

# Symmetrische Blockchiffren

DES (Data Encryption Standard)

und

AES (Advanced Encryption Standard)

# DES – Hintergrund

- Ausschreibung des NBA (National Bureau of Standards)
- Entwickelt von IBM (1975)
- Weiterentwicklung des Algorithmus „Lucifer“ von Horst Feistel
- Zusammenarbeit mit der NSA
- Genügt dem Kerckhoff'schen Prinzip

# DES – Eigenschaften

- DES ist ein „Feistel-Chiffre“
  - Symmetrisch
  - Blockchiffre
- Effektive Schlüssellänge: 56 Bit
- Tatsächliche Schlüssellänge: 64 Bit
- Blocklänge = Schlüssellänge: 64 Bit

# Feistel-Chiffren

- Voraussetzungen:
  - Schlüssellänge  $t$
  - Schlüssel  $K$
  - Rundenanzahl  $r$
  - Methode, um aus einem Schlüssel  $k$  die Rundenschlüssel  $K_1, \dots, K_r$  zu erzeugen
  - Verschlüsselungsfunktion  $f_K$  für Schlüssel  $K$
  - Klartext  $p$  der Blocklänge  $2t$

# Feistel-Chiffren

- Vorgehen bei Feistel-Verschlüsselung
  - Teile den Klartext in zwei Teilblöcke  $L_0$  und  $R_0$  der Länge  $t$  auf:

$$p = (L_0, R_0)$$

- Für  $i = 1, \dots, r$  definiere rekursiv:

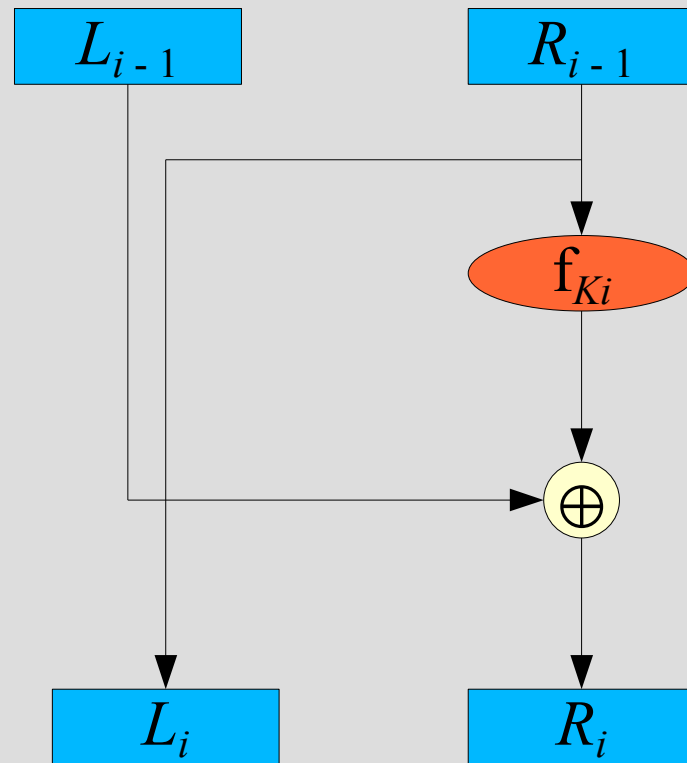
$$(L_i, R_i) := (R_{i-1}, L_{i-1} \oplus f_{K_i}(R_{i-1}))$$

- Den Ciphertext  $c$  erhält man durch Konkatenierung von  $L_r$  und  $R_r$ :

$$c = (L_r, R_r)$$

# Feistel-Chiffren

- Feistel-Verschlüsselung graphisch:



