Symmetrische Blockchiffren

Inhaltsverzeichnis

Kapitel 1

Einführung

Von einer symmetrischen Blockchiffre spricht man, wenn sie

1. auf *Blöcken* von Nachrichten operiert und
2. die zur Ver- und Entschlüsselung verwendeten Schlüssel *symmetrisch*, normalerweise identisch sind.

Dabei beachte man, daß die Blöcke unabhängig voneinander verschlüsselt werden![1] Wenn eine Verkettung von Blöcken gewünscht wird kann die Blockchiffre in Verbindung mit einer geeigneten Betriebsart[2] verwendet werden.

In diesem Versuch wird häufig die symmetrische Blockchiffre DES[3] als Beispiel einer konkreten Chiffre verwendet. Der DES gilt wegen seiner geringen Schlüssellänge mittlerweile als veraltet bzw. unsicher, allerdings lassen sich an dieser Chiffre die Prinzipien moderner symmetrischer Blockchiffen sehr gut zeigen, so daß sie für diesen Versuch trotzdem als Beispiel ausgewählt wurde.

Im folgenden sollen die Merkmale von symmetrischen Blockchiffren zusammengetragen werden sowie Möglichkeiten zu ihrer Realisierung. Es wird ein Überblick über die Eigenschaften der Funktionen, die zu ihrer Realisierung führen gegeben.

1.1 Definition einer symmetrischen Blockchiffre

Definition 1 Wählt man ein *Alphabet* A, dem die Klartext- und Chiffretextzeichen angehören und ein Alphabet B aus dem die Zeichen des Schlüssels gebildet werden, so kann man definieren:

$$\begin{array}{rcll} \mathcal{M} & \in & A^n & \text{Klartextblock} \\ \mathcal{C} & \in & A^m & \text{Chiffretextblock} \\ K & \in & B^l & \text{Schlüssel} \end{array}$$

$$\mathcal{C} = \mathbf{E}(K, \mathcal{M}) \quad \text{Verschlüsselungsfunktion } \mathbf{E} \tag{1.1}$$

$$\mathcal{M} = \mathbf{D}(K, \mathcal{C}) \quad \text{Entschlüsselungsfunktion } \mathbf{D} \tag{1.2}$$

Dabei gelte
$n, m, l \in \mathbb{N}$ und $m \geq n$.

[1] vgl. [FW88,S.175]
[2] siehe [FW88,HP85,FH93]
[3] siehe `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

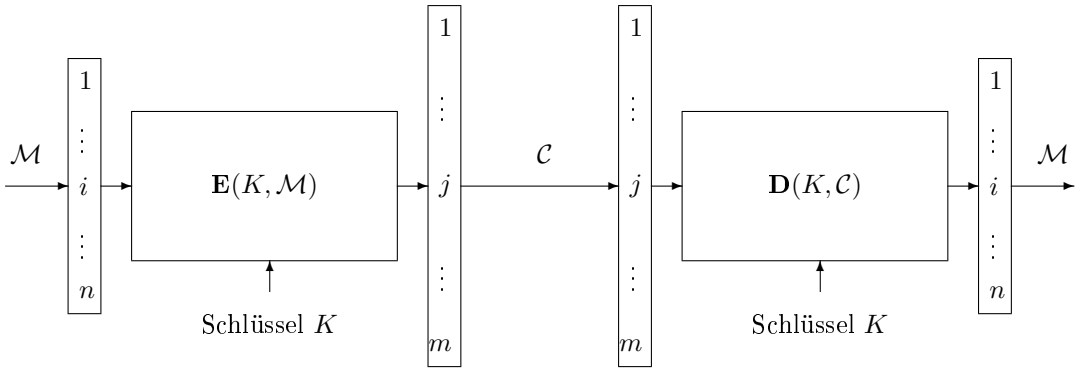


Abbildung 1.1: Symmetrische Blockchiffre

$\frac{m}{n}$ definiert den *Expansionsfaktor* der Blockchiffre.
$\frac{m}{n} = 1$ definiert eine *längentreue* Blockchiffre.
k_A und k_B seien die Mächtigkeiten der Alphabete A und B. ‡

Für die Klasse der *Feistel-Chiffren* (siehe Abschnitt 1.3) gelte einschränkend, daß n, m, l geradzahlig sind sowie $m = n$.

Eine Blockchiffre substituiert einen Klartextblock der Größe n durch einen Chiffretextblock der Größe m in Anhängigkeit vom Schlüssel K.

Für jeden Schlüssel muß eine *rechtseindeutige* Abbildung $\mathbf{E}$ mit

$$\mathbf{E} : K \times \mathcal{M} \to \mathcal{C}$$

existieren. Für bestimmte Anwendungen genügt das bereits. Man denke beispielsweise an die Paßwortverschlüsselung. Zur Passwortkontrolle werden die Passwörter nicht zurück in den Klartext entschlüsselt, sondern lediglich die Chiffretexte der Passwörter verglichen. Der Vergleich findet also im sog. Bildbereich von $\mathbf{E}$ statt. Eine Möglichkeit zur Entschlüsselung des Passwortes ist also gar nicht erforderlich, meist sogar unerwünscht, d.h. die Umkehrabbildung $\mathbf{D}$ mit

$$\mathbf{D} : K \times \mathcal{C} \to \mathcal{M}$$

muß nicht notwendigerweise rechtseindeutig sein für jeden Schlüssel!

Im folgenden soll aber in jedem Fall die *Injektivität (Eindeutigkeit, Existenz der Umkehrabbildung)* von $\mathbf{E}$ und $\mathbf{D}$ gefordert werden, damit aus dem Chiffrat der Klartext wiedergewonnen werden kann.

Damit sollen die hier behandelten Blockchiffren als eine Familie *bijektiver (total, injektiv, surjektiv)* Abbildungen (Substitutionen) k_A^n verschiedener Klartextblöcke in k_A^m Chiffretextblöcke verstanden werden[4].

Ist die Blockchiffre längentreu ($n = m$), kann man einen Chiffretextblock als Permutation des Klartextblockes auffassen. Es existieren dann genau $k_A^n!$ Permutationen auf $\mathcal{M}$.

Abbildung 1.1 zeigt den grundsätzlichen Aufbau einer symmetrischen Blockchiffre.

[4]Anhang A gibt die Bedeutungen der verwendeten Begriffe *rechtseindeutig, total, injektiv, surjektiv, bijektiv* vollständigkeitshalber an.

1.2 Realisierung einer Blockchiffre

Ziel des Designs einer Blockchiffre muß es nun sein, eine möglichst komplexe Abbildung zu finden, die bei Kenntnis von Klartext-/Chiffretextpaaren $\mathcal{M}/\mathcal{C}$ keine Rückschlüsse auf K zuläßt. Diese Forderung legt die Realisierung einer solchen Substitution in Form einer Tabelle nahe.

Für Alphabete A und B der Mächtigkeit k_A und k_B bedeutet das eine Tabelle von k_A^n Zeilen und k_B^l Spalten mit m Zeichen je Eintrag, also

$$MR = k_A^n \cdot k_B^l \cdot m \qquad \text{nach Formel (1.1)} \tag{1.3}$$

Zeichen des Alphabets A.

Dieser enorme Speicherbedarf läßt sich reduzieren wenn $\mathbf{E}$ in der Form

$$\mathbf{E}(K, \mathcal{M}) = \mathbf{S}(K \circ M) \tag{1.4}$$

aufgebaut ist. Dabei sei $A = B$ und $l = n$ vorausgesetzt. Die Verknüpfung $\circ$ stellt beispielsweise eine Addition modulo 2, d. h. XOR, dar. Der Speicherplatzbedarf einer solchen Tabelle beträgt dann

$$MR = k_A^n \cdot m \qquad \text{nach Formel (1.4)} \tag{1.5}$$

Zeichen des Alphabets A.

Das folgende Beispiel soll den immensen Speicherbedarf solcher Substitutionen verdeutlichen.

Beispiel 1 Es sei $A = B = \{0, 1\}$ so ist $k_A = k_B = 2$. Es gelte $m = n = l$.

Blockbreite	Speicherbedarf MR der Tabelle	
m,n,l	nach (1.3)	nach (1.5)
8 Bit	512 kBit	2 kBit
16 Bit	64 GBit	1 MBit
32 Bit		128 GBit

BSP

Da die Implementation solcher Substitutionen für realistische Blockbreiten von z.B. 64 oder 128 Bit nicht möglich ist, geht man folgenden Weg:
Es werden Teile des Inputblockes substituiert und anschließend permutiert. Dieser Prozeß wird iterativ durchgeführt, so daß eine *Produktchiffre* aus Substitutionen und Permutationen entsteht.

Abbildung 1.2 zeigt eine solche Produktchiffre.

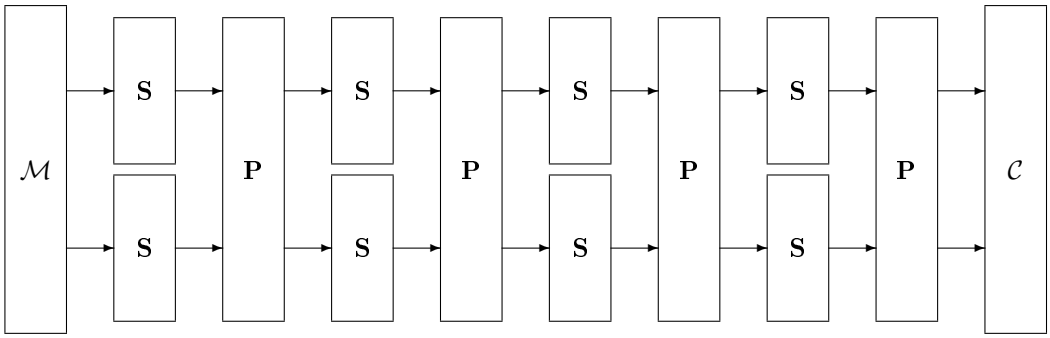
Im folgenden wird davon ausgegangen, daß die Alphabete $A = B = \{0, 1\}$ vorliegen. Die Zeichen der Blockchiffre sind also Bits.

Kryptographisch „gute“ Blockchiffren fordern also Substitutionen und Permutationen mit bestimmten Eigenschaften. Diese werden in den Abschnitten 1.3.1 bis 1.3.3 behandelt.

Zunächst soll aber die Klasse der Feistel-Chiffren vorgestellt werden.

1.3 Feistel-Chiffren

Feistel-Chiffren – nach FEISTEL – liegt die Idee zugrunde, die gleiche Funktion zur Ver- und Entschlüsselung eines Blockes zu verwenden. Sie besitzen einen Expansionsfaktor von 1, sind also längentreu. Die Klartext- und Chiffretextblöcke bestehen aus einer geradzahligen Anzahl von binären Zeichen (Bits). Ein Block $\mathcal{M}$ bzw. $\mathcal{C}$ wird in zwei gleichgroße Teile L und R zerlegt.



S ... Substitution **P** ... Permutation

Abbildung 1.2: Beispiel einer Produktchiffre aus Substitutionen und Permutationen

Definition 2 Die Verschlüsselungsfunktion einer Feistel-Chiffre lautet:

$$\begin{array}{rcll} (L_0, R_0) & := & \mathcal{M} & \\ (L_i, R_i) & := & (R_{i-1}, F(K_i\,,\, R_{i-1}) \oplus L_{i-1}) & 1 \leq i \leq N \\ \mathbf{E}(K, \mathcal{M}) & := & (L_N, R_N). & \end{array} \tag{1.6}$$

Analog ist die Entschlüsselungsfunktion definiert mit

$$\begin{array}{rcll} (L_0, R_0) & := & \mathcal{C} & \\ (L_i, R_i) & := & (F(K_{N-i+1}, L_{i-1}) \oplus R_{i-1}\,,\, L_{i-1}) & 1 \leq i \leq N \\ \mathbf{D}(K, \mathcal{C}) & := & (L_N, R_N). & \end{array} \tag{1.7}$$

N bezeichne die Anzahl durchgeführter Iterationen, die sog. *Rundenzahl*.
K_i sind *Teilschlüssel*, die aus K gewonnen werden müssen. ‡

Abbildung 1.3 zeigt diese iterierte Anwendung grafisch.

Das Schema der Feistel-Chiffre garantiert automatisch die Bijektivität der Verschlüsselungsfunktion **E** bei Kenntnis des Schlüssels K.
Das Verfahren selbst ist invers bis auf die Vertauschung der Teilblöcke und Umkehrung der Teilschlüssel!

Der kryptographisch interessante Teil ist die Funktion $F(K_i, R_{i-1})$. Da sie die kryptographische Güte der gesammten Blockchiffre bestimmt.

Damit kann man das Design einer kryptographisch guten Blochchiffre genau auf das Design der Funktion F reduzieren. Daß diese Reduktion jedoch keine Vereinfachung, sondern eher eine Verschärfung der Forderungen an F darstellt, liegt auf der Hand: iterierte Anwendung einer Funktion legt bestimmte zyklische Eigenschaften dieser Funktion offen. Allerdings bietet die iterative Anwendung einer *nicht iterativen Blockchiffre* selbstverständlich die gleichen Ansätze zur Erforschung von Zyklen. Nur muß man annehmen, daß die Zyklenlänge iterativer Blockchiffren um den Faktor N kleiner ist.

Bei Ver- und Entschlüsselung wird nur die Funktion F verwendet, nicht aber ihre Inverse F^{-1}. Damit liegt die Verwendung sog. *Einwegfunktionen* für F nahe. Einwegfunktionen zeichnen sich durch die Eigenschaft aus, daß es leicht möglich ist, ein $y = F(x)$ zu berechnen, jedoch ein $x = G(y)$ gar nicht oder nur mit erheblich höherem Aufwand. $G = F^{-1}$ stellt die Umkehrfunktion zu F dar.

Wenn eine Blockchiffre — interpretiert als Produktchiffre — aus Substitutionen und Permutationen bestehen soll, so müssen genau diese durch F realisiert werden.

