Prof. Dr. Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

Codierungstheorie
und
Kryptographie

Wintersemester 2008

Inhaltsverzeichnis

Kapitel 3

Fehlerkorrigierende Codes

3.1 Fehlertypen und Fehlerkorrektur

Bei der Übertragung der codierten Nachrichten können aufgrund technischer Defekte bzw. Störungen im Übertragungskanal Fehler auftreten. Daher sind in der Codierungstheorie die beiden folgenden Probleme zu lösen.

- Der Code wird so konstruiert, dass der Empfänger der Nachricht in der Lage ist, bei der Decodierung gewisse Übertragungsfehler zu bemerken. Codes dieser Art heißen fehlererkennend.
- Der Code wird so konstruiert, dass der Empfänger der Nachricht in der Lage ist, Korrekturen so vorzunehmen, dass eine korrekte Decodierung möglich ist. Codes mit dieser Eigenschaft heißen fehlerkorrigierend.

Bei den bisher betrachteten Codierungen von Objekten über (endlichen) Alphabeten ist die Erkennung von Übertragungsfehlern mittels der im Beweis von Satz 1.4 gegebenen Decodierung gesichert. Der decodierende Automat gibt uns eine Fehlermeldung, falls dass empfangene Wort nicht in C^+ liegt.

Schwieriger gestaltet sich die Fehlererkennung, falls die zugrundeliegende Menge von zu codierenden Objekten potentiell unendlich und nicht als Menge von Wörtern angesehen wird. Ein solches Beispiel ist die zur Identifikation von Büchern benutzte ISBN-Kennzeichnung (Internationale Standard-Buch-Nummer). Diese Nummer besteht aus vier Teilen,

- einer Ziffernfolge, die das Land kennzeichnet,
- einer Ziffernfolge, die den Verlag widerspiegelt,
- einer Ziffernfolge, die dem Buch entspricht, und
- einem Prüfsymbol aus $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{X}\}$,

die jeweils durch – getrennt werden und deren Gesamtlänge 13 ist. Z.B. ist die ISBN-Kennzeichnung des Buches [19] von Peter Sweeney zur Fehlererkennung und -korrektur

$$3\text{–}446\text{–}16439\text{–}1 \quad ,$$

wobei 3 für Deutschland, 446 für den Hanser-Verlag, 16436 für das angegebene Buch steht, während die in den USA (entspricht 0) bei Prentice-Hall International (entspricht 13) erschienene Originalausgabe durch

0–13–278706–7

gekennzeichnet ist.

Das Prüfsymbol wird dabei wie folgt ermittelt: Sei

$$x_{10}x_9x_8x_7x_6x_5x_4x_3x_2x_1$$

die Folge, die aus der ISBN entsteht, indem man die – fortlässt, so soll

$$\sum_{k=1}^{10} k \cdot x_k \equiv 0 \bmod 11 \tag{3.1}$$

gelten, wobei dem Symbol X der numerische Wert 10 entspricht (dieser ist notwendig, da es die Restklassen 0,1,2,3,4,5,6,7,8,9,10 zu 11 gibt und bei Verwendung von 10 die Länge 13 der ISBN nicht einhaltbar wäre). Damit ergibt sich das Prüfsymbol x_1 als

$$x_1 = -\sum_{k=2}^{10} k \cdot x_k \bmod 11.$$

Wird nun eine ISBN-Kennzeichnung übermittelt, so kann der Empfänger mittels der Folgenlänge und (3.1) überprüfen, ob die empfangene Folge einem Buch entspricht. Stellt der Empfänger die Folgenlänge 13 und die Gültigkeit von (3.1) fest, so ist – mit hoher Wahrscheinlichkeit – die Folge korrekt übermittelt worden. Ist dagegen z.B. die Folge zu kurz, so ist bei der Übertragung mindestens ein Symbol verlorengegangen, bzw. bei Ungültigkeit von (3.1) mindestens ein Symbol nicht korrekt übertragen worden. Durch die Einführung des Prüfsymbols sind wir zwar in der Lage zu erkennen, ob das übermittelte Wort eine ISBN-Kennzeichnung sein kann, aber wir können bei Auftreten eines Übertragungsfehlers nicht ermitteln, welcher Fehler vorliegt. Z.B. entsteht die Folge

$$\text{S} = 3\text{–}446\text{–}16419\text{–}1$$

aus der oben gegebenen ISBN-Kennzeichnung von [19], indem als x_3 nicht 3 sondern 1 übermittelt wurde. Für diese fehlerhafte Folge ergibt sich der Wert

$$10 \cdot 3 + 9 \cdot 4 + 8 \cdot 4 + 7 \cdot 6 + 6 \cdot 1 + 5 \cdot 6 + 4 \cdot 4 + 3 \cdot 1 + 2 \cdot 9 + 1 \cdot 1 = 214 \equiv 5 \bmod 11.$$

Aus dieser Folge S erhalten wir durch Änderung von x_4 um 1 die Folge

$$3\text{–}446\text{–}26419\text{–}1 \quad ,$$

die ebenfalls der Bedingung (3.1) genügt und daher eine ISBN-Kennzeichnung sein kann. Somit ist nicht klar, wie die Folge S richtig zu decodieren ist.

Als Zweites betrachten wir den Blockcode

$$C = \{11000, 10110, 01101\}$$

und nehmen an, dass bei der Übertragung nur Fehler auftreten, bei denen anstelle einer 1 eine 0 bzw. anstelle einer 0 eine 1 übermittelt wird. Die Länge des Wortes wird daher nicht

verändert und durch stückweisen Vergleich mit den Codewörtern (siehe die Ausführungen vor von Satz 1.4) ist leicht feststellbar, ob die Übertragung korrekt erfolgte.

Wir wollen nun annehmen, dass bei der Übertragung des Codewortes 11000 ein Fehler an der dritten Stelle eingetreten ist, d.h. es wird 11100 empfangen. Es ist nun leicht zu sehen, dass durch Änderung der anderen Codewörter von C an *einer* beliebigen Stelle immer ein Wort entsteht, dass von 11100 verschieden ist. Gehen wir davon aus, dass bei der Übertragung maximal ein Fehler gemacht wurde, so ist klar, dass beim Empfang von 11100 das Wort 11000 gesendet wurde. Hieraus folgt, dass C dem Empfänger die Möglichkeit gibt, diesen Fehler zu korrigieren. Wir werden unten nachweisen, dass C ein Code mit der Möglichkeit zur Korrektur eines jeden Fehlers der eben behandelten Art ist.

In diesem Abschnitt wollen wir unserer Betrachtungen auf Blockcodes über dem Alphabet $\{0,1\}$ und auf die Korrektur von Fehlern bei der Übertragung von Codewörtern beschränken (bei den hauptsächlich betrachteten Fehlern lässt sich die Methode zur Korrektur auch auf beliebige Nachrichten übertragen, da die Länge des empfangenen Codewortes durch den Blockcode vorgegeben ist; ansonsten verwenden wir ein Trennzeichen zur Trennung der Codewörter).

Wir betrachten die folgenden Typen von Fehlern.

Definition 3.1 *Unter einem Austauschfehler verstehen wir die Übertragung einer* 0 *anstelle einer* 1 *bzw. die Übertragung einer* 1 *anstelle einer* 0.

Unter einem Ausfallfehler verstehen wir den Ausfall eines Symbols während der Übertragung, d.h. an einer Stelle wird das übertragene Wort durch Löschen eines Buchstaben gekürzt.

Unter einem Einschubfehler verstehen wir die Übertragung eines zusätzlichen Symbols, d.h. das empfangene Wort wird durch den Einschub eines Symbols an einer Stelle im übertragenen Wort verlängert.

Wir bezeichnen diese Typen von Fehlern durch

$$1 \to 0,\ 0 \to 1,\ 0 \to \lambda,\ 1 \to \lambda,\ \lambda \to 0,\ \lambda \to 1.$$

Sei G die Menge aller dieser Fehler.

Definition 3.2 *Eine Teilmenge von G bezeichnen wir als Fehlertyp. Ein Fehlertyp F heißt symmetrisch, falls F durch Vereinigung aus den folgenden Mengen $\{0 \to 1, 1 \to 0\}$, $\{\lambda \to 0, 0 \to \lambda\}$ und $\{\lambda \to 1, 1 \to \lambda\}$ gewonnen werden kann.*

Bei einem symmetrischen Fehlertyp enthält F mit einem Fehler f auch den Fehler g, durch den das ursprüngliche Wort wieder gewonnen werden kann.

Für einen Fehlertyp F und Wörter w und v über $\{0,1\}$ setzen wir

$$w \overset{F,t}{\Longrightarrow} v,$$

falls bei der Übertragung durch das simultane Auftreten von t Fehlern aus F aus dem Wort w das Wort v entsteht. Offenbar gilt $w \overset{F,t}{\Longrightarrow} v$ genau dann, wenn es Wörter $w_1, w_2, \ldots, w_{t-1}$ so gibt, dass

$$w = w_0 \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_2 \overset{F,1}{\Longrightarrow} \ldots \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} w_t = v \tag{3.2}$$

gilt.

Beispiel 3.1 Sei

$$F = \{0 \to 1, 1 \to 0, \lambda \to 1, 0 \to \lambda\}.$$

Dann gelten

- $0011 \overset{F,1}{\Longrightarrow} 0111$
  (Auftreten des Fehlers $0 \to 1$ an zweiter Stelle),
- $0011 \overset{F,2}{\Longrightarrow} 1001$
  (bei Auftreten des Fehlers $0 \to 1$ an erster und $1 \to 0$ an dritter Stelle oder durch Ausfall einer 1 und Einschub einer 1 an erster Stelle),
- $0011 \overset{F,1}{\Longrightarrow} 00111$
  (bei Auftreten des Fehlers $\lambda \to 1$ zwischen zweiter und dritter Stelle),
- $0011 \overset{F,3}{\Longrightarrow} 10$
  (bei Ausfall des ersten und zweiten Buchstaben entsprechend Fehler $0 \to \lambda$ und Austausch des Symbols an vierter Stelle entsprechend Fehler $1 \to 0$).

Wir führen nun den zentralen Begriff dieses Abschnitts ein.

Definition 3.3 *Sei F ein Fehlertyp und C ein Blockcode über $\{0,1\}$. C heißt Code mit Korrektur von s Fehlern aus F, falls für jedes Wort $v \in \{0,1\}^*$ höchstens ein Wort $w \in C$ mit $w \overset{F,t}{\Longrightarrow} v$ und $t \leq s$ existiert.*

Wir bemerken, dass bei einer Übertragung von Wörtern des Codes C, bei der höchstens s Fehler auftreten, nur Wörter v empfangen werden können, für die $w \overset{F,t}{\Longrightarrow} v$ mit $w \in C$ und $t \leq s$ gilt. Ist C ein Code mit Korrektur von s Fehlern, so kann dem empfangenen Wort v eindeutig ein Codewort w zugeordnet werden, das übertragen werden sollte.

Besteht F nur aus Austauschfehlern, so kann offensichtlich bei der Übertragung die Länge der Wörter nicht verändert werden. Daher gibt es für Wörter v, deren Länge von der des Blockcodes verschieden ist, kein Wort $w \in C$ mit $w \overset{F,t}{\Longrightarrow} v$. Somit ist gerechtfertigt, dass in der Definition eines fehlerkorrigierenden Codes die Existenz von *höchstens* einem $w \in C$ zu v gefordert wurde.

Die Definition fehlerkorrigierender Codes ist nicht effektiv in dem Sinn, dass aus ihr direkt ein Algorithmus folgt, mittels dessen entschieden werden kann, ob ein Code s Fehler korrigieren kann. Wir streben nun ein solches Kriterium an. Dazu führen wir folgende Begriffe ein.

Für einen Fehlertyp F und Wörter $w, v \in \{0,1\}^*$ definieren wir

$$d_F(w,v) = \begin{cases} \min\{t : w \overset{F,t}{\Longrightarrow} v\} & \text{falls dies existiert} \\ \infty & \text{sonst} \end{cases}.$$

Offenbar gilt $d_F(w,v) = \infty$ genau dann, wenn w mittels Fehlern aus F nicht in v überführt werden kann.

Sei

$$H = \{0 \to 1,\ 1 \to 0\}$$

der Fehlertyp, der aus den beiden Austauschfehlern besteht. Für H und zwei Wörter

$$w = x_1 x_2 \dots x_n \quad \text{und} \quad v = y_1 y_2 \dots y_m$$

ergibt sich

$$d_H(w,v) = \begin{cases} \#(\{i : x_i \neq y_i\}) & n = m \\ \infty & \text{sonst} \end{cases}. \tag{3.3}$$

Die Funktion D_H wird Hamming-Abstand genannt.

Satz 3.1 *Für einen symmetrischen Fehlertyp F ist durch d_F eine Abstandsfunktion in $\{0,1\}^*$ definiert.*

Beweis. Wir zeigen die drei Eigenschaften einer Abstandsfunktion:

1. $d_F(w,v) = 0$ gilt genau dann, wenn $w = v$ ist.

   $d_F(w,v) = 0$ gilt genau dann, wenn kein Fehler vorliegt. Dies ist offensichtlich gleichwertig zu $w = v$.

2. $d_F(w,v) = d_F(v,w)$ gilt für beliebige $w, v \in F$.

   Da F symmetrisch ist, folgt, dass entweder beide Werte unendlich oder beide Werte endlich sind.

   Für den Fall, dass beide unendlich sind, gilt die Behauptung offenbar.

   Sei $d_F(w,v) = t$. Dann gilt

   $$w = w_0 \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_2 \overset{F,1}{\Longrightarrow} \dots \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} w_t = v$$

   für gewisse $w_1, w_2, \dots, w_{t-1}$. Da F ein symmetrischer Fehlertyp ist, erhalten wir auch die Beziehung

   $$v = w_t \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} w_{t-2} \overset{F,1}{\Longrightarrow} \dots \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_0 = w,$$

   woraus nach Definition

   $$d_F(v,w) \leq t = d_F(w,v)$$

   folgt. Analog kann man auch $d_F(w,v) \leq d_F(v,w)$ zeigen, womit die Gleichheit folgt.

3. $d_F(w,v) + d_F(v,z) \geq d_F(w,z)$ gilt für alle $w, v, z \in \{0,1\}^*$.

   Ist einer der Werte $d_F(w,v)$ oder $d_F(v,z)$ unendlich, so ist die Behauptung offenbar gültig.

   Seien daher $d_F(w,v) = t$ und $d_F(v,z) = s$. Dann gibt es Wörter $w_1, w_2, \dots, w_{t-1}$, $v_1, v_2, \dots, v_{s-1}$ mit

   $$w = \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_2 \overset{F,1}{\Longrightarrow} \dots \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} = v$$

   und

   $$v \overset{F,1}{\Longrightarrow} v_1 \overset{F,1}{\Longrightarrow} v_2 \overset{F,1}{\Longrightarrow} \dots \overset{F,1}{\Longrightarrow} v_{s-1} \overset{F,1}{\Longrightarrow} z.$$

Somit erhalten wir

$$w \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_2 \overset{F,1}{\Longrightarrow} \ldots \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} v \overset{F,1}{\Longrightarrow} v_1 \overset{F,1}{\Longrightarrow} v_2 \overset{F,1}{\Longrightarrow} \ldots \overset{F,1}{\Longrightarrow} v_{s-1} \overset{F,1}{\Longrightarrow} z,$$

woraus

$$d_F(w,z) \leq t + s = d_F(w,v) + d_F(v,z)$$

resultiert. □

Definition 3.4 *Für einen symmetrischen Fehlertyp F und einen endlichen Code C definieren wir den Codeabstand $d_F(C)$ als*

$$d_F(C) = \min\{d_F(x,y) : x,y \in C, x \neq y\}.$$

Wir charakterisieren nun die Fähigkeit, s Fehler zu korrigieren durch den Codeabstand.

Satz 3.2 *Sei F ein symmetrischer Fehlertyp. Dann ist ein endlicher Code C genau dann ein Code mit Korrektur von s Fehlern aus F, wenn*

$$d_F(C) \geq 2s + 1$$

gilt.

Beweis. Sei C ein Code mit $d_F(C) \leq 2s$. Dann gibt es eine natürliche Zahl $t \leq 2s$, Codewörter $x, y \in C$ mit $x \neq y$ und Wörter $w_1, w_2, \ldots, w_{t-1}$ so, dass

$$x \overset{F,1}{\Longrightarrow} w_1 \overset{F,1}{\Longrightarrow} w_2 \overset{F,1}{\Longrightarrow} \ldots \overset{F,1}{\Longrightarrow} w_{t-1} \overset{F,1}{\Longrightarrow} y$$

gilt. Für $w = w_{\lceil t/2 \rceil}$ erhalten wir

$$d_F(x,w) = r_1 \leq s \quad \text{und} \quad d_F(y,w) = r_2 \leq s$$

und damit

$$x \overset{F,r_1}{\Longrightarrow} w \quad \text{und} \quad y \overset{F,r_2}{\Longrightarrow} w \tag{3.4}$$

mit $r_1 \leq s$ und $r_2 \leq s$, womit C kein Code mit Korrektur von s Fehlern aus F sein kann.

Ist umgekehrt C kein Code mit Korrektur von s Fehlern aus F, so gibt es ein Wort $w \in \{0,1\}^*$, Codewörter $x, y \in C$ mit $x \neq y$ und Zahlen $r_1 \leq s$ und $r_2 \leq s$ derart, dass (3.4) erfüllt ist. Deshalb gilt

$$d_F(x,y) \leq d_F(x,w) + d_F(w,y) \leq r_1 + r_2 \leq 2s,$$

womit

$$d_F(C) \leq 2s$$

nachgewiesen ist. □

Beispiel 3.2 Für den oben betrachteten Blockcode

$$C = \{11000, 10110, 01101\}$$

und den Fehlertyp H, der aus den beiden Austauschfehlern $0 \to 1$ und $1 \to 0$ besteht, ergibt sich wegen (3.3) $d_H(w,v) = 3$ für jedes Paar $x, y \in C$ und somit

$$d_H(C) = 3.$$

Daher ist C ein Code mit Korrektur eines Austauschfehlers.

