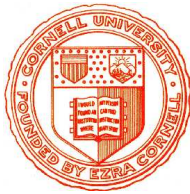


Kryptographie und Komplexität



Einheit 3

Praktisch sichere Blockchiffren



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

- **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

● 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: $S_1 \times S_2$ für $S_i = (\mathcal{P}_i, \mathcal{C}_i, \mathcal{K}_i, e, d)$

● Iteration ist die häufigste Form

- $S^n = S \times S \times \dots \times S$ (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

- **Wiederholte Verschlüsselung mit Rundenfunktion**
 - Schlüssel K wird in n Teilschlüssel K^1, \dots, 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

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 : $v_{(i)}^r := \sigma(u_{(i)}^r)$
- Permutation aller Bits des entstehenden Blocks: $w_j^r := u_{\pi(j)}^r$
- σ ist bijektive Substitution auf $\{0, 1\}^l$, π Permutation auf $\{1..l \cdot m\}$
- Länge der Blöcke (und Teilschlüssel) ist $l \cdot m$

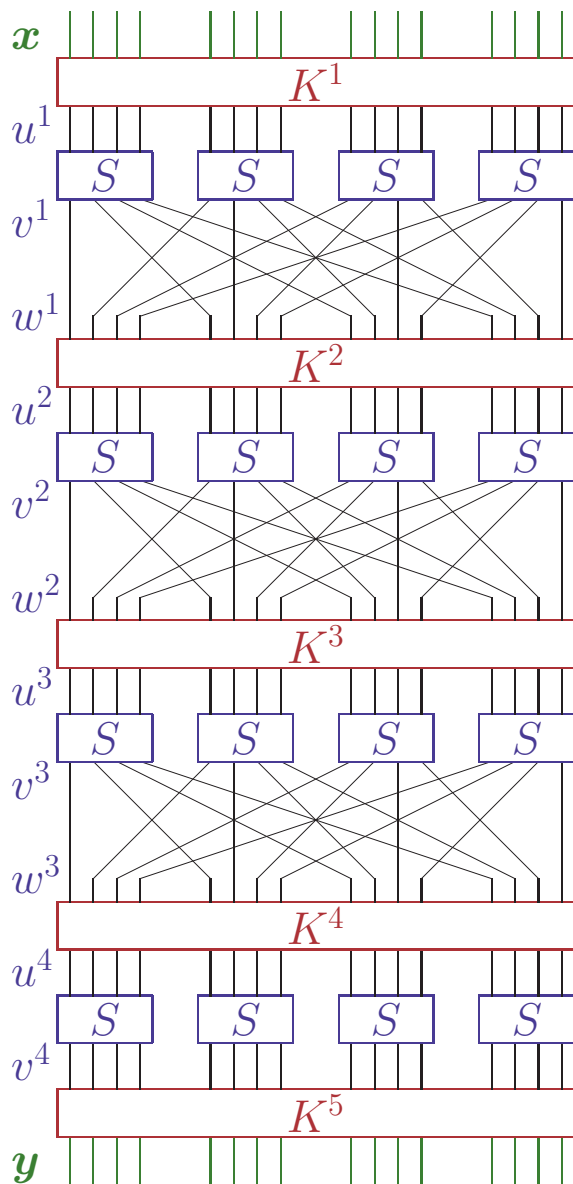
● Letzte Runde ist verändert

- Binäre Addition des Teilschlüssels K^n : $u^n := w^{n-1} \oplus K^n$
- Substitution der Unterblöcke $u_{(i)}^n$: $v_{(i)}^n := \sigma(u_{(i)}^n)$
- 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



$K = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$

$\sigma =$

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.)

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

$x = w^0 = 0010\ 0110\ 1011\ 0111$

$K^1 = 0011\ 1010\ 1001\ 0100$

$u^1 = 0001\ 1100\ 0010\ 0011$

$v^1 = 0100\ 0101\ 1101\ 0001$

$w^1 = 0010\ 1110\ 0000\ 0111$

$K^2 = 1010\ 1001\ 0100\ 1101$

$u^2 = 1000\ 0111\ 0100\ 1010$

$v^2 = 0011\ 1000\ 0010\ 0110$

$w^2 = 0100\ 0001\ 1011\ 1000$

$K^3 = 1001\ 0100\ 1101\ 0110$

$u^3 = 1101\ 0101\ 0110\ 1110$

$v^3 = 1001\ 1111\ 1011\ 0000$

$w^3 = 1110\ 0100\ 0110\ 1110$

$K^4 = 0100\ 1101\ 0110\ 0011$

$u^4 = 1010\ 1001\ 0000\ 1101$

$v^4 = 0110\ 1010\ 1110\ 1001$

$K^5 = 1101\ 0110\ 0011\ 1111$

$y = 1011\ 1100\ 1001\ 0110$

SICHERHEIT VON SP-NETZWERKEN

- **Alles außer dem Schlüssel liegt offen**
 - Key-Scheduling Verfahren zur Erzeugung der Teilschlüssel K^1, \dots, 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 möglich bei zu kleinen Blocklängen (32 Bit), geringer Anzahl von Runden oder kleinen 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
- **Variationen erhöhen Diffusion und Konfusion**
 - Verwendung großer, verschiedenartiger S-Boxen ↪ DES
 - Verwendung zusätzlicher linearer Transformationen ↪ AES