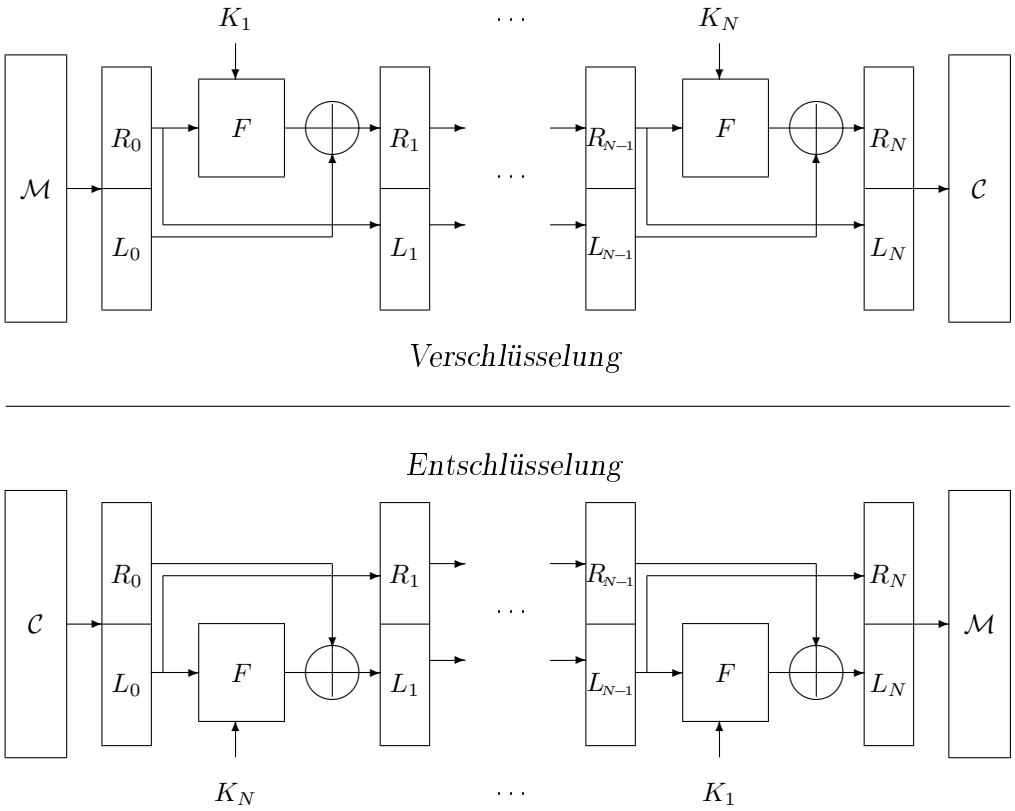


Abbildung 1.3: Feistel-Chiffre

Es sollen nun die Anforderungen an solche Substitutionen und Permutationen beschrieben werden. Ein weiterer Abschnitt beschreibt Anforderungen an die Generierung der Teilschlüssel K_i, die in den Runden $1 \ldots N$ verwendet werden.

1.3.1 Anforderungen an Substitutionen

Von GORDON und RETKIN stammen folgende Forderungen für „gute“ Substitutionen:[5]

(v) Jedes Outputbit sollte von jedem Inputbit echt abhängig sein. (Vollständigkeit)

(a) Die Änderung eines Inputbits soll im Mittel die Hälfte der Outputbits ändern. (Avalanche)

(l) Kein Outputbit sollte linear von einem Inputbit abhängen. (Nichtlinearität)

(k) Solange nichts über Inputbits bekannt ist, sollte man keine Informationen über Outputbits besitzen und umgekehrt. (Korrelationsimmunität)

Die hier aufgeführten Eigenschaften für Substitutionen beziehen sich nicht nur auf F einer Feistel-Chiffre sondern gelten ebenso allgemein für eine „gute“ Blockchiffre.

Abschnitt 2.1 geht auf diese Forderungen näher ein.

[5] nach [FW88,S.180]

1.3.2 Anforderungen an Permutationen und Expansionen

(p) Jedes Outputbit der Substitution soll – von Runde zu Runde – möglichst viele Inputbits der nachfolgenden Substitution beeinflussen.

Beispiel 2 Beim DES beeinflussen die 4 Outputbits jeder S-Box in der nächsten Runde 6 von 8 S-Boxen! BSP

Das kann man mit Hilfe geschickter Expansionsabbildungen erreichen. Expansionsabbildungen haben außerdem den Effekt, daß F bei festem Schlüssel nicht bijektiv ist[6].

1.3.3 Teilschlüsselerzeugung

Der Begriff *Teilschlüssel* wurde in Definition 2 eingeführt. Folgende Anforderungen werden an die Teilschlüsselerzeugung gestellt:

(t1) Jedes Bit des externen Schlüssels soll etwa gleichhäufig in den Teilschlüsseln verwendet werden.

(t2) Ein in einem Teilschlüssel K_i nicht verwendetes Bits wird mit hoher Wahrscheinlichkeit im Teilschlüssel K_{i+1} verwendet.

(t3) Kein Bit des externen Schlüssels steht mehr als einmal an einer bestimmten Stelle des Teilschlüssels.

(t4) Alle Teilschlüssel sollen möglichst verschieden voneinander sein

(t5) Geringfügige Änderung des externen Schlüssels soll zu möglichst großen Änderungen der Teilschlüssel führen.

Auch hier kommen Permutationen und Substitutionen zur Teilschlüsselerzeugung in Frage. Für jeden zu generierenden Teilschlüssel muß dann eine andere Permutation verwendet werden.

Als implementierungsfreundliches Mittel zur Teilschlüsselgenerierung dient in vielen Fällen ein zyklisches Rotieren der Bits im Schlüsselvektor mit nachfolgender Permutierung. Weiterhin bieten sich zur Teilschlüsselgenerierung parametrisierte Pseudozufallszahlengeneratoren, Einwegfunktionen u.a. an.

[6] [FW88,S.235]

Kapitel 2

Kryptographische Güte einer Funktion F

Wie bereits angedeutet, soll die kryptographisch entscheidende Funktion F bestimmten Anforderungen genügen (vgl. Abschnitte 1.3.1 bis 1.3.3).
Wie kann man nun beurteilen, ob eine Funktion F diese Eigenschaften aufweist?

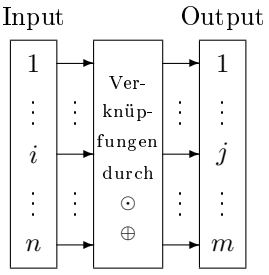
Die folgenden Abschnitte präzisieren die Bewertungskriterien für F und zeigen Möglichkeiten zur Beurteilung der kryptographischen Güte von F.

Es muß an dieser Stelle noch einmal ausdrücklich erwähnt werden, daß die vorgestellten Kriterien ebenso für die gesamte Blockchiffre wie auch für die kryptographisch relevante Funktion F einer Feistel-Chiffre bedeutsam sind.

2.1 Bewertungskriterien

2.1.1 Vollständigkeit

Definition 3 Eine Funktion $F : \{0,1\}^n \to \{0,1\}^m$ ist dann vollständig, wenn jedes Bit des Outputs von jedem Bit des Inputs abhängt.
Hängen die Output-Bits im Mittel von k Input-Bits ab, so bezeichnet $\frac{k}{n}$ den *Grad der Vollständigkeit* von F. ‡



Für alle j und für alle i existiert also mindestens eine Belegung des Inputs mit der Eigenschaft, daß eine Änderung des Input-Bits i zu einer Änderung des Output-Bits j führt, wenn F vollständig ist.
Daß Vollständigkeit kein hinreichendes Kriterium für eine gute Substitution ist, wird am folgenden Beispiel gezeigt [a].

[a] aus [FW88,S.191]

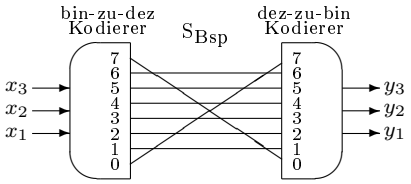
Beispiel 3 Gegeben sei eine S-Box S_{Bsp} mit $n = m = 3$, den Input-Bits x_i und Output-Bits y_j $(i, j = 1, 2, 3)$ die durch

$$\begin{aligned} y_1 &= x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_2 \oplus x_3 \oplus 1 \\ y_2 &= x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1 \oplus x_3 \oplus 1 \\ y_3 &= x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1 \oplus x_2 \oplus 1 \end{aligned} \tag{2.1}$$

beschrieben ist. In jeder Gleichung kommen alle x_i vor und können nicht eliminiert werden.

Die so beschriebene S-Box ist also vollständig. Verwendet man als Darstellungsmittel für eine solche S-Box eine Tabelle, so fällt auf, daß 6 von 8 möglichen Belegungen des Inputs identisch ausgegeben werden.

Input		Output	
dez	bin	bin	dez
X	$x_3x_2x_1$	$y_3y_2y_1$	Y
0	000	111	7
1	001	001	1
2	010	010	2
3	011	011	3
4	100	100	4
5	101	101	5
6	110	110	6
7	111	000	0

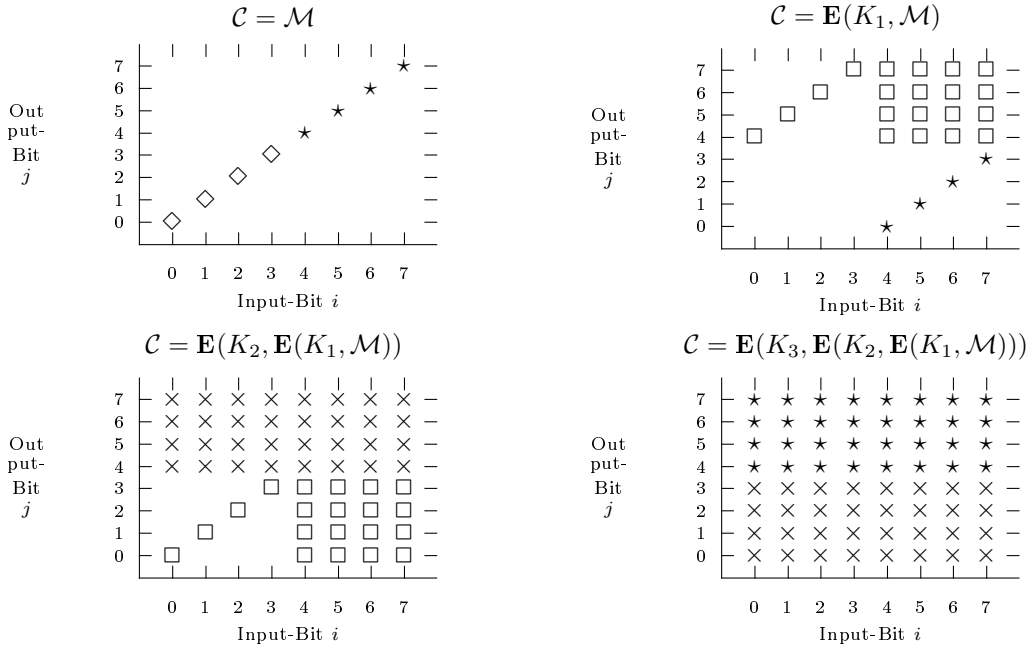


Damit ist diese vollständige Substitution nicht geeignet für eine sichere Blockchiffre. BSP

Wieviele Runden einer Feistel-Chiffre sind nun notwendig, bis der Output-Vektor der Chiffre vollständig vom Input-Vektor abhängt? Dies hängt zunächst vom Grad der Vollständigkeit der Funktionen F_i $(i = 1 \dots N)$ ab.

Es sind jedoch mindestens 3 Runden der Feistel-Chiffre nötig, falls die F_i alle vollständig sind.

Beispiel 4 Es sei $n = m = 8$. Die Feistel-Chiffre wird untersucht für $N = 1 \dots 3$ Runden. Die Funktionen F_i $(i = 1 \dots N)$ seien vollständig.
Bedeutungen:
Markierung: Output-Bit j hängt von Input-Bit i ab
Zeile eines Output-Bits vollständig markiert: Output-Bit hängt vollständig vom Input-Vektor ab.
Alle Positionen markiert: Blocksubstitution ist vollständig.



Es ist nachvollziebar, daß die Feistel-Chiffre nach 3 Runden vollständig ist, falls F_1 und F_2 und F_3 vollständig sind. BSP

Praktisch wird man aber mehr als 3 Runden benötigen, um die Vollständigkeit einer Blocksubstitution, basierend auf der Feistel-Chiffre zu garantieren.

Ob und wie von einer gegebenen Funktion F bzw. Blocksubstitution ihre Vollständigkeit entscheidbar ist wird — ebenso wie für die folgenden Kriterien — in Abschnitt 2.2 untersucht.

2.1.2 Avalanche

Definition 4 Eine Funktion $F : \{0,1\}^n \to \{0,1\}^m$ besitzt dann den Avalanche-Effekt[1], wenn die Änderung *eines* Input-Bits im Mittel die Hälfte aller Output-Bits ändert.
Wird durch Änderung *eines* Input-Bits *jedes* Output-Bit mit einer Wahrscheinlichkeit von 50 % verändert, so erfüllt F das strikte *Avalanche-Kriterium*.
Erfüllt F das strikte Avalanche-Kriterium, so ist F stets vollständig. ‡

Den Begriff „Avalanche-Effekt“ führte Feistel ein.
Praktisch bedeutet Avalanche-Effekt, daß „kleine“ Änderungen in $\mathcal{M}$ oder K zu „großen“ Änderungen in $\mathcal{C}$ führen. Er verringert damit die Erkennbarkeit von Äbhängigkeiten zwischen Klartextblöcken und Chiffretextblöcken oder Schlüsseln.

Wie das Vorhandensein von Avalanche einer Funktion F festgestellt werden kann, zeigt Anschnitt 2.2. Zunächst soll die durch das Gleichungssystem (2.1) aus Beispiel 3 gegebene Blocksubstitution den Avalanche-Effekt illustrieren!

Beispiel 5 (Weiterführung von Beispiel 3)
Es seien $\mathcal{M}$ und $\mathcal{M}$' zwei Input-Vektoren, die sich in genau 1 Bit unterscheiden. Die Tabelleneinträge zeigen die Anzahl der Bits, um die sich die Output-Vektoren $\mathcal{C} = \mathrm{S}_{\mathrm{Bsp}}(\mathcal{M})$ und $\mathcal{C}' = \mathrm{S}_{\mathrm{Bsp}}(\mathcal{M}')$ unterscheiden. Bei den freibleibenden Tabellenplätzen unterscheiden sich $\mathcal{M}$ und $\mathcal{M}$' nicht oder in mehr als genau 1 Bit. Diese Paare werden nicht untersucht.

$\mathcal{M}$, $\mathcal{M}$'	000	001	010	011	100	101	110	111	$\sum$
000		2	2		2				6
001	2			1		1			4
010	2			1			1		4
011		1	1					2	4
100	2					1	1		4
101		1			1			2	4
110			1		1			2	4
111				2		2	2		6
Gesamtzahl geänderter Bits									36

Insgesamt wurden damit 24 Fälle untersucht. Die Gesamtzahl geänderter Bits beträgt 36. Im Mittel ändern sich $36/24 = 1.5$ Bit. Bei einer Blockbreite von $n = m = 3$ sind das genau 50% geänderter Bits des Output-Vektors bei Änderung eines Bits des Input-Vektors.
Die S-Box $\mathrm{S}_{\mathrm{Bsp}}$ besitzt also den Avalanche-Effekt. Wie später in Abschnitt 2.2 gezeigt werden wird, erfüllt $\mathrm{S}_{\mathrm{Bsp}}$ sogar das strikte Avalanche-Kriterium. BSP

2.1.3 Linearität

Definition 5 Eine Funktion $F : \{0,1\}^n \to \{0,1\}^m$ ist dann linear, wenn *jedes* Output-Bit y_j linear von den Input-Bits x_i abhängt.
Wenn *wenigstens ein* Output-Bit linear von den Input-Bits abhängt bezeichnet man F als *partiell linear*. ‡

F kann als Gleichungssystem der Form

$$\begin{bmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & & & a_{1,n} \\ \vdots & \ddots & & & \vdots \\ a_{j,1} & \cdots & a_{j,i} & \cdots & a_{j,n} \\ \vdots & & & & \vdots \\ a_{m,1} & & \cdots & & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \oplus \begin{bmatrix} b_1 \\ \vdots \\ b_j \\ \vdots \\ b_m \end{bmatrix} \tag{2.2}$$

dargestellt werden mit $a_{j,i}$, $b_j \in \{0,1\}$.