3.2 Beispiele für fehlerkorrigierende Codes

Mittels des Satzes 3.2 kann zu einem gegebenen Code die Anzahl s der korregierbaren symmetrischen Fehler bestimmt werden. Der Satz liefert aber keine Möglichkeit, den oder die Fehler zu korregieren. Hierfür steht uns entsprechend den Definitionen bisher nur der folgende Algorithmus zur Verfügung. Wir bilden den Abstand zwischen dem empfangenen Wort und den Codewörtern und wissen dann, dass das Codewort gesendet wurde, bei dem dieser Abstand höchstens s beträgt. Wir wollen nun einige spezielle Codes betrachten, bei denen die Stellen, an denen der Fehler aufgetreten ist, direkt ermittelt werden können.

a) HAMMING-*Codes zur Korrektur eines Austauschfehlers*

Sei n eine beliebige positive natürliche Zahl. Wir setzen $l = \lfloor log_2(n) \rfloor + 1$, d.h. es gilt $2^{l-1} \leq n < 2^l$, und bezeichnen für $1 \leq i \leq n$ mit $e_l(i)$ den Vektor $(u_1, u_2, \ldots, u_l) \in \{0,1\}^l$, für den $u_1 u_2 \ldots u_l$ die Binärdarstellung von i ist. Für ein beliebiges Wort $X = x_1 x_2 \ldots x_n \in \{0,1\}^n$ setzen wir

$$H(X) = \sum_{i=1}^{n} x_i \cdot e_l(i)\,,$$

wobei die Addition der Vektoren komponentenweise mod 2 erfolgt. $H(X)$ ist also ein Vektor aus $\{0,1\}^l$. Wir definieren nun den HAMMING-Code H_n durch

$$H_n = \{X \mid X \in \{0,1\}^n, H(X) = (0,0,\ldots,0)\}\,.$$

Nach dieser Definition können die Elemente von H_n als Lösungen eines homogenen Gleichungssystems mit n Unbekannten und l Gleichungen aufgefasst werden. Beachten wir noch, dass die Vektoren $e_l(2^i)$, $0 \leq i \leq l-1$, jeweils genau eine Eins enthalten und paarweise verschieden sind, so ist klar dass die von ihnen gebildete Matrix die Einheitsmatrix $E_{l,l}$ ist und dass damit das Gleichungssystem den Rang l besitzt. Damit ergibt sich, dass die Elemente des HAMMING-Codes einen $n-l$-dimensionalen Vektorraum über dem Körper $\{0,1\}$ bilden. Folglich hat H_n genau 2^{n-l} Elemente. Wegen $l = \lfloor log_2(n) \rfloor + 1$ erhalten wir

$$\frac{2^{n-1}}{n} = \frac{2^{n-1}}{2^{\log_2(n)}} = 2^{n-1-log_2(n)} \leq 2^{n-l} \leq 2^{n-log_2(n+1)} = \frac{2^n}{2^{\log_2(n+1)}} = \frac{2^n}{n+1}$$

und damit

$$\frac{2^{n-1}}{n} \leq \#(H_n) = 2^{n-(\lfloor log_2(n) \rfloor + 1)} \leq \frac{2^n}{n+1}\,.$$

Wir geben nun eine Methode zur Berechnung der Elemente von H_n an. Wir setzen

$$M = \{1,2,\ldots,n\} \setminus \{2^i \mid 0 \leq i \leq l-1\}\,.$$

Dann können wir den HAMMING-Code dadurch bestimmen, dass wir die Werte x_j mit $j \in M$, beliebig in $\{0,1\}$ wählen, und dann die Werte x_{2^i}, $0 \leq i \leq l-1$, entsprechend dem Gleichungssystem berechnen. Jedoch ist die Lösung des Gleichungssystems sehr einfach, denn wegen

$$(0,0,\ldots,0) = \sum_{k=0}^{n} x_k e_k(l) = \sum_{i=0}^{l-1} x_{2^i} e_l(2^i) + \sum_{j \in M} x_j e_l(j)$$

ergibt sich

$$\sum_{i=0}^{l-1} x_{2^i} e_l(2^i) = \sum_{j \in M} x_j e_l(j)\,.$$

Setzen wir

$$(u_l, u_{l-1}, \ldots u_1) = \sum_{i \in M} x_i e_l(i)\,.$$

und beachten, dass die Vektoren $e_l(2^i)$ die Einheitsmatrix bilden, so muss

$$x^{2^i} = u_{i+1}$$

gelten.

Beispiel 3.3 Sei $n = 6$. Dann ergibt sich $l = 3$. Der *Hamming*-Code H_6 besteht also aus $2^{n-l} = 2^3 = 8$ Elementen. Um ein Wort $X = x_1x_2x_3x_4x_5x_6$ aus H_6 zu berechnen, können wir die Werte x_3, x_5 und x_6 frei wählen und bestimmen dann x_1, x_2 und x_4. Es ergibt sich die folgende Tabelle

x_3	x_5	x_6	$x_3e_3(3) + x_5e_3(5) + x_6e_3(6)$	x_1	x_2	x_4	X
0	0	0	$(0,0,0)$	0	0	0	000000
0	0	1	$(1,1,0)$	0	1	1	010101
0	1	0	$(1,0,1)$	1	0	1	100110
0	1	1	$(0,1,1)$	1	1	0	110011
1	0	0	$(0,1,1)$	1	1	0	111000
1	0	1	$(1,0,1)$	1	0	1	101101
1	1	0	$(1,1,0)$	0	1	1	011110
1	1	1	$(0,0,0)$	0	0	0	001011

und damit

$$H_6 = \{000000, 010101, 100110, 110011, 111000, 101101, 011110, 001011\}\,.$$

Sei bei der Übertragung von $X = x_1x_2 \ldots x_n$, $x_i \in \{0,1\}$ für $1 \leq i \leq n$ ein Austauschfehler – sagen wir an der j-ten Stelle – aufgetreten. Dann ist das von uns empfangene Wort $Y = x_1x_2 \ldots x_{j-1}(x_j \oplus 1)x_{j+1}x_{j+2} \ldots x_n$. Weiterhin gilt

$$H(Y) = \sum_{i=1}^{j-1} x_i e_l(i) \oplus (x_j \oplus 1)e_l(j) \oplus \sum_{i=j+1}^{n} x_i e_l(i) = e_l(j) \oplus \sum_{i=1}^{n} x_i e_i(l) = e_l(j) \oplus H(X)\,.$$

Wenn wir nun annehmen, dass $X \in H_n$ gilt, so ist $H(X)$ der l-dimensionale Nullvektor, und es ergibt sich

$$H(Y) = e_l(j)\,.$$

Interpretieren wir $e_l(j)$ als eine Dualzahl, so erhalten wir nach Definition j und damit die Stelle, an der der Austauschfehler vorliegt.

Beispiel 3.3 (Fortsetzung) Sei $Y = 010111$ das empfangene Wort. Wegen $Y \notin H_6$, kann nicht Y gesendet worden sein. Wir nehmen nun an, dass Y durch einen Austauschfehler während der Übertragung entstanden ist. Wegen

$$H(Y) = (0,1,0) \oplus (1,0,0) \oplus (1,0,1) \oplus (1,1,0) = (1,0,1)$$

muss der Fehler an der fünften Stelle vorliegen, da 101 die Dualdarstellung von 5 ist. Es muss also $X = 010101$ gesendet worden sein.

Wir können natürlich keine Aussage über das gesendete Wort treffen, falls sogar zwei (oder mehr) Fehler zugelassen sind. Y kann dann sowohl durch einen Fehler an der fünften Stelle aus $010101 \in H_6$ als auch durch Fehler an der dritten und sechsten Stelle aus $011110 \in H_6$ entstanden sein.

b) Codes zur Korrektur eines Fehlers vom Typ $\{0 \to 1\}$

Seien n und k zwei beliebige positive natürliche Zahlen. Für ein Wort $X = x_1x_2 \dots x_n \in \{0,1\}^n$ setzen wir

$$W(X) = \sum_{i=1}^{n} x_i \cdot i = x_1 + 2x_2 + 3x_3 + \dots + nx_n \,.$$

Nach Definition ist $W(X)$ die Summe der Indizes, bei denen der Buchstabe im Wort eine 1 ist. Ferner setzen wir

$$W_{n,k} = \{X \mid X \in \{0,1\}^n,\ W(X) = 0 \bmod k\}.$$

Beispiel 3.4 Sei $n = 6$. Wir betrachten die Wörter

$$X = 110100, \quad Y = 010101, \quad Z = 010010, \quad Z' = 110011 \,.$$

Dann gelten wegen

$$W(X) = 1{+}2{+}4 = 7,\ W(Y) = 2{+}4{+}6 = 12,\ W(Z) = 2{+}5 = 7,\ W(Z') = 1{+}2{+}5{+}6 = 14$$

die Beziehungen

$$X \in W_{6,7}, Y \notin W_{6,7}, Z \in W_{6,7}, Z' \in W_{6,7} \text{ und } X \notin W_{6,12}, Y \in W_{6,12}, Z \notin W_{6,12}, Z' \notin W_{6,12} \,.$$

Wir zeigen nun, dass jeder Code $W_{n,k}$ mit $k \geq n+1$ ein Code mit Korrektur von einem Fehler vom Typ $\{0 \to 1\}$ ist. Sei $X = x_1x_2 \dots x_n$ ein Wort aus $W_{n,k}$ und entstehe Y aus X durch einen Fehler vom Typ $\{0 \to 1\}$ an der Stelle j. Dann gelten $x_j = 0$ und $Y = x_1x_2 \dots x_{j-1}1x_{j+1}x_{j+2} \dots x_n$ und daher auch

$$W(Y) = W(X) + j \,.$$

Wegen $k \geq n+1$ und $j \leq n$ folgt

$$W(Y) = W(X) + j = 0 + j = j \bmod k \,.$$

Somit gibt der Wert $W(Y)$ direkt die Stelle an, an der der Fehler aufgetreten ist.

Wir bemerken, dass $W_{n,k}$ für $k \geq n+1$ nicht unbedingt auch ein Code mit Korrektur eines Fehlers vom Typ $\{1 \to 0\}$ ist. Dies ist wie folgt einzusehen: ersetzt man an der j-ten Stelle im Wort $X = x_1x_2 \dots x_n$ den Buchstaben $x_j = 1$ durch eine Null, so ergibt sich $W(X) - W(Y) = -j$. Ist X ein Element des Codes $W_{n,k}$, so ergibt sich

$$W(X) - W(Y) = k - j \bmod k \,.$$

Wir betrachten nun den Fall $n = 6$ und $k = 7$ und das empfangene Wort $V = 110010$. Dann gilt $W(V) = 1 \bmod 7$. Entsprechend obigem bedeutet dies, dass dies durch die Änderung $0 \to 1$ an der ersten Stelle oder durch die Änderung $1 \to 0$ an der Stelle $k - 1 = 6$ entstanden sein. Im ersten Fall wurde $Z = 010010 \in W_{6,7}$ und im zweiten Fall $Z' = 110011 \in W_{6,7}$ (siehe Beispiel 3.4) gesendet.

Sei nun $k \geq 2n$. Für $1 \leq l \leq n$ gilt dann $l \leq n < k - l$, Haben wir das Wort Y mit $W(Y) = j \bmod k$ empfangen, so liegt bei $j \leq n$ ein Fehler vom Typ $\{0 \to 1\}$ an der j-ten Stelle vor und bei $n < j = k - l$ liegt ein Fehler vom Typ $\{1 \to 0\}$ an der l-ten Stelle vor. Folglich ist $W_{n,k}$ für $k \geq 2n$ sogar ein Code mit Korrektur eines Austauschfehlers.

c) Codes zur Korrektur eines Ausfallfehlers

Wir wollen jetzt zeigen, dass die Codes $W_{n,k}$ mit $k \geq n + 1$ auch zur Korrektur eines Ausfallfehlers geeignet sind.

Sei $X = x_1 x_2 \dots x_n$ ein Wort aus $W_{n,k}$ und entstehe Y aus X durch einen Ausfallfehler an der Stelle j. Dann gilt $Y = x_1 x_2 \dots x_{j-1} x_{j+1} x_{j+2} \dots x_n$. Wir bezeichnen mit n_1 bzw n_0 die Anzahl der Einsen bzw. Nullen, die rechts vom ausgefallenen Symbol x_j stehen, d.h. $n_i = \#_i(x_{j+1} x_{j+2} \dots x_n)$, $i \in \{0, 1\}$. Offenbar gilt

$$n_0 + n_1 = n - j \,. \tag{3.5}$$

Wir betrachten zuerst den Fall $x_j = 0$. Da die Werte x_k, $j+1 \leq k \leq n$, in Y gegenüber X um eine Stelle nach vorn gerückt wurden, erhalten wir

$$W(Y) = \sum_{i=1}^{j-1} x_i \cdot i + \sum_{i=j+1}^{n} x_i (i-1) \,.$$

Aus $x_j = 0$ und der Definition von $W(X)$ folgt nun

$$W(X) - W(Y) = \sum_{i=j+1}^{n} x_i = n_1 \,.$$

Offenbar gelten $\#_1(x_{j+1} x_{j+2} \dots x_n) = n_1 \leq \#_1(X)$. Da eine Null ausgefallen ist, haben wir auch noch $\#_1(X) = \#_1(Y)$. Somit erhalten wir

$$W(X) - W(Y) = n_1 \leq \#_1(Y) \,.$$

Sei nun $x_j = 1$. Dann trägt x_j bei de Berechnung von $W(X)$ den Wert j bei, der bei Y entfällt. Daher ergibt sich

$$W(X) - W(Y) = j + n_1 = n - n_0 \,,$$

wobei die letzte Beziehung aus (3.5) folgt. Weiterhin gilt $\#_1(X) \leq j + n_1 = n - n_0$. Da eine 1 ausgefallen ist, folgt $\#_1(Y) = \#_1(X) - 1$ und damit

$$W(X) - W(Y) = n - n_0 > \#_1(Y) \,.$$

Wegen $W(X) = 0 \bmod k$ gilt $W(X) - W(Y) = -W(Y) \bmod k$. Damit können wir bei gegebenem Y den Wert $W(X) - W(Y)$ berechnen. Diesen vergleichen wir mit $\#_1(Y)$.

Gilt $W(X) - W(Y) \leq \#_1(Y)$, so muss nach obigem eine Null ausgefallen sein, und wir fügen eine Null derart ein, dass hinter ihr noch $n_1 = W(X) - W(Y)$ Einsen stehen.

Gilt dagegen $W(X) - W(Y) > \#_1(Y)$, so muss nach obigem eine Eins ausgefallen sein, und wir fügen eine Eins derart ein, dass hinter ihr noch n_0 Nullen stehen, wobei sich n_0 aus $W(X) - W(Y) = n - n_0$ ergibt.

Beispiel 3.5 Wir betrachten den Code $W_{6,7}$, d.h. $n = 6$ und $k = 7$. Sei das Wort $Y = 10100$ empfangen worden. Da Y nur die Länge 5 hat, muss ein Symbol bei der Übertragung ausgefallen sein. Wir berechnen nun zuerst $W(Y) = 1+3 = 4$. Damit ergibt sich $W(X) - W(Y) = 3$. Ferner haben wir $\#_1(Y) = 2$. Wegen $3 > 2$, muss also eine Eins ausgefallen sein. Ferner gilt $n_0 = n - (W(X) - W(Y)) = 6 - 3 = 3$. Daher muss die Eins so eingefügt werden, dass hinter ihr noch drei Nullen stehen. Damit ergibt sich $X = 110100$ (und nach Beispiel 3.4 liegt X in $W_{6,7}$). Dabei ist es egal, ob wir die zusätzlich Eins in Y vor oder hinter der Eins am Beginn von Y einfügen.

Wir bemerken, dass mit analogen Betrachtungen gezeigt werden kann, dass $W_{n,k}$ für $k \geq n + 1$ auch ein Code mit Korrektur eines Einschubfehlers ist.

d) Codes zur Korrektur eines Fehlers vom Typ $\{1 \to 0, 0 \to 1, 0 \to \lambda, 1 \to \lambda, \lambda \to 0, \lambda \to 1\}$

Wir betrachten den Code $W_{n,k}$ mit $k \geq 2n$. Für das empfangene Wort Y unterscheiden wir drei Fälle.

Fall 1. Y hat die Länge n. Gilt $W(Y) = 0 \bmod k$, so ist Y ein Element von $W_{n,k}$ und es ist kein Fehler bei der Übertragung aufgetreten. Ist dagegen $W(Y) \neq 0 \bmod k$, so ist ein Austauschfehler aufgetreten (da sich die Länge des Wortes bei der Übertragung nicht geändert hat). Daher sind wir in der Lage das gesendete Wort zu ermitteln, wenn genau ein Austauschfehler aufgetreten ist, wie am Ende des Abschnitts b) gezeigt wurde.

Fall2. Y habe die Länge $n - 1$. Dann muss ein Ausfallfehler aufgetreten sein, und wir können das gesendete Wort wie in Abschnitt c) gezeigt ermitteln.

Fall 3. Y habe die Länge $n + 1$. Dann muss ein Einschubfehler aufgetreten sein, den wir korrigieren können, da nach der Bemerkung am Ende von Abschnitt c) $W_{n,k}$ ein Code mit Korrektur eines Einschubfehlers ist.

Ist also genau ein Fehler aufgetreten, so sind wir in der Lage zuerst festzustellen, ob es ein Austausch-, Ausfall- oder Einschubfehler ist und diesen dann zu korrigieren. Damit ist $W_{n,k}$ für $k \geq 2n$ ein Code mit Korrektur eines Fehlers vom Typ $\{1 \to 0,\ 0 \to 1,\ 0 \to \lambda,\ 1 \to \lambda,\ \lambda \to 0,\ \lambda \to 1\}$

3.3 Abschätzungen für fehlerkorrigierende Codes

In diesem Abschnitt betrachten wir nur Austauschfehler, d.h. es gilt stets $F = \{0 \to 1, 1 \to 0\}$. Der Einfachheit halber verwenden wir deshalb zur Bezeichnung des Abstandes d anstelle von $d_{\{0\to1,1\to0\}}$.

