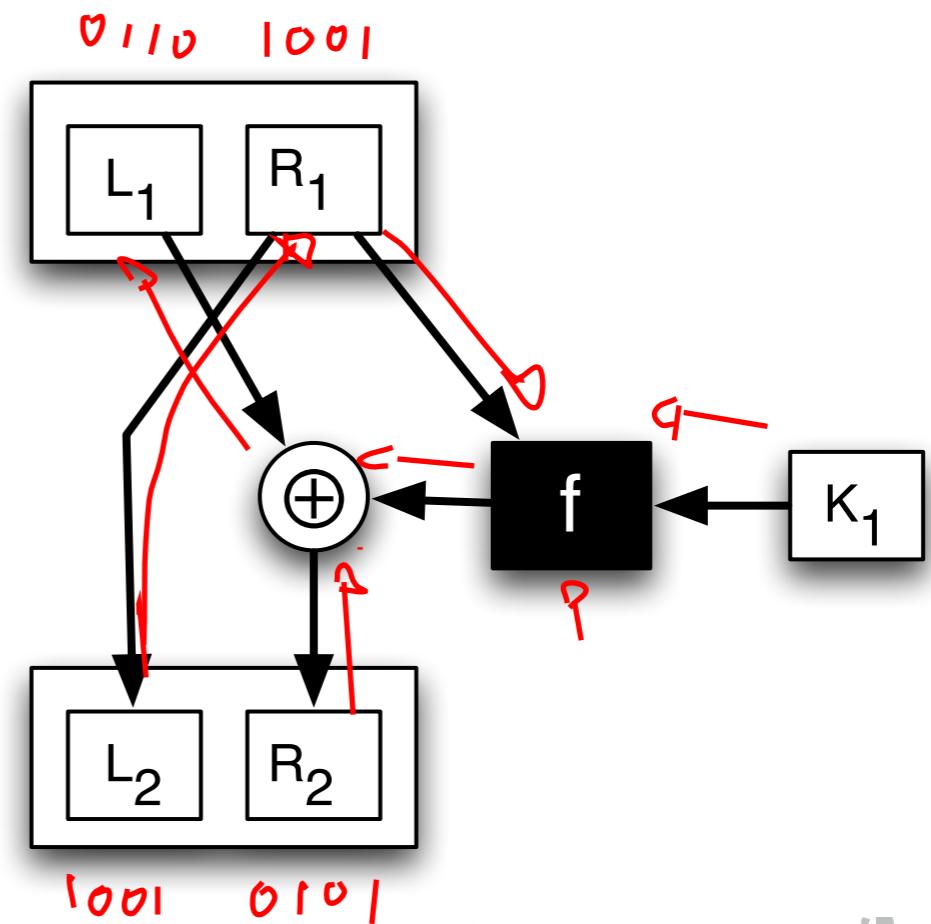
$$f(x_3) = (n-2)$$

Block-Chiffre

- Wie viele mögliche Abbildungen gibt es für k-Bit Block-Chiffre?
 - Im allgemeinen: $2^k!$
 - riesig für $k=64$
 - und absolut sicher, wenn man sie zufällig auswählt
- Problem:
 - Die meisten dieser Abbildungen benötigen große Tabellen um sie zu berechnen
- Lösung
 - Statt einer Tabelle, verwendet man eine Funktion, die diese Tabelle simuliert
 - Dadurch verliert man möglicherweise wieder die Sicherheit

Feistel-Chiffre

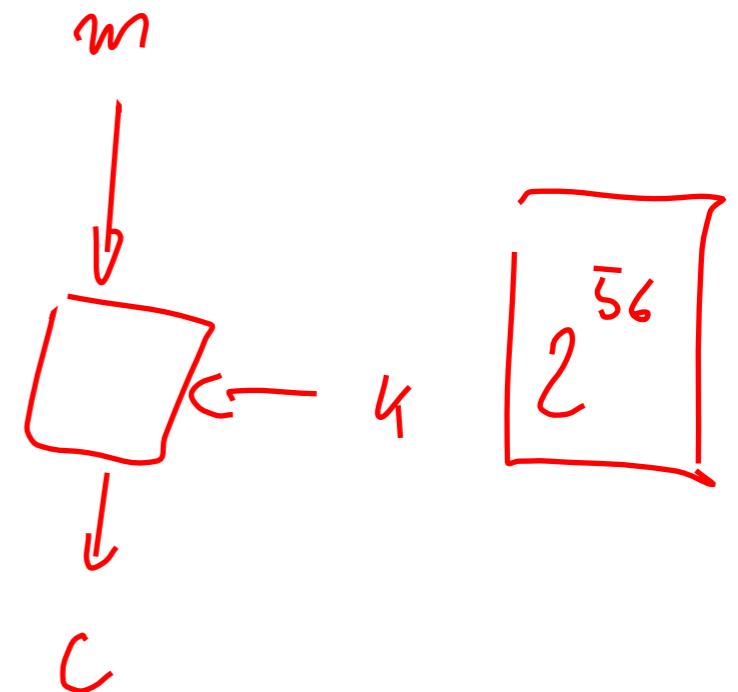
- Aufteilung der Nachricht in zwei Hälften L_1, R_1
 - Schlüssel K_1, K_2, \dots
 - Mehrere Runden: resultierender Code: L_n, R_n
- Verschlüsselung
 - $L_i = R_{i-1}$
 - $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- Entschlüsselung
 - $R_{i-1} = L_i$
 - $L_{i-1} = R_i \oplus f(L_i, K_i)$
- f : beliebige, komplexe Funktion



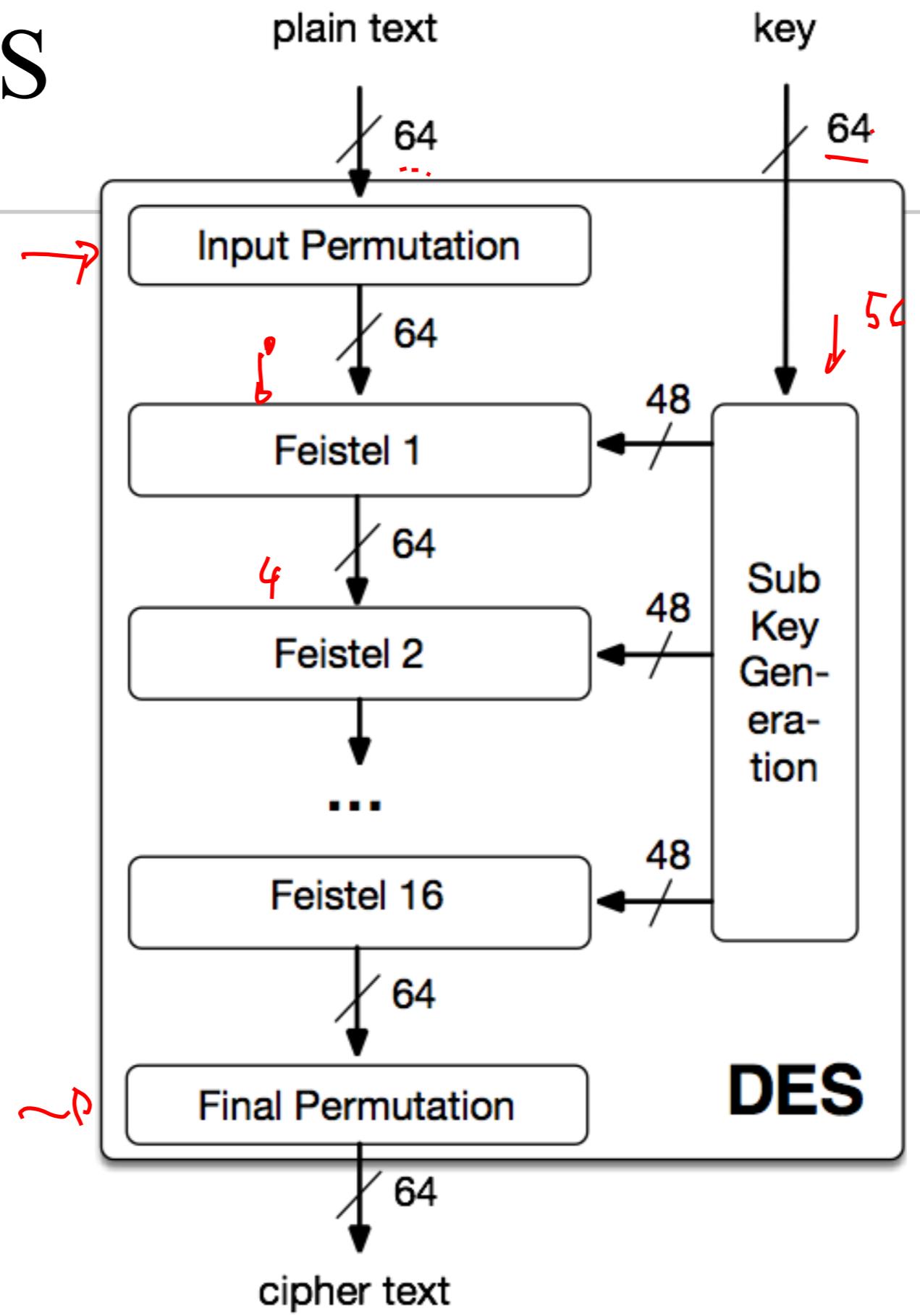
- **Skipjack**
 - 80-Bit symmetrischer Code
 - baut auf Feistel-Chiffre auf
 - wenig sicher
- **RC5**
 - Schlüssellänge 1-2048 Bits
 - Rivest Code 5 (1994)
 - Mehrere Runden der Feistel-Chiffre