[1] avalanche = Lawine

Oft wird eine solche Abbildung auch als affine Abbildung ($b_j \in \{0,1\}$) bezeichnet und eine Abbildung mit $b_j = 0$ als (homogen) linear. Hier soll diese Unterscheidung nicht vorgenommen werden.

Als weiteres Gütekriterium einer Funktion F kann man die *Güte ihrer besten linearen Approximation* G ermitteln. Der Anteil der Argumente, für den die Funktionswerte von F und G übereinstimmen bezeichnet man als *Güte der Approximation*. An dieser Stelle soll nicht auf das Verfahren der linearen Approximation eingegangen werden. Nähere Informationen hierzu finden sich in [Ru86].

Abschließend sollen noch einige Sätze, die jedoch ohne Beweis angegeben werden, die Nichtlinearität einer S-Box als Gütekriterium illustrieren[2]:

Satz 1 Jede bijektive 2-Bit Substitution ist linear. †

Satz 2 Für jede 3-Bit Substitution existiert keine lineare Approximation mit einer Güte kleiner als 0.75. †

Satz 3 Jede binäre Funktion ist mit einer Güte von mindestens 0.5 linear approximierbar. †

Nicht nur Linearität einer Funktion F selbst, sondern auch die Existenz sog. **linearer Faktoren** kann F als kryptographisch gute Substitution unbrauchbar machen, da ihr Vorhandensein den zeitlichen Aufwand einer Kryptoanalyse erheblich verringern kann.
Ziel des Angreifers ist es, bei vorhandenen Paaren $\mathcal{M}/\mathcal{C}$ den zugehörigen Schlüssel K zu ermitteln. Dabei geht man nach folgendem Schema vor: Man kennt die Beschreibung der Substitution

$$\mathbf{E}: \quad \mathcal{M} \quad \times \quad K \quad \rightarrow \quad \mathcal{C}$$

mit

$$\begin{pmatrix} p_1 \\ p_2 \\ \\ \vdots \\ \\ \\ p_n \end{pmatrix} \times \begin{pmatrix} k_1 \\ k_2 \\ \\ \vdots \\ \\ \\ k_l \end{pmatrix} \rightarrow \begin{pmatrix} c_1 \\ \vdots \\ c_j \\ \vdots \\ c_k \\ \vdots \\ c_m \end{pmatrix} \tag{2.3}$$

und sucht 3 Abbildungen

$$\begin{array}{rrcl} f_{\mathcal{M}}: & \mathcal{M} & \rightarrow & \mathcal{M}' \\ f_K: & K & \rightarrow & K' \\ f_{\mathcal{C}}: & \mathcal{C} & \rightarrow & \mathcal{C}' \end{array} \tag{2.4}$$

Es entsteht damit eine Substitution

$$\mathbf{E}': \quad \mathcal{M}' \quad \times \quad K' \quad \rightarrow \quad \mathcal{C}'$$

Wenn für alle $\mathcal{M}$ und K

$$\mathcal{C}' = \mathbf{E}'(f_K(K), f_{\mathcal{M}}(\mathcal{M})) = f_{\mathcal{C}}(\mathbf{E}(K, \mathcal{M})) \tag{2.5}$$

gilt, und $f_{\mathcal{M}}$, f_K und $f_{\mathcal{C}}$ lineare Abbildungen sind, so ist das Gleichungssystem, das **E**' beschreibt linear. Damit läßt es sich mit geringerem Aufwand analysieren als die ursprüngliche Substitution **E**.

Bei geeigneter Abbildung $f_K : K \rightarrow K'$ ist die Rückführung von K aus K' einfach. Damit hätte man **E** durch den „Umweg“ über **E**' gebrochen.

[2] aus [FW88,S.201]

2.1.4 Korrelationsimmunität

Bei der Beschreibung der Korrelationsimmunität soll nicht auf die Funktion $F: \{0,1\}^n \to \{0,1\}^m$ als Gesamtheitbeschreibung der Abhängigkeit der Output-Bits von den Input-Bits eingegangen werden. Vielmehr wird die Korrelationsimmunität *eines* Outputbits zu seinen Input-Variablen beschrieben.

Definition 6 Sei $f(x_1, \ldots, x_n)$ eine boolesche Funktion in n Variablen.
f ist dann m-korrelationsimmun, wenn man aus Kenntnis einer *beliebigen* Menge von m Eingangswerten keine Informationen über des resulierenden Ausgangswert erhalten kann und umgekehrt. ‡

Warum eine Substitution korrelationsimmun sein soll, liegt auf der Hand: Jede Teilmenge der Output-Vektoren, die Rückschlüsse auf Teilmengen der Input-Vektoren zuläßt, verringert den Aufwand für das vollständige Durchsuchen des Schlüsselraumes. Lassen sich z.B. bei einer 8-Bit-Substitution durch die Werte der Output-Bits auch nur 2 Input-Bits rekonstruieren, hat man den Aufwand zum Vollständigen Suchen des Schlüssels bereits um den Faktor 4 gesenkt. Es sind statt maximal 256 Verschlüsselungsschritten nur noch maximal 64 Schritte notwendig.

Leider stehen Korrelationimmunität und Nichtlinearität in einer gegenläufigen Abhängigkeit. Dadurch ist es nicht möglich beide Gütekriterien einer Funktion F gleichzeitig zu maximieren[3].

2.2 Beurteilungsmethode ‚Berechnung der Abhängigkeitsmatrix'

Es soll in diesem Abschnitt eine Methode vorgestellt werden, nach der es möglich ist, insbesondere die Gütekriterien *Vollständigkeit, Linearität/parielle Linearität/Nichtlinearität, Avalache-Effekt* einer Funktion F zu beurteilen.

Definition 7 Die Abhängigkeitsmatrix AM einer Funktion $F: \{0,1\}^n \to \{0,1\}^m$ ist eine $(n \times m)$-Matrix, deren Einträge $a_{i,j}$ die Wahrscheinlichkeit angeben, daß bei Änderung des i-ten Inputbits das j-te Outputbit komplementiert wird. ‡

Eigenschaften von AM:

AM$(F = const)$ ist eine Nullmatrix
AM einer Permutation ist eine Permutationsmatrix
AM(F) = AM$(F \oplus 1)$

Eigenschaftem von F — aus AM ablesbar:

[3] vgl. [FW88,S.168],[Si84,MGB87]

$\exists i,j : a_{i,j} = 0$	$\longrightarrow$	Das j-te Output-Bit ist nicht vom i-ten Input-Bit abhängig.
	$\longrightarrow$	*F ist nicht vollständig*
$\forall i,j : a_{i,j} > 0$	$\longrightarrow$	*F ist vollständig*
$\exists i,j : a_{i,j} = 1$	$\longrightarrow$	Das j-te Output-Bit ändert sich bei jeder Änderung des i-ten Input-Bits.
	$\longrightarrow$	Das j-te Output-Bit hängt linear vom i-ten Input-Bit ab.
$\exists j.\forall i : a_{i,j} \in \{0,1\}$	$\longrightarrow$	Die Spalte a_j ist ein binärer Vektor.
	$\longrightarrow$	*F ist partiell linear.*
$\forall i,j : a_{i,j} \in \{0,1\}$	$\longrightarrow$	AM ist eine binäre Martix.
	$\longrightarrow$	*F ist linear.*
$\frac{1}{n}\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} \approx 0.5$	$\longrightarrow$	*F besitzt den Avalanche-Effekt.*
$\forall i,j : a_{i,j} \approx 0.5$	$\longrightarrow$	*F erfüllt das strikte Avalanche- Kriterium.*

Die exakte Berechnung der Abhängigkeitsmatrix ist normalerweise nur für kleine n und m möglich. Im folgenden Beispiel werden die Abhängigkeitsmatrizen für ausgewählte bijektive 3-Bit-Substitutionen angegeben. Sie zeigen deutlich die Aussagekraft solcher Matrizen, aber auch deren Grenzen.

Beispiel 6

b) Abhängigkeitsmatrix der folgenden S-Box

$X_{dex} \rightarrow Y_{dez}$
$0 \rightarrow 4$
$1 \rightarrow 5$
$2 \rightarrow 7$
$3 \rightarrow 6$
$4 \rightarrow 3$
$5 \rightarrow 2$
$6 \rightarrow 0$
$7 \rightarrow 1$

$$\begin{aligned} y_1 &= x_1 + x_2 + x_3 \\ y_2 &= x_2 + x_3 \\ y_3 &= x_3 + 1 \end{aligned} \quad \Rightarrow$$

AM	y_1	y_2	y_3
x_1	1	0	0
x_2	1	1	0
x_3	1	1	1

AM ist eine binäre Matrix. Es handelt sich um eine lineares Gleichungssystem zur Beschreibung der Substitution.

c) Abhängigkeitsmatrix AM(S_{Bsp}) (siehe Beispiel 3, Gleichungssystem (2.1))

Man sieht, daß S_{Bsp} vollständig ist, den Avalanche-Effekt besitzt und sogar das strikte Avalanche-Kriterium erfüllt. Wie bereits in Beispiel 3 erwähnt, ist jedoch S_{Bsp} kaum geeignet für eine kryptographisch gute Substitution, da sie zu 75% die identische Abbildung realisiert.

AM	y_1	y_2	y_3
x_1	0.5	0.5	0.5
x_2	0.5	0.5	0.5
x_3	0.5	0.5	0.5

BSP

2.3 Zusammenfassung

Es ist anhand der Beispiele gezeigt worden, daß die genannten Kriterien zwar notwendig, keineswegs aber hinreichend sind für gute Blockchiffren.

Vielmehr ist es in bestimmten Fällen gar nicht möglich, eine Substitution nach mehreren Kriterien zu *maximieren*. Hier muß eine *Optimierung* zwischen den einzelnen Kriterien vorgenommen werden.

Zusammenfassend können als Designkriterien für gute Substitutionsfunktionen F festgehalten werden:

NOTWENDIG: ein Höchstmaß an Vollständigkeit, Avalanche, Nichtlinearität und Korrelationsimmunität sowie Geringhaltung der Existenz linearer Faktoren von F.

GEWÜNSCHT: gute Implementierbarkeit, Schnelligkeit, Längentreue,
Minimierung von Fehlerfortpflanzungsmöglichkeiten vs. Integritäts-/ Authentizitätsprüfbarkeit

Die gewünschten Kriterien sollen das Bild abrunden. Hierauf wird in dieser Arbeit nicht näher eingegangen. In [SB82,S.14ff] finden sich weitere Erläuterungen.

2.4 Aufgaben

Aufgabe 1

Ist es möglich, daß eine lineare Funktion den Avalanche-Effekt oder sogar den strikten Avalanche-Effekt besitzen kann?

Aufgabe 2

Gegeben ist folgende Funktion $Y = F(X)$ und deren Abhängigkeitsmatrix.

$X_{dez} \rightarrow Y_{dez}$
$0 \rightarrow 4$
$1 \rightarrow 5$
$2 \rightarrow 0$
$3 \rightarrow 2$
$4 \rightarrow 6$
$5 \rightarrow 7$
$6 \rightarrow 1$
$7 \rightarrow 3$

$$\begin{aligned} y_1 &= x_1x_2 + x_2x_3 + x_1 \\ y_2 &= x_1x_2 + x_2x_3 + x_3 \\ y_3 &= x_2 + 1 \end{aligned} \quad \Rightarrow$$

AM	y_1	y_2	y_3
x_1	0.5	0.5	0
x_2	0.5	0.5	1
x_3	0.5	0.5	0

Beschreiben Sie welche Bewertungskriterien in welchem Maße erfüllt sind.

Kapitel 3

Kryptoanalyse von Blockchiffren

In diesem Kapitel soll ein kurzer Überblick über die wichtigsten Analysemethoden von Chiffren gegeben werden, die insbesondere im Zusammenhang mit Blockchiffren eine Rolle spielen.

Unter Kryptoanalyse soll das *Brechen* der jeweiligen Chiffre verstanden werden. Hierbei wird stets davon ausgegangen, daß das Chiffrierverfahren dem Angreifer vollständig bekannt ist. Die Kryptoanalyse „beschränkt“ sich also auf das Finden des Schlüssels K, wobei dem Angreifer

- bekannte Chiffretextblöcke *(ciphertext-only-attack)*, oder
- Paare von Klartext-, Chiffretextblöcken, die unter K entstanden *(known-plaintext-attack)*, oder
- vom Angreifer selbst wählbare und mit dem unbekannten K verschlüsselbare Klartextblöcke *(chosen-plaintext-attack)*

zur Verfügung stehen.

3.1 Extreme Ansätze

3.1.1 Vollständiges Durchsuchen des Schlüsselraumes

Die Grundidee des vollständigen Durchsuchens *(engl.: exhaustive search)* des Schlüsselraumes besteht im Durchprobieren aller Werte, die ein Schlüssel annehmen kann. Besitzt man z.B. Klartext-, Chiffretextpaare *(known-plaintext-attack)*, so verschlüsselt man die Klartexte mit jedem Schlüssel und vergleicht die entstandenen Chiffrate mit den bekannten Chiffretextblöcken.
Hier ist zu beachten, daß ein fester Klartextblock mit verschiedenen Schlüsseln den gleichen Chiffretextblock erzeugen kann! Es sind also u.U. für eine eindeutige Ermittlung von K mehrere Klartext-, Chiffretextpaare erforderlich.

Unter bestimmten Randbedingungen ist es einem Angreifer durch Vollständiges Suchen auch unter der Bedingung *ciphertext-only-attack* möglich, den Schlüssel K zu ermitteln.

Vollständiges Durchsuchen des Schlüsselraumes ist ein sehr rechenzeitaufwendiges Verfahren und in den meisten Fällen deshalb untauglich zum Brechen einer Chiffre. Einen stark eingeschränkten Schlüsselraum kann man jedoch in vielen Fällen durchsuchen.

3.1.2 Zugriff auf eine vorab berechnete Tabelle

Diese Angriffsart *(engl.: table lookup)* ist eine *chosen-plaintext-attack*. Man wählt einen festen Klartextblock und berechnet für jeden Schlüssel den Chiffretextblock. In einer durch die Chiffretextblöcke indizierten Tabelle wird nun der zugehörige Schlüssel abgespeichert.

Das Brechen der Chiffre mit dem unbekannten Schlüssel besteht im Verschlüsseln des gewählten Klartextblockes und Auslesen der vorab berechneten Tabelle an der durch den Chiffretextblock angegebenen Stelle.

Der Aufwand des Verfahrens beschränkt sich damit im wesentlichen auf die vorherige Berechnung und Speicherung der Tabelle.

Beispiel 7 Um für den DES eine solche Tabelle zu berechnen, benötigt man *vorab* denselben Zeitaufwand wie beim Vollständigen Durchsuchen des Schlüsselraumes und einen gigantischen Speicher. BSP

Vollständiges Durchsuchen des Schlüsselraumes und Zugriff auf eine vorab berechnete Tabelle kann man damit als die extremen Ansätze zur Kryptoanalyse von Chiffren bezeichnen.

