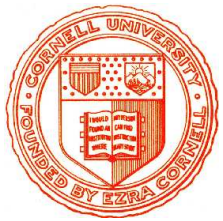
Kryptographie und Komplexität

Einheit 3



Praktisch sichere Blockchiffren

1. Substitutions-Permutations Netzwerke
2. Feistel-Chiffren und der DES
3. Der Advanced Encryption Standard AES

Sichere und effiziente Kryptosysteme

- **Absolute Sicherheit ist unerreichbar**
  - Perfekt geheime Systeme sind kostspielig und ineffizient
  - Unbrauchbar für alltägliche Sicherheitsanwendungen
- **Einfache Kryptosysteme sind nicht sicher**
  - Effiziente Chiffrierung, aber mit Computern leicht zu brechen
  - Codierung der Klartextblöcke ist keine echte Einwegfunktion und kann mit mathematischen Methoden invertiert werden
- **Moderne Systeme sind nichttriviale Blockchiffren**
  - Hohe Diffusion bei Codierung großer Datenblöcke
- **Effiziente Systeme verhärten einfache Systeme**
  - Aufwendige Kombination von Substitutionen und Permutationen
  - Mehrfachverschlüsselung mit wechselnden (Teil-)schlüsseln
  - Hardwarenahes Vorgehen benötigt symmetrische Verschlüsselung

Produktchiffren: Verhärtung von Blockchiffren

- **Leicht herzustellen aus bestehenden Systemen**
  - Schlüssel sind Paare einfacher Schlüssel: $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$
  - Verschlüsselungen nacheinander ausgeführt: $e_{(K_1,K_2)}(x) = e_{K_2}(e_{K_1}(x))$
  - Entschlüsselung in umgekehrter Reihenfolge: $d_{(K_1,K_2)}(y) = d_{K_1}(d_{K_2}(y))$
  - Notation: $\boldsymbol{S_1 \times S_2}$ für $S_i = (\mathcal{P}_i, \mathcal{C}_i, \mathcal{K}_i, e, d)$
- **Iteration ist die häufigste Form**
  - $\boldsymbol{S^n}$ = $S \times S \times .. \times S$ (n-faches Produkt von S mit sich selbst)
  - Benötigt endomorphe Kryptosysteme ($\mathcal{P} = \mathcal{C}$)
- **Eigenschaften leicht zu untersuchen**
  - Produktchiffren sind assoziativ: $e_{((K_1,K_2),K_3)}(x) = e_{(K_1,(K_2,K_3))}(x)$
  - Produktchiffren sind i.a. nicht kommutativ: $e_{(K_1,K_2))}(x) \neq e_{(K_2,K_1)}(x)$
  - Iteration nutzt wenig bei idempotenten Kryptosystemen ($S \times S = S$)
    in diesem Fall gibt es immer ein $K_3 \in \mathcal{K}$ mit $e_{(K_1,K_2)}(x) = e_{K_3}(x)$
  - Idempotenz bleibt erhalten bei kommutierenden Systemen
    $(S_1 \times S_2) \times (S_1 \times S_2) = S_1 \times S_2$, falls die S_i idempotent sind

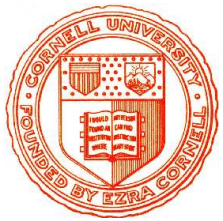
Aufbau iterativer Chiffren

- **Wiederholte Verschlüsselung mit Rundenfunktion**
  - Schlüssel K wird in n Teilschlüssel $K^1, .., K^n$ zerlegt

    Key-Scheduling Verfahren zur Erzeugung der K^r liegt offen
  - Rundenfunktion g erzeugt in Runde r mit K^r einen Zustand w^r
    - Anfangszustand ist Klartext: $w^0 = x$
    - Endzustand liefert Schlüsseltext: $y = w^n$
    - Runde r berechnet $w^r = g(w^{r-1}, K^r)$
- **Rundenfunktion muß injektiv in den K^r sein**
  - Es muß ein g^{-1} mit $g^{-1}(g(w,k),k) = w$ für alle k, w geben
  - Entschlüsselung mit gleichen Schlüsseln in umgekehrter Reihenfolge
    - Anfangszustand ist Schlüsseltext: $w^n = y$
    - Runde $n-r$ berechnet $w^r = g^{-1}(w^{r+1}, K^{r+1})$
    - Runde n liefert Klartext: $x = w^0$
- **Vermeide idempotente Kryptosysteme**
  - Wegen $S^n = S$ würde Iteration keine Verhärtung liefern

Kryptographie und Komplexität

Einheit 3.1



Substitutions-Permutations Netzwerke

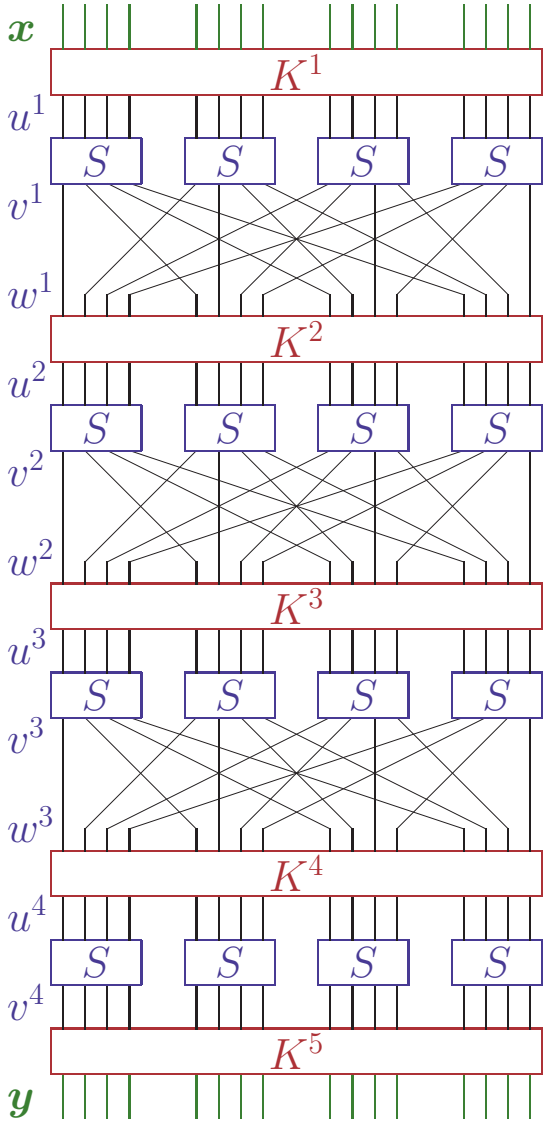
1. Architektur von SPNs
2. Lineare Kryptoanalyse
3. Differentielle Kryptoanalyse

Substitutions-Permutations Netzwerke

Iteration von Substitutionen und Permutationen

- **Rundenfunktion g hat drei Komponenten**
  - Binäre Addition des Teilschlüssels K^r: $u^r := w^{r-1} \oplus K^r$
  - Substitution von m Unterblöcken $u_{(i)}$ der Länge l: $v^r_{(i)} := \sigma(u^r_{(i)})$
  - Permutation aller Bits des entstehenden Blocks: $w^r_j := u^r_{\pi(j)}$
  - σ ist bijektive Substitution auf $\{0,1\}^l$, π Permutation auf $\{1..l{\cdot}m\}$
  - Länge der Blöcke (und Teilschlüssel) ist $l{\cdot}m$
- **Letzte Runde ist verändert**
  - Binäre Addition des Teilschlüssels K^n: $u^n := w^{n-1} \oplus K^n$
  - Substitution der Unterblöcke $u^n_{(i)}$: $v^n_{(i)} := \sigma(u^n_{(i)})$
  - Letzter Schritt Addition des Teilschlüssels K^{n+1}: $y := w^n \oplus K^{n+1}$
- **Einfache Implementierung in Soft-/Hardware**
  - Codierung von σ ("S-Box") und π als Tabelle (darf offenliegen)
  - Einfache Dechiffrierung von σ und π mit Umkehrtabelle
  - Auswahl der Bits von K^r aus K mit Schieberegister möglich

Ein einfaches SP-Netzwerk



K = 0011 1010 1001 0100 1101 0110 0011 1111

σ =

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

(hex.)