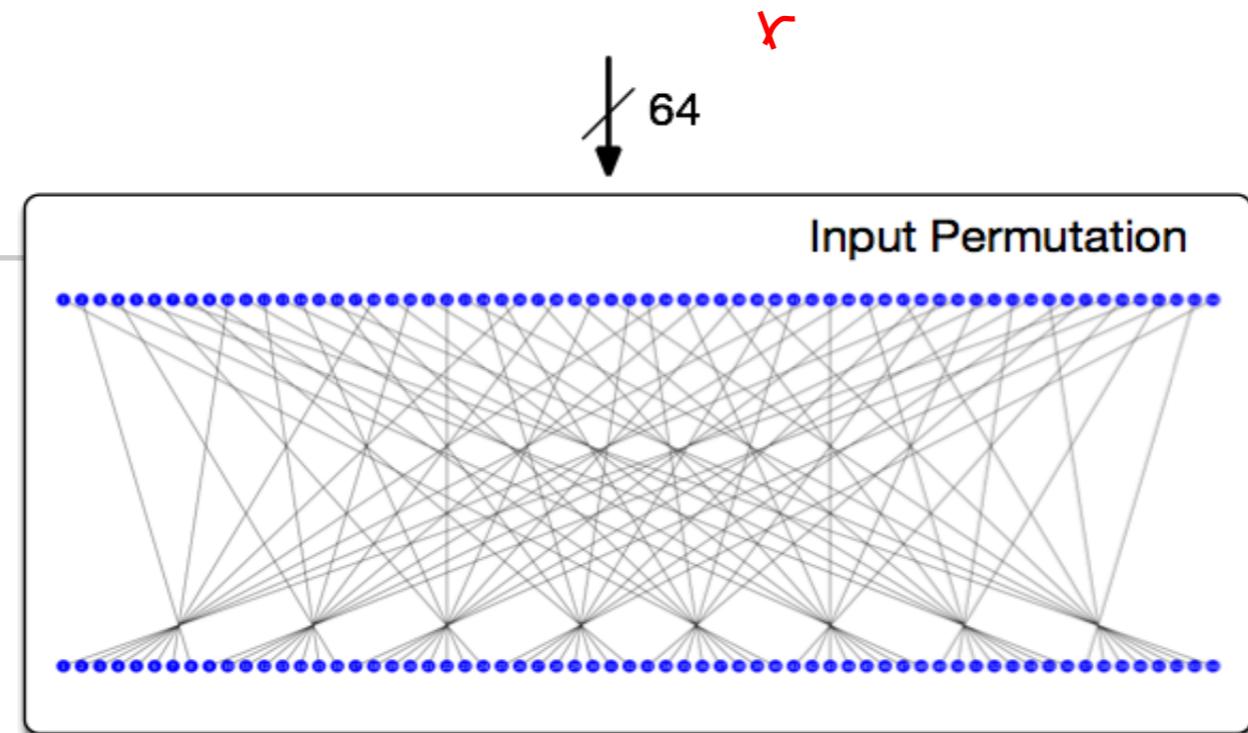
Digital Encryption Standard

- Geschickt gewählte Kombination von
 - Xor-Operationen
 - Feistel-Chiffre ↙
 - Permutationen
 - Table-Lookups
 - verwendet 56-Bit Schlüssel
- 1975 entwickelt von Wissenschaftlern von IBM
 - Mittlerweile nicht mehr sicher
 - leistungsfähigere Rechner
 - Erkenntnisse in Kryptologie
- Nachfolger: AES (2001)

