



Kapitel 7: Symmetrische Kryptosysteme



■ Symmetrische Verschlüsselungsverfahren

- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

■ Kryptoregulierung

- 1977 vom NBS (National Bureau of Standards; heute: National Institute of Standards (NIST)) in USA zum Standard erklärt
- 2002 durch AES (Advanced Encryption Standard) ersetzt
- DES entwickelt von IBM aus dem 128-Bit-Verfahren LUCIFER
- Klassifikation:
 - Symmetrisches Verfahren
 - Mit Permutation, Substitution und bitweiser Addition modulo 2
 - Blockchiffre mit 64 Bit großen Ein- und Ausgabeblöcken
 - Schlüssellänge 64 Bit, davon 8 Paritätsbits, d.h. effektive Schlüssellänge (nur) 56 Bit
- Bedeutung von DES:
 - Erstes standardisiertes Verfahren mit intensiver, weltweiter Nutzung
 - Aus heutiger Sicht einfach zu knacken (Verbesserung: 3DES)
 - Zeigt aber viele Bestandteile moderner symmetrischer Verschlüsselungsverfahren.

Deep Crack

- 1998 von der Electronic Frontier Foundation (EFF) für rund \$250.000 gebaut.
- 29 beidseitig bestückte Platinen mit je 64 Deep Crack Chips
- Knackte DES-Schlüssel innerhalb weniger Tage.
- Sollte demonstrieren, dass DES nicht mehr sicher ist.

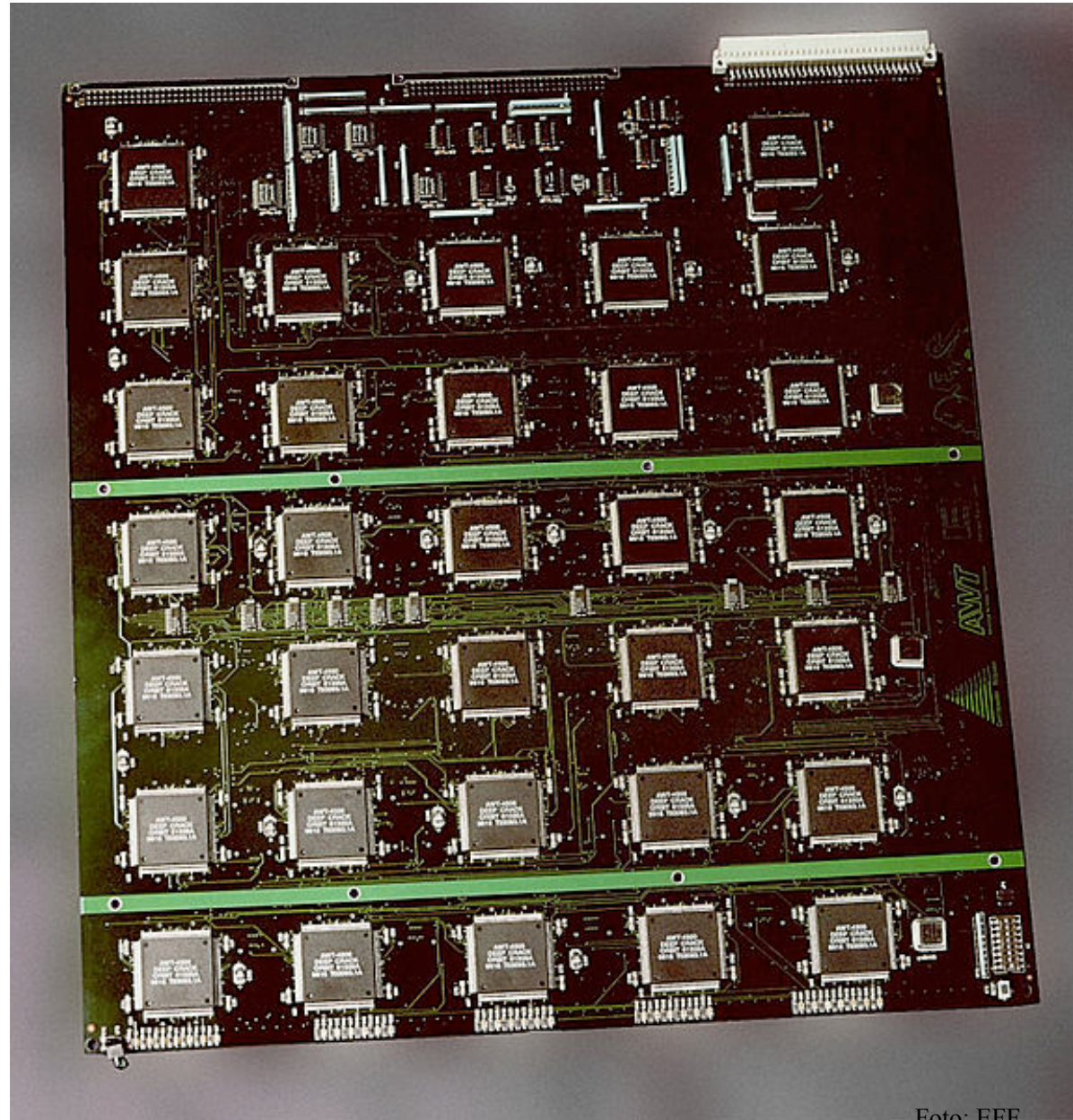
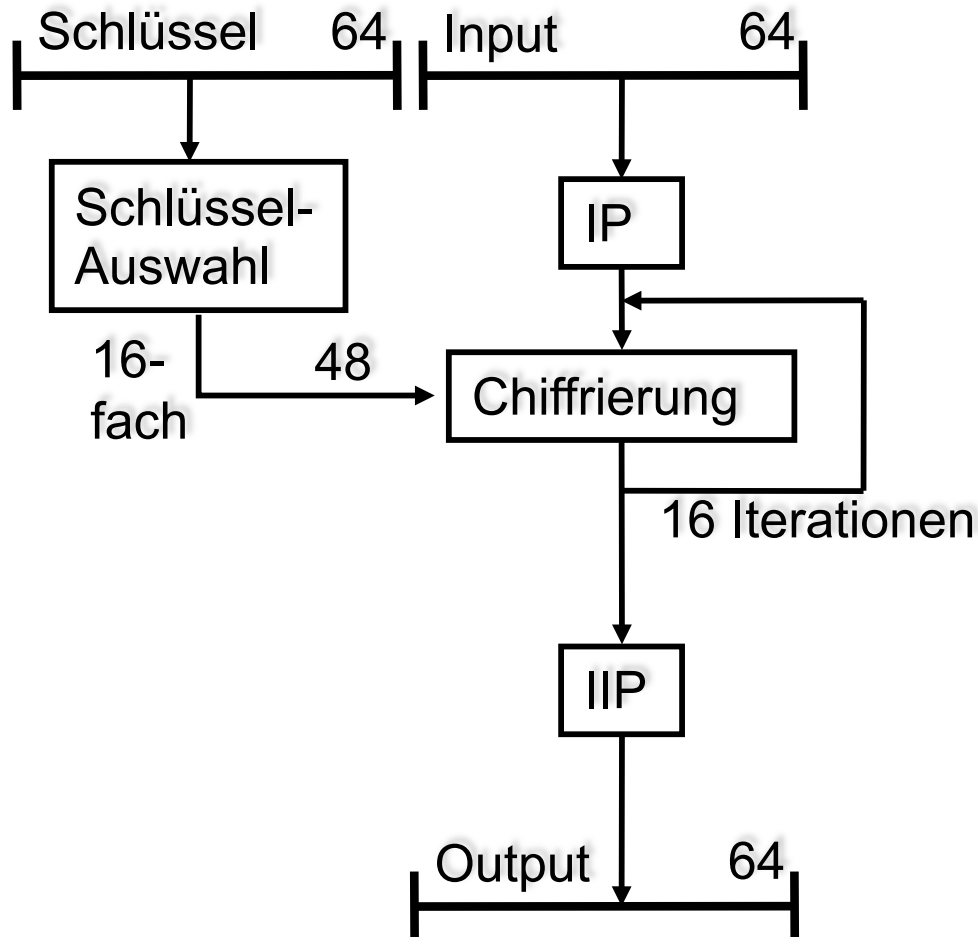


Foto: EFF

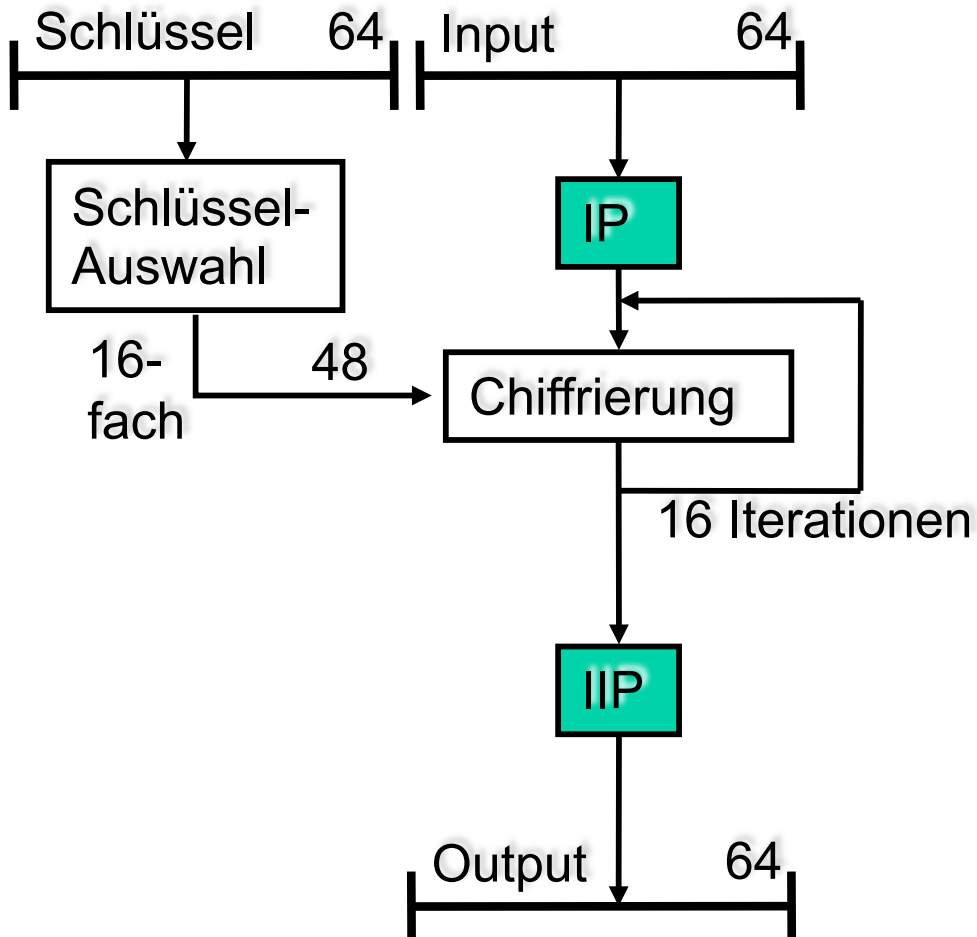
DES: Grundlegender Ablauf



■ Ablauf der Verschlüsselung:

1. Initialpermutation (IP) des 64-bit Input-Blocks
2. 16 schlüsselabhängige Iterationen
 - 48 Bit lange Teilschlüssel
 - werden aus 64 Bit langem Schlüssel generiert (davon 8 Paritätsbits)
3. Inverse Initialpermutation (IIP) als Ausgabepermutation

- Entschlüsselung analog zur Verschlüsselung mit Teilschlüsseln in umgekehrter Reihenfolge im Schritt 2.