- **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

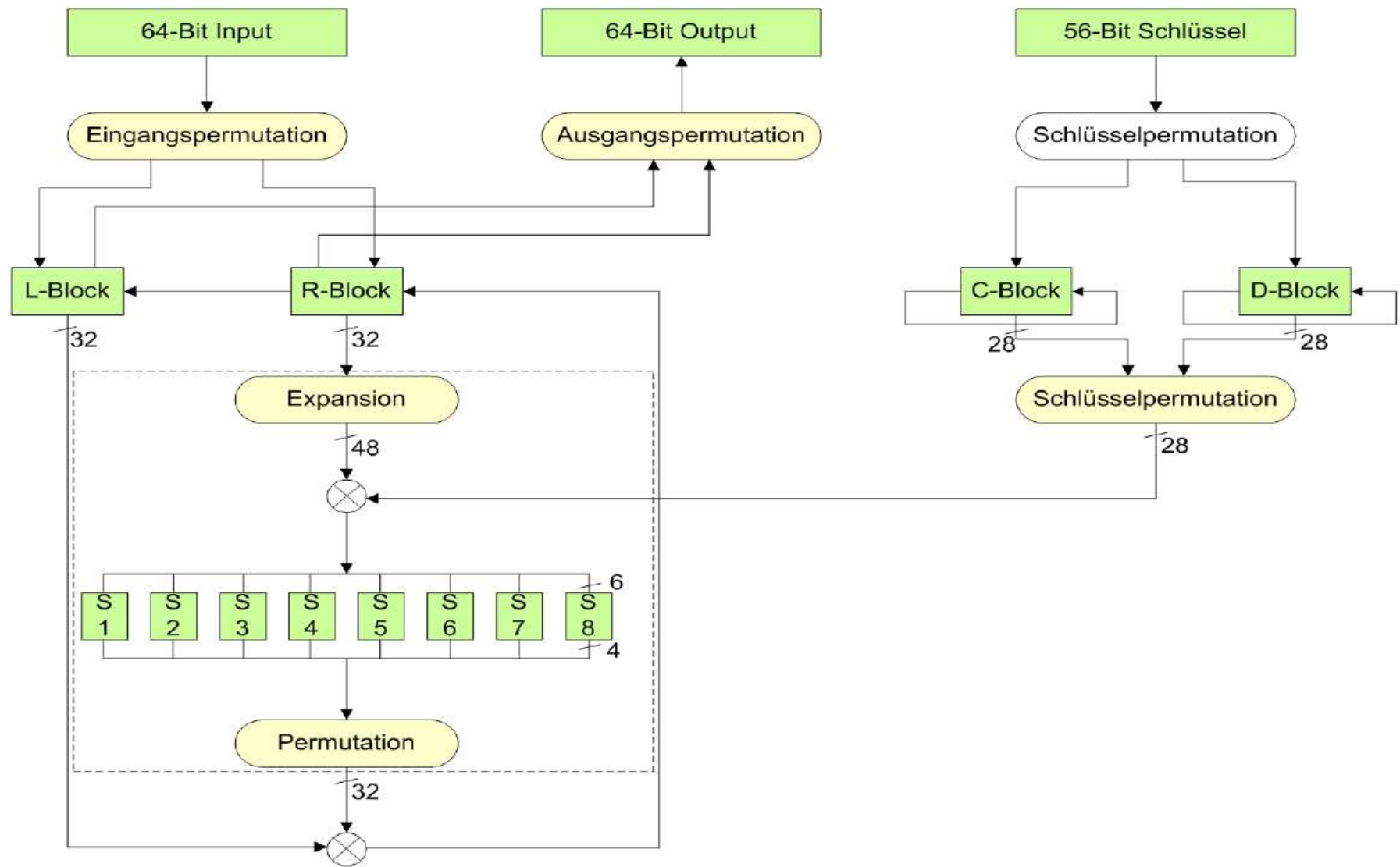
- **Einfache Entschlüsselung**

- 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:** $\{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$

- 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:** $\{0, 1\}^{32} \rightarrow \{0, 1\}^{48}$