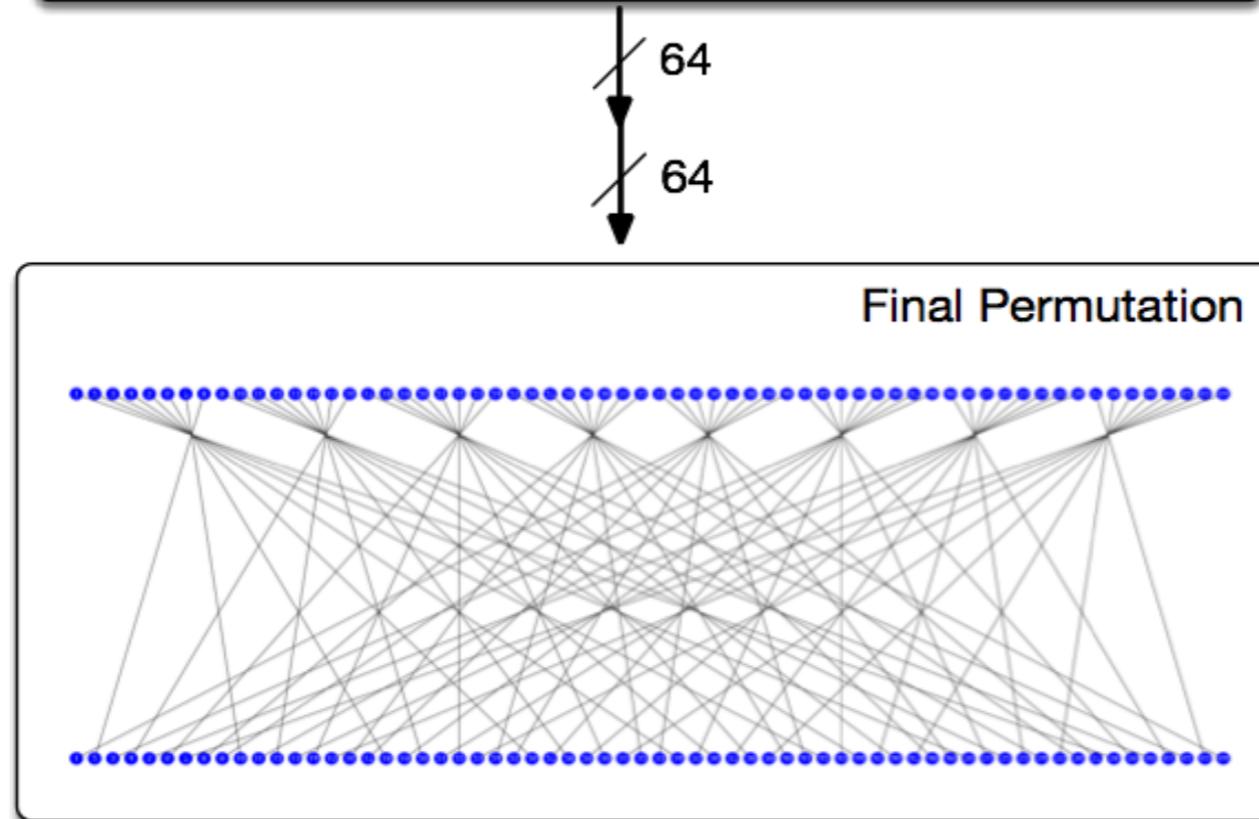


DES





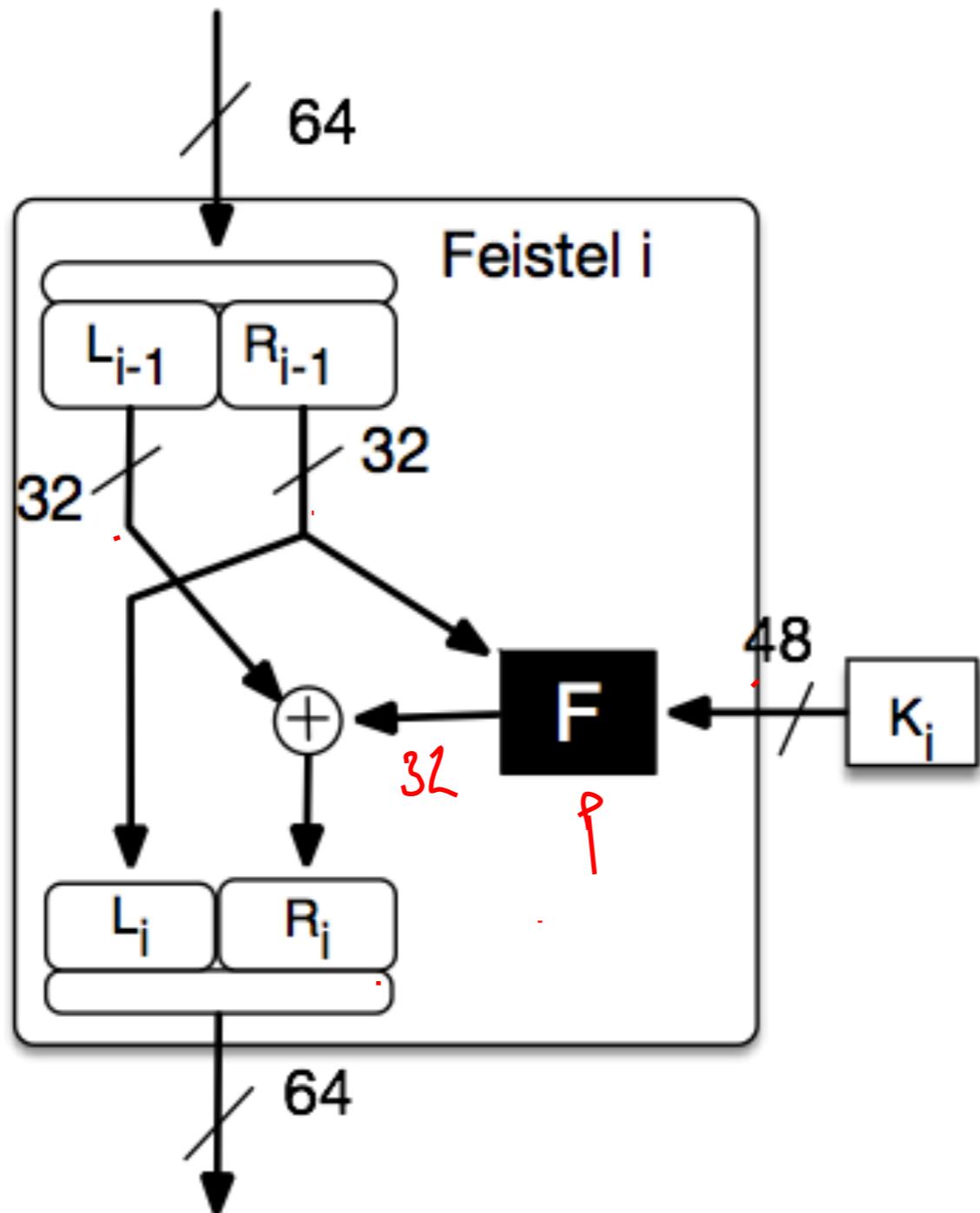
64
↓
X



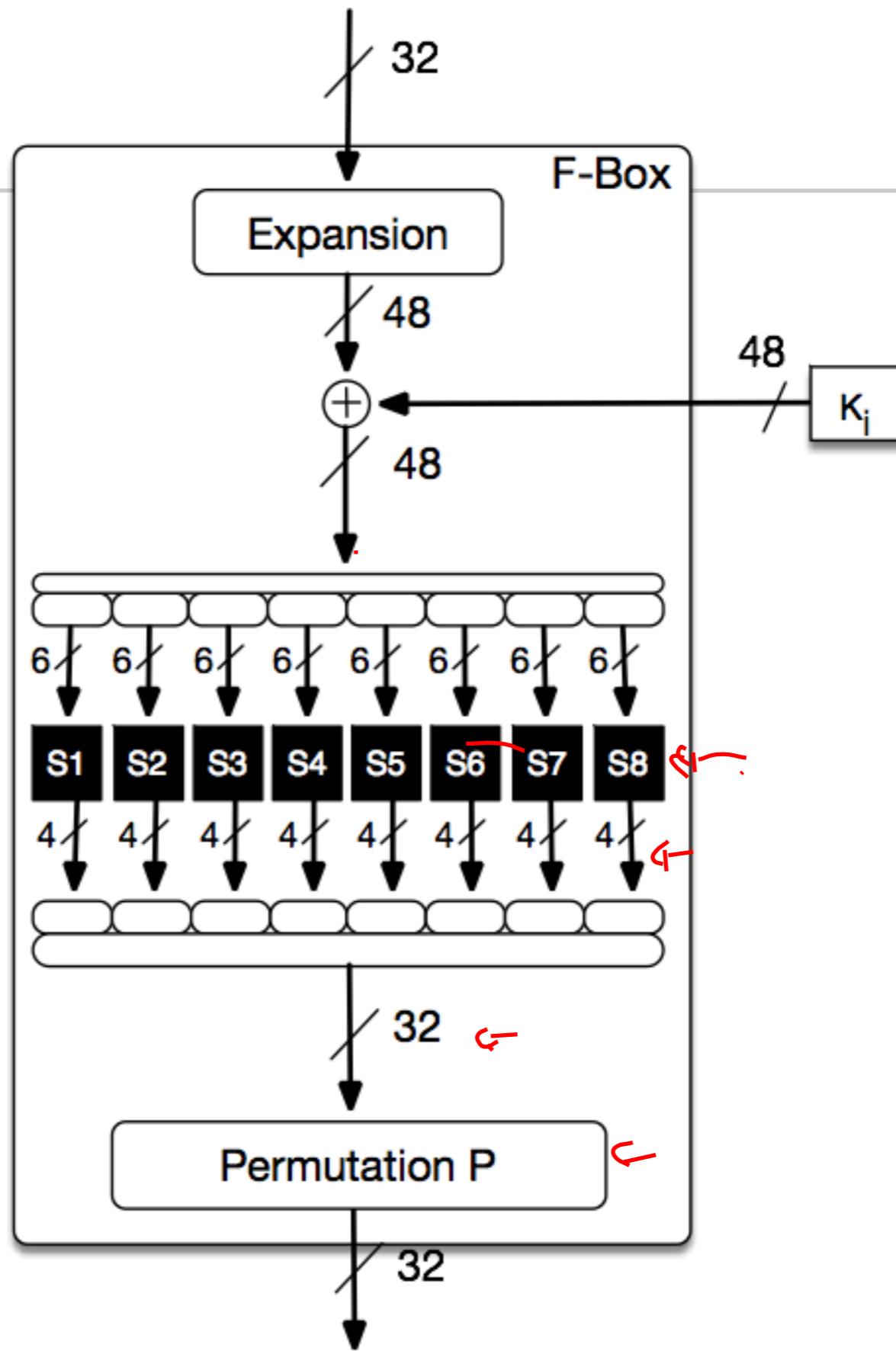
64
↓
64
↓

64
↓
X

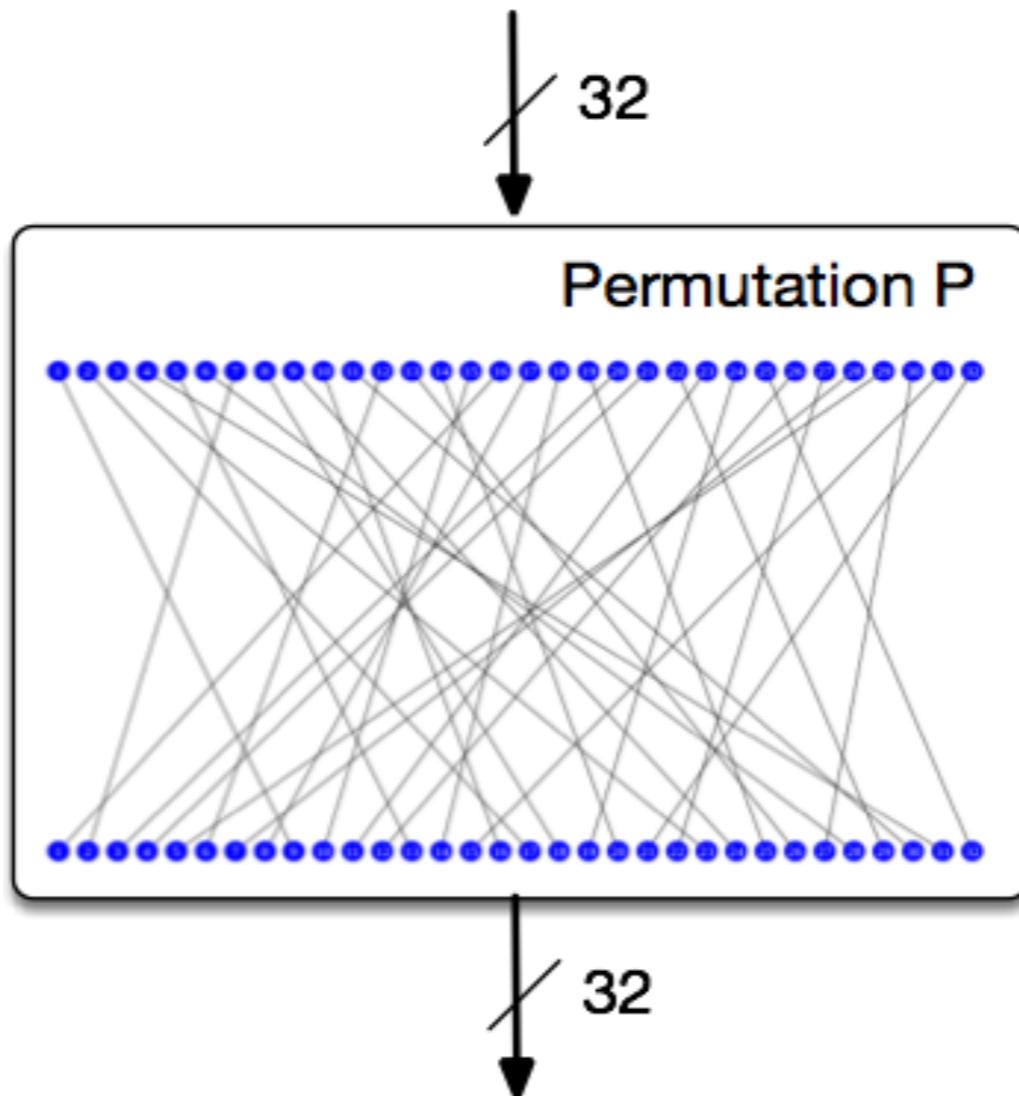
Feistel



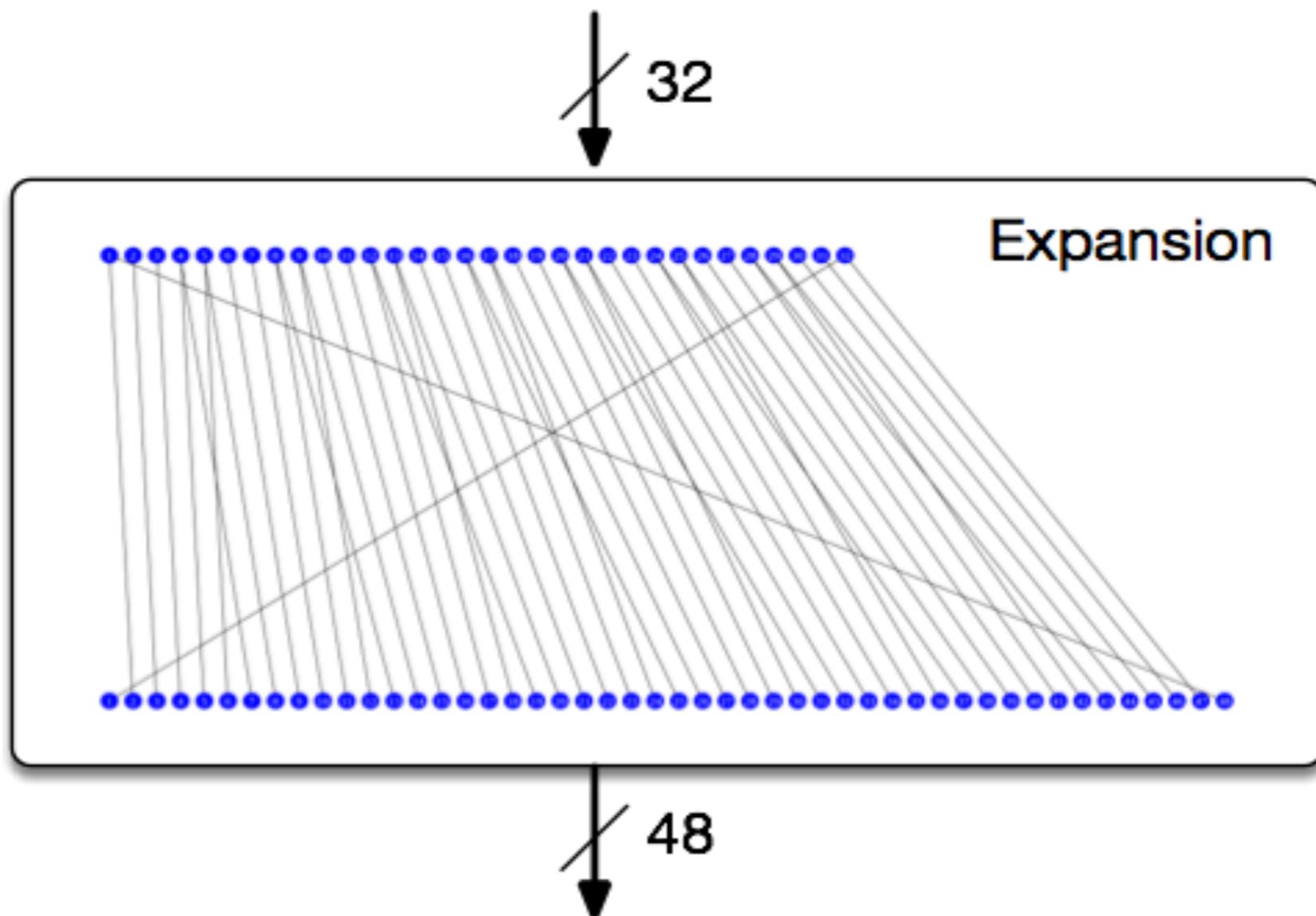
F-Box



Permutation



Expansion

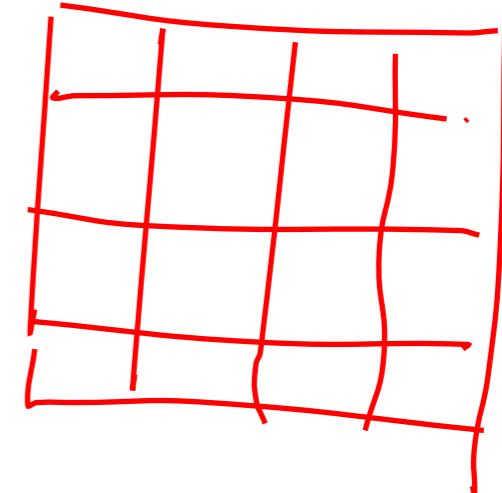


S-Box

S-boxes																
S₁	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S₂	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S₃	