π = [1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16]

$x = w^0 =$	0010 0110 1011 0111
$K^1 =$	0011 1010 1001 0100
$u^1 =$	0001 1100 0010 0011
$v^1 =$	0100 0101 1101 0001
$w^1 =$	0010 1110 0000 0111
$K^2 =$	1010 1001 0100 1101
$u^2 =$	1000 0111 0100 1010
$v^2 =$	0011 1000 0010 0110
$w^2 =$	0100 0001 1011 1000
$K^3 =$	1001 0100 1101 0110
$u^3 =$	1101 0101 0110 1110
$v^3 =$	1001 1111 1011 0000
$w^3 =$	1110 0100 0110 1110
$K^4 =$	0100 1101 0110 0011
$u^4 =$	1010 1001 0000 1101
$v^4 =$	0110 1010 1110 1001
$K^5 =$	1101 0110 0011 1111
$y =$	1011 1100 1001 0110

Sicherheit von SP-Netzwerken

- **Alles außer dem Schlüssel liegt offen**
  - Key-Scheduling Verfahren zur Erzeugung der Teilschlüssel $K^1, .., K^n$
  - Struktur des Netzwerkes, Tabellen der S-Box σ und der Permutation π
- **S-Boxen sind einzige nichtlineare Komponenten**
  - Permutation und binäre Addition sind affin-lineare Abbildungen
  - Identische kleine S-Boxen führen zu annähernd linearen Abbildungen
- **Mögliche Attacken**
  - Brute-Force Attacken attackiert kleine Blocklängen
    geringe Anzahl von Runden oder kleine S-Boxen
  - Lineare Kryptoanalyse versucht, Chiffrierfunktion des SPN
    durch lineare Abbildungen zu approximieren
  - Differentielle Kryptoanalyse versucht, Einflüsse von Änderungen im Klartext auf Änderungen im Schlüsseltext statistisch zu analysieren

Lineare Kryptoanalyse – Idee

- **SPNs haben wenig nichtlineare Anteile**
  - Einzige nichtlineare Komponenten sind S-Boxen
- **Approximiere Chiffrierfunktion durch lineare Abbildung**
  - Ein Ausgabebit y_j könnte 'wahrscheinliche Linearkombination' der Eingabebits $x_1, .., x_m$ sein
  - D.h. die Wahrscheinlichkeit, daß $\sum_{k=1}^{m} a_k x_k$ und y_j für bestimmte $a_1, .., a_m$ den gleichen Wert annehmen, ist nicht 1/2 ($\hat{=}$ völliger Zufall)
  - Untersuche Wahrscheinlichkeit des Ereignisses $\sum_{k=1}^{m} a_k x_k \oplus y_j = 0$
- **Analysiere S-Boxen und Rundenfunktion**

  $\mapsto$ Approximation der Chiffrierung bis zur Addition von K^n in Runde n
- **Known plaintext (+ brute force) Attacke auf letzte Runde**
  - Für jeden möglichen Rundenschlüssel K^{n+1} und jedes Klartext-/Schlüsseltextpaar bestimme Zustand u^n der letzten Runde
  - Zähle, wie oft lineare Beziehung der relevanten Bits erfüllt ist
  - Schlüssel mit höchster Abweichung von 50% liefert Kandidat für K^{n+1}

Lineare Kryptoanalyse von SP-Netzwerken

- **Betrachte S-Box $S : \{0,1\}^m \rightarrow \{0,1\}^n$**
  - Eingaben $x_1, .., x_m$ sind gleichverteilt
  - Wenn $S(x_1, .., x_m) = (y_1, .., y_n)$, dann $Pr(x_1, .., x_m, y_1, .., y_n) = 2^{-m}$
  - Wenn $S(x_1, .., x_m) \neq (y_1, .., y_n)$, dann $Pr(x_1, .., x_m, y_1, .., y_n) = 0$
- **Bestimme Häufigkeit linearer Abhängigkeiten**
  - Für $a_1..a_m, b_1..b_n$ zähle, wie oft $\sum_{k=1}^{m} a_k x_k \oplus \sum_{k=1}^{n} b_k y_k = 0$ ist

    Alle 2^m Kombinationen der x_k müssen überprüft werden
  - Hohe Abhängigkeit besteht, wenn Häufigkeit stark von 2^{m-1} abweicht
  - Erstelle Häufigkeitstabelle für alle 2^{m+n} Kombinationen der a_k und b_k
- **Netzwerkanalyse verwendet Häufigkeitstabelle**
  - Verfolge Pfad der größten Abhängigkeit von x bis u^n
  - Bestimme "Abweichung vom Zufall" für diesen Pfad
  - Beschreibe Pfad durch Ein-/Ausgabebits und verwendete Schlüsselbits

Analyse der S-Box

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

- **Wie sehr hängt y_2 von x_1, x_4 ab?**
  - Wie oft ist $x_1 \oplus x_4 \oplus y_2 = 0$?
  - Summe der entsprechenden Spalten ist 8 mal 0
  - Wahrscheinlichkeit für Gleichheit von y_2 und $x_1 \oplus x_4$ ist 1/2
  - Keine Abweichung vom Zufall
  - Im Kontrast dazu: Wahrscheinlichkeit für Gleichheit von y_4 und $x_1 \oplus x_3$ ist 3/4 (große Abhängigkeit)
- **Erstelle Tabelle aller Kombinationen**
  - 256 Einträge liefern Werte zwischen 1/4 und 3/4
  - Werte der Approximationstabelle werden zu Werten für Abhängigkeitspfade zusammengesetzt

Mathematik: Fortpflanzung von Abhängigkeiten

- **Diskrete Zufallsvariable**
  - Variable X über endlicher Menge S mit Wahrscheinlichkeitsverteilung
  - Wahrscheinlichkeit $Pr(E)$ ist genau besehen die Wahrscheinlichkeit, daß X einen Wert aus E annimmt (also $Pr[X = x] := Pr(x)$)
  - Liefert präzisiere Formulierung von Wahrscheinlichkeiten
  - z.B. $Pr[X_1 \oplus X_4 \oplus Y_2 = 0]$ ist Wahrscheinlichkeit der Menge aller möglichen Werte x_1, x_4, y_2 von X_1, X_4, Y_2 mit $x_1 \oplus x_4 \oplus y_2 = 0$
- **Bias einer binären Zufallsvariablen X**
  - Abweichung der Variablen vom perfekten Zufall (Gleichverteilung)
  - $\epsilon_X := Pr[X{=}0] - \frac{1}{2}$
- **Piling up Lemma**

  Sind $X_1, .., X_k$ unabhängige Variablen mit Bias ϵ_i und $\epsilon_{1,..,k}$ Bias von $X_1 \oplus .. \oplus X_k$, dann ist $\epsilon_{1,..,k} = 2^{k-1} \prod_{i=1}^{k} \epsilon_i$
  - Abhängigkeiten in einem Netzwerk pflanzen sich multiplikativ fort

