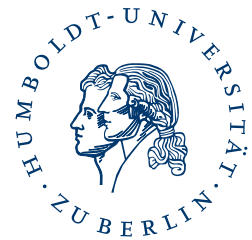
# Einführung in die Kryptologie

Johannes Köbler




Institut für Informatik
Humboldt-Universität zu Berlin

SS 2022

# Produktchiffren

- Produktchiffren erhält man durch die sequentielle Anwendung mehrerer Verschlüsselungsverfahren
- Sie können extrem schwer zu brechen sein, auch wenn die einzelnen Komponenten leicht zu brechen sind

## Definition

- Seien $KS_i = (M_i, C_i, E_i, D_i, K_i, \mathcal{S}_i)$, $i \in \{1, 2\}$, Kryptosysteme mit $C_1 = M_2$
- Dann ist das Produktkryptosystem $KS_1 \times KS_2$ von $KS_1$ und $KS_2$ definiert als $(M_1, C_2, E, D, K_1 \times K_2, \mathcal{S})$ mit $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ und

$$E(k_1, k_2; x) = E_2(k_2, E_1(k_1, x)) \text{ sowie } D(k_1, k_2; y) = D_1(k_1, D_2(k_2, y))$$

für alle $x \in M_1$, $y \in C_2$ und $(k_1, k_2) \in K_1 \times K_2$

- Der Schlüsselraum von $KS_1 \times KS_2$ umfasst also alle Schlüsselpaare $(k_1,k_2) \in K_1 \times K_2$, wobei wir voraussetzen, dass die beiden Schlüssel unabhängig gewählt werden (d.h. es gilt $p(k_1, k_2) = p(k_1)p(k_2)$)

## Beispiel

- Bilden wir das Produkt $KS = KS^* \times KS^+$ der
  - multiplikativen Chiffre $KS^* = (M, C, \mathbb{Z}_m^*, E^*, D^*)$ mit $M = C = \mathcal{A} = \{a_0, \ldots, a_{m-1}\}$ und der
  - additiven Chiffre $KS^+ = (M, C, \mathbb{Z}_m, E^+, D^+)$,

  so gilt für jeden Schlüssel $(k_1, k_2) \in K = \mathbb{Z}_m^* \times \mathbb{Z}_m$:

  $$E(k_1, k_2; x) = E^+(k_2, E^*(k_1, x)) = k_1 x + k_2$$

- Dies bedeutet, dass das Produkt $KS^* \times KS^+ = (M, C, K, E, D)$ der multiplikativen und additiven Chiffre die affine Chiffre ist
- Welche Chiffre erhalten wir, wenn wir das Produkt $KS^+ \times KS^*$ der additiven und der multiplikativen Chiffre bilden?

## Beispiel (Fortsetzung)

- Das Produkt $KS^+ \times KS^*$ ist das Kryptosystem $KS' = (M, C, K', E', D')$, in dem für jeden Schlüssel $(k_2, k_1) \in K' = \mathbb{Z}_m \times \mathbb{Z}_m^*$ gilt:

$$E'(k_2, k_1; x) = k_1(x + k_2) = k_1 x + k_1 k_2 = E(k_1, k_1 k_2; x)$$

- Die Abbildung $(k_2, k_1) \mapsto (k_1, k_1 k_2)$ ist also eine Bijektion zwischen den Schlüsselräumen $\mathbb{Z}_m \times \mathbb{Z}_m^*$ und $\mathbb{Z}_m^* \times \mathbb{Z}_m$, die jeden Schlüssel $(k_2, k_1) \in \mathbb{Z}_m \times \mathbb{Z}_m^*$ auf einen Schlüssel $(k_1, k_1 k_2) \in \mathbb{Z}_m^* \times \mathbb{Z}_m$ mit $E_{(k_1, k_1 k_2)} = E'_{(k_2, k_1)}$ abbildet
- Somit können wir auch jeden Schlüsselgenerator $\mathcal{S}$ für $KS^* \times KS^+$ in einen Generator $\mathcal{S}'$ für $KS^+ \times KS^*$ transformieren (und $\mathcal{S}'$ auch wieder zurück in $\mathcal{S}$), so dass $\mathcal{S}'$ in $KS^+ \times KS^*$ jede Chiffrierfunktion mit der gleichen Wahrscheinlichkeit erzeugt wie $\mathcal{S}$ in $KS^* \times KS^+$
- Daher können die beiden Kryptosysteme $KS^* \times KS^+$ und $KS^+ \times KS^*$ als gleich (oder besser äquivalent, siehe Übungen) angesehen werden, d.h. die beiden Kryptosysteme $KS^*$ und $KS^+$ kommutieren ◁

# Produktchiffren

## Definition

- Ein Kryptosystem $KS = (M, C, K, D, E)$ mit $M = C$ heißt endomorph
- Ein endomorphes Kryptosystem $KS$ heißt idempotent, falls $KS \times KS$ äquivalent zu $KS$ ist (in Zeichen: $KS \times KS = KS$)

## Beispiel

- Eine leichte Rechnung zeigt, dass
  - die additive Chiffre,
  - die multiplikative Chiffre und
  - die affine Chiffre

  idempotent sind
- Dies trifft auch auf
  - die Blocktransposition sowie
  - die Vigenère- und Hill-Chiffre

  zu (siehe Übungen)

◁

# Produktchiffren

- Will man durch mehrmalige Anwendung (Iteration) derselben Chiffre eine höhere Sicherheit erreichen, so darf diese nicht idempotent sein
- Man kann versuchen, durch sequentielle Ausführung zweier idempotenter Systeme $KS_1$ und $KS_2$ ein System $KS = KS_1 \times KS_2$ zu erhalten, das nicht idempotent ist
- Da $KS$ im Fall $KS_1 \times KS_2 = KS_2 \times KS_1$ wegen

$$\begin{aligned} KS \times KS &= (KS_1 \times KS_2) \times (KS_1 \times KS_2) \\ &= KS_1 \times (KS_2 \times KS_1) \times KS_2 \\ &= KS_1 \times (KS_1 \times KS_2) \times KS_2 \\ &= (KS_1 \times KS_1) \times (KS_2 \times KS_2) \\ &= KS_1 \times KS_2 \\ &= KS \end{aligned}$$

idempotent ist, dürfen hierbei $KS_1$ und $KS_2$ jedoch nicht kommutieren

# Iterierte Blockchiffren

- Im Folgenden betrachten wir Blockchiffren über dem Binäralphabet $A = \{0, 1\}$ und auch der Schlüsselraum wird von der Form $\{0, 1\}^k$ sein
- Die Schlüssellänge bezeichnen wir bis auf weiteres mit $k$ und einzelne Schlüssel eines Kryptosystems mit $K$
- Eine iterierte Blockchiffre wird durch eine Rundenfunktion (round function) $g$ und einen Key-Schedule Algorithmus $f$ beschrieben
- Ist $N$ die Rundenzahl, so erzeugt $f$ bei Eingabe eines Schlüssels $K$ eine Folge $f(K) = (K^1, \ldots, K^N)$ von $N$ Rundenschlüsseln $K^i$ für $g$

# Iterierte Blockchiffren

- Mit diesen wird ein Klartext $x = w^0$ durch $N$-malige Anwendung der Rundenfunktion $g$ zu einem Kryptotext $y = w^N$ verschlüsselt:

  $$w^1 := g(K^1, w^0)$$
  $$\vdots$$
  $$w^N := g(K^N, w^{N-1})$$

- Um $y$ wieder zu entschlüsseln, muss die inverse Rundenfunktion $g^{-1}$ mit umgekehrter Rundenschlüsselfolge $K^N, \ldots, K^1$ benutzt werden:

  $$w^{N-1} := g^{-1}(K^N, w^N)$$
  $$\vdots$$
  $$w^0 := g^{-1}(K^1, w^1)$$

- Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir später behandeln werden

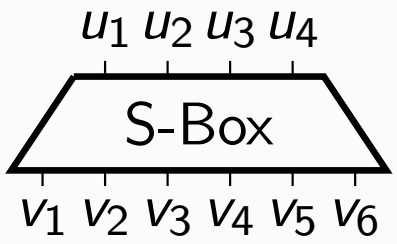
# Substitutions-Permutations-Netzwerke

- Als Basisbausteine für die Rundenfunktion von iterierten Blockchiffren eignen sich Substitutionen und Transpositionen besonders gut
- Aus Effizienzgründen sollten die Substitutionen nur eine relativ kleine Blocklänge haben

## Definition

- Für ein Wort $u = u_1 \cdots u_n \in \{0,1\}^n$ und Indizes $1 \leq i \leq j \leq n$ bezeichne $u[i,j]$ das Teilwort $u_i \cdots u_j$ von $u$
- Im Fall $n = ml$ bezeichnen wir das Teilwort $u[(i-1)l+1, il]$ auch einfach mit $u_{(i)}$, d.h. es gilt $u = u_{(1)} \cdots u_{(m)}$, wobei $|u_{(i)}| = l$ ist

# Substitutions-Permutations-Netzwerke




- Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ eine Substitution, die Binärblöcke $u$ der Länge $l$ in Blöcke $v = \sigma_S(u)$ der Länge $l'$ überführt (auch S-Box $S$ genannt)
- Für $\sigma_S(u)$ schreiben wir auch einfach $S(u)$
- Durch parallele Anwendung von $m$ Kopien der S-Box $S$ erhalten wir die Substitution $\sigma_{mS} : \{0,1\}^{ml} \to \{0,1\}^{ml'}$ mit

$$\sigma_{mS}(u_1 \cdots u_{ml}) = S(u_{(1)}) \cdots S(u_{(m)})$$

- Auch hier schreiben wir für $\sigma_{mS}(u_1 \cdots u_{ml})$ auch einfach $S(u_1 \cdots u_{ml})$
- Für die Speicherung einer S-Box $S$ mit $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ auf einem Chip werden $l' \cdot 2^l$ Bit Speicherplatz benötigt (im Fall $l = l'$ also $l2^l$ Bit)
- Für $l = l' = 16$ wären dies beispielsweise $2^{20}$ Bit, was Smartcard-Anwendungen bereits ausschließen würde
- Für eine Transposition $P$ auf $\{0,1\}^\ell$ bezeichnen wir die zugehörige Permutation auf $[\ell]$ mit $\pi_P$ oder einfach mit $\pi$, falls $P$ aus dem Kontext bekannt ist, d.h. $P(u_1 \cdots u_\ell) = u_{\pi(1)} \cdots u_{\pi(\ell)}$

# Substitutions-Permutations-Netzwerke

## Definition

- Für natürliche Zahlen $m, l \geq 1$ sei $M = C = \{0,1\}^\ell$ mit $\ell = ml$
- Ein Substitutions-Permutations-Netzwerk (SPN) wird durch eine S-Box $S$, eine Blocktransposition $P$ mit Blocklänge $\ell = ml$ und durch eine Funktion $f : \{0,1\}^k \to \{0,1\}^{\ell(N+1)}$ beschrieben
- Hierbei realisiert die S-Box $S$ eine Permutation $\sigma_S$ auf $\{0,1\}^l$ und $N$ ist die Rundenzahl des SPN
- Die Funktion $f$ transformiert einen (externen) Schlüssel $K \in \{0,1\}^k$ in ein Key-Schedule $f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln
- Unter ihnen wird ein Klartext $x \in \{0,1\}^\ell$ durch folgenden Chiffrieralgorithmus in einen Kryptotext $y = E_{f,S,P}(K,x) \in \{0,1\}^\ell$ überführt:

1 $w^0 := x$
2 **for** $r := 1$ **to** $N-1$ **do**
3 $\quad u^r := w^{r-1} \oplus K^r$
4 $\quad v^r := S(u^r)$
5 $\quad w^r := P(v^r)$
6 $u^N := w^{N-1} \oplus K^N$
7 $v^N := S(u^N)$
8 $y := v^N \oplus K^{N+1}$

# Substitutions-Permutations-Netzwerke

## Die Chiffrierfunktion $E_{f,S,P}(K, x)$

- Zu Beginn jeder Runde $r \in \{1, \ldots, N\}$ wird $w^{r-1}$ zuerst einer XOR-Operation mit dem Rundenschlüssel $K^r$ unterworfen (round key mixing) und das Resultat $u^r$ den S-Boxen zugeführt
- Auf die Ausgabe $v^r$ der S-Boxen wird in jeder Runde $r \leq N-1$ die Transposition $P$ angewendet, was die Eingabe $w^r$ für die nächste Runde $r+1$ liefert

```
1  w^0 := x
2  for r := 1 to N − 1 do
3     u^r := w^{r−1} ⊕ K^r
4     v^r := S(u^r)
5     w^r := P(v^r)
6  u^N := w^{N−1} ⊕ K^N
7  v^N := S(u^N)
8  y := v^N ⊕ K^{N+1}
```

- Am Ende der letzten Runde $r = N$ wird nicht die Transposition $P$ angewandt, sondern der Rundenschlüssel $K^{N+1}$ auf $v^N$ addiert
- Dies wird whitening genannt und bewirkt, dass auch für den letzten Chiffrierschritt der Schlüssel benötigt und somit der Gegner an einer partiellen Entschlüsselung des Kryptotextes gehindert wird
- Zudem wird dadurch eine (legale) Entschlüsselung nach fast demselben Verfahren ermöglicht (siehe Übungen)

# Substitutions-Permutations-Netzwerke

- Die S-Boxen sorgen dafür, dass jedes einzelne Bit des Kryptotextes $y$ von mehreren Bits des Klartextes und des Schlüssels abhängt
- Wichtig ist hierbei, dass die Abhängigkeit möglichst komplex ist (also z.B. nicht linear)
- Dadurch werden Angriffe erschwert, die versuchen, Rückschlüsse vom Kryptotext auf den Schlüssel oder Klartext zu ziehen
  (Shannon nannte diese Eigenschaft Konfusion)
- Die Transpositionen über den gesamten Block sorgen dafür, dass sich eine Änderung von einzelnen Bits des Klartextes oder des Schlüssels potentiell auf jedes Kryptotextbit auswirken kann
  (von Shannon als Diffusion bezeichnet)
- Idealerweise wird hierdurch erreicht, dass sich bei Änderung eines einzelnen Eingabe- oder Schlüsselbits jedes Ausgabebit mit Wahrscheinlichkeit $1/2$ ändert

## Beispiel

- Wir betrachten ein SPN $SP$ mit Parametern $l = m = N = 4$ und $k = 32$
- Für $f$ wählen wir die Funktion $f(K) = (K^1, \ldots, K^5)$ mit $K^r = K[4(r-1)+1, 4(r-1)+16]$
- Weiter seien $S : \{0,1\}^4 \to \{0,1\}^4$ und $\sigma_P : \{1, \ldots, 16\} \to \{1, \ldots, 16\}$ die folgenden Permutationen:

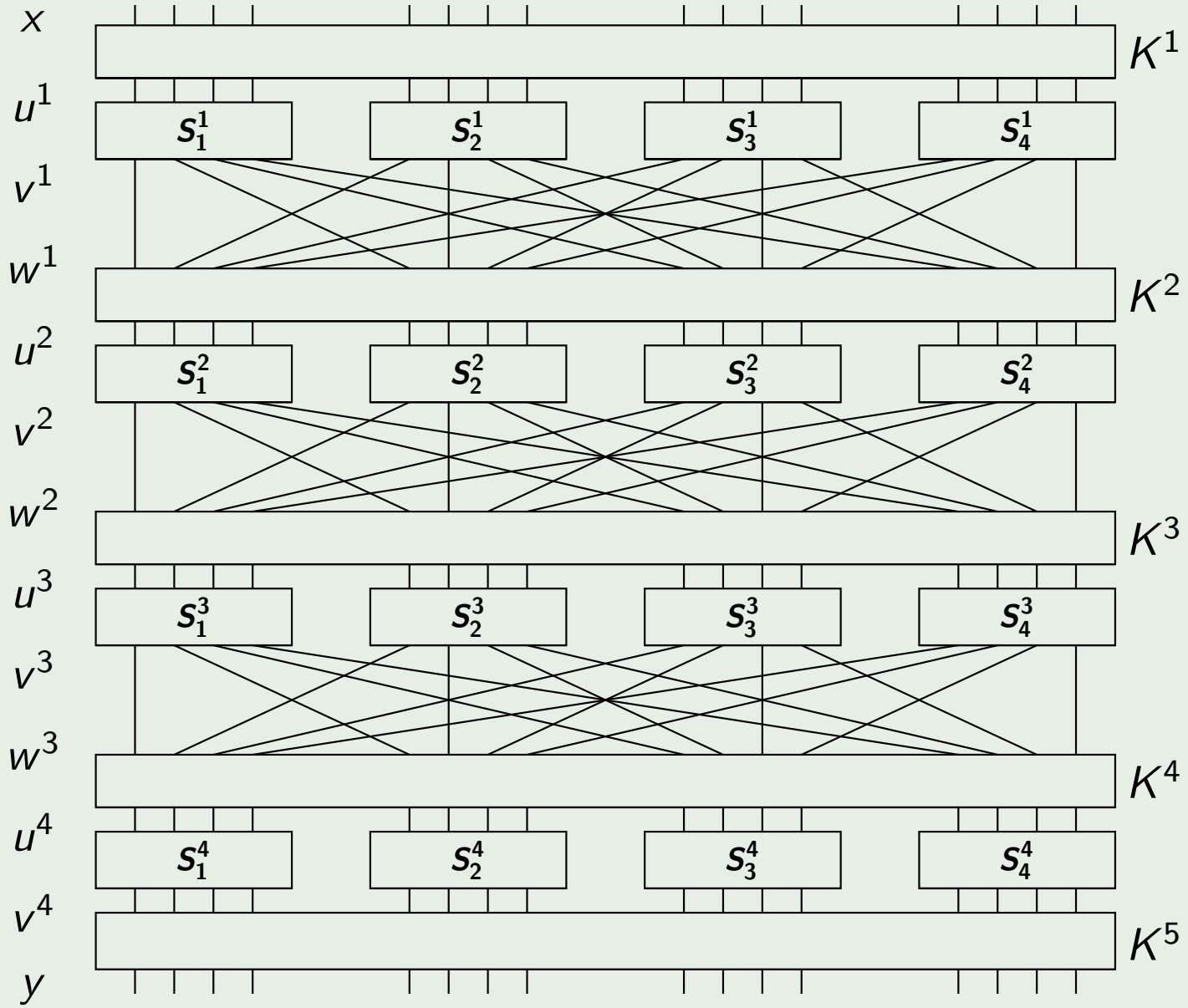
| $z$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(z)$ | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

wobei die Argumente und Werte von $S$ hexadezimal dargestellt sind, und

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma_P(i)$ | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