3.2 Time-memory tradeoff

Diese Analysemethode ist eine Verbindung beider extremer Ansätze. Die Idee ist, daß man mit Einsatz von mehr Speicher einen hohen Rechenzeitaufwand kompensieren will und umgekehrt. Sie verwendet ebenfalls eine vorab berechnete Tabelle und Vollständiges Suchen. Dieses Verfahren geht von einer *chosen-plaintext-attack* aus. Dabei betrachtet man das Problem des Findens der Lösung[1] von N möglichen Lösungen[2] als eine Verknüpfung aus Speicherplatzaufwand und Zeitaufwand des Verfahrens.

Für Blockchiffren wurde von HELLMAN ein Angriffsszenario eines time-memory tradeoffs entwickelt, mit dem es möglich sein soll, sowohl den Zeit- als auch den Speicherplatzaufwand auf jeweils $\mathsf{N}^{2/3}$ zu verringern.

3.3 Codebuchanalyse

Diese Methode weicht von den bisherigen Angriffsmethoden ab: Hier wird versucht, aus einem Chiffretextblock den Klartextblock zu rekonstruieren, ohne dabei den Schlüssel zu ermitteln. Codebuchanalyse ist eine *chosen-plaintext-attack*. Man berechnet zu einer sehr großen Anzahl von Klartexten den zugehörigen Chiffretext unter dem unbekannten Schlüssel und speichert die so erhaltenen Paare in einer Tabelle (Codebuch) ab.

So kann man, falls das Codebuch groß genug und das zu analysierende Chiffrat lang genug ist, evtl. einige Klartextblöcke rekonstruieren und sich so, falls der Klartext eine gewisse Struktur (z.B. natürlichsprachlicher Klartext) besitzt, durch das gesamte Chiffrat arbeiten und Block für Block des Klartextes rekonstruieren.

Der Aufwand zur Kryptoanalyse ist somit nicht von der Größe des Schlüsselraumes abhängig, sondern von der Struktur des Klartextes.

[1] *(des gesuchten Schlüssels)*
[2] *(des Schlüsselraumes)* N ist also gewöhnlich 2^l mit l als Bitbreite des Schlüssels.

3.4 Methode des formalen Kodierens (MFC)

Bei der Methode des formalen Kodierens handelt es sich um eine *known-plaintext-attack*. Man setzt den Chiffriervorgang mit einem bekannten Klartextblock und den unbekannten Teilschlüsseln K_i allgemein an und erhält auf diese Weise m Terme in den Variablen k_j mit $1 \leq j \leq l$, wobei l die Anzahl der Schlüsselbits von K ist. m ist die Anzahl der Bits des Chiffretextblockes. Diese m Terme setzt man mit dem ebenfalls bekannten Chiffretextblock gleich. So entsteht ein Gleichungssystem, das stets eine Lösung besitzt. Die Eindeutigkeit der Lösung ist jedoch nicht gewährleistet. Existieren mehrere Lösungen, so werden die Scheinlösungen für K entweder durch probeweises Verschlüsseln des Klartextblockes mit allen K eliminiert, oder das Verfahren muß mit anderen Klartext-, Chiffretextpaaren wiederholt werden.

Für nichtlineare Funktionen F hat man einen sehr hohen Speicheraufwand zum Brechen einer Chiffre.

Wie das MFC-Verfahren konkret funktioniert, wird in Versuch 4 (siehe Seite 28) gezeigt.

3.5 Differentielle Kryptoanalyse

Die Methode der Differentiellen Kryptoanalyse ist ein Ansatz, der eng verbunden ist mit der wohl bekanntesten Feistel-Chiffre, dem DES.

Obwohl erst Anfang der 1990er Jahre Veröffentlichungen zur Differentiellen Kryptoanalyse verfügbar waren, kannten schon die Designer des DES im Jahre 1974 diese Methode[3]. Natürlich versuchten sie, einen Angriff mittels Differentieller Kryptoanalyse möglichst schwer zu gestalten. Damit läßt sich heute auch erklären, warum die Designkriterien der verwendeten S-Boxen jahrelang geheim gehalten wurden.

Der Aufwand des Verfahrens ist zunächst geringer als bei den extremen Ansätzen zur Kryptoanalyse. Er hängt insbesondere von der Struktur der nichtlinearen Komponenten, den S-Boxen, der Chiffre ab. So kann man annehmen, daß die Analyse einer Chiffre mit speziell gewählten S-Boxen – so wie beim DES – schwieriger ist die Analyse einer Chiffre mit stochastischen S-Boxen, was in [BS93] gezeigt wird.

Beispiel 8 Um den DES (mit 16 Runden und S-Boxen wie im Standard) zu brechen benötigt man ca. 2^{47} wählbare Klartexte bei ca. 2^{37} Verschlüsselungsschritten. Um den DES (mit 16 Runden und stochastischen S-Boxen) zu brechen benötigt man ca. 2^{21} wählbare Klartexte[4]. BSP

[3] vgl. [NEWS92]
[4] vgl. [BS93,S.87f]

Kapitel 4

Versuchsanordnungen

In zwei Teilversuchen sollen Klartextfolgen mit bestimmten statistischen Eigenschaften im Vordergrund stehen. In einer Gegenüberstellung

– redundanzbehafteter vs. stochastischer Klartext –

wird gezeigt, daß eine kryptographisch gute Feistel-Chiffre (wovon hier zunächst ausgegangen werden soll)

- statistische Eigenschaften des Klartextes durch die Eigenschaft der Vollständigkeit sowie Blockung verschleiert,
- die Kryptoanalysemethode *Vollständiges Durchsuchen des Schlüsselraumes* unter der Bedingung ciphertext-only-attack nicht zum Erfolg führt, wenn der Angreifer keine Informationen über die Beschaffenheit des Klartextes hat bzw. der Klartext redundanzfrei ist.

Drei weitere Teilversuche betrachten den Einfluß der Funktion F auf die Kryptosicherheit der Chiffre.
Es soll versucht werden,

- die Abhängigkeit der Ausgangsfolge von Änderungen der Eingangsfolge, der Rundenzahl und dem Aufbau der S-Boxen darzustellen,
- eine einfache Blockchiffre mit linearer Funktion F durch formales Dekodieren zu brechen,
- eine einfache Blockchiffre mit Hilfe differentieller Kryptoanalyse zu brechen.

Die Teilversuche sind nach folgendem Schema aufgebaut:
Zunächst werden allgemeine Anmerkungen über die zu untersuchende Problematik gemacht. Die bereitgestellten *Hilfsmittel* sowie ein mögliches *Vorgehen* zur Untersuchung werden beschrieben.

4.1 Versuche zur Problematik – redundanzbehafteter vs. stochastischer Klartext –

Für diese Versuchsgruppe soll als Chiffre der DES verwendet werden. Von dieser Chiffre weiß man, daß sie die in Abschnitt 1.3.1 aufgeführten Forderungen für gute Blocksubstitutionen in hohem Maße erfüllt.

Die Alphabetzeichen von Klartext und Chiffretext sollen 8-Bit-Zeichen sein, d.h. es gilt $A = \{0,1\}^8$. Ein DES-Block entsteht also durch Konkatenation von acht 8-Bit-Zeichen (Bytes). Natürlichsprachlicher Klartext soll kodiert sein im ASCII-Code.

Beispiel 9

Klartext:	D	e	u	t	l	i	c	h	e	r	_	j	e	d	o	c
(hexadez.)	44	65	75	74	6C	69	63	68	65	72	20	6A	65	64	6F	63
DES-Block:	Deutlich								er jedoc							
(hexadez.)	44 65 75 74 6C 69 63 68								65 72 20 6A 65 64 6F 63							

BSP

Jeder DES-Block wird unabhängig vom vorhergehenden Block verschlüsselt. Der DES arbeitet also im ECB-Modus.

Die Berechnung der *Redundanz* der Klartext-/ Chiffretextfolgen bezieht sich auf das Alphabet A. Damit ist die maximale Entropie der Folge $H_0 = 8$ bit/Zeichen. Die Redundanz berechnet sich nach

$$d = H_0 + \sum_{i=00_{hex}}^{FF_{hex}} (p_i \operatorname{ld} p_i) \text{ bit/Zeichen} \tag{4.1}$$

wobei p_i die Auftrittswahrscheinlichkeit des Zeichens i in einer Folge von Zeichen ist.

Die Berechnung der Bitverteilung der Klartext-/ Chiffretextfolgen bezieht sich ebenfalls auf das Alphabet A und erfolgt nach folgendem Schema: (Die Verteilung soll in einem Vektor $\mathsf{BV} = (v_0, v_1, \ldots, v_7)$ gespeichert werden.)

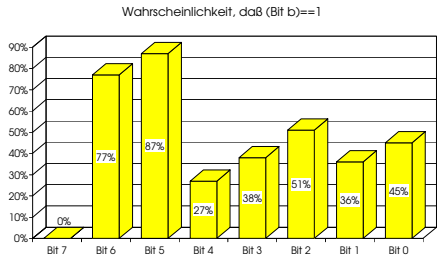
$\Rightarrow$ Die Komponenten v_b des Vektors BV geben die Wahrscheinlichkeit an, daß (Bit b)==1.

1. Setze alle $v_b := 0$.
2. DO für alle i von 1 bis *len* der Folge
   (a) DO für alle b von 0 bis $n-1$
      i. Addiere das b-te Bit des i-ten Zeichens auf den Inhalt des Vektorelementes v_b und speichere das Ergebnis wieder in v_b.
3. Dividiere jedes v_b durch Länge *len* der Folge und speichere das Ergebnis wieder in v_b.

Abbildung 4.1: Algorithmus BV zur Ermittung der Bitverteilung einer Folge von 8-Bit-Zeichen

Betrachtet man die Bitverteilung von natürlichsprachlichem Text, der im ASCII-Code codiert wurde, so fällt auf, daß die Verteilung der Nullen und Einsen je Bit recht unterschiedlich ist. So sind die Bits 5 und 6 recht häufig **1**, jedoch Bit 7 (MSB) stets **0**. Der ASCII-Code ist schließlich ein 7-Bit-Code.

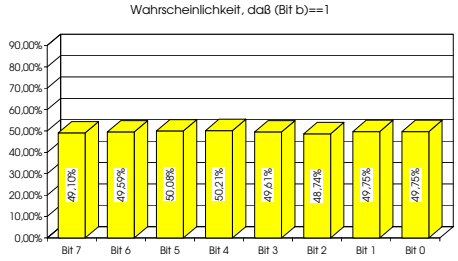
Beispiel 10 Beispiel für die Bitverteilung des vorliegenden Textes als 'plain'-ASCII.



BSP

Bei gleichverteilten und unabhängigen stochstischen Klartextzeichen aus dem Alphabet A kann man annehmen, daß auch die Bitverteilung der Klartextfolge gleichmäßig ist. Das folgende Beispiel soll das illustrieren.

Beispiel 11 Beispiel für die Bitverteilung eines Textes von 10000 gleichverteilten und statistisch unabhängigen Zeichen aus $A = \{0,1\}^8$.



BSP

Hilfsmittel: Programm `ClassCiph.jar`, „Datei->Textdatei öffnen“, Datei mit Klartext (hier: `dat2`), „Analysieren->DES Analysieren“, „Zeige Bitverteilung bei Verschlüsselung“

Vorgehen: In einem DES-Verschlüsselungsschritt werden 8 stochastische oder natürlichsprachliche Zeichen verschlüsselt. Es wird für jede Rundenzahl von 1 bis 16 Runden das Chiffrat berechnet sowie dessen Bitverteilung ermittelt.

„Zufälliger Text“: Aufzeichnung der Bitverteilung der Klartextzeichen einer zur Laufzeit generierten gleichverteilten Folge von 304 Klartextzeichen aus dem Alphabet $A = \{0,1\}^8$ sowie deren Chiffretextzeichen für 1...16 Runden.

„Benutze Arbeitstext“: Aufzeichnung der Bitverteilung der Klartextzeichen der Datei `dat2` (vorher als Arbeitstext geöffnet) sowie der Chiffretextzeichen für 1...16 Runden.

Es werden jeweils die ermittelten Werte für die Wahrscheinlichkeit, daß (Bit b)==1 ist, ausgegeben.

Ergebnisse: Da gemäß dem Avanlance-Effekt bereits bei Änderung eines Input-Bits im Mittel 50 % der Output-Bits verändert werden, muß man davon ausgehen, daß die für ASCII-kodierten natürlichsprachlichen Klartext typische Bitverteilung (Benachteiligung des MSB, Verteilung ähnlich der in Beispiel 10) im Chiffrat nicht mehr erkennbar sein wird, da jedes Klartextbit auf jedes Chiffretextbit Einfluß nimmt (Vollständigkeit[1]).

Dieser Prozeß der „Verschleierung“ soll in Abhängigkeit von der Rundenzahl dargestellt werden.

Die Gegenüberstellung von natürlichsprachlichem und stochastischem Klartext verdeutlicht die tatsächliche Verschleierung der Bitverteilung des Klartextes.

Wo die Bitverteilungen der Klartexte sich noch stark unterscheiden, nehmen die Unterschiede in den ersten 3 Runden deutlich ab:
Die a-priori-Gleichverteilung von stochastischem Klartext bleibt erhalten. Dagegen nähert sich die Verteilung bei natürlichsprachlichem Klartext deutlich der Gleichverteilung an bis sie schließlich ununterscheidbar von dieser geworden ist.

[1] siehe Seite 7

Die Bitverteilungen des Chiffrats sind sich in hohem Maße ähnlich. Bereits nach 3 bis 4 Runden kann man von einer Gleichverteilung der Bits des Chiffrats sprechen.
Das unterstreicht die Aussage, daß der DES nach 5 Runden vollständig ist[2].

Die Abbildungen 4.2 und 4.3 zeigen die Bitverteilungen für die Datei `dat2` sowie die zur Laufzeit des Programms generierte stochastische Folge.

Aufgabe 3

Nennen Sie verschiedene Möglichkeiten, wie man den Klartext in eine weniger redundante Form bringen könnte! Begründen Sie Ihre Antwort!

