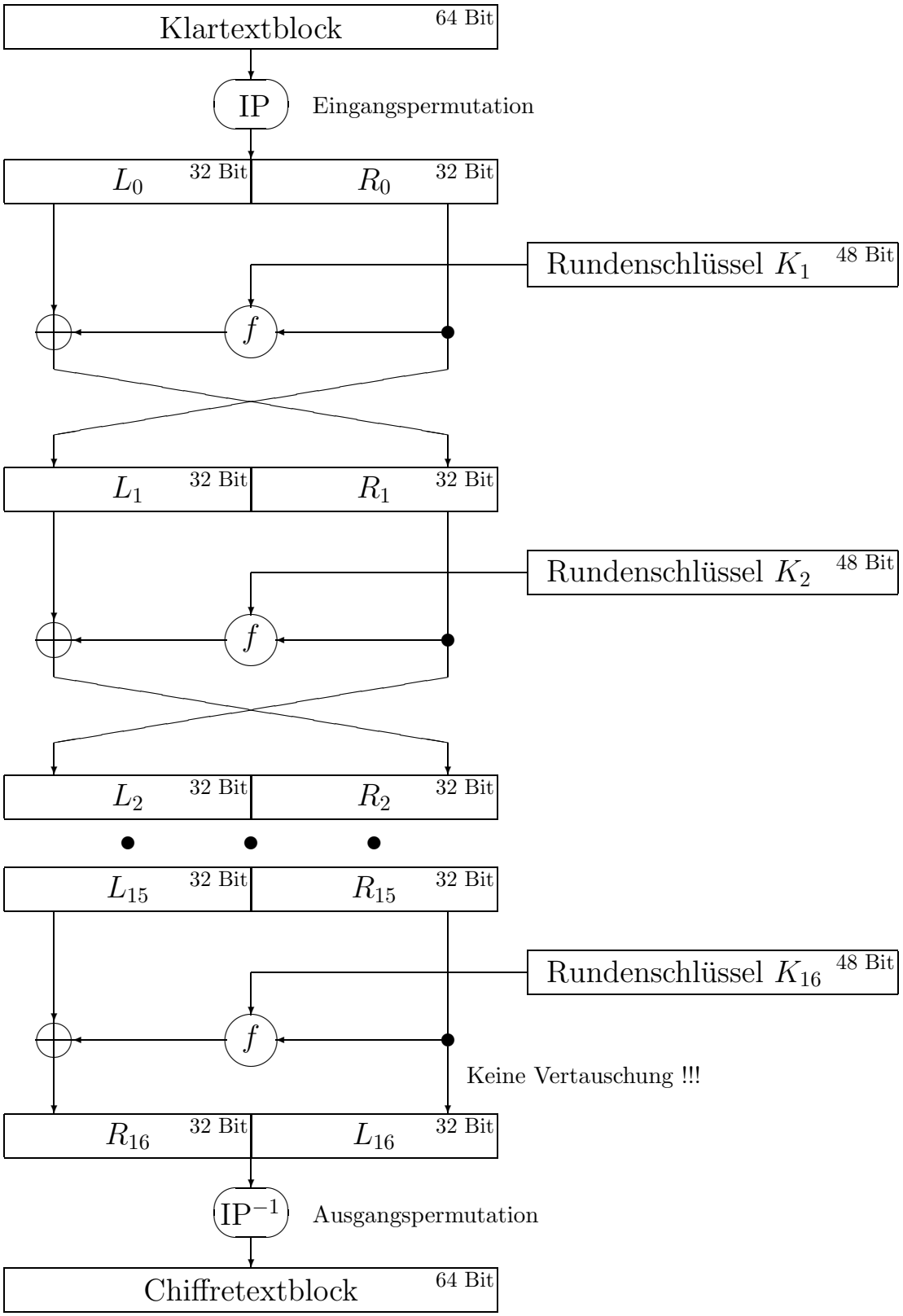
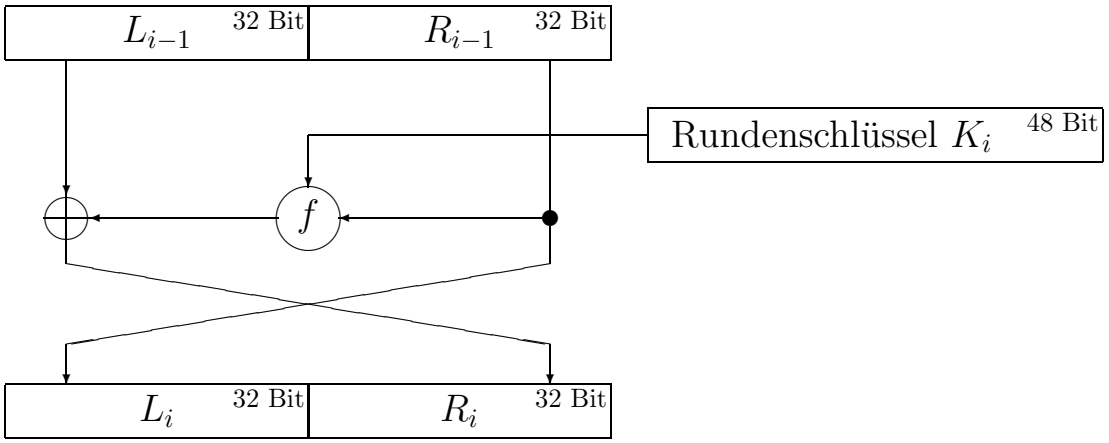
Der Aufbau von DES