- Wie arbeiten Initialpermutation (IP) und Inverse Initialpermutation (IIP)?

■ Initialpermutation IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

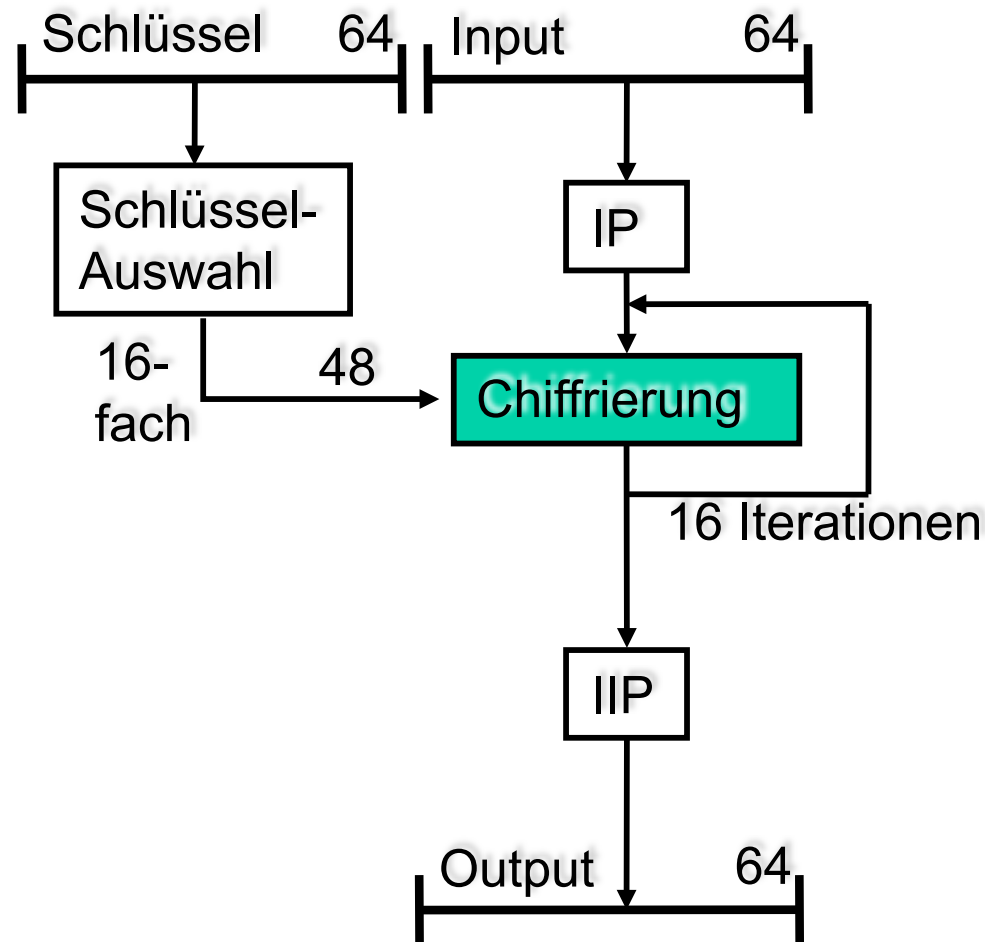
- D.h. aus Bit 58 des Input wird Bit 1, aus Bit 50 wird Bit 2,....., aus Bit 7 wird Bit 64

■ Inverse Initialpermutation IIP

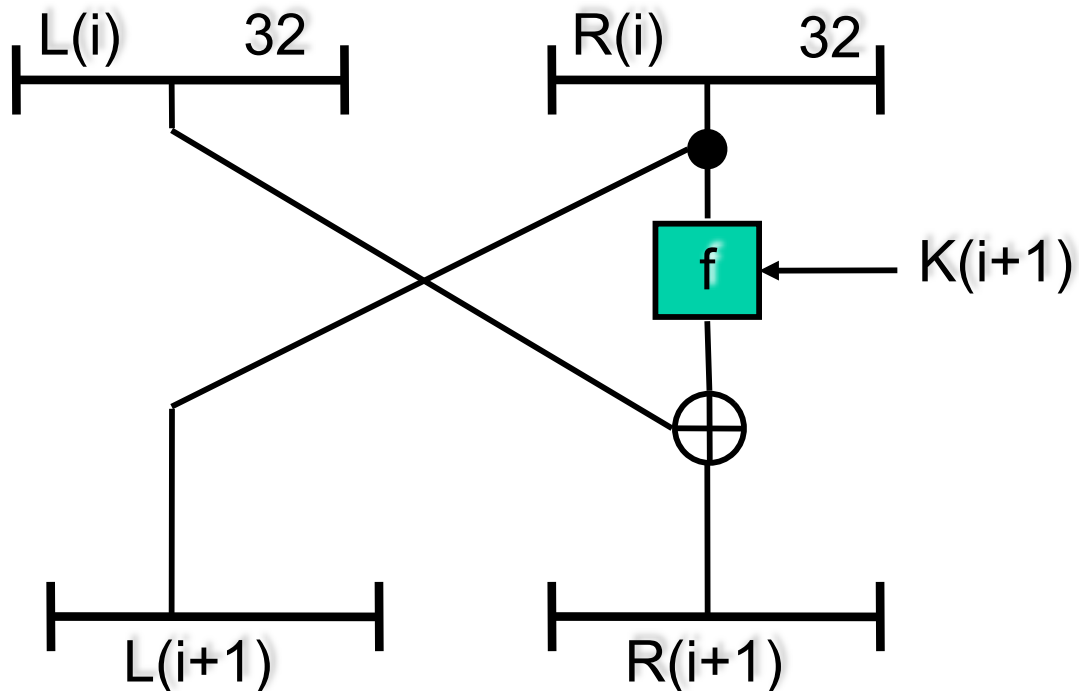
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- IP und IIP heben sich gegenseitig auf (Inversion)
- Gleichmäßige Aufteilung auf die linke bzw. rechte Hälfte (vgl. Folie 9).

DES Funktion: Grundschema



■ Ein Schritt (Runde) der Chiffrierung:



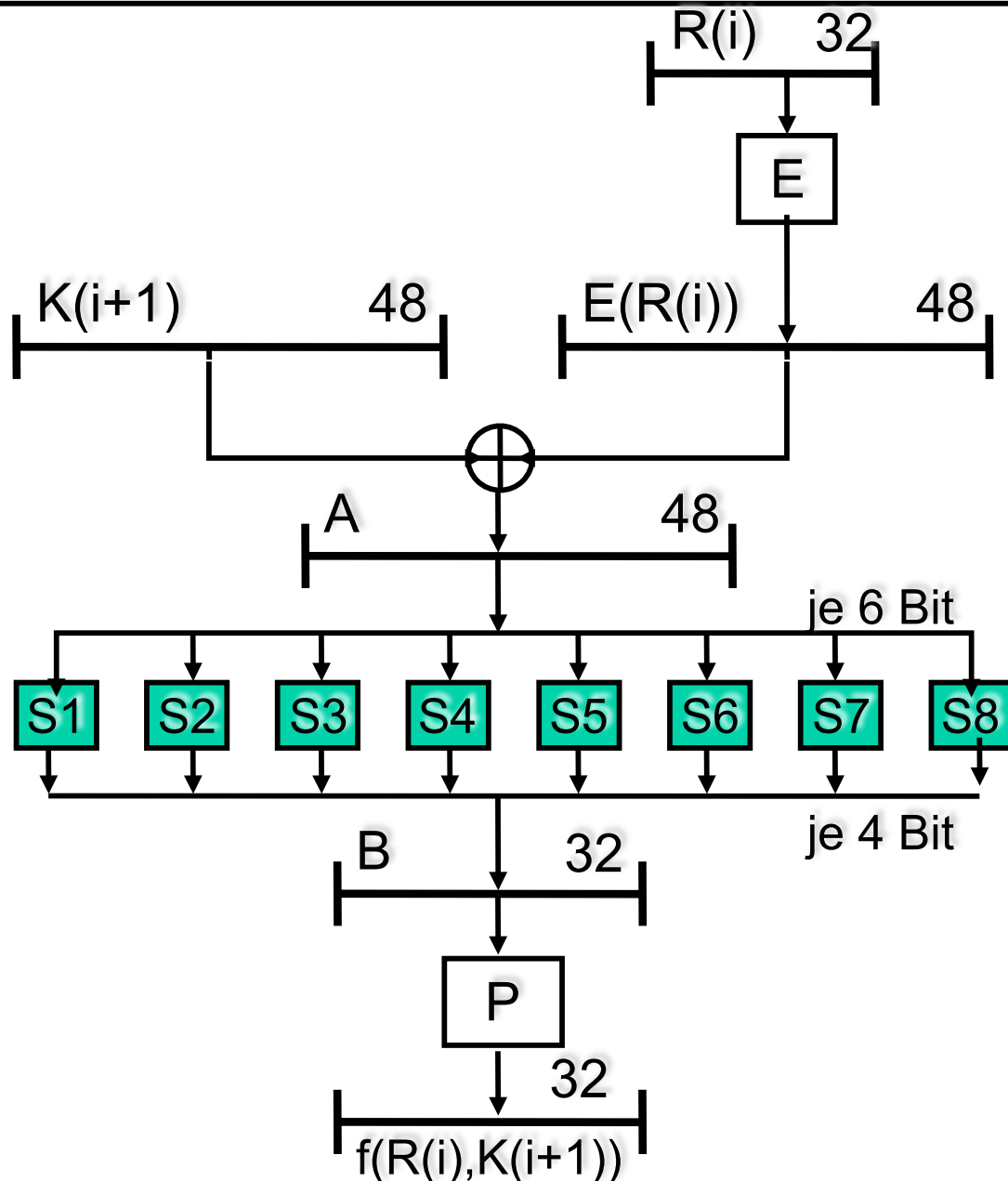
\oplus Addition modulo 2; entspricht XOR

0 xor 0 = 0 1 xor 0 = 1
0 xor 1 = 1 1 xor 1 = 0

- Verschlüsselungsblock (64 Bit) wird in linken (L) und rechten (R) Block zu je 32 Bit aufgeteilt
- $K(i)$ = Schlüssel in Runde i
- Anwendung der Verschlüsselungsiteration:
 $L(0) = L$ und $R(0) = R$
 $L(i+1) = R(i)$
 $R(i+1) = L(i) \text{ XOR } f(R(i), K(i+1))$

für $i=0, \dots, 15$

- Funktion f stellt Kern des Verfahrens dar.



- Rechter 32 Bit Input Block wird mittels Expansion E auf 48 Bit expandiert
- XOR-Verknüpfung mit dem (Runden-) Schlüssel zum 48 Bit langen Block A
- A wird in 8 Blöcke zu je 6 Bit aufgeteilt
- Jeder dieser 8 Blöcke wird durch S-Box (Substitution) in 4 Bit lange Ausgabeblöcke (nichtlinear!) abgebildet
- Konkatenation der acht 4 Bit langen Blöcke ergibt Block B, der noch der (lokalen) Ausgangspermutation P unterworfen wird (nicht IIP!)