- Beschrieben durch festgewählte Expansionstabelle mit 48 Einträgen

- **Acht S-Boxen S_i :** $\{0, 1\}^6 \rightarrow \{0, 1\}^4$

- Je 4 festgewählte Permutationstabellen auf $\{0, 1\}^4$

- Erstes und letztes Bit eines 6er-Blocks bestimmt Auswahl der Tabelle

- **Permutation P:** $\{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$

- 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 Teilttextblöcken
- 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
- 56 Bit Schlüssellänge erlaubt Brute-Force Attacken mit Spezialhardware
1999: 100000 PC's benötigen nur 22 Stunden (DES Challenge III)
- 1994: Lineare Kryptoanalyse mit 2^{43} Klartext-/Schlüsseltextpaaren
benötigt 10 Tage mit 12 Workstations, um Schlüssel zu brechen

Einfacher DES ist heute nicht mehr sicher genug

- **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_2 := 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
 - Zyklischen 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:** $\{0, 1\}^8 \rightarrow \{0, 1\}^8$

- Bytes werden mit Elementen des endlichen Körpers $GF(2^8)$ identifiziert

- $b = (b_7, \dots, b_0)$ 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**)

WEITERE KOMPONENTEN DES AES

- **Permutation ShiftRows:** $\{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$

- Bytepermutation auf Byte-Vektoren der Länge 16 (Tabelle)

- Vektoren werden als Matrix $\begin{pmatrix} v_0 & v_4 & v_8 & v_{12} \\ v_1 & v_5 & v_9 & v_{13} \\ v_2 & v_6 & v_{10} & v_{14} \\ v_3 & v_7 & v_{11} & v_{15} \end{pmatrix}$ geschrieben

- **ShiftRows** verschiebt Zeile i zyklisch um i Positionen nach links

- **MixColumns: multipliziere Spalten in $GF(2^8)$**

- **MixColumns** multipliziert Spalten mit $\begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix}$

wobei jedes Byte einer Spalte als Polynom betrachtet wird

- Lineare Abbildung, die **starke Diffusion** der Spalten erzeugt

- **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**

Approximiere Chiffrierung durch lineare Abbildung

- **SPNs und DES haben wenig nichtlineare Anteile**
 - Ein Ausgabebit y_j könnte ‘wahrscheinliche Linearkombination’ der Eingabebits x_1, \dots, x_m sein
 - D.h. die Wahrscheinlichkeit, daß $\sum_{k=1}^m a_k x_k$ und y_j für bestimmte a_1, \dots, a_m den gleichen Wert annehmen, ist nicht $1/2$ ($\hat{=}$ völliger Zufall)
 - Untersuche Wahrscheinlichkeit des Ereignisses $\sum_{k=1}^m a_k x_k \oplus y_j = 0$
- **S-Boxen sind einzige nichtlineare Komponenten**
 - Bestimme lineare Approximation der S-Boxen und der Rundenfunktion
 - Approximiere Chiffrierfunktion bis zur Addition von K^n in Runde n
- **Known plaintext 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}

- **Betrachte S-Box $S : \{0, 1\}^m \rightarrow \{0, 1\}^n$**
 - Eingaben x_1, \dots, x_m sind gleichverteilt
 - Wenn $S(x_1, \dots, x_m) = (y_1, \dots, y_n)$, dann $Pr(x_1, \dots, x_m, y_1, \dots, y_n) = 2^{-m}$
 - Wenn $S(x_1, \dots, x_m) \neq (y_1, \dots, y_n)$, dann $Pr(x_1, \dots, x_m, y_1, \dots, y_n) = 0$
- **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
- 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

● Diskrete Zufallsvariable

- Variable X über endlicher Menge S mit Wahrscheinlichkeitsverteilung
- Wahrscheinlichkeit $Pr(E)$ ist genau gesehen die Wahrscheinlichkeit, daß X einen Wert aus E annimmt (also $Pr[X = x] := Pr(x)$)
- Liefert präzisere 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, \dots, X_k unabhängige Variablen mit Bias ϵ_i und $\epsilon_{1,\dots,k}$ Bias von $X_1 \oplus \dots \oplus X_k$, dann ist $\epsilon_{1,\dots,k} = 2^{k-1} \prod_{i=1}^k \epsilon_i$