[2] vgl. [FW88,S.235]

Wkt., daß (Bit b) == **1** in % für $\mathcal{M}$, nach 1...16 Runden								
	Bit							
	7	6	5	4	3	2	1	0
$\mathcal{M}$	0	87	95	22	31	51	30	51
1	0	37	95	56	31	54	30	50
2	37	49	56	52	54	50	50	50
3	49	56	52	51	50	50	50	50
4	56	52	51	48	50	52	50	49
5	52	47	48	47	52	51	49	47
6	47	50	47	50	51	47	47	49
7	50	51	50	47	47	50	49	49
8	51	48	47	57	50	49	49	49
9	48	50	57	49	49	47	49	49
10	50	50	49	44	47	45	49	51
11	50	54	44	52	45	46	51	49
12	54	50	52	47	46	49	49	52
13	50	51	47	47	49	44	52	50
14	51	46	47	59	44	44	50	49
15	46	48	59	52	44	47	49	47
16	48	48	52	52	47	47	47	53

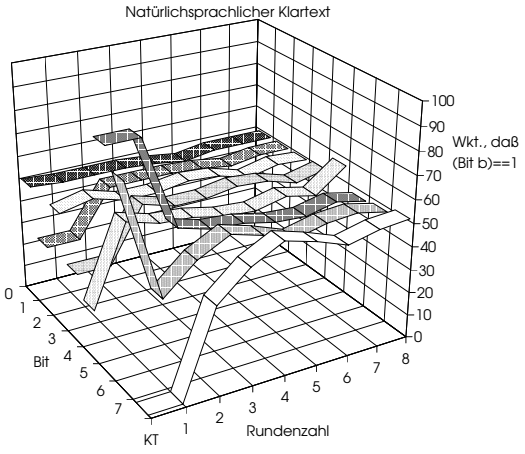


Abbildung 4.2: Bitverteilung – natürlichsprachlicher Klartext – Tabelle und Grafik

Wkt., daß (Bit b) == **1** in % für $\mathcal{M}$, nach 1...16 Runden								
	Bit							
	7	6	5	4	3	2	1	0
$\mathcal{M}$	52	48	50	46	51	47	50	53
1	52	53	50	48	51	52	50	48
2	53	47	48	46	52	47	48	53
3	47	52	46	49	47	50	53	48
4	52	52	49	50	50	50	48	49
5	52	50	50	49	50	46	49	51
6	50	49	49	55	46	49	51	50
7	49	59	55	49	49	50	50	55
8	59	52	49	49	50	51	55	46
9	52	53	49	52	51	48	46	49
10	53	54	52	47	48	47	49	48
11	54	50	47	48	47	51	48	52
12	50	43	48	49	51	46	52	43
13	43	48	49	50	46	52	43	51
14	48	48	50	49	52	48	51	47
15	48	50	49	50	48	49	47	49
16	50	51	50	55	49	51	49	49

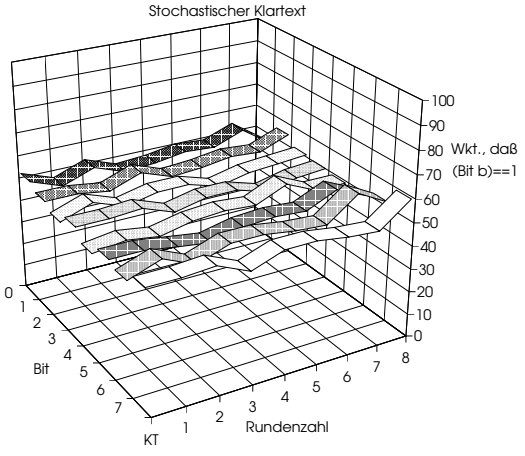


Abbildung 4.3: Bitverteilung – stochastischer Klartext – Tabelle und Grafik

Im Folgenden soll davon ausgegangen werden, daß keine Klartext-/Chiffretextpaare bekannt sind, sondern nur Chiffretext zur Verfügung steht. Es handelt sich also um eine *ciphertext-only-attack*.

Bei vollständiger Suche müssen maximal 2^l Verschlüsselungsschritte durchgeführt werden (mit l als Bitbreite des verwendeten Schlüssels).

Dieser Versuch soll zeigen, daß einerseits die Methode Vollständiges Suchen unter gewissen Bedingungen (ciphertext-only, Gleichverteilung der Klartextzeichen) nicht zwingend zum Ziel führen muß, andererseits die Bitverteilung der Klartextzeichen Anhaltspunkte geben kann zur Kryptoanalyse.

Der Angreifer besitzt folgende Informationen:

- Die Blockbreite des Verfahrens ist $n = 64$ Bit.

- Es werden 1-Byte-Informationen (aus dem Alphabet $A = \{0,1\}^8$) verarbeitet. Das Chiffrierverfahren verschlüsselt 8 Byte in einem Schritt.
- Der Angreifer kennt nur Chiffretext.
- Der Schlüsselraum ist $K = 1 \ldots 127$.

Hilfsmittel: Programm `ClassCiph.jar`, „Datei->Textdatei öffnen“, Datei mit Klartext oder Chiffretext (hier: `dat2`), „Analysieren->DES Analysieren“, „Vollständige Suche durchführen“

Wenn „Text verschlüsseln“ gewählt wird, wird davon ausgegangen, daß der verwendete Text ein Klartext ist. Dies dient der Demonstration des Vollständigen Suchens. Es kann ein Schlüssel $K = 1 \ldots 127$ gewählt werden. Das erhaltene Chiffrat wird angezeigt. Für jeden Schlüssel wird nun der Chiffretext entschlüsselt und die Redundanz d der erhaltenen Nachricht berechnet.

Wird bei „Textauswahl“ ein „Zufälliger Text“ ausgewählt, wird eine zur Laufzeit generierte gleichverteilte Folge von ca. 300 Klartextzeichen aus dem Alphabet $A = \{0,1\}^8$ mit selbst wählbarem Schlüssel verschlüsselt und ebenfalls die Methode Vollständiges Suchen auf das erhaltene Chiffrat angewendet.

Wenn „Text als Schlüsseltext verwenden“ gewählt wird, wird angenommen, dass der verwendete Text ein Chiffretext ist. Es wird die Methode Vollständiges Suchen direkt auf den Text angewendet. Für jeden Schlüssel $K = 1 \ldots 127$ wird nun der Text entschlüsselt und die Redundanz d der erhaltenen Nachricht berechnet.
Es liegt nun am Angreifer, aus der Menge der ausgegebenen Nachrichten den korrekten Klartext zu ermitteln. Im Falle von natürlichsprachlichem Text stellt das kein Problem dar. Stochastischer Klartext jedoch besitzt keine Merkmale, die ihn von Falschentschlüsselungen unterscheidbar machen ...

Vorgehen: Es sollen 2 Fälle betrachtet werden:

1. Es wurde sinnvoller Klartext verschlüsselt. In diesem Fall muß die Redundanz der Nachricht größer als 1 Bit sein, da es sich um ASCII-Code handeln muß. Ein empirischer Wert ist ca. 3.5 Bit Redundanz von sinnvollem Klartext auf einem Alphabet von 2^8 Zeichen.
2. Es wurde stochastischer Klartext verschlüsselt. In diesem Fall muß man davon ausgehen, daß die Zeichen der Nachricht über dem Alphabet gleichmäßig verteilt sind. Damit ist die Redundanz erheblich geringer. Außerdem ist die Streuung der Redundanzen aller möglichen Klartexte geringer.

Hier wird zur Veranschaulichung der Demonstrationsmodus verwendet:

Text `dat2`: Verschlüsselung der Datei `dat2` mit selbst wählbarem Schlüssel ($K = 1 \ldots 127$), Vollständiges Durchsuchen des Schlüsselraumes, Berechnung der Redundanz des ermittelten „Klartextes“

Zufälliger Text: Verschlüsselung einer zur Laufzeit generierten gleichverteilten Folge von 300 Klartextzeichen aus dem Alphabet $A = \{0,1\}^8$ mit selbst wählbarem Schlüssel ($K = 1 \ldots 127$), Vollständiges Durchsuchen des Schlüsselraumes, Berechnung der Redundanz des ermittelten „Klartextes“

Aufgabe 4

Man wähle eine Datei aus den zur Verfügung gestellten Chiffretextdateien `ctxt0`, `ctxt1`, ..., `ctxt9` aus (Auswahlkriterium z.B. Endziffer der Matrikelnummer, bitte angeben!) und ermittle durch Vollständiges Suchen den Schlüssel K, falls möglich, oder zumindest z.B. die Redundanz.

4.2 Versuche zum Design der Funktion F

4.2.1 Auswirkung kleiner Änderungen eines Klartextblocks auf den Chiffretextblock bei starken bzw. schwachen Kryptofunktionen

Zunächst wird ein Begriff eingeführt, der im Zusammenhang mit diesem Versuch eine Rolle spielt – das Hamming-Gewicht.

Sei $X = (x_1, x_2, \ldots, x_n)$ ein binärer Vektor, dann bezeichnet die Zahl seiner von Null verschiedenen Koeffizienten das Hamming-Gewicht $\mathsf{HG}(X)$ des Vektors X.

Beispiel 12 Das Hamming-Gewicht des Vektors $X = (1, 0, 0, 1, 1, 0, 1)$ ist $\mathsf{HG}(X) = 4$. BSP

Dieser Versuch soll ebenfalls mit Hilfe des DES durchgeführt werden. Die Kenntnis der inneren Struktur des DES dient als Voraussetzung für das Verständnis des Versuches!

Hilfsmittel: Programm `ClassCiph.jar`, „Analysieren->DES Analysieren“, S-Boxen analysieren

Folgende S-Boxen sind für diesen Versuch verfügbar:

`linear`	Es werden 8 lineare S-Boxen entsprechend Formel (2.2) bereitgestellt ($n = 6$, $m = 4$). Die $a_{j,i}$ und b_j wurden zufällig gewählt.
`stochastisch`	Weiterhin werden 8 stochastische S-Boxen und
`identität`	8 S-Boxen der Form $(i_1, i_2, i_3, i_4, i_5, i_6) \rightarrow (i_2, i_3, i_4, i_5)$ bereitgestellt.
`nicht-linear`	Die S-Boxen des DES-Standards sind ebenfalls verfügbar.

Vorgehen: „Hamming-Gewichte“ berechnet zu einem zufällig gewählten Klartextblock $\mathcal{M}$ jeweils einen passenden, aber zufälligen Klartextblock $\mathcal{M}^*$, so daß $\mathsf{HG}(\mathcal{M} \oplus \mathcal{M}^*)$ Schritt für Schritt um Eins erhöht wird $(1 \ldots 64)$. Für jede Rundenzahl $(1 \ldots 16)$ der Feistel-Chiffre wird $\mathsf{HG}(\mathbf{E}(K, \mathcal{M}) \oplus \mathbf{E}(K, \mathcal{M}^*))$ berechnet.

„Abhängigkeitsmatrix“ berechnet die Abhängigkeitsmatrizen der S-Boxen. Um eine Entscheidung über das Besitzen des Avalanche-Effekts einer S-Box zu erleichtern, werden die Mittelwerte aller Matrixelemente berechnet. Liegt dieser Mittelwert um 0.5, kann man davon ausgehen, daß die S-Box den Avalanche-Effekt besitzt.

Ergebnisse: Die Abbildungen 4.4 und 4.5 zeigen die Ergebnisse der Berechnungen der Hamming-Gewichte für die Boxen `linear` und `nicht-linear`.

Es zeigt sich, daß die S-Boxen `linear` und `nicht-linear` keine deutlichen Unterschiede bezüglich des Avalanche-Effekts aufweisen. Die Ursache hierfür liegt natürlich im Vorhandensein der Expansions- und Permutationsabbildung, aber auch in den S-Boxen selbst. Dies unterstreicht die Tatsache, daß Avalanche und Nichtlinearität nicht zwingend zusammengehören: Es kann eine Feistel-Chiffre mit linearer S-Box durchaus den Avalanche-Effekt besitzen.

Zur Bestätigung dieser These wurden die Abhängigkeitsmatrizen von linearer (`lin`) und nichtlinearer (`nlin`) S-Box berechnet. Die Mittelwerte der Matrixelemente (der jeweils 8 S-Boxen) betragen:

`lin`	0.33 , 0.38 , 0.50 , 0.62 , 0.54 , 0.54 , 0.54 , 0.38
`nlin`	0.62 , 0.63 , 0.66 , 0.61 , 0.63 , 0.65 , 0.66 , 0.62

Die ermittelten Werte weichen in `lin` um maximal 0.17 und in `nlin` um maximal `nlin` 0.16 vom idealen Wert von 0.5 ab. Man könnte also davon ausgehen, daß `lin` und `nlin` bezüglich des Avalanche-Effekts ähnlich gute Ergebnisse erwarten lassen.

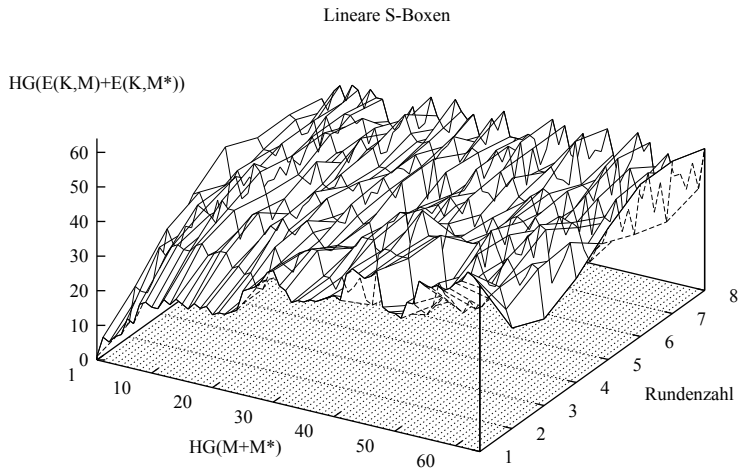


Abbildung 4.4: Versuch zu den S-Boxen: S-Box `lin` (Lineare Boxen) – Grafik

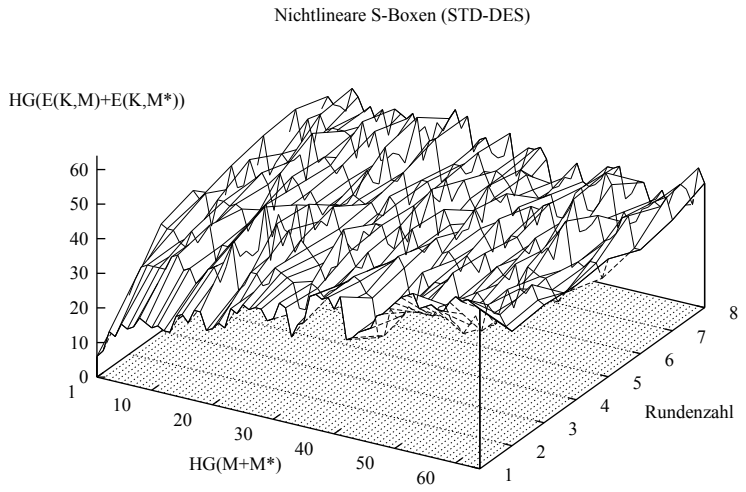


Abbildung 4.5: Versuch zu den S-Boxen: S-Box `nlin` (DES-Standard) – Grafik

Aufgabe 5

Man ermittle die Mittelwerte der Matrixelemente der Abhängigkeitsmatrix der stochastischen S-Boxen, vergleiche das Ergebnis mit den nicht-linearen S-Boxen und interpretiere.
Zusatzaufgabe: Warum wird im Verschlüsselungsalgorithmus DES (trotzdem) die nicht-lineare S-Box (`nlin`) verwendet?