■ Expansion E:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- Bit 32 aus $R(i)$ wird sowohl Bit 1 als auch Bit 47 von $E(R(i))$
- Bit 1 aus $R(i)$ wird sowohl Bit 2 als auch Bit 48 von $E(R(i))$
- ...

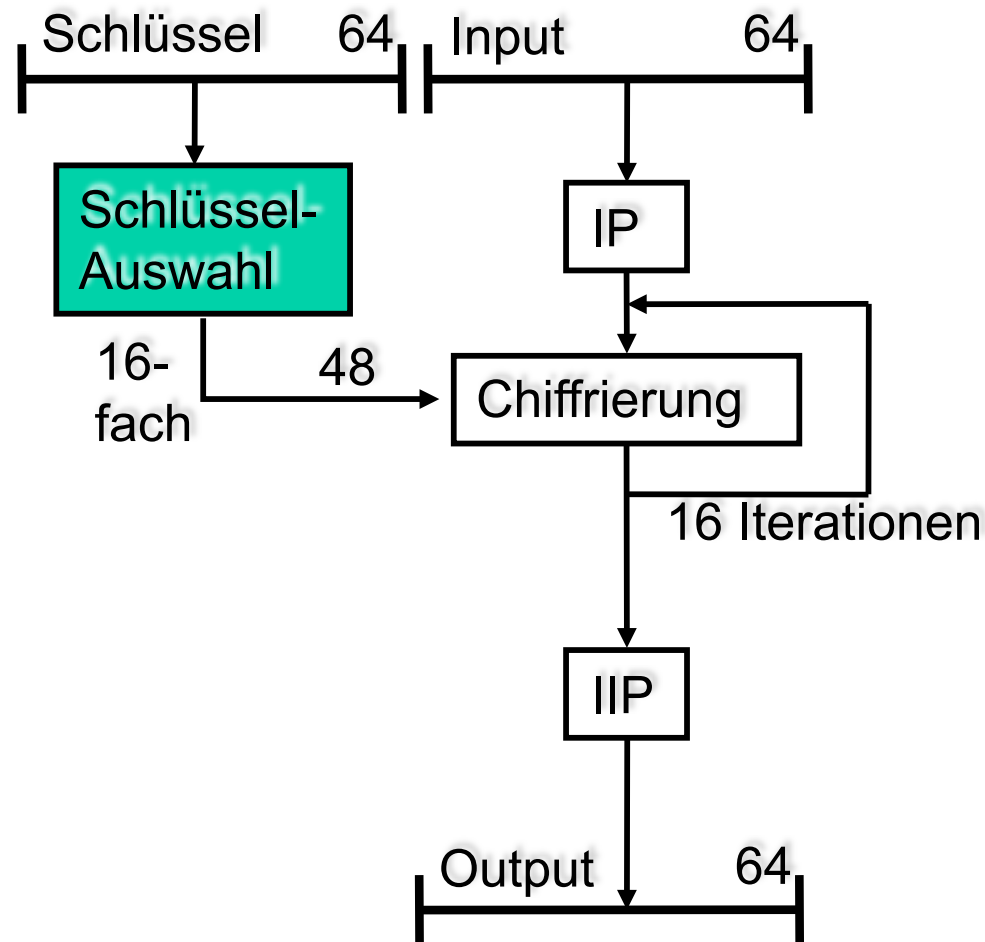
■ Ausgangspermutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

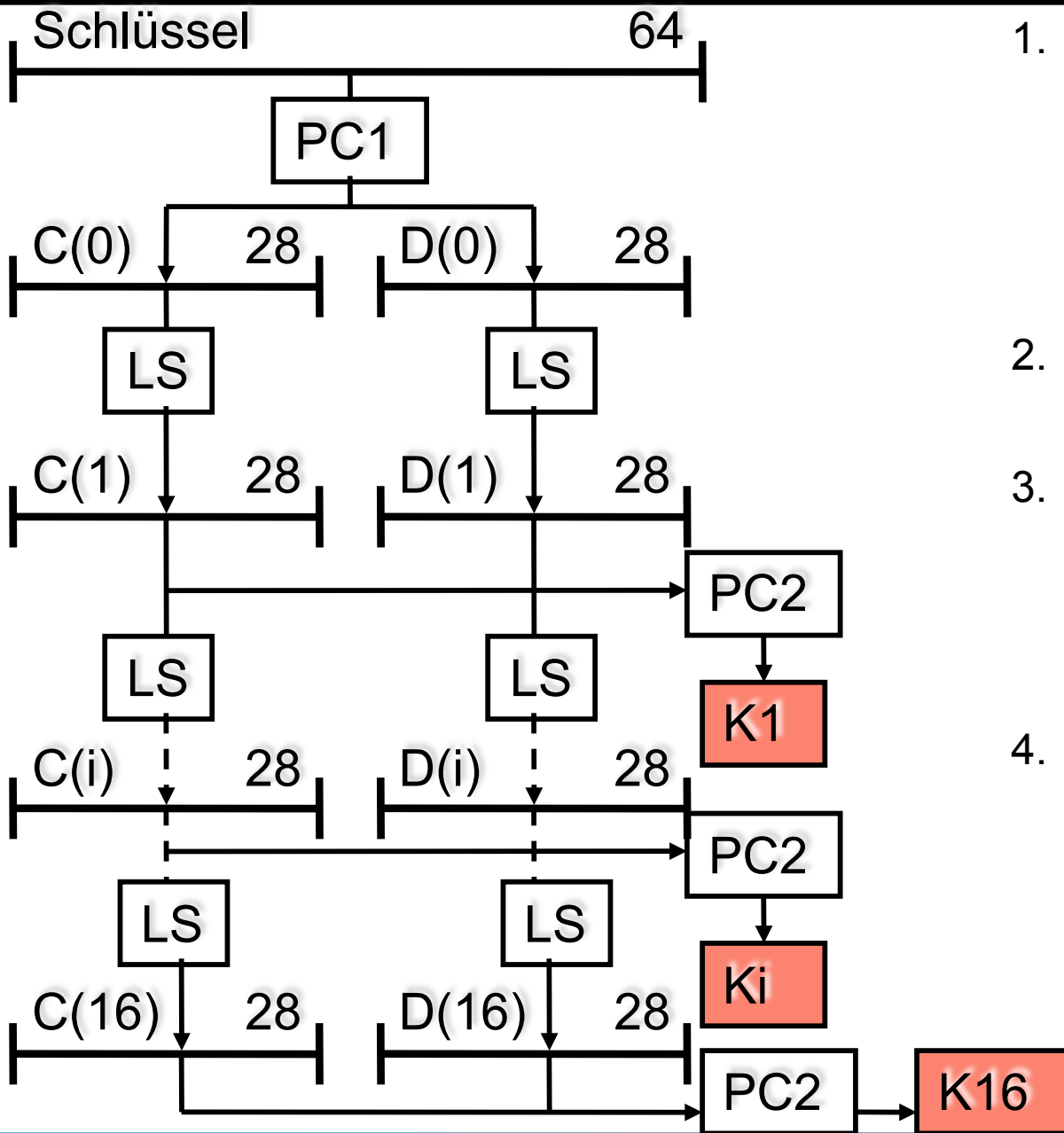
- 6 Bit Input Block (i1,i2,i3,i4,i5,i6) wird auf 4 Bit Outputblock (o1,o2,o3,o4) abgebildet:
 - Redundante Bits (i1,i6) des Inputblocks bestimmen die **Zeile** der entspr. S-Box
 - Bits (i2,i3,i4,i5) bestimmen **Spalte**
 - Element in der Matrix bestimmt Wert des Outputblocks
- Bsp. S-Box S1:
 - Design der S-Boxen (IBM, NSA) ist ausschlaggebend für die Sicherheit des Verfahrens.
 - Beispiel
 - S-Box S1
 - Input (0,1,1,0,1,1)
 - Zeile (0,1) = 1
 - Spalte (1,1,0,1) = 13
 - Output = 5 = (0,1,0,1)