Wir betrachten den HAMMING-Code H_7. Aus den Betrachtungen des vorigen Abschnitts wissen wir, dass H_7 wegen $l = \lfloor log_2(7) \rfloor + 1 = 3$ aus $2^{7-3} = 2^4 = 16$ Elementen besteht. Folglich sind wir bei Verwendung von H_7 nur in der Lage 16 Symbole durch

Wörter der Länge 7 über $\{0,1\}$ zu codieren. Es erhebt sich die Frage, ob dies optimal ist, d.h. ob es einen Blockcode $C \subseteq \{0,1\}^7$ mit mindestens 17 Elementen gibt, der ebenfalls einen Austauschfehler korrigieren kann.

Wir wollen zeigen, dass dies nicht der Fall ist. Sei deshalb $C \subseteq \{0,1\}^7$ ein Code mit Korrektur eines Austauschfehlers. Für ein Element $X \in C$ setzen wir

$$U_1(X) = \{Y \mid d(Y,X) \leq 1\}$$

und betrachten die Vereinigung V dieser Mengen, d.h.

$$V = \bigcup_{X \in C} U_1(X)\,.$$

Da C einen Austauschfehler korrigieren kann, gilt für den Codeabstand $d(C) = 3$ und folglich sind die Mengen $U_1(X)$ und $U_1(X')$ für $X, X' \in C$ disjunkt. Da jede der Mengen $U_1(X)$ genau 8 Elemente enthält, nämlich X selbst und die 7 Wörter, die durch Änderung von X an genau einer Stelle entstehen, gilt

$$\#(V) = \#(\bigcup_{X \in H_7} U_1(X)) = \#(C) \cdot 8 \leq 128\,,$$

da es genau $2^7 = 128$ verschiedene Wörter der Länge 7 gibt. Damit ergibt sich

$$\#(C) \leq \frac{128}{8} = 16\,.$$

Folglich ist 16 die maximale Zahl von Symbolen, die durch Codes in $\{0,1\}^7$ mit Korrektur von einem Austauschfehler, codiert werden können.

In diesem Abschnitt wollen wir der Frage nach der maximalen Mächtigkeit von Blockcodes mit einer gegebenen Länge und mit einer gegebenen Anzahl von korrigierbaren Austauschfehlern nachgehen. Für natürliche Zahlen $n \geq 1$ und $d \geq 1$ setzen wir dazu

$$m(n,d) = \max\{\#(C) \mid C \subseteq \{0,1\}^n, d(C) \geq d\}\,.$$

Wegen Satz 3.2 gibt $m(n,d)$ die maximale Mächtigkeit von Codes aus Wörtern der Länge n und mit Korrektur von $\frac{d-1}{2}$ Austauschfehlern an.

Als erstes geben wir eine Abschätzung für $m(n,d)$ an, die dem am Beispiel demonstriertem Vorgehen entspricht.

Satz 3.3 *Für $n \geq 3$ und $s \geq 1$ gilt*

$$\frac{2^n}{\sum_{k=0}^{2s} \binom{n}{k}} \leq m(n, 2s+1) \leq \frac{2^n}{\sum_{k=0}^{s} \binom{n}{k}}\,.$$

Beweis. Sei $C \subseteq \{0,1\}^n$ ein Blockcode mit Korrektur von s Austauschfehlern, d.h. mit Codeabstand $2s+1$ nach Satz 3.2, mit maximaler Mächtigkeit, d.h. $\#(C) = m(n, 2s+1)$. Für $X \in C$ und eine positive natürliche Zahl r setzen wir

$$U_r(X) = \{Y \mid Y \in \{0,1\}^n,\ d(Y,X) \leq r\}\,.$$

Wenn $d(Y,X) = k$ gilt, so unterscheiden sich X und Y an genau k Stellen. Umgekehrt erhalten wir bei beliebiger Wahl von k Stellen in X und der Änderung der Buchstaben an diesen Stellen ein Wort Y mit $d(Y,X) = k$. Da wir $\binom{n}{k}$ Möglichkeiten zur Wahl der k Stellen in X haben, gibt es genau $\binom{n}{k}$ Wörter Y der Länge n mit $d(Y,X) = k$. Somit ergibt sich wegen

$$U_r(X) = \bigcup_{k=0}^{r} \{Y \mid d(Y,X) = k\}$$

für die Anzahl der Elemente in $U_r(X)$

$$\#(U_r(X)) = \sum_{k=0}^{r} \binom{n}{k} .$$

Man beachte, dass diese Zahl für alle Elemente X gleich ist. Da die Mengen $U_s(X)$ und $U_s(X')$ für $X, X' \in C$ disjunkt sind (siehe Beweis von Satz 3.2), erhalten wir

$$m(n, 2s+1) \cdot \sum_{k=0}^{s} \binom{n}{k} = \#(C) \cdot \sum_{k=0}^{s} \binom{n}{k} = \#(\bigcup_{X \in C} U_s(X)) \leq \#(\{0,1\}^n) = 2^n ,$$

woraus sich sofort die zweite Ungleichung der Behauptung ergibt.

Wir betrachten nun

$$V = \bigcup_{X \in C} U_{2s}(X) .$$

Da die Mengen $U_{2s}(X)$ und $U_{2s}(X')$ für $X, X' \in C$ nicht notwendig disjunkt sein müssen, erhalten wir

$$\#(V) \leq \#(C) \cdot \sum_{k=0}^{2s} \binom{n}{k} = m(n, 2s+1) \cdot \sum_{k=0}^{2s} \binom{n}{k} .$$

Nehmen wir nun an, dass

$$m(n, 2s+1) \cdot \sum_{k=0}^{2s} \binom{n}{k} < 2^n \tag{3.6}$$

gilt, so haben wir auch $\#(V) < 2^n$. Daher muss es dann ein Wort Z der Länge n geben, das nicht in V liegt. Damit gilt dann auch $Z \notin U_{2s}(X)$ für alle $X \in C$. Folglich gilt $d(Z,X) \geq 2s+1$ für alle $X \in C$. Folglich hat auch der Code $C \cup \{Z\}$ den Codeabstand $2s+1$ und ist in $\{0,1\}^n$ enthalten. Dies widerspricht aber der vorausgesetzten Maximalität von C hinsichtlich der Mächtigkeit. Deshalb kann (3.6) nicht gelten, womit

$$2^n \leq m(n, 2s+1) \cdot \sum_{k=0}^{2s} \binom{n}{k} ,$$

gültig ist. Hieraus ergibt sich die erste Ungleichung der Behauptung sofort. □

Wir bemerken, dass aus dem letzten Teil des Beweises eine induktive Methode zur Konstruktion von hinsichtlich der Mächtigkeit maximalen Blockcodes vorgegebener Länge folgt. Diese ist durch das folgende „Programm“ gegeben, das einen Code C mit maximaler Mächtigkeit ermittelt:

$C := \emptyset$;
A: $V := \bigcup_{X \in C} U_{2s}(X)$;
if $V = \{0,1\}^n$ **then goto** B ;
Wähle $X \in \{0,1\}^n \setminus V$;
$C := C \cup \{X\}$;
goto A ;
B: **stop**

Wir geben jetzt einige Beziehungen zwischen den Werten $m(n,d)$ und $m(n',d')$ für gewisse Parameter n, n', d, d'.

Satz 3.4 *Für zwei beliebige positive natürliche Zahlen $n$und d (mit $n \geq d$) gilt*

$$m(n,d) \leq 2 \cdot m(n-1,d)\,.$$

Beweis. Es sei C ein maximaler Blockcode der Länge n mit Codeabstand d, d.h. $\#(C) = m(n,d)$. Seien C_0 bzw. C_1 die Teilmengen von C, die aus allen Wörtern bestehen, die mit 0 bzw. 1 beginnen. Dann gilt offenbar $\#(C_0)+\#(C_1) = \#(C)$. Daher gibt es ein $i \in \{0,1\}$ mit

$$\#(C_i) \geq \frac{\#(C)}{2} = \frac{m(n,d)}{2}\,. \tag{3.7}$$

Wir betrachten nun den Blockcode C_i', der aus C_i entsteht, indem wir in jedem Wort den ersten Buchstaben (das ist i) streichen. Dann gilt sicher $C_i' \in \{0,1\}^{n-1}$, da jeweils ein Buchstabe gestrichen wurde. Für ein Wort $X \in C_i'$ setzen wir $X' = iX$. Offenbar gilt $X' \in C_i$. Weiterhin haben wir für zwei Wörter X und Y aus C_i' die Beziehung $d(X,Y) = d(X',Y')$, da X' und Y' den gleichen ersten Buchstaben haben. Wegen $d(C) = d$ und $X', Y' \in C_i \subseteq C$ erhalten wir $d \leq d(X',Y') = d(X,Y)$, woraus $d(C_i') \geq d$ folgt. Ein maximaler Code $C' \subseteq \{0,1\}^{n-1}$ mit Codeabstand d enthält mindestens soviel Elemente wie C_i' enthalten. Mit (3.7) ergibt sich daraus

$$m(n-1,d) \geq \#(C_i') = \#(C_i) \geq \frac{m(n,d)}{2}$$

und damit die Behauptung. □

Satz 3.5 *Seien n und d zwei beliebige positive natürliche Zahlen (mit $n \geq d$).*

i) Dann gilt $m(n,d) \geq m(n+1,d+1)$.
ii) Ist d ungerade, so gilt sogar $m(n,d) = m(n+1,d+1)$.

Beweis. i) Sei C ein Code mit $C \subseteq \{0,1\}^{n+1}$, $d(C) = d+1$ und $\#(C) = m(n+1,d+1)$. Wir betrachten den Code C', der aus C entsteht, indem wir in jedem Wort aus C den ersten Buchstaben streichen. Offenbar gelten dann $C' \subseteq \{0,1\}^n$, $\#(C') = \#(C)$ und $d(C') \geq d$, da bei einem Unterschied im ersten Buchstaben der Abstand beim Übergang von Wörtern aus C zu Wörtern aus C' um 1 sinken kann. Für einen maximalen Blockcode B der Länge n und Codeabstand d erhalten wir somit

$$m(n,d) \geq \#(C') = \#(C) = m(n+1,d+1)$$

und damit Teil i) der Behauptung.

ii) Wegen i) reicht es $m(n+1,d+1) \geq m(n,d)$ für ungerades d zu zeigen. Sei dazu $B \subseteq \{0,1\}^n$ ein Blockcode mit $d(B) = d$ und $\#(B) = m(n,d)$. Jedem Wort $X = x_1x_2\ldots x_n \in B$ ordnen wir das Wort

$$X' = x_1x_2\ldots x_nx_{n+1} \quad \text{mit} \quad x_{n+1} = x_1 \oplus x_2 \oplus \ldots \oplus x_n$$

zu und betrachten den Code

$$B' = \{X' \mid X \in B\}\,.$$

Offenbar ist B' ein Blockcode aus Wörtern der Länge $n+1$ und enthält die gleiche Anzahl von Elementen wie B.

Wir zeigen nun, dass B' bei ungeradem d den Codeabstand $d+1$ hat. Dazu betrachten wir zwei Wörter X und Y aus B und die zugeordneten Wörter X' und Y' aus B'. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass X und Y die Form

$$\begin{aligned} X &= x_1x_2\ldots x_r\ \underbrace{11\ldots1}_{s}\underbrace{00\ldots0}_{t}, \\ Y &= x_1x_2\ldots x_r\ \underbrace{00\ldots0}_{s}\underbrace{11\ldots1}_{t} \end{aligned}$$

haben (durch Umsortieren der Komponenten kann dies erreicht werden) und $s \leq t$ gilt. Hieraus ergibt sich $d(X,Y) = s + t \geq d(B) = d$. Ferner haben X' und Y' die Form

$$\begin{aligned} X' &= x_1x_2\ldots x_r\ \underbrace{11\ldots1}_{s}\underbrace{00\ldots0}_{t}\ x_{n+1}\,, \\ Y' &= x_1x_2\ldots x_r\ \underbrace{00\ldots0}_{s}\underbrace{11\ldots1}_{t}\ y_{n+1} \end{aligned}$$

mit

$$\begin{aligned} x_{n+1} &= x_1 \oplus x_2 \oplus \ldots \oplus x_r \oplus \underbrace{1 \oplus 1 \oplus \ldots \oplus 1}_{s}, \\ y_{n+1} &= x_1 \oplus x_2 \oplus \ldots \oplus x_r \oplus \underbrace{1 \oplus 1 \oplus \ldots \oplus 1}_{t}\,. \end{aligned}$$

Gilt $s + t \geq d + 1$, so ist auch $d(X',Y') \geq s + t \geq d + 1$ gültig.

Wir betrachten nun den Fall $s + t = d$. Da d ungerade ist, impliziert dies, dass genau eine der Zahlen s und t gerade ist und die andere ungerade ist. Folglich sind die Werte x_{n+1} und y_{n+1} verschieden. Somit ergibt sich $d(X',Y') = d + 1$.

Damit gilt in jedem Fall $d(X',Y') \geq d+1$ für $X', Y' \in B'$, woraus $d(B') \geq d+1$ folgt. Für einen maximalen Blockcode C der Länge $n + 1$ mit Codeabstand $d + 1$ gilt folglich

$$m(n+1,d+1) = \#(C) \geq \#(B') = \#(B) = m(n,d)\,. \tag{3.8}$$

□

Für zwei Wörter $X = x_1x_2\ldots x_n$ und $Y = y_1y_2\ldots y_n$ der Länge n definieren wir ihre Summe $X \oplus Y$ durch

$$X \oplus Y = (x_1 \oplus y_1)(x_2 \oplus y_2)\ldots(x_n \oplus y_n)\,.$$

Offenbar gilt

$$d(X,Y) = \#_1(X \oplus Y)\,. \tag{3.9}$$

Satz 3.6 *Für zwei beliebige positive natürliche Zahlen n und d (mit $n \geq d$) gilt*

$$m(2n, 2d) \geq m(n, d) \cdot m(n, 2d)\,.$$

Beweis. Seien $C_1 \subseteq \{0,1\}^n$ ein Code mit $d(C_1) = d$ und $\#(C_1) = m(n,d)$ und $C_2 \subseteq \{0,1\}^n$ ein Code mit $d(C_2) = 2d$ und $\#(C_1) = m(n,2d)$. Zwei Wörtern $X \in C_1$ und $Y \in C_2$ ordnen wir das Wort

$$w(X,Y) = XX \oplus Y0^n$$

zu und betrachten den Code

$$C = \{w(X,Y) \mid X \in C_1, Y \in C_2\}\,.$$

Nach Definition ist C ein Blockcode mit Wörtern der Länge $2n$ Außerdem besteht C aus $\#(C_1) \cdot \#(C_2) = m(n,d) \cdot m(n,2d)$ Elementen. Wir zeigen nun noch $d(C) \geq 2d$. Hieraus folgt dann

$$m(2n, 2d) \geq \#(C) = m(n,d) \cdot m(n,2d)$$

und damit die Behauptung.

Daher reicht es, für beliebige $X', X'' \in C_1$ und $Y', Y'' \in C_2$ zu zeigen, dass

$$2d \leq d(w(X',Y'), w(X'',Y'')) = d(X'X' \oplus Y'0^n, X''X'' \oplus Y''0^n)$$

für $w(X',Y') \neq w(X'',Y'')$, d.h. für $X' \neq X''$ oder $Y' \neq Y''$, gilt. Durch getrennte Betrachtung der ersten und letzten n Buchstaben erhalten wir aus (3.9)

$$\begin{aligned} d(w(X',Y'), w(X'',Y'')) &= \#_1((X' \oplus Y') \oplus (X'' \oplus Y'')) + \#_1((X' \oplus 0^n) \oplus (X'' \oplus 0^n)) \\ &= \#_1((X' \oplus X'') \oplus (Y' \oplus Y'')) + \#_1(X' \oplus X'')\,. \end{aligned} \tag{3.10}$$

Falls $Y' = Y''$ (und damit $Y' \oplus Y'' = 0^n$) und $X' \neq X''$ gelten, ergibt sich aus (3.10) sofort

$$d(w(X',Y'), w(X'',Y'')) = 2 \cdot \#_1(X' \oplus X'') = 2 \cdot d(X', X'') \geq 2 \cdot d(C_1) = 2 \cdot d\,.$$

Sei daher $Y' \neq Y''$. Aus (3.9), (3.10) und den Eigenschaften der Abstandsfunktion d folgt dann

$$\begin{aligned} d(w(X',Y'), w(X'',Y'')) &= \#_1((X' \oplus X'') \oplus (Y' \oplus Y'')) + \#_1((X' \oplus X'') \oplus 0^n) \\ &= d(X' \oplus X'', Y' \oplus Y'') + d(X' \oplus X'', 0^n) \\ &= d(Y' \oplus Y'', X' \oplus X'') + d(X' \oplus X'', 0^n) \\ &\geq d(Y' \oplus Y'', 0^n) = \#_1(Y' \oplus Y'') = d(Y', Y'') \geq d(C_2) \\ &= 2d \end{aligned}$$

□

Satz 3.7 *Seien n und d zwei beliebige positive natürliche Zahlen nund d (mit $n \geq d$).*
i) Für gerades d gilt

$$m(n,d) \leq 2 \cdot \lfloor \frac{d}{2d-n} \rfloor \text{ für } 2d > n,$$
$$m(n.d) \leq 2n \qquad \text{für } 2d = n\,.$$

ii) Für ungerades d gilt

$$m(n,d) \leq 2 \cdot \lfloor \frac{d+1}{2d+1-n} \rfloor \text{ für } 2d+1 > n,$$
$$m(n,d) \leq 2n \qquad \text{für } 2d+1 = n\,.$$

iii) Für $n \geq 2d$ gilt

$$m(n,d) \leq d \cdot 2^{n-2d+2} \quad \text{für gerades } d\,,$$
$$m(n,d) \leq (d+1) \cdot 2^{n-2d+1} \quad \text{für ungerades } d\,.$$

Beweis. Sei C ein beliebiger Code. Wir setzen

$$R(C) = \sum_{\substack{X,Y \in C \\ X \neq Y}} d(X,Y)\,,$$

d.h. $R(C)$ ist die Summe aller Abstände zwischen Wörtern aus C. Für $1 \leq i \leq n$ bezeichnen wir mit h_i die Anzahl der Wörter aus C, deren i-ter Buchstabe eine 1 ist. Folglich ist $\#(C) - h_i$ die Anzahl der Codewörter mit 0 als i-tem Buchstaben. Jedes Paar von Wörtern X und Y aus C mit 1 bzw. 0 an der i-ten Stelle durch diesen Unterschied 1 zu $R(C)$ bei. Daher gilt

$$R(C) = \sum_{i=1}^{n} h_i(\#(C) - h_i)\,. \tag{3.11}$$

Weiterhin gilt für je zwei Worte $X, Y \in C$ mit $X \neq Y$ die Beziehung $d(C) \leq d(X,Y)$. Da es $\frac{\#(C)(\#(C)-1)}{2}$ verschiedene ungeordnete Paare von verschiedenen Wörtern aus C gibt, gilt

$$\frac{\#(C)(\#(C)-1)}{2} \cdot d(C) \leq R(C)\,. \tag{3.12}$$

Sei nun C ein Code mit Wörtern der Länge n, $d(C) = d$ und $\#(C) = m(n,d)$. Wir setzen zur Abkürzung $m = m(n,d)$.