Mathematik: Beweis des Piling up Lemmas

Sind $X_1, .., X_k$ unabhängige Variablen mit Bias ϵ_i und $\epsilon_{1,..,k}$
Bias von $X_1 \oplus .. \oplus X_k$, dann ist $\epsilon_{1,..,k} = 2^{k-1} \prod_{i=1}^{k} \epsilon_i$

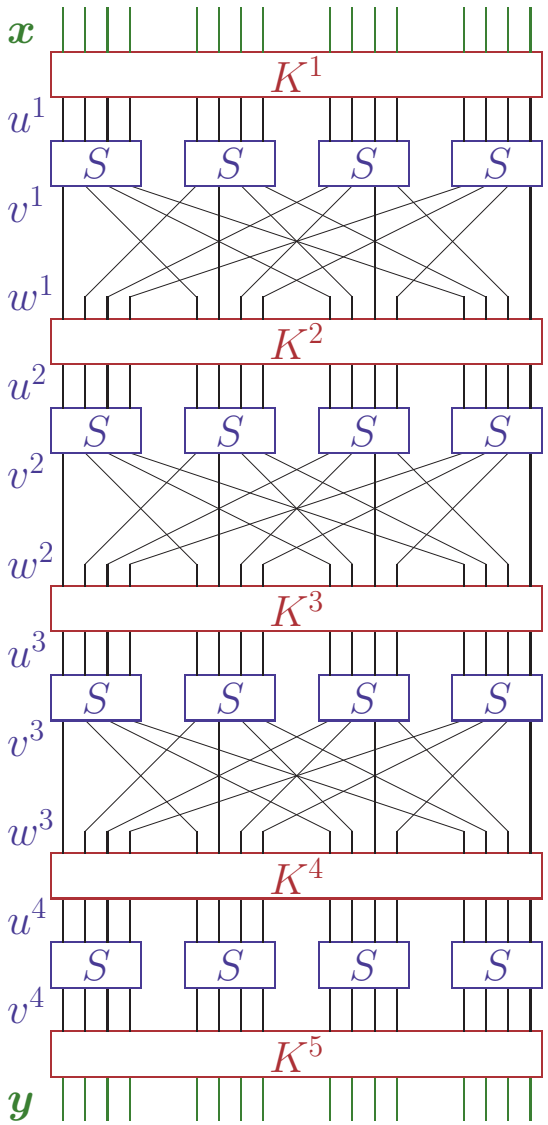
- **Beweis durch Induktion über k**
  - Aussage ist trivialerweise wahr für $k{=}1$
  - Wir nehmen an $\epsilon_{1,..,k} = 2^{k-1} \prod_{i=1}^{k} \epsilon_i$ für ein $k{\geq}1$
  - Dann gilt

$$
\begin{aligned}
& Pr[X_1 \oplus .. \oplus X_{k+1}{=}0] \\
= \; & Pr[X_1 \oplus .. \oplus X_k{=}0] \cdot Pr[X_{k+1}{=}0] + Pr[X_1 \oplus .. \oplus X_k{=}1] \cdot Pr[X_{k+1}{=}1] \\
= \; & (\tfrac{1}{2} + \epsilon_{1,..,k})(\tfrac{1}{2} + \epsilon_{k+1}) + (\tfrac{1}{2} - \epsilon_{1,..,k})(\tfrac{1}{2} - \epsilon_{k+1}) \\
= \; & \tfrac{1}{2} + 2\epsilon_{1,..,k}\epsilon_{k+1}) \\
= \; & \tfrac{1}{2} + 2^k \prod_{i=1}^{k+1} \epsilon_i
\end{aligned}
$$

- **Korollar: Perfekter Zufall pflanzt sich fort**

  Ist $\epsilon_j = 0$ für ein j, dann ist $\boldsymbol{Pr[X_1 \oplus .. \oplus X_{k+1}{=}0] = \frac{1}{2}}$

Lineare Analyse eines SP-Netzwerkes



- **Verfolge die größten Biaswerte**
  - In $S^1_{(2)}$: $T_1 = U^1_5 \oplus U^1_7 \oplus U^1_8 \oplus V^1_6$ hat Bias $\frac{1}{4}$
    V^1_6 wird permutiert zu $U^2_6 = V^1_6 \oplus K^2_6$
  - In $S^2_{(2)}$: $T_2 = U^2_6 \oplus V^2_6 \oplus V^2_8$ hat Bias $-\frac{1}{4}$
    Permutation: $U^3_6 = V^2_6 \oplus K^3_6$, $U^3_{14} = V^2_8 \oplus K^3_{14}$
  - In $S^3_{(2)}$: $T_3 = U^3_6 \oplus V^3_6 \oplus V^3_8$ hat Bias $-\frac{1}{4}$
  - In $S^3_{(4)}$: $T_4 = U^3_{14} \oplus V^3_{14} \oplus V^3_{16}$ hat Bias $-\frac{1}{4}$
- **Pfad $T_1 \oplus T_2 \oplus T_3 \oplus T_4$ hat Bias $-\frac{1}{32}$**
  - Setze ein: $U^1_j = X_j \oplus K^1_j$, $V^3_6 = U^4_6 \oplus K^4_6$, …
    $X_5 \oplus X_7 \oplus X_8 \oplus U^4_6 \oplus U^4_8 \oplus U^4_{14} \oplus U^4_{16} \oplus$
    $K^1_5 \oplus K^1_7 \oplus K^1_8 \oplus K^2_6 \oplus K^3_6 \oplus K^3_{14} \oplus K^4_6 \oplus K^4_8 \oplus K^4_{14} \oplus K^4_{16}$
- **Schlüsselbits sind fest**
  - $X_5 \oplus X_7 \oplus X_8 \oplus U^4_6 \oplus U^4_8 \oplus U^4_{14} \oplus U^4_{16}$
    hat Bias $-\frac{1}{32}$ (Schlüsselbits 0) oder $\frac{1}{32}$

Abhängigkeit zwischen $x_{(2)}$ und $u^4_{(2,4)}$ nicht zufällig

Lineare Kryptoanalyse: Schlüsselbitextraktion

- **Abhängigkeit $x_{(2)}/u^4_{(2,4)}$ liefert 8 Schlüsselbits**
  - Überprüfe $K^5_5, K^5_6, K^5_7, K^5_8, \ K^5_{13}, K^5_{14}, K^5_{15}, K^5_{16}$ (256 Kandidaten)
- **Known plaintext Attacke**
  - Analysiere alle Kandidaten $K^5_{(2,4)}$
  - Für Klar-/Schlüsseltextpaare (x, y) berechne $u^4_{(2,4)} = S^{-1}(y_{(2,4)} \oplus K^5_{(2,4)})$
  - Berechne, wie oft $x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16}$ den Wert 0 ergibt
  - Kandidaten, bei denen die relative Häufigkeit nahe bei $\frac{1}{2} - \frac{1}{32}$ oder bei $\frac{1}{2} + \frac{1}{32}$ liegt, sind wahrscheinlich korrekte Rundenschlüsselteile
- **Aufwendige aber durchführbare Attacke**
  - Anzahl notwendiger Klar-/Schlüsseltextpaare liegt in $\mathcal{O}(\epsilon^{-2})$
  - Konkreter Angriff benötigt 8000 Paare um die 8 bits zu bestimmen
  - Andere Pfade liefern weitere Rundenschlüsselbits (ϵ ist kleiner)