# Substitutions-Permutations-Netzwerke

## Beispiel (Fortsetzung)



# Substitutions-Permutations-Netzwerke

## Beispiel (Schluss)

- Beispielsweise liefert $f$ für den Schlüssel $K = 0011\,1010\,1001\,0100\,1101\,0110\,0011\,1111$ die Rundenschlüssel $f(K) = (K^1, \ldots, K^5)$ mit

$$K^1 = 0011\,1010\,1001\,0100 \qquad K^2 = 1010\,1001\,0100\,1101$$
$$K^3 = 1001\,0100\,1101\,0110 \qquad K^4 = 0100\,1101\,0110\,0011$$
$$K^5 = 1101\,0110\,0011\,1111$$

unter denen der Klartext $x = 0010\,0110\,1011\,0111$ die folgenden Chiffrierschritte durchläuft:

$$\begin{aligned}
x &= 0010\,0110\,1011\,0111 = w^0 \\
w^0 \oplus K^1 &= 0001\,1100\,0010\,0011 = u^1 \\
S(u^1) &= 0100\,0101\,1101\,0001 = v^1 \\
P(v^1) &= 0010\,1110\,0000\,0111 = w^1 \\
&\vdots \\
P(v^3) &= 1110\,0100\,0110\,1110 = w^3 \\
w^3 \oplus K^4 &= 1010\,1001\,0000\,1101 = u^4 \\
S(u^4) &= 0110\,1010\,1110\,1001 = v^4 \\
u^4 \oplus K^5 &= 1011\,1100\,1101\,0110 = y
\end{aligned}$$

◁

# Lineare Approximationen

- Sei $f : \{0,1\}^l \to \{0,1\}^{l'}$ eine boolesche Funktion
- Wählen wir die Eingabe $\mathcal{U} = \mathcal{U}_1 \cdots \mathcal{U}_l$ zufällig unter Gleichverteilung, so gilt für die zugehörige Ausgabe $\mathcal{V} = f(\mathcal{U}) = \mathcal{V}_1 \cdots \mathcal{V}_{l'}$ sowie für alle $u \in \{0,1\}^l$ und $v \in \{0,1\}^{l'}$:

$$\Pr[\mathcal{V} = v \mid \mathcal{U} = u] = \begin{cases} 1 & f(u) = v \\ 0 & \text{sonst} \end{cases}$$

- Wegen $\Pr[\mathcal{U} = u] = 2^{-l}$ folgt

$$\Pr[\mathcal{V} = v, \mathcal{U} = u] = \begin{cases} 2^{-l} & f(u) = v, \\ 0 & \text{sonst} \end{cases}$$

- Die Funktion $f$ ist linear, wenn eine binäre $(l \times l')$-Matrix $A$ mit $f(u) = uA$ existiert
- In diesem Fall ist jedes Ausgabebit $v_j$ in der Form $v_j = u_{i_1} \oplus \cdots \oplus u_{i_k}$ mit $1 \leq i_1 < \cdots < i_k \leq l$ darstellbar, d.h. $\Pr[\mathcal{V}_j = \mathcal{U}_{i_1} \oplus \cdots \oplus \mathcal{U}_{i_k}] = 1$

# Lineare Approximationen

- Für eine lineare Kryptoanalyse benötigen wir Gleichungen der Form

$$\mathcal{V}_{j_1} \oplus \cdots \oplus \mathcal{V}_{j_{k'}} = \mathcal{U}_{i_1} \oplus \cdots \oplus \mathcal{U}_{i_k} \oplus c$$

  mit $1 \le i_1 < \cdots < i_k \le l$, $1 \le j_1 < \cdots < j_{k'} \le l'$ und $c \in \{0,1\}$, die mit möglichst großer Wahrscheinlichkeit gelten
- Definieren wir für $u, a \in \{0,1\}^l$ und $v, b \in \{0,1\}^{l'}$ die xor-Teilsummen

$$u_a = \bigoplus_{i=1}^{l} a_i u_i \quad \text{und} \quad v_b = \bigoplus_{i=1}^{l'} b_i v_i,$$

  so suchen wir also nach Werten für $a$, $b$ und $c$, für die das Ereignis $\mathcal{V}_b = \mathcal{U}_a \oplus c$ (oder $\mathcal{U}_a \oplus \mathcal{V}_b = c$) mit großer Wahrscheinlichkeit eintritt
- In diesem Fall lässt sich nämlich der Wert von $\mathcal{V}_b$ bei Kenntnis von $\mathcal{U}_a$ entsprechend gut vorhersagen
- Wegen $\Pr[\mathcal{U}_a \oplus \mathcal{V}_b = c] = 1 - \Pr[\mathcal{U}_a \oplus \mathcal{V}_b = c \oplus 1]$ kommt es nur darauf an, wie stark die Wahrscheinlichkeit $\Pr[\mathcal{U}_a \oplus \mathcal{V}_b = 0]$ von $1/2$ abweicht
- $\mathcal{V}_b$ lässt sich also in Abhängigkeit von $\mathcal{U}_a$ um so besser vorhersagen, je größer der Absolutbetrag $|\Pr[\mathcal{U}_a \oplus \mathcal{V}_b = 0] - 1/2|$ ist

# Güte einer linearen Approximation an eine S-Box

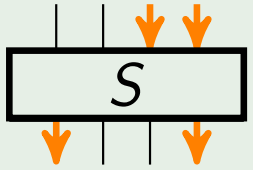
## Definition

- Der Bias einer Zufallsvariablen $\mathcal{X}$ mit Wertebereich $W(\mathcal{X}) = \{0, 1\}$ ist definiert als $\beta(\mathcal{X}) = \Pr[\mathcal{X} = 0] - 1/2$
- Eine lineare Approximation an eine boolesche Funktion $f : \{0,1\}^l \to \{0,1\}^{l'}$ wird durch ein Paar $(a, b) \in \{0,1\}^l \times \{0,1\}^{l'}$ beschrieben
- Die Güte der durch $(a, b)$ beschriebenen Approximation an $f$ ist der doppelte Absolutbetrag $\gamma_f(a, b) = 2|\beta(\mathcal{U}_a \oplus \mathcal{V}_b)|$ des Bias-Wertes von $\mathcal{U}_a \oplus \mathcal{V}_b$, wobei $\mathcal{U}$ auf $\{0,1\}^l$ gleichverteilt und $\mathcal{V} = f(\mathcal{U})$ ist

# Lineare Approximationen

## Beispiel

- Wir betrachten die durch das Paar $(0011, 1001)$ beschriebene lineare Approximation $\mathcal{A} = \mathcal{U}_3 \oplus \mathcal{U}_4 \oplus \mathcal{V}_1 \oplus \mathcal{V}_4$ an die S-Box $S$ aus vorigem Beispiel




- Dann nimmt die Zufallsvariable $(\mathcal{U}_1, \ldots, \mathcal{U}_4, \mathcal{V}_1, \ldots, \mathcal{V}_4, \mathcal{A})$ die 16 Werte in folgender Tabelle jeweils mit Wahrscheinlichkeit $2^{-4} = 1/16$ an:

| $u_1$ | $u_2$ | $u_3$ | $u_4$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $u_3 \oplus u_4 \oplus v_1 \oplus v_4$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

# Lineare Approximationen

## Beispiel (Fortsetzung)

- Um $\beta(\mathcal{U}_a \oplus \mathcal{V}_b)$ zu berechnen, genügt es, die Anzahl $L(a,b)$ der Eingaben $u \in \{0,1\}^l$ zu bestimmen, für die $f(u)_b = u_a$ ist
- Dann gilt $\Pr[\mathcal{U}_a \oplus \mathcal{V}_b = 0] = \Pr[\mathcal{U}_a = \mathcal{V}_b] = L(a,b)/16$ und somit

$$\beta(\mathcal{U}_a \oplus \mathcal{V}_b) = L(a,b)/16 - 1/2 = (L(a,b) - 8)/16$$

sowie

$$\gamma(a,b) = |2\beta(\mathcal{U}_a \oplus \mathcal{V}_b)|$$

- Für $a = 0011$ und $b = 1001$ gibt es z.B. $L(0011, 1001) = 2$ Eingaben $u$ mit $u_{0011} = S(u)_{1001}$ ($u = 0100$ und $u = 1001$), d.h. es gilt

$$\beta(\mathcal{U}_{0011} \oplus \mathcal{V}_{1001}) = (L(3,9) - 8)/16 = (2 - 8)/16 = -3/8$$

- Die Güte von $\mathcal{A}$ ist also $\gamma(0011, 1001) = |2\beta(\mathcal{U}_{0011} \oplus \mathcal{V}_{1001})| = 3/4$
- Die Tabelle auf der nächsten Folie enthält die Anzahlen $L(a,b)$ für bestimmte Werte von $a$ und $b$ (diese sind hexadezimal dargestellt)

# Lineare Approximationen

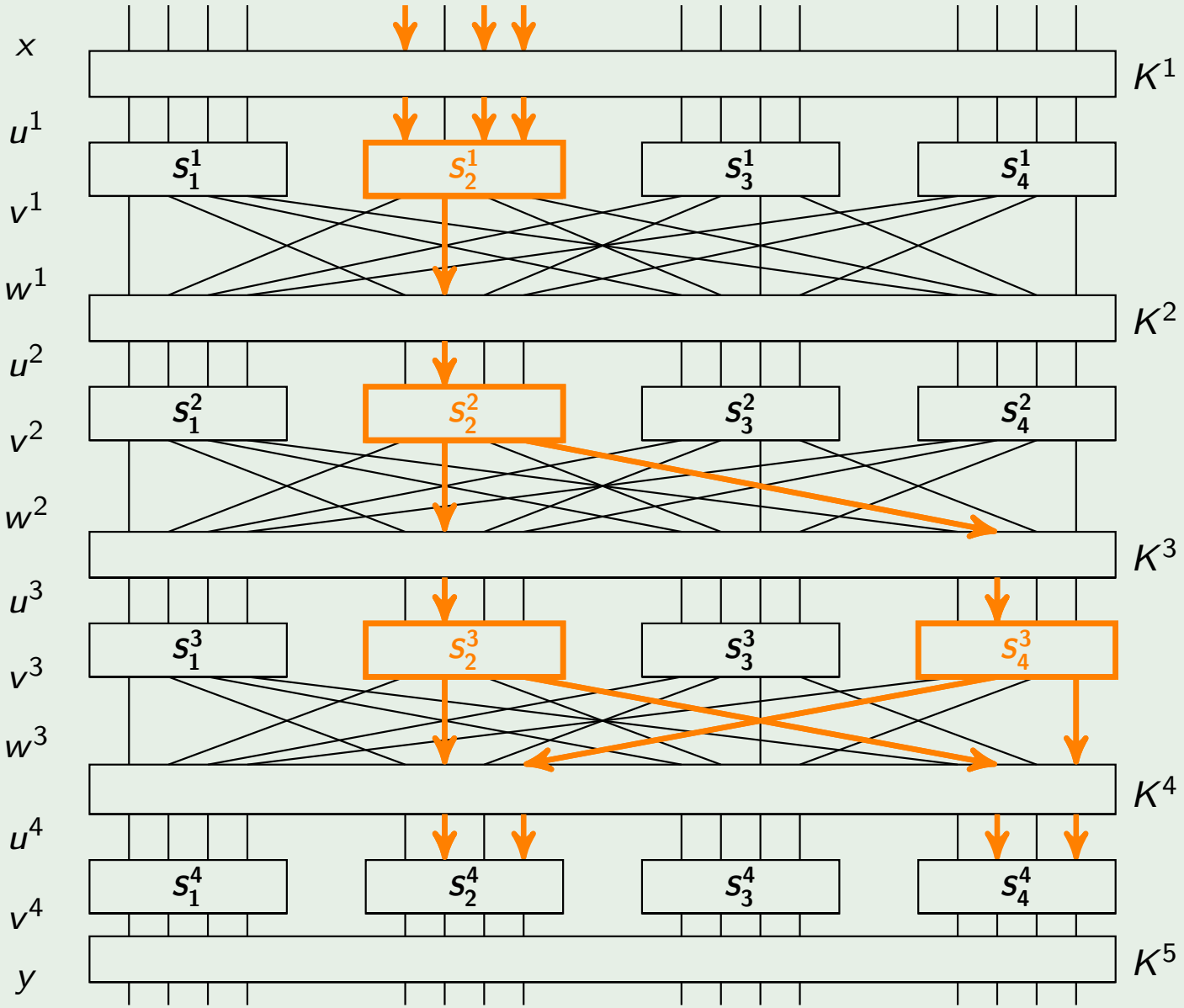
## Beispiel (Fortsetzung)

| | $b$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 16 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 1 | 8 | 8 | 6 | 6 | 8 | 8 | 6 | 14 | 10 | 10 | 8 | 8 | 10 | 10 | 8 | 8 |
| 2 | 8 | 8 | 6 | 6 | 8 | 8 | 6 | 6 | 8 | 8 | 10 | 10 | 8 | 8 | 2 | 10 |
| 3 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 10 | 2 | 6 | 6 | 10 | 10 | 6 | 6 |
| 4 | 8 | 10 | 8 | 6 | 6 | 4 | 6 | 8 | 8 | 6 | 8 | 10 | 10 | 4 | 10 | 8 |
| | | | | | | | | ⋮ | | | | | | | | |
| B | 8 | 12 | 8 | 4 | 12 | 8 | 12 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | | | | | | | | ⋮ | | | | | | | | |
| F | 8 | 6 | 4 | 6 | 6 | 8 | 10 | 8 | 8 | 6 | 12 | 6 | 6 | 8 | 10 | 8 |

Unser nächstes Ziel ist, geeignete lineare Approximationen für bestimmte S-Boxen im SPN zu finden, die sich zu einer linearen Approximation der Abbildung $x \mapsto u^4$ zusammenschalten lassen (siehe nächste Folie)

# Substitutions-Permutations-Netzwerke

## Beispiel (Schluss)



# Lineare Kryptoanalyse von SPNs

- Seien $K^1, \ldots, K^5$ die aus $K$ berechneten Rundenschlüssel (diese sind wie $K$ unbekannt, aber konstant)
- Unser erstes Ziel ist, eine lineare Approximation $\mathcal{A}$ an die Abbildung $x \mapsto u^4$ zu finden
- Diese Abbildung benutzt die vier Rundenschlüssel $K^1, \ldots, K^4$, um den Klartext $x$ in den Block $u^4$ zu überführen, welcher als Eingabe für die S-Boxen in Runde 4 dient, auf deren Ausgabe $v^4 = S(u^4)$ der Rundenschlüssel $K^5$ addiert wird, um den Kryptotext $y = v^4 \oplus K^5$ zu erhalten
- Für die Konstruktion von $\mathcal{A}$ verwenden wir die durch $(B, 4)$ und $(4, 5)$ beschriebenen linearen Approximationen

$$\mathcal{T} = \mathcal{U}_1 \oplus \mathcal{U}_3 \oplus \mathcal{U}_4 \oplus \mathcal{V}_2 \quad \text{und} \quad \mathcal{T}' = \mathcal{U}_2 \oplus \mathcal{V}_2 \oplus \mathcal{V}_4$$

an die S-Box $S$ mit den Bias-Werten

  - $\beta(\mathcal{T}) = (L(1011, 0100) - 8)/16 = (12 - 8)/16 = 1/4$ und
  - $\beta(\mathcal{T}') = (L(0100, 0101) - 8)/16 = (4 - 8)/16 = -1/4$

# Lineare Kryptoanalyse von SPNs

- Konkret verwenden wir $\mathcal{T}$ für die S-Box $S_2^1$:

  $$\mathcal{T}_1 = \mathcal{U}_5^1 \oplus \mathcal{U}_7^1 \oplus \mathcal{U}_8^1 \oplus \mathcal{V}_6^1$$

  und $\mathcal{T}'$ für die drei S-Boxen $S_2^2$, $S_2^3$ und $S_4^3$:

  $$\mathcal{T}_2 = \mathcal{U}_6^2 \oplus \mathcal{V}_6^2 \oplus \mathcal{V}_8^2,\ \mathcal{T}_3 = \mathcal{U}_6^3 \oplus \mathcal{V}_6^3 \oplus \mathcal{V}_8^3 \text{ und } \mathcal{T}_4 = \mathcal{U}_{14}^3 \oplus \mathcal{V}_{14}^3 \oplus \mathcal{V}_{16}^3$$

- Nun schalten wir die vier linearen Approximationen $\mathcal{T}_1, \ldots, \mathcal{T}_4$ an die S-Boxen $S_2^1$, $S_2^2$, $S_2^3$ und $S_4^3$ zur linearen Approximation

  $$\mathcal{A} = \underbrace{\mathcal{X}_5 \oplus \mathcal{X}_7 \oplus \mathcal{X}_8}_{\mathcal{X}_a \text{ für } a=\texttt{0x0B00}} \oplus \underbrace{\mathcal{U}_6^4 \oplus \mathcal{U}_8^4 \oplus \mathcal{U}_{14}^4 \oplus \mathcal{U}_{16}^4}_{\mathcal{U}_b^4 \text{ für } b=\texttt{0x0505}}$$

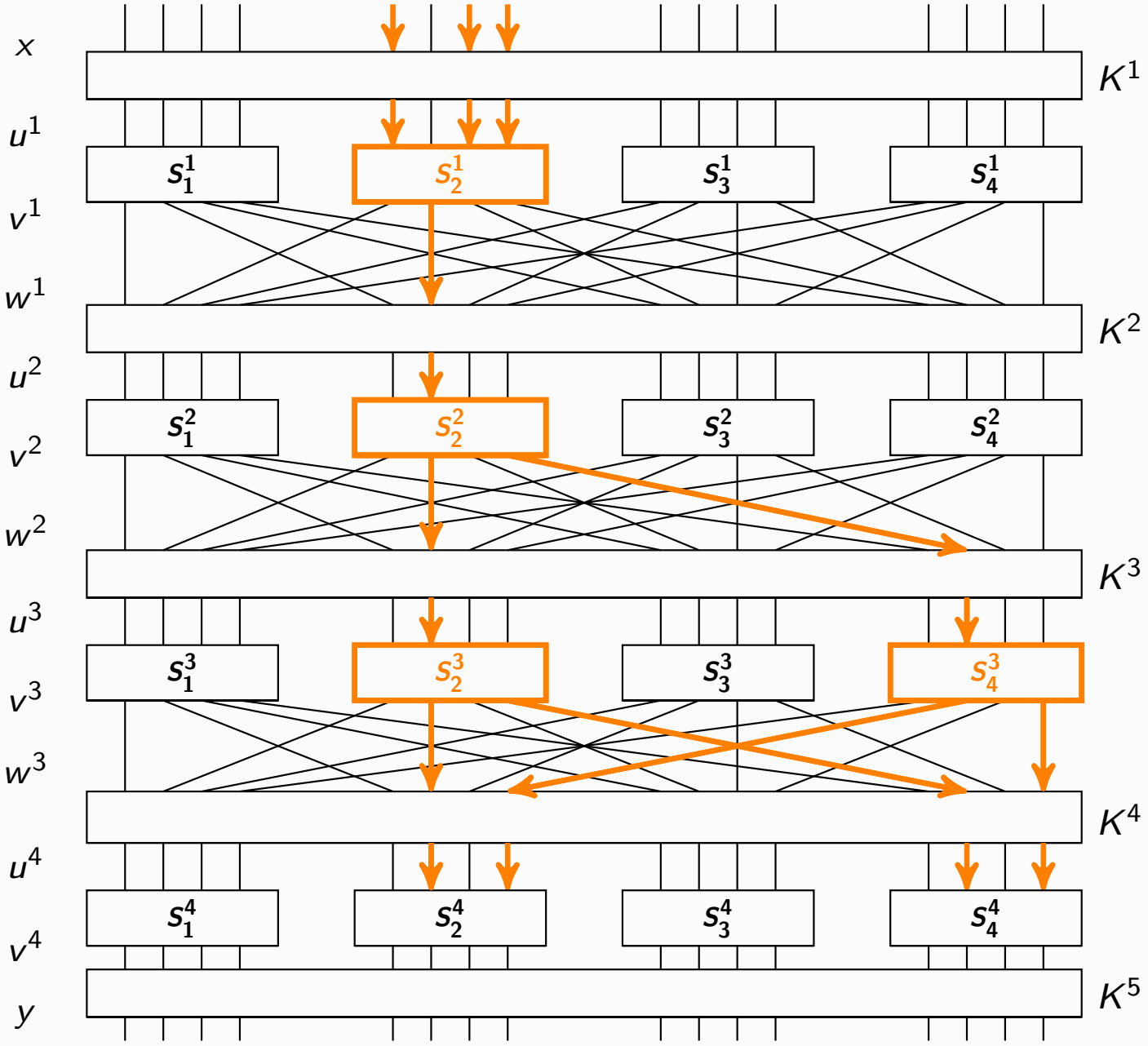
  an die Abbildung $x \mapsto u^4$ zusammen
- Dann gilt für ein Bit $c \in \{0, 1\}$ die Gleichung

  $$\mathcal{X}_a \oplus \mathcal{U}_b^4 = \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus c$$

An dieser Stelle ergeben sich nun folgende drei Fragen:

1. Warum gilt die Gleichung $\mathcal{X}_a \oplus \mathcal{U}_b^4 = \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus c$ ?
2. Wie gut ist die durch $(a, b)$ beschriebene lineare Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ an die Abbildung $x \mapsto u^4$ ?
3. Wie können wir die Approximation $\mathcal{A}$ benutzen, um einzelne Schlüsselbits zu bestimmen?