# Feistel-Chiffren

- Feistel-Entschlüsselung:
  - Durch die Rekursionsformel für  $(L_i, R_i)$  erhält man die Umkehrung:

$$(R_{i-1}, L_{i-1}) := (L_i, R_i \oplus f_{K_i}(L_i))$$

- Wendet man also die Verschlüsselung mit den Rundenschlüsseln  $(R_r, R_{r-1}, \dots, R_1)$  auf das Schlüsseltextpaar  $(R_r, L_r)$  an, so erhält man in  $r$  Runden das Klartextpaar  $(R_0, L_0)$  wieder.
- Also sind Feistel-Chiffren stets symmetrisch

# DES - Schlüssel

- DES ist ein leicht modifizierter Feistel-Chiffre:
  - 64 Bit Blocklänge
  - Rundenzahl  $r = 16$
  - Der Klartext wird am Anfang (und der Cyphertext am Ende) der Verschlüsselung permutiert
  - DES hat Schlüssellänge von 64Bit, aber pro Byte ist 1 Bit redundant (Quersumme = 1 in jedem Byte)
  - Die interne Verschlüsselungsfunktion  $f_K(R)$  arbeitet mit 8 sog. „S-Boxen“

# DES – initiale Permutation

- Die initiale Permutation *IP*
  - feste (bekannte) Permutation
  - Permutiert Klartext, vor der ersten Runde
- Umkehrung der initialen Permutation:
  - ebenfalls bekannt
  - Permutiert Schlüsseltext, nach der letzten Runde
- Ziel: Verbesserung der Robustheit gegen Angriffe
- Die Entschlüsselbarkeit wird durch die initiale Permutation nicht gefährdet

# DES – initiale Permutation

IP

Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	61	53	45	37	29	21	13	5
57	63	55	47	39	31	23	15	7

IP<sup>-1</sup>

Bit	0	1	2	3	4	5	6	7
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

# DES - Unterschüsselerzeugung

- Aus dem 64-Bit Schlüssel müssen 16 Rundenschlüssel von 48 Bit erzeugt werden
- Dazu definiere eine Schrittweitefunktion  $v_i$ :  
 $v_i := 1$  falls  $i \in \{1, 2, 9, 16\}$ , 2 sonst
- Definiere zwei weitere Funktionen PC1 und PC2:

$$PC1: \{0, 1\}^{64} \rightarrow \{0, 1\}^{28} \times \{0, 1\}^{28}$$

$$PC2: \{0, 1\}^{28} \times \{0, 1\}^{28} \rightarrow \{0, 1\}^{48}$$

# DES - Unterschlüsselerzeugung

**PC1**

Bit	0	1	2	3	4	5	6
<b>1</b>	57	49	41	33	25	17	9
<b>8</b>	1	58	50	42	34	26	18
<b>15</b>	10	2	59	51	43	35	27
<b>22</b>	19	11	3	60	52	44	36
<b>29</b>	63	55	47	39	31	23	15
<b>36</b>	7	62	54	46	38	30	22
<b>43</b>	14	6	61	53	45	37	29
<b>50</b>	21	13	5	28	20	12	4