Der DES-Algorithmus ist eine Feistel-Chiffre

Das DES-Verfahren beinhaltet 16 Runden des folgenden elementaren Algorithmus:



Es gilt also folgender Zusammenhang:

$$(*) \quad L_i = R_{i-1} \quad \text{und} \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \quad i = 1, \ldots, 16.$$

Chiffren, die aus j Runden mit der Eigenschaft (*) zusammengesetzt sind, heißen **Feistel-Chiffren** bzw. **Feistel-Netzwerke**.

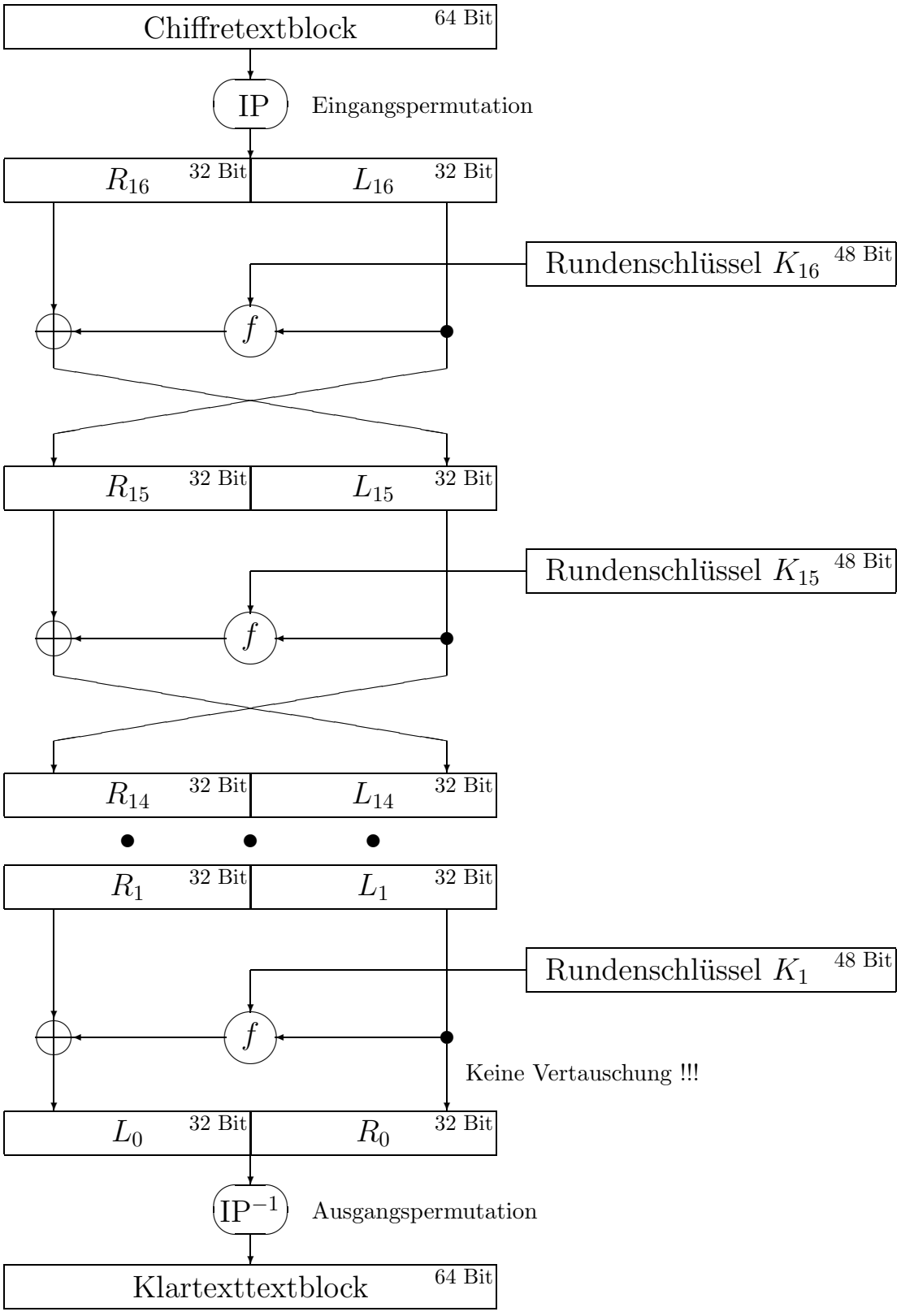
Feistel-Chiffren lassen sich einfach dechiffrieren, da aus (*) folgt:

$$\begin{aligned}(**) \quad R_{i-1} = L_i \quad \text{und} \quad L_{i-1} &= L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) \\ &= R_i \oplus f(R_{i-1}, K_i) \\ &= R_i \oplus f(L_i, K_i), \quad i = 1, \ldots, 16.\end{aligned}$$

Die Feistel-Chiffre wird also dechiffriert, indem man den Geheimtext mit umgekehrter Schlüsselfolge chiffriert (die Rollen der linken und rechten Hälfte sind dabei vertauscht).

Die Funktion f muß also nicht invertiert werden und kann unter kryptologischen Gesichtspunkten beliebig kompliziert konstruiert werden.

Die DES-Dechiffrierung



Die initiale Permutation IP und ihre Inverse IP^{-1}

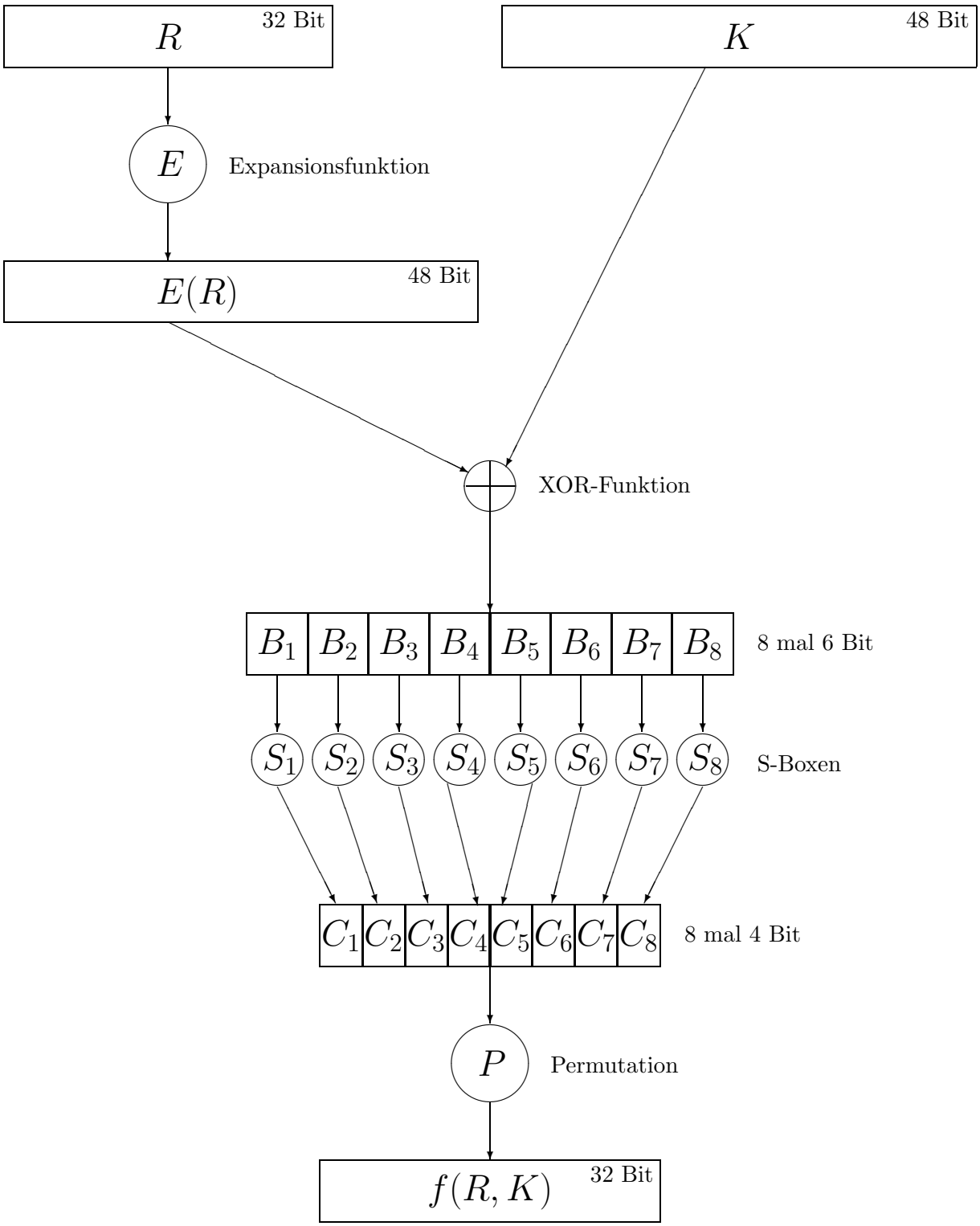
IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
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

Beispiel:

Ist $m \in \{0,1\}^{64}$, $m = m_1\, m_2\, m_3\, \ldots\, m_{64}$, dann ist
$\mathrm{IP}(m) = m_{58}\, m_{50}\, m_{42}\, \ldots\, m_7$.