# Warum ist $\mathcal{X}_a \oplus \mathcal{U}_b^4 = \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus c$?



# Warum ist $\mathcal{X}_a \oplus \mathcal{U}_b^4 = \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus c$ ?

- Sei $c_1 = K_5^1 \oplus K_7^1 \oplus K_8^1$, $c_2 = K_6^2$, $c_3 = K_6^3 \oplus K_{14}^3$, $c_4 = K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$
- Wegen $\mathcal{T}_1 = \mathcal{U}_5^1 \oplus \mathcal{U}_7^1 \oplus \mathcal{U}_8^1 \oplus \mathcal{V}_6^1$, $\mathcal{T}_2 = \mathcal{U}_6^2 \oplus \mathcal{V}_6^2 \oplus \mathcal{V}_8^2$, $\mathcal{T}_3 = \mathcal{U}_6^3 \oplus \mathcal{V}_6^3 \oplus \mathcal{V}_8^3$ und $\mathcal{T}_4 = \mathcal{U}_{14}^3 \oplus \mathcal{V}_{14}^3 \oplus \mathcal{V}_{16}^3$ folgt dann

$$
\begin{aligned}
\mathcal{X}_a &= \mathcal{X}_5 \oplus \mathcal{X}_7 \oplus \mathcal{X}_8 \\
&= \mathcal{U}_5^1 \oplus \mathcal{U}_7^1 \oplus \mathcal{U}_8^1 \oplus c_1 \\
&= \mathcal{T}_1 \oplus \mathcal{V}_6^1 \oplus c_1 \\
&= \mathcal{T}_1 \oplus \mathcal{W}_6^1 \oplus c_1 \\
&= \mathcal{T}_1 \oplus \mathcal{U}_6^2 \oplus c_1 \oplus c_2 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{V}_6^2 \oplus \mathcal{V}_8^2 \oplus c_1 \oplus c_2 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{W}_6^2 \oplus \mathcal{W}_{14}^2 \oplus c_1 \oplus c_2 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{U}_6^3 \oplus \mathcal{U}_{14}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus \mathcal{V}_6^3 \oplus \mathcal{V}_8^3 \oplus \mathcal{V}_{14}^3 \oplus \mathcal{V}_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus \mathcal{W}_6^3 \oplus \mathcal{W}_8^3 \oplus \mathcal{W}_{14}^3 \oplus \mathcal{W}_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus \mathcal{T}_3 \oplus \mathcal{T}_4 \oplus \mathcal{U}_b^4 \oplus \underbrace{c_1 \oplus c_2 \oplus c_3 \oplus c_4}_{=: c}
\end{aligned}
$$

# Wie gut ist die Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ ?

- Wären die linearen Approximationen $\mathcal{T}_1, \ldots, \mathcal{T}_4$ an die vier ausgewählten S-Boxen $S_2^1$, $S_2^2$, $S_2^3$ und $S_4^3$ (diese werden auch als aktiv bezeichnet) unabhängig, so würde uns das Piling-up-Lemma (siehe nächste Folie) die folgenden Bias-Werte liefern:
  - $\beta(\mathcal{T}_1 \oplus \cdots \oplus \mathcal{T}_4) = 2^3(1/4)(-1/4)^3 = -1/32$ und
  - $\beta(\mathcal{A}) = (-1)^{c+1}/32$ bzw. $\gamma(\mathcal{A}) = 1/16$
- Sind nämlich $\mathcal{X}_1, \mathcal{X}_2$ unabhängige Zufallsvariablen mit Wertebereich $W(\mathcal{X}_i) = \{0, 1\}$ und Bias $\beta_i = \beta(\mathcal{X}_i)$, $i \in \{1, 2\}$, dann ist

$$\begin{aligned} \Pr[\mathcal{X}_1 \oplus \mathcal{X}_2 = 0] &= \Pr[\mathcal{X}_1 = \mathcal{X}_2 = 0] + \Pr[\mathcal{X}_1 = \mathcal{X}_2 = 1] \\ &= (1/2 + \beta_1)(1/2 + \beta_2) + (1/2 - \beta_1)(1/2 - \beta_2) \\ &= 1/2 + 2\beta_1\beta_2 \end{aligned}$$

- Es gilt also $\beta(\mathcal{X}_1 \oplus \mathcal{X}_2) = 2\beta_1\beta_2$
- Das Piling-up Lemma verallgemeinert dies auf die Summe von mehreren unabhängigen Zufallsvariablen

# Wie gut ist die Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ ?

**Lemma (Piling-up Lemma)**

Für unabhängige $\{0,1\}$-wertige Zufallsvariablen $\mathcal{X}_1, \ldots, \mathcal{X}_n$ mit Bias-Werten $\beta_i = \beta(\mathcal{X}_i)$ gilt

$$\beta(\mathcal{X}_1 \oplus \cdots \oplus \mathcal{X}_n) = 2^{n-1} \prod_{i=1}^{n} \beta_i$$

**Beweis.**

Wir führen den Beweis durch Induktion über $n$:

- Induktionsanfang ($n \leq 2$): Bereits bewiesen
- Induktionsschritt ($n \rightsquigarrow n+1$): Nach IV hat $\mathcal{Z} = \mathcal{X}_1 \oplus \cdots \oplus \mathcal{X}_n$ den Bias

$$\beta(\mathcal{Z}) = 2^{n-1}\beta(\mathcal{X}_1) \cdots \beta(\mathcal{X}_n) = 2^{n-1}\beta_1 \cdots \beta_n$$

und daher folgt

$$\begin{aligned} \beta(\mathcal{X}_1 \oplus \cdots \oplus \mathcal{X}_{n+1}) &= \beta(\mathcal{Z} \oplus \mathcal{X}_{n+1}) = 2\beta(\mathcal{Z})\beta_{n+1} \\ &= 2^n \beta_1 \cdots \beta_{n+1} \end{aligned}$$

□

# Wie gut ist die Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ ?

## Beispiel

- Seien $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ unabhängige Zufallsvariablen mit $\beta(\mathcal{X}_i) = 1/4$ für $i = 1, 2, 3$
- Dann liefert das Piling-up Lemma die Bias-Werte $\beta(\mathcal{X}_i \oplus \mathcal{X}_j) = 1/8$ für $1 \leq i < j \leq 3$
- Da die beiden Zufallsvariablen $\mathcal{Y} = \mathcal{X}_1 \oplus \mathcal{X}_2$ und $\mathcal{Z} = \mathcal{X}_2 \oplus \mathcal{X}_3$ nicht unabhängig sind, liefert das Piling-up-Lemma nicht den richtigen Bias-Wert für $\beta(\mathcal{Y} \oplus \mathcal{Z})$
- Damit würden wir nämlich den Wert $2 \cdot 1/8 \cdot 1/8 = 1/32$ erhalten, wogegen

$$\beta(\mathcal{Y} \oplus \mathcal{Z}) = \beta(\mathcal{X}_1 \oplus \mathcal{X}_2 \oplus \mathcal{X}_2 \oplus \mathcal{X}_3) = \beta(\mathcal{X}_1 \oplus \mathcal{X}_3) = 1/8$$

ist

# Wie gut ist die Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ ?

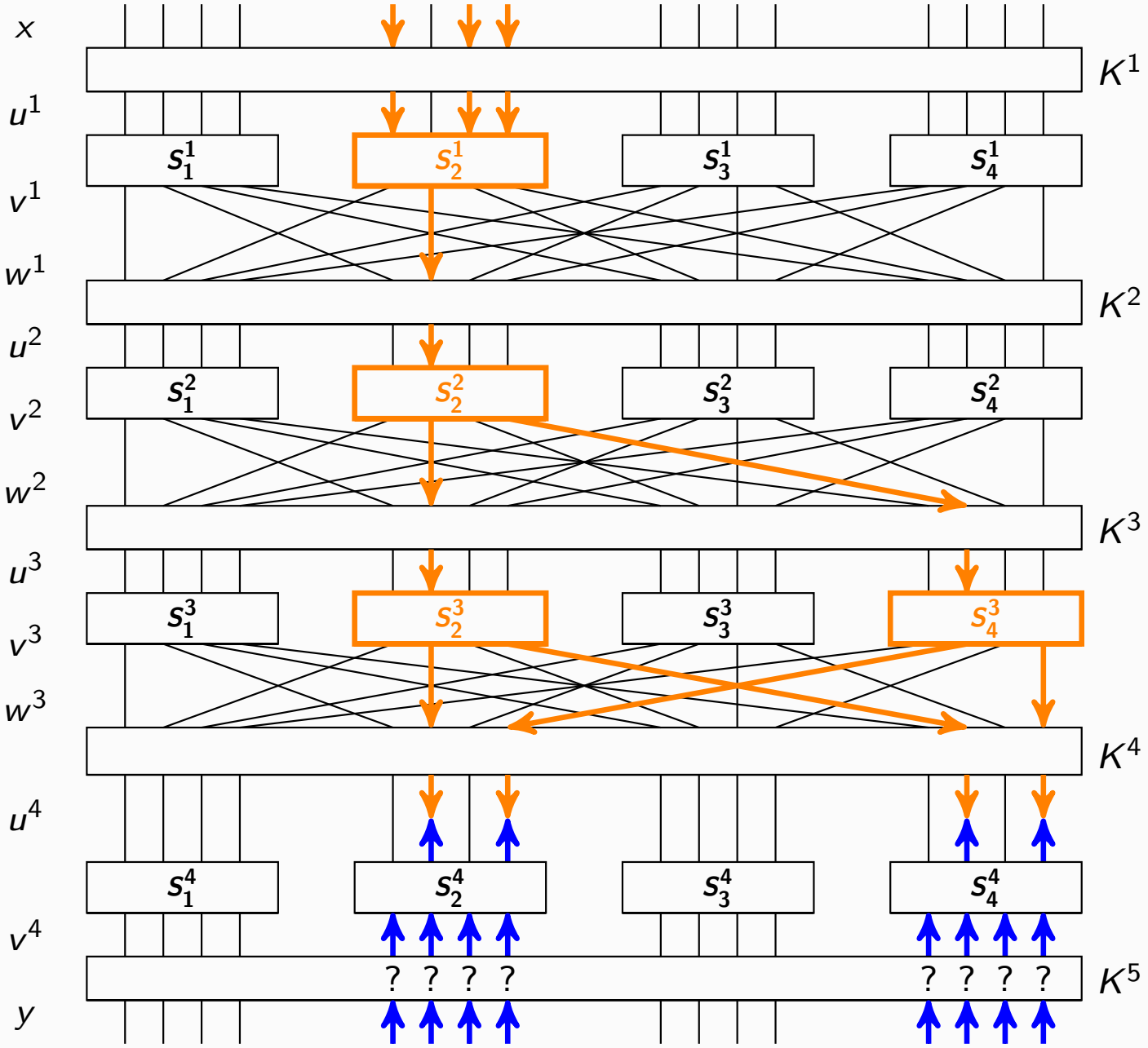
- Zwar sind die Zufallsvariablen $\mathcal{T}_i$, aus denen eine lineare Approximation $\mathcal{X}_a \oplus \mathcal{U}_b^N = \mathcal{T}_1 \oplus \cdots \oplus \mathcal{T}_k \oplus c$ an die Abbildung $x \mapsto u^N$ gebildet wird, in der Regel nicht unabhängig
- Dennoch zeigt sich in praktischen Anwendungen, dass der tatsächliche Bias von $\mathcal{T}_1 \oplus \cdots \oplus \mathcal{T}_k$ meist nicht sehr von dem hypothetischen Wert $2^{k-1} \prod_{i=1}^{k} \beta(\mathcal{T}_i)$ unterscheidet, welcher sich aus dem Piling-up Lemma ergibt
- Daher können wir in unserem Beispiel

$$\beta(\mathcal{T}_1 \oplus \cdots \oplus \mathcal{T}_4) \approx 2^3(1/4)(-1/4)^3 = -1/32$$

bzw. $\gamma(\mathcal{A}) \approx 1/16$ annehmen

# Wie lassen sich mit $\mathcal{A}$ einzelne Schlüsselbits bestimmen?





# Wie lassen sich mit $\mathcal{A}$ einzelne Schlüsselbits bestimmen?



- Wir nehmen an, dass eine Menge $M$ von Klartext-Kryptotext-Paaren bekannt ist und damit einzelne Schlüsselbits berechnet werden sollen
- Wir betrachten zuerst den (für den Gegner günstigen) Fall, dass die lineare Approximation $\mathcal{X}_a \oplus \mathcal{U}_b^4$ an $x \mapsto u^4$ die Güte $\gamma(\mathcal{X}_a \oplus \mathcal{U}_b^4) = 1$ hat (dies ist z.B. der Fall, wenn die S-Box $S$ affin ist, d.h. $S(u) = uA \oplus w$)
- In diesem Fall führt jede Eingabe $x$ auf einen Vektor $u^4$ mit $x_a \oplus u_b^4 = c$, wobei das Bit $c$ nur von $K$ und nicht von $x$ abhängt
- Da $y$ und $S^{-1}$ bekannt sind, können wir für jeden Kandidaten $L$ für $K^5$ den zugehörigen $u^4$-Block $u^4(y, L) = S^{-1}(L \oplus y)$ berechnen
- Dann wird der Kandidat $L = K^5$ den Test $u^4(y, L)_b = x_a$ für alle $(x, y) \in M$ bestehen (falls $c = 0$ ist) oder für alle $(x, y) \in M$ nicht bestehen (falls $c = 1$ ist)

# Wie lassen sich mit $\mathcal{A}$ einzelne Schlüsselbits bestimmen?



- Dann wird der Kandidat $L = K^5$ den Test $u^4(y, L)_b = x_a$ für alle $(x, y) \in M$ bestehen (falls $c = 0$ ist) oder für alle $(x, y) \in M$ nicht bestehen (falls $c = 1$ ist)
- Zudem hängt $u^4(y, L)_b$ wegen $b = \texttt{0x0505}$ nur vom 2. und 4. Teilblock von $u^4(y, L)$ ab, welche wiederum nur von den Teilblöcken $I = L_{(2)}$ bzw. $J = L_{(4)}$ von $L$ abhängen, d.h. $u^4(y, L)_b = u^4(y, I, J)_b$
- Daher genügt es, den Test für alle Kandidaten $(I, J)$ für $(K^5_{(2)}, K^5_{(4)})$ und alle $(x, y) \in M$ auszuführen
- Der richtige Kandidat $(I, J) = (K^5_{(2)}, K^5_{(4)})$ besteht dann den Test $x_a = u^4(y, I, J)_b$ entweder für jedes oder für kein Paar $(x, y)$ in $M$
- Dagegen besteht ein falscher Kandidat $(I, J) \neq (K^5_{(2)}, K^5_{(4)})$ diesen Test für etwa die Hälfte aller Paare $(x, y)$ in $M$
- Daher können wir den richtigen Kandidaten daran erkennen, dass er als einziger den Test für jedes (bzw. kein) Paar $(x, y)$ in $M$ besteht

# Wie lassen sich mit $\mathcal{A}$ einzelne Schlüsselbits bestimmen?



- Nun zum Fall, dass die Güte der linearen Approximation $\mathcal{A} = \mathcal{X}_a \oplus \mathcal{U}_b^4$ an die Abbildung $x \mapsto u^4$ zwar nicht gleich 1 ist, aber auch nicht sehr klein ist
- Ist $\beta$ der Bias von $\mathcal{A}$, so besteht der gesuchte Subkey $(K_{(2)}^5, K_{(4)}^5)$ ungefähr einen Anteil von $(1/2 + \beta)$ der durchgeführten Tests, sofern die benutzten Klartext-Kryptotext-Paare hinreichend zufällig sind
- Dagegen besteht ein falscher Kandidat $(I, J) \neq (K_{(2)}^5, K_{(4)}^5)$ nur etwa die Hälfte aller Tests
- Falls also $M$ eine hinreichend repräsentative Menge von genügend vielen Klartext-Kryptotext-Paaren $(x, y)$ ist, können wir den richtigen Subkey daran erkennen, dass die Anzahl der von ihm bestandenen Tests am weitesten von $|M|/2$ abweicht