**PC2**

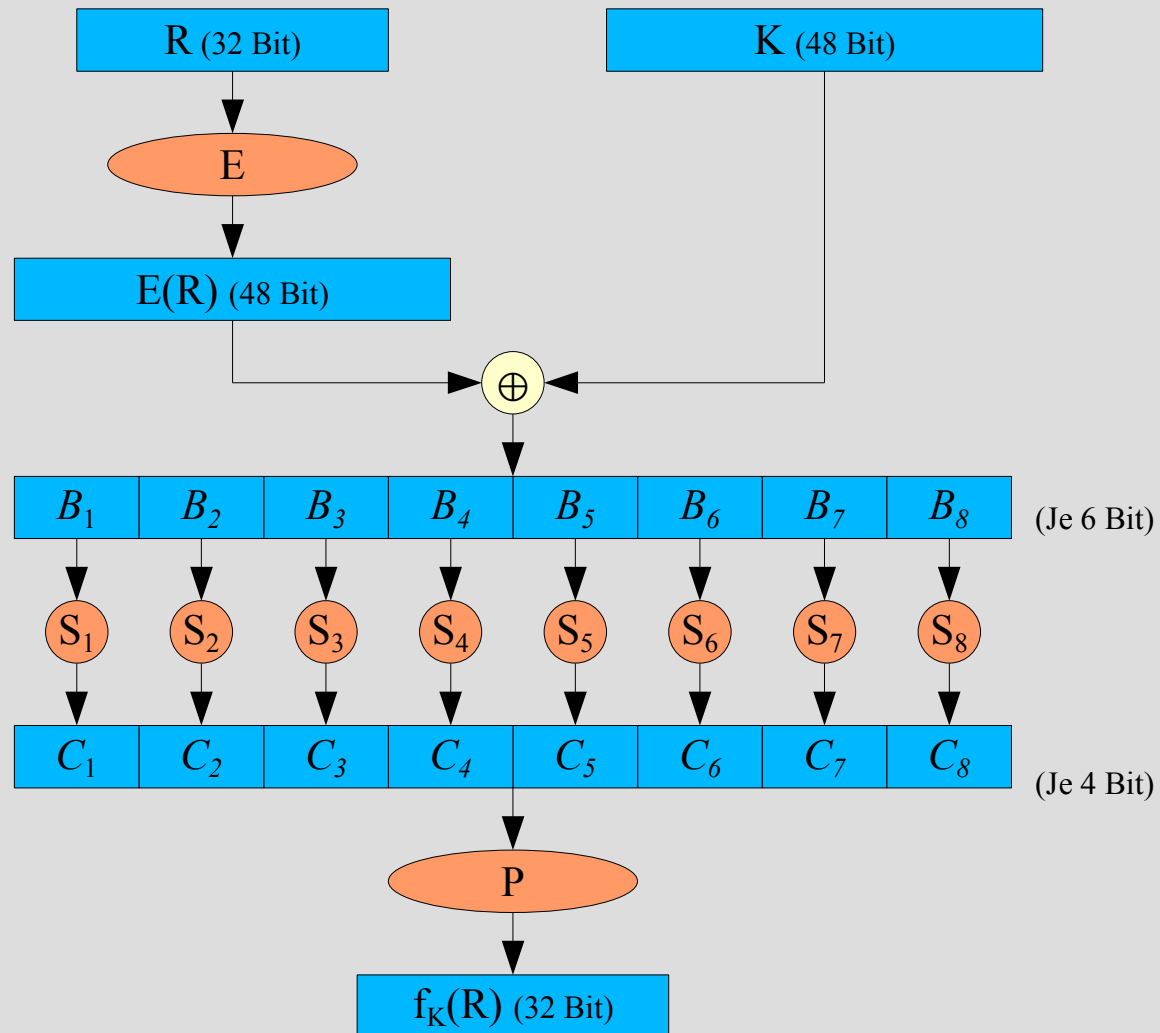
Bit	0	1	2	3	4	5
<b>1</b>	14	17	11	24	1	5
<b>7</b>	3	28	15	6	21	10
<b>13</b>	23	19	12	4	26	8
<b>19</b>	16	7	27	20	13	2
<b>25</b>	41	52	31	37	47	55
<b>31</b>	30	40	51	45	33	48
<b>37</b>	44	49	39	56	34	53
<b>43</b>	46	42	50	36	29	32