S-Boxen und Rundenzahl müssen größer sein um Attacke zu erschweren

Differentielle Kryptoanalyse

Analysiere Differenzen zwischen Klartexten

- **Chosen plaintext Attacke**
  - Wähle Klartextpaare x, x^* mit fester Differenz $x' = x \oplus x^*$
  - Bestimme Differenz $y' = y \oplus y^*$ der zugehörigen Schlüsseltexte
  - Zähle, welche Differenz am häufigsten erzeugt wird
  - Für kleine S-Boxen werden alle Klartextdifferenzen und alle Klartexte analysiert und Häufigkeiten in Differenzentabelle gespeichert
- **Netzwerkanalyse verfolgt Differenzenpfad**
  - Starte mit Klartextdifferenz mit großer Fortpflanzungsrate
  - Verfolge zugehörige Ausgabedifferenz der S-Box durch das Netz
    - Permutationen verteilen Differenz über mehrere S-Boxen
    - Addieren des Schlüssels hat keinen Einfluß auf die Differenz
  - Bestimme Fortpflanzungsrate des gesamten Pfades von x' bis $u^{n\prime}$
  - Extrahiere Schlüsselbits wie bei linearer Analyse

Differentielle Kryptoanalyse im Detail

- **Bestimme Eingabedifferenzen einer S-Box**
  - Eingabedifferenz von x, x^* ist $x' = x \oplus x^*$
  - $\Delta(x')$ = $\{(x, x^*) \mid x' = x \oplus x^*\}$ Menge der Paare mit Eingabedifferenz x'

    Für $S : \{0,1\}^m \to \{0,1\}^n$ hat jedes $\Delta(x')$ genau 2^m Elemente
- **Bestimme Differenzen bei Ein-/Ausgaben einer S-Box**
  - Ausgabedifferenz von x, x^* ist $y' = S(x) \oplus S(x^*)$
  - $D(x', y')$ = $\{(x, x^*) \in \Delta(x') \mid y' = S(x) \oplus S(x^*)\}$

    Menge der Paare, die zu einer festen Ausgabedifferenz y' führen
  - Anzahl der Elemente von $D(x', y')$ ist meistens nicht gleichverteilt
- **Berechne Verteilung der $D(x', y')$ für S**
  - Liefert Informationen über Abhängigkeiten im SP-Netzwerk
  - Abhängigkeiten müssen durch Permutationen verfolgt werden
  - Schlüsseladdition $u^r := w^{r-1} \oplus K^r$ r läßt Differenzen unverändert

    Es ist $u^r \oplus u^{r\,*} = (w^{r-1} \oplus K^r) \oplus (w^{r-1\,*} \oplus K^r) = w^{r-1} \oplus w^{r-1\,*}$

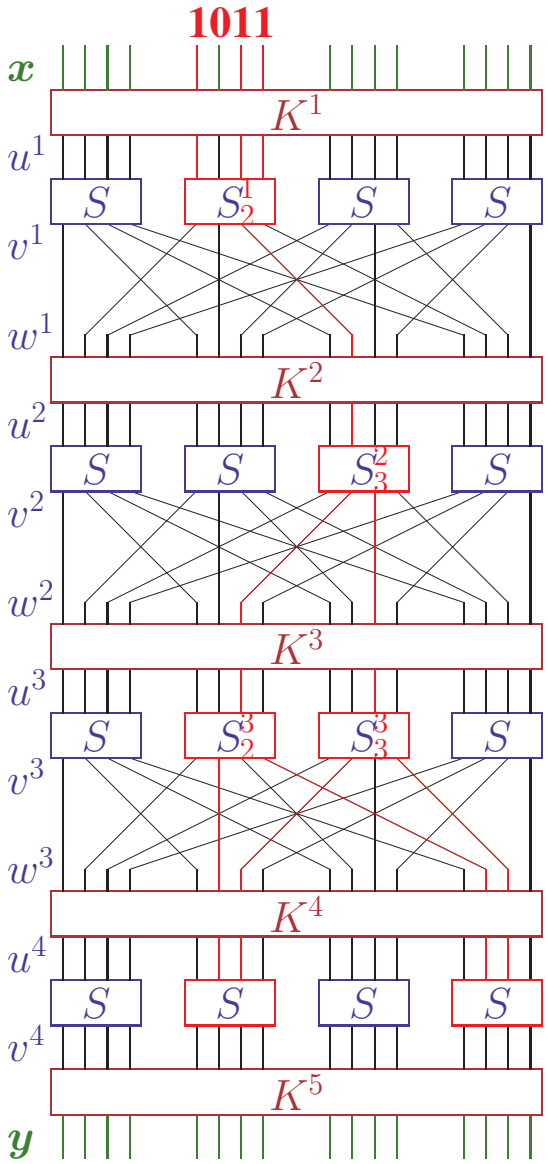
Analyse der S-Box

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

x	x^*	y	y^*	y'
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
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

- **Berechne $D(x', y')$ für $x' = 1011$**
  - Es kommen nur 5 verschiedene Ausgabedifferenzen vor
  - 0010 erscheint 8 mal
- **Berechne Fortpflanzungsrate eines Differentials (x', y')**
  - Wahrscheinlichkeit der Ausgabedifferenz y', wenn Eingabedifferenz x' vorliegt
  - $R_p(x', y') = |D(x', y')|/2^m$
    z.B. $R_p(1011, 0010) = 1/2$
    $R_p(1011, 1101) = 1/8$
    $R_p(1011, 0000) = 0$
  - $R_p(0000, 0000) = 1$ gilt für jede S-Box
  - R_p muß für alle $2^m * 2^n$ Differentiale berechnet werden

Differentielle Analyse eines SP-Netzwerkes



- **Verfolge Pfad der Ausgabedifferenzen**
  - Kritische Differenz wird an eine S-Box angelegt
    alle anderen S-Boxen erhalten 0000
  - Analysiere Weg der Einerbits durch das Netzwerk
    Multipliziere Fortpflanzungsraten auf
  - Bestimme Wahrscheinlichkeit verschiedener $u^{4\prime}$
- **Verfolge Pfade der Differenz** 1011
  - Differenz passiert Schlüssel K^1 unverändert
  - Häufigste Ausgabedifferenz in S_2^1 ist 0010
  - Einerbit erscheint als zweites Bit in S_3^2
  - Häufigste Ausgabedifferenz für 0100 ist 0110
  - Einerbits erscheinen als drittes Bit in S_2^3 und S_3^3
  - Häufigste Ausgabedifferenz für 0010 ist 0101
  - Einerbits erscheinen als zweites/drittes Bit in $u^{4\prime}$
  - $x' =$ 0000 1011 0000 0000 liefert
    $u^{4\prime} =$ 0000 0110 0000 0110
    als wahrscheinlichste "Enddifferenz"

Differentielle Analyse: Schlüsselextraktion

Known plaintext Attacke auf letzte Runde

- **Untersuche Menge sinnvoller Tupel (x, x^*, y, y^*)**
  - (x, x^*) müssen in $\Delta(0000101100000000)$ liegen
  - Erster und dritter Block von y und y' müssen identisch sein
    - $u^{4\prime} = 0000\ 0110\ 0000\ 0110$ ist wahrscheinlichste Differenz
    - Schlüsseladdition und S-Boxen lassen Differenz 0000 unverändert
- **Teste alle Teilschlüssel für Blöcke 2 und 4**
  - Durch Rückwärtsanwendung von Schlüssel und S-Box auf y, y^* bestimme Blöcke u_2, u_4 bzw. u_2^*, u_4^* und berechne Differenz u_2' und u_4'
  - Zähle, für wieviele sinnvolle Tupel der Sollwert 0110 getroffen wird
  - Teilschlüssel mit höchster Trefferquote ist wahrscheinlich korrekt
  - Bestimme weitere Teilschlüssel durch andere Initialdifferenzen
- **Aufwand abhängig von Diffusion & Fortpflanzungsrate**
  - Bei mäßiger Diffusion hat u_4' mehrere 0000-Blöcke und es müssen weniger Teilschlüssel geprüft werden (2^{k*n} bei k nicht-0 Blöcken)
  - Bei Fortpflanzungsrate ϵ sind $\mathcal{O}(\epsilon^{-1})$ Tupel nötig ($100 * \epsilon^{-1}$ reicht)