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S₄	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S₅	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S₆	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S₇	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S₈	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Advanced Encryption Standard

- Geschickt gewählte Kombination von
 - Xor-Operationen
 - Permutationen
 - Multiplikation in $GF[2^8]$
 - symmetrische 128, 192 oder 256-Bit Schlüssel
- Joan Daemen und  Vincent Rijmen
 - 2001 als AES unter vielen ausgewählt worden
 - bis heute als sicher erachtet

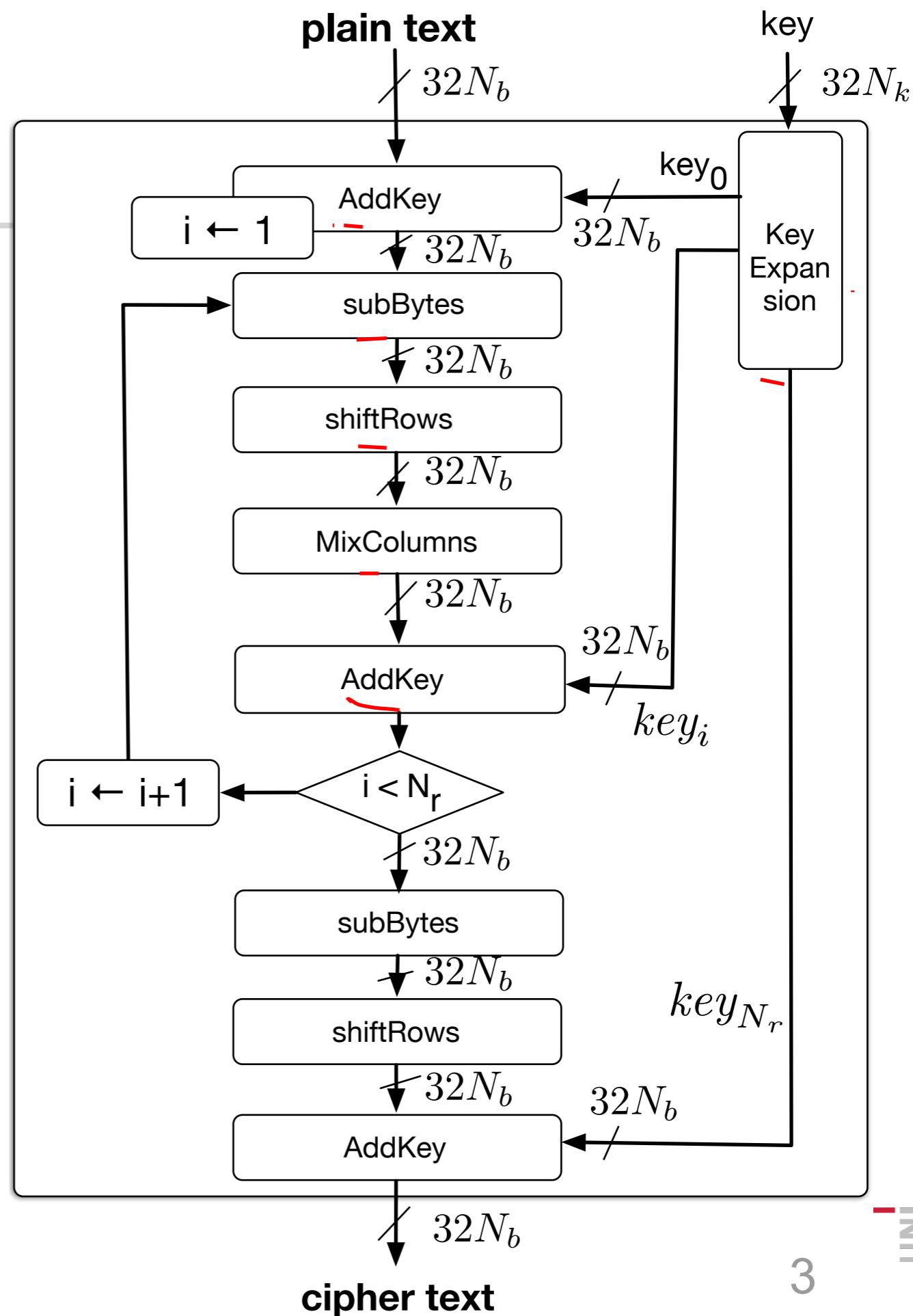


Zustände

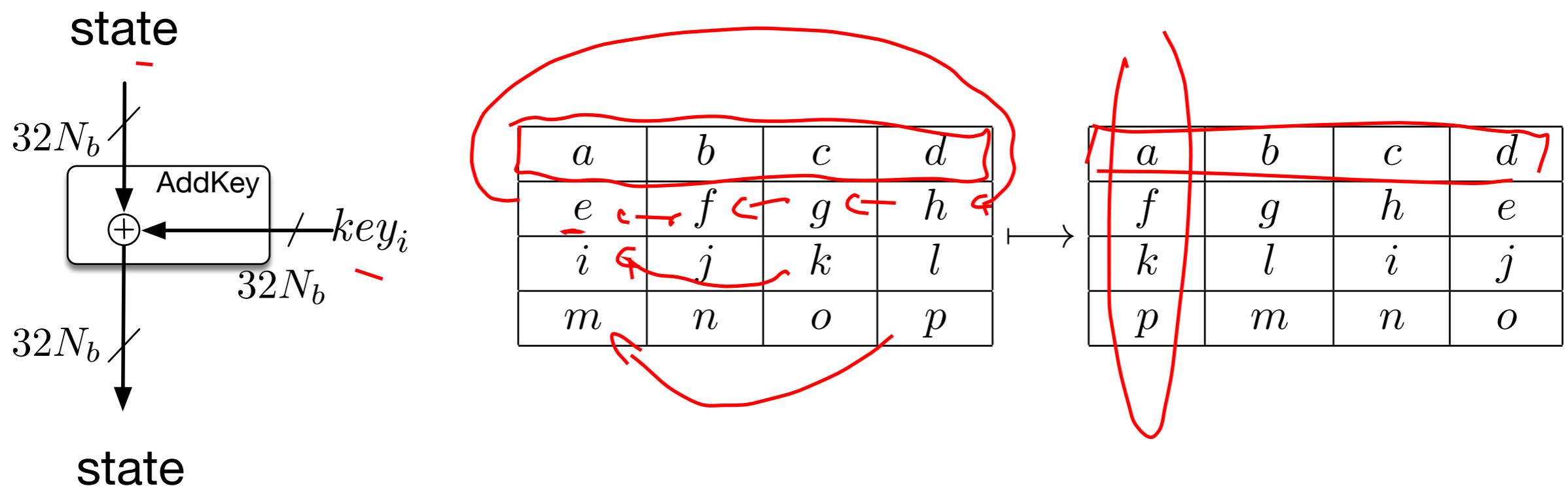
$$\begin{pmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N_b-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N_b-1} \\ a_{2,0} & a_{2,1} & \dots & a_{2,N_b-1} \\ a_{3,0} & a_{3,1} & \dots & a_{3,N_b-1} \end{pmatrix}$$