(i1,i6) \ (i2,i3,i4,i5)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

DES Funktion: Grundschemata

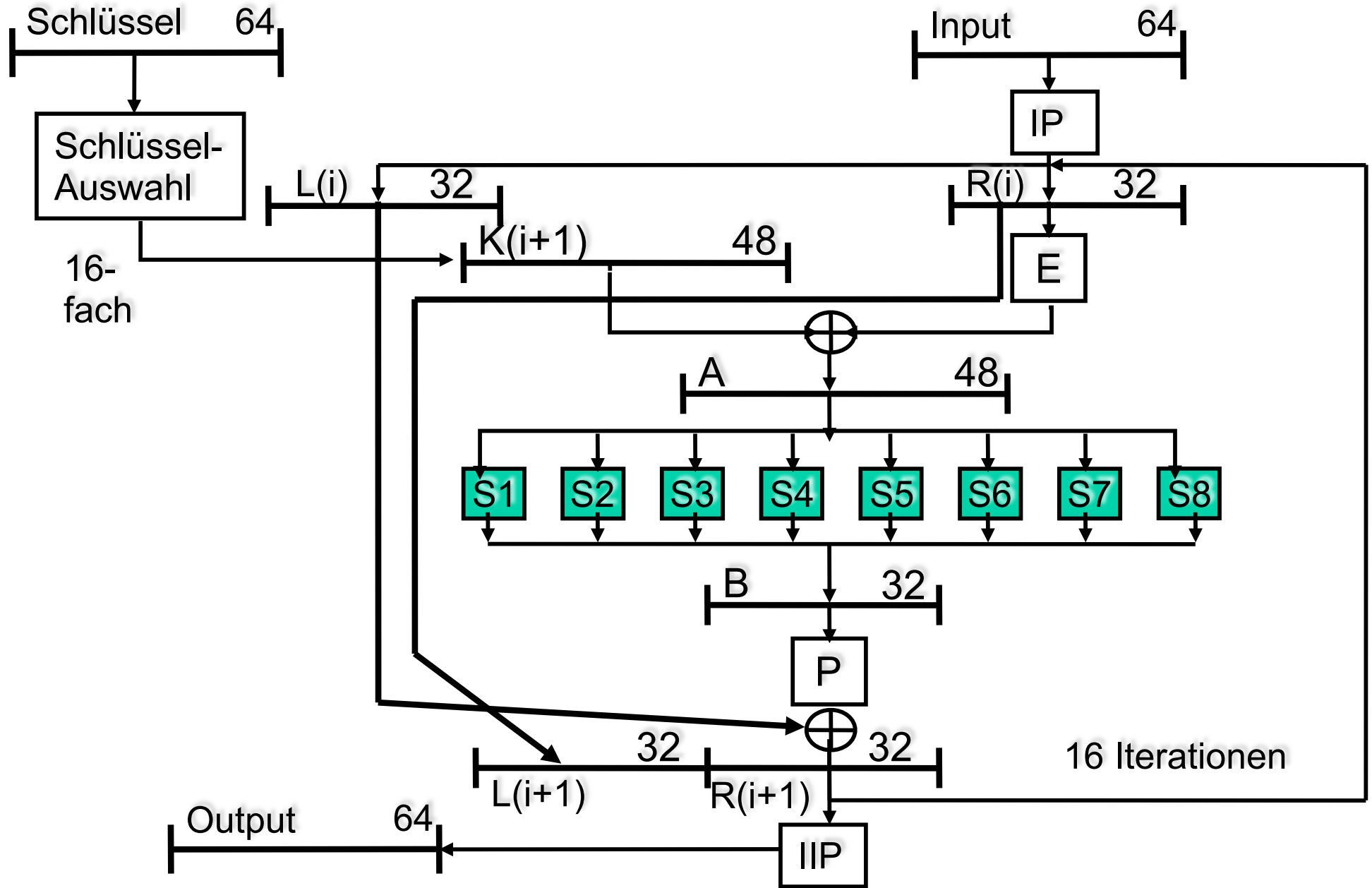


DES Schlüsselauswahl

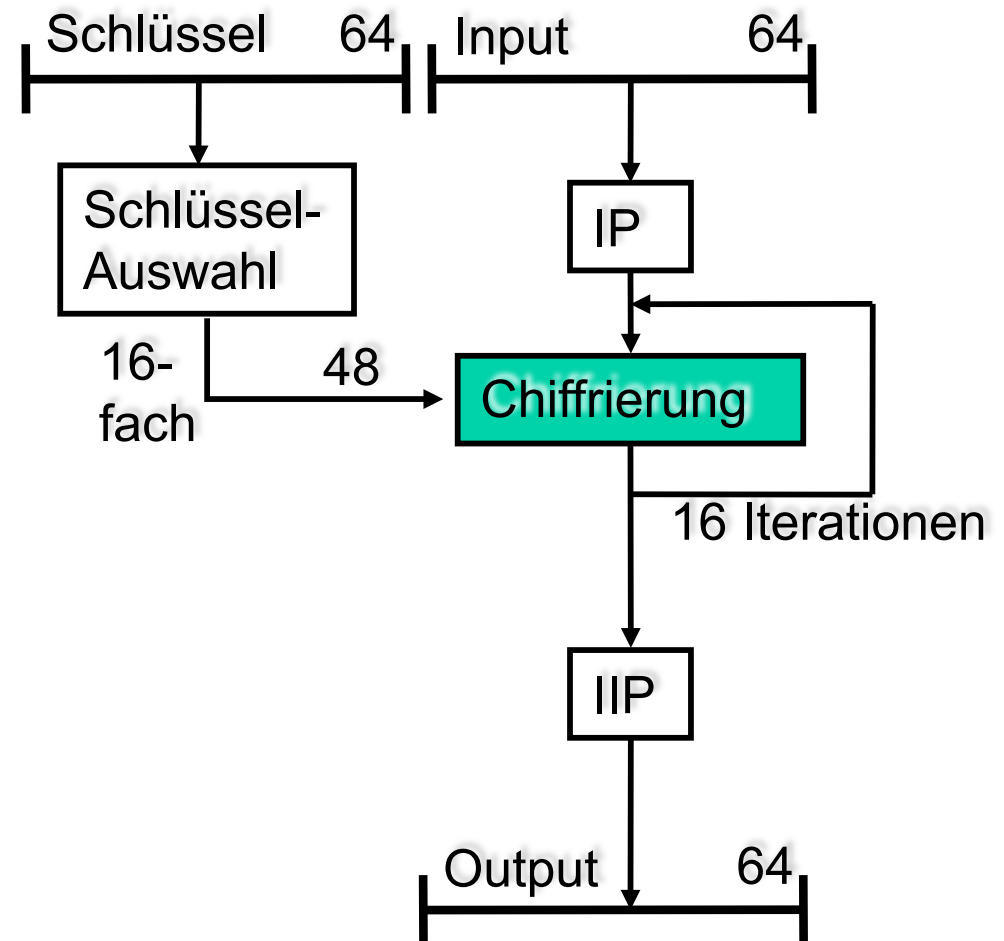


- 64 Bit Schlüssel wird Permuted Choice 1 (PC1) unterworfen:
 - Key wird auf 56 relevante Bits gekürzt (jedes 8. Bit Parity)
 - Key wird permutiert
- Schlüssel wird in zwei Teile C(i) und D(i) zu je 28 Bit aufgeteilt
- Blöcke werden zyklisch nach links geschifft
 - In Runden 1,2,9 u. 16 um 1 Bit
 - In allen anderen Runden um 2 Bit
- Teilblöcke werden zusammengefasst und PC2 unterworfen:
 - Entfernen der Bits 9,18,22,25; 35,38,43 u. 56
 - Permutation der verbleibenden 48 Bit

DES: Zusammenfassung



- DES wird für Ver- und Entschlüsselung prinzipiell gleich verwendet, außer
 - Umkehrung der Schlüsselreihenfolge
 - D.h. in Runde i wird $K(16-i)$ verwendet



- **Starker Avalanche-Effekt**
(Lawineneffekt; große Streuung)
durch S-Boxen und Permutation P:
Kleine Änderungen in der Eingabe,
die nur eine S-Box betreffen, breiten
sich schnell aus.
Eine Änderung eines Bits in der
Eingabe verursacht eine Änderung
von durchschnittlich 50% der
Ausgabe.
- 16 Iterationen:
Known-plaintext Angriff auf DES mit
< 16 Runden immer effizienter als
Brute force
- Stark gegen analytische Angriffe:
Differentielle Kryptoanalyse braucht
 2^{58} Operationen.
- ⚡ (teilweise) geheimes Design
- ⚡ Deutlich zu geringe Schlüssellänge:
Schlüsselraum der Größe 2^{56}
- ⚡ 4 schwache Schlüssel mit:
 $\text{DES}(\text{DES}(x,K),K) = x$
- ⚡ 6 semi-schwache Schlüsselpaare:
 $\text{DES}(\text{DES}(x,K),K') = x$
- ⚡ Optimiert auf Implementierung in
Hardware:
Initialpermutation IP und inverse IP
verbessern die Sicherheit nicht,
sondern erhöhen nur den Aufwand
für Software-Implementierungen.

DES Varianten: Double und Triple DES

■ Double-DES:

- $\text{DES}(\text{DES}(m, K1), K2)$

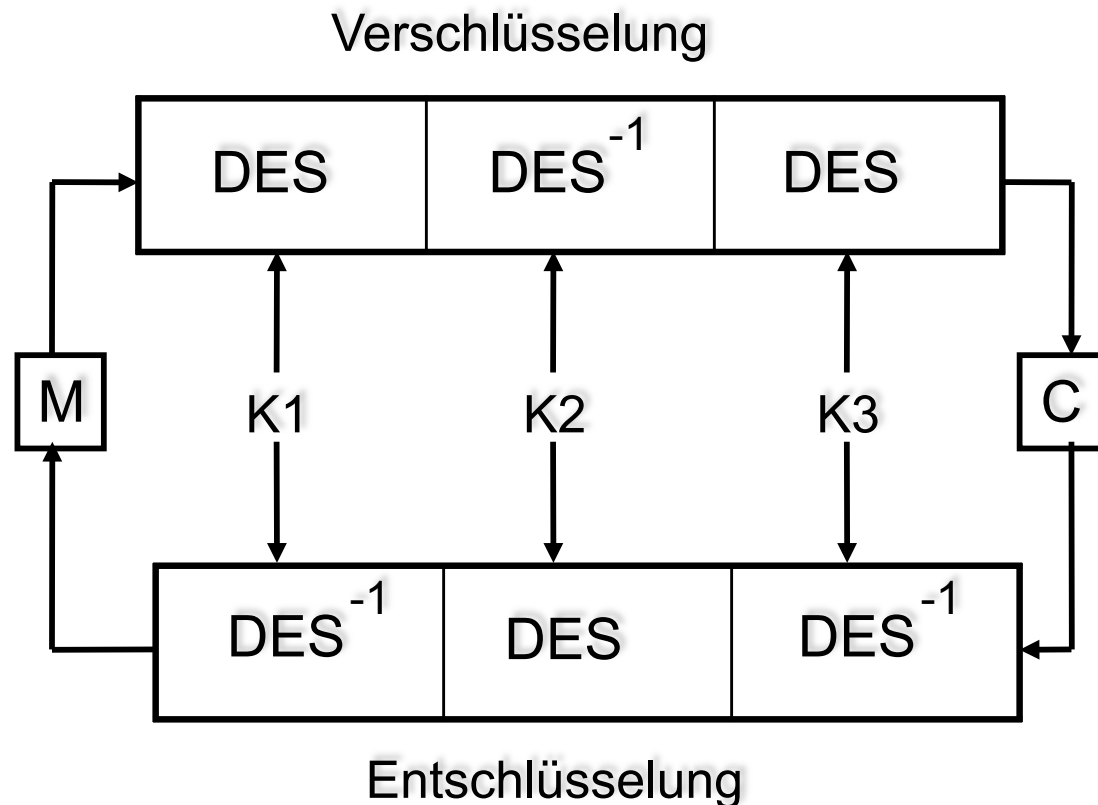
■ Erwartete Komplexität:

- bei Schlüssellänge n : 2^{2n}

■ Merkle und Hellman haben gezeigt, dass ein Known-Plaintext Angriff möglich ist mit Komplexität 2^{n+1}

■ D.h. doppelte Ausführung von DES bringt **KEINE** relevante Steigerung der Sicherheit!

■ Triple-DES (3DES)



- Schlüssellänge eigentlich 168 Bit
- Wegen Meet-in-the-Middle-Angriff effektiv aber nur 112 Bit

- Claude Shannon forderte bereits 1949:
 - **Konfusion**: Vom Chiffretext kann möglichst wenig auf den Klartext geschlossen werden.
 - **Diffusion**: Kleine Änderungen an der Eingabe bewirken große Änderungen an der Ausgabe.
- DES gehört zur Klasse der **Feistel-Chiffren**
 - Horst Feistel (1915-1990), arbeitete für IBM an DES mit
 - Bezeichnung für bijektive symmetrische Blockverschlüsselungsverfahren mit typischen Eigenschaften:
 - Zerlegung des Eingabeblocks in zwei Teile
 - n Runden mit verschiedenen Rundenschlüsseln
 - Funktion f muss nicht umkehrbar sein
 - Alternierende Substitutionen und Permutationen setzen Konfusion und Diffusion um (**Avalanche**-Effekt nach Feistel).
 - Iterationen und zueinander ähnliche Ver-/Entschlüsselung ermöglichen günstige Hardwareimplementierungen.

■ **Polizeifunk** (Sondereinheiten, Verfassungsschutz)

- ❑ Sprechfunkgeräte von Motorola
- ❑ Neuer Schlüssel für jeden Einsatz / nach mehreren Stunden
 - Dezentral über Key Variable Loader oder
 - zentral über Key Management Centre per Over-the-Air-Rekeying
- ❑ Spätere Entschlüsselung (nach Einsatzende) ist irrelevant

■ **Geldautomaten**

- ❑ Geheimzahl wird bereits in der Tastatur verschlüsselt und
- ❑ zusammen mit Kontonummer, Bankleitzahl, ...
- ❑ an einen Server der kontoführenden Bank geschickt.
- ❑ Dort wird die PIN entschlüsselt und überprüft.