4.2.2 Brechen einer einfachen Feistel-Chiffre mit der Methode des formalen Kodierens (MFC)

In diesem Versuch soll dargestellt werden, wie man mit Hilfe der Methode des formalen Kodierens (MFC) eine Feistel-Chiffre brechen kann. Der Schwerpunkt der Betrachtungen liegt in der Gegenüberstellung – lineare vs. nichtlineare Funktion F –

Zunächst soll aber eine „überschaubare“ Feistel-Chiffre VDES („Vereinfachter DES“) definiert werden, damit sich der Versuch überhaupt durchführbar gestaltet[3]:

Es sei $n = m = l = 4$ Bit. Es handelt sich also um eine 4-Bit längentreue Blockchiffre (Register L_i und R_i also jeweils 2 Bit) mit einem 4-Bit Hauptschlüssel $K = (k_1, k_2, k_3, k_4)$. Die Chiffre arbeitet mit 3 Runden, d.h. $N = 3$. Die Funktion $F(K_i, R_{i-1})$ mit $(i = 0 \ldots 3)$ sei zusammengesetzt aus einer Expansionsabbildung E und einer Substitution S, so daß

$$F(K_i, R_{i-1}) = \mathrm{S}(\mathrm{E}(R_{i-1}) \oplus K_i)$$

Es ist E durch

$$\mathrm{E} : (r_1, r_2) \rightarrow (r_1, r_2, r_1)$$

definiert. S soll je nach Variante (lineares / nichtlineares F) sein:

$$\begin{array}{lll} \mathrm{S_{lin}} : & (x_1, x_2, x_3) \rightarrow & (x_1 \oplus x_2 \oplus x_3, x_1 \oplus \overline{x}_3) \\ \mathrm{S_{nlin}} : & (x_1, x_2, x_3) \rightarrow & (x_1 \oplus x_2 \oplus x_3, \overline{x}_1\overline{x}_2 \oplus x_1x_3) \end{array}$$

oder als Tabelle:

x_1, x_2, x_3	$\mathrm{S_{lin}}$	$\mathrm{S_{nlin}}$
000	01	01
001	10	11
010	11	10
011	00	00
100	10	10
101	01	01
110	00	00
111	11	11

Tabelle 4.1: Zuordnungstabelle der S-Boxen $\mathrm{S_{lin}}$ und $\mathrm{S_{nlin}}$

Die Teilschlüssel K_i seien

$$\begin{array}{rcl} K_1 & = & (k_1, k_2, k_3) \\ K_2 & = & (k_3, k_4, k_1) \\ K_3 & = & (k_2, k_3, k_4) \end{array}$$

[3] siehe auch [SB82,S.28ff]

Hilfsmittel: Da es nötig sein wird, äquivalente Umformungen vorzunehmen, dient die folgende Übersicht der Darstellung von Äquivalenzen: ($a, b \in \{0,1\}$ und A ein Term aus $(\{0,1\}, \oplus, \odot)$)

$$\begin{aligned}
A \oplus a \oplus 1 &= A \oplus \overline{a} \\
A \oplus a \oplus a &= A \\
A \oplus a \oplus \overline{a} &= A \oplus 1 \\
A \oplus \overline{a} \oplus \overline{b} &= A \oplus a \oplus b \\
Aaa &= Aa \\
Aa\overline{a} &= 0 \\
A \oplus A &= 0 \\
Aa \oplus A &= A\overline{a} \\
Aa \oplus A\overline{a} &= A \\
Aa\overline{b} \oplus A\overline{a}b &= Aa \oplus Ab \\
Aab \oplus A\overline{a} &= Aa\overline{b} \oplus A
\end{aligned}$$

Vorgehen: Bei der Methode des formalen Kodierens handelt es sich stets um eine *known-plaintext-attack*. Man setzt den Chiffriervorgang mit dem bekannten $\mathcal{M} = (p_1, p_2, p_3, p_4)$ und den Teilschlüsseln K_i allgemein an und erhält auf diese Weise 4 Terme in den Variablen k_1, k_2, k_3 und k_4. Diese setzt man mit dem ebenfalls bekannten $\mathcal{C} = (c_1, c_2, c_3, c_4)$ gleich. Das so entstandene Gleichungssystem besitzt stets eine Lösung für die k_1 bis k_4, dessen Eindeutigkeit jedoch nicht gewährleistet ist. Existieren mehrere Lösungen, so werden die Scheinlösungen für K entweder durch Probieren eliminiert, oder das Verfahren muß mit weiteren $\mathcal{M}/\mathcal{C}$-Paaren wiederholt werden, bis die Lösungsmenge soweit eingeschränkt ist, daß nur wenige Lösungen zum Probieren übrig bleiben.

1. *Dieses Verfahren wird zunächst für die Chiffre mit* $\mathrm{S} \equiv \mathrm{S}_{\mathrm{nlin}}$ *durchgeführt. Es sei ein* $\mathcal{M}/\mathcal{C}$*-Paar* $(0,1,1,0)/(0,0,1,1)$ *bekannt. Gesucht ist* K*:*

1.1 Formaler Ansatz mit $\mathcal{M} = (p_1, p_2, p_3, p_4) = (0,1,1,0)$

$$\begin{aligned}
L_0 &= (0,1) & R_0 &= (1,0) \\
L_1 &= (1,0) & R_1 &= (0,1) \oplus F((k_1,k_2,k_3),(1,0)) \\
&& &= (0,1) \oplus \mathrm{S}(\overline{k}_1, k_2, \overline{k}_3) \\
&& &= (0,1) \oplus (\overline{k}_1 \oplus k_2 \oplus \overline{k}_3, k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3) \\
&& R_1 &= (k_1 \oplus k_2 \oplus k_3, k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus 1) \\
L_2 &= R_1 & R_2 &= (1,0) \oplus \mathrm{S}((k_1 \oplus k_2 \oplus k_3, k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus 1, k_1 \oplus k_2 \oplus k_3) \oplus (k_3, k_4, k_1)) \\
&& &= (1,0) \oplus \mathrm{S}(k_1 \oplus k_2, k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus \overline{k}_4, k_2 \oplus k_3)) \\
&& &= (1,0) \oplus (k_1 \oplus k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus \overline{k}_4 \oplus k_3, (k_1 \oplus \overline{k}_2)(k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus k_4) \oplus (k_1 \oplus k_2)(k_2 \oplus k_3)) \\
&& R_2 &= (k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4, k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus \overline{k}_3) \\
L_3 &= R_2 & R_3 &= L_2 \oplus \mathrm{S}((k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4, k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus \overline{k}_3, k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4) \oplus (k_2, k_3, k_4)) \\
&& &= L_2 \oplus \mathrm{S}(\overline{k}_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4, k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus 1, k_1k_2 \oplus k_1\overline{k}_3 \oplus 1) \\
&& &= L_2 \oplus ((\overline{k}_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4 \oplus k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus 1 \oplus k_1k_2 \oplus k_1\overline{k}_3 \oplus 1), \\
&& && \quad ((\overline{k}_1k_2 \oplus k_1\overline{k}_3 \oplus k_4)(k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4) \oplus (\overline{k}_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4)(k_1k_2 \oplus k_1\overline{k}_3 \oplus 1))) \\
&& &= L_2 \oplus (k_1\overline{k}_2k_3 \oplus \overline{k}_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus k_1 \oplus k_2, k_1\overline{k}_2k_3\overline{k}_4 \oplus k_1\overline{k}_3 \oplus k_1k_2k_4 \oplus k_2 \oplus k_4) \\
&& &= (k_1 \oplus k_2 \oplus k_3 \oplus k_1\overline{k}_2k_3 \oplus \overline{k}_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus k_1k_2, \\
&& && \quad k_1\overline{k}_2 \oplus \overline{k}_1\overline{k}_3 \oplus 1 \oplus k_1\overline{k}_2k_3\overline{k}_4 \oplus k_1\overline{k}_3 \oplus k_1k_2k_4 \oplus k_2 \oplus k_4) \\
&& R_3 &= (k_1\overline{k}_2k_3 \oplus \overline{k}_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus k_3, k_1\overline{k}_2k_3k_4 \oplus \overline{k}_1\overline{k}_2 \oplus \overline{k}_1k_4 \oplus \overline{k}_3)
\end{aligned}$$

1.2 Gleichsetzen der erhaltenen Terme mit $\mathcal{C} = (c_1, c_2, c_3, c_4) = (0,0,1,1)$

$$\begin{aligned}
&\text{(I):} & c_1 &= k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4 &&= 0 \\
&\text{(II):} & c_2 &= k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus \overline{k}_3 &&= 0 \\
&\text{(III):} & c_3 &= k_1\overline{k}_2k_3 \oplus \overline{k}_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus k_3 &&= 1 \\
&\text{(IV):} & c_4 &= k_1\overline{k}_2k_3k_4 \oplus \overline{k}_1\overline{k}_2 \oplus \overline{k}_1k_4 \oplus \overline{k}_3 &&= 1
\end{aligned}$$

1.3 Lösen des Gleichungssystems (I)-(IV)

1.3.1 $\overline{c_2} = c_3$
$= k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus \overline{k}_3 \oplus 1 = k_1\overline{k}_2k_3 \oplus \overline{k}_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus k_3$
$0 = k_1k_2k_3 \oplus k_1\overline{k}_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_1\overline{k}_4$
$\overline{k}_4 = k_1k_3$ bzw. $k_4 = k_1k_3 \oplus 1$ $(*)$

1.3.2 $(*)$ in (I)
$0 = k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4 = k_1k_2 \oplus k_1\overline{k}_3 \oplus k_1k_3 = k_1(k_2 \oplus \overline{k}_3 \oplus k_3)$
$0 = k_1k_3$ $(**)$

1.3.3 $(*)$ und $(**)$ in (II)
$0 = k_1k_2k_3 \oplus k_1\overline{k}_4 \oplus \overline{k}_2k_4 \oplus \overline{k}_3 = 0 \oplus k_1k_3 \oplus \overline{k}_2(k_1k_3 \oplus 1) \oplus \overline{k}_3$
$0 = k_1k_3 \oplus k_1\overline{k}_2k_3 \oplus \overline{k}_2 \oplus \overline{k}_3 = k_1k_2k_3 \oplus \overline{k}_2 \oplus \overline{k}_3$
$0 = 0 \oplus \overline{k}_2 \oplus \overline{k}_3$
$\overline{k}_2 = \overline{k}_3$ bzw. $k_2 = k_3$ $(***)$

1.3.4 mit $(***)$ in $(**)$ und $(*)$ folgt
$0 = k_1k_3 = \overline{k}_4$
$k_4 = 1$

1.3.5 (I) mit $\overline{k}_4 = 0$ und $(*)$
$0 = k_1k_2 \oplus k_1\overline{k}_3 \oplus \overline{k}_4 = k_1k_2 \oplus k_1\overline{k}_2 \oplus 0$
$k_1 = 0$

1.3.6 Eine eindeutige Lösung für $K_2 = k_3$ wird nicht gefunden. Probeweise Verschl. von $\mathcal{M}$ liefert $\mathcal{C}$ bei
$0 = k_2 = k_3$

1.4 Die Lösung bei $S \equiv S_{nlin}$ ist $\underline{\underline{K = (k_1, k_2, k_3, k_4) = (0, 0, 0, 1)}}$

---

2. *Nun soll die Chiffre mit* $S \equiv S_{lin}$ *durch MFC gebrochen werden. Es sei wieder das* $\mathcal{M}/\mathcal{C}$*-Paar* $(0,1,1,0)/(0,0,1,1)$ *bekannt. Gesucht ist* K*:*

2.1 Formaler Ansatz mit $\mathcal{M} = (p_1, p_2, p_3, p_4) = (0, 1, 1, 0)$

$L_0 = (0,1)$ $R_0 = (1,0)$

$L_1 = (1,0)$ $R_1 = (0,1) \oplus F((k_1,k_2,k_3),(1,0))$
$= (0,1) \oplus S((1,0,1) \oplus (k_1,k_2,k_3))$
$= (0,1) \oplus (\overline{k}_1 \oplus k_2 \oplus \overline{k}_3, \overline{k}_1 \oplus k_3)$
$= (0,1) \oplus (k_1 \oplus 1 \oplus k_2 \oplus k_3 \oplus 1, \overline{k}_1 \oplus k_3)$
$= (0,1) \oplus (k_1 \oplus k_2 \oplus k_3, \overline{k}_1 \oplus k_3)$
$R_1 = (k_1 \oplus k_2 \oplus k_3, k_1 \oplus k_3)$

$L_2 = R_1$ $R_2 = (1,0) \oplus \mathrm{S}((k_1 \oplus k_2 \oplus k_3, k_1 \oplus k_3, k_1 \oplus k_2 \oplus k_3) \oplus (k_3, k_4, k_1))$
$= (1,0) \oplus S(k_1 \oplus k_2, k_1 \oplus k_3 \oplus k_4, k_2 \oplus k_3)$
$= (1,0) \oplus (k_4, k_1 \oplus k_3 \oplus 1)$
$R_2 = (\overline{k}_4, k_1 \oplus \overline{k}_3)$

$L_3 = R_2$ $R_3 = (k_1 \oplus k_2 \oplus k_3, k_1 \oplus k_3) \oplus S((\overline{k}_4, k_1 \oplus \overline{k}_3, \overline{k}_4) \oplus (k_2, k_3, k_4))$
$= (k_1 \oplus k_2 \oplus k_3, k_1 \oplus k_3) \oplus S(k_2 \oplus \overline{k}_4, k_1 \oplus 1, 1)$
$= (k_1 \oplus k_2 \oplus k_3, k_1 \oplus k_3) \oplus (k_1 \oplus k_2 \oplus \overline{k}_4, k_2 \oplus \overline{k}_4)$
$R_3 = (k_3 \oplus \overline{k}_4, k_1 \oplus k_2 \oplus k_3 \oplus \overline{k}_4)$

2.2 Gleichsetzen der erhaltenen Terme mit $\mathcal{C} = (c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$

(I): $c_1 = \overline{k}_4 = 0$
(II): $c_2 = k_1 \oplus \overline{k}_3 = 0$
(III): $c_3 = k_3 \oplus \overline{k}_4 = 1$
(IV): $c_4 = k_1 \oplus k_2 \oplus k_3 \oplus \overline{k}_4 = 1$

2.3 Lösen des Gleichungssystems (I)-(IV)

2.3.1 aus (I) folgt unmittelbar
$$k_4 = 1$$

2.3.2 mit $\overline{k}_4 = 0$ in (III)
$$1 = k_3 \oplus \overline{k}_4 = k_3 \oplus 0$$
$$k_3 = 1$$

2.3.3 mit $\overline{k}_3 = 0$ in (II)
$$0 = k_1 \oplus \overline{k}_3 = k_1 \oplus 0$$
$$k_1 = 0$$

2.3.4 aus (IV) folgt damit
$$1 = k_1 \oplus k_2 \oplus k_3 \oplus \overline{k}_4 = 0 \oplus k_2 \oplus 1 \oplus 0$$
$$k_2 = 0$$

2.4 Die Lösung bei $S \equiv S_{lin}$ ist $\underline{\underline{K = (k_1, k_2, k_3, k_4) = (0, 0, 1, 1)}}$

Ergebnisse: In beiden betrachteten Fällen konnte die Feistel-Chiffre gebrochen werden. Im Fall S_{nlin} war der Aufwand wegen der Nichtlinearität von S erheblich höher als im Fall S_{lin}. Da S_{nlin} wegen der linearen Abhängigkeit des ersten Bits sogar partiell-linear war, kann man annehmen, daß bei einer S-Box, in der alle Output-Bits nichtlinear von den Input-Bits abhängen, der Aufwand zur Kryptoanalyse noch höher gewesen wäre.
Weiterhin war im Fall S_{nlin} keine eindeutige Lösung des Gleichungssystems zu finden.
Dagegen wurde im linearen Fall sofort eine eindeutige Lösung für K gefunden.