## Algorithmus LinearAttack

- Der Algorithmus LinearAttack (siehe nächste Folie) ermittelt für jeden Subkey-Kandidaten $(I, J)$ die Anzahl $\alpha(I, J)$ der Paare $(x, y) \in M$, für die $(I, J)$ den Gleichheitstest $x_a = u_b^4(y, I, J)$ besteht
- Ausgegeben wird derjenige Kandidat $(I, J)$, der den Betrag $\hat{\alpha}(I, J)$ der Abweichung der Zahl $\alpha(I, J)$ von $|M|/2$ maximiert
- Im Allgemeinen werden für eine erfolgreiche lineare Attacke circa $t \approx c\beta^{-2}$ Klartext-Kryptotext-Paare benötigt
- Dabei ist $c$ eine „kleine“ Konstante (im aktuellen Beispiel reichen $t \approx 8000$ Paare; d.h. $c \approx 8$, da $\beta^{-2} \approx 1024$ ist)

**Algorithmus** LinearAttack

1 $max := -1$
2 **for** $(I, J) := (0,0)$ **to** (F,F) **do**
3 $\quad \alpha(I, J) := 0$
4 $\quad$ **for** each $(x, y) \in M$ **do**
5 $\quad\quad v^4_{(2)} := I \oplus y_{(2)}$
6 $\quad\quad v^4_{(4)} := J \oplus y_{(4)}$
7 $\quad\quad u^4_{(2)} := S^{-1}(v^4_{(2)})$
8 $\quad\quad u^4_{(4)} := S^{-1}(v^4_{(4)})$
9 $\quad\quad$ **if** $x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16} = 0$ **then**
10 $\quad\quad\quad \alpha(I, J) := \alpha(I, J) + 1$
11 $\quad \hat{\alpha}(I, J) := |\alpha(I, J) - {}^t/_2|$
12 $\quad$ **if** $\hat{\alpha}(I, J) > max$ **then** $(max, maxkey) := (\hat{\alpha}(I, J), (I, J))$
13 **output**(maxkey)

# Differentielle Kryptoanalyse von SPNs

- Wie die lineare hat auch die differentielle Kryptoanalyse das Ziel, den unbekannten Schlüssel $K$ zu finden
- Für die Durchführung wird jedoch frei wählbarer Klartext benötigt
- Genauer basiert der Angriff auf einer Menge $M$ von $t$ Doppelpaaren $(x, x^*, y, y^*)$ mit der Eigenschaft, dass $E_K(x) = y$ und $E_K(x^*) = y^*$ ist und alle Klartext-Paare $(x, x^*)$ die gleiche Differenz $x' = x \oplus x^*$ bilden

## Definition

- Seien $u, u^* \in \{0,1\}^l$ Eingaben für eine S-Box $S : \{0,1\}^l \to \{0,1\}^{l'}$ und seien $v = S(u)$ und $v^* = S(u^*)$ die zugehörigen Ausgaben
- Dann heißt $u' = u \oplus u^*$ die Eingabedifferenz (input-xor) und $v' = S(u) \oplus S(u^*)$ die Ausgabedifferenz (output-xor) von $(u, u^*)$
- Für eine vorgegebene Eingabedifferenz $a' \in \{0,1\}^l$ sei weiter

$$\Delta(a') = \{(u, u^*) \mid u \oplus u^* = a'\} = \{(u, u \oplus a') \mid u \in \{0,1\}^l\}$$

die Menge aller Eingabepaare, die die Differenz $a'$ realisieren

# Differentielle Kryptoanalyse von SPNs

- Berechnen wir für alle Paare $(u, u^*) \in \Delta(a')$ die zugehörigen Ausgabedifferenzen, so verteilen sich diese auf die $2^{l'}$ möglichen Werte in $\{0,1\}^{l'}$
- Man beachte, dass im Fall einer affinen S-Box $S(u) = uA \oplus w$ alle Paare $(u, u^*) \in \Delta(a')$ auf dieselbe Ausgabedifferenz $b' = a'A$ führen:

$$S(u) \oplus S(u^*) = (uA \oplus w) \oplus (u^*A \oplus w) = (u \oplus u^*)A = u'A = a'A$$

(hierbei ist $A$ eine Matrix in $\{0,1\}^{l \times l'}$ und $w$ ein Vektor in $\{0,1\}^{l'}$)

- Ist $S$ nicht affin, können die $2^l$ Eingabepaare $(u, u^*) \in \Delta(a')$ zu unterschiedlichen Ausgabedifferenzen $v' = S(u) \oplus S(u^*)$ führen, da in diesem Fall die Ausgabedifferenz nicht nur von der Eingabedifferenz $u'$, sondern vom Eingabepaar $(u, u^*)$ abhängt
- Um einer differentiellen Kryptoanalyse widerstehen zu können, sollten die auftretenden Ausgabedifferenzen möglichst gleichmäßig verteilt sein

# Differentielle Kryptoanalyse von SPNs

## Definition

- Ein Differential für eine S-Box $S : \{0,1\}^l \to \{0,1\}^{l'}$ ist ein Paar $(a', b') \in \{0,1\}^l \times \{0,1\}^{l'}$
- Dabei heißt $a'$ die Eingabe- und $b'$ die Ausgabedifferenz des Differentials
- Die Anzahl der Eingabepaare $(u, u^*)$, die die Eingabedifferenz $a'$ in die Ausgabedifferenz $b'$ überführen, bezeichnen wir mit $D(a', b')$, d.h.

$$D(a', b') = |\{(u, u^*) \in \Delta(a') \mid S(u) \oplus S(u^*) = b'\}|$$

- Der Weitergabequotient (engl. propagation ratio) von $S$ für ein Differential $(a', b')$ ist der Anteil

$$\rho_S(a', b') = \frac{D(a', b')}{|\Delta(a')|} = 2^{-l} D(a', b')$$

der Eingabepaare $(u, u^*)$ in $\Delta(a')$, die auf die Ausgabedifferenz $b'$ führen

- Der Weitergabequotient $\rho_S(a', b')$ ist die bedingte Wahrscheinlichkeit

$$\Pr[\mathcal{V}' = b' | \mathcal{U}' = a'] = \underbrace{\Pr[\mathcal{V}' = b' \wedge \mathcal{U}' = a']}_{D(a',b')/2^{2l}} / \underbrace{\Pr[\mathcal{U}' = a']}_{|\Delta(a')|/2^{2l}}$$

$$= D(a', b')/|\Delta(a')|$$

dass zwei zufällig mit der Eingabedifferenz $\mathcal{U}' = \mathcal{U} \oplus \mathcal{U}^* = a'$ gewählte Eingaben $\mathcal{U}$ und $\mathcal{U}^*$ die Ausgabedifferenz $\mathcal{V}' = S(\mathcal{U}) \oplus S(\mathcal{U}^*) = b'$ bilden

# Differentielle Kryptoanalyse von SPNs

## Beispiel

- Betrachten wir die S-Box $S : \{0,1\}^4 \to \{0,1\}^4$ aus obigem Beispiel, so erhalten wir für $a' = 1011$ folgende Menge von Eingabepaaren

$$\Delta(a') = \{(0000, 1011), \ldots, (1111, 0100)\},$$

- Diese führen auf die folgenden Ausgabedifferenzen $v' = \underbrace{S(u)}_{v} \oplus \underbrace{S(u^*)}_{v*}$:

| $u$ | $u^*$ | $v$ | $v^*$ | $v'$ |
|---|---|---|---|---|
| 0000 | 1011 | 1110 | 1100 | 0010 |
| 0001 | 1010 | 0100 | 0110 | 0010 |
| 0010 | 1001 | 1101 | 1010 | 0111 |
| 0011 | 1000 | 0001 | 0011 | 0010 |
| 0100 | 1111 | 0010 | 0111 | 0101 |
| 0101 | 1110 | 1111 | 0000 | 1111 |
| 0110 | 1101 | 1011 | 1001 | 0010 |
| 0111 | 1100 | 1000 | 0101 | 1101 |

| $u$ | $u^*$ | $v$ | $v^*$ | $v'$ |
|---|---|---|---|---|
| 1000 | 0011 | 0011 | 0001 | 0010 |
| 1001 | 0010 | 1010 | 1101 | 0111 |
| 1010 | 0001 | 0110 | 0100 | 0010 |
| 1011 | 0000 | 1100 | 1110 | 0010 |
| 1100 | 0111 | 0101 | 1000 | 1101 |
| 1101 | 0110 | 1001 | 1011 | 0010 |
| 1110 | 0101 | 0000 | 1111 | 1111 |
| 1111 | 0100 | 0111 | 0010 | 0101 |

## Beispiel (Fortsetzung)

- Die Ausgabedifferenz $b' = 0010$ kommt also $D(a', 0010) = 8$ Mal vor, während die Differenzen 0101, 0111, 1101 und 1111 je zwei Mal und die übrigen Werte überhaupt nicht vorkommen (siehe Zeile B in nachfolgender Tabelle)
- Führen wir diese Berechnungen für jede der $2^4 = 16$ Eingabedifferenzen $a' \in \{0,1\}^4$ aus, so erhalten wir die folgenden Werte für die Häufigkeiten $D(a', b')$ der Ausgabedifferenz $b'$ bei Eingabedifferenz $a'$ ($a'$ und $b'$ sind hexadezimal dargestellt)

# Differentielle Kryptoanalyse von SPNs

## Beispiel (Schluss)

- Die Tabelle zeigt die Häufigkeiten $D(a', b')$ der Ausgabedifferenzen $b'$ der S-Box $S$ für eine Auswahl von Eingabedifferenzen $a' \in \{0,1\}^4$

| | $b'$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a'$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 4 | 0 | 4 | 2 | 0 | 0 |
| 2 | 0 | 0 | 0 | 2 | 0 | 6 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| 3 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 0 | 0 | 4 |
| ⋮ | | | | | | | | ⋮ | | | | | | | | |
| B | 0 | 0 | 8 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 |
| ⋮ | | | | | | | | ⋮ | | | | | | | | |
| F | 0 | 2 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 2 | 0 |

◁

# Differentielle Kryptoanalyse von SPNs

- Wir versuchen nun, für ausgewählte S-Boxen in Runde 1 bis $N-1$ (diese werden wieder aktiv genannt) Differentiale $(a', b')$ zu finden, so dass die (permutierte) Ausgabedifferenz dieser Differentiale in Runde $i$ mit der Eingabedifferenz in Runde $i+1$ übereinstimmt (siehe nächste Folie)
- Die ausgewählten Differentiale der aktiven S-Boxen lassen sich zu einer sog. Differentialspur (differential trail) zusammensetzen
- Falls die einzelnen aktiven S-Boxen $S_i$ die vorgegebenen Differenzen unabhängig gemäß den ausgewählten Differentialen $(a'_i, b'_i)$ weitergeben, folgt ein zufällig gewähltes Klartextpaar $(x, x^*) \in \Delta(a')$ dieser Spur mit der Wahrscheinlichkeit $\prod_i \rho(a'_i, b'_i)$
- Auch wenn die Unabhängigkeit meist nicht gegeben ist, weicht der tatsächliche Wert dieser Wahrscheinlichkeit in konkreten Anwendungen nur wenig von diesem hypothetischen Wert ab

# Differentielle Kryptoanalyse von SPNs

## Beispiel

- Betrachten wir das SPN $SP$ aus dem vorigen Beispiel, so lassen sich folgende Differentiale zu einer Spur kombinieren:
  - für $S_2^1$ das Differential $(1011, 0010) = (\mathtt{B}, 2)$ mit $\rho(\mathtt{B}, 2) = 1/2$
  - für $S_3^2$ das Differential $(0100, 0110) = (4, 6)$ mit $\rho(4, 6) = 3/8$ und
  - für $S_2^3$ und $S_3^3$ das Differential $(0010, 0101) = (2, 5)$ mit $\rho(2, 5) = 3/8$
- Gemäß dieser Spur führt also die Klartextdifferenz

  $$x' = 0000\,1011\,0000\,0000$$

  mit hypothetischer Wahrscheinlichkeit $1/2(3/8)^3$ auf die Differenz

  $$(v^3)' = 0000\,0101\,0101\,0000,$$

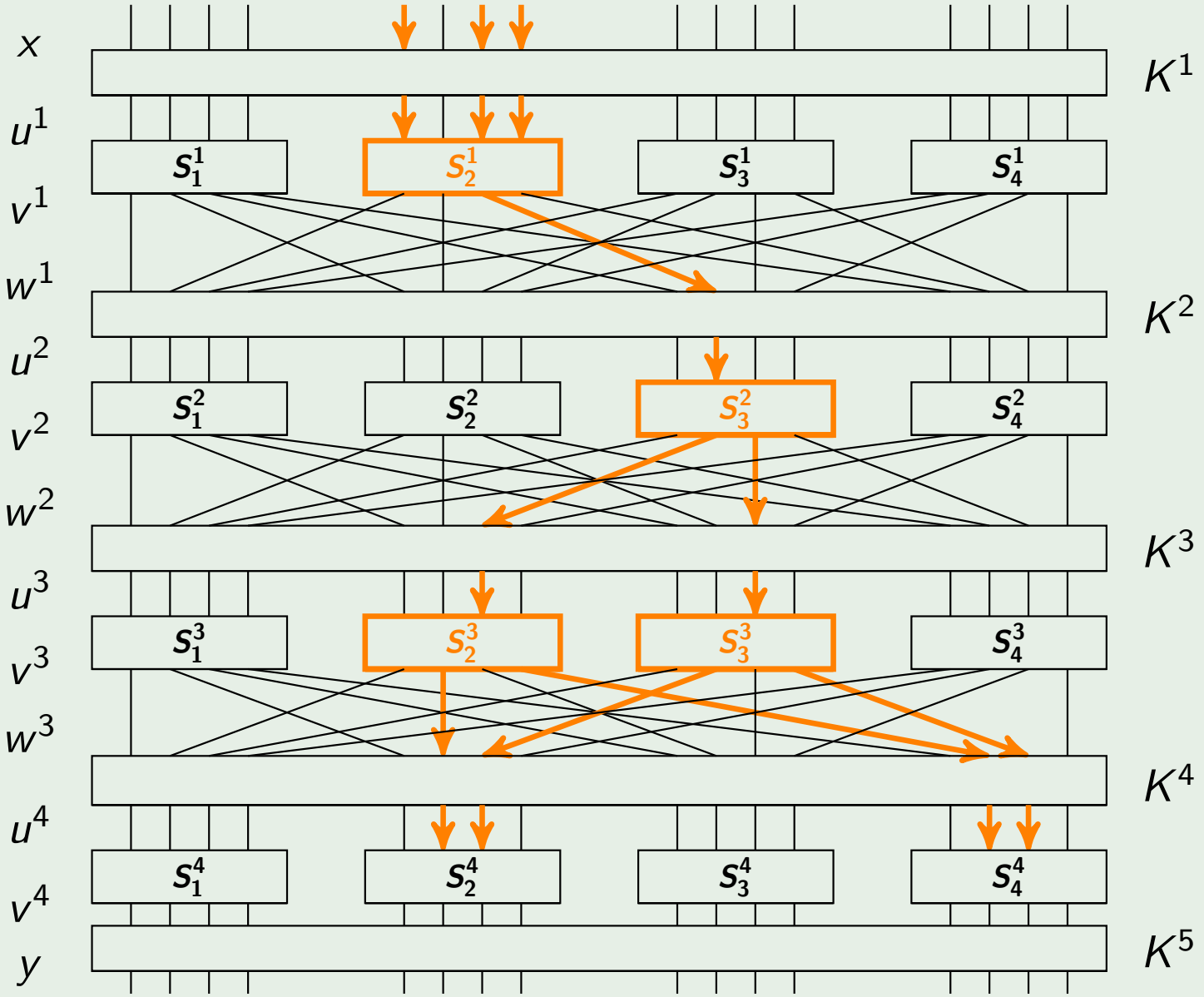
  und diese führt mit Wahrscheinlichkeit 1 auf die Differenz

  $$(u^4)' = 0000\,0110\,0000\,0110$$

- Das Differential $(a', b') = (0000\,1011\,0000\,0000, 0000\,0110\,0000\,0110)$ für die Abbildung $x \mapsto u^4$ hat also den hypothetischen Weitergabequotienten $\rho(a', b') = 1/2(3/8)^3 = 27/1024 \approx 0,026$

# Differentielle Kryptoanalyse von SPNs

## Beispiel (Fortsetzung)



# Differentielle Kryptoanalyse von SPNs

- Sei nun $(a', b')$ ein Differential für die Abbildung $x \mapsto u^N$ mit einem hypothetischen Weitergabequotienten $\rho = \rho(a', b')$
- Weiter sei $M$ eine Menge von $t$ Doppelpaaren $(x, x^*, y, y^*)$, die
  - alle mit dem gleichen unbekannten Schlüssel $K$ erzeugt wurden und
  - zusätzlich die Bedingung $x' = x \oplus x^* = a'$ erfüllen
- Dann werden für ca. ein $\rho$-Anteil dieser Doppelpaare die Klartexte $x$ und $x^*$ in der letzten Runde auf Blöcke $u^N$ und $(u^N)^*$ mit

  $$(u^N)' = u^N \oplus (u^N)^* = b'$$

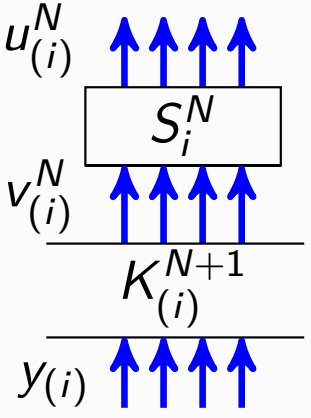
  abgebildet (solche Doppelpaare werden als richtig bezeichnet)
- Ein Teil der falschen Doppelpaare lässt sich daran erkennen, dass die Kryptotext-Differenzen $y'$ nicht die erwarteten $0'$-Blöcke aufweisen (im aktuellen Beispiel sind dies die Blöcke $y'_{(1)}$ und $y'_{(3)}$)
- Es empfiehlt sich, diese offensichtlich falschen Doppelpaare auszufiltern, da sie (wie alle falschen Doppelpaare) nur „Hintergrundrauschen“ erzeugen und somit die Bestimmung des Schlüssels behindern