- Abhängigkeiten in einem Netzwerk pflanzen sich multiplikativ fort

MATHEMATIK: BEWEIS DES PILING UP LEMMAS

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

● Beweis durch Induktion über k

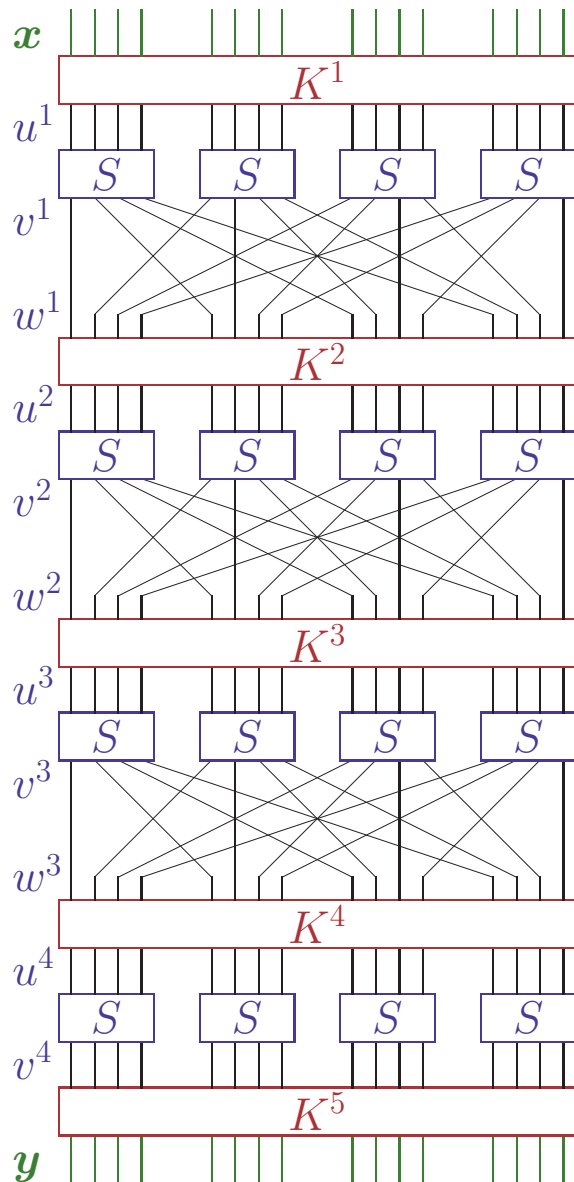
- Aussage ist trivialerweise wahr für $k=1$
- Wir nehmen an $\epsilon_{1,\dots,k} = 2^{k-1} \prod_{i=1}^k \epsilon_i$ für ein $k \geq 1$
- Dann gilt

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

● Korollar: Perfekter Zufall pflanzt sich fort

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

LINEARE ANALYSE EINES SP-NETZWERKES



● Bestimme größte Biaswerte

– In $S_{(2)}^1$: $T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1$ hat Bias $\frac{1}{4}$

V_6^1 wird permutiert zu $U_6^2 = V_6^1 \oplus K_6^2$

– In $S_{(2)}^2$: $T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2$ hat Bias $-\frac{1}{4}$

Permutation: $U_6^3 = V_6^2 \oplus K_6^3$, $U_{14}^3 = V_8^2 \oplus K_{14}^3$

– In $S_{(2)}^3$: $T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3$ hat Bias $-\frac{1}{4}$

– In $S_{(4)}^3$: $T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3$ hat Bias $-\frac{1}{4}$

● Pfad $T_1 \oplus T_2 \oplus T_3 \oplus T_4$ hat Bias $-\frac{1}{32}$

– Setze ein: $U_j^1 = X_j$, $V_6^3 = U_6^4 \oplus K_6^4, \dots$

$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \oplus$

$K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3 \oplus K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$

● Schlüsselbits sind fest

– $X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4$

hat Bias $-\frac{1}{32}$ (Schlüsselbits 0) oder $\frac{1}{32}$

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

- **Abhängigkeit $x_{(2)}/u_{(2,4)}^4$ liefert 8 Schlüsselbits**

- Überprüfe $K_5^5, K_6^5, K_7^5, K_8^5, K_{13}^5, K_{14}^5, K_{15}^5, K_{16}^5$ (256 Kandidaten)

- **Known plaintext Attacke**

- Analysiere alle Kandidaten $K_{(2,4)}^5$
- Für Klar-/Schlüsseltextpaare (x, y) berechne $u_{(2,4)}^4 = S^{-1}(y_{(2,4)} \oplus K_{(2,4)}^5)$
- Berechne, wie oft $x_5 \oplus x_7 \oplus x_8 \oplus u_6^4 \oplus u_8^4 \oplus u_{14}^4 \oplus u_{16}^4$ 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

Analysiere Differenzen zwischen Klartexten

● Chosen plaintext Attacke

- Wähle Klartextpaare x_1, x_2 mit fester Differenz $x' = x_1 \oplus x_2$
- Bestimme Differenz $y' = y_1 \oplus y_2$ 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 $u^{n'}$
- Extrahiere Schlüsselbits wie bei linearer Analyse

- **Iterative Blockchiffren sind sehr erfolgreich**
 - 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
- **Verfahren sind symmetrisch**
 - Schlüssel müssen zuvor über sichere Kanäle ausgetauscht werden
 - Aufbau einer spontanen sicheren Verbindung nicht möglich
 - Praktischer Verwendung erfordert Kombination mit Verfahren der Public-Key Kryptographie