■ Blockchiffren (Beispiel: DES)

- Erwartet Eingabe fester Blocklänge n (meist 64 oder 128 Bit)
- Nachricht m der Länge $|m|$ wird in r Blöcke der Blocklänge n zerlegt
- Letzter Block hat Länge $1 \leq k \leq n$
- Falls $k < n$: Auffüllen mit sog. Padding
- Länge des Padding muss geeignet hinterlegt werden
- Ciphertext ergibt sich durch Konkatination der Output-Blöcke

■ Stromchiffren (Beispiel: RC4 bei WEP-WLAN-Verschlüsselung)

- Verschlüsseln kleine Klartext-Einheiten, z.B. 1 Bit oder 1 Byte
- Klartext-Einheit wird mit einem frischen Zeichen aus dem sog. Keystream XOR-verknüpft
- Keystream wird von Pseudo-Zufallszahlen-Generator (PRNG) erzeugt
- PRNG wird von Absender und Empfänger mit Shared Secret initialisiert

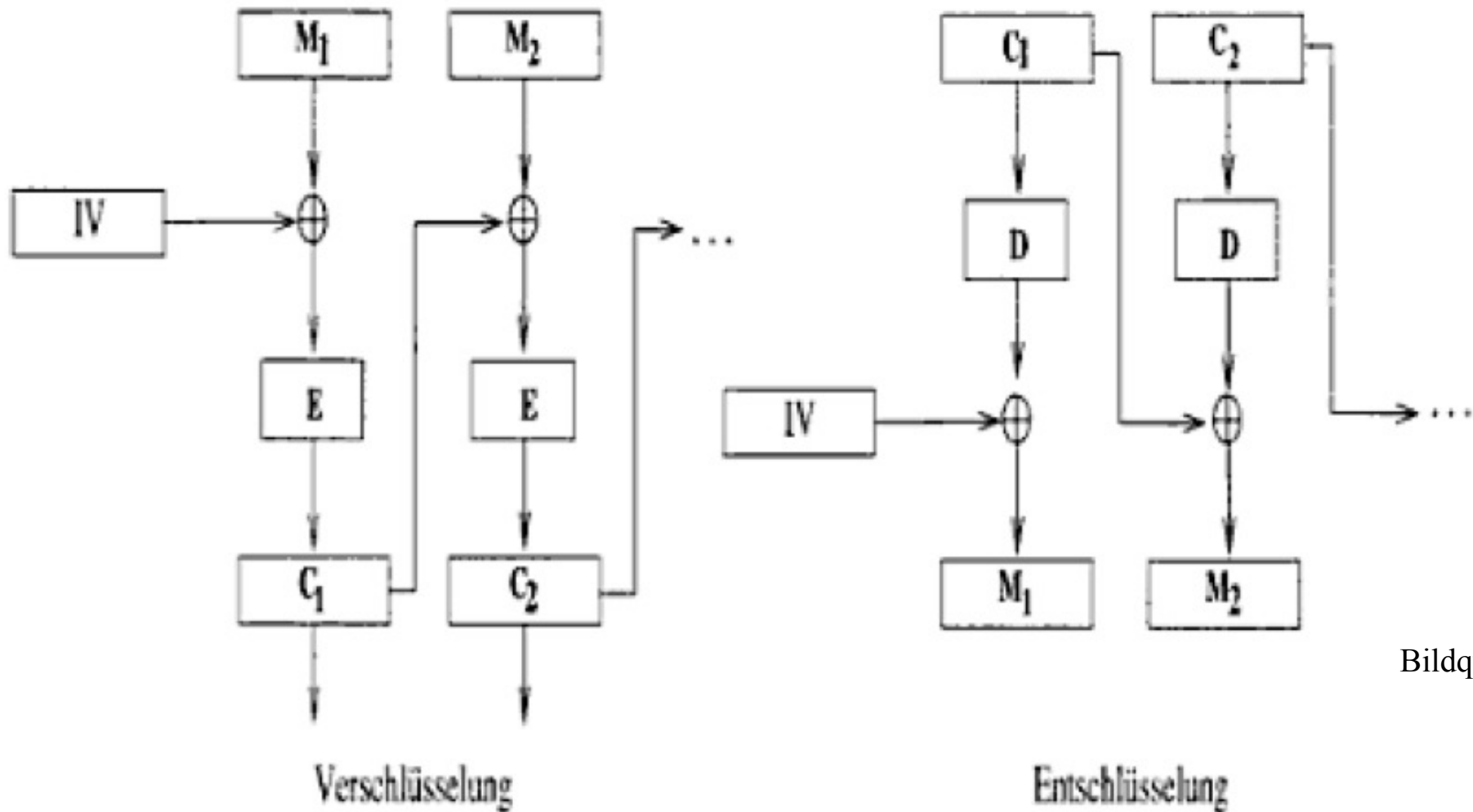
■ Electronic Codebook Mode (ECB)

- ❑ Jeder Klartext-Block wird einzeln mit demselben Schlüssel verschlüsselt.
- ❑ **Identische Klartext-Blöcke liefern somit identische Ciphertext-Blöcke.**
- ❑ Erleichtert Angriffe, z.B.
 - Vertauschen/Löschen/Wiedereinspielen von Ciphertext-Nachrichten fällt nicht sofort beim Entschlüsseln auf.
 - Rückschlüsse auf den Klartext aufgrund statistischer Eigenschaften.
- ❑ Einfach zu implementieren, aber nur für kurze Nachrichten geeignet (vgl. Kritik an „Staatstrojaner“).

■ Cipher Block Chaining (CBC)

- ❑ Jeder Klartext-Block wird vor der Verschlüsselung mit dem vorhergehenden Ciphertext-Block XOR-verknüpft.
- ❑ Benötigt einen **Initialisierungsvektor (IV)** für die XOR-Verknüpfung des ersten Klartext-Blocks.
- ❑ Beseitigt die Defizite des ECB-Modes; aber: Kein wahlfreier Zugriff.

Cipher Block Chaining (CBC-Modus)



Bildquelle: [Eckert]

■ Fortpflanzung von Übertragungsfehlern?

■ Symmetrische Kryptosysteme

- Data Encryption Standard (DES)

- Advanced Encryption Standard (AES)

■ Kryptoregulierung

- 1997 öffentliche Ausschreibung des Dept. Of Commerce (Request for Candidate Algorithms for AES):
 - Algorithmus öffentlich und nicht klassifiziert
 - Mindestblocklänge 128 Bit, Schlüssellängen 128, 192 und 256 Bit
 - Weltweit frei von Lizenzgebühren
 - Nutzbar für 30 Jahre, effizient sowohl in SW als auch versch. HW
- Dreistufiges (Vor-)Auswahlverfahren
 1. Pre-Round 1 (1/97 – 7/98)
 - Call for Candidates
 2. Round 1 (8/98 – 4/99)
 - Vorstellung, Analyse und Test
 - Auswahl der Kandidaten für Round 2
 3. Round 2 (8/99 – 5/2000)
 - Analyse und Tests
 - Auswahl der Finalisten
- Endgültige Auswahl durch NIST

AES Kandidaten

- Pre-Round 1: 21 Kandidaten, 6 aus formalen Gründen abgelehnt

Algo.	Land	Autor(en)	Algo.	Land	Autor(en)
CAST-256	Kanada	Entrust	MAGENTA	Deutschland	Deutsche Telekom
CRYPTON	Korea	Future Systems	MARS	USA	IBM
DEAL	Kanada	R. Outbridge, L. Knudsen	RC6	USA	RSA Laboratories
DFC	Frankreich	CNSR	RIJNDAEL	Belgien	J. Daeman, V. Rijmen
E2	Japan	NTT	SAFER+	USA	Cylink
FROG	Costa Rica	TecApro	SERPENT	UK, Norwegen, Israel	R. Anderson, E. Biham u.a.
HPC	USA	R.Schroepfel	TWOFISH	USA	B. Schneier, J. Kelsey, u.a.
LOKI97	Australien	L. Brown, J. Pieprzyk u.a.			

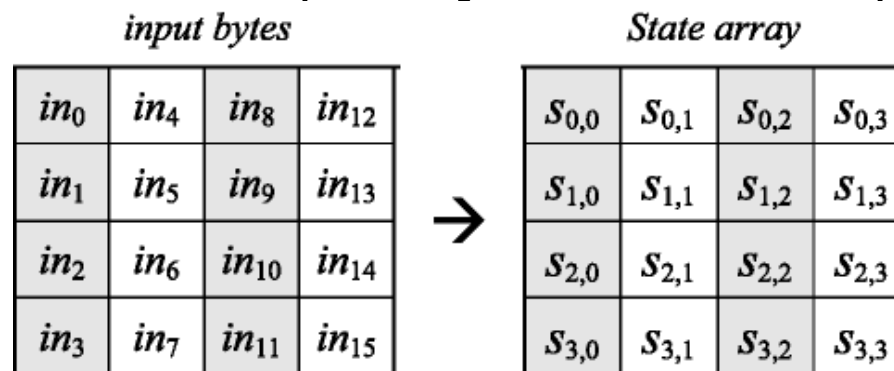
AES: Round 2 Finalisten und Ergebnis

Finalisten der Runde 2:

MARS	USA	IBM
RC6	USA	RSA Laboratories
RIJNDAEL	Belgien	J. Daeman, V. Rijmen
SERPENT	UK, Norwegen, Israel	R. Anderson, E. Biham, L. Knudsen
TWOFISH	USA	B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson

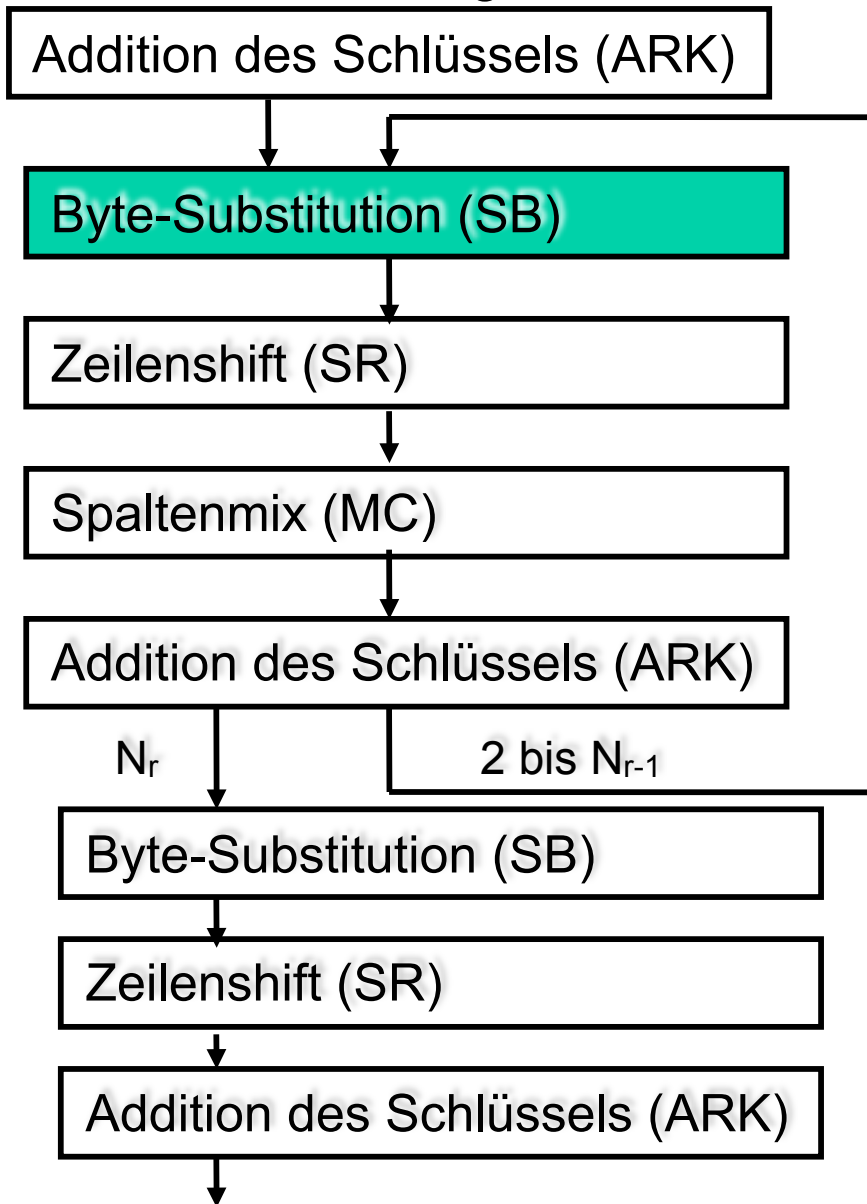
- 2. Oktober 2000: Rijndael wird gewählt
- 26. Nov. 2001: Veröffentlichung des FIPS-197 (Federal Information Processing Std.) durch NIST (National Institute for Standards and Technology)
- 26. Mai 2002: Inkrafttreten des Standards
- Informationen: www.nist.gov/aes mit Link auf AES-Homepage