Sei zuerst m eine gerade Zahl. Dann nimmt der Ausdruck $h(m-h)$ den maximalen Wert bei $h = \frac{m}{2}$ an. Folglich ergibt sich aus (3.11) und (3.12)

$$\frac{m(m-1)}{2} \cdot d \leq R(C) \leq n \cdot \frac{m}{2}(m - \frac{m}{2}) = n \cdot \frac{m^2}{4}\,.$$

Ist m ungerade, so liegt das Maximum der Funktion bei $\frac{m-1}{2}$. Damit ergibt aus (3.11) und (3.12)

$$\frac{m(m-1)}{2} \cdot d \leq R(C) \leq n \cdot \frac{m-1}{2}(m - \frac{m-1}{2}) = n \cdot \frac{m-1}{2} \cdot \frac{m+1}{2} = n \cdot \frac{m^2-1}{4} \leq n \cdot \frac{m^2}{4}\,.$$

In beiden Fällen gilt also

$$\frac{m(m-1)}{2} \cdot d \leq n \cdot \frac{m^2}{4}\,.$$

Durch einfache algebraische Umformung erhalten wir hieraus

$$m^2 \cdot \frac{2d-n}{2} \leq md\,.$$

Für $2d \geq n$ ergibt sich

$$m \leq \frac{2d}{2d-n}\,,$$

woraus wegen der Ganzzahligkeit von m die schärfere Aussage

$$m \leq 2 \cdot \lfloor \frac{d}{2d-n} \rfloor$$

folgt. Wegen $m = m(n,d)$ ist damit der erste Teil von i) bereits gezeigt.

Sei nun $2d = n$. Dann folgt unter Verwendung von Satz 3.4 und der gerade bewiesenen Ungleichung aus i) (für $2d \geq 2d-1 = n'$)

$$m(n,d) = m(2d,d) = 2 \cdot m(2d-1,d) \leq 2 \cdot 2 \cdot \lfloor \frac{d}{2d-(2d-1)} \rfloor = 4d = 2n\,.$$

Damit ist auch der zweite Teil von i) bewiesen.

Wir verschärfen[1] nun die beiden Aussagen aus i) für ungerades d unter Verwendung von Satz 3.6. Wir erhalten

$$m(n,d) = m(n+1,d+1) \leq 2 \cdot \lfloor \frac{d+1}{2(d+1)-(n+1)} \rfloor = 2 \cdot \lfloor \frac{d+1}{2d+1-n} \rfloor \quad \text{für} \quad 2d+1 \geq n$$

und

$$m(2d+1,d) = m(2d+2,d+1) \leq 4(d+1) \quad \text{für} \quad 2d+1 = n\,.$$

Dies sind gerade die Beziehungen aus ii).

Wir beweisen nun iii) durch vollständige Induktion über n.

Sei zuerst d gerade. Für $n = 2d$ erhalten wir aus der zweiten Aussage von Teil i)

$$m(n,d) \leq 2n = 4d = d \cdot 2^2 = d \cdot 2^{n-2d+2}$$

und somit den Induktionsanfang für $n = 2d$. Ist die Aussage schon für $n \geq 2d$ bewiesen, so folgt sie für $n+1$ wegen Satz 3.4 aus

$$m(n+1,d) \leq 2 \cdot m(n,d) \leq 2 \cdot d \cdot 2^{n-2d+2} = d \cdot 2^{(n+1)-2d+2}\,.$$

Für ungerades d folgt der Induktionsanfang für $n = 2d+1$ aus der zweiten Aussage von ii) und einem Induktionsschritt wie für den geraden Fall. □

[1] Der Leser möge sich überlegen, dass tatsächlich $\frac{d+1}{2d+1-n} \leq \frac{d}{2d-n}$ für $2d \geq n$ gilt.

Kapitel 4

Lineare Codes

Bisher haben wir Codes als Mengen von Wörtern aufgefasst. Um Codeeigenschaften zu ermitteln oder zu untersuchen, haben wir im Wesentlichen kombinatorische Eigenschaften der Wortmenge bzw. der Wörter selbst betrachtet. In Abschnitt 3.2 haben wir bei den Hamming-Codes festgestellt, dass die Menge der Codewörter sogar einen linearen Vektorraum bildet. Daher können neben kombinatorischen Eigenschaften auch die algebraischen Eigenschaften der Wortmenge beim Studium der Hamming-Codes benutzt werden. Diese Möglichkeit soll in diesem Kapitel für eine ganze Klasse von Codes genutzt werden.

Jedem Wort $w = a_1a_2 \dots a_n$ der Länge n kann in eineindeutiger Weise der n-dimensionale Zeilenvektor $v_w = (a_1, a_2, \dots, a_n)$ zugeordnet werden. In diesem Abschnitt werden wir oft nicht zwischen dem Wort und seinem zugeordneten Vektor unterscheiden. Daher erhalten wir auch eine Addition von Wörtern der Länge n, da im Vektorraum aller n-dimensionalen Vektoren eine Addition definiert ist. Dies liefert

$$a_1a_2 \dots a_n \oplus b_1b_2 \dots b_n = c_1c_2 \dots c_n \text{ mit } c_i = a_i \oplus b_i \text{ für } 1 \leq i \leq n\,.$$

Definition 4.1 *Ein Blockcode $C \subseteq \{0,1\}^n$ heißt linearer Code, wenn die Elemente aus C einen linearen Vektorraum über dem Körper $\{0,1\}$ bilden.*[1]

Aus den Eigenschaften eines linearen Vektorraumes folgt sofort, dass das Wort 0^n in jedem linearen Code $C \subseteq \{0,1\}^n$ ist.

Der lineare Code $C \subseteq \{0,1\}^n$ hat als Vektorraum eine Dimension, die wir mit $dim(C)$ bezeichnen. Wir sagen dann auch, dass C ein $[n, dim(C)]$-Code ist.

Definition 4.2 *Sei C ein $[n,k]$-Code.*

i) Eine Matrix G vom Typ (k,n) heißt Erzeugendenmatrix für C, falls die k Zeilen von G ein Erzeugendensystem für C (als Vektorraum) bilden.

ii) Eine Matrix H vom Typ $(n-k,n)$ heißt Kontrollmatrix für C, falls

$$C = \{c \mid c \in \{0,1\}^n, Hc^T = (0^{n-k})^T\}$$

gilt.

[1] Wie schon im vorhergehenden Kapitel beschränken wir uns auch in diesem Kapitel auf Codes über $\{0,1\}$. Einige unserer Konzepte und Resultate können aber auch für den Fall formuliert bzw. bewiesen werden, wenn man anstelle des Körpers $\{0,1\}$ einen anderen endlichen Körper K und Codes über K (genauer Blockcodes, die in K^* enthalten sind) betrachtet.

Da jeder lineare Vektorraum eine Basis besitzt und die Elemente der Erzeugenmatrix eine Basis bilden, gibt es zu jedem linearen Code eine Erzeugendenmatrix.

Sei C ein $[n,k]$-Code. Dann bilden die n-dimensionalen Vektoren, die senkrecht auf allen Vektoren aus C stehen, d.h. die Menge aller v mit $vc^T = 0$ für alle $c \in C$, einen linearen Vektorraum C' der Dimension $n-k$. Wählen wir nun eine Basis von C' und nehmen deren Elemente als Zeilen einer Matrix, so bilden diese eine Kontrollmatrix H für C. Dies folgt daraus, dass die Menge C'' aller Vektoren x mit $Hx^T = (0^n)^T$ als Lösungsmenge eines linearen homogenen Gleichungssystems einen linearen Vektorraum der Dimension $n-(n-k) = k$ bildet und nach Definition alle Elemente aus C in C'' liegen, woraus sich $C'' = C$ ergibt. Damit besitzt auch jeder lineare Code eine Kontrollmatrix.

Für den Hamming-Code aus Abschnitt 3.2. ergeben sich als Erzeugendenmatrix

$$G = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix},$$

indem wir die Codewörter so wählen, dass die ersten drei Komponenten eine Permutation der Einheitsmatrix bilden, und als Kontrollmatrix

$$H = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix},$$

indem wir aus der Tabelle in Abschnitt 3.2 die Wahlen von x_3, x_5 und x_6 auswählen, bei denen diese Komponenten erneut im Wesentlichen die Einheitsmatrix bilden.

Definition 4.3 *i) Unter dem Gewicht $w(c)$ eines Wortes $c \in \{0,1\}^*$ verstehen wir die Anzahl der in c vorkommenden Einsen.*

ii) Das Gewicht $w(C)$ eines Blockcodes $C \subseteq \{0,1\}^n$ wird durch

$$w(C) = \min\{w(c) \mid w \in C \setminus \{0^n\}$$

definiert.

Offensichtlich gelten $w(c) = \#_1(c)$ und $d(c_1,c_2) = w(c_1 \oplus c_2)$ für alle Wörter $c_1, c_2 \in \{0,1\}^n$. Für $c = c_1c_2\ldots c_n$ setzen wir

$$Tr(c) = \{i \mid c_i = 1\}.$$

Dann gilt offenbar $w(c) = \#(Tr(c))$.

Seien c_1 und c_2 zwei Wörter der Länge n. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass

$$c_1 = 1^t0^s1^r0^{n-t-s-r} \quad \text{und} \quad c_2 = 0^t1^s1^r0^{n-t-s-r}$$

gilt (durch Umordnen kann dies stets erreicht werden). Es gilt dann

$$w(c_1 \oplus c_2) = t+s = (t+r)+(s+r)-2r = w(c_1)+w(c_2)-2\cdot\#(Tr(c_1)\cap Tr(c_2)). \quad (4.1)$$

Wir geben nun eine Charakterisierung des Gewichtes eines Codes durch die Kontrollmatrix an.

Satz 4.1 *Es sei C ein linearer $[n,k]$-Code und H eine Kontrollmatrix für C. Dann gilt*

$$\begin{aligned} w(C) &= \min\{r \mid \textit{es gibt } r \textit{ linear abhängige Spalten von } H\} \\ &= \max\{r \mid \textit{je } r-1 \textit{ Spalten von } H \textit{ sind linear unabhängig}\} \end{aligned}$$

Beweis. Es seien $h_1, h_2, \ldots, h_n$ die Spalten von H. Da H nur $n-k$ Zeilen hat, sind die n Spalten linear abhängig. Es sei nun p die minimale Anzahl linear abhängiger Spalten von H. $h_{i_1}, h_{i_2}, \ldots, h_{i_p}$ seien p linear abhängige Spalten. Dann gilt wegen der Minimalität von p die Beziehung $\sum_{j=1}^{p} h_{i_j} = (0^{n-k})^T$. Wir definieren nun den Vektor $v = v_1, v_2, \ldots v_n$ dadurch, dass wir genau dann $v_i = 1$ setzen, wenn $i \in \{i_1, i_2, \ldots, i_p\}$. Dann gilt

$$Hv^T = \sum_{i=1}^{n} v_i h_i = \sum_{j=1}^{p} h_{i_j} = (0^{n-k})^T .$$

Damit ist nach Definition der Kontrollmatrix $v \in C$. Da $w(v) = p$ ist, erhalten wir $p \geq w(C)$.

Angenommen, es gibt ein $c \in C$ mit $w(c) = t < p$. Es seien $c_{k_1}, c_{k_2}, \ldots, c_{k_t}$ die von Null verschiedenen Komponenten von c. Da $Hc^T = (0^{n-k})^T$ gilt, ist

$$(0^{n-k})^T = \sum_{i=1}^{n} c_i h_i = \sum_{j=1}^{t} h_{k_t} .$$

Damit sind die t Spalten $h_{k_1}, h_{k_2}, \ldots, h_{k_t}$ linear abhängig, was wegen der Minimalität von p unmöglich ist. Folglich gilt

$$w(C) = \min\{r \mid \text{es gibt } r \text{ linear abhängige Spalten von } H\} .$$

Die andere Gleichheit folgt sofort. □

Wir wollen nun zeigen, dass die Berechnung des Codeabstandes bei linearen Codes (erheblich) einfacher ist als bei beliebigen Blockcodes.

Satz 4.2 *Für einen linearen Code C gilt $d(C) = w(C)$.*

Beweis. Sei zuerst c ein Codewort aus $C \subseteq \{0,1\}^n$, für das $w(C) = w(c)$ gilt. Da C ein linearer Code ist, ist $0^n \in C$. Offenbar gilt $w(c) = d(c, 0^n) \geq d(C)$. Folglich haben wir $w(C) \geq d(C)$.

Es seien nun c_1 und c_2 zwei (verschiedene) Codewörter aus C mit $d(c_1, c_2) = d(C)$. Dann erhalten wir $d(C) = d(c_1, c_2) = w(c_1 \oplus c_2) \geq w(C)$.

Somit folgt $d(C) = w(C)$. □

Zur Bestimmung des Codeabstandes müssen wir im allgemeinen Fall alle Abstände zwischen zwei Codewörtern betrachten und dann das Minimum bestimmen. Dies erfordert einen quadratischen Aufwand in der Anzahl der Codewörter. Bei linearen Codes haben dagegen nur das Minimum der Gewichte zu ermitteln, was mit linearen Aufwand erfolgen kann.

Als zweites Beispiel für die Effizienz von linearen Codes betrachten wir die Decodierung. Nehmen wir an, dass wir ein Wort v empfangen haben. Falls es kein Codewort ist, so ist es sehr natürlich anzunehmen, dass das Codewort x gesendet wurde, für das

$$d(v, x) = \min\{d(v, c) \mid c \in C\} \tag{4.2}$$

erfüllt ist. Dies erfordert im Allgemeinen einen Aufwand $k \cdot n \cdot \#(C)$, wobei k eine Konstante ist (für jedes Codewort erfordert die Berechnung des Abstandes n Vergleiche).

Sei nun C ein linearer $[n,k]$-Code. Wir definieren die Äquivalenzrelation ϱ in $\{0,1\}^n$ durch

$$(x,y) \in \varrho \quad \text{genau dann, wenn} \quad Hx^T = Hy^T \,.$$

(Es ist leicht zu sehen, dass ϱ tatsächlich eine Äquivalenzrelation ist.) Die Nebenklasse bez. ϱ, die 0^n enthält, besteht dann genau aus den Vektoren x mit $Hx^T = H(0^n)^T = (0^{n-k})^T$). Damit besteht diese Nebenklasse genau aus den Elementen aus C. Sei nun f ein Repräsentant einer Nebenklasse N von ϱ. Dann gilt $N = f \oplus C$. Somit gibt es $2^n/\#(C)$ Nebenklassen.

Für jede Nebenklasse N bestimmen wir nun das Element f_N mit

$$w(f_N) = \min\{w(y) \mid y \in N\} \,.$$

Das empfangene Wort v liegt in einer Äquivalenzklasse, sagen wir in N. Für das Codewort x mit (4.2) und $f = v - x$ gilt

$$Hf^T = H(v-x)^T = Hv^t - Hx^T = Hv^T \,,$$

d.h., dass v und f in der gleichen Nebenklasse, also in N liegen. Weiterhin haben wir aber auch $f_N = v \oplus c'$ für ein Codewort c'. Damit gilt $w(f_N) = d(v,c')$. Wegen der Wahl von f gilt daher $c(f) \leq c(f_N)$. Nach Wahl von f_N gilt aber auch $w(f_N) \leq w(f)$. Folglich ist $w(f_N) = w(f)$. Somit können wir zur Decodierung von v, das Codewort c' mit $v + c' = f_N$ verwenden.

Um das gesendete Codewort zu ermitteln, reicht es also die Elemente f_N, wobei N eine Nebenklasse ist, durchzumustern und festzustellen, welches von diesen $Hf_N^T = Hv^T$ erfüllt. Da die Vektoren Hf_N^T vorab berechnet werden können, muss also nur Hv^T berechnet werden und mit den Hf_N^T verglichen werden. Die Berechnung von Hv^T kann in $k' \cdot n^2$ Schritten erfolgen, wobei k' eine Konstante ist (man gehe entsprechend der Definition des Produktes vor), jeder der Vergleiche erfordert n Schritte. Folglich haben wir höchstens

$$k' \cdot n^2 + n \cdot \frac{2^n}{\#(C)}$$

Schritte auszuführen. Falls $dim(C) = t > n/2$ ist, so ist der Aufwand kleiner als $k'n^2 + n2^{n-t}$. Dieser Aufwand ist wegen $t < n/2$ geringer als der des obigen allgemeinen Verfahrens, das $kn2^t$ Schritte erfordert.

Wir wollen nun ein paar Aussagen über die maximale Mächtigkeit linearer Codes machen. Da die Anzahl der Elemente eines linearen Codes der Dimension k gerade 2^k ist, reicht es die Dimension zu maximieren.