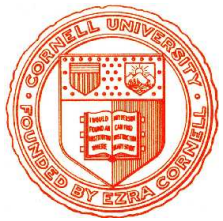
Fazit: Sicherheit von SP-Netzwerken

- **Brute-Force Attacken**
  - Möglich bei zu kleinen Blocklängen (32 Bit), geringer Anzahl von Runden oder kleinen S-Boxen
- **Lineare Kryptoanalyse**
  - Bestimmt wahrscheinliche Schlüsselbits durch lineare Approximation der Chiffrierfunktion des SPN Abbildung
  - Erfolgreich, wenn Chiffrierfunktion nahezu linear und "einfach"
- **Differentielle Kryptoanalyse**
  - Analysiert Einflüsse von Änderungen im Klartext auf Änderungen im Schlüsseltext und extrahiert Schlüsselfragmente
  - Erfolgreich, wenn Diffusion des SPN nicht hoch genug
- **Praktische Versionen erhöhen Diffusion und Konfusion**

  Sicherheit gegenüber bekannten Attacken steht im Vordergrund
  - Verwendung verschiedenartiger S-Boxen $\mapsto$ DES
  - Verwendung zusätzlicher linearer Transformationen $\mapsto$ AES

Kryptographie und Komplexität

Einheit 3.2



Der Data Encryption Standard

1. Feistel-Chiffren
2. DES Architektur und Komponenten
3. Sicherheit des DES

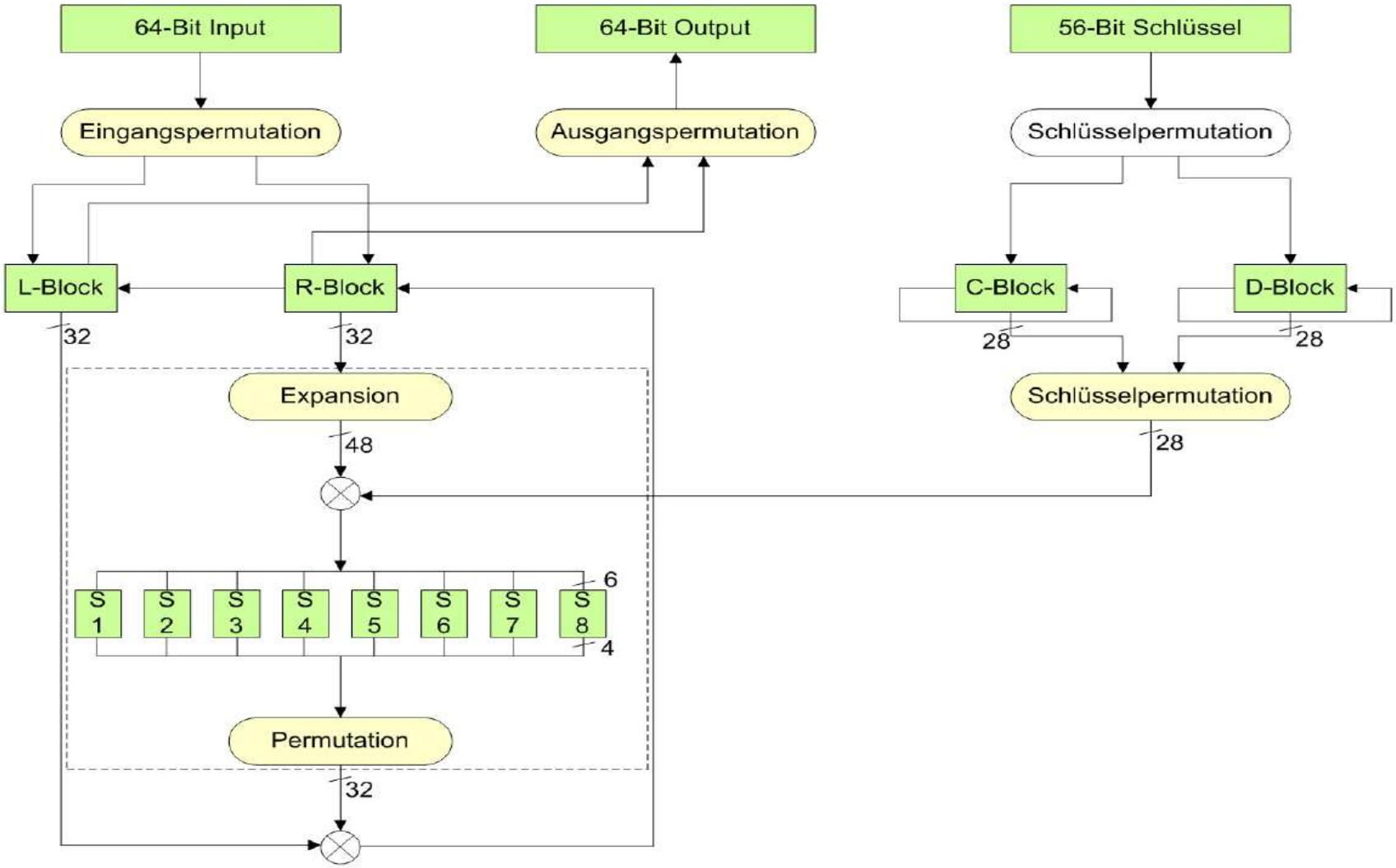
Feistel Chiffren

- **Aufteilung des Klartextes in zwei Teilblöcke**
  - Ein Block x der Länge $2l$ wird in zwei gleichgroße Teile L^0, R^0 zerlegt
  - Zuvor kann initiale Operation (z.B. Permutation) Konfusion erhöhen
- **Rundenfunktion g verarbeitet beide Teilblöcke**
  - Rechter Teilblock geht nach links: $L^r := R^{r-1}$
  - Rechter Teilblock wird verschlüsselt: $u^r := f(R^{r-1}, K^r)$
  - Binäre Addition des linken Teilblocks: $R^r := L^{r-1} \oplus u^r$
  - f ist beliebige Verschlüsselungsfunktion auf $\{0,1\}^l \times \mathcal{K}$
  - In Runde n wird $y := L^n R^n$ zusammengesetzt
- **Einfache Entschlüsselung**
  - Rechter Teilblock kommt von links: $R^{r-1} = L^r$
  - Linker Teilblock kommt von links: $L^{r-1} = R^r \oplus f(L^r, K^r)$

Der Data Encryption Standard (DES)

- **Erfolgreichstes Verschlüsselungsverfahren**
  - 1975 von IBM im Zuge einer öffentlichen Ausschreibung vorgestellt
  - 1977 in den USA als nationaler Standard zertifiziert
    mit einer geschätzten Lebensdauer von 10–15 Jahren
  - Vielfältiger Einsatz in Soft- und Hardwaremodulen
  - Erst 2000 durch den Advanced Encryption Standard (AES) abgelöst
- **Modifizierte Feistel-Chiffre mit 64 Bit Schlüssel**
  - Netto-Schlüssellänge nur 56 Bit (8 Bit Parität)
  - Zusätzliche initiale Permutation im ersten Schritt
  - 16 Iterationsrunden
  - Inverse initiale Permutation im letzten Schritt
  - Rechte und linke Teilblöcke werden auf je 48 Bit expandiert
  - 8 verschiedene S-Boxen mit je 6 Bit Eingabe / 4 Bit Ausgabe