Um die Problematik der „Kompliziertheit“ einer Kryptofunktion F und damit der Schwierigkeit der Brechbarkeit einer Chiffre durch MFC etwas exakter zu fassen, soll ein einfaches Komplexitätsmaß Verwendung finden:
Die Anzahl aller Summanden eines Termes $T(x_1, \ldots, x_n)$ gibt die MFC-Komplexität des Termes T an.

Die Summe der MFC-Komplexitäten der Terme T_j, die gleichgesetzt wurden mit gebundenen Chiffretext-Variablen c_j gibt die MFC-Komplexität des Gleichungssystems, das durch die Gleichungen $c_j = T_j$ mit $(j = 1 \ldots m)$ entsteht, an.

Beispiel 13

1. Die MFC-Komlexität von $T_1(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ ist 3.
2. Die MFC-Komlexität von $T_2(x_1, x_2, x_3) = x_1x_2 \oplus x_1\overline{x}_3 \oplus x_2x_3 \oplus x_1 \oplus x_2 \oplus \overline{x}_3$ ist 6.
3. Die MFC-Komplexität des Gleichungssystems

$$c_1 = x_1 \oplus x_2 \oplus x_3$$
$$c_2 = x_1x_2 \oplus x_1\overline{x}_3 \oplus x_2x_3 \oplus x_1 \oplus x_2 \oplus \overline{x}_3$$

ist 9. BSP

Man nimmt an, daß die Terme, die zum Gleichsetzen mit den Output-Bits c_j dienen, in einer minimalen, d.h. einer in der Anzahl der auftretenden Summanden minimierten Form vorhanden sind. Gerade hier liegt jedoch die Problematik des Verfahrens, denn sowohl die automatisierte Reduktion von Termen in ihre minimale Form[4] als auch die Verfügbarkeit ausreichender Ressourcen für die Speicherung realer (und damit um Größenordnungen komplizierterer) Terme machen die Grenzen des Verfahrens deutlich.

[4] Nach [SB82,S.34] ist das Finden solcher minimaler Terme mit einem vertretbaren Aufwand bis heute nicht gelöst. Es werden jedoch in [SB82] zwei Algorithmen angegeben, die auf den hier ebenfalls angeführten Äquivalenzen in $(\{0,1\}, \oplus, \odot)$ beruhen.

Aufgabe 6

Man gebe die MFC-Komplexität der Gleichungssysteme zur Analyse der oben behandelten Feistel-Chiffre für beide Fälle (S_{nlin} und S_{lin}) an.

Aufgabe 7

Die obere Schranke der MFC-Komplexität des DES ist nach [SB82,S.38]

$$64 \cdot \sum_{i=0}^{56} \left(\begin{array}{c} 56 \\ i \end{array} \right) = 64 \cdot 2^{56} \approx 5 \cdot 10^{18}.$$

Erklären sie, wie diese Zahl zustandekommt.

Aufgabe 8

Man gebe eine obere Schranke für die MFC-Komplexität des DES mit linearen S-Boxen an.

Aufgabe 9

Es sei ein $\mathcal{M}/\mathcal{C}$-Paar $(1,0,1,1)/(1,1,1,0)$ bekannt. Als S-Box wurde S_{lin} verwendet. Man ermittle K.

4.2.3 Brechen einer einfachen Feistel-Chiffre durch Differentielle Kryptoanalyse

Dieser Versuch soll ebenfalls durchgeführt werden anhand der Feistel-Chiffre VDES aus [SB82], die bereits in Teilversuch 4 (siehe Seite 4.2.2) Anwendung fand.

Zunächst jedoch folgt eine Einführung in die Differentielle Kryptoanalyse[5].

Differentielle Kryptoanalyse geht von einer *chosen-plaintext-attack* aus. Sie analysiert die Auswirkungen bestimmter „Differenzen“[6] in Klartext-Paaren auf die „Differenzen“ der resultierenden Chiffretext-Paare. Sie kann auch angewendet werden unter der Bedingung einer *known-plaintext-attack*, allerdings sind hier sehr viel mehr Paare erforderlich als im chosen-plaintext-Fall!

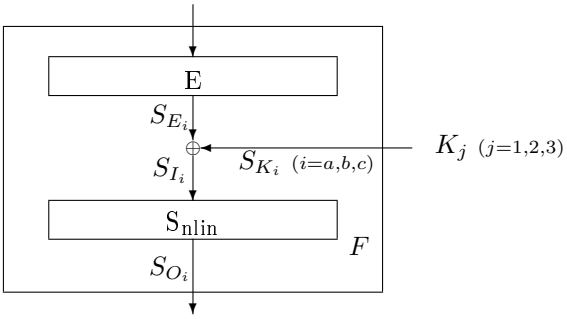
Diese „Differenzen“ werden verwendet, um den möglichen Schlüsseln des Kryptosystems Wahrscheinlichkeiten zuzuordnen um damit den wahrscheinlichsten Schlüssel zu finden, mit dem gewählte oder vorhandene Klartextblöcke in die zugehörigen Chiffretextblöcke verschlüsselt wurden.

Die Methode der Differentiellen Kryptoanalyse arbeitet mit vielen Paaren mit *derselben* „Differenz“.

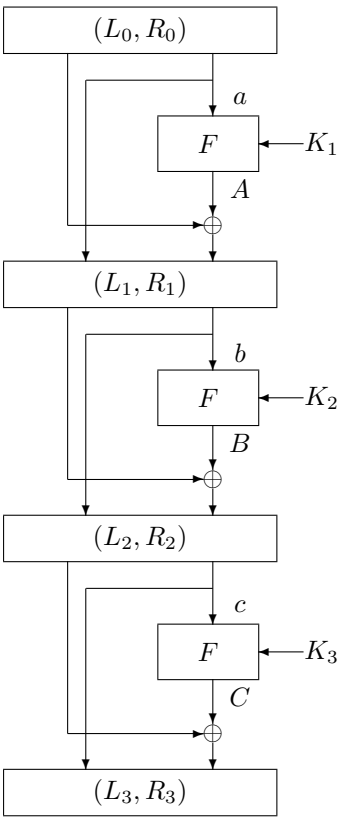
Für Feistel-Chiffren, auf deren Differentielle Kryptoanalyse im folgenden etwas genauer eingegangen wird, sind diese „Differenzen“ der XOR-Wert zweier Blöcke, Teilblöcke bzw. Register.

Bezeichnungen: Es sei X ein solcher Block bzw. Register. Es sei X^* ebenfalls ein solcher Block bzw. Register. Damit bilden X und X^* ein *Paar*. Der XOR-Wert (oder kurz „XOR“) von X und X^* soll gekennzeichnet werden mit X'. Es gilt also $X' = X \oplus X^*$. Eine mit $'$ ergänzte Variable soll also hier stets ein XOR bezeichnen!

Mit den Abbildungen werden die übrigen Bezeichungen klar.



Die Funktion F des VDES



VDES mit 3 Runden

Hilfsmittel: Es werden einige „XOR-Äquivanlenzen“ solcher Paare X und X^* bzw. Y und Y^* angegeben. Es bezeichne K einen Schlüssel und E die Expansionsabbildung:

$$\begin{aligned} \mathrm{E}(X) \oplus \mathrm{E}(X^*) &= \mathrm{E}(X \oplus X^*) \\ (X \oplus K) \oplus (X^* \oplus K) &= X \oplus X^* \\ (X \oplus Y) \oplus (X^* \oplus Y^*) &= (X \oplus X^*) \oplus (Y \oplus Y^*) \end{aligned}$$

Vorgehen: Es soll nun schrittweise eine Einführung in die Vorgehensweise erfolgen. Dabei wird jeder Sachverhalt zunächst formal beschrieben und dann in einem Beispiel illustriert.

Schließlich soll versucht werden, die bereits in Teilversuch 4 verwendete 3-Runden-Chiffre VDES mit $\mathrm{S} \equiv \mathrm{S}_{\text{nlin}}$ durch Differentielle Kryptoanalyse zu brechen.

[5] siehe auch [BS93]

[6] engl.: differences. Differenz ist hier eher im Sinne von „Unterschied“ gemeint, als im mathematischen Sinne.

Definition 8 Die *Differenzverteilungstabelle* (difference distribution table) zeigt die Verteilung der Input-XOR's und Output-XOR's aller möglichen Paare einer Substitutionsbox S. Jede Zeile entspricht einem bestimmten Input-XOR. Jede Spalte entspricht einem bestimmten Output-XOR. Die Tabelleneinträge geben die Anzahl möglicher Paare mit diesem Input-XOR und Output-XOR an. ‡

Beispiel 14 Die Tabelle gibt die Differenzverteilungstabelle der S-Box S_{nlin} aus VDES an. (Alle XOR's binär!)

Differenzverteilungstabelle von S_{nlin}

Input-XOR	Output-XOR			
	00	01	10	11
000	8	0	0	0
001	0	0	4	4
010	0	0	4	4
011	0	8	0	0
100	0	0	4	4
101	4	4	0	0
110	4	4	0	0
111	0	0	4	4

BSP

Definition 9 Seien X' und Y' Werte (z.B. Input- und Output-XOR's einer S-Box). Es gilt $X' \rightarrow Y'$, in Worten: *„X' kann Y' verursachen durch die S-Box S" (engl.: „X' may cause Y' by the S-box")*, wenn ein Paar existiert, dessen Input-XOR gleich X' und dessen Output-XOR gleich Y' ist. Existiert kein solches Paar, so schreibt man $X' \not\rightarrow Y'$. ‡

Beispiel 15 Gegeben sei S_{nlin} und deren Differenzverteilungstabelle (siehe Beispiel 14). Man findet beispielsweise (001) $\rightarrow$ (10) oder (001) $\rightarrow$ (11), das heißt, ein Input-XOR von (001) der S-box S_{nlin} kann ein Output-XOR von (10) verursachen, aber auch ein Output-XOR von (11). Man sieht auch (001) $\not\rightarrow$ (00), d.h., es existiert kein Input-Paar mit einem Input-XOR von (001), das ein Output-XOR von (00) verursacht. BSP

Definition 10 Seien X' und Y' Werte (z.B. Input- und Output-XOR's einer S-Box). Es gilt *„X' kann Y' verursachen mit der Wahrscheinlichkeit p durch die S-Box S"* mit

$$p = \frac{\text{Zahl der Paare mit } X' \rightarrow Y' \text{ (Tabelleneitrag)}}{2^n}$$

(n ist die Anzahl der Input-Bits der S-Box), wenn ein Paar existiert, dessen Input-XOR gleich X' und dessen Output-XOR gleich Y' ist. ‡

Beispiel 16 Gegeben sei S_{nlin} und deren Differenzverteilungstabelle (siehe Beispiel 14). Es gilt beispielsweise: Ein Input-XOR von (101) kann mit einer Wahrscheinlichkeit von $p = 0.5$ ein Output-XOR von (00) verursachen. Ein Input-XOR von (011) verursacht mit einer Wahrscheinlichkeit von $p = 1$ ein Output-XOR von (01). BSP

Mit Hilfe der Differenzverteilungstabelle ist es aus den vorhandenen Input- und Output-XOR's möglich, Kandidaten für die Belegungen der Input- und Output-Vektoren der S-Box zu finden oder in bestimmten Fällen sofort die Belegung der Vektoren.

Beispiel 17 Der Tabelleneintrag (010) $\rightarrow$ (10) hat den Wert 4, d.h., es existieren 4 Input-Paare mit einem Input-XOR von (010) und einem Output-XOR von (10). Die folgende Tabelle gibt die möglichen Input-Vektoren für ein Input-XOR von (010) an. Diese Inputvektoren können mit

Hilfe von Tabelle 4.1 auf Seite 28 ermittelt werden. Hierzu ermittelt man zunächst alle Paare von Input-Vektoren (X, X^*) für die $X \oplus X^* = (010)$ gilt. Anhand von Tabelle 4.1 werden nun die Output-Vektoren Y bzw. Y^* für die Input-Vektoren X bzw. X^* ermittelt. Entsprechend dem Output-XOR $Y \oplus Y^*$ werden die Input-Vektoren-Paare in die Tabelle eingeordnet.

für Input-XOR von (010)	
Output-XOR	mögliche Input-Vektoren
10	(100, 110), (110, 100), (101, 111), (111, 101)
11	(000, 010), (010, 000), (001, 011), (011, 001)

BSP

Im nächsten Schritt soll gezeigt werden, wie man unter Verwendung bekannter Input- und Output-Paare einer S-Box, die innerhalb der Funktion F einer Feistel-Chiffre verwendet wird, Schlüssel-Bits finden kann.

Beispiel 18 Es soll $S_{nlin}(Z)$ untersucht werden mit $Z = X \oplus K$. X, K und Z haben eine Blockbreite von 3 Bit. Gesucht ist K.

Es ist ein Paar $X = (001), X^* = (011)$ bekannt oder wird gewählt. Der Output-XOR von S_{nlin} für das Paar X, X^* und K sei (10). Zunächst gilt unter Verwendung der Äquivalenz $(X \oplus K) \oplus (X^* \oplus K) = X \oplus X^* = Z \oplus Z^* = X' = Z'$, daß der Input-XOR Z' von S_{nlin} unabhängig von K ist. Entsprechend der Differenzverteilungstabelle (siehe Beispiel 14) hat der Input von S_{nlin} 4 mögliche Belegungen für diesen Output-XOR von (10). Diese Belegungen sind aus der Tabelle in Beispiel 17 ablesbar. Dementsprechend existieren auch 4 mögliche Belegungen für den Schlüssel K, da $K = X \oplus Z$.

für Output-XOR von (10)		
Input (X, X*)	S-Box-Input Z	mögliche Schlüssel K
(001, 011)	(100, 110)	101
(001, 011)	(110, 100)	111
(001, 011)	(101, 111)	100
(001, 011)	(111, 101)	110

Der richtige Schlüssel muß in dieser Tabelle enthalten sein. Unter Verwendung weiterer Paare kann man weitere Kandidaten für K finden:

Es ist ein Paar $X = (101), X^* = (011)$ bekannt oder wird gewählt. Der Output-XOR von S_{nlin} für das Paar X, X^* und K sei (01). Die folgende Tabelle soll wieder die möglichen Input-Vektoren für ein Input-XOR von (110) darstellen:

für Input-XOR von (110)	
Output-XOR	mögl. Input-Vektoren Z
00	(001, 111), (010, 100)
01	(000, 110), (011, 101)

Da der Input-XOR der S-Box unabhängig von K ist, findet man nach $K = X \oplus Z$ folgende mögliche Schlüssel K:

für Output-XOR von (01)	
S-Box-Input Z	mögl. Schlüssel K
000, 110	101, 011
011, 101	110, 000

Der richtige Schlüssel muß in beiden Tabellen der möglichen Schlüsselkandidaten enthalten sein! Betrachtet man die beiden Tabellen mit den möglichen Schlüsselkandidaten, so sind in beiden Tabellen die Schlüsselkandidaten (101) und (110) enthalten. Einer dieser beiden Schlüssel ist der richtige Schlüssel K.