# DES - Unterschlüsselerzeugung

- Setze  $(C_0, D_0) := PC1(k)$
- Für  $i = 1, \dots, 16$ :
  - Erhalte  $C_i$  aus  $C_{i-1}$  durch zirkulären Linksshift um  $v_i$  Stellen
  - Erhalte  $D_i$  aus  $D_{i-1}$  durch zirkulären Linksshift um  $v_i$  Stellen
  - Die Rundenschlüssel  $R_i$  erhält man vermöge

$$R_i := PC2(C_i, D_i)$$

# DES – interne Verschlüsselung



# DES – interne Verschlüsselung

- Der Klartextblock  $R$  wird durch die Expansionsfunktion  $E$  auf 48 Bit expandiert
- $E(R)$  wird mit dem Rundenschlüssel  $K_i$  extern verodert ( $\oplus$ )
- Das Ergebnis wird in 8 Blöcke  $B_i$  von je 6 Bit aufgeteilt
- Auf jeden Block  $B_i$  wird die S-Box Funktion  $S_i$  angewandt und man erhält je einen 4-Bit Block  $C_i$
- Die Konkatenation aller  $C_i$  wird mit der Funktion  $P$  permutiert

# DES – interne Verschlüsselung

**Expansionsfunktion E**

Bit	0	1	2	3	4	5
1	32	1	2	3	4	5
7	4	5	6	7	8	9
13	8	9	10	11	12	13
19	12	13	14	15	16	17
25	16	17	18	19	20	21
31	20	21	22	23	24	25
37	24	25	26	27	28	29
43	28	29	30	31	32	1

**Permutation P**

Bit	0	1	2	3
1	16	7	20	21
5	29	12	28	17
9	1	15	23	26
13	5	18	31	10
17	2	8	24	14
21	32	27	3	9
25	19	13	30	6
29	22	11	4	25

# DES - S-Boxen

- S-Boxen sind Funktionen  $S_i: \{0,1\}^6 \rightarrow \{0,1\}^4$
- Design der S-Boxen macht einen Großteil der Sicherheit von DES aus und bietet guten Schutz gegen differentielle Kryptoanalyse
- Interpretation der S-Box Tabellen:  
Für ein Eingabewort  $b = b_1b_2b_3b_4b_5b_6$  sei  $b_1b_6$  der Zeilenindex und  $b_2b_3b_4b_5$  der Spaltenindex. Interpretiere die Zahl in der Zelle als binären Wert

# DES - S-Boxen

**S1**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
<b>1</b>	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
<b>2</b>	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
<b>3</b>	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

**S2**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
<b>1</b>	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
<b>2</b>	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
<b>3</b>	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

**S3**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
<b>1</b>	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
<b>2</b>	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
<b>3</b>	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

**S4**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
<b>1</b>	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
<b>2</b>	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
<b>3</b>	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

# DES - S-Boxen

**S5**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
<b>1</b>	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
<b>2</b>	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
<b>3</b>	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

**S6**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
<b>1</b>	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
<b>2</b>	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
<b>3</b>	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

**S7**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
<b>1</b>	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
<b>2</b>	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
<b>3</b>	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

**S8**

Zeile / Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>0</b>	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
<b>1</b>	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
<b>2</b>	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
<b>3</b>	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

# DES - Entschlüsselung

- Wie jeder Feistel-Chiffre kann DES trivialerweise durch Verwenden derselben Rundenschlüssel in inverser Reihenfolge ( $R_{16}, R_{15}, \dots, R_1$ ) entschlüsselt werden.

# DES – schwache Schlüssel

- Es gibt 4 „schwache Schlüssel“ (nur 1 Teilschlüssel):
  - 01 01 01 01 01 01 01 01
  - FE FE FE FE FE FE FE FE
  - 1F 1F 1F 1F 0E 0E 0E 0E
  - E0 E0 E0 E0 F1 F1 F1 F1
- Es gibt 12 „semi-schwache Schlüssel“ (nur 2 Teilschlüssel):
  - E0 FE E0 FE F1 FE F1 FE und FE E0 FE E0 FE F1 FE F1
  - 1F FE 1F FE 0E FE 0E FE und FE 1F FE 1F FE 0E FE 0E
  - 01 FE 01 FE 01 FE 01 FE und FE 01 FE 01 FE 01 FE 01
  - 1F E0 1F E0 0E F1 0E F1 und E0 1F E0 1F F1 0E F1 0E
  - 01 E0 01 E0 01 F1 01 F1 und E0 01 E0 01 F1 01 F1 01
  - 01 1F 01 1F 01 0E 01 0E und F1 01 1F 01 0E 01 0E 01
- Diese Schlüssel sind bei der Schlüsselerstellung unbedingt zu vermeiden!