- **Variable Blocklänge:** $32 \cdot N_b$ Bits
- **Variable Schlüssellänge:** $32 \cdot N_k$ Bits
- N_b und N_k aus $[4;8]$; im Standard eingeschränkt auf 4, 6 oder 8
- **Abgeleitete Runden-Anzahl** $N_r = \max(N_b, N_k) + 6$
- Folgende Beispiele für $N_b=N_k=4$
(Block- und Schlüssellänge 128 Bits; 10 Runden)
- Rijndael arbeitet auf sog. **States**:
Input-Bytes $in_0, in_1, \dots, in_{15}$ (16 Bytes=128 Bits) werden in den State kopiert:



- Runden arbeiten auf dem State

■ Verschlüsselung



■ Runden arbeiten auf sog. States

■ Verschlüsselung:

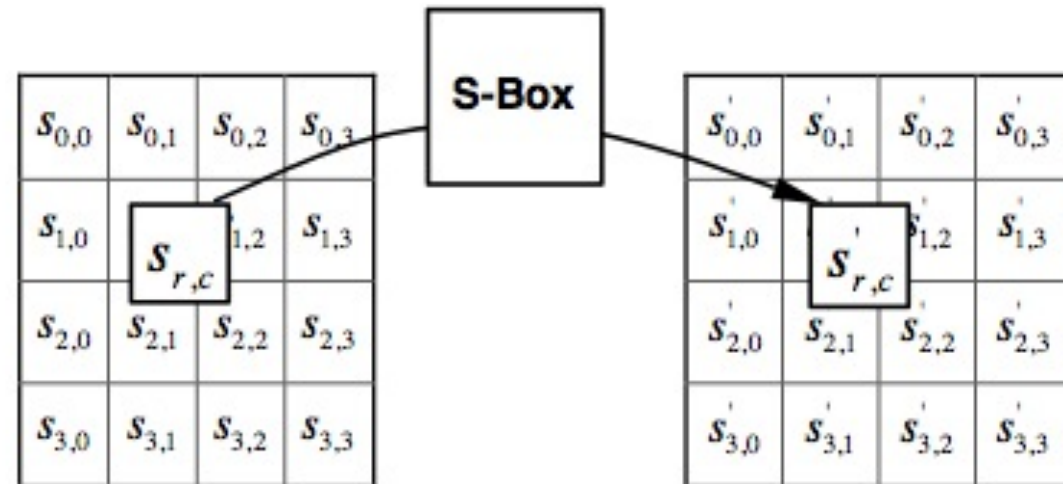
- Ablauf der Runden 1 bis N_{r-1} :
 1. Byte-Substitution ($SubBytes$, SB)
 2. Zeilenshift ($ShiftRows$, SR)
 3. Spaltenmix ($MixColumns$, MC)
 4. Addition des Rundenschlüssels ($AddRoundKey$, ARK)

■ Entschlüsselung:

- Runde 1 bis N_{r-1} :
 1. Inverser Zeilenshift
 2. Inverse Byte-Substitution
 3. Addition des Rundenschlüssels
 4. Inverser Spaltenmix

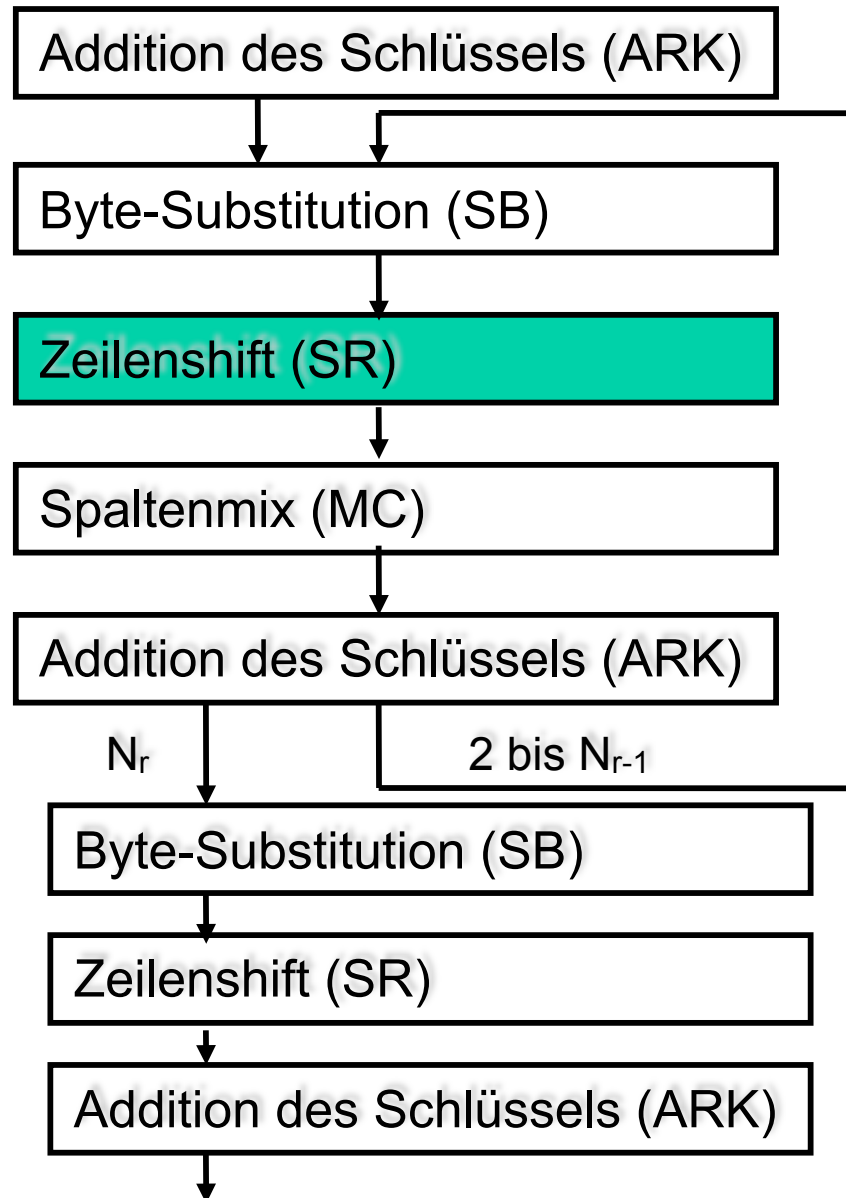
■ Letzte Runden N_r analog, aber **ohne** (inversen) Spaltenmix

- Rijndael S-Box (aus FIPS 197)
- Eingabe 53 wird zu Ausgabe ed

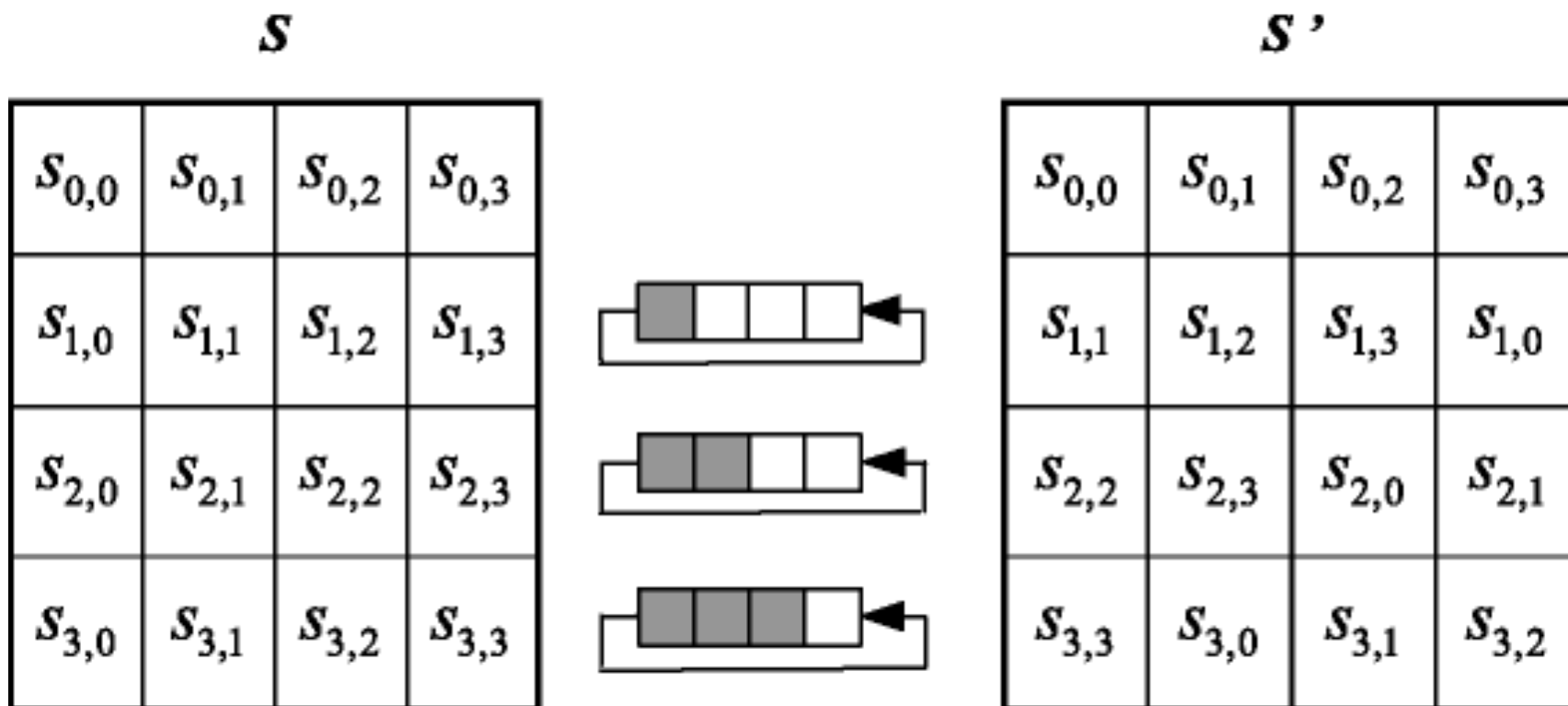


		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3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