Für $n \geq 1$ und $d \geq 1$ definieren wir

$$k(n,d) = \max\{dim(C) \mid C \subseteq \{0,1\}^n \text{ ist linearer Code mit } d(C) \geq d\} \,.$$

Aus den Aussagen von Satz 3.4 – 3.6 und der Tatsache dass einer Multiplikation bei der Mächtigkeit des Codes eine Addition der Dimension entspricht erhalten wir sofort

$$\begin{aligned} k(n,d) &\leq k(n-1,d)+1 \,, \\ k(n,d) &= k(n+1,d-1) \text{ für ungerades } d \,, \\ k(2n,2d) &\geq k(n,d)+k(n,2d) \,. \end{aligned}$$

Weiterhin definieren wir $n(k,d)$ als die minimale Zahl n, so dass ein linearer Code C mit $C \subseteq \{0,1\}^n$, $d(C) = d$ und $dim(C) = k$ existiert.

Ohne Beweis bemerken wir, dass n eine sowohl in k als auch in d wachsende Funktion ist, d.h. es gelten für beliebiges $k \geq 1$ und $d \geq 1$ die Ungleichungen

$$n(k+1,d) > n(k,d) \quad \text{und} \quad n(k,d+1) > n(k,d)\,.$$

Satz 4.3 *Für $k > 1$ ist*

$$n(k,d) \geq n(k-1, \lceil \frac{d}{2} \rceil) + d\,.$$

Beweis. Es sei C ein linearer $[n,k]$-Code mit $n = n(k,d)$ und Codeabstand d. Ferner sei G eine Erzeugendenmatrix von C. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass eine Zeile von G durch das Codewort $c_1 = 0^{n-d}1^d$ gebildet wird (wir wählen als eines der erzeugenden Elemente von C ein Wort c_1' mit $d = d(c_1', 0^n)$ und vertauschen notfalls die Reihenfolge (d.h. die Spalten der Matrix), um c zu erhalten). Die Zeilen von G seien $c_1, c_2, \ldots, c_k$. Für $2 \leq i \leq k$ sei d_i die Zeile aus G, die aus c_i durch Streichen der letzten d Komponenten hervorgeht. G' sei die Matrix mit den Zeilen $d_2, d_3, \ldots, d_k$.

Angenommen, die Zeilen von G' wären linear abhängig. Dann hätten wir $\sum_{i=2}^{k-1} \alpha_i d_i = 0^{n-d}$, wobei $\alpha_i = 1$ für mindestens ein i mit $2 \leq i \leq d$ gilt. Wir betrachten nun $c = \sum_{i=2}^{k-1} \alpha_i c_i$. Offensichtlich ist $c \in C$ wegen der Linearität von C. Außerdem gilt $c = 0^{n-d}c'$. Falls $c \neq c_1$, so gilt $d(c, 0^n) < d$, womit ein Widerspruch zu $d = d(C)$ besteht. Somit muss $c = c_1$ sein. Dann gilt aber (wegen $1 = -1$) $c_1 \oplus \sum_{i=2}^{k-1} \alpha_i c_i = 0^n$ im Widerspruch zur linearen Unabhängigkeit der Zeilen von G. Daher ist unsere Annahme falsch, und folglich sind die Zeilen $d_2, d_3, \ldots, d_k$ linear unabhängig und bilden einen $[n-d, k-1]$-Code C'.

Es sei $d' = d(C')$. Dann gibt es eine Zeile b' in G' mit $w(b') = d'$. In G gibt es dann eine Zeile $b = b'b''$ mit $b'' \in \{0,1\}^d$. Damit erhalten wir $c_1 \oplus b \in C$ und

$$w(c_1 \oplus b) = d' + d - w(b'') \geq d \text{ und } w(b) = d' + w(b'') \geq d\,,$$

woraus $2d' \geq d$ resultiert. Folglich ist $d(C') \geq \lceil d/2 \rceil$ und $n - d \geq n(k-1, \lceil d/2 \rceil)$, was zu beweisen war. □

Unter Berücksichtigung von

$$n(1,d) = d \quad \text{und} \quad \Big\lceil \frac{\lceil d/2 \rceil}{2^i} \Big\rceil = \lceil \frac{d}{2^{i+1}} \rceil$$

ergibt sich aus Satz 4.3 durch Induktion sofort die folgende Aussage, die als Griesmer-Schranke für lineare Codes bekannt ist.

Folgerung 4.4 *Für $k \geq 1$ gilt*

$$n(k,d) \geq \sum_{i=0}^{k-1} \lceil \frac{d}{2^i} \rceil\,.$$

Aus der Griesmer-Schranke ergibt sich auch eine Abschätzung für $k(n,d)$.

Folgerung 4.5 *Es gilt*

$$k(n,d) \leq \max\{k \mid \sum_{i=1}^{k-1} \lceil \frac{d}{2^i} \rceil \leq n\}\,.$$

Beweis. Sei

$$l = \max\{k \mid \sum_{i=1}^{k-1} \lceil \frac{d}{2^i} \rceil \leq n\}\,.$$

Dann gilt wegen Folgerung 4.4

$$n(l+1,d) \geq \sum_{i=1}^{l} \lceil \frac{d}{2^i} \rceil > n\,.$$

Wäre nun $k(n,d) > l$, so wäre wegen der Monotonie von $n(k,d)$ auch $n = n(k(n,d),d) \geq n(l+1,d) > n$, woraus ein Widerspruch resultiert. □

Wir wollen nun lineare Codes konstruieren, die beweisen, dass die Griesmer-Schranke optimal ist. Dazu geben wir zuerst eine Methode an, mit der aus linearen Codes neue lineare Codes gewonnen werden können und die Parameter des neuen Codes sich aus denen der gegebenen Codes einfach errechnen lassen.

Lemma 4.6 *Seien die linearen Codes C_1 und C_2 mit den Dimensionen k_1 bzw. k_2 und den Codeabständen d_1 und d_2 gegeben. Dann ist*

$$C = C_1 \alpha C_2 = \{(c_1, c_1 \oplus c_2) \mid c_1 \in C_1, c_2 \in C_2\}$$

ein linearer Code mit

$$C \subseteq \{0,1\}^{2n}, \quad dim(C) = k_1 + k_2 \quad und \quad d(C) = \min\{2d_1, d_2\}\,.$$

Beweis. Nach Definition gilt $C \subseteq \{0,1\}^{2n}$. Damit ist C ein Blockcode. Es bleibt daher zu zeigen, dass C ein linearer Vektorraum ist, um C als linearen Code nachzuweisen. Dafür reicht es nach den Kriterien für Vektorräume zu beweisen, dass für beliebige Elemente x und y aus C und $\gamma \in \{0,1\}$ auch $x \oplus y$ und γx in C liegen. Seien $x = (c_1, c_1 \oplus c_2)$ und $y = (c_1', c_1' \oplus c_2')$ mit $c_1, c_1' \in C_1$ und $c_2, c_2' \in C_2$. Dann ergibt sich

$$\begin{aligned} x \oplus y &= (c_1, c_1 \oplus c_2) \oplus (c_1', c_1' \oplus c_2') = (c_1 \oplus c_1', (c_1 \oplus c_2) \oplus (c_1' \oplus c_2')) \\ &= (c_1 \oplus c_1', (c_1 \oplus c_1') \oplus (c_2 \oplus c_2'))\,, \end{aligned}$$

woraus mit $c_1 \oplus c_1' \in C_1$ und $c_2 \oplus c_2' \in C_2$ folgt, dass $x \oplus y \in C$ gilt. Wegen $0 \cdot x = 0^{2n} = (0^n, 0^n \oplus 0^n) \in C$ und $1 \cdot x = x \in C$ ist auch die zweite Forderung erfüllt.

Offensichtlich ist die Abbildung $\tau : (C_1 \times C_2) \to C$ vermöge $(c_1, c_2) \to (c_1, c_1 \oplus c_2)$ eine Isomorphie zwischen den Vektorräumen $(C_1 \times C_2)$ (mit komponentenweiser Addition) und C (denn es gelten

$$\begin{aligned} \tau((c_1, c_1') \oplus (c_2, c_2')) &= \tau((c_1 \oplus c_2, c_1' \oplus c_2')) \\ &= (c_1 \oplus c_1', (c_1 \oplus c_1') \oplus (c_2 \oplus c_2')) \\ &= (c_1, c_1 \oplus c_1') \oplus (c_1', c_1' \oplus c_2') \\ &= \tau((c_1, c_1')) \oplus \tau((c_2, c_2')) \end{aligned}$$

und

$$\tau(\gamma(c_1, c_2)) = \tau((\gamma c_1, \gamma c_2)) = (\gamma c_1, \gamma c_1 \oplus \gamma c_2) = (\gamma c_1, \gamma(c_1 \oplus c_2)) = \gamma\tau((c_1, c_2))\,.$$

Damit gilt $dim(C) = dim(C_1 \times C_2) = k_1 + k_2$.

Sei $c = (c_1, c_1 \oplus c_2)$. Zuerst betrachten wir den Fall, dass $c_2 = 0^n$ gilt. Dann ergibt sich

$$w(c) = w(c_1) + w(c_1) = 2 \cdot w(c_1) \geq 2d_1\,.$$

Für $c_2 \neq 0^n$ erhalten wir

$$\begin{aligned} w(c) &= w(c_1) + w(c_1 \oplus c_2) \\ &= w(c_1) + w(c_1) + w(c_2) - 2 \cdot \#(Tr(c_1) \cap Tr(c_2)) \quad \text{(wegen (4.1))} \\ &\geq w(c_2) \qquad (\text{wegen } w(c_1) = \#(Tr(c_1)) \geq \#(Tr(c_1) \cap Tr(c_2)))) \\ &\geq d_2\,. \end{aligned}$$

Somit haben wir $w(c) \geq \min\{2d_1, d_2\}$ für alle $c \in C$. Damit gilt $d(C) = w(C) \geq \min\{2d_1, d_2\}$.

Es seien c_1 und c_2 Codewörter aus C_1 bzw. C_2 so, dass $w(c_1) = d(C_1) = d_1$ bzw. $w(c_2) = d(C_2) = d_2$ gelten (also von minimalen Gewicht in den Codes). Dann erhalten wir wegen $0^n \in C_1 \cap C_2$

$$w((c_1, c_1 \oplus 0^n)) = 2w(c_1) = 2d_1 \quad \text{und} \quad w((0^n, 0^n \oplus c_2)) = w(c_2) = d_2\,.$$

Daher gilt

$$d(C) = w(C) = \min\{w(c) \mid c \in C\} \leq \min\{2d_1, d_2\}\,,$$

woraus mit Obigem sofort $d(C) = \min\{2d_1, d_2\}$ folgt. □

Wir nutzen die Konstruktion von linearen Codes aus linearen Codes, die in Lemma 4.6 eingeführt wurde, um Reed-Muller-Codes $RM(r, m)$ für $r, m \in \mathbf{N}$ und $0 \leq r \leq m$ zu definieren. Dazu definieren wir zuerst

$$RM(0, m) = \{0^{2^m}, 1^{2^m}\} \quad \text{und} \quad RM(m, m) = \{0, 1\}^{2^m}$$

und setzen für $1 \leq r \leq m$

$$RM(r, m) = RM(r, m-1)\alpha RM(r-1, m-1)\,.$$

Wir bestimmen die Reed-Muller-Codes für kleine Werte von r und m. Für $m = 1$ erhalten wir die Codes

$$RM(0, 1) = \{00, 11\} \quad \text{und} \quad RM(1, 1) = \{00, 01, 10, 11\}\,.$$

Für $m = 2$ und $m = 3$ ergeben sich die Codes

$$\begin{aligned} RM(0, 2) &= \{0000, 1111\}\,, \\ RM(1, 2) &= RM(1, 1)\alpha RM(0, 1) = \{0000, 0101, 1010, 1111, 0011, 0110, 1001, 1100\}, \\ RM(2, 2) &= \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, \\ &\qquad 1001, 1010, 1011, 1100, 1101, 1110, 1111\}\,. \end{aligned}$$

und

$$\begin{aligned}
RM(0,3) &= \{00000000, 11111111\}, \\
RM(1,3) &= \{00000000, 01010101, 10101010, 11111111, 00110011, 01100110, \\
&\quad 10011001, 11001100, 00001111, 01011010, 10100101, \\
&\quad 11110000, 00111100, 01101001, 10010110, 11000011\}, \\
RM(2,3) &= \{w_1 w_2 \mid w_1 \in \{0,1\}^4,\ w_2 = w_1 \oplus v,\ v \in RM(1,2)\}, \\
RM(3,3) &= \{0,1\}^8.
\end{aligned}$$

Reed-Muller-Codes wurden in den siebziger Jahre des vorigen Jahrhundert bei der Weltraumfahrt benutzt.

Wir wollen nun zeigen, dass die Reed-Muller-Codes $RM(r,m)$ lineare $[2^m, \sum_{i=0}^{r} \binom{m}{i}]$-Codes mit dem Codeabstand 2^{m-r} sind.

Wir beweisen dies mittels vollständiger Induktion über r. Für $r = 0$ ergibt sich die Aussage sofort aus der Definition von $RM(0,m)$, der ein linearer Code in $\{0,1\}^{2^m}$ mit der Dimension $1 (= \sum_{i=0}^{0} \binom{m}{i})$ und dem Codeabstand 2^m ist.

Wegen Lemma 4.6 erhalten wir, dass $RM(r,m)$ ein linearer Code mit

$$\begin{aligned}
RM(r,m) &= RM(r,m-1) \alpha RM(r-1,m-1) \subseteq \{0,1\}^{2^{m-1}} \cdot \{0,1\}^{2^{m-1}} = \{0,1\}^{2^m}, \\
d(RM(r,m)) &= \min\{2 \cdot d(RM(r-1,m)), d(RM(r-1,m-1))\} \\
&= \min\{2 \cdot 2^{m-1-r}, 2^{m-1-(r-1)}\} \\
&= 2^{m-r}, \\
dim(RM(r,m)) &= dim(RM(r,m-1)) + dim(RM(r-1,m-1) \\
&= \sum_{i=0}^{r} \binom{m-1}{i} + \sum_{i=0}^{r-1} \binom{m-1}{i} \\
&= (1 + \sum_{i=1}^{r} \binom{m-1}{i}) + \sum_{i=0}^{r-1} \binom{m-1}{i} \\
&= (1 + \sum_{i=0}^{r-1} \binom{m-1}{i+1}) + \sum_{i=0}^{r-1} \binom{m-1}{i} \\
&= 1 + \sum_{i=0}^{r-1} (\binom{m-1}{i+1} + \binom{m-1}{i}) \\
&= 1 + \sum_{i=0}^{r-1} \binom{m}{i+1} = 1 + \sum_{i=1}^{r} \binom{m}{i} \\
&= \sum_{i=0}^{r} \binom{m}{i}
\end{aligned}$$

ist.

Für die Reed-Muller-Codes $RM(1,m) \subseteq \{0,1\}^{2^m}$ haben wir

$$dim(RM(1,m)) = \sum_{i=0}^{1} \binom{m}{i} = m+1 \text{ und } d(RM(1,m)) = 2^{m-1}.$$

Damit ergibt sich aus Folgerung 4.4

$$\begin{aligned} n(k,d) &\geq \sum_{i=0}^{m} \lceil \frac{d}{2^i} \rceil = \sum_{i=0}^{m} \frac{2^{m-1}}{2^i} \\ &= 2^{m-1} + 2^{m-2} + \ldots + 2 + 1 + 1 = 2^m \,. \end{aligned}$$

Da andererseits $RM(1,m) \subseteq \{0,1\}^{2^m}$ gilt, ist damit die Griesmer-Schranke nicht zu verbessern.

In Folgerung 4.5 haben wir eine obere Schranke für die maximale Dimension $k(n,d)$ angegeben. Wir geben abschließend noch eine untere Schranke an, die auf Gilbert und Varshamov zurückgeht.

Satz 4.7 *Wenn die natürlichen Zahlen n, k und d die Bedingungen*

$$k \leq n \quad \textit{und} \quad 2^{n-k} > \sum_{i=0}^{d-2} \binom{n-1}{i}$$

erfüllen, so gibt es einen linearen Code C mit $C \subseteq \{0,1\}^n$, $dim(C) = k$ und $d(C) \geq d$ (es gilt also $k(n,d) \geq k$).

Beweis. Ist $k = n$, also $2^{n-k} = 2^0 = 1$, so muss wegen der zweiten vorausgesetzten Ungleichung $d = 1$ sein. Für die Parameter $n = k$ und $d = 1$ ist $\{0,1\}^n$ ein Code der gewünschten Art.

Sei also $k < n$. Es sei $v_1, v_2, \ldots, v_{n-k}$ eine Basis von $\{0,1\}^{n-k}$. Ferner seien v_{n-k+1}, $v_{n-k+2}, \ldots, v_{n-k+s}$ weitere Elemente aus $\{0,1\}^{n-k}$ derart, dass je $d-1$ Vektoren linear unabhängig sind. Die Anzahl der Vektoren, die sich als Linearkombination von höchstens $d-2$ Elementen aus $\{v_1, v_2, \ldots v_{n-k+s}\}$ bilden lassen ist

$$\sum_{i=0}^{d-2} \binom{n-k+s}{i} .$$

Ist diese Summe echt kleiner als 2^{n-k}, so läßt sich in $\{0,1\}^{n-k} \setminus \{v_1, v_2, \ldots, v_{n-k+s}\}$ ein Element $v_{n-k+s+1}$ finden, so dass je $d-1$ Vektoren aus $\{v_1, v_2, \ldots v_{n-k+s}, v_{n-k-s+1}\}$ linear unabhängig sind. Nach Voraussetzung erhalten wir die Existenz einer Menge $\{v_1, v_2, \ldots, v_n\}$. Aus $v_1, v_2, \ldots, v_n$ bilden wir nun eine Matrix K vom Typ $(n-k, n)$ und verwenden diese als Kontrollmatrix eines Codes C. C ist dann ein linearer $[n,k]$-Code, dessen Codeabstand nach den Sätzen 4.1 und 4.2 mindestens d ist. □

Kapitel 5

Klassische Verschlüsselungen

Im Rahmen der Codierungstheorie werden vor allem Fragen betrachtet, die daraus resultieren, dass bei der Übertragung ein Kanal benutzt wird. So werden u.a. eindeutige Dekodierbarkeit, eine schnelle Übertragung bzw. die Möglichkeiten zur Korrektur von Übertragungsfehlern angestrebt. Eine Geheimhaltung, d.h. die übermittelte Nachricht soll nicht von unbefugten Personen decodiert werden können, ist dagegen kein angestrebtes Ziel.