# DES - Sicherheit

- Trotz beträchtlichen Alters relativ gute Sicherheit gegen die differentielle und lineare Kryptoanalyse (durch starke Konfusion/Diffusion)
- Bester Angriff: Brute-Force
- relativ kurze Schlüssellänge (56 Bit)
  - heute nicht sicher genug (Computer werden schneller!)
- Nachfolger: AES

# Triple DES

- DES-Kaskade für größere Schlüssellänge
- Das ist wirkungsvoll, da DES keine Gruppe bezüglich Komposition ist
- Eine 2-fache Kaskade ( $\text{DES}_{K_2} \circ \text{DES}_{K_1}$ ) bietet kaum (1 Bit) Sicherheitsgewinn wegen „Meet-In-The-Middle“ Angriff
- 3DES oder „Triple DES“ bietet als dreifach Kaskade eine gute Verdopplung der effektiven Schlüssellänge auf 118 Bit

# Triple DES

- Bei 3DES ist üblicherweise die zweite DES-Operation eine Entschlüsselung:

$$3DES(K_1, K_2, K_3, p) := (DES_{K_3} \circ DES_{K_2}^{-1} \circ DES_{K_1})(p) = DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(p)))$$

- Dadurch ist DES selbst ein Spezialfall von 3DES:

$$DES(K, p) = DES_K(p) = DES_K(DES_K^{-1}(DES_K(p))) = 3DES(K, K, K, p)$$

- Die Variante  $K_1 = K_3$  bietet höhere Angriffsfläche und sollte vermieden werden

# AES - Hintergrund

- **1997: Ausschreibung des NIST** (National Institute of Standards and Technology) **nach einem Nachfolger von DES**
- **Gewinner: Rijndael-Chiffre** von Vincent Rijmen und Joan Daemen
- **Weitere zur Auswahl stehende Chiffren:**
  - MARS
  - RC6
  - Serpent
  - Twofish

# AES - Hintergrund

- Anforderungen an AES:
  - symmetrischer Blockchiffre
  - Blocklänge 128 Bit
  - Schlüssellängen 128, 192, 256 Bit möglich
  - Leicht zu implementieren (Hard-/Software)
  - gute Performance, geringe Ressourcenanforderungen
  - hohe Widerstandsfähigkeit gegen alle kryptoanalytischen Angriffe
  - patentrechtlich frei

# AES - Eigenschaften

- Arbeitet mit Substitution
- zugrundeliegendes Alphabet:  $\mathbb{Z}_2$
- Rundenanzahl  $N_r$  ist abhängig von der Schlüssellänge:
  - AES-128: 10 Runden
  - AES-192: 12 Runden
  - AES-256: 14 Runden
- Textblöcke (128 Bit) werden in  $N_b = 4$  Wörter zu je 32 Bit aufgeteilt
- Schlüssel bestehen aus  $N_k$  32-Bit Wörtern (AES:  $N_k = 4, 6$  oder  $8$ )

# AES - Definitionen

- *byte*: Bitvektor der Länge 8
- *word*: Bitvektor der Länge 32 (= 4 Byte)
- Klartext, Chiffretext:
  - Darstellung als Byte-Matrix:
    - Höhe 4 Zeile, Breite  $Nb = 4$  Spalten
    - Jede Spalte entspricht einem *word*
  - Beispielsweise:

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix}$$

# AES - CIPHERALGORITHMUS

```
Cipher(byte[4,Nb] in, word[Nk] k) returns byte[4,Nb] {
  byte[4,Nb] state = in;
  word[Nb*(Nr+1)] w = KeyExpansion(k);
  state = AddRoundKey(state, w[0 to Nb-1]);
  for (round = 1 to Nr-1) {
    state = SubBytes(state);
    state = ShiftRows(state);
    state = MixColumns(state);
    state = AddRoundKey(state, w[round*Nb to (round+1)*Nb-1]);
  }
  state = SubBytes(state);
  state = ShiftRows(state);
  state = AddRoundKey(state, w[Nr*Nb to (Nr+1)*Nb-1]);
  return state;
}
```

# AES - CIPHERALGORITHMUS

- AES operiert auf sog. „states“. Ein „state“ ist zu Beginn der Klartext (in Matrixform) und sonst der aktuelle Zustand des Ciphertextes. Ein state ist stets eine  $4 \times (Nb) = 4 \times 4$  Matrix
- Funktion `KeyExpansion` erzeugt aus dem Schlüssel die  $r+1$  Rundenschlüssel
- Funktion `AddRoundKey` addiert (XOR!) den aktuellen Rundenschlüssel zum „state“
- Funktion `SubBytes` wendet die S-Box auf den „state“ an.
- Funktion `ShiftRows` verschiebt die Zeilen des „state“ gegeneinander
- Funktion `MixColumns` sorgt für Diffusion innerhalb der Spalten von „state“

# AES - Substitutionsbox

- Substitution in AES byteweise durch nichtlineare Funktion *subst*
- Zunächst: Darstellung jedes Bytes  $b$  durch Polynom über endlichem Körper  $GF(2^8)$ :

$$b = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 = \sum_{i=0}^7 2^i b_i$$

$$b(x) := \sum_{i=0}^7 b_i x^i = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0 x^0$$

# AES - Substitutionsbox

- Addition in  $GF(2^8)$ : Bitweise XOR

$$(b+c)(x) := \sum_{i=0}^7 (b_i \oplus c_i) x^i$$

- Multiplikation in  $GF(2^8)$ : (m ist irreduzibles Polynom)

$$m(x) := x^8 + x^4 + x^3 + x + 1$$

$$(b*c)(x) \equiv \left( \sum_{i=0}^7 b_i x^i \right) * \left( \sum_{i=0}^7 c_i x^i \right) \text{ mod } m(x)$$

- Inverses: Es existiert stets ein eindeutiges Inverses  $b^{-1}$  zu  $b \neq 0$ , so dass  $b^{-1} * b \equiv 1 \text{ mod } m$
- Setze  $0^{-1} := 0$ .

# AES - Substitutionsbox

- Definiere nun eine nichtlineare, umkehrbare Funktion *subst*, vermöge:

$$\text{subst}(b) := Ab^{-1} \oplus c$$
$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$
$$c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

- Häufig durch eine Substitutionstabelle realisiert, da *b* nur 256 Werte annehmen kann
- Die Funktion `SubBytes` wendet die Funktion *subst* auf jedes Byte von `state` an und gibt das Ergebnis zurück

# AES – Zeilenverschiebung

- Die Funktion `ShiftRows` arbeitet folgendermaßen:
  - erste Zeile: keine Änderung
  - zweite Zeile: zyklischer Linksshift um 1 Byte
  - dritte Zeile: zyklischer Linksshift um 2 Bytes
  - vierte Zeile: zyklischer Linksshift um 3 Bytes

s0,0	s0,1	s0,2	s0,3	→	s0,0	s0,1	s0,2	s0,3
s1,0	s1,1	s1,2	s1,3		s1,1	s1,2	s1,3	s1,0
s2,0	s2,1	s2,2	s2,3		s2,2	s2,3	s2,0	s2,1
s3,0	s3,1	s3,2	s3,3		s3,3	s3,0	s3,1	s3,2

# AES - Spaltenmischung

- Funktion `MixColumns` sorgt für Diffusion innerhalb der Spalten.
- Interpretation der Spalten als Polynom:

$$s_j = (s_{0,j}, s_{1,j}, s_{2,j}, s_{3,j}) \hat{=} s_{0,j} + s_{1,j}x + s_{2,j}x^2 + s_{3,j}x^3$$