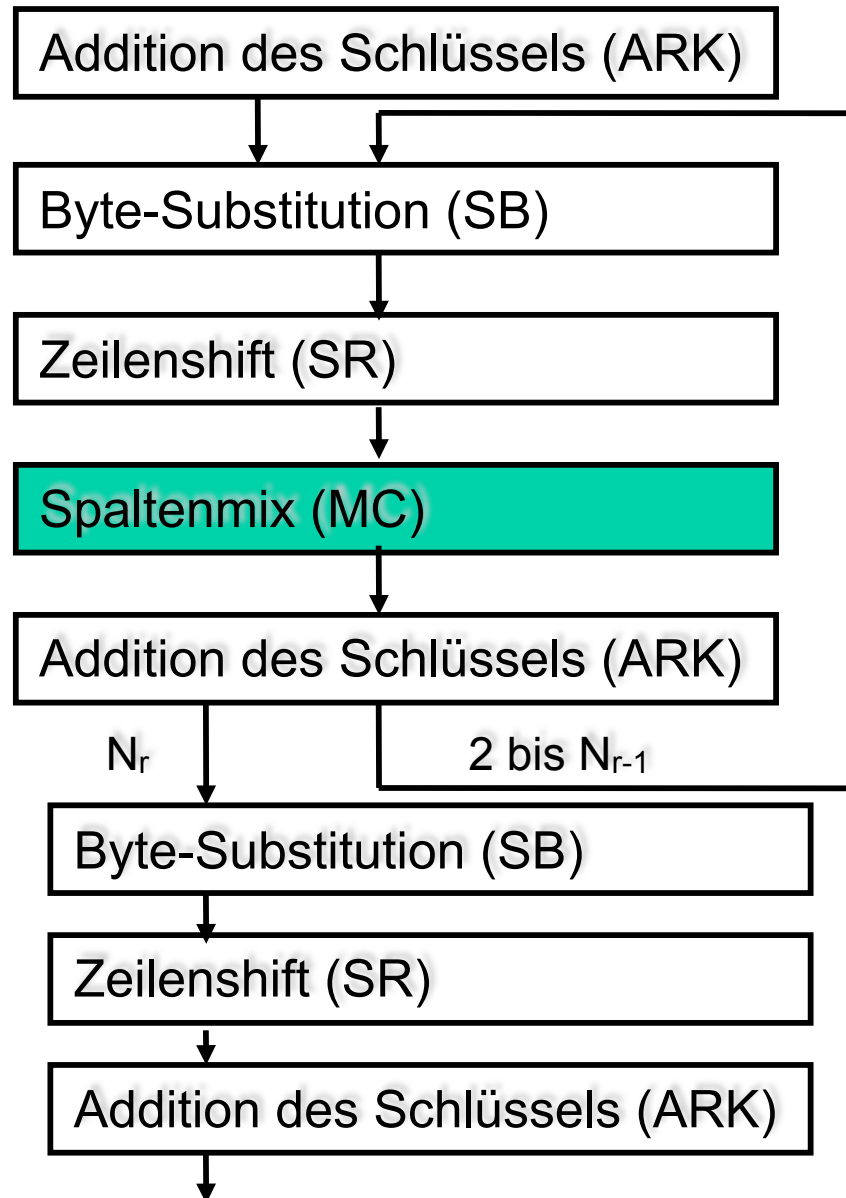
AES: Ver- und Entschlüsselung



- Zyklischer Shift der letzten drei Zeilen des State:
 - Zeile 1 bleibt unverändert
 - Zeile 2 um 1 Byte
 - Zeile 3 um 2 Byte
 - Zeile 4 um 3 Byte



AES: Ver- und Entschlüsselung



- Addition (= Subtraktion) modulo 2 = stellenweise XOR-Verknüpfung \oplus ; Beispiel:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \quad (\text{polynomial notation});$$

$$\{01010111\} \oplus \{10000011\} = \{11010100\} \quad (\text{binary notation});$$

$$\{57\} \oplus \{83\} = \{d4\} \quad (\text{hexadecimal notation}).$$

- Multiplikation \bullet in $GF(2^8)$ entspricht Polynommultiplikation modulo irreduziblem (nur durch 1 oder sich selbst teilbar)

Polynom vom Grad 8. Für AES: $m(x) = x^8 + x^4 + x^3 + x + 1$; Beispiel:

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + \quad \{57\} \bullet \{83\} = \{c1\}$$

$$x^7 + x^5 + x^3 + x^2 + x +$$

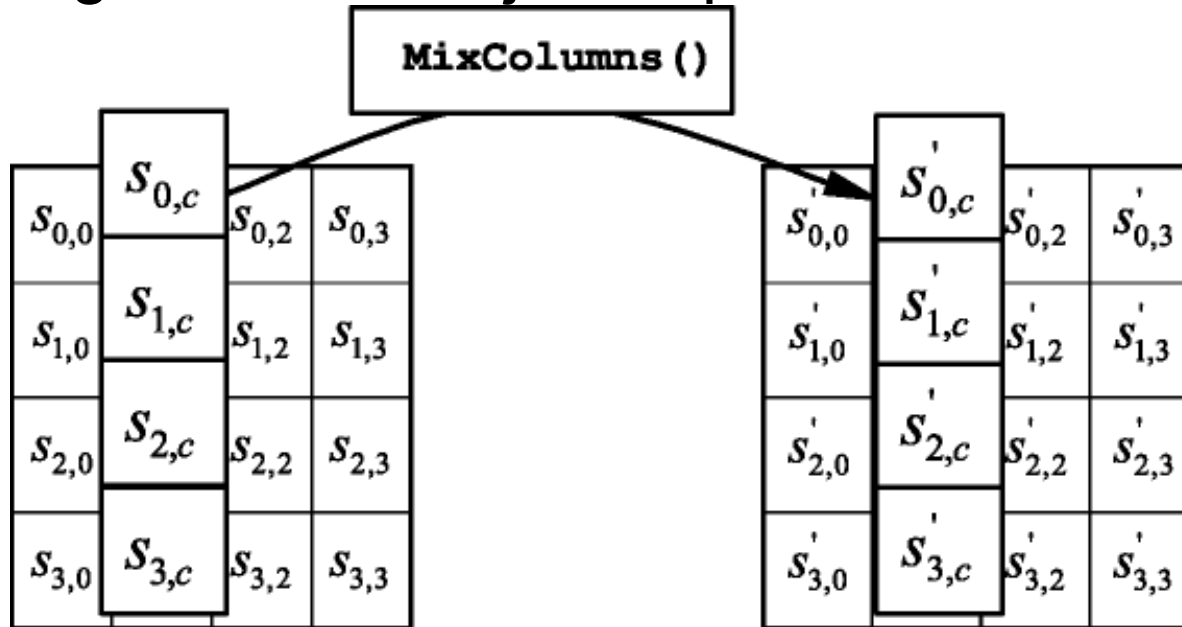
$$x^6 + x^4 + x^2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)$$

$$= x^7 + x^6 + 1.$$

- Angewendet auf jede Spalte des State



- Jede Spalte wird als Polynom vom Grad 3 mit Koeffizienten aus $GF(2^8)$ aufgefasst:
 - Multiplikation mit dem festen Polynom $a(x)$ modulo x^4+1

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

- Darstellbar als Matrizenmultiplikation:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{für } 0 \leq c < Nb.$$

Ausmultipliziert:

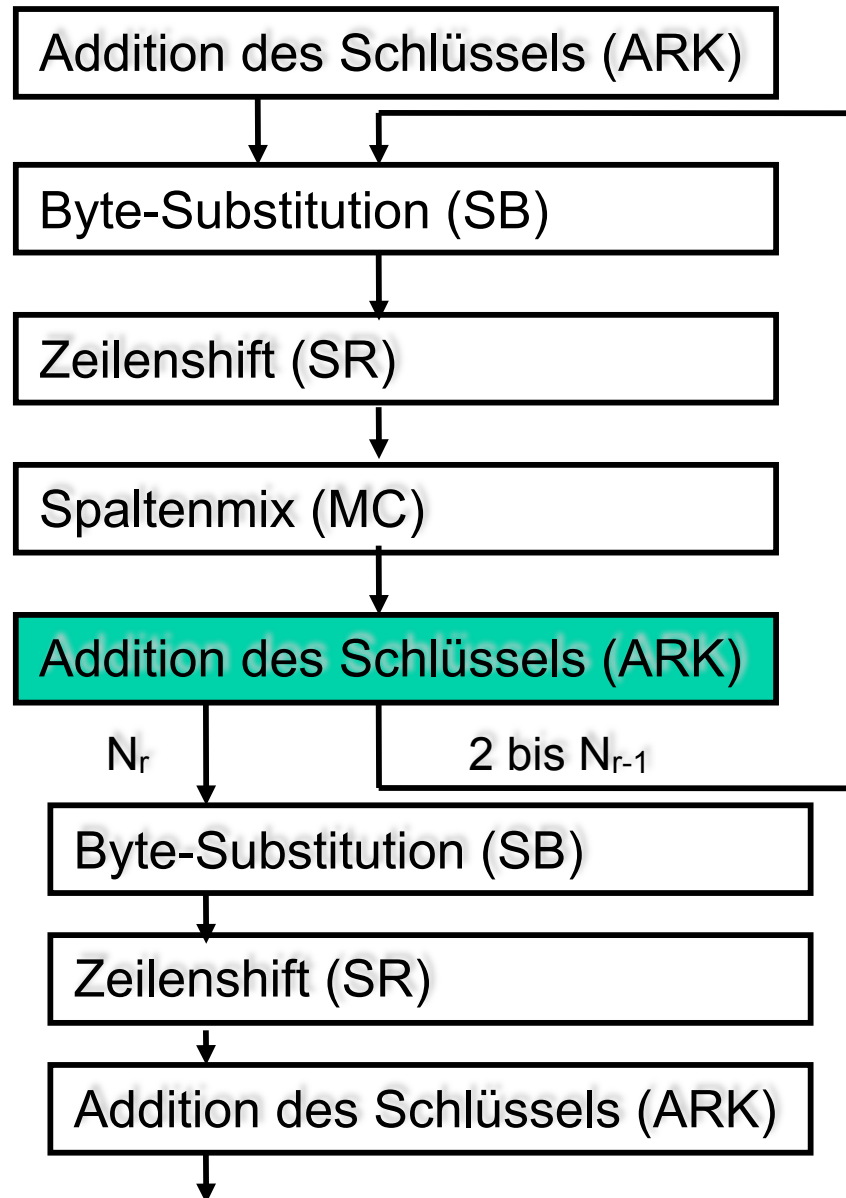
$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