N_r

N_k	N_b				
	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14



AddKey & ShiftRows



SubBytes

$$S_{RD}(x) = \underline{g} \circ f(x) = g(f(x))$$

$$f : \mathbb{F}_{2^8} \longrightarrow \mathbb{F}_{2^8}, \quad x \longmapsto \begin{cases} x^{-1} & \text{if } x \neq 0, \\ 0 & \text{if } x = 0. \end{cases}$$

$$g : \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8, \quad x \longmapsto \underline{Ax + b},$$

$$A := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N_b-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N_b-1} \\ a_{2,0} & a_{2,1} & \dots & a_{2,N_b-1} \\ a_{3,0} & a_{3,1} & \dots & a_{3,N_b-1} \end{pmatrix}$$

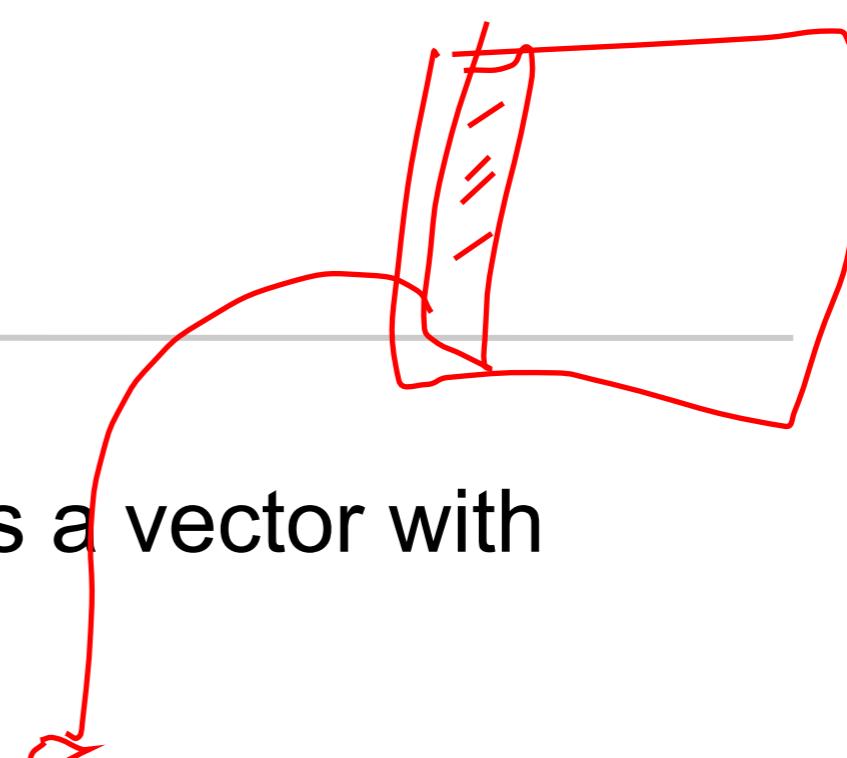
$$\mapsto \begin{pmatrix} S_{RD}(a_{0,0}) & S_{RD}(a_{0,1}) & \dots & S_{RD}(a_{0,N_b-1}) \\ S_{RD}(a_{1,0}) & S_{RD}(a_{1,1}) & \dots & S_{RD}(a_{1,N_b-1}) \\ S_{RD}(a_{2,0}) & S_{RD}(a_{2,1}) & \dots & S_{RD}(a_{2,N_b-1}) \\ S_{RD}(a_{3,0}) & S_{RD}(a_{3,1}) & \dots & S_{RD}(a_{3,N_b-1}) \end{pmatrix}$$

MixColumns

- Interpret a column of the status as a vector with four elements over GF[256]

$$\begin{pmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{pmatrix}, \quad j = 0, 1, 2, 3.$$

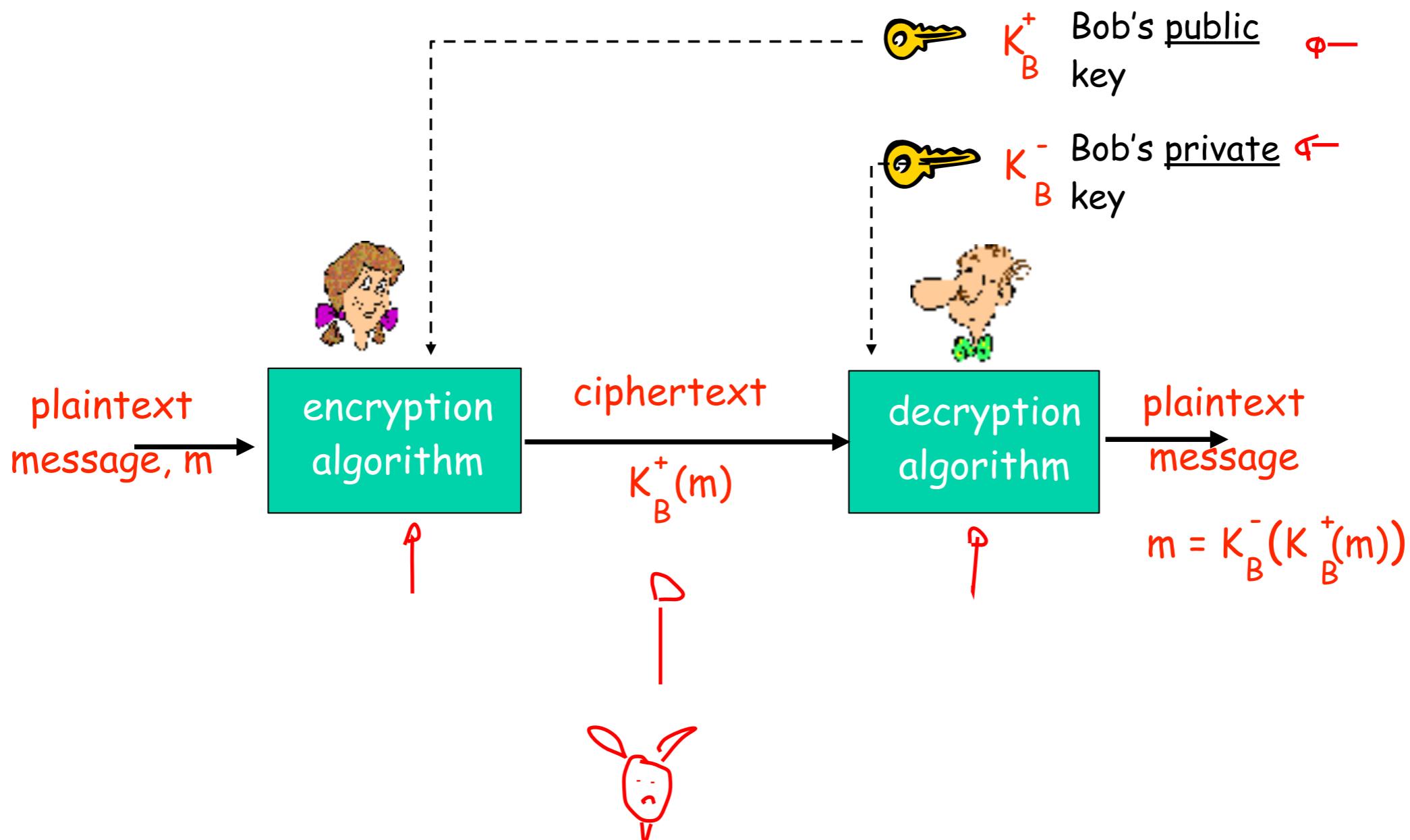
A



- Inverse matrix:

$$\begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}^{-1} = A^{-1}$$

Public key cryptography



Asymmetrische Verschlüsselungsmethoden

- z.B. RSA, Ronald Rivest, Adi Shamir, Lenard Adleman, 1977
 - Diffie-Hellman, PGP
- Geheimer Schlüssel privat: kennt nur der Empfänger der Nachricht
- Öffentlichen Schlüssel offen: Ist allen Teilnehmern bekannt
- Wird erzeugt durch Funktion
 - keygen(privat) = offen
- Verschlüsselungsfunktion f und Entschlüsselungsfunktion g
 - sind auch allen bekannt
- Verschlüsselung
 - f(offen,text) = code
 - kann jeder berechnen
- Entschlüsselung
 - g(privat,code) = text
 - nur vom Empfänger

$$199 = 2 \cdot 98 = 2 \cdot 2 \cdot 7^2$$

199
 317

- R. Rivest, A. Shamir, L. Adleman
 - On Digital Signatures and Public Key Cryptosystems, Communication of the ACM

- Verfahren beruht auf der Schwierigkeit der Primfaktorzerlegung

- 1. Beispiel:

- $15 = ? * ?$
 - $15 = 3 * 5$

Shor

$$P \cdot q = m$$


- 2. Beispiel:

- $3865818645841127319129567277348359557444790410289933586483552047443 = 1234567890123456789012345678900209 * 31313131313131313131313131300227$

q p

- Bis heute ist kein effizientes Verfahren zur Primfaktorzerlegung bekannt
 - Aber das Produkt von Primzahlen kann effizient bestimmt werden
 - Primzahlen können effizient bestimmt werden
 - Primzahlen sehr häufig

26
30
40

$$\#\text{Primzahlen } \in [2^m, 2^n] \sim \frac{2^n}{m}$$

■ Erzeugung der Schlüssel

- Wähle zufällig zwei Primzahlen p und q mit k bits ($k \geq 500$).
- $n = p \cdot q$
- e ist Zahl, die teilerfremd ist mit $(p - 1) \cdot (q - 1)$.
- $d = e^{-1} = 1/e \bmod (p - 1)(q - 1)$
 - es gilt $d \cdot e \equiv 1 \bmod (p - 1)(q - 1)$

$$d \cdot e \equiv 1 \bmod (p - 1)(q - 1)$$

■ Public Key $P = (\underline{e}, \underline{n})$

■ Secret Key $S = (\underline{d}, \underline{n})$

$$\begin{array}{c}
 \boxed{M^e} \\
 = \dots M^8 \cdot \underbrace{M^4 \cdot M^1}_{M^4} \\
 M^{101011101} \mod n
 \end{array}$$

$$M^4 = \underbrace{M^2 \mod n}$$

Kodierung

- Teile Nachricht in Blöcke der Größe 2^{2k} auf
- Interpretiere Block M als Zahl $0 \leq M < 2^{2k}$
- Chiffre: $P(M) = \underline{M^e \mod n} \cdot \underbrace{C}_a$

Dekodierung

- $S(C) = C^d \mod n$

$$M^{e \cdot d} \stackrel{\substack{\mod (p-1)(q-1) \\ 1}}{=} M \quad (\text{mod } n)$$

Korrektheit gilt nach dem kleinen Satz von Fermat

- Für Primzahl p und von p teilerfremde Zahl a gilt:

$$\begin{array}{l}
 a^p \equiv a \quad (\text{mod } p) \\
 a^{p-1} \equiv 1 \quad (\text{mod } p)
 \end{array}$$

$$\begin{array}{l}
 a^{(p-1)(q-1)} \equiv a^0 \quad (\text{mod } n) \\
 \Rightarrow a^x \equiv a^{x \mod (p-1)(q-1)} \quad (\text{mod } n)
 \end{array}$$

$$(M^e)^d \equiv M \quad (\text{mod } n)$$

RSA Beispiel

- Bob wählt $p=5$, $q=7$
 - $n = \underline{35}$, $\varphi = \underline{24}$
 - $e = 5$
 - $d = \underline{29}$
 - $e d = 1 \bmod 24$
- Verschlüsselung von 8-Bit-Nachrichten

Bit pattern	m	$\cancel{m^e} \cancel{\bmod n}$	$c = m^e \bmod n$
00001000	12	248 832	17

- Entschlüsselung

$$\begin{array}{ll} c & c^d = 17^{29} \\ 17 & 481968572106750915091411825223071697 \end{array} \quad \begin{array}{l} m = c^d \bmod n \\ 12 \end{array}$$

RSA Beispiel

- Berechnung von $17^{29} \bmod 35$
- $29 = 11101_2$
 - $17^{29} = 17 \cdot (17^{14})^2 \bmod 35$
 - $17^{14} = (17^7)^2 \bmod 35$
 - $17^7 = 17 \cdot (17^3)^2 \bmod 35$
 - $17^3 = 17 \cdot (17)^2 \bmod 35$
- Einsetzen:
 - $17^3 = 4913 = 13 \bmod 35$
 - $17^7 = 17 \cdot 13^2 = 2873 = 3 \bmod 35$
 - $17^{14} = 3^2 = 9 \bmod 35$
 - $17^{29} = 17 \cdot 9^2 = 1377 = \underline{12} \bmod 35$