Die Kryptologie (oder Kryptographie) stellt sich nun gerade dieser Aufgabe. Man ist daran interessiert, eine Codierung einer Nachricht so vorzunehmen, dass folgende Bedingungen erfüllt sind:

- Die Codierung/Verschlüsselung der Nachricht durch den Sender soll einfach sein.
- Die Decodierung/Entschlüsselung der verschlüsselten Nachricht durch den befugten Empfänger soll einfach sein.
- Die Decodierung der verschlüsselten Nachricht durch unbefugte Personen soll unmöglich oder zumindest sehr schwer sein.

Die Kryptologie hat daher zwei Aspekte. Der eine Aspekt ist der des Entwurfs von Verschlüsselungen, die die oben genannten Bedingungen erfüllen, und der andere Aspekt betrifft die Situation des Kryptoanalysten, der versucht, als unbefugte Person die Entschlüsselung vorzunehmen. Natürlich sind diese Aspekte nicht unabhängig voneinander. So hat man sich schon beim Entwurf zu überlegen, welche Möglichkeiten ein (sehr intelligenter und fähiger) Kryptoanalyst bei der entworfenen Verschlüsselung besitzt, um sein Ziel zu erreichen.

Wir werden uns im Wesentlichen auf Chiffrierungen beschränken, bei denen Wörter über den Alphabet

$$alph = \{\text{A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}\}$$

der lateinischen Buchstaben wieder auf Wörter über *alph* abgebildet werden. In einigen Fällen werden wir auch Wörter über dem Alphabet $\{0, 1\}$, d.h. Binärfolgen oder Bit-Folgen, betrachten. In der Regel basiert die Chiffrierung auf einer Abbildungsvorschrift für einzelne Buchstaben, Paare von Buchstaben oder relativ kurze Wörter, die dann auf beliebig lange Wörter (homomorph) fortgesetzt werden kann. Diese Abbildungsvorschrift

ist in den meisten Fällen einfach zu realisieren, wenn man ein hierzu gehöriges Grundelement, den sogenannten Schlüssel, kennt. Auch die Dechiffrierung ist bei Kenntnis des Schlüssels einfach. Ist dagegen der Schlüssel nicht bekannt, so sind sehr viele mögliche Abbildungsvorschriften als Grundlage der Chiffrierung möglich, wodurch der Kryptoanalyst vor einer schwierigen Aufgabe steht.

Da die Schlüssel nur den befugten Personen bekannt sein sollen, müssen sie „merkbar“ sein. Insbesondere müssen sie eine sehr viel kürzer Beschreibung haben, als die Nachricht selbst (sonst wäre die Geheimhaltung des Schlüssels nicht einfacher als die der unverschlüsselten Nachricht). Besonders in der Vergangenheit bestanden die Schlüssel aus einer bzw. wenigen Zahlen oder einem Wort bzw. einem leicht merkbaren kurzen Text.

In der Folge werden wir den unverschlüsselten Text, der vom Sender an den Empfänger geschickt wird als Klartext bezeichnen. Unter dem Kryptotext verstehen wir den Text, der aus dem Klartext durch die Verschlüsselung/Chiffrierung entsteht.

Wir behandeln als erstes einige klasssische Chiffren, die teilweise schon seit Zeiten des Altertums benutzt wurden, aber teiweise bis die Neuzeit noch angewendet wurden.

5.1 Monoalphabetische Substitutionschiffren

Als erstes Beispiel für eine Chiffrierung wollen wir Verschiebungschiffren behandeln, die schon im Altertum benutzt wurden. In den Schriften von Sueton wird berichtet, dass Caesar (100 - 44 v.Chr.) diese Methode zur Verschlüsselung seiner geheimen Nachrichten benutzt hat. Daher werden sie manchmal auch als Caesar-Chiffren bezeichnet. Verschiebungschiffren werden durch zyklische Verschiebung der Buchstaben des Alphabets entsprechend ihrer Ordnung erzeugt. Formal ergibt sich folgendes Vorgehen.

Wir definieren die Funktion φ, die die Buchstaben des Alphabets in eineindeutiger Weise Zahlen zwischen 0 und 25 zuordnet, entsprechend der Tabelle in Abbildung 5.1. Als

α	A	B	C	D	E	F	G	H	I
$\varphi(\alpha)$	0	1	2	3	4	5	6	7	8
α	J	K	L	M	N	O	P	Q	R
$\varphi(\alpha)$	9	10	11	12	13	14	15	16	17
α	S	T	U	V	W	X	Y	Z	
$\varphi(\alpha)$	18	19	20	21	22	23	24	25	

Abbildung 5.1: Funktion φ

Schlüssel wählen wir eine Zahl i mit $0 \leq i \leq 25$. Wir definieren dann die Chiffriervorschrift $v_i : alph \to alph$ der i-ten Verschiebungschiffre durch

$$v_i(\alpha) = \varphi^{-1}(\varphi(\alpha) + i \bmod 26)) \,.$$

Bei Wahl von $i = 5$ erhalten wir aus dem Klartext MAGDEBURG den Kryptotext RFLIJGZWL.

Für den Schlüssel i, $0 \leq i \leq 25$, ist die Vorschrift für das Dechiffrieren dann offensichtlich durch die Funktion dv_i zu bewerkstelligen, die durch

$$dv_i(\alpha) = \varphi^{-1}(\varphi(\alpha) - i \bmod 26) = \varphi^{-1}(\varphi(\alpha) + (26 - i) \bmod 26)$$

(wir benutzen auch zur Bezeichnung der Äquivalenz $\bmod 26$ das Gieichheitszeichen) definiert ist.

Offenbar gibt es 26 verschiedene Verschiebungschiffren.

Wir betrachten nun den Kryptoanalysten. Liegt ein Paar *(Klartext, Kryptotext)* vor, so ist seine Aufgabe sehr leicht zu lösen. Er betrachtet nur den ersten Buchstaben α des Klartexts und den ersten Buchstaben β des Kryptotextes. Dann ergibt sich der verwendete Schlüssel i der Wert $\varphi(\beta) - \varphi(\alpha)$.

Ist nur ein Kryptotext gegeben, so könnte der Kryptoanalyst alle 26 möglichen Schlüssel und die zugehörigen Dechiffrierungen durchtesten und dabei sicher auch den Klartext finden. Es ist jedoch im Allgemeinen nicht notwendig alle 26 Möglichkeiten zu testen, wenn zusätzlich die Häufigkeit des Vorkommen der Buchstaben im Text berücksichtigt wird.

Für deutsche Texte gilt die in Abbildung 5.2 angegebene Wahrscheinlichkeitsverteilung der Buchstaben.[1]

α	A	B	C	D	E	F	G	H	I
$p(\alpha)$	6,51	1,89	3,06	5,08	17,40	1,66	3,01	4,76	7,55
α	J	K	L	M	N	O	P	Q	R
$p(\alpha)$	0,27	1,21	3,44	2,53	9,78	2,51	0,79	0,02	7,00
α	S	T	U	V	W	X	Y	Z	
$p(\alpha)$	7,27	6,15	4,35	0,67	1,89	0,03	0,04	1,13	

Abbildung 5.2: Wahrscheinlichkeitsverteilung der Buchstaben in deutschen Texten

Der Kryptoanalyst ermittelt zuerst die Wahrscheinlichkeiten des Auftretens der einzelnen Buchstaben im Text. Er geht dann davon aus, dass der im Kryptotext am häufigsten auftretende Buchstabe α_1 dem Buchstaben E entspricht. Er nimmt also die Entschlüsselung mit $i = \varphi(\alpha) - 4$ vor. Ergibt sich hierbei als vermuteter Klartext kein echter deutscher Text, so wird als nächstes angenommen, dass der zweithäufigste Buchstabe α_2 dem Buchstaben E entspricht. Ergibt sich hier kein brauchbares Resultat, so wird mit dem dritthäufigsten Buchstaben des Textes getestet usw.

Als Beispiel betrachten wir den Kryptotext

C T J C J C S C T J C O X V A J U I Q P A A D C H K D C C T C P X H I V J I

Die Häufigkeit des Vorkommens von Buchstaben im Kryptotext ergibt sich wie folgt:

[1] Die Wahrscheinlichkeitsverteilung ist nicht eindeutig zu ermitteln; sie hängt entscheidend davon ab, welche Texte analysiert wurden. Daher differieren auch die in den Büchern angegebenen Verteilungen leicht. Jedoch ist die Reihenfolge hinsichtlich der Häufigkeit auf den ersten sechs Plätzen stets gleich.

9-mal : C
5-mal : J
3-mal : A, I, T
2-mal : D, H, P, V, X
1-mal : K, N, O, Q, S
0-mal : B, E, F, G, L, M, R, W, Y, Z

Die Annahme, dass C als Originalbuchstaben E hat, liefert den Text

E V L E L E U E V L E Q Z X C L W K S R C C F E J M F E E V E R Z J K X L K

und die Annahme, dass J dem E entspricht führt zu

X O E X E X N X O E X J S Q V E P D L K V V Y X C F Y X X O X K S C D Q E D

Erst wenn der Kryptologe T als den E entsprechenden Buchstaben testet, erhält er den brauchbaren Text

N E U N U N D N E U N Z I G L U F T B A L L O N S V O N N E N A I S T G U T

(oder lesbarer NEUNUNDNEUNZIG LUFTBALLONS VON NENA IST GUT).

Jede Verschiebungschiffre v_i, $0 \leq i \leq 25$, stellt eine Permutation der Buchstaben dar und bei der Chiffrierung wird jeder Buchstabe immer durch den gleichen Buchstaben ersetzt. Chiffren mit dieser Eigenschaft nennen wir *monoalphabetische Subsitutionschiffren*. Bei der Substitution muss es sich selbstverständlich um eine Permutation handeln, da sonst keine eindeutige Dechiffrierung möglich wäre.

Die Verschiebungschiffren haben den Vorteil, dass man sich als Schlüssel nur eine Zahl merken muss (wir gehen natürlich davon aus, dass man die alphabetische Reihenfolge der Buchstaben kennt). Eine komplizierte Permutation kann man sich dagegen kaum merken, d.h. sie muss irgendwo notiert werden, wodurch Unbefugten ein Zugriff unter Umständen möglich wäre. Jedoch gibt es Permutationen, die man sich sehr leicht merken kann. Ein derartiges Beispiel kann wie folgt gebildet werden. Wir wählen als Schlüssel ein (möglichst langes Wort, in dem kein Buchstabe doppelt vorkommt. Dieses Wort schreiben wir dann zuerst hin und lassen ihm in umgekehrter alphabetischer Reihenfolge die Buchstaben folgen, die im Wort nicht vorkommen. Ausgehend vom Schlüsselwort GREIFSWALD erhalten wir die Permutation

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
G	R	E	I	F	S	W	A	L	D	Z	Y	X	V	U	T	Q	P	O	N	M	K	J	H	C	B

Aus dem Klartext MAGDEBURG entsteht hierbei der Kryptotext XGWIFRMPW.

Eine weitere Methode, komplizierte Permutationen zu erhalten, sind *affine* Chiffren. Hierbei werden als Schlüssel zwei natürliche Zahlen a und b so gewählt, dass $0 \leq a \leq 25$ und $0 \leq b \leq 25$ gelten und a und 26 den größten gemeinsamen Teiler 1 haben. Dann wird die Funktion $v_{(a,b)} : alph \to alph$ durch

$$v_{(a,b)}(\alpha) = \varphi^{-1}(a \cdot \varphi(\alpha) + b \bmod 26)$$

definiert. Die Verschiebungschiffren sind damit die affinen Chiffren, bei denen $a = 1$ ist.

α	A	B	C	D	E	F	G	H	I	J	K	L	M
$\varphi(\alpha)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$a \cdot \varphi(\alpha) + b \bmod 26$	5	8	11	14	17	20	23	0	3	6	9	12	15
$v_{(3,5)}(\alpha)$	F	I	L	O	R	U	X	A	D	G	J	M	P
α	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$\varphi(\alpha)$	13	14	15	16	17	18	19	20	21	22	23	24	25
$a \cdot \varphi(\alpha) + b \bmod 26$	18	21	24	1	4	7	10	13	16	19	22	25	2
$v_{(3,5)}(\alpha)$	S	V	Y	B	E	H	K	N	Q	T	W	Z	C

Abbildung 5.3: Beispiel einer affinen Chiffrierung mit $a = 3$ und $b = 5$

Für die Werte $a = 3$ und $b = 5$ erhalten wir die Zuordnungen aus Abbildung 5.3. Daraus ergibt sich für den Klartext MAGDEBURG der Kryptotext PFXORINEX.

Die Forderung, dass a und 26 den größten Teiler 1 haben sollen, wird aus drei Gründen erhoben:

- Wir erreichen dadurch eine eineindeutige Funktion. Zum Beispiel besteht bei Verwendung von $a = 10$ und $b = 1$ die Gleichheit $10 \cdot 0 + 1 = 10 \cdot 13 + 1 = 1 \bmod 26$, was bedeutet, dass A und N beide durch den gleichen Buchstaben B verschlüsselt werden müssten.

- Es wird abgesichert, dass $v_{(a,b)}$ eine Abbildung auf *alph* ist. Bei $a = 10$ und $b = 1$ würde kein Buchstabe durch O chiffriert werden, da es keine Zahl z mit $10z + 1 = 16 \bmod 26$ gibt, wie man leicht nachrechnet.

- Für a existiert ein inverses Element, d.h. es gibt ein a^{-1}, $0 \leq a^{-1} \leq 25$, mit $a \cdot a^{-1} = 1 \bmod 26$. Für $a = 3$ ist dieser Wert z.B. 9, da $3 \cdot 9 = 27 = 1 \bmod 26$ ist. Damit ergibt sich für die Dechiffrierung eines Buchstabens β der Buchstabe

$$\varphi^{-1}(a^{-1}(\varphi(\beta) - b)) .$$

Alle monoalphabetischen Subsitutionschiffren lassen sich aber vom Kryptoanalysten unter Verwendung der Häufigkeitsverteilung von Buchstaben und von Paaren (und Tripeln) von Buchstaben (die ebenfalls bestimmt wurden) ermitteln. Er hat jetzt im Wesentlichen nur mehrere der häufig auftretenden Buchstaben gleichzeitig zu betrachten, da die Kenntnis der richtigen Zuordnung eines Buchstaben jetzt nicht mehr ausreicht, um die anderen Zuordnungen zu ermitteln. Daher kann der Kryptoanalyst mittels der Trial-and-error-Methode den Klartext zu (langen) Kryptotexten ermitteln, ohne alle 26! Permutationen auszuprobieren.

5.2 Polyalphabetische Substitutionschiffren

Um dem Kryptoanalysten die Arbeit zu erschweren ist es also erforderlich, die Häufigkeitsverteilung zu verschleiern. Dies kann zum Beispiel dadurch geschehen, dass bei der

Substitution einem Buchstaben nicht stets der gleiche Wert zugeordnet wird. Die Substitutionen heißen daher *polyalphabetische Chiffren*. Wir betrachten hier drei Beispiele.

Unser erstes Beispiel sind HILL-Chiffren. Bei diesen wird als Schlüssel eine Matrix M vom Typ (2,2) gewählt, deren Elemente a_{ij}, $i,j \in \{1,2\}$, die Bedingung $0 \leq a_{ij} \leq 25$ erfüllen und für die es eine inverse Matrix M^{-1} gibt (für die

$$M \cdot M^{-1} = M^{-1} \cdot M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \bmod 26$$

gilt). Wir unterteilen den Text in Teilwörter der Länge 2 und verschlüsseln ein Wort $\alpha\beta$ durch $\alpha'\beta'$, wobei folgende Bedingungen gelten:

$$M \cdot \begin{pmatrix} \varphi(\alpha) \\ \varphi(\beta) \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \bmod 26\,, \quad \alpha' = \varphi^{-1}(x)\,, \quad \beta' == \varphi^{-1}(y)\,,$$

d.h. wir bilden zuerst einen Spaltenvektor mit den zu α und β gehörenden Zahlen, multiplizieren diesen Vektor mit der Schlüsselmatrix M und überführen die beiden erhaltenen Komponenten des Vektors wieder in Buchstaben. Die Dechiffrierung erfolgt jetzt genauso, nur dass anstelle von M die Matrix M^{-1} verwendet wird.

Als Beipiel betrachten wir die Matrix

$$M = \begin{pmatrix} 3 & 5 \\ 3 & 24 \end{pmatrix}$$

mit der inversen Matrix $M^{-1} = \begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix}$. Wir wollen das Wort SAHARA verschlüsseln. Den zugehörigen Teilwörtern SA, HA und RA entsprechen die Vektoren $(18,0)^T$, $(7,0)^T$ und $(17,0)^T$, woraus sich wegen

$$\begin{pmatrix} 3 & 5 \\ 3 & 24 \end{pmatrix} \begin{pmatrix} 18 \\ 0 \end{pmatrix} = \begin{pmatrix} 54 \\ 54 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \bmod 26\,,$$
$$\begin{pmatrix} 3 & 5 \\ 3 & 24 \end{pmatrix} \begin{pmatrix} 7 \\ 0 \end{pmatrix} = \begin{pmatrix} 21 \\ 21 \end{pmatrix}\,,$$
$$\begin{pmatrix} 3 & 5 \\ 3 & 24 \end{pmatrix} \begin{pmatrix} 17 \\ 0 \end{pmatrix} = \begin{pmatrix} 51 \\ 51 \end{pmatrix} = \begin{pmatrix} 25 \\ 25 \end{pmatrix} \bmod 26$$

als Kryptotext das Wort BBVVYY ergibt. Man erkennt, dass sich zum einen der Buchstabe A bei allen drei Vorkommen durch verschiedene Buchstaben verschlüsselt wird und zum anderen verschiedene Buchstaben wie z.B. S und A durch den gleichen Buchstaben verschlüsselt werden. Dadurch wird die Häufigkeit des Auftretens von Buchstaben stark verschleiert und der Kryptoanalyst kann nicht wie bei monoalphabetischen Substitutionen aus der Häufigkeit der Buchstaben Rückschlüsse auf die Chiffre ziehen.