# Differentielle Kryptoanalyse von SPNs

## Beobachtung

- Sei $S_i^N$ eine aktive S-Box in Runde $N$ und sei $(x, x^*, y, y^*) \in M$
- Dann können wir für jeden Kandidaten $J$ für den Subkey $K_{(i)}^{N+1}$ von $K^{N+1}$ die Ausgabe
  - $v_{(i)}^N(y, J) = y_{(i)} \oplus J$ von $S_i^N$ und die Eingabe
  - $u_{(i)}^N(y, J) = S^{-1}(v_{(i)}^N) = S^{-1}(y_{(i)} \oplus J)$ von $S_i^N$

  zurückrechnen




- Falls die S-Box $S_i^N$ nicht affin ist, hängt die aus den Kryptotexten $y$ und $y^*$ zurückgerechnete Eingabedifferenz

$$(u_{(i)}^N)'(y, y^*, J) = S^{-1}(y_{(i)} \oplus J) \oplus S^{-1}(y_{(i)}^* \oplus J)$$

  von dem Subkey-Kandidaten $J$ ab

# Differentielle Kryptoanalyse von SPNs

## Beobachtung (Fortsetzung)

- Da jedes richtige Doppelpaar $(x, x^*, y, y^*)$ auf die Differenz $(u^N)' = b'$ führt, erfüllt der richtige Kandidat $J = K_{(i)}^{N+1}$ den Test

$$S^{-1}(y_{(i)} \oplus J) \oplus S^{-1}(y^*_{(i)} \oplus J) = b'_{(i)} \qquad (*)$$

für alle richtigen Doppelpaare $(x, x^*, y, y^*)$

- Erfüllt ein Subkey-Kandidat $J$ die Gleichung $(*)$ für ein Doppelpaar $(x, x^*, y, y^*)$, so sagen wir auch, $J$ ist mit $(x, x^*, y, y^*)$ konsistent
- Gemäß obiger Beobachtung ist der gesuchte Subkey $J = K_{(i)}^{N+1}$ mit allen richtigen Doppelpaaren konsistent
- Dagegen ist ein Subkey $J \neq K_{(i)}^{N+1}$ nur mit einem Teil der richtigen Doppelpaare konsistent
- Ist $M$ hinreichend groß, so lässt sich der gesuchte Subkey daran erkennen, dass er mit mehr Doppelpaaren als jeder andere Kandidat konsistent ist

# Differentielle Kryptoanalyse von SPNs

Wir führen nun mit der Spur aus obigem Beispiel einen Angriff mittels differentieller Analyse auf das SPN $SP$ durch

## Beispiel

- Der Algorithmus DifferentialAttack (siehe nächste Folie) erhält eine Menge $M$ von Doppelpaaren $(x, x^*, y, y^*)$ mit $x \oplus x^* = a'$
- Zuerst blendet er alle offensichtlich falschen Doppelpaare aus (Zeile 1)
- Anschließend bestimmt er für jeden Kandidaten $(I, J)$ für den Subkey $(K^5_{(2)}, K^5_{(4)})$ die Anzahl $\kappa(I, J)$ aller mit $(I, J)$ konsistenten Doppelpaare
- Ausgegeben wird der Kandidat $(I, J)$ mit dem größten $\kappa$-Wert
- Im Allgemeinen werden für eine erfolgreiche differentielle Attacke circa $t \approx c\rho^{-1}$ Klartext-Kryptotext-Doppelpaare benötigt
- Dabei ist $\rho$ der Weitergabequotient der Spur und $c$ eine Konstante (im Beispiel reichen $t \approx 80$ Doppelpaare, wobei $\rho^{-1} \approx 38$ ist, d.h. $c \approx 2$) $\triangleleft$

**Algorithmus** DifferentialAttack

1 $M := \{(x, x^*, y, y^*) \in M \mid y_{(1)} = y^*_{(1)} \wedge y_{(3)} = y^*_{(3)}\}$
2 $max := -1$
3 **for** $(I, J) :=$ (0,0) **to** (F,F) **do**
4 $\quad \kappa(I, J) := 0$
5 $\quad$ **for** each $(x, x^*, y, y^*) \in M$ **do**
6 $\quad\quad v^4_{(2)} := I \oplus y_{(2)}; \quad v^4_{(4)} := J \oplus y_{(4)}$
7 $\quad\quad u^4_{(2)} := S^{-1}(v^4_{(2)}); \quad u^4_{(4)} := S^{-1}(v^4_{(4)})$
8 $\quad\quad (v^4_{(2)})^* := I \oplus y^*_{(2)}; \quad (v^4_{(4)})^* := J \oplus y^*_{(4)}$
9 $\quad\quad (u^4_{(2)})^* := S^{-1}((v^4_{(2)})^*); \quad (u^4_{(4)})^* := S^{-1}((v^4_{(4)})^*)$
10 $\quad\quad (u^4_{(2)})' := u^4_{(2)} \oplus (u^4_{(2)})^*; \quad (u^4_{(4)})' := u^4_{(4)} \oplus (u^4_{(4)})^*$
11 $\quad\quad$ **if** $(u^4_{(2)})' = 0110 \wedge (u^4_{(4)})' = 0110$ **then** $\kappa(I, J) := \kappa(I, J) + 1$
12 $\quad$ **if** $\kappa(I, J) > max$ **then** $(max, maxkey) := (\kappa(I, J), (I, J))$
13 **output**(maxkey)

# Der Data Encryption Standard (DES)

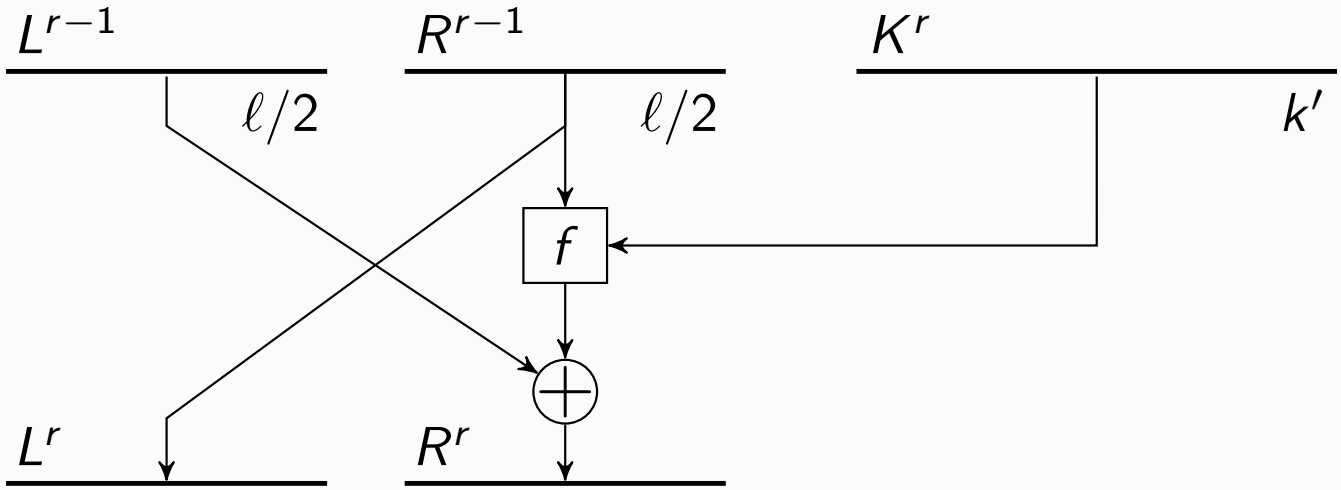
- Der DES wurde von IBM im Zuge einer Ausschreibung des NBS (National Bureau of Standards; heute National Institute of Standards and Technology, NIST) als ein Nachfolger von Lucifer entwickelt
- Er wurde 1975 veröffentlicht, und 1977 als Verschlüsselungsstandard der US-Regierung für nicht geheime Nachrichten genormt
- Obwohl DES ursprünglich nur für einen Zeitraum von 10 bis 15 Jahren als Standard dienen sollte, wurde er circa alle 5 Jahre (zuletzt 1999) überprüft und als Standard fortgeschrieben
- Bereits 1997 veröffentlichte das NIST eine Ausschreibung für den AES (Advanced Encryption Standard) genannten Nachfolger des DES
- Nach einer mehrjährigen Auswahlprozedur wurde im November 2001 der Rijndael-Algorithmus als AES genormt und im Mai 2002 wurde DES von AES als Standard abgelöst
- Allerdings wurde Triple DES (auch TDES oder 3DES genannt) vom NIST als Standard bis 2030 fortgeschrieben

# Aufbau einer Feistel-Chiffre

- Die Rundenfunktion $g$ einer Feistel-Chiffre berechnet das Zwischenergebnis $w^r = g(K^r, w^{r-1}) \in \{0,1\}^\ell$ in Runde $r$ gemäß der Vorschrift

  $$L^r = R^{r-1} \quad \text{und} \quad R^r = L^{r-1} \oplus f(K^r, R^{r-1})$$

  aus den beiden Hälften $L^{r-1}$ und $R^{r-1}$ von $w^{r-1} = L^{r-1}R^{r-1} \in \{0,1\}^\ell$:



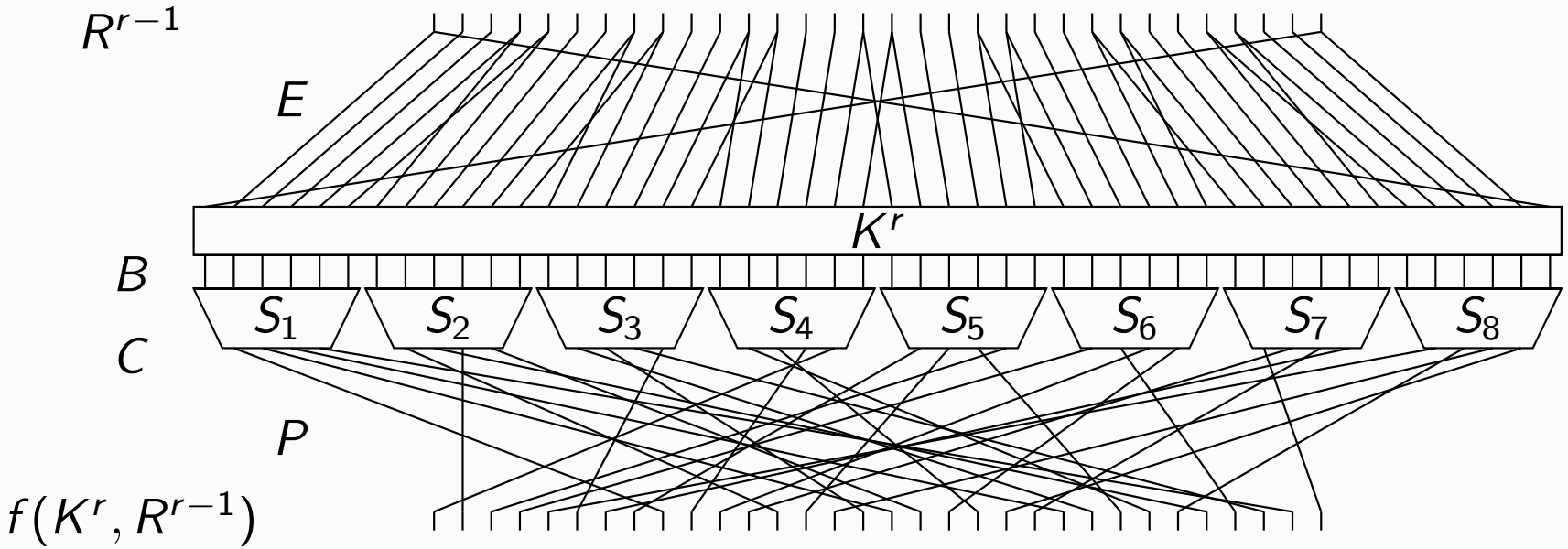

- Hierbei ist $f : \{0,1\}^{k'} \times \{0,1\}^{\ell/2} \to \{0,1\}^{\ell/2}$ eine beliebige Funktion und $k'$ ist die Länge der Rundenschlüssel $K^1, \ldots, K^N$
- Aus dem letzten Zwischenergebnis $w^N = L^N R^N$ in Runde $N$ wird dann durch Vertauschung von $L^N$ und $R^N$ der Kryptotext $y = R^N L^N$ gebildet

# Aufbau der DES-Chiffrierfunktion

- Der DES ist eine Feistel-Chiffre mit einer Blocklänge von $\ell = 64$ Bit und $N = 16$ Runden
- Die 64 Bit Schlüssel haben eine effektive Länge von 56 Bit, da sie acht Paritätsbits (Bit 8, 16,. . . , 64) enthalten
- Es gibt somit nur $2^{56} \approx 7.2 \cdot 10^{16}$ verschiedene Schlüssel
- Bei Eingabe eines Schlüssels $K$ und eines Klartextes $x$ führt der DES-Algorithmus die folgenden Chiffrierschritte aus:
  - Zuerst wird der Klartext $x$ einer Initialpermutation $IP\colon x_1 x_2 \cdots x_{64} \mapsto x_{58} x_{50} \cdots x_7$ unterzogen (siehe nächste Folie)
  - Danach erfolgen 16 Runden mit einer Feistel-Rundenfunktion $g$ unter Verwendung von sechzehn Rundenschlüsseln $K^1, \ldots, K^{16}$ (die Berechnung der Rundenschlüssel wird später beschrieben)
  - Aus dem am Ende von Runde 16 ausgegebenen Block $w^{16} = L^{16} R^{16}$ wird durch Vertauschen von $L^{16}$ und $R^{16}$ und Anwendung von $IP^{-1}$ der Kryptotextblock $y = \mathrm{DES}(K, x) = IP^{-1}(R^{16} L^{16})$ gebildet

# Graphische Darstellung der DES-Funktion $f$

- Die DES-Funktion $f$:




- Initialpermutation $IP$, Expansion $E$ und Permutation $P$:

| $IP$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

| $E$ | | | | | |
|---|---|---|---|---|---|
| 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 |

# Berechnung der DES-Funktion $f$

- Die Funktion $f : \{0,1\}^{48} \times \{0,1\}^{32} \to \{0,1\}^{32}$ wird wie folgt berechnet:
- Bei Eingabe $(K^r, R^{r-1})$ wird zuerst der 32-Bit Block $R^{r-1}$ mittels der Expansionsabbildung $E$ (s.u.) auf einen 48-Bit Block $E(R^{r-1})$ erweitert
- Auf diesen wird bitweise der Rundenschlüssel $K^r$ addiert
- Als Ergebnis erhalten wir den 48-Bit Block $B = E(R^{r-1}) \oplus K^r$
- Dieser wird in acht 6-Bit Blöcke $B_{(1)}, \ldots, B_{(8)}$ aufgeteilt, die mit den 8 S-Boxen $S_1, \ldots, S_8$ auf 4-Bit Blöcke $C_{(i)} = S_i(B_{(i)})$ verkleinert werden
- Die Konkatenation der von den acht S-Boxen berechneten 4-Bit Blöcke ergibt einen 32-Bit Block $C = C_{(1)} \ldots C_{(8)}$
- Zum Schluss wird auf $C$ noch die Transposition $P$ angewandt (siehe vorige Folie; der Werteverlauf der Transpositionen $IP$, $E$ und $P$ ist dort jeweils zeilenweise dargestellt)

# Tabellarische Darstellung der acht DES S-Boxen

- Die Werte $S_i(B_{(j)})$ lassen sich aus folgenden Tabellen wie folgt ablesen:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$: | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |
| | 0 | F | 7 | 4 | E | 2 | D | 1 | A | 6 | C | B | 9 | 5 | 3 | 8 |
| | 4 | 1 | E | 8 | D | 6 | 2 | B | F | C | 9 | 7 | 3 | A | 5 | 0 |
| | F | C | 8 | 2 | 4 | 9 | 1 | 7 | 5 | B | 3 | E | A | 0 | 6 | D |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_2$: | F | 1 | 8 | E | 6 | B | 3 | 4 | 9 | 7 | 2 | D | C | 0 | 5 | A |
| | 3 | D | 4 | 7 | F | 2 | 8 | E | C | 0 | 1 | A | 6 | 9 | B | 5 |
| | 0 | E | 7 | B | A | 4 | D | 1 | 5 | 8 | C | 6 | 9 | 3 | 2 | F |
| | D | 8 | A | 1 | 3 | F | 4 | 2 | B | 6 | 7 | C | 0 | 5 | E | 9 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_3$: | A | 0 | 9 | E | 6 | 3 | F | 5 | 1 | D | C | 7 | B | 4 | 2 | 8 |
| | D | 7 | 0 | 9 | 3 | 4 | 6 | A | 2 | 8 | 5 | E | C | B | F | 1 |
| | D | 6 | 4 | 9 | 8 | F | 3 | 0 | B | 1 | 2 | C | 5 | A | E | 7 |
| | 1 | A | D | 0 | 6 | 9 | 8 | 7 | 4 | F | E | 3 | B | 5 | 2 | C |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_4$: | 7 | D | E | 3 | 0 | 6 | 9 | A | 1 | 2 | 8 | 5 | B | C | 4 | F |
| | D | 8 | B | 5 | 6 | F | 0 | 3 | 4 | 7 | 2 | C | 1 | A | E | 9 |
| | A | 6 | 9 | 0 | C | B | 7 | D | F | 1 | 3 | E | 5 | 2 | 8 | 4 |
| | 3 | F | 0 | 6 | A | 1 | D | 8 | 9 | 4 | 5 | B | C | 7 | 2 | E |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_5$: | 2 | C | 4 | 1 | 7 | A | B | 6 | 8 | 5 | 3 | F | D | 0 | E | 9 |
| | E | B | 2 | C | 4 | 7 | D | 1 | 5 | 0 | F | A | 3 | 9 | 8 | 6 |
| | 4 | 2 | 1 | B | A | D | 7 | 8 | F | 9 | C | 5 | 6 | 3 | 0 | E |
| | B | 8 | C | 7 | 1 | E | 2 | D | 6 | F | 0 | 9 | A | 4 | 5 | 3 |