Aufbau des DES



Komponenten des DES

- **Klartext- und Schlüsselraum**
  - 64 Bit Blöcke ($\{0,1\}^{64}$), üblicherweise hexadezimal notiert
- **Initiale Permutation IP:** $\{0,1\}^{64} \to \{0,1\}^{64}$
  - Festgewählte Bitpermutation auf Vektoren der Länge 64 (Tabelle)

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

  - Keine erkennbare Methodik für Auswahl der spezifischen Permutation
- **Expansionsfunktion E:** $\{0,1\}^{32} \to \{0,1\}^{48}$
  - Beschrieben durch festgewählte Expansionstabelle mit 48 Einträgen
- **Acht S-Boxen S_i:** $\{0,1\}^{6} \to \{0,1\}^{4}$
  - Je 4 festgewählte Permutationstabellen auf $\{0,1\}^4$
  - Erstes und letztes Bit eines 6er-Blocks bestimmt Auswahl der Tabelle
- **Permutation P:** $\{0,1\}^{32} \to \{0,1\}^{32}$
  - Festgewählte Bitpermutation auf Vektoren der Länge 32 (Tabelle)

Bestimmung der Rundenschlüssel des DES

- **Auswahl von zweimal 28 Bit aus 64**
  - Paritätsbits 8 16 24 32 40 48 56 64 werden entfernt
  - Tabelle PC1 ordnet je 28 Bit den Teilschlüsselblöcken C_0 und D_0 zu
- **Zirkuläre Verschiebung in jeder Runde**
  - In Runde i bestimme C_i und D_i aus C_{i-1} bzw. D_{i-1} durch zirkuläre Linksverschiebung um 2 Stellen (nur eine Stelle in Runde 1,2,9,16)
- **Auswahl von 48 Bit aus zweimal 28 Bit**
  - Tabelle PC2 wählt 48 Bit aus den Teilschlüsselblöcken C_i und D_i aus
  - Resultierender Schlüssel K_i wird in Runde i zum Resultat der Expansion binär addiert und an S-Boxen übergeben

Eigenschaften des DES

- **Einfache Entschlüsselung des Chiffretextes**
  - Prinzip der Feistel-Chiffren: $R^{r-1} = L^r,\ L^{r-1} = R^r \oplus f(L^r, K^r)$
  - Ausführung des DES mit vertauschten Teiltextblöcken
  - Anwendung der Rundenschlüssel in umgekehrter Reihenfolge
- **Schnelle Verarbeitung großer Datenmengen**
  - Alle Bestandteile sind als Hardwarekomponenten realisierbar
- **Sicherheitsaspekte**
  - Sicherheit basiert auf S-Boxen (einzige nichtlineare Komponente)
    Spezifische Tabellen sollten differentielle Analyse erschweren
  - 1994: Lineare Kryptoanalyse mit 2^{43} Klartext-/Schlüsseltextpaaren
    benötigt 10 Tage mit 12 Workstations, um Schlüssel zu brechen
  - 56 Bit Schlüssellänge erlaubt Brute-Force Attacken mit Spezialhardware
    1999: 100000 PC's benötigen nur 22 Stunden (DES Challenge III)

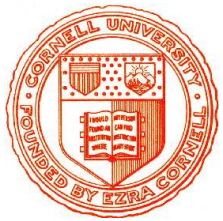
Fazit: Sicherheit des DES

- **Besser als ursprünglich erwartet**
  - Ursprünglich geschätzte Verwendung bis 1992
  - Für normale Sicherheitsbedürfnisse bis 2000 mehr als ausreichend
    Geldautomaten, Sprachverschlüsselung im Mobilfunk, ...
  - Erweiterung Triple-DES gilt heute noch als sicher
- **Aus heutiger Sicht nicht mehr sicher genug**
  - Verfahren hat zu wenig nichtlineare Anteile (kleine S-Boxen)
    Lineare und differentielle Analysen werden praktisch durchführbar
  - Geringe Blocklänge ermöglicht Brute-Force Attacken
  - FPGA-basierte Parallelrechner können "preiswerte" Attacken
    durchführen (Copacobana, 2006, Materialkosten $10.000, 9 Tage)
  - Unterschiedliche Schlüssel führen zu unterschiedlichen Laufzeiten
    Power-/Timing-Attacken extrahieren Teilschlüssel aus DES Hardware

DES-Schwächen sind Anforderungen an Nachfolger (AES)

Kryptographie und Komplexität

Einheit 3.3



Der Advanced Encryption Standard

1. Aufbau und Komponenten des AES
2. Polynomringe über endlichen Körpern
3. Sicherheit des AES

Der Advanced Encryption Standard

- **Aktueller Standard für 'normale' Anwendungen**
  - 1998 im Rahmen einer öffentlichen Ausschreibung entwickelt
  - 2000 in öffentlichem Verfahren ausgewählt
  - Vollständig offengelegt und frei verfügbar (ohne Patentansprüche)
  - Einsatz in nahezu allen Anwendungen mit großem Datendurchsatz
- **Entwurfs- und Auswahlkriterien**
  - Blocklänge mindestens 128 Bit
  - Unterstützung für Schlüssellänge 128, 192 und 256
  - Algorithmische Qualität: leichte Umsetzung in Hard- und Software
  - Effizienz: überdurchschnittlich hohe Performanz in Hard- und Software
    Geringer Ressourcenbedarf
  - Sicherheit gegenüber allen bekannten Attacken,
    auch gegenüber Power- und Timingattacken auf Hardware

Aufbau des AES (Rijndael Algorithmus)

- **Modifiziertes SP Netzwerk**
  - Feste Blocklänge 128 Bit, üblicherweise notiert als 16 bytes
  - Rundenanzahl abhängig von Schlüssellänge
    10 Runden bei 128 Bit, 12 bei 192 Bit, 14 bei 256 Bit
- **Runden haben vier Komponenten**
  - Binäre Addition des Rundenschlüssels K^r — AddRoundKey
  - Anwendung einer 8x8 S-Box auf 16 Byte-Unterblöcke — SubBytes
  - Permutation aller Bits des entstehenden 128-Bit Blocks — ShiftRows
  - Anwendung einer zusätzlichen linearen Transformation — MixColumns
  - Konkrete Komponenten wurden über algebraische Operationen entwickelt, sind aber als Tabellen dargestellt
  - Alle Operationen sind invertierbar
- **Letzte Runde ohne Transformation**
  - Ausführung von AddRoundKey, SubBytes, ShiftRows, AddRoundKey

Bestimmung der Rundenschlüssel des AES

- **Key-Scheduling berechnet 44 Worte a' 32 Bit**
  - Je 4 Worte werden zu einem Rundenschlüssel zusammengesetzt
- **Initialisierung der ersten 4 Worte aus Schlüssel**
  - w_0 := erste vier Bytes von K, w_1 := zweite vier Bytes, usw.
- **Iterative Berechnung von $w_4..w_{43}$**
  - w_i berechnet sich aus $w_{i-1} \oplus w_{i-4}$
  - In jeder vierten Runde wird w_{i-1} zuvor modifiziert durch
    - Zyklyschen Linksshift der Bytes — RotWord
    - Anwendung der AES S-Box auf jedes entstehende Byte — SubWord
    - Binäre Addition eines vordefinierten Wortes — RCon[i/4]
- **Modifizierte Version für 192/256 Bit Schlüssel**
  - Es werden 52 bzw. 60 Worte erzeugt (12 bzw. 14 Rundenschlüssel)
  - Initiale Auswahl kann 6 bzw. 8 Worte generieren