Die f-Funktion des DES

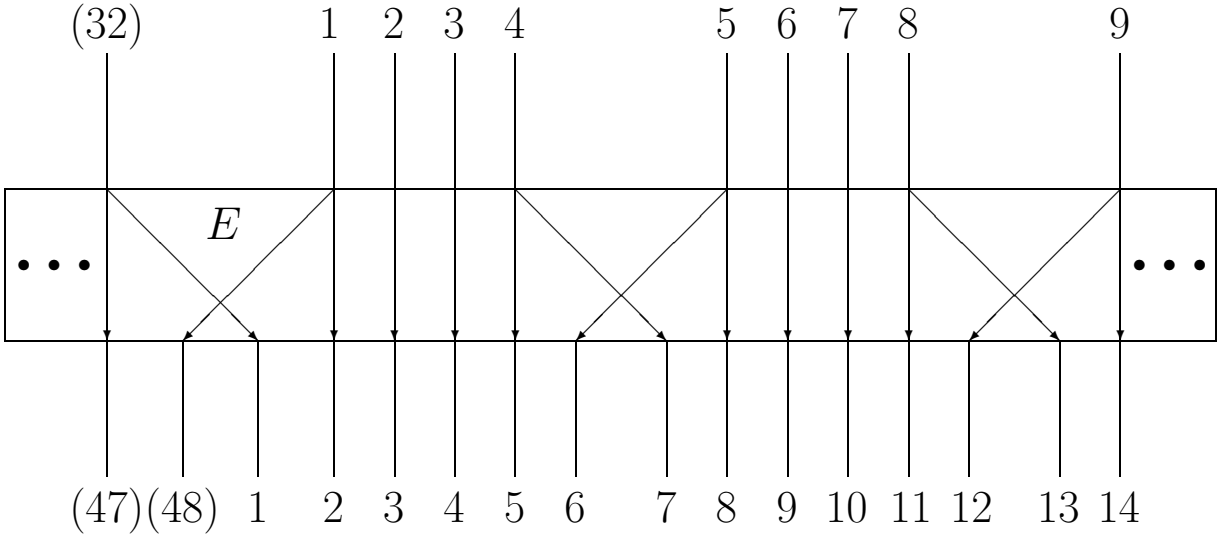


Die Expansionsfunktion E und die Permutation P

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

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

Anschauliche Darstellung der Expansionsfunktion E:



Die S-Boxen

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Anwendung:

$S_i(b_1\,b_2\,b_3\,b_4\,b_5\,b_6)$ = Zahl (in Binärdarstellung), die sich in der Box S_i in der Zeile Nr. $b_1\,b_6$ (Zeilennr = 0,1,2,3) und Spalte Nr. $b_2\,b_3\,b_4\,b_5$ (Spaltennr = 0,...,15) befindet, z.B. $S_8(111010) = 0011$.

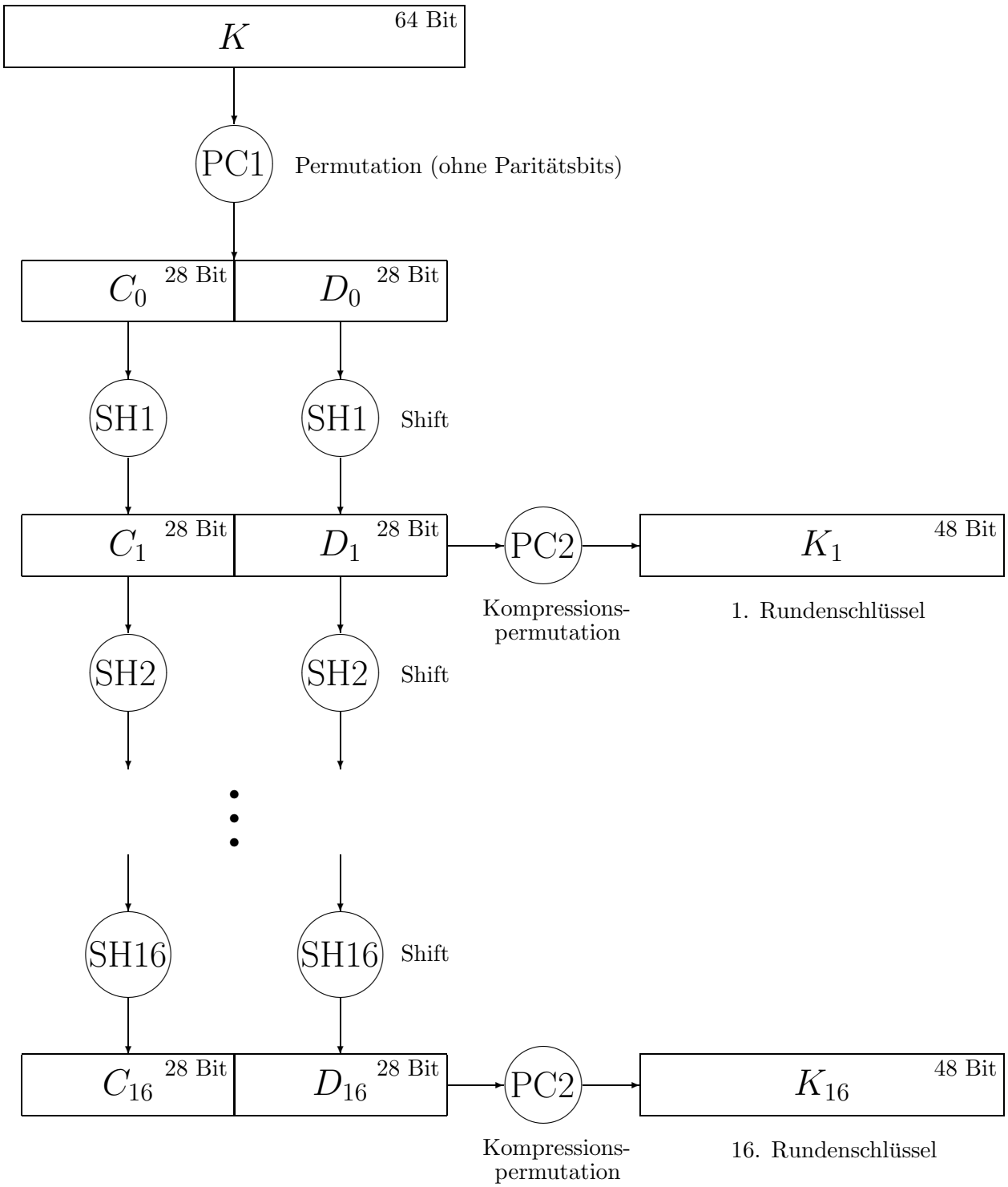
Bemerkungen zum Design der S-Boxen, der E- und P-Funktion