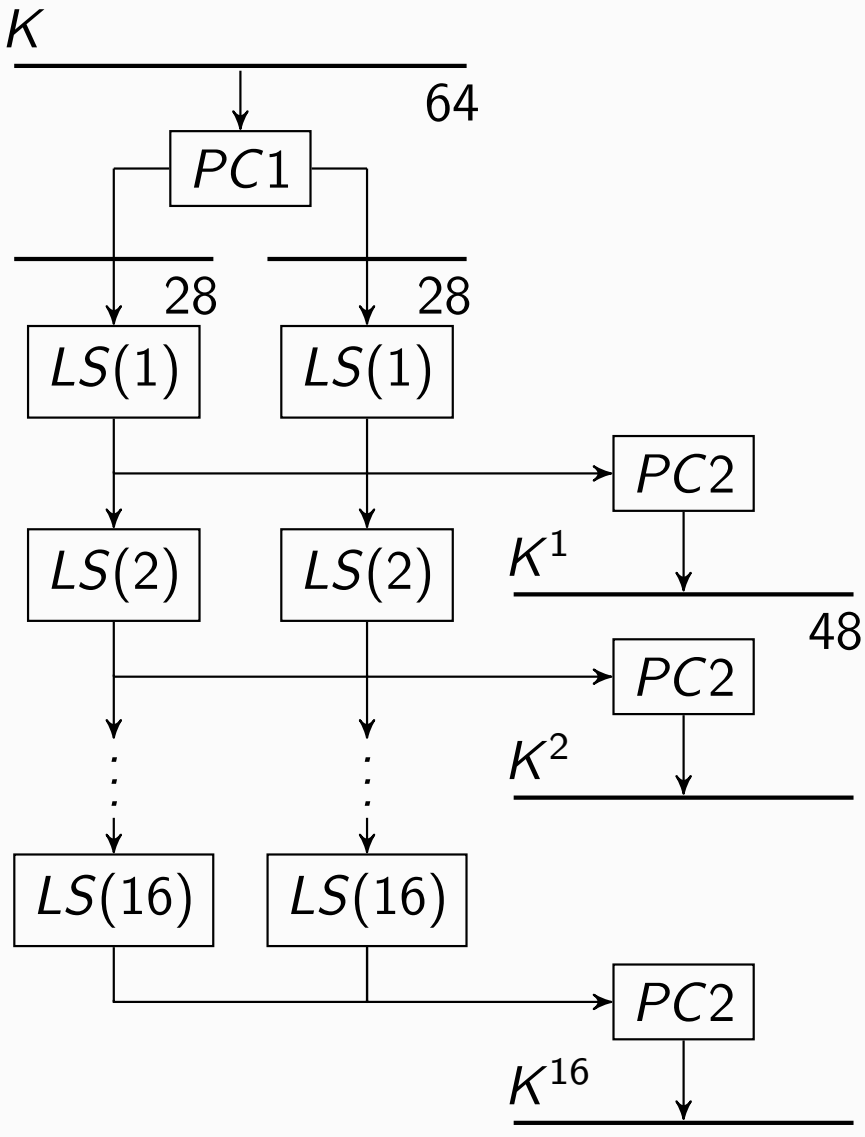
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_6$: | C | 1 | A | F | 9 | 2 | 6 | 8 | 0 | D | 3 | 4 | E | 7 | 5 | B |
| | A | F | 4 | 2 | 7 | C | 9 | 5 | 6 | 1 | D | E | 0 | B | 3 | 8 |
| | 9 | E | F | 5 | 2 | 8 | C | 3 | 7 | 0 | 4 | A | 1 | D | B | 6 |
| | 4 | 3 | 2 | C | 9 | 5 | F | A | B | E | 1 | 7 | 6 | 0 | 8 | D |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_7$: | 4 | B | 2 | E | F | 0 | 8 | D | 3 | C | 9 | 7 | 5 | A | 6 | 1 |
| | D | 0 | B | 7 | 4 | 9 | 1 | A | E | 3 | 5 | C | 2 | F | 8 | 6 |
| | 1 | 4 | B | D | C | 3 | 7 | E | A | F | 6 | 8 | 0 | 5 | 9 | 2 |
| | 6 | B | D | 8 | 1 | 4 | A | 7 | 9 | 5 | 0 | F | E | 2 | 3 | C |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_8$: | D | 2 | 8 | 4 | 6 | F | B | 1 | A | 9 | 3 | E | 5 | 0 | C | 7 |
| | 1 | F | D | 8 | A | 3 | 7 | 4 | C | 5 | 6 | B | 0 | E | 9 | 2 |
| | 7 | B | 4 | 1 | 9 | C | E | 2 | 0 | 6 | A | D | F | 3 | 5 | 8 |
| | 2 | 1 | E | 7 | 4 | A | 8 | D | F | C | 9 | 0 | 3 | 5 | 6 | B |

- Ist $B_{(j)} = b_1 \cdots b_6$, so findet man $S_i(B_{(j)})$ in der Tabelle für $S_i$ in Zeile $b_1 b_6$ und Spalte $b_2 b_3 b_4 b_5$
- Zum Beispiel ist $S_1(011010) = 1001$, da in Zeile $(00)_2 = 0$ und Spalte $(1101)_2 = \mathtt{D}$ die Hexadezimalziffer $9 = (1001)_2$ steht

# Der Key-Schedule Algorithmus des DES




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

| $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $LS(r)$ | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |

| $r$ | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|
| $LS(r)$ | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

# Der Key-Schedule Algorithmus des DES

- Die 16 Rundenschlüssel $K^1, \ldots, K^{16}$ werden wie folgt aus dem externen 64-Bit Schlüssel $K$ berechnet (siehe vorige Folie):
- Zuerst wählt die Funktion $PC\,1$ (permuted choice 1) aus dem Schlüssel $K$ die 56 kryptografisch relevanten Bits aus und permutiert sie
- Das Resultat wird in zwei 28-Bit Blöcke unterteilt, die in 16 Runden $r = 1, \ldots, 16$ jeweils zyklisch um $LS(r) \in \{1, 2\}$ Bit verschoben werden
- Aus den beiden Blöcken nach Runde $r$ bestimmt die Funktion $PC\,2$ (permuted choice 2) jeweils den Rundenschlüssel $K^r$ und entfernt dabei die 8 Bits an den Positionen 9, 18, 22, 25, 35, 38, 43 und 56

# Der Key-Schedule Algorithmus des DES

**Definition**

Ein DES-Schlüssel $K$ heißt schwach, falls alle durch ihn erzeugten Rundenschlüssel gleich sind (d.h. es gilt $\{K^1, \ldots, K^{16}\} = \{K^1\}$)

- Der DES hat folgende vier schwache Schlüssel (hexadezimal):

| $K$ | erzeugter Rundenschlüssel |
|---|---|
| 0101010101010101 | 000000000000 |
| 1F1F1F1F0E0E0E0E | 000000111111 |
| E0E0E0E0F1F1F1F1 | 111111000000 |
| FEFEFEFEFEFEFEFE | 111111111111 |

- Für jeden von ihnen gilt $\mathrm{DES}(K, \mathrm{DES}(K, x)) = x$ (siehe Übungen)
- Zudem existieren noch sechs sogenannte semischwache Schlüsselpaare $(K, K')$ mit der Eigenschaft $\mathrm{DES}(K', \mathrm{DES}(K, x)) = x$ (siehe Übungen)

# Sicherheitsbedenken gegen den DES

- Der DES konnte sich nicht sofort nach seiner Veröffentlichung im Jahr 1975 durchsetzen
- Er wurde von manchen Behörden und Banken in den USA zunächst nicht verwendet, da folgende Sicherheitsbedenken gegen ihn bestanden:
  - Die 56-Bit Schlüssellänge bietet eine zu geringe Sicherheit gegen einen Brute-Force Angriff bei bekanntem oder wählbarem Klartext
  - Die Entwurfskriterien für die einzelnen Komponenten, insbesondere die $S$-Boxen, sind nicht veröffentlicht worden; daher wurde der Verdacht geäußert, dass der DES Falltüren besitzt, um einen Angriff durch die National Security Agency (NSA) zu ermöglichen
  - Kryptoanalytische Untersuchungen, die von IBM und der NSA durchgeführt wurden, blieben unter Verschluss
  - Als jedoch Biham und Shamir Anfang der 90er Jahre das Konzept der differentiellen Kryptoanalyse veröffentlichten, gaben die Entwickler bekannt, dass sie diesen Angriff beim Entwurf von DES bereits kannten und sie die S-Boxen dahingehend optimiert haben

# Sicherheitseigenschaften des DES

- Im Fall von DES ist die lineare Kryptoanalyse effizienter als die differentielle Kryptoanalyse
- Da hierzu jedoch circa $2^{43}$ Klartext-Kryptotext-Doppelpaare notwendig sind, stellen diese Angriffe keine realistische Bedrohung dar (allein die Generierung der Doppelpaare dauerte bei einem von Matsui, dem Erfinder der linearen Kryptoanalyse, durchgeführten Angriff 40 Tage)
- Dagegen gelang bereits im Juli 1998 mit einer von der Electronic Frontier Foundation (EFF) für 250 000 Dollar gebauten Maschine namens "DES Cracker" eine vollständige Schlüsselsuche in 56 Stunden (dies bedeutete den Gewinn der von RSA Laboratory ausgeschriebenen "DES Challenge II-2")

# Sicherheitseigenschaften des DES

- Im Jahr 1999 gewann `Distributed.Net`, eine weltweite Vereinigung von Computerfans, den mit 10 000 Dollar dotierten "DES Challenge III"
- Ihnen gelang es, durch den kombinierten Einsatz des Supercomputers "Deep Crack" und 100 000 PCs, die per Internet kommunizierten, nach 22:15h den Schlüssel zu dem bekannten Klartext „See you in Rome (second AES Conference, March 22-23, 1999)" zu finden
- Es gibt sogar kommerzielle Angebote im Internet (z.B. `crack.sh`), die bei bekanntem Klartext innerhalb von 26 Stunden eine vollständige Schlüsselsuche auf spezieller Hardware ausführen und alle passenden DES-Schlüssel finden

# Endliche Körper

- Als Vorbereitung zum AES-Algorithmus gehen wir kurz auf die Arithmetik in endlichen Körpern ein
- Diese spielt beim AES eine wichtige Rolle
- Wie wir bereits wissen, bildet $\mathbb{Z}_p$ für primes $p$ einen endlichen Körper der Größe $p$
- Dieser Körper lässt sich für jede Zahl $n \geq 1$ auf die Größe $p^n$ erweitern
- Da bis auf Isomorphie nur ein Körper der Größe $p^n$ existiert, wird er einfach mit $\mathbb{F}(p^n)$ oder $\mathbb{F}_{p^n}$ bezeichnet
- Um den Körper $\mathbb{F}_{p^n}$ zu konstruieren, betrachten wir zunächst den Polynomring $\mathbb{Z}_p[x]$ über $\mathbb{Z}_p$

# Polynomringe

**Definition.** Sei $R$ ein Ring.

- Der Polynomring $R[x]$ enthält für alle $n \geq 0$ alle Polynome $q(x)$ in der Variablen $x$ mit Koeffizienten in $R$, d.h. $q(x)$ hat die Form

  $$q(x) = a_n x^n + \cdots a_1 x + a_0 \quad \text{mit} \quad a_0, \ldots, a_n \in R$$

  Man sagt, $R[x]$ entsteht aus $R$ durch Adjunktion der Variablen $x$
- Der Grad von $q(x)$ (bezeichnet mit $\deg(q)$) ist im Fall $a_n \neq 0$ gleich $n$ und im Fall $n = a_n = 0$ gleich $-1$

Man überprüft leicht, dass $R[x]$ mit der üblichen Polynomaddition und Polynommultiplikation tatsächlich einen Ring bildet

**Definition (Fortsetzung).**

- Ein Polynom $q(x)$ teilt ein Polynom $s(x)$ (kurz: $q(x)|s(x)$), falls ein Polynom $d(x) \in R[x]$ existiert mit $s(x) = d(x)q(x)$
- Teilt $q(x)$ die Differenz $a(x) - b(x)$ zweier Polynome, so schreiben wir

  $$a(x) \equiv_{q(x)} b(x)$$

  und sagen, $a(x)$ ist kongruent zu $b(x)$ modulo $q(x)$
- Weiterhin bezeichne

  $$s(x) \bmod q(x)$$

  das bei der Polynomdivision von $s(x)$ durch $q(x)$ auftretende Restpolynom, also dasjenige Polynom $r(x)$ vom Grad $\deg(r) < \deg(q)$, für das ein Polynom $d(x) \in R[x]$ existiert mit $s(x) = d(x)q(x) + r(x)$

# Faktorringe von Polynomringen

- Ähnlich wie beim Übergang von $\mathbb{Z}$ zum Restklassenring $\mathbb{Z}_m$ können wir nun jedem Polynom $s(x) \in \mathbb{Z}_m[x]$ mittels

  $$s(x) \mapsto s(x) \bmod m(x)$$

  eindeutig ein Polynom vom Grad höchstens $n-1$ zuordnen, wobei $m(x)$ ein fest gewähltes Polynom und $n$ der Grad von $m(x)$ ist
- Auf diese Weise erhalten wir den Polynomring $\mathbb{Z}_m[x]/m(x)$ aller Polynome vom Grad höchstens $n-1$
- Dieser endliche Ring heißt Faktorring von $\mathbb{Z}_m[x]$ modulo $m(x)$
- Die Addition und Multiplikation sind hierbei wie in $\mathbb{Z}_m[x]$, gefolgt von einer Reduktion modulo $m(x)$, definiert
- In den Übungen werden wir sehen, dass $\mathbb{Z}_m[x]/m(x)$ genau dann ein Körper ist, wenn $m$ prim ist und $m(x)$ nur triviale Teiler besitzt

# Irreduzible Polynome

**Definition. Sei $p$ prim.**

Ein Polynom $m(x) \in \mathbb{Z}_p[x]$ vom Grad $n \geq 2$ heißt irreduzibel (über $\mathbb{Z}_p$), falls keine Polynome $r(x), s(x) \in \mathbb{Z}_p[x]$ vom Grad $\deg(r), \deg(s) \geq 1$ existieren mit

$$m(x) = r(x)s(x)$$

**Satz**

Der Faktorring $\mathbb{Z}_m[x]/m(x)$ ist genau dann ein Körper, wenn $m = p$ prim und $m(x)$ in $\mathbb{Z}_p[x]$ irreduzibel ist.

**Beweis.**

Siehe Übungen. □

# Arithmetik in endlichen Körpern

- Man kann zeigen, dass für primes $p$ und jede Zahl $n \geq 2$ ein irreduzibles Polynom $m(x) = x^n + \sum_{i=0}^{n-1} m_i x^i$ vom Grad $n$ in $\mathbb{Z}_p[x]$ existiert
- Daher lässt sich für jede Primzahlpotenz $p^n$ ein Körper $\mathbb{Z}_p[x]/m(x)$ der Größe $p^n$ konstruieren
- Tatsächlich gibt es bis auf Isomorphie nur einen solchen Körper, den wir mit $\mathbb{F}_{p^n}$ bezeichnen
- Wir können Polynome $a(x) = \sum_{i=0}^{n-1} a_i x^i$ auch als Koeffizientenvektoren $\vec{a} = (a_{n-1}, \ldots, a_0)$ darstellen
- Die Addition von $a(x)$ und $b(x)$ in $\mathbb{F}_{p^n}$ entspricht dann der üblichen Vektoraddition (komponentenweise Addition modulo $p$) von $\vec{a}$ und $\vec{b}$
- Die Vektordarstellung $\vec{c}$ von $c(x) = a(x) + b(x)$ ist also

$$(c_{n-1}, \ldots, c_0) = (a_{n-1} + b_{n-1}, \ldots, a_0 + b_0)$$

- Im Fall $p = 2$ ist dies die bitweise Addition modulo 2 (xor)

# Arithmetik in endlichen Körpern

- Die Multiplikation in $\mathbb{Z}_p[x]/m(x)$ lässt sich wegen

$$a(x)b(x) = \sum_{i=0}^{n-1} a_i x^i b(x)$$

auf die Addition und (iterierte) Multiplikation mit dem Polynom $p(x) = x$ zurückführen

- Da $m(x)$ die Form $m(x) = \sum_{i=0}^{n} m_i x^i$ mit $m_n = 1$ hat, ist

$$\begin{aligned} xb(x) &\equiv_{m(x)} xb(x) - b_{n-1}m(x) \\ &= \sum_{i=1}^{n} b_{i-1}x^i - b_{n-1}\sum_{i=0}^{n} m_i x^i \\ &= \sum_{i=0}^{n-1}(b_{i-1} - b_{n-1}m_i)x^i \end{aligned}$$

wobei wir $b_{-1} = 0$ setzen

- Die Multiplikation von $b(x)$ mit $x$ entspricht somit einem Linksshift von $\vec{b}$ um eine Stelle, dem sich im Fall $b_{n-1} \neq 0$ noch die Subtraktion des Vektors $(b_{n-1}m_{n-1}, \ldots, b_{n-1}m_0)$ anschließt

# Arithmetik in endlichen Körpern

- Im Fall $p = 2$ erhalten wir also

$$xb(x) = \begin{cases} \sum_{i=1}^{n-1} b_{i-1} x^i, & b_{n-1} = 0 \\ \sum_{i=0}^{n-1} (b_{i-1} \oplus m_i) x^i, & b_{n-1} = 1 \end{cases}$$

und die Vektordarstellung $\vec{c}$ von $c(x) = xb(x)$ ist

$$(c_{n-1}, \ldots, c_0) = \begin{cases} (b_{n-2}, \ldots, b_0, 0), & b_{n-1} = 0 \\ (b_{n-2}, \ldots, b_0, 0) \oplus (m_{n-1}, \ldots, m_0), & b_{n-1} = 1 \end{cases}$$

## Beispiel

- Wir möchten einen endlichen Körper der Größe $p^n = 2^3$ konstruieren
- Dazu benötigen wir ein irreduzibles Polynom $m(x) \in \mathbb{Z}_2[x]$ vom Grad 3
- Setzen wir $m(x) = x^3 + a_2x^2 + a_1x + a_0$, so sehen wir, dass $m(x)$ im Fall $a_0 = 0$ den nichttrivialen Teiler $p(x) = x$ hat
- Daher genügt es, die 4 Kandidaten

$$\begin{aligned} m_1(x) &= x^3 + 1 \\ m_2(x) &= x^3 + x + 1 \\ m_3(x) &= x^3 + x^2 + 1 \\ m_4(x) &= x^3 + x^2 + x + 1 \end{aligned}$$

zu betrachten

# Arithmetik in endlichen Körpern

## Beispiel

- Wegen

$$x^3+1=(x+1)(x^2+x+1) \quad \text{und} \quad x^3+x^2+x+1=(x+1)(x^2+1)$$

sowie

$$x^3+x+1=(x+1)(x^2+x)+1 \quad \text{und} \quad x^3+x^2+1=(x+1)x^2+1$$

gibt es in $\mathbb{Z}_2[x]$ genau zwei irreduzible Polynome vom Grad 3, nämlich $x^3+x+1$ und $x^3+x^2+1$ (da sie weder $x$ noch $x+1$ als Teiler haben)

- Wählen wir $m(x)=x^3+x+1$, so erhalten wir in $\mathbb{Z}_2[x]/m(x)$ bspw. die Summe

$$(x^2+1)+(x+1)=x^2+x+1+1=x^2+x$$

da in $\mathbb{Z}_2$ die Gleichung $1+1=0$ gilt, sowie das Produkt

$$(x^2+1)(x+1)=x^3+x^2+x+1=x^2+(x^3+x+1)=x^2$$

da in $\mathbb{Z}_2[x]$ die Kongruenz $x^3+x+1=m(x)\equiv_{m(x)} 0$ gilt ◁

# Berechnung von multiplikativen Inversen in $\mathbb{Z}_m[x]/m(x)$



Das multiplikative Inverse eines Polynoms $a(x) \in \mathbb{Z}_m[x]/m(x)$ lässt sich mit dem erweiterten euklidischen Algorithmus berechnen (falls es existiert)