- Jede Spalte  $s_j$  (für  $0 \leq j < Nb = 4$ ) wird transformiert:

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

$$s_j \rightarrow (s_j * a(x)) \bmod (x^4 + 1)$$

# AES - Spaltenmischung

- Alternative Darstellung: Matrixmultiplikation

$$\begin{pmatrix} s_{0,j}' \\ s_{1,j}' \\ s_{2,j}' \\ s_{3,j}' \end{pmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{pmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{pmatrix}$$

Die Multiplikation ist als die Anfangs definierte Multiplikation auf  $GF(2^8)$  zu verstehen.

Die Addition ebenso.

# AES - Schlüsselexpansion

- $(r+1)$  Rundenschlüssel von der Länge des Klartextes benötigt
- $w$ : byte-Matrix der Form  $4 \times (Nb * (Nr+1))$
- Vorgehen:
  - Die ersten  $Nk$  Spalten von  $w$  werden mit  $k$  gefüllt
  - Für alle weiteren  $i$ -ten Spalten von  $w$ :
    - $Temp := (i-1)$ te Spalte von  $w = w[i-1]$
    - Falls  $i$  Vielfaches von  $Nk$ :  
 $temp = \text{SubWord}(\text{RotWord}(temp)) \oplus rcon[i/Nk]$
    - Falls  $Nk = 8$  und  $n$  modulo  $8 = 4$ :  
 $temp = \text{SubWord}(temp)$
    - Die  $i$ -te Spalte von  $w$  ist:  $w[i] = temp \oplus w[i-Nk]$

# AES - Schlüsselexpansion

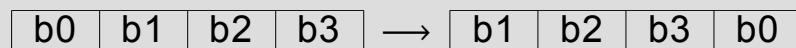
```
KeyExpansion(word[Nk] k) returns word[Nb*(Nr+1)] {
    word[Nb*(Nr+1)] w;
    word temp;
    i = 0;
    for (i = 0 to Nk - 1) {
        w[i] = k[i];
    }
    for (i = Nk to Nb*(Nr+1)-1) {
        temp = w[i-1];
        if (i mod Nk == 0) {
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk];
        }
        if ((Nk == 8) and (i mod Nk == 4)) {
            temp = SubWord(temp);
        }
        w[i] = w[i-Nk] xor temp;
    }
    return w;
}
```

# AES - Schlüsselexpansion

- $Rcon[i] := (\{02\}^{i-1}, \{00\}, \{00\}, \{00\})$
- Die Potenzreihe von  $\{02\}$  in  $GF(2^8)$ :

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\{02\}^{i \bmod m}$	{01}	{02}	{04}	{08}	{10}	{20}	{40}	{80}	{1b}	{36}	{6c}	{d8}	{ab}	{4d}	{9a}

- Funktion SubWord: Wendet die S-Box *subst* auf jedes *byte* des Wortes an
- Funktion RotWord: Zyklischer Linksshift um 1 byte:



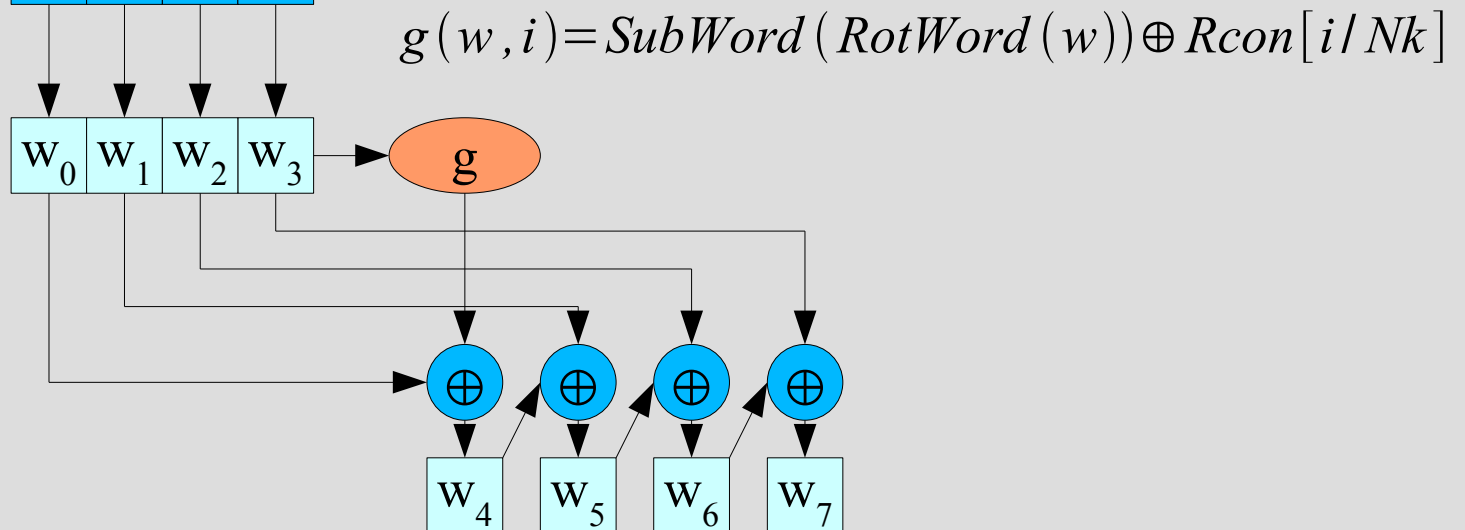
# AES - Schlüsselexpansion

- Beispiel für  $Nk = 4$  ( $w_8 - w_{43}$  analog erzeugt)

Schlüssel (Bytes)

$K_0$	$K_4$	$K_8$	$K_{12}$
$K_1$	$K_5$	$K_9$	$K_{13}$
$K_2$	$K_6$	$K_{10}$	$K_{14}$
$K_3$	$K_7$	$K_{11}$	$K_{15}$

Expandierter  
Schlüssel (words)



# AES - Entschlüsselung

```
DeCipher(byte[4,Nb] in, word[Nk] k) returns byte[4,Nb] {
    byte[4,Nb] state = in;
    word[Nb*(Nr+1)] w = KeyExpansion(k);
    state = AddRoundKey(state, w[Nr*Nb to (Nr+1)*Nb-1]);
    for (round = Nr-1 downto 1) {
        state = InvShiftRows(state);
        state = InvSubBytes(state);
        state = AddRoundKey(state, w[round*Nb to (round+1)*Nb-1]);
        state = InvMixColumns(state);
    }
    state = InvShiftRows(state);
    state = InvSubBytes(state);
    state = AddRoundKey(state, w[0 to Nb-1]);
    return state;
}
```

# AES - Entschlüsselung

- Rückwärtsausführung der Schritte in Cipher
- Inverse Funktionen zu SubBytes, ShiftRows, MixColumns:
  - InvSubBytes: Rücksubstituierung anhand Tabelle
  - InvShiftRows: zyklischer Shift um entsprechend viele bytes nach rechts
  - InvMixColumns: wie MixColumns, nur mit einem anderen (inversen) Polynom:

$$a^{-1}(x) := \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$$

# AES - Sicherheit

- Insgesamt sehr gute Diffusion und Konfusion
  - Robust gegen statistische Angriffe
- S-Boxen sorgen für nichtlinearität von AES
  - Robust gegen lineare und differentielle Kryptoanalyse
- große Schlüssellänge (bis zu 256 Bit)
  - Robust gegen Brute-Force
- Bisher nur theoretische Angriffe bekannt, die eine extrem hohe Komplexität haben ( $2^{100}$  Rechenschritte)