Veröffentlichte (von IBM bzw. der National Security Agency (NSA)) Design-Kriterien der S-Boxen sind unter anderen:

- Jede Zeile von jeder S-Box ist eine Permutation der Zahlen $0, \ldots, 15$.
- Keine S-Box ergibt eine lineare oder affine Funktion.
- Wenn sich zwei Eingabewerte einer S-Box in genau einem Bit unterscheiden, müssen sich die Ausgabewerte in mindestens zwei Bits unterscheiden.
- Für jede S-Box und jeden Eingabewert x gilt: $S(x)$ und $S(x \oplus 001100)$ unterscheiden sich in mindestens zwei Bits.

Daneben haben IBM bzw. die NSA weitere, komplizierter zu formulierende Design-Kriterien veröffentlicht. Dazu sei auf die Spezialliteratur verwiesen (z.B. Stinson, S. 82, oder Schneier, S. 341/342). Es ist aber nicht bekannt, ob IBM bzw. die NSA **alle** Kriterien veröffentlicht haben.
Die Permutation P ist dafür verantwortlich, daß ein Klartextbit bei seinem Weg durch den DES-Algorithmus in jeder Runde möglichst eine (bzw. mehrere) andere S-Box(en) durchläuft.
Die Expansionsfunktion E zusammen mit der Permutation P sorgen dafür, daß sich Änderungen von einem Klartextbit in möglichst vielen Chiffretextbits auswirken.
Bereits nach fünf DES-Runden hängt (laut Schneier, S. 330) jedes Chiffretextbit von jedem Bit des Klartextes und von jedem Bit des Schlüssels ab (**Lawineneffekt**).

Berechnung der DES-Rundenschlüssel



Die Funktionen PC1 und PC2

PC1							
57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36				
63	55	47	39	31	23	15	7
62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5
28	20	12	4				

Bemerkung:
Die Parittsbits 64, 56, 48, 40, 32, 24, 16 und 8 werden bei der Rundenschlüsselerzeugung ignoriert.
Ist $k = k_1\, k_2\, .\, .\, k_{64} \in \{0,1\}^{64}$, dann ist $\mathrm{PC1}(k) = (C, D)$ mit $C = k_{57}\, k_{49}\, .\, .\, k_{36} \in \{0,1\}^{28}$ und $D = k_{63}\, k_{55}\, .\, .\, k_4 \in \{0,1\}^{28}$.