## Beispiel

- Sei $p = 2$ und seien $m(x) = x^8 + x^4 + x^3 + x + 1$ und $a(x) = x^6 + x^4 + x + 1$ zwei Polynome in $\mathbb{Z}_2[x]$
- Dann können wir den (in Bezug auf den Grad) größten gemeinsamen Teiler von $m(x)$ und $a(x)$ mit dem euklidischen Algorithmus berechnen:

| $i$ | $r_{i-1}(x) =$ | $d_i(x) \cdot r_i(x)$ | $+ r_{i+1}(x)$ |
|---|---|---|---|
| 1 | $x^8 + x^4 + x^3 + x + 1 =$ | $(x^2 + 1) \cdot (x^6 + x^4 + x + 1)$ | $+ x^2$ |
| 2 | $x^6 + x^4 + x + 1 =$ | $(x^4 + x^2) \cdot x^2$ | $+ x + 1$ |
| 3 | $x^2 =$ | $(x + 1) \cdot (x + 1)$ | $+ 1$ |
| 4 | $x + 1 =$ | $(x + 1) \cdot \mathbf{1}$ | $+ 0$ |

- Es gilt also $\mathrm{ggT}(a(x), m(x)) = r_4(x) = 1$

# Berechnung von multiplikativen Inversen in $\mathbb{F}_{p^n}$

## Beispiel (Fortsetzung)

- Der erweiterte euklidische Algorithmus berechnet neben den Polynomen $r_i(x)$ und $d_i(x)$ für $i = 0, \ldots, s$ zusätzlich Polynome $p_i(x)$ und $q_i(x)$ mit

$$p_0(x) = 1,\ p_1(x) = 0 \text{ und } p_i(x) = p_{i-2}(x) - d_{i-1}(x)p_{i-1}(x) \text{ und}$$
$$q_0(x) = 0,\ q_1(x) = 1 \text{ und } q_i(x) = q_{i-2}(x) - d_{i-1}(x)q_{i-1}(x)$$

  für $i = 2, \ldots, s$

- Dann lässt sich wieder induktiv über $i = 0, \ldots, s$ die Gleichung

$$p_i(x)m(x) + q_i(x)a(x) = r_i(x)$$

  zeigen

- Im Fall $r_i(x) = 1$ ist also $q_i(x)$ das multiplikative Inverse von $a(x)$ modulo $m(x)$

# Berechnung von multiplikativen Inversen in $\mathbb{F}_{p^n}$

## Beispiel (Schluss)

- Der erweiterte euklidische Algorithmus berechnet nun die Polynome $p_i(x)$ und $q_i(x)$ nach den Formeln

  $$p_i(x) = p_{i-2}(x) - d_{i-1}(x)p_{i-1}(x) \text{ und } q_i(x) = q_{i-2}(x) - d_{i-1}(x)q_{i-1}(x)$$

  wie folgt:

| $i$ | $d_i(x)$ | $p_i(x)\cdot m(x) +$ | $q_i(x)\cdot a(x) = r_i(x)$ |
|---|---|---|---|
| 0 | | $1\cdot m(x) +$ | $0\cdot a(x) = m(x)$ |
| 1 | $x^2+1$ | $0\cdot m(x) +$ | $1\cdot a(x) = a(x)$ |
| 2 | $x^4+x^2$ | $1\cdot m(x) +$ | $(x^2+1)\cdot a(x) = x^2$ |
| 3 | $x+1$ | $(x^4+x^2)\cdot m(x) +$ | $(x^6+x^2+1)\cdot a(x) = x+1$ |
| 4 | | $(x^5+x^4+x^3+x^2+1)\cdot m(x) +$ | $(x^7+x^6+x^3+x)\cdot a(x) = 1$ |

- Aus der letzten Zeile können wir nun das multiplikative Inverse $a^{-1}(x) = q_4(x) = x^7 + x^6 + x^3 + x$ von $a(x)$ modulo $m(x)$ ablesen ◁

# Der Advanced Encryption Standard (AES)

- Im September 1997 veröffentlichte das NIST eine Ausschreibung für den AES mit einer Blocklänge von 128 Bit und variablen Schlüssellängen von 128, 192 und 256 Bit; Einreichungsschluss war der 15. Juni 1998
- Es gab 21 Einreichungen, aber nur 15 erfüllten alle Kriterien
- Diese wurden auf der 1. AES-Konferenz im Aug. 1998 ausgewählt und stammten aus Australien, Belgien, Costa Rica, Deutschland, Frankreich, Großbritannien, Israel, Japan, Korea, Norwegen sowie den USA
- Aus den 15 Kandidaten wurden auf der 2. AES-Konferenz im Aug. 1999 fünf Finalisten ausgewählt:
  - MARS (IBM, USA)
  - RC6 (RSA Security, USA)
  - Rijndael (Joan Daemen und Vincent Rijmen, Belgien)
  - Serpent (Ross Anderson, Großbritannien; Eli Biham, Israel; Lars Knudsen, Norwegen)
  - Twofish (Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson, USA)

# Der Advanced Encryption Standard (AES)

- Die wichtigsten Entscheidungskriterien waren
  - Sicherheit,
  - Effizienz bei Software-, Hardware- und Smartcard-Implementierungen
  - sowie Algorithmen- und Implementations-Charakeristika (u.a. Flexibilität und Einfachheit des Designs)
- Vom NIST erhielten die Finalisten folgende Bewertungspunkte:

| | Rijndael | Serpent | Twofish | MARS | RC6 |
|---|---|---|---|---|---|
| Sicherheit | 2 | 3 | 3 | 3 | 2 |
| einfache Implementierung | 3 | 3 | 2 | 1 | 1 |
| Software-Performanz | 3 | 1 | 1 | 2 | 2 |
| Smartcard-Performanz | 3 | 3 | 2 | 1 | 1 |
| Hardware-Performanz | 3 | 3 | 2 | 1 | 2 |
| Design-Merkmale | 2 | 1 | 3 | 2 | 1 |
| Summe | 16 | 14 | 13 | 10 | 9 |

# Unterschiede zwischen Rijndael und AES

- Die Blocklänge $\ell$ und die Schlüssellänge $k$ sind beim Rijndael in 32 Bit Schritten jeweils zwischen 128 und 256 Bit wählbar
- Nebenstehende Tabelle zeigt die Rundenzahl $N$ in Abhhängigkeit von $\ell$ und $k$
- Beim AES-Standard wurde die Blocklänge auf 128 Bit fixiert und die Schlüssellänge auf die Werte 128, 192 oder 256 Bit beschränkt

| | $k$ | | | | |
|---|---|---|---|---|---|
| $\ell$ | 128 | 160 | 192 | 224 | 256 |
| 128 | **10** | 11 | **12** | 13 | **14** |
| 160 | 11 | 11 | 12 | 13 | 14 |
| 192 | 12 | 12 | 12 | 13 | 14 |
| 224 | 13 | 13 | 13 | 13 | 14 |
| 256 | 14 | 14 | 14 | 14 | 14 |

- Wir beschränken uns im Folgenden auf die Beschreibung des 10-Runden AES mit $\ell = 128$ Bit Blocklänge und $k = 128$ Bit Schlüssellänge

# Die AES S-Box SubByte

- Die Elemente $a(x) = \sum_{i=0}^{7} a_i x^i$ des Körpers $\mathbb{F}_{2^8} = \mathbb{Z}_2[x]/m(x)$ mit $m(x) = x^8 + x^4 + x^3 + x + 1$ können jeweils durch ein Byte $a_7 \ldots a_0$ dargestellt werden
- Zur expliziten Konvertierung der beiden Darstellungen verwenden wir folgende Funktionen:
  - Die Funktion BinaryToField: $\{0,1\}^8 \to \mathbb{F}_{2^8}$ überführt die Byte-Darstellung $a_7 \ldots a_0$ in das zugehörige Polynom $a(x) = \sum_{i=0}^{7} a_i x^i$
  - Die Funktion FieldToBinary: $\mathbb{F}_{2^8} \to \{0,1\}^8$ ist die Umkehrfunktion von BinaryToField
- Sowohl der Key-Schedule Algoritmus als auch die AES-Chiffrierfunktion verwenden eine S-Box SubByte
- Diese benutzt als nicht-linearen Bestandteil die Funktion

  $$\text{FieldInv} : \mathbb{F}_{2^8}^* \to \mathbb{F}_{2^8}^*,$$

  die das multiplikative Inverse aller Einheiten des Körpers $\mathbb{F}_{2^8}$ berechnet

# Die AES S-Box SubByte

Konkret wird SubByte wie folgt berechnet

**S-Box** SubByte($a_7 \cdots a_0$)

1 **input** $a_7 \cdots a_0$
2 $z := \text{BinaryToField}(a_7 \cdots a_0)$
3 **if** $z \neq 0$ **then** $z := \text{FieldInv}(z)$
4 $a_7 \cdots a_0 := \text{FieldToBinary}(z)$
5 $c_7 \cdots c_0 := 01100011$
6 **for** $i := 0$ **to** $7$ **do**
7 $b_i := a_i \oplus a_{i+4} \oplus a_{i+5} \oplus a_{i+6} \oplus a_{i+7} \oplus c_i$
8 **output** $b_7 \cdots b_0$

- Die Indexrechnung in Zeile 7 erfolgt modulo 8
- Die Zeilen 5-8 realisieren eine affine Hill-Chiffrierfunktion

# Die AES S-Box SubByte

## Beispiel

- Bei Eingabe 01010011 liefert die Funktion BinaryToField das zugehörige Polynom $z = \text{BinaryToField}(01010011) = x^6 + x^4 + x + 1$
- FieldInv($z$) berechnet das multiplikative Inverse $z^{-1} = x^7 + x^6 + x^3 + x$
- FieldToBinary($x^7 + x^6 + x^3 + x$) liefert den Vektor $a_7 \dots a_0 = 11001010$
- Es folgt die Berechnung der Ausgabe $b_7 \dots b_0$ mittels

$$b_7 \dots b_0 \quad = 11001010 \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \oplus 01100011 = 11101101$$

- Somit ist SubByte(01010011) = 11101101
- Oder in Hexadezimaldarstellung: SubByte(53) = ED ◁

# Die AES S-Box SubByte

- Wir können die AES S-Box SubByte in Form einer $(16 \times 16)$-Matrix angeben, die in Zeile X und Spalte Y den Eintrag SubByte(XY) enthält

| | Y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 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 | 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 |
| | | | | | | | | ⋮ | | | | | | | | |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

# Der Key-Schedule Algorithmus des AES

- Beim 10-Runden AES mit Block- und Schlüssellänge $\ell = k = 128$ werden 11 Rundenschlüssel $K^0, \ldots, K^{10}$ der Länge 128 benutzt
- Jeder Rundenschlüssel $K^i$ besteht also aus 16 Bytes bzw. 4 Worten mit jeweils 4 Bytes
- Bei der Berechnung der Rundenschlüssel werden die folgenden Wort-Konstanten benutzt:

$$RCon[i] = \text{FieldToBinary}(x^{i-1})0^{24} \in \{0,1\}^{32} \text{ für } i = 1, \ldots, 10$$

- In Hexadezimal-Darstellung ergeben sich die folgenden Werte:

| $i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $RCon[i]$ | 01000000 | 02000000 | 04000000 | 08000000 | 10000000 |

| $i$ | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|
| $RCon[i]$ | 20000000 | 40000000 | 80000000 | 1B000000 | 36000000 |

# Der Key-Schedule Algorithmus des AES

- Reihen wir die 11 Rundenschlüssel $K^0, \ldots, K^{10}$ aneinander, so entsteht ein Array von 44 Worten $K^0 \ldots K^{10} = w[0] \ldots w[43]$
- Diese werden gemäß folgendem Algorithmus aus dem 128-Bit Schlüssel $K$ berechnet:

**Algorithmus** KeyExpansion($K$)

```
1  input K ∈ {0,1}^128
2    w[0]...w[3] := K
3    for i := 4 to 43 do
4      temp := w[i − 1]
5      if i ≡_4 0 then
6        temp := SubWord(RotWord(temp)) ⊕ RCon[i/4]
7      w[i] := w[i − 4] ⊕ temp
8  output w[0]...w[43]
```

# Der Key-Schedule Algorithmus des AES

Die hierbei benutzten Funktionen sind wie folgt definiert:

- Die Funktion RotWord ist eine 32-Bit Transposition, die die vier Eingabebytes zyklisch um ein Byte nach links verschiebt:

$$\text{RotWord}(B_0 B_1 B_2 B_3) = B_1 B_2 B_3 B_0$$

- Die Funktion SubWord ist eine 32-Bit Substitution, die durch vier parallel geschaltete S-Boxen SubByte realisiert wird

# Der AES Chiffrieralgorithmus

- Die AES Chiffrierfunktion chiffriert einen 128 Bit Klartextblock wie folgt

**Algorithmus** AES$(K, x)$

```
1  K^0 ... K^10 := KeyExpansion(K)
2  AddRoundKey(K^0)
3  for r := 1 to 9 do
4     SubBytes
5     ShiftRows
6     MixColumns
7     AddRoundKey(K^r)
8  SubBytes
9  ShiftRows
10 AddRoundKey(K^10)
```

- Nach Generierung der elf 128-Bit Rundenschlüssel $K^0, \ldots, K^{10}$ wird der Klartextblock $x$ einer Addition mit $K^0$ unterworfen
- Diese Operation wird mit AddRoundKey$(K^0)$ bezeichnet

# Der AES Chiffrieralgorithmus

- Danach werden neun Runden ausgeführt, wobei in Runde $r$ der Reihe nach folgende Operationen ausgeführt werden:
  - SubBytes: eine nichtlineare 128-Bit Substitution, die durch 16 parallel geschaltete S-Boxen SubByte realisiert wird
  - ShiftRows: eine 128-Bit Transposition (siehe unten)
  - MixColumns: eine lineare 128-Bit Substitution (siehe unten) und
  - AddRoundKey($K^r$): eine Addition mit dem Rundenschlüssel $K^r$
- Es folgt Runde 10 mit den Operationen
  - SubBytes
  - ShiftRows
  - AddRoundKey($K^{10}$)
- Abgesehen von der zusätzlichen linearen Substitution MixColumns in den ersten neun Runden und von der Transposition ShiftRows in Runde 10 entspricht der Aufbau des AES also exakt dem Aufbau eines SPNs

# Die Transposition ShiftRows

- Um die beiden Operationen ShiftRows und MixColumns zu beschreiben, stellen wir den Klartext $x = x_0 \cdots x_{15}$, $x_i \in \{0,1\}^8$, und alle daraus berechneten Zwischenergebnisse in Form einer Matrix $M \in \mathbb{F}_{2^8}^{(4\times 4)}$ dar
- Diese wird wie folgt initialisiert:

$$\begin{array}{cccc} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{array}$$

- Die Operation ShiftRows ist eine 128-Bit Transposition, die wie folgt definiert ist:

$$\text{ShiftRows}: \begin{array}{cccc} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{array} \mapsto \begin{array}{cccc} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{array}$$

# Die S-Box MixColumn

- Die lineare Substitution MixColumn ist wie folgt definiert:

**S-Box** MixColumn$(c_3, c_2, c_1, c_0)$

1 **input** $(c_3, c_2, c_1, c_0)$
2 **for** $i := 0$ **to** 3 **do** $t_i :=$ BinaryToField$(c_i)$
3 $u_0 :=$ FieldMult$(x, t_0) +$ FieldMult$(x+1, t_1) + t_2 + t_3$
4 $u_1 :=$ FieldMult$(x, t_1) +$ FieldMult$(x+1, t_2) + t_3 + t_0$
5 $u_2 :=$ FieldMult$(x, t_2) +$ FieldMult$(x+1, t_3) + t_0 + t_1$
6 $u_3 :=$ FieldMult$(x, t_3) +$ FieldMult$(x+1, t_0) + t_1 + t_2$
7 **for** $i := 0$ **to** 3 **do** $c'_i :=$ FieldToBinary$(u_i)$
8 **output** $(c'_3, c'_2, c'_1, c'_0)$

- Die Operation MixColumns führt die 32-Bit S-Box MixColumn parallel auf den vier Spalten $(s_{3,j}, s_{2,j}, s_{1,j}, s_{0,j})$, $j = 0, \ldots, 3$, des aktuellen Zwischenergebnisses aus
- Bei ihrer Berechnung wird die Funktion FieldMult: $\mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ benutzt, die ihre beiden Argumente im Körper $\mathbb{F}_{2^8}$ multipliziert

# Die S-Box MixColumn

- MixColumn führt also eine lineare Transformation in dem Vektorraum $(\mathbb{F}_{2^8})^4$ aus, die sich auch wie folgt beschreiben lässt:

$$\text{MixColumn} : c_3 \dots c_0 \mapsto c_3 \dots c_0 \underbrace{\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}}_{Z} = c'_3 \dots c'_0$$

- Hierbei repräsentieren wir die Elemente des Körpers $\mathbb{F}_{2^8}$ als Koeffizientenvektoren in Hexadezimaldarstellung, d.h. 03 steht für $x + 1$ usw.
- Die S-Box MixColumn realisiert somit eine lineare 32-Bit Substitution $c \mapsto cZ$ auf dem Vektorraum $(\mathbb{F}_{2^8})^4$
- Zudem ist jede Zeile von $Z$ relativ zur darüber liegenden Zeile um eine Position zyklisch nach rechts verschoben

# Die S-Box MixColumn

- Aufgrund der speziellen Form von $Z$ lässt sich MixColumn als multiplikative Chifrierfunktion

  $$c(y) \mapsto z(y)c(y)$$

  im Faktorring $R = \mathbb{F}_{2^8}[y]/(y^4+1)$ beschreiben
- Stellen wir nämlich die Polynome $c(y) = \sum_{i=0}^{3} c_i y^i$ in $R$ durch ihre Koeffizientenvektoren $\vec{c} = (c_3, c_2, c_1, c_0) \in (\mathbb{F}_{2^8})^4$ dar, so berechnet MixColumn$(\vec{c})$ den Koeffizientenvektor $\vec{c}' = (c_3', c_2', c_1', c_0')$ des Produkts

  $$z(y)c(y) = c_3'y^3 + c_2'y^2 + c_1'y + c_0'$$

  der beiden Polynome $z(y) = 03y^3 + 01y^2 + 01y + 02$ und $c(y)$ in $R$
- Der Ring $\mathbb{F}_{2^8}[y]/(y^4+1)$ ist zwar kein Körper, da das Polynom $y^4+1$ wegen $y^4+1 = (y+1)^4$ in $\mathbb{F}_{2^8}[y]$ nicht irreduzibel ist
- Da aber die Matrix $Z$ invertierbar ist, kann die inverse Abbildung MixColumn$^{-1}$ von MixColumn mittels $c \mapsto cZ^{-1}$ berechnet werden
- Somit hat auch das Polynom $z(y)$ im Ring $\mathbb{F}_{2^8}[y]/(y^4+1)$ ein multiplikatives Inverses $z^{-1}(y)$