Da der Lösungsraum an dieser Stelle stark eingeschränkt ist, kann man durch Probieren den richtigen Schlüssel $K = (110)$ ermitteln. BSP

Erweiterung dieser Methode auf eine 3-Runden-Kryptoanalyse des Kryptosystems VDES:

Vorbemerkungen:

Zum besseren Verständnis sollte man eine Abbildung des VDES vor sich haben (siehe Seite 33)!

Man wählt $\mathcal{M}/\mathcal{C}$-Paare, hier bezeichnet mit $(L_0, R_0)/(L_3, R_3)$, mit bestimmten Eigenschaften:

Die $\mathcal{M}/\mathcal{C}$-Paare seien so gewählt, daß

$$a' = R'_0 = (00)$$

ist. Man kann sich überlegen, daß dann auch $S'_{E_a} = S_{E_a} \oplus S^*_{E_a} = \mathrm{E}(R_0) \oplus \mathrm{E}(R^*_0) = (000)$ ist. Wie bereits im vorigen Beispiel illustriert, ist der Input-XOR S_{I_a} der S-Box unabhängig vom Teilschlüssel K_1. Man kann also davon ausgehen, daß

$$A' = S'_{O_a} = \mathrm{S_{nlin}}(\mathrm{E}(R_0) \oplus K_1) \oplus \mathrm{S_{nlin}}(\mathrm{E}(R^*_0) \oplus K_1) = (00).$$

R_3 ergibt sich aus dem XOR-Wert der linken Hälfte des Klartextes, also L_0, dem Output der 1. Runde, also $A = \mathrm{S_{nlin}}(\mathrm{E}(R_0) \oplus K_1)$ und dem Output von F der 3. Runde, also $C = \mathrm{S_{nlin}}(\mathrm{E}(R_2) \oplus K_3)$: $R_3 = L_0 \oplus A \oplus C$.

Da sowohl die Klartext- und Chiffretext-XOR's als auch der Output-XOR von $\mathrm{S_{nlin}}$ der 1. Runde bekannt sind, kann man den Output-XOR von $\mathrm{S_{nlin}}$ der 3. Runde berechnen durch

$$C' = L'_0 \oplus R'_3 \oplus A'.$$

Das Paar $S_{E_c} = \mathrm{E}(L_3)$, $S^*_{E_c} = \mathrm{E}(L^*_3)$ der 3. Runde ist ermittelbar aus dem Chiffretextpaar, analog dem vorausgegangenen Beispiel.

Ebenso findet man eine Menge von Kandidaten für K_3, die unter Verwendung weiterer $\mathcal{M}/\mathcal{C}$-Paare eingeschränkt werden kann.

Damit ist der Schlüsselraum für die noch verbleibenden unklaren Schlüssel-Bits so stark eingeschränkt, daß z.B. durch Vollständiges Suchen in diesem – eingeschränkten – Schlüsselraum K ermittelt werden kann.

Analyse des VDES *mit 3 Runden:*

Der Angreifer kann Klartextblöcke selbst wählen und verschlüsseln (chosen-plaintext-attack). Der Hauptschlüssel K ist gesucht!

1. Ein Paar $\mathcal{M}, \mathcal{M}^*$ so wählen, daß $a' = R'_0 = (00)$:

$\mathcal{M}$	$=$	(0010)	$\mathcal{C}$	$=$	(1101)
$\mathcal{M}^*$	$=$	(0110)	$\mathcal{C}^*$	$=$	(0011)
$\mathcal{M}'$	$=$	(0100)	$\mathcal{C}'$	$=$	(1110)
	$=$	(L'_0, R'_0)		$=$	(L'_3, R'_3)

Da $a' = (00)$ folgt $A' = (00)$ (vgl. die Vorbemerkungen zu dieser Analyse).

2. $c' = L'_3 = (11)$ ist bekannt und $C' = L'_0 \oplus R'_3 \oplus A' = (01) \oplus (10) \oplus (00) = (11)$ wird berechnet.

3. Finden von Kandidaten für $K_3 = (k_2, k_3, k_4)$:
Man weiß:

c	$=$	(11)			
c^*	$=$	(00)			
c'	$=$	(11)	C'	$=$	(11)

Zunächst werden berechnet $S_{E_c} = \mathrm{E}(c) = (111)$ und $S^*_{E_c} = \mathrm{E}(c^*) = (000)$, woraus $S'_{E_c} = (111)$ folgt. Auch hier ist der Input-XOR S'_{I_c} von $\mathrm{S_{nlin}}$ unabhängig von K_3, also $S'_{I_c} = S'_{E_c} = (111)$. Entsprechend der Differnzverteilungstabelle (Bsp. 14) hat der Input S_{I_c} von $\mathrm{S_{nlin}}$ 4 mögliche Belegungen[7] für einen Input-XOR von $S'_{I_c} = (111)'$ und einen Output-XOR von $S'_{O_c} = C' = (11)$:

für Input-XOR von $S'_{I_c} = (111)$	
Output-XOR S'_O	mögliche Input-Vektoren S_{I_c}
10	(000, 111), (011, 100)
11	(001, 110), (010, 101)

Ausgehend von dieser Tabelle findet man nach $K_3 = S_{E_c} \oplus S_{I_c}$:

für Output-XOR von $S'_O = (11)$	
S-Box-Input S_{I_c}	mögliche Teilschlüssel K_i
001, 110	110, 001
010, 101	101, 010

Die Kandidaten für K_3 sind also $(110), (001), (101)$ und (010).

Mit einem weiteren Paar soll der Suchraum weiter eingeschränkt werden.

4. Ein weiteres Paar $\mathcal{M}, \mathcal{M}^*$ so wählen, daß $a' = R'_0 = (00)$:

$$\begin{array}{rclrcl} \mathcal{M} & = & (0010) & \mathcal{C} & = & (1101) \\ \mathcal{M}^* & = & (1110) & \mathcal{C}^* & = & (0110) \\ \hline (L'_0, R'_0) & = & (1100) & (L'_3, R'_3) & = & (1011) \end{array}$$

Es folgt aus $a' = (00)$ wieder $A' = (00)$.

5. $c' = L'_3 = (10)$ ist bekannt und $C' = L'_0 \oplus R'_3 \oplus A' = (11) \oplus (11) \oplus (00) = (00)$ wird berechnet.

6. Finden von weiteren Kandidaten für $K_3 = (k_2, k_3, k_4)$:
Berechnung von $S_{E_c} = \mathrm{E}(c) = (111)$ und $S^*_{E_c} = \mathrm{E}(c^*) = (010)$, woraus $S'_{E_c} = (101)$ folgt.

Die anderen Schritte sind analog 3.:
Wieder ist $S'_{I_c} = S'_{E_c}$.

für Input-XOR von $S'_{I_c} = (101)$	
Output-XOR S'_O	mögliche Input-Vektoren S_{I_c}
00	(000, 101), (011, 110)
01	(001, 100), (010, 111)

für Output-XOR von $S'_O = (00)$	
S-Box-Input S_{I_c}	mögliche Teilschlüssel K_i
000, 101	111, 010
011, 110	100, 001

Die hier ermittelten Kandidaten für K_3 sind also $(111), (010), (100)$ und (001).

7. Die Teilschlüssel

$$\begin{array}{rcll} K_3 & = & (k_2, k_3, k_4) & \\ & \stackrel{?}{=} & (0,1,0) & \text{oder} \\ & \stackrel{?}{=} & (0,0,1) & \end{array}$$

sind in beiden Tabellen enthalten.
Der Hauptschlüssel kann damit nur noch die Belegungen

$$\begin{array}{rcll} K & = & (k_1, k_2, k_3, k_4) & \\ & \stackrel{?}{=} & (0,0,1,0) & \text{oder} \\ & \stackrel{?}{=} & (1,0,1,0) & \text{oder} \end{array}$$

[7] siehe Beispiel 17 zur Ermittlung der Input-Vektoren S_{I_c} .

$$\stackrel{?}{=} (0,0,0,1) \quad \text{oder}$$
$$\stackrel{?}{=} (1,0,0,1) \quad \text{annehmen!}$$

Ergebnisse: Der Aufwand für Vollständiges Suchen ist an dieser Stelle um 75 % reduziert worden.

Aufgabe 10

Führen Sie anhand der oben angegebenen Vorgehensweise selbst eine differentielle Kryptoanalyse des VDES für folgende zwei Klartext-Schlüsseltext-Paare durch und geben sie die verbleibenden Schlüsselkandidaten an.

Erstes Klartext-Schlüsseltext-Paar:

$$\begin{array}{lcllcl} \mathcal{M} & = & (1011) & \mathcal{C} & = & (0001) \\ \mathcal{M}^* & = & (1111) & \mathcal{C}^* & = & (1001) \end{array}$$

Zweites Klartext-Schlüsseltext-Paar:

$$\begin{array}{lcllcl} \mathcal{M} & = & (1100) & \mathcal{C} & = & (1010) \\ \mathcal{M}^* & = & (0000) & \mathcal{C}^* & = & (0110) \end{array}$$

Vertiefende Literatur:

Biham, Eli; Shamir, Adi:
Differential cryptoanalysis of the Data Encryption Standard
Springer Verlag New York, Berlin, Heidelberg, 1993

Anhang A

Mathematische Prämissen*

Es seinen

		X, Y	Mengen
f	:	$X \to Y$	eine Abbildung
f^{-1}	:	$Y \to X$	die Umkehrabbildung zu f
		$x \in X$	
		$y, y' \in Y$	

Die Abbildung f ist

rechtseindeutig (auch: partiell)

$$\Longleftrightarrow \forall x.\forall y, y' : ((f(x) = y \wedge f(x) = y') \to (y = y'))$$

total

$$\Longleftrightarrow \forall x.\exists y : f(x) = y$$

surjektiv

$$\Longleftrightarrow \forall y.\exists x : f(x) = y$$

injektiv

$$\Longleftrightarrow f^{-1} \text{ ist rechtseindeutig}$$

bijektiv

$$\Longleftrightarrow f \text{ ist total, injektiv, surjektiv}$$

*siehe auch [HP85,S.8ff]

Anhang B

Abkürzungen und Symbole

$\mathsf{AM}(F)$	Abhängigkeitsmatrix einer Funktion F
BV	Bitverteilung
$\mathcal{C}$	Chiffretext-Vektor
d	Redundanz
$\mathbf{D}$	Entschlüsselungsfunktion
DES	Data Encryption Standard
E	expansion
$\mathbf{E}$	Verschlüsselungsfunktion
ECB	electronic-code-book-mode
F	eine Funktion
$\mathsf{HG}(X)$	Hamming-Gewicht eines Vektors X
l	Blockbreite des Schlüsselvektors K
L	ein Register
LSB	least significant bit
m	Blockbreite des Chiffretextvektors $\mathcal{C}$
$\mathcal{M}$	Klartext-Vektor
MFC	Methode des formalen Kodierens zum Brechen einer Chiffre
MR	Speicherplatzbedarf (memory requirement)
MSB	most significant bit
n	Blockbreite des Klartextvektors $\mathcal{M}$
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{P}$	Menge der Primzahlen
P	permutation
R	ein Register
S	eine Substitution
XOR	Exklusiv-Oder-Verknüpfung, bitweise-modulo-2-Addition $\oplus$

Anhang C

Literaturverzeichnis

[BA91] Beutelsbacher, Albrecht:
Kryptologie
Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, 1991

[BS93] Biham, Eli; Shamir, Adi:
Differential cryptoanalysis of the Data Encryption Standard
Springer Verlag New York, Berlin, Heidelberg, 1993

[Bro89] Bronstein, I.N.;Semendjajew, K.A.:
Taschenbuch der Mathematik
Verlag Nauka Moskau; BSB B.G.Teubner Verlagsgesellschaft Leipzig, 1989

[CM88] Clauß, Matthias; Fischer, Günther:
Programmieren mit C;
Berlin : Verlag Technik, 1988

[FH93] Federrath, Hannes:
Implementierung des DES zur Demonstration seiner kryptographischen Eigenschaften
Großer Beleg, TU Dresden, Fakultät Informatik, Institut für Theoretische Informatik, Lehrgebiet Informations- und Kodierungstheorie
eingereicht am 26. August 1993

[FW88] Fumy, Walter; Rieß, Hans Peter:
Kryptographie: Entwurf und Analyse symmetrischer Kryptosysteme
München; Wien : Oldenbourg, 1988

[HP85] Horster, Patrik:
Kryptologie;
Mannheim, Wien, Zürich : Bibliographisches Institut, 1985

[Kai85] Kaiser, Hans; Mlitz, R.; Zeilinger, G.:
Algebra für Informatiker
Wien; New York : Springer Verlag, 1985

[KA93] Klam, Andrea:
Erarbeitung und Implementierung eines Praktikumsversuches zur Problematik „Klassische Chiffrierverfahren“
Diplomarbeit, TU Dresden, Fakultät Informatik, Institut für Theoretische Informatik, Lehrgebiet Informations- und Kodierungstheorie
eingereicht am 30.04.1993

[KJ93] Kupferschmidt, Jens:
Untersuchung der Ausgangsfolge eines rückgekoppelten DES-Moduls auf seine statistischen Eigenschaften bei der Nutzung als Zufallszahlengenerator
Diplomarbeit, TU Dresden, Fakultät Informatik, Institut für Theoretische Informatik, Lehrgebiet Informations- und Kodierungstheorie
Leipzig, 26. Februar 1993

[MGB87] Mund, S.; Gollmann, D.; Beth, T.:
Some Remarks on the Cross Correlation Analysis of Pseudo Random Generators
Proc. of Eurocrypt ’87, Springer LNCS 304, S.25-36

MM82] Meyer, C.; Matyas, S.M.:
Cryptography - A New Dimension in Computer Data Security
(3rd printing) John Wiley & Sons, 1982

[NEWS92] newsgroup: sci.crypt
from: kempf@rhrk.uni-kl.de
Subject: DES an differential cryptoanalysis
Date: 23 Mar 1992

[PfAß83] Pfitzmann, A.; Aßmann, R.:
More efficient software implementations of (generalizend) DES
Computers & Security, 12 (1983) S.477-500

[Si84] Siegenthaler, T.:
Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Application, IEEE Trans. on Inf. Theory, 31 (1983), S.776-780.

[UNIX91] UNIX SYSTEM V RELEASE 4
Leitfaden für Programmierer: ANSI-C und Programmierwerkzeuge
München : Carl Hanser Verlag, 1991

[Ru86] Rueppel, R.A.:
Analysis and Design of Stream Ciphers
Springer Verlag Berlin, 1986

[SB82] Schaumüller-Bichl, Ingrid:
Zur Analyse des Data encryption standard und Synthese verwandter Chiffriersysteme
Wien : VWGÖ, 1982. Dissertationen der Johannes-Kepler-Universität Linz; 40