Key-Expansion Algorithmus des AES

```
proc KeyExpansion(K, w);
begin external RotWord; SubWord;
  RCon[1] := 01000000;  RCon[2] := 02000000;
  RCon[3] := 04000000;  RCon[4] := 08000000;
  RCon[5] := 10000000;  RCon[6] := 20000000;
  RCon[7] := 40000000;  RCon[8] := 80000000;
  RCon[9] := 1B000000; RCon[10] := 36000000;
  for i := 0 to 3 do
    w[i] := (K[4i]; K[4i+1]; K[4i+2]; K[4i+3])
  endfor;
  for i := 4 to 43 do
    temp := w[i-1];
    if i mod 4 = 0
       then temp := SubWord(RotWord(temp)) XOR RCon[i/4];
    w[i] := w[i-4] XOR temp
  endfor;
  return (w[0] ... w[43])
end
```

Komponenten des AES

- **S-Box** SubBytes**:** $\{0,1\}^8 \to \{0,1\}^8$
  - Bytes werden mit Elementen des endlichen Körpers $GF(2^8)$ identifiziert

    $b = (b_7, .., b_0)$ entspricht dem Polynom $\sum_{i=0}^{7} b_i x^i$
  - Aus einem Byte b berechnet SubBytes den Wert $b^{-1} \star A \oplus c$

    wobei $A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ und $c = (1,1,0,0,0,1,1,0)$
  - Multiplikation/Invertierung wird modulo $x^8 + x^4 + x^3 + x + 1$ ausgeführt
  - Abbildung erzeugt Nichtlinearität und wird als Tabelle gespeichert
- **Erforderliche Hintergrundmathematik**
  - Polynomringe über endlichen Körpern
  - Erzeugung endlicher Körper (Galoistheorie)

Körper der Polynome mit Koeffizienten aus $\mathbb{Z}_p$

- **Für jede Primzahl p ist $(\mathbb{Z}_p, +_p, \cdot_p)$ ein Körper**
  - $(\mathbb{Z}_p, +_p)$ ist abelsche Gruppe der Ordnung p
  - $(\mathbb{Z}_p^*, \cdot_p)$ ist abelsche Gruppe der Ordnung $p-1$
  - Das Distributivgesetz gilt für $+_p$ und $\cdot_p$
- **$K[x]$: Polynome über Körper K in Variable x**
  - Ausdrücke der Form $f(x) = a_n x^n + a_{n-1} x^{n-1} + ..a_1 x + a_0$
    mit Koeffizienten a_i und Belegungen der Variablen x aus K
  - n ist der Grad von f ($n = deg\ f$)
  - Monome sind Polynome, für die $a_i{=}0$ für alle $i{\neq}n$ gilt
  - Eine Nullstelle von f ist ein Element $r \in K$ mit $f(r) = 0$
    z.B. hat $f(x) = x^2{+}1$ über $\mathbb{Q}$ keine Nullstelle, über $\mathbb{Z}_2$ die Nullstelle 1

Mathematik: Polynomringe

- **Addition und Multiplikation in $K[x]$**
  - Sei $f(x) = \sum_{i=0}^{m} a_i x^i$ und $g(x) = \sum_{i=0}^{n} b_i x^i$ (o.B.d.A. $n \geq m$)
    - $(f+g)(x) = \sum_{i=0}^{n} (a_i + b_i) x^i$ $\mathcal{O}(n)$ Additionen
    - $(f \cdot g)(x) = \sum_{i=0}^{n+m} c_i x^i$ mit $c_k = \sum_{i=0}^{k} a_i b_{k-i}$ $\mathcal{O}(n \cdot m)$ Add./Mult.
  - z.B. ist $(x^2+2x+1) \cdot (x^3-2x^2+2) = x^5-3x^3+4x+2$
- **$(K[x], +, \cdot)$ ist ein Ring mit Division**
  - $(K[x], +)$ ist eine abelsche Gruppe
  - $(K[x], \cdot)$ ist nullteilerfreie abelsche Halbgruppe mit Einselement 1
  - Das Distributivgesetz gilt für $+$ und $\cdot$
  - Division: Für $f, g \in K[x]$ gibt es eindeutige Polynome $q, r \in K[x]$ mit $f = q \cdot g + r$ und $r = 0$ oder $deg\ r < deg\ g$
  - Beweis: Konstruktion durch schriftliche Division $\mathcal{O}(m \cdot (n-m))$ Ops.
  - Bezeichung: $q = \lfloor f/g \rfloor$, $r = f \bmod g$
  - z.B. $\lfloor x^3-2x^2+2 \ / \ x^2+2 \rfloor = x-2$, $x^3-2x^2+2 \bmod x^2+2 = -2x+6$

Mathematik: Polynomkörper

- **Konstruktion verallgemeinert $\mathbb{Z}_p$**
  - Restklassen modulo eines Ringelements $f \in K[x]$ (anstelle von $p \in \mathbb{Z}$)
  - Ringelement f muß **irreduzibel** sein (anstelle von "p Primzahl")
    d.h. f darf nicht durch ein g mit $deg\ f > deg\ g \geq 1$ teilbar sein
  - z.B. ist $f_1(x) = x^3+1$ reduzibel in $\mathbb{Z}_2[x]$, da $f_1(x) = (x+1)(x^2+x+1)$
  - $h(x) = x^2+x+1$ ist irreduzibel in $\mathbb{Z}_2[x]$, da $h \equiv 1 \in \mathbb{Z}_2$, aber für jeden echten Teiler $g(x) = x+a$ der Wert a Nullstelle von h in $\mathbb{Z}_2$ wäre
- **$(K[x]/f, +, \cdot)$ ist Körper mit $|K|^{deg\ f}$ Elementen**
  - $g \equiv h \bmod f$ falls $f \mid g-h$
  - $[g]_f = g+f \cdot K[x] := \{h \mid h \equiv g \bmod f\}$ ist die Restklasse von g modulo f
  - $K[x]/f$ ist die Menge aller Restklassen modulo f
  - $K[x]/f$ ist Körper, da zu jedem $0 \neq g \in K[x]/f$ ein Inverses mit dem erweiterten euklidischen Algorithmus konstruiert werden kann (Folie 14, §2.1)
  - $|K[x]/f| = |K|^{deg\ f}$, weil jedes $[g]_f$ ein h mit $deg\ h < deg\ f$ enthält und die Restklassen dieser Polynome paarweise verschieden sind

Mathematik: Polynomkörper der Grösse p^n

- **Eindeutige Konstruktion möglich**
  - Wähle Körper K mit p Elementen, wobei p Primzahl (z.B. $(\mathbb{Z}_p, +_p, \cdot_p)$)
  - Bestimme ein irreduzibles Polynom $f \in K[x]$ mit Grad n
  - $\boldsymbol{GF(p^n)} = K[x]/f$ ist bis auf Isomorphie eindeutig bestimmt
    Insbesondere ist $GF(p)$ isomorph zu $(\mathbb{Z}_p, +_p, \cdot_p)$
  - Satz: Für jeden endlichen Körper K gibt es eine Primzahl p sodaß
    K isomorph zu $GF(p^n)$ für ein n ist
- **Endliche Körper sind zyklisch** (Buchmann, Theorem 3.21.1)
  - Satz: Ist $(K, +, \cdot)$ Körper und $q = |K|$ so ist $(K^*, \cdot)$ eine zyklische
    Gruppe der Ordnung $q-1$ mit $\varphi(q-1)$ Erzeugern
  - $(GF(p^n), \cdot)$ ist eine zyklische Gruppe der Ordnung p^n-1
  - $GF(2^8)$ ist zyklische Gruppe der Ordnung 255
    mit erzeugendem Polynom $g(x) = x^8 + x^4 + x^3 + x + 1$
  - Für ungerades p ist p^n-1 gerade, aber 2^n-1 kann eine Primzahl sein