Weiterhin wollen wir den Kryptotext HHTQ entschlüsseln. Wir erhalten die Vektoren $(7,7)^T$ zu HH und $(19,15)^T$ zu TQ und unter Verwendung von M^{-1}

$$\begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 7 \end{pmatrix} = \begin{pmatrix} 11 \\ 0 \end{pmatrix} \bmod 26 \text{ und } \begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix} \begin{pmatrix} 19 \\ 16 \end{pmatrix} = \begin{pmatrix} 18 \\ 19 \end{pmatrix} \bmod 26\,,$$

woraus der Klartext LAST resultiert.

Wir betrachten nun die Situation des Kryptoanalysten. Wenn er einen Klartext wählen und zu diesem den Kryptotext erhalten kann, so ist die Aufgabe für ihn einfach. Er wählt den kurzen Klartext HELP mit den zugehörigen Vektoren $(7,4)^T$ und $(11,15)^T$ und erhält den Kryptotext $\alpha\beta\gamma\delta$. Daraus erhält er die Gleichungen

$$N \cdot \begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} \varphi(\alpha) \\ \varphi(\beta) \end{pmatrix} \quad \text{und} \quad N \cdot \begin{pmatrix} 11 \\ 15 \end{pmatrix} = \begin{pmatrix} \varphi(\gamma) \\ \varphi(\delta) \end{pmatrix}$$

oder in anderer Schreibweise

$$N \cdot \begin{pmatrix} 7 & 11 \\ 4 & 15 \end{pmatrix} = \begin{pmatrix} \varphi(\alpha) & \varphi(\gamma) \\ \varphi(\beta) & \varphi(\delta) \end{pmatrix}$$

für die Schlüsselmatrix N. Die Wahl HELP wurde deshalb vorgenommen, weil die entstandene Matrix $U = \begin{pmatrix} 7 & 11 \\ 4 & 15 \end{pmatrix}$ eine inverse Matrix U^{-1} besitzt, für die sich $U^{-1} = \begin{pmatrix} 19 & 19 \\ 14 & 21 \end{pmatrix}$ ergibt. Damit erhalten wir

$$N = N \cdot (U \cdot U^{-1}) = (N \cdot U) \cdot U^{-1} = \begin{pmatrix} \varphi(\alpha) & \varphi(\gamma) \\ \varphi(\beta) & \varphi(\delta) \end{pmatrix} \cdot \begin{pmatrix} 19 & 19 \\ 14 & 21 \end{pmatrix}.$$

Der Kryptoanalyst kann also die Schlüsselnachricht N berechnen.

Die Situation ändert sich schon, wenn er nur ein Paar *(Klartext, Kryptotext)* hat. Nehmen wir an, man gibt ihm das oben bestimmte Paar (SAHARA,BBVVYY) und HHTQ als weiteren Kryptotext. Dem Kryptoanalysten ist die Schlüsselmatrix $N = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ nicht bekannt. Da das ersten Teilwort SA der Länge 2 durch BB verschlüsselt wird, ergibt sich

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 18 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

woraus die Gleichungen

$$18 \cdot a = 2 \quad \text{und} \quad 18 \cdot c = 2$$

resultieren. Die möglichen Lösungen müssen $a, c \in \{3, 16\}$ erfüllen. Betrachten wir nun das Paar HA, das durch VV verschlüsselt wird. Der Vektor $(7,0)^T$ muss also in $(21,21)^T$ überführt werden. Da $3 \cdot 7 = 21$ und $16 \cdot 7 = 112 = 8 \bmod 26$ gelten, bleiben nur noch die Möglichkeiten $a = c = 3$ übrig. Das Paar RA liefert keine weiteren Informationen, da $\begin{pmatrix} 3 & b \\ 3 & d \end{pmatrix} \begin{pmatrix} 17 \\ 0 \end{pmatrix} = \begin{pmatrix} 25 \\ 25 \end{pmatrix} \bmod 26$ gilt. Somit bleiben für die Entschlüsselung des Kryptotext HHTQ mit den Vektoren $(7,7)^T$ zu HH und $(19,16)^T$ zu TQ die Gleichungen

$$\begin{pmatrix} 3 & b \\ 3 & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 7 \end{pmatrix} \text{ und } \begin{pmatrix} 3 & b \\ 3 & d \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 19 \\ 16 \end{pmatrix}$$

zu lösen. Der Kryptoanalyst muss also aus vier Gleichungen sechs Unbekannte bestimmen; dabei hat er noch die Zusatzinformation, dass $\begin{pmatrix} 3 & b \\ 3 & d \end{pmatrix}$ eine inverse Matrix besitzen muss.

Letzte Bedingung ist erfüllt, wenn er $b = 2$ und $d = 1$ wählt, da die Zeilen der Matrix offensichtlich linear unabhängig sind. Damit ergeben sich die Gleichungen

$$\begin{pmatrix} 3 & 2 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 7 \end{pmatrix} \text{ und } \begin{pmatrix} 3 & 2 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 19 \\ 16 \end{pmatrix},$$

deren Lösungen

$$x = 11 \text{ und } y = 0 \quad \text{bzw.} \quad x' = 13 \text{ und } y' = 3$$

sind. Damit ergibt sich der Klartext LAND. Da dies ein richtiges deutsches Wort ist, vermutet der Kryptograph, dass der Klartext LAND ist. Er irrt aber, denn die obige Entschlüsselung mit der richtigen Schlüsselmatrix M ergibt LAST.

Unser zweites Beispiel für polyalphabetische Chiffrierungen sind die *Fairplay-Chiffren*. Bei diesen Chiffren wählen wir 25 Buchstaben des Alphabets, d.h. wir lassen einen selten vorkommenden Buchstaben fort, in einer Matrix mit 5 Zeilen und 5 Spalten an. Um sich diese Anordnung zu merken, kann man wie bei der Konstruktion merkbarer Permutationen vorgehen. Man merkt sich ein Schlüsselwort und ordnet die verbleibenden Buchstaben in alphabetischer oder umgekehrter alphabetischer Reihenfolge an.

Bei Fortlassung von J, erneuter Wahl von GREIFSWALD und alphabetischer Anordnung der restlichen Buchstaben ergibt sich die Fairplay-Matrix

G	R	E	I	F
S	W	A	L	D
B	C	H	K	M
N	O	P	Q	T
U	V	X	Y	Z

Die Verschlüsselung wird wie folgt vorgenommen. Wir unterteilen den Text erneut in Paare und verlangen, dass keines dieser Paare aus zwei gleichen Buchstaben besteht. Sollte dies der Fall sein, so fügen wir ein Vorkommen von Q ein. Sollte der dadurch entstehende Text nicht gerade Länge haben, fügen wir ein Q am Ende hinzu. Nun verschlüsseln wir den Text, indem wir die Paare des Klartextes durch Buchstabenpaare verschlüsseln. Sei $\alpha\beta$ ein Buchstabenpaar. Wir ersetzen es durch das Paar α', β', das wie folgt ermittelt wird:

1. Falls α und β in einer Zeile zu finden sind, so sind α' und β' die Buchstaben, die auf α bzw. β zyklisch in der Zeile folgen.

2. Falls α und β in einer Spalte zu finden sind, so sind α' und β' die Buchstaben, die auf α bzw. β zyklisch in der Spalte folgen.

3. Falls sich α und β sowohl in verschiedenen Zeilen als auch verschiedenen Spalten befinden, so wählen wir für α' und β' die Buchstaben so, dass

$$\begin{pmatrix} \alpha & \dots & \alpha' \\ \vdots & & \vdots \\ \beta' & \dots & \beta \end{pmatrix} \text{ oder } \begin{pmatrix} \alpha' & \dots & \alpha \\ \vdots & & \vdots \\ \beta & \dots & \beta' \end{pmatrix} \text{ oder } \begin{pmatrix} \beta & \dots & \beta' \\ \vdots & & \vdots \\ \alpha' & \dots & \alpha \end{pmatrix} \text{ oder } \begin{pmatrix} \beta' & \dots & \beta \\ \vdots & & \vdots \\ \alpha & \dots & \alpha' \end{pmatrix}$$

eine Teilmatrix der Fairplay-Matrix bilden.

Auf diese Art verschlüsseln wir unter Verwendung obiger Fairplay-Matrix das Wort SAHARAWIND durch WLPHEWLPTS. Man erkennt sofort, dass bei der Verschlüsselung für A drei verschiedene Buchstaben (L,H und W) benutzt werden.

Der befugte Empfänger kann mit der Kenntnis der Fairplay-Matrix leicht die Dechiffrierung vornehmen. Stehen die empfangene Buchstaben $\alpha'\beta'$ in einer Zeile, so stehen die Originalbuchstaben α und β in der Fairplay-Matrix in der Zeile zyklisch vor α' bzw. β'. Stehen beide Buchstaben in einer Spalte, so gehen wir analog in der Spalte vor, um α und β zu ermitteln. Stehen α' und β' weder in einer Spalte noch in einer Zeile, so finden wir α und β entsprechend Punkt 3 der Verschlüsselung.

Die Aufgabe des Kryptoanalysten ist nicht einfach. Während des Zweiten Weltkrieges hat das britische Militär Fairplay-Chiffrierungen verwendet, und diese wurden von der gegnerischen Seite nicht geknackt.

Als letztes Beispiel für polyalphabetische Substitutionschiffren behandeln wir die Vigenère-Chiffren.[2] Sie besteht im wesentlichen in einer periodischen Anwendung von Verschiebungschiffren.

Zuerst wird ein Schlüsselwort $\alpha_1\alpha_2\ldots\alpha_n$ gewählt. Ferner sei der Klartext $t_1t_2\ldots t_m$ gegeben. Dann verschlüsseln wir den Buchstaben t_i durch die Verschiebungschiffre $v_{\varphi(\alpha_j)}$ mit

$$j = \begin{cases} i \bmod n & i \text{ ist kein Vielfaches von } n \\ n & \text{sonst} \end{cases} .$$

Wenn wir die Tabelle aus Abbildung 5.4 verwenden, so bedeutet dies, dass wir den Buchstaben t_i durch sein Bild in der Zeile zum Buchstaben α_j mit $1 \leq j \leq n$ und $i = j + kn$ für ein gewisses $k \geq 0$ verschlüsseln. Wir wenden also die Verschiebungschiffren, die zu den Buchstaben des Schlüsselwortes gehören, periodisch an.

Unter Verwendung des Schlüsselwortes ABEND erhalten wir für den Klartext SAHARAWIND den Kryptotext SBLNUAXMAG. Erneut wird der Buchstabe A bei jedem seiner Vorkommen anders verschlüsselt.

Um den empfangenen Kryptotext zu dechiffrieren, hat der (befugte) Empfänger nur die Entschlüsselungen zu den Verschiebungschiffren ebenfalls periodisch anzuwenden. Bei Kenntnis des Schlüsselwortes ist dies eine leichte Aufgabe.

Dem Kryptoanalalysten mögen ein Kryptotext $\beta_1\beta_2\ldots\beta_m$ vorliegen. Der Kryptoanalyst hat für die Entschlüsselung eigentlich das Schlüsselwort zu ermitteln. Ihm reicht es aber, die Länge des Schlüsselwortes zu bestimmen. Hat er diese mit n ermittelt, so weiß er, dass die Buchstaben $s_is_{i+n}s_{i+2n}\ldots s_{i+un}$, wobei $1 \leq i \leq n$ und $i + un \leq m < i + (u + 1)n$ gelten, durch die gleiche Verschiebungschiffre gewonnen wurden. Durch eine Analyse der Häufigkeiten der Buchstaben, kann er nun sehr wahrscheinliche Kandidaten $v_{i,1}v_{i,2},\ldots v_{i,k_i}$ für diese Chiffre bestimmen. Durch Durchtesten dieser Kandidaten für $1 \leq i \leq n$ gelingt es ihm dann das Schlüsselwort zu erhalten und damit kann er dann in gleicher Weise wie der Empfänger die Entschlüsselung (auch neuer Kryptotexte) vornehmen. Daher ist das zentrale Problem für den Kryptoanalysten die Bestimmung der Länge des Schlüsselwortes.

Durch den deutschen Kryptoanalysten F.W. Kasinski wurde um 1860 vorgeschlagen, Teilwörter der Länge ≤ 3 zu suchen, die im Kryptotext mehrfach vorkommen. Dabei

[2]Sie wurden nach dem französischen Diplomaten Blaise de Vigenère (1523–1596) benannt, der diese Verschlüsselungen erstmals 1586 benutzte.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Abbildung 5.4: Vigenère-Tabelle

wird davon ausgegangen, dass bei einer Länge ≤ 3 eine gleiche Chiffrierung mit großer Wahrscheinlichkeit nur dann vorliegt, wenn das gleiche Wort des Klartext verschlüsselt wurde. Folglich ist die Differenz des Auftretens der gleichen Wörter ein Vielfaches der Schlüsselwortlänge.

Als Beispiel betrachten wir den Kryptotext aus Abbildung 5.5, in dem die mehrfach auftretende Teilwörter der Länge ≥ 3 unterstrichen sind.. Man erhält daraus die folgende Tabelle:

Folge	Abstand
JTD	$50 = 2 \cdot 5^2$
VIQM	$265 = 5 \cdot 53$
TDMHZGNMWK	$90 = 2 \cdot 3^2 \cdot 5$
MWK	$75 = 3 \cdot 5^2$
ZHUM	$40 = 2^3 \cdot 5$
KAH	$128 = 2^7$

Nimmt man den größten gemeinsamen Teiler der Abstände so ergibt sich 1. Wenn das

U E Q P C V C K A H V N R Z U R N L A O K I R V G
J T D V R V R I C V I D L M Y I Y S B C C O J Q S
Z N Y M B V D L O K F S L M W E F R Z A V I Q M F
J T D I H C I F P S E B X M F F T D M H Z G N M W
K A X A U V U H J H N U U L S V S J I P J C K T I
V S V M Z J E N Z S K A H Z S U I H Q V I B X M F
F I P L C X E Q X O C A V B V R T W M B L N G N I
V R L P F V T D M H Z G N M W K R X V R Q E K V R
L K D B S E I P U C E A W J S B A P M B V S Z C F
U E G I T L E U O S J O U O H U A V A G Z E Z I S
Y R H V R Z H U M F R R E M W K N L K V K G H A H
F E U B K L R G M B J I H L I I F W M B Z H U M P
L E U W G R B H Z O L C K V W T H W D S I L D A G
V N E M J F R V Q S V I Q M U V S W M Z C T H I I
W G D J S X E O W S J T K I H K E Q

Abbildung 5.5: Ein Kryptotext mit mehrfach auftretenden Teilwörtern der Länge ≥ 3

Schlüsselwort aber die Länge 1 hat, so erfolgt die Verschlüsselung eine einfache Verschiebechiffre. Dies ist aber nicht anzunehmen, da eine Vigenère-Chiffre vorliegen soll. Es ist also anzunehmen, dass eines der Teilwörter nicht durch Verschlüsselung des gleichen Wortes sondern zufällig entstanden ist. Hierfür ist KAH der erste Kandidat, da die Primfaktoren des zu diesem Wort gehörenden Abstands beide nicht Primfaktoren anderer Abstände sind. Ausgehend von den verbleibenden Wörtern ist ein Schlüsselwort der Länge 5 zu erwarten, da nun 5 der größte gemeinsame Teiler ist.

Eine andere Methode zur Berechnung der Länge des Schlüsselwortes stammt von William Friedman, der sie 1925 vorschlug. Hierbei wird die Länge nur approximiert, aber dadurch werden gewisse Werte weitgehend ausgeschlossen.

Gegeben sei ein Kryptotext der Länge n. Für einen Buchstaben α, sei n_α die Anzahl seines Auftretens im gegebenen Kryptotext. Wir bestimmen nun die Anzahl des Auftretens von Paaren nicht notwendig benachbarter gleicher Buchstaben. Für einen Buchstaben α ist diese Zahl durch $n_\alpha(n_\alpha - 1)/2$ gegeben, da der erste Buchstabe an n_α Stellen und der zweite Buchstabe an $n_\alpha - 1$ Stellen auftauchen kann und die Reihenfolge keine Rolle spielt. Für die Anzahl des Auftretens von Paaren gleicher Buchstaben im Kryptotext erhalten wir daher

$$I = \sum_{\alpha \in alph} \frac{n_\alpha(n_\alpha - 1)}{2}, .$$

Nehmen wir nun den Klartext. In ihm taucht der Buchstabe α mit der Häufigkeit p_α entsprechend der Abbildung 5.2 auf. Tritt ein Buchstabe α darin mit der Häufigkeit p_α auf (man beachte, dass wir diese Häufigkeit nicht kennen, da wir den Text nicht kennen) so hat die Wahrscheinlichkeit für das Auftretens eines Paares des Buchstaben α den Wert p_α^2 (dies ist eigentlich nur bei Beachtung der Reihenfolge in den Paaren richtig, aber bei großem n kann dieser Unterschied vernachlässigt werden) und damit für das Auftreten eines Paares $\sum_{\alpha \in alph} p_\alpha^2$. Falls die Buchstaben so verteilt sind, wie dies nach

Abbildung 5.2 bei deutschen Texten der Fall ist, so ergibt sich für die Wahrscheinlichkeit $\sum_{\alpha \in alph} p_\alpha^2 = 0,0762$. Sind die Buchstaben dagegen im Klartext gleichverteilt, so wäre $p_\alpha = 1/26$ für alle α und damit $\sum_{\alpha \in alph} p_\alpha^2 = 1/26 = 0{,}0385$.