PC2											
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Bemerkung:
Die Bits 9, 18, 22 und 25 von C und die Bits 35, 38, 43 und 54 von D werden ignoriert.
Ist $C = b_1\, b_2\, .\, .\, b_{28} \in \{0,1\}^{28}$ und $D = b_{29}\, b_{30}\, .\, .\, b_{56} \in \{0,1\}^{28}$, so ist $\mathrm{PC2}(C, D) = b_{14}\, b_{17}\, .\, .\, b_2\, b_{41}\, b_{52}\, .\, .\, b_{32}$.

Die Funktionen SH1 bis SH16

a) Verschlüsselung:

Die Funktionen SH1 bis SH16 sind zirkuläre **Linksshifte**, die auf die 28-Bit Teilschlüssel $C_0, \ldots, C_{15}, D_0, \ldots, D_{15}$ angewendet werden. Die Anzahl von Bits, um die verschoben wird, hängt von den Funktionen SH1 bis SH16 ab und ist in folgender Tabelle aufgelistet:

SH1	SH2	SH3	SH4	SH5	SH6	SH7	SH8
1	1	2	2	2	2	2	2
SH9	SH10	SH11	SH12	SH13	SH14	SH15	SH16
1	2	2	2	2	2	2	1

Bemerkung:
Die Teilschlüssel C_0 und C_{16} bzw. D_0 und D_{16} stimmen also überein (zirkulärer Linksshift um 28 Bit (auf 28 Bit Wörter) = identische Abbildung).

b) Entschlüsselung:

Da die Schlüssel in umgekehrter Reihenfolge erzeugt werden müssen, sind die Funktionen SH1 bis SH16 zirkuläre **Rechtsshifte**, die die Bits wie folgt verschieben:

SH1	SH2	SH3	SH4	SH5	SH6	SH7	SH8
0	1	2	2	2	2	2	2
SH9	SH10	SH11	SH12	SH13	SH14	SH15	SH16
1	2	2	2	2	2	2	1

Triple-DES

Es seien E_k bzw. D_k die (einfache) DES-Verschlüsselungs- bzw. Entschlüsselungsfunktion mit dem 56-Bit Schlüssel k.
Ein Klartext m wird dann vom Triple-DES wie folgt verschlüsselt:

$$c = E_{k_3}(D_{k_2}(E_{k_1}(m))).$$

Dabei sind k_1, k_2, k_3 56-Bit Schlüssel. Diese Art der Verschlüsselung nennt man auch Dreifach-Verschlüsselung im E-D-E-Modus. Im Falle von $k_1 = k_2 = k_3$ hat man Kompabilität mit dem einfachen DES.

Sind alle drei Schlüssel verschieden, so beträgt die Schlüssellänge $3 \times 56 = 168$ Bit.
Häufig verwendet wird aber derzeit die Variante des **2Key3DES**, bei dem $k_1 \neq k_2$, aber $k_1 = k_3$ gilt. Die Schlüssellänge beim 2Key3DES ist nur noch $2 \times 56 = 112$ Bit.

Bemerkung:

Die Zweifach-Verschlüsselung $c = E_{k_2}(E_{k_1}(m))$ wird nicht verwendet, da es dagegen die **meet-in-the-middle** Attacke gibt.

Ist ein Klartext m und sein Chiffretext c bekannt, so folgt aus der obigen Gleichung $D_{k_2}(c) = E_{k_1}(m)$. Man kann nun $E_{k_1}(m)$ für alle k_1 berechnen und speichern. Danach berechnet man $D_{k_2}(c)$ für alle k_2 und vergleicht diese mit den gespeicherten Ergebnissen. Hat man eine Übereinstimmung, so sind k_1 und k_2 Kandidaten für ein richtiges Schlüsselpaar. Der zeitliche Aufwand beträgt also nur noch $2^{56} + 2^{56} = 2^{57}$ anstelle von 2^{112} für das Probieren aller 112-bit Schlüssel. Allerdings sind 2^{56} 64-Bit Wörter zu speichern (= 576 Millionen GByte (ca. 850 Millionen CDs)).