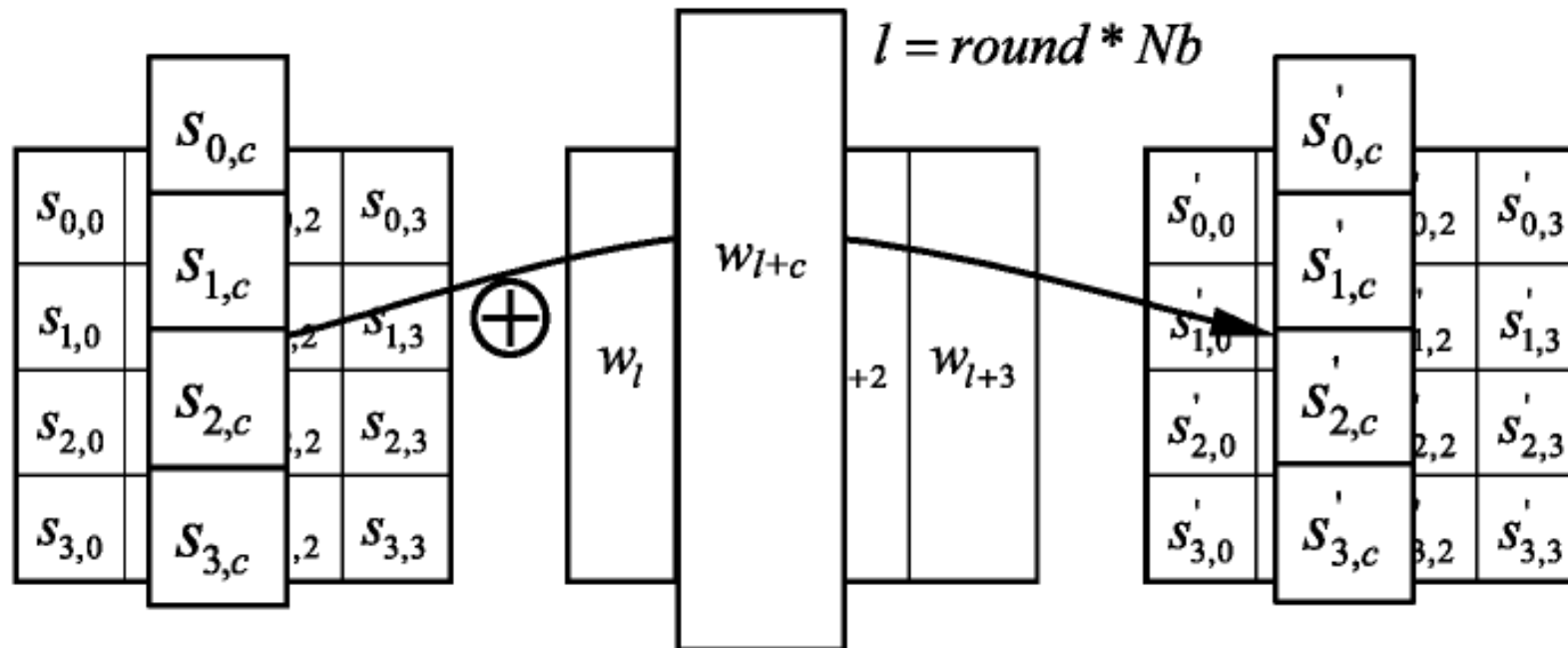
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

AES: Ver- und Entschlüsselung



- Funktion `AddRoundKey()`
- Jede Spalte des State wird mit einem „Wort“ des Rundenschlüssels XOR-verknüpft



AES: Bestimmung des Rundenschlüssels

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp

    i = 0

    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while

    i = Nk

    while (i < Nb * (Nr+1))
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end
```

- Schlüssel k besteht aus $32 * N_k$ Bits bzw. $4 * N_k$ Bytes
- Ein Wort $W[i]$ besteht aus 4 Bytes
- $W[0]$ sind die ersten 4 Bytes des Schlüssels, $W[1]$ die zweiten 4 Bytes,, $W[N_k-1]$ die letzten 4 Bytes
- Insgesamt müssen $N_b * (N_r + 1)$ Wörter berechnet werden
- Die ersten N_k Wörter entsprechen dem vom Anwender gewählten Schlüssel
- Wort $W[i]$ entspricht $W[i-1] \text{ XOR } W[i-N_k]$
- Falls $i \bmod N_k == 0$:
 - SubWord() wendet die S-Box auf ein Wort an
 - RotWord() verwandelt $a_0a_1a_2a_3$ in $a_1a_2a_3a_0$
 - Rcon[i] entspricht vordefinierten Rundenkonstanten

Ablauf Verschlüsselung

```

Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state, w[0, Nb-1])           // See Sec. 5

  for round = 1 step 1 to Nr-1
    SubBytes(state)                       // See Sec. 5
    ShiftRows(state)                      // See Sec. 5
    MixColumns(state)                    // See Sec. 5
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
  end for

  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

  out = state
end

```

Ablauf Entschlüsselung

```

InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5

  for round = Nr-1 step -1 downto 1
    InvShiftRows(state)                  // See Sec. 5
    InvSubBytes(state)                  // See Sec. 5
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    InvMixColumns(state)                // See Sec. 5
  end for

  InvShiftRows(state)
  InvSubBytes(state)
  AddRoundKey(state, w[0, Nb-1])

  out = state
end

```

- Design-Kriterien mussten offen gelegt werden
- Abschätzung und Stellungnahme zur Widerstandsfähigkeit gegen bekannte Angriffe

- Schlüsselauswahl mit nichtlinearer Durchmischung wegen Verwendung der S-Box;
damit widerstandsfähig gegen folgende Angriffe:
 - Kryptanalyst kennt Teile des Schlüssels und versucht, den Rest zu berechnen.
 - Zwei ähnliche Schlüssel haben **keine** große Zahl von gemeinsamen Rundenschlüsseln.
 - **Rundenkonstante verhindert Symmetrien** im Verschlüsselungsprozess; jede Runde ist anders.

- **Keine Feistel-Chiffre**, sondern deutlich höhere Diffusion: nach 2 Runden hängen 50% Output-Bits von jedem Input-Bit ab.
- Algebraische S-Box-Konstruktion; offengelegt; in hohem Maße nichtlinear.
- Damit **stabil gegen lineare und differentielle Kryptoanalyse**.
- `ShiftRow` wurde eingefügt, um zwei neue Angriffsarten zu verhindern (truncated differentials und Square attack).
- `MixColumn` für hohe Diffusion; Änderung in einem Input-Byte verursacht Änderung in allen Output-Bytes
- Auswahl von 10 Runden:
Bei AES-128 mit bis zu 7 Runden sind Angriffe bekannt, die besser sind als Brute Force. Bei mehr als 7 Runden sind keine solchen Angriffe bekannt. D.h. 3 Runden „Reserve“, die zudem sehr leicht erweitert werden können.

- Aufgrund von Standardisierung und Qualität sehr weit verbreitet
- Beispiele:
 - In der Vorlesung behandelte Protokolle:
 - WLAN-Verschlüsselung mit WPA2
 - Remote-Zugriff auf Rechner mit SSH
 - Verschlüsselung auf OSI-Schicht 3: IPsec
 - Weitere Protokolle und Produkte:
 - Festplattenverschlüsselung z.B. mit Apple FileVault, Windows EFS, TrueCrypt
 - Skype
 - Kompressions-/Archivierungsprogramme (ZIP, RAR, ...)
 - viele viele mehr...

- Recherchen im Heise-Verlag 12/2008
- Hersteller bewirbt Festplatte mit Hardware-AES-Verschlüsselung.
- In Wirklichkeit wird jeder Sektor der Festplatte mit demselben 512-Byte-Block XOR-verschlüsselt.
- Triviale Rekonstruktion des 512-Byte-Schlüssels möglich: *„Aufschrauben des Gehäuses dauert länger als Knacken der Verschlüsselung.“*



- <http://www.heise.de/security/artikel/Verschusselt-statt-verschluesselt-270058.html>

■ Symmetrische Kryptosysteme

- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

■ Kryptoregulierung

- **Gesetzliche Beschränkung** der Nutzung kryptographischer Verfahren
 - (Offizielle) Motivation: Verbrechensbekämpfung
 - Ganz verbieten würde zu wirtschaftlichen Nachteilen führen, deshalb: Schlüssel hinterlegung (*key escrow*)

- Häufig genannte **Gegenargumente**:
 - Zentral hinterlegte Schlüssel sind attraktives Angriffsziel
 - Arbeitsgrundlage u.a. für Ärzte, Journalisten, ...
 - Verbindlichkeit elektronischer Signaturen würde in Frage gestellt
 - In Deutschland: Verfassungsrechtliche Bedenken - Grundrechte auf
 - (wirtschaftliche) Entfaltungsfreiheit (aus Art. 12 Abs. 1 GG)
 - Vertraulichkeit der Kommunikation (aus Art. 10 GG)
 - informationelle Selbstbestimmung (aus Art. 2 Abs. 1 GG)

■ OECD-Richtlinien

- ❑ empfehlen unbeschränkte Entwicklung und Nutzung kryptographischer Produkte und Dienste;
- ❑ lehnen Key-escrow-Verfahren ab.

■ Wassenaar-Gruppe:

- ❑ Abkommen von 1998 regelt Exportbeschränkungen für dual-use goods (hier: militärisch und zivil nutzbare Güter) in 33 Ländern.
- ❑ Einschränkungen für Hard-/Softwareprodukte mit Schlüssellänge ab 56 Bits.
- ❑ **Ausnahmen: Verfahren für elektronische Signaturen und Authentifizierung.**
- ❑ **Jedes Land entscheidet selbst, welche Produkte exportiert werden dürfen.**
 - EU: Keine Exportbeschränkungen für Produkte des Massenmarkts.
 - USA:
 - bis 1998: Exportverbot ab Schlüssellänge > 40 Bits
 - 1998 - 2000: Freier Export in 45 Länder, u.a. Deutschland
 - seit 2000: Nur noch Begutachtungsprozess bei Schlüssellänge >64 Bits

- Entwicklung, Herstellung, Vermarktung und Nutzung von Verschlüsselungsverfahren *innerhalb von Deutschland* ohne Restriktionen.
- **Export** von Verschlüsselungstechnik ist prinzipiell **genehmigungspflichtig**.
 - Vorgehen:
 - Außenwirtschaftsverordnung fordert Antrag auf individuelle **Ausfuhrgenehmigung beim Bundesausfuhramt (BAFA)**.
 - Abstimmung dieser Anträge mit dem BSI.
 - Ausschlaggebend sind Empfänger und Zweck.
 - **Ausnahmen**:
 - Keine Exportrestriktionen innerhalb der Europäischen Union.
 - Keine Exportkontrolle bei elektronischen Signaturen und Authentifizierungsverfahren für die Anwendungsbereiche Banking, Pay-TV, Copyright-Schutz und schnurlose Telefone (ohne Ende-zu-Ende-Verschlüsselung).

- US Department of Commerce, Bureau of Industry and Security verhängt \$ 750.000 Geldstrafe gegen Wind River Systems (Intel).
- Wind River Systems hatte ohne Exportgenehmigung ein Betriebssystem mit Kryptofunktionen u.a. an Kunden in China, Hong Kong, Russland, Israel, Südafrika und Südkorea geliefert.
- Erste Geldstrafe, bei der keine der in USA explizit sanktionierten Länder (u.a. Kuba, Iran, Nordkorea, Sudan, Syrien) involviert waren.
- = **Signalwirkung** auch für andere Hersteller