# Kryptoanalytische Betrachtungen

- Bis heute konnten keine Schwachstellen gefunden werden, d.h. alle bekannten Angriffe gegen den AES bringen keinen signifikanten Vorteil gegenüber einer vollständigen Schlüsselsuche
- Die Tatsache, dass für die S-Box SubByte die Inversenbildung in einem endlichen Körper benutzt wird, führt dazu, dass die Tabellen für die Güte der linearen Approximationen und für die Weitergabequotienten der Differenzenpaare einen hohen Grad an Uniformität aufweisen
- Dadurch wird die S-Box resistent gegen lineare und differentielle Analysen
- Zudem verhindert die lineare Substitution MixColumns lineare und differentielle Angriffe mit nur wenigen aktiven S-Boxen
- Diese Technik wird von den AES-Entwicklern als wide trail strategy bezeichnet

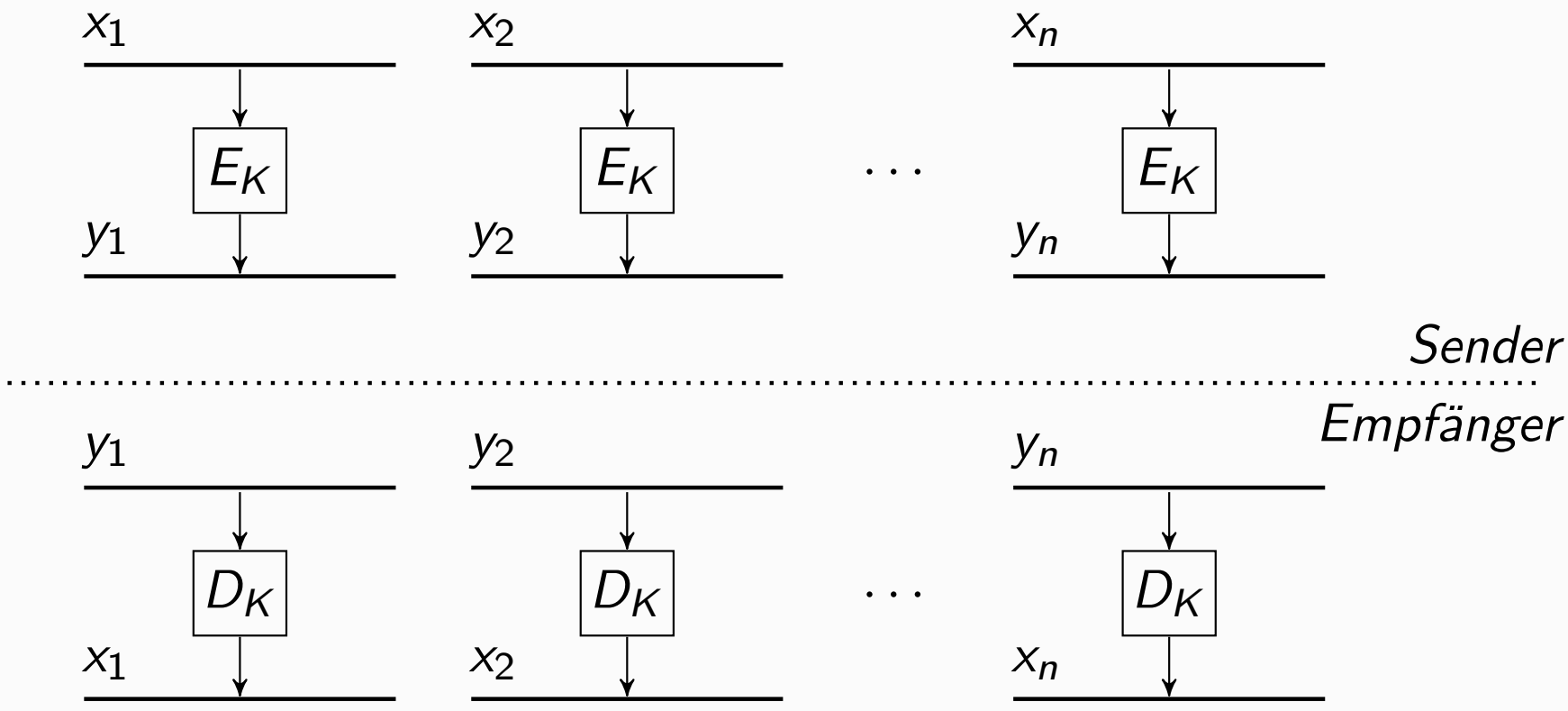
# Betriebsarten von Blockchiffren

- Für den DES wurden vier verschiedene Betriebsarten vorgeschlagen, in denen grundsätzlich jede Blockchiffre $E$ mit beliebiger Blocklänge $\ell$ betrieben werden kann
- Bei den ersten beiden Betriebsarten (ECB und CBC) werden Kryptotextblöcke der Länge $\ell$ übertragen
- Mit einer Blockchiffre kann aber auch ein Stromsystem realisiert werden, mit dem sich Kryptotextblöcke einer beliebigen Länge $t$, $1 \leq t \leq \ell$, übertragen lassen (siehe OFB-, CFB- und CTR-Modus)
- Der ECB-Modus ist die naheliegendste Betriebsart, wird aber in der Praxis so gut wie nie benutzt, da sie eine Reihe von Angriffsmöglichkeiten bietet
- Zum Beispiel hatten wir gesehen, dass ein effizienter IND-CPA Gegner bei einer ECB-Übertragung leicht einen Vorteil von 1 erzielen kann

# ECB-Modus (electronic codebook; elektron. Codebuch)

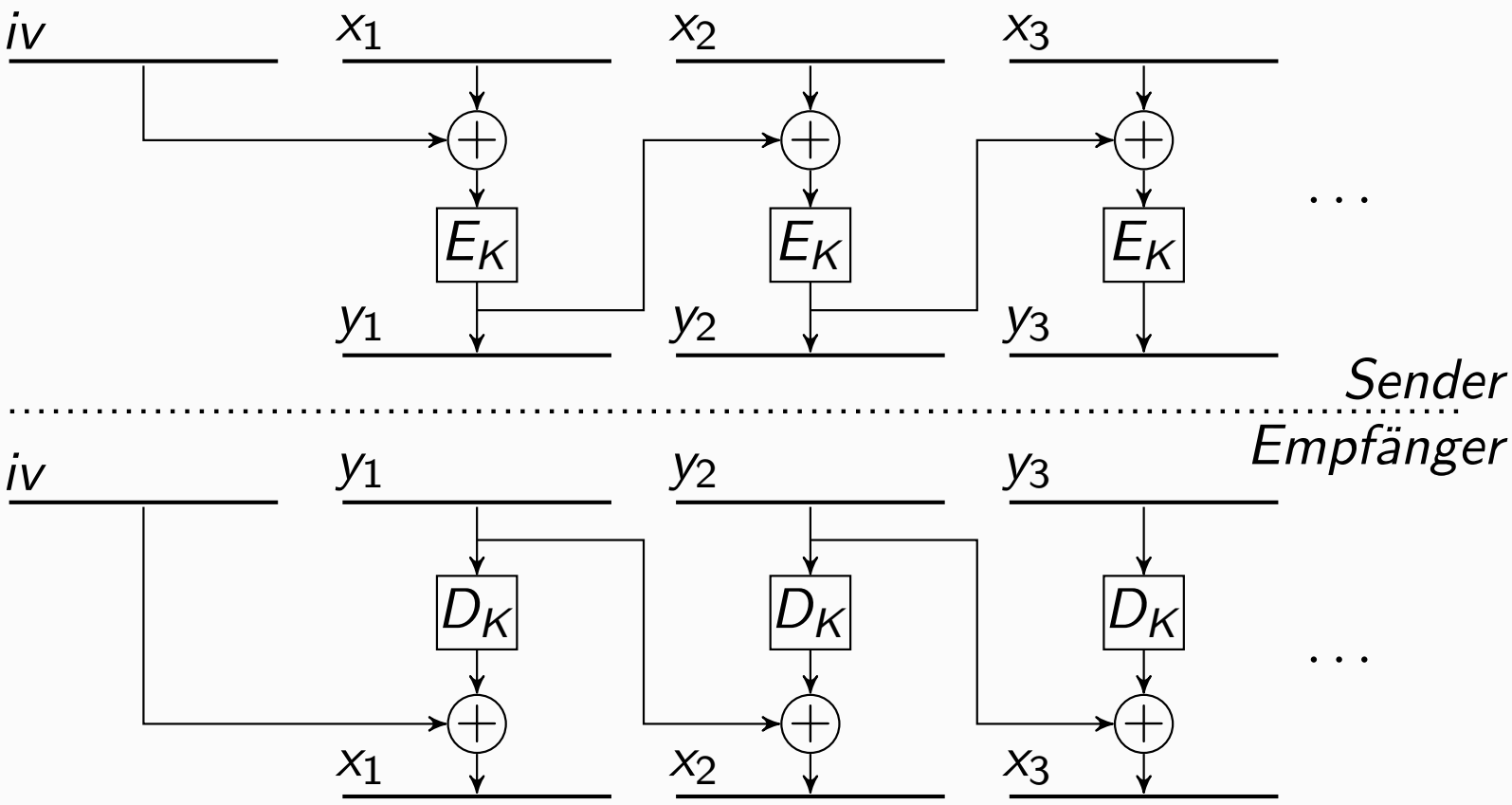


- Die Binärnachricht $x$ wird in Klartextblöcke $x_i$ der Länge $\ell$ zerlegt
- Der letzte Block $x_n$ wird ggf. nach einer vereinbarten Regel aufgefüllt
- Die Blöcke werden nacheinander mit demselben Schlüssel $K$ einzeln verschlüsselt, übertragen und auf Empfängerseite wieder entschlüsselt:



# CBC-Modus (cipher block chaining)

- Um unbemerkte Manipulationen des Kryptotextes zu verhindern wird jeder Kryptotextblock nicht nur in Abhängigkeit des aktuellen Klartextblocks, sondern aller vorausgehenden Blöcken verschlüsselt
- Als Folge hiervon werden gleiche Klartextblöcke auf unterschiedliche Kryptotextblöcke abgebildet:

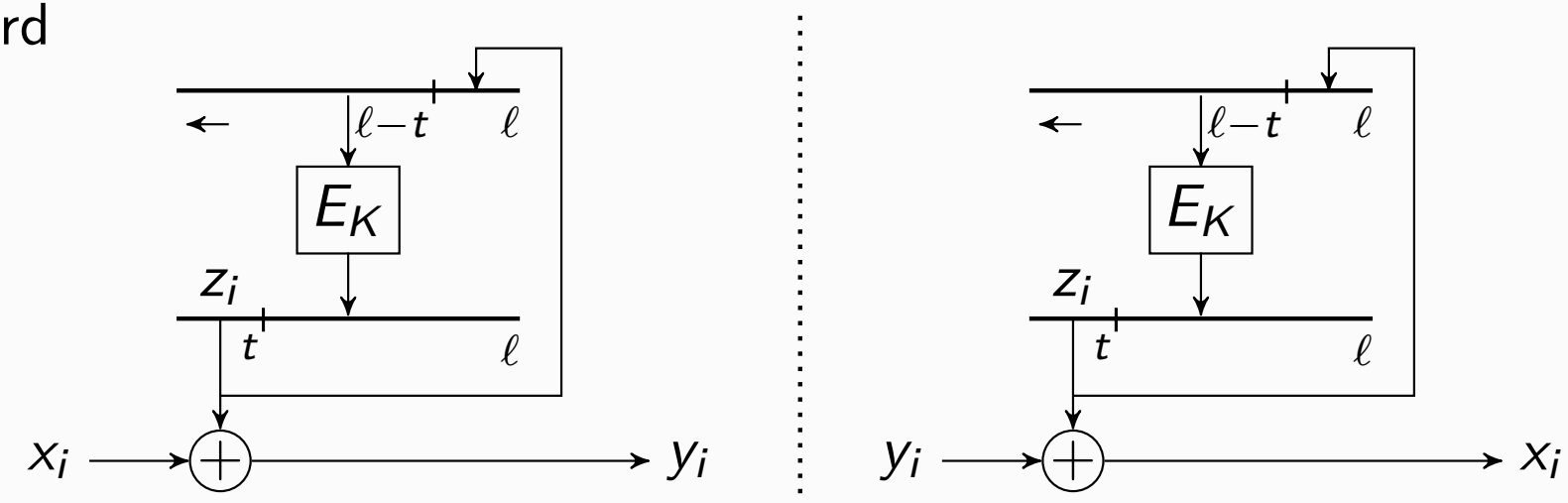


# CBC-Modus (cipher block chaining)

- Jeder Klartextblock $x_i$ wird mit dem Kryptotextblock $E_K(x_{i-1})$ bitweise (modulo 2) addiert, bevor er verschlüsselt wird
- Zur Verschlüsselung von $x_1$ wird ein Initialisierungsvektor $iv$ verwendet
- Dieser wird üblicherweise unverschlüsselt übertragen, sollte aber nicht mehrmals verwendet werden
- Er wird meist durch einen Pseudozufallsgenerator erzeugt
- Der CBC-Modus verhindert auch, dass sich bestimmte Klartextmuster direkt auf den Kryptotext übertragen (was im ECB-Modus der Fall ist)
- Ein extremes Beispiel ist die Verschlüsselung einer Graphikdatei $x$, deren Pixel durch $\ell$-Bit Blöcke kodiert sind
- Zwar werden dann im ECB-Modus die Pixel substituiert, aber ansonsten stellt der Kryptotext $y$ exakt dieselbe Graphik wie der Klartext $x$ dar
- Dadurch wird eine „visuelle Entschlüsselung“ durch Betrachten der verschlüsselten Graphikdatei $y$ ermöglicht

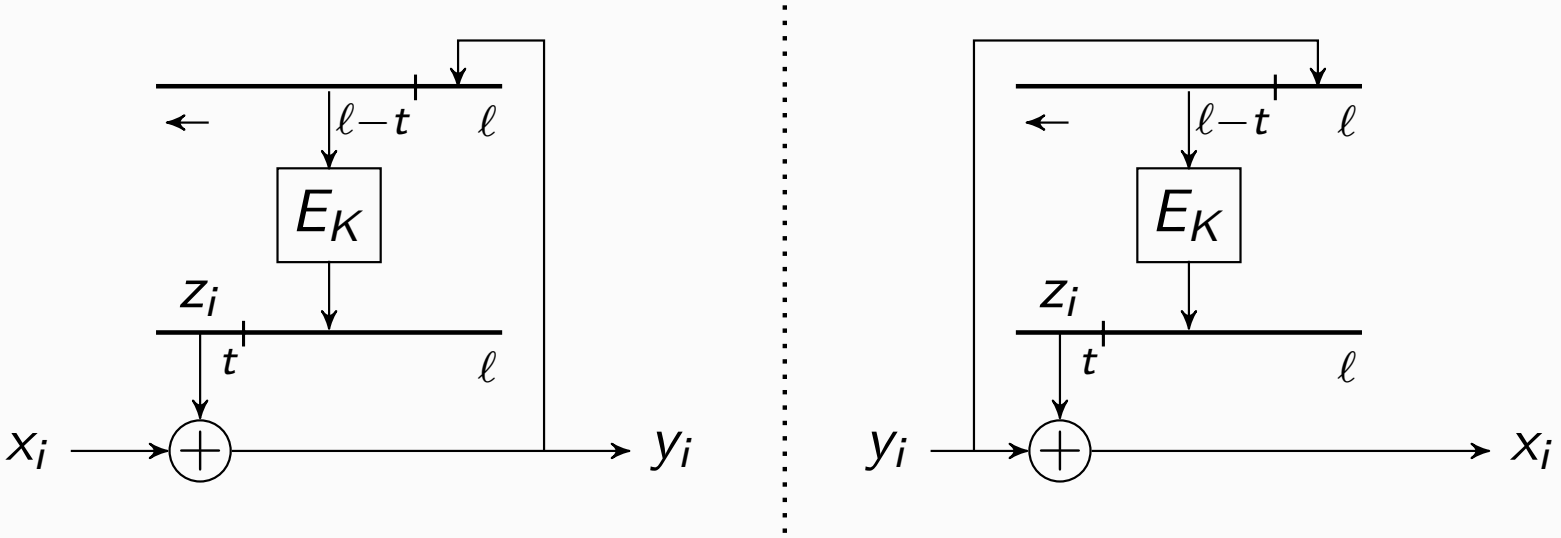
# OFB-Modus (output feedback)

- Die Binärnachricht $x$ wird in $t$-Bit Blöcke ($1 \leq t \leq \ell$) zerlegt
- Die Chiffrierfunktion $E_K$ erzeugt eine pseudozufällige Folge von Blöcken $z_i$, die auf die entsprechenden Klartextblöcke $x_i$ addiert werden
- Als Eingabe für die Chiffrierfunktion $E_K$ dient ein Schieberegister, das anfangs mit einem Initialisierungsvektor $iv$ geladen wird
- Zur Verschlüsselung jedes $t$-Bit Klartextblockes $x_i$ erzeugt $E_K$ zunächst einen Ausgabevektor, von dem nur die ersten $t$ Bits verwendet werden
- Mit diesem Block $z_i$ wird der Kryptoxtblock $y_i = x_i \oplus z_i$ berechnet und auch das Eingaberegister modifiziert, in das $z_i$ von rechts geschoben wird



# CFB-Modus (cipher feedback)

- Ähnlich zum OFB-Modus, nur dass zur Erneuerung des Eingaberegisters nicht die ersten $t$ Bits $z_i$ der $E_K$-Ausgabe, sondern der zugehörige $t$-Bit Kryptotextblock $y_i = x_i \oplus z_i$ verwendet wird



# CTR-Modus (counter mode)

- Bei dieser Variante des OFB-Modus' wird die Pseudozufallsfolge mit Hilfe von $E_K$ aus einer fortlaufenden Binärblockfolge $c_0, c_1, \ldots$ mit $c_{i+1} = c_i + 1 \bmod 2^\ell$ erzeugt
- Dies hat den Vorteil, dass spätere Blöcke der Pseudozufallsfolge nicht von den vorhergehenden abhängen
- Daher können mehrere Blöcke $E_K(c_i)$ parallel berechnet werden