Nehmen wir nun an, dass das Schlüsselwort der Vigenère-Chiffre die Länge l hat. Wir schreiben den Text nun so, dass jede Zeile (vielleicht bis auf die letzte) genau l Buchstaben enthält. In jeder der so entstehenden Spalten stehen daher n/l Buchstaben (wir nehmen hierfür Ganzzahligkeit an, was bei hinreichender Länge n gemacht werden kann). Wir betrachten nun die Häufigkeit für das Auftreten eines Paaren von Buchstaben. Wir wählen dazu eine Position aus. Damit ist auch eine Spalte gewählt. Innerhalb der Spalte können wir noch $\frac{n}{l} - 1$ Positionen für den zweiten Buchstaben auswählen. Da es auf die Reihenfolge nicht ankommt, liefert dies

$$\frac{n \cdot (\frac{n}{l} - 1)}{2} = \frac{n(n-l)}{2l}$$

Möglichkeiten. Für die Anzahl der Paare mit dem zweiten Buchstaben in einer anderen Spalte ergeben sich (da nun die n/l Positionen der ersten Spalte entfallen)

$$\frac{n \cdot (n - \frac{n}{l})}{2} = \frac{n^2(l-l)}{2l}$$

Möglichkeiten.

Die Buchstaben einer Spalten werden durch die gleiche Verschiebechiffre geliefert. Die Buchstabenverteilung der Spalten des Kryptotextes entspricht daher der des Klartextes, da bei einer Verschiebechiffre die Zuordnung der Buchstaben eineindeutig ist. Für das Auftreten eines Paares gleicher Buchstaben in einer Spalte erhalten wir daher (bei einem deutschen Text) $0,0762 \cdot \frac{n(n-l)}{2l}$ Möglichkeiten. Entstammen die Buchstaben dagegen verschiedenen Spalten, so können wir davon ausgehen, dass sie relativ gleichverteilt sind (und bei Gleichverteilung der Buchstaben im Schlüsselwort wäre Gleichverteilung im Kryptotext auch gegeben). Daher gibt $0,0385 \cdot \frac{n^2(l-1)}{2l}$ die Anzahl der Möglichkeiten für das Auftreten eines Paares gleicher Buchstaben in verschiedenen Spalten. Damit ist die Gesamtzahl der zu erwartenden Paare gleicher Buchstaben im Kryptotext

$$I' = 0,0762 \cdot \frac{n(n-l)}{2l} + 0,0385 \cdot \frac{n^2(l-1)}{2l}\,.$$

Offensichtlich ist zumindest angenähert $I = I'$ zu erwarten. Damit erhalten wir

$$I = 0,0762 \cdot \frac{n(n-l)}{2l} + 0,0385 \cdot \frac{n^2(l-1)}{2l}$$

und daraus

$$l = \frac{0,0377n}{(n-l)I - 0,0385n + 0,0762}\,.$$

Da I durch Auszählen am Kryptotext ermittelt werden kann, ist somit l berechenbar.

Für unseren Beispieltext aus Abbildung 5.5 erhalten wir $I = 5924$ und damit $l = 6,5$. Somit ist die Länge des Schlüsselwortes in der Nähe von $6,5$ zu suchen. Unter Berücksichtigung unserer Ergebnisse aus der Methode von Kasiski liegt damit 5 als Länge des

Schlüsselwortes nahe. Wir bestimmen daher für i mit $0 \leq i \leq 4$, die am häufigsten auftauchenden Buchstaben in allen Spalten mit einer Nummer k mit $k = i \bmod 5$. Hierfür ergeben sich V, E, H, M und S. Damit entsprechen diese Buchstaben bei den fünf Verschiebechiffren jeweils dem Buchstaben E. Hieraus resultiert das Schlüsselwort RADIO. Nimmt man nun hiermit die Dechiffrierung vor, so ergibt sich der Text aus Abbildung 5.6 oder in lesbarerer Form

```
D E N H O E C H S T E N O R G A N I S A T I O N S
S T A N D E R F U H R D I E K R Y P T O L O G I E
I N V E N E D I G W O S I E I N F O R M E I N E R
S T A A T L I C H E N B U E R O T A E T I G K E I
T A U S G E U E B T W U R D E E S G A B S C H L U
E S S E L S E K R E T A E R E D I E I H R B U E R
O I M D O G E N P A L A S T H A T T E N U N D F U
E R I H R E T A E T I G K E I T R U N D Z E H N D
U K A T E N I M M O N A T B E K A M E N E S W U R
D E D A F U E R G E S O R G T D A S S S I E W A E
H R E N D I H R E R A R B E I T N I C H T G E S T
O E R T W U R D E N S I E D U R F T E N I H R E B
U E R O S A B E R A U C H N I C H T V E R L A S S
E N B E V O R S I E E I N E G E S T E L L T E A U
F G A B E G E L O E S T H A T T E N
```

Abbildung 5.6: Der entschlüsselte Text zum Krytotext aus Abbildung 5.5

DEN HÖCHSTEN ORGANISATIONSSTAND ERFUHR DIE KRYPTOLOGIE IN VENEDIG, WO SIE IN FORM EINER STAATLICHEN BÜROTÄTIGKEIT AUSGEÜBT WURDE. ES GAB SCHLÜSSELSEKRETÄRE, DIE IHR BÜRO IM DOGENPALAST HATTEN UND FÜR IHRE TÄTIGKEIT RUND ZEHN DUKATEN IM MONAT BEKAMEN. ES WURDE DAFÜR GESORGT, DASS SIE WÄHREND IHRER ARBEIT NICHT GESTÖRT WURDEN. SIE DURFTEN IHRE BÜROS ABER AUCH NICHT VERLASSEN, BEVOR SIE EINE GESTELLTE AUFGABE GELÖST HATTEN.

5.3 Der Data Encryption Standard

In diesem Abschnitt behandeln wir den Data Encryption Standard, der 1977 vom National Bureau of Standards in den USA angekündigt und von IBM realisiert wurde. Hierbei geht es darum 64-Bit-Folgen (Wörter der Länge 64 über $\{0, 1\}$) zu übertragen und als Schlüssel dient dabei eine (geheimzuhaltende) 56-Bit-Folge. Die Verschlüsselung ist dann standardisiert. Dafür wurden sehr effektive Hardware- und Softwarerealisierungen geschaffen. Die Methode erfuhr seit den siebziger Jahren einige Veränderungen. Wir werden hier aber als Prototyp die Originalvariante behandeln.

Die Chiffrierung ist ein iterativer Prozess mit 16 Iterationsschritten. Innerhalb dieser Schritte werden zwei wesentliche Operationen benutzt. Die erste Operation ist die Überführung der Wörter $a_1a_2\dots a_k$ und $b_1b_2\dots b_k$ in das Wort $(a_1\oplus b_1)(a_2\oplus b_2)\dots(a_k\oplus b_k)$. Die andere besteht in der Transformation des Wortes $a_1a_2\dots a_k$ in das Wort $a_{i_1}a_{i_2}\dots a_{i_l}$, wobei wir dann stets nur die Folge $i_1, i_2, \dots, i_l$ angeben, die eine Auswahl (meist eine Permutation) von der Folge $1, 2, \dots, k$ ist.

Der Algorithmus zur Verschlüsselung des Klartextes in den Kryptotext ist in Abbildung 5.7 dargestellt. Dabei bedeuten die umkreisten Knoten jeweils eine Transformation, während die durch ein Rechteck angegeben Knoten nur das aktuelle Zwischenergebnis enthalten.

Wir haben nun die einzelnen angewendeten Operationen zu erklären. Bei IP handelt es sich um die initiale Permutation der Bits. Sie ist durch

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

gegeben. Das Ergebnis sei $t_1t_2\dots t_{64}$. Bei der nachfolgenden Aufspaltung erhält der linke Strang die ersten 32 Bits, also das Wort $t_1t_2\dots t_{32}$, während an den linken Strang das Wort $t_{33}t_{34}\dots t_{64}$ der letzten 32 Bits übergeben wird. Vor Realisierung der IP inversen Transformation werden die beiden Wörter der Länge 32 wieder zu einem Wort der Länge 64 zusammengefügt (dabei ist die Vertauschung davor zu beachten).

Die Funktion f wird durch das Schema in Abbildung 5.8 berechnet. Als Eingabe werden eine 32-Bit-Folge R_{i-1} und eine 48-Bit-Folge K_i verarbeitet (i bezieht sich dabei auf den Iterationsschritt im Algorithmus). Zuerst wird die Operation E auf R_{i-1} angewendet, wodurch die 32-Bit-Folge R_{i-1} auf eine 48-Bit-Folge A erweitert wird.. Welches Element aus R_{i-1} an welcher Stelle in A steht, zeigt das folgende Schema:

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

Danach erfolgt die bitweise Addition von K_i. Das dadurch entstehende Resultat ist eine 48-Bit-Folge $f_1f_2\dots f_{48}$, die nun in acht Teilfolgen $U_i = f_{6(i-1)+1}f_{6(i-1)+2}\dots f_{6i}$, $1 \le i \le 8$, der Länge 6 unterteilt wird. Die Folge U_i wird an S_i übergeben und wie folgt verarbeitet. Zuerst werden $f_{6(i-1)+1}f_{6i}$ und $f_{6(i-1)+2}f_{6(i-1)+3}f_{6(i-1)+4}f_{6i-1}$ (man beachte $6i-1 = 6(i-1)+5$) als binäre Darstellungen von Zahlen a und b mit $0 \le a \le 3$ bzw. $0 \le b \le 15$ interpretiert. Für das Paar (a, b) wird entsprechend der Tabelle aus Abbildung 5.9 eine Zahl c mit $0 \le c \le 15$ ermittelt, deren zugehörige Binärfolge U_i' der Länge 4 als Ausgabe von S_i dient. Die Verkettung $U_1'U_2'\dots U_8'$ mit der Länge 32 wird durch P wie folgt permutiert:

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

Es bleibt zu klären, wie die 16 Eingaben K_i, $1 \le i \le 16$, in den 16 Iterationsschritten ermittelt werden. Das Verfahren ist in Abbildung 5.10 dargestellt. Als Eingabe dient der Schlüssel K. Er ist eine 56-Bit-Folge, die zuerst entsprechend dem Schema

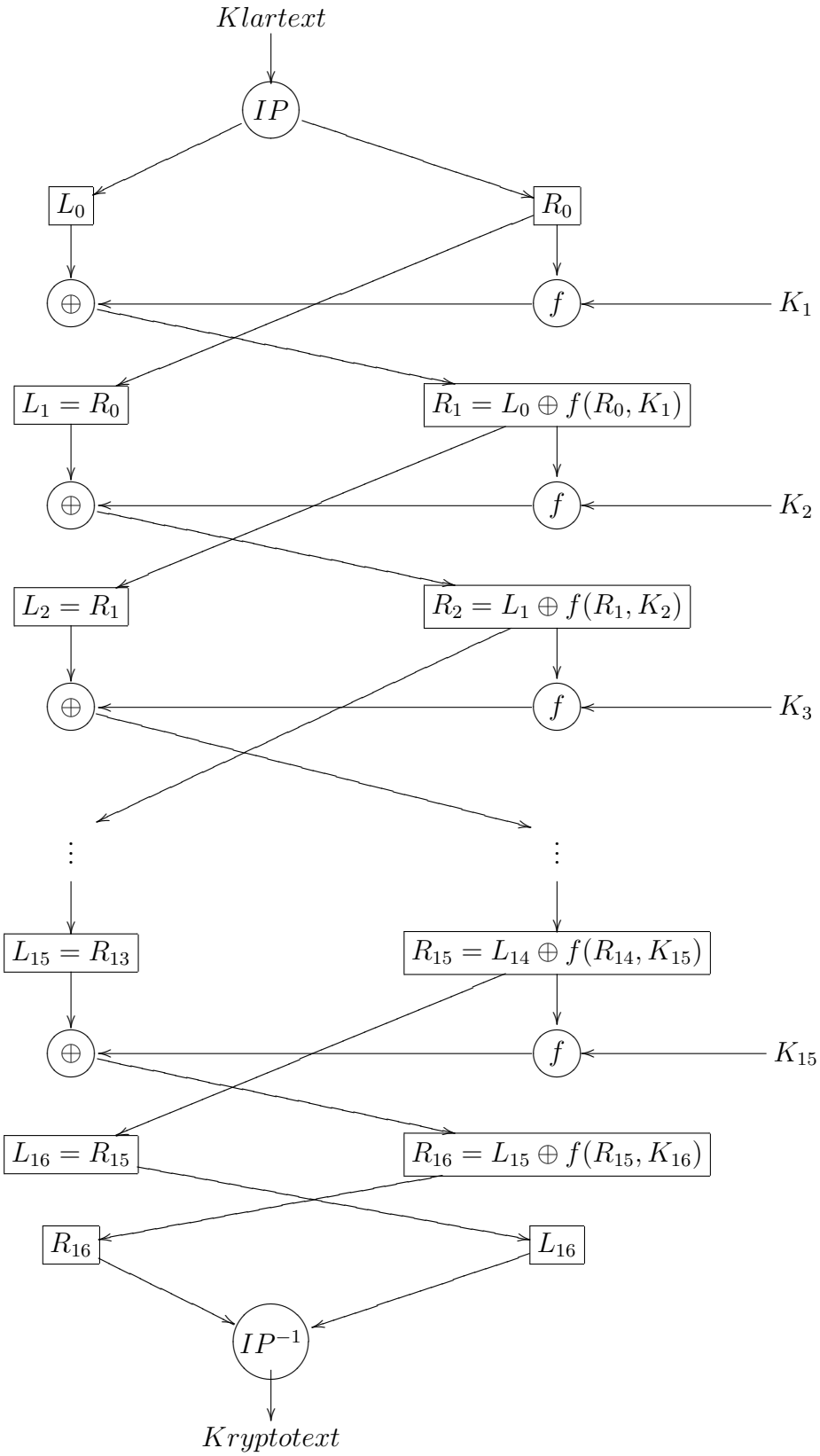


Abbildung 5.7: DES-Verschlüsselungsalgorithmus

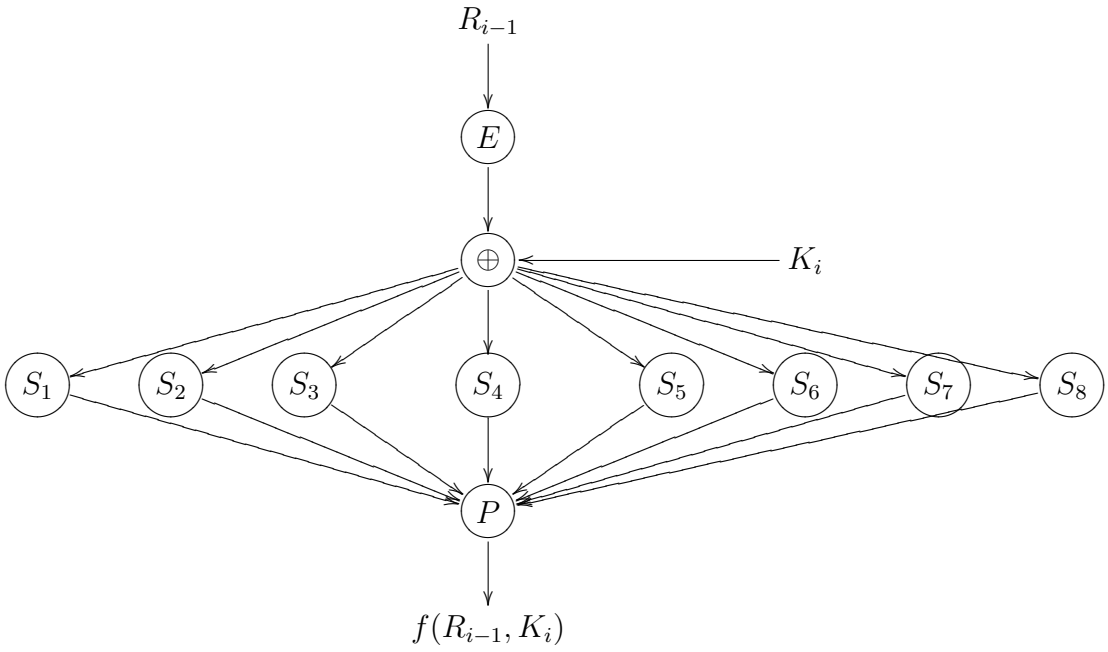


Abbildung 5.8: Berechnungsschema der Funktion f

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

permutiert wird. Danach wird sie in zwei Teilwörter der Länge 28 aufgeteilt. Es schließen sich 16 gleichartige Berechnungen für die K_i, $1 \leq i \leq 16$ an. Zuerst erfolgt eine zyklische Verschiebung innerhalb der beide Wörter der Länge 28 um jeweils $j(i)$ Elemente nach links. Die folgende Tabelle gibt die Größen $j(i)$ an.

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

Dann erfolgt eine Verkettung der so entstandenen Wörter. Aus dem entstandenen Wort der Länge 56 streicht PC_2 die Buchstaben an den Positionen 9, 18, 22, 25, 38, 43 und 54 und permutiert die verbleibenden Buchstaben. Es erfolgt damit eine Auswahl entsprechend dem folgenden Schema:

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

Dadurch entsteht das Wort K_i der Länge 48 im i-ten Schritt.

Wir kommen nun zur Dechiffrierung. Dazu wird das gleiche Schema verwendet, jedoch werden die Wörter K_i, $1 \leq i \leq 16$ in umgekehrter Reihenfolge angewendet. Zuerst wenden wir auf einen Buchstaben a_K des Kryptotextes, d.h. einem 64-Bit-Wort, die Permutation IP an. Da die Verschlüsselung mit der Anwendung von IP^{-1} abschloss, liefert dies die

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	9	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	8	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Abbildung 5.9: Die Transformationen S_i, $1 \leq i \leq 8$

Teilwörter R_{16} und L_{16}, die nun entsprechend dem Verschlüsselungsalgorithmus (mit K_i in