Nachrichtenintegrität

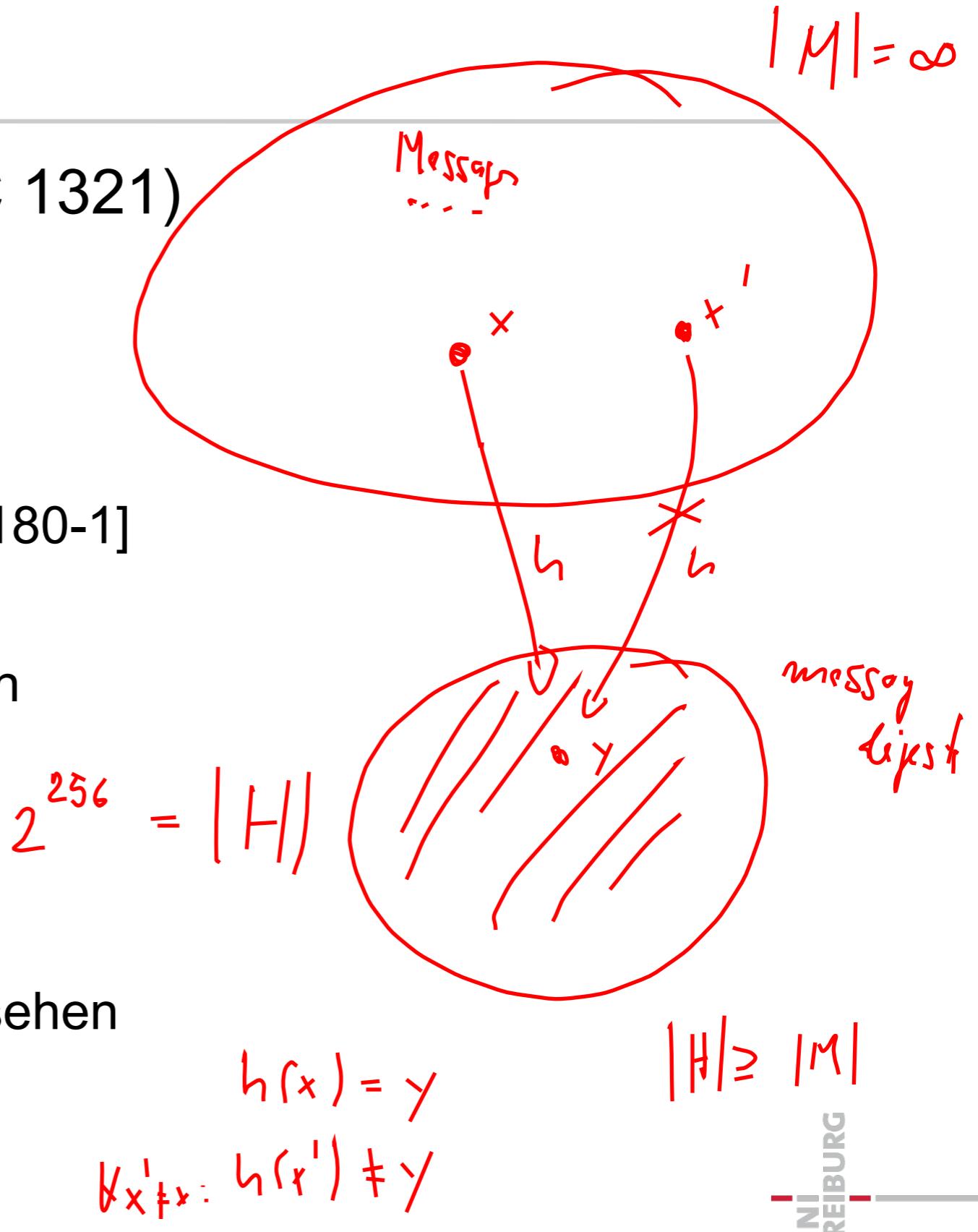
- Erlaubt den Kommunikationspartnern die Korrektheit und Authentizität der Nachricht zu überprüfen
 - Inhalt ist unverändert
 - Urheber ist korrekt
 - Nachricht ist keine Wiederholung
 - Reihenfolge der Nachrichten ist korrekt
- Message Digests

Kryptographische Hash-Funktion

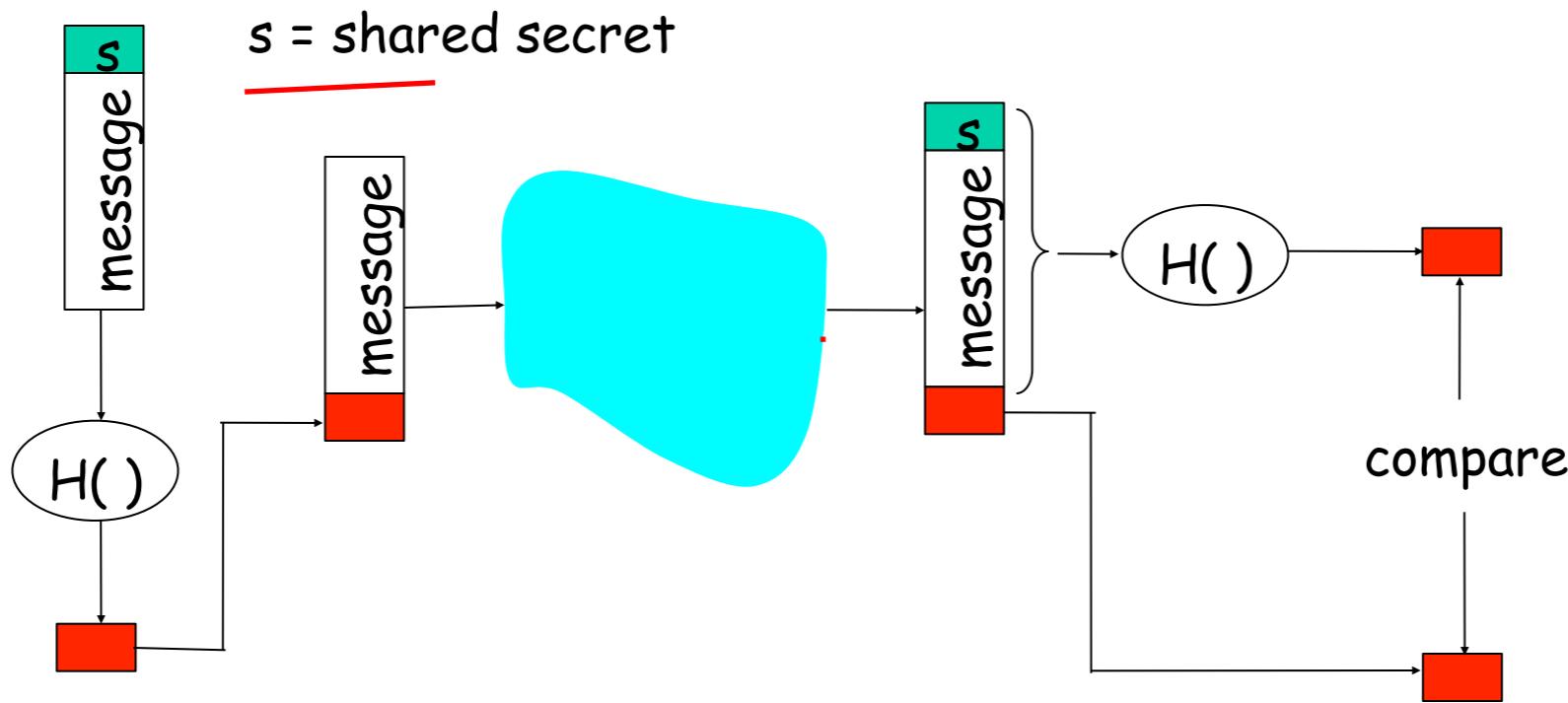
- z.B. SHA-1, SHA-2, MD5
- Ein kryptographische Hash-Funktion h bildet einen Text auf einen Code fester Länge ab, so dass
 - $h(\underline{\text{text}}) = \underline{\text{code}}$
 - es unmöglich ist einen anderen Text zu finden mit:
 - $\underline{h(\text{text}')} = h(\text{text})$ und $\text{text} \neq \text{text}'$
- Mögliche Lösung:
 - Verwendung einer symmetrischen Kodierung

Kryptographische Hash-Algorithmen

- MD5 ist sehr verbreitet (RFC 1321)
 - berechnet 128-bit Nachricht
 - unsicher
- SHA-1 auch gebräuchlich
 - US standard [NIST, FIPS PUB 180-1]
 - 160-bit Message Digest
 - nicht mehr als sicher angesehen
- SHA-2
 - SHA-256/224
 - SHA-512/384
 - bis jetzt (2012) als sicher angesehen
- SHA-3
 - 2011 veröffentlicht



Message Authentication Code (MAC)



- Authentifiziert Absender
- Überprüft Nachrichtenintegrität
- Keine Verschlüsselung
- “keyed hash”
- Notation: $MDm = H(s \parallel m)$; sende $m \parallel MDm$