  $\mapsto$ **Konstruktion zyklischer Gruppen mit Primzahlordnung möglich**

Beispiel: Konstruktion von $GF(2^3)$

- **Es gibt 8 Polynome $f(x)=x^3+a_2x^2+a_1x+a_0$**
  - Polynome mit $a_0=0$ sind teilbar durch $g(x) = x$
  - $f_1(x) = x^3+1$ ist reduzibel, da $f_1(x) = (x+1)(x^2+x+1)$
    $f_2(x) = x^3+x^2+1$ und $f_3(x) = x^3+x+1$ sind irreduzibel ($f_2 \equiv f_3 \equiv 1$)
    $f_4(x) = x^3+x^2+x+1$ ist reduzibel, da $f_4(x) = (x+1)(x^2+1)$
  - f_2 und f_3 sind geeignet als Basis für die Konstruktion
- **Operationen auf Koeffizienten reichen aus**
  - $GF(2^3)$ enthält nur Polynome $g(x) = a_2x^2+a^1x+a_0$ mit $a_i \in \mathbb{Z}_2$
  - Additions- und Multiplikation sind als Bitblockoperationen darstellbar
    z.B. $(x^2+1)\cdot(x^2+x+1) \bmod f_3 = x^4+x^3+x+1 \bmod f_3 = x^2+x$
    entspricht `101·111 mod 1011 = 11011 mod 1011 = 110`

·	001	010	011	100	101	110	111
001	001	010	011	100	101	110	111
010	010	100	110	011	001	111	101
011	011	110	101	111	100	001	010
100	100	011	111	110	010	101	001
101	101	001	100	010	111	011	110
110	110	111	001	101	011	010	100
111	111	101	010	001	110	100	011

als Zahl

·	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	3	1	7	5
3	3	6	5	7	4	1	2
4	4	3	7	6	2	5	1
5	5	1	4	2	7	3	6
6	6	7	1	5	3	2	4
7	7	5	2	1	6	4	3

  - **Liefert Körper mit Basis $\mathbb{Z}_8^*$ aber mit Nichtstandard-Multiplikation**

Weitere Komponenten des AES: ShiftRows

- **Permutation** ShiftRows: $\{0,1\}^{128} \to \{0,1\}^{128}$
  - Bytepermutation auf Byte-Vektoren der Länge 16 (Tabelle)

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}

  - Vektoren werden als 4x4 Matrix geschrieben

$$\begin{pmatrix} v_0 & v_4 & v_8 & v_{12} \\ v_1 & v_5 & v_9 & v_{13} \\ v_2 & v_6 & v_{10} & v_{14} \\ v_3 & v_7 & v_{11} & v_{15} \end{pmatrix} \longrightarrow \begin{pmatrix} v_0 & v_4 & v_8 & v_{12} \\ v_{13} & v_1 & v_5 & v_9 \\ v_{10} & v_{14} & v_2 & v_6 \\ v_7 & v_{11} & v_{15} & v_3 \end{pmatrix}$$

  - ShiftRows verschiebt Zeile i zyklisch um i Positionen nach links
  - 4x4 Matrix wird als 128bit Vektor interpretiert

v_0	v_{13}	v_{10}	v_7	v_4	v_1	v_{14}	v_{11}	v_8	v_5	v_2	v_{15}	v_{12}	v_9	v_6	v_3

Leicht als Hardwareschaltung realisierbar

Weitere Komponenten des AES: MixColumns

- MixColumns: **multipliziere Spalten in $GF(2^8)$**
  - Betrachte jedes Byte des 128bit Vektors als Polynom $v_i = \sum_{j=0}^{7} a_{i,j} x^j$

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}

  - Multiplikation der 4x4 Darstellung des Vektors mit fester Matrix

$$\begin{pmatrix} v'_0 & v'_4 & v'_8 & v'_{12} \\ v'_1 & v'_5 & v'_9 & v'_{13} \\ v'_2 & v'_6 & v'_{10} & v'_{14} \\ v'_3 & v'_7 & v'_{11} & v'_{15} \end{pmatrix} = \begin{pmatrix} x & x{+}1 & 1 & 1 \\ 1 & x & x{+}1 & 1 \\ 1 & 1 & x & x{+}1 \\ x{+}1 & 1 & 1 & x \end{pmatrix} \star \begin{pmatrix} v_0 & v_4 & v_8 & v_{12} \\ v_1 & v_5 & v_9 & v_{13} \\ v_2 & v_6 & v_{10} & v_{14} \\ v_3 & v_7 & v_{11} & v_{15} \end{pmatrix}$$

  - Lineare Abbildung, die starke Diffusion der Spalten erzeugt

    z.B. $v'_0 = x{\cdot}v_0 + (x{+}1){\cdot}v_1 + v_2 + v_3 \bmod x^8{+}x^4{+}x^3{+}x{+}1$

  - Berechnung ist effizient in Binärdarstellung durchführbar

    z.B. $x{\cdot}(a_7x^7{+}a_6x^6{+}a_5x^5{+}a_4x^4{+}a_3x^3{+}a_2x^2{+}a_1x{+}a_0) \bmod x^8{+}x^4{+}x^3{+}x{+}1$

    $\hat{=}\ 00000010{\cdot}a_7a_6a_5a_4a_3a_2a_1a_0 \bmod 100011011$

    $= a_7a_6a_5a_4a_3a_2a_1a_00 \bmod 100011011 = \begin{cases} a_6a_5a_4a_3a_2a_1a_00 & \text{falls } a_7{=}0 \\ a_6a_5a_4\bar{a}_3\bar{a}_2a_1\bar{a}_01 & \text{falls } a_7{=}1 \end{cases}$

Eigenschaften des AES

- **Einfache Entschlüsselung des Chiffretextes**
  - Rückwärtsanwendung des gesamten Verfahrens
  - Jede einzelne Operation ist invertierbar
  - Rundenschlüssel müssen in umgekehrter Reihenfolge eingesetzt werden
- **Schnelle Verarbeitung großer Datenmengen**
  - Einfache algorithmische Struktur, sehr effiziente Ausführung möglich
  - Alle Bestandteile sind als Hardwarekomponenten realisierbar
- **Sicherheitsaspekte**
  - Verfahren liegt vollständig offen und wurde von Fachwelt inspiziert
  - 128 Bit Schlüssellänge verhindert Brute-Force Attacken
  - Algebraische Konstruktion der Tabellen und lineare Transformation erzeugen nahezu uniforme Verteilung der Schlüsseltexte
    Lineare und differentielle Analyse ist praktisch unmöglich

Zur Zeit sicher gegen alle bekannten Attacken

Iterative Blockchiffren: Stärken & Schwächen

- **Blockchiffren sind erfolgreich in der Praxis**
  - Bei großen Schlüsseln sicher gegen alle bekannten Attacken
    - Große S-Boxen erzeugen starke Nichtlinearität
    - Lineare Transformationen erzeugen hohe Diffusion
    - Große Block- und Schlüsselgröße verhindert statistische Analysen
  - Effizient durch Verwendung von $\oplus$ und einfachen Operationen
    - Komponenten benötigen nur Tabellensuche oder ähnliches
    - Großer Datendurchsatz möglich
- **Es gibt keine theoretischen Sicherheitsgarantien**
  - Sicherheit basiert auf Erfahrung, nicht auf mathematischen Beweisen
- **Verfahren sind symmetrisch**
  - Schlüssel müssen zuvor über sichere Kanäle ausgetauscht werden
  - Aufbau einer spontanen sicheren Verbindung nicht möglich
  - **Praktische Verwendung erfordert Kombination mit Verfahren der Public-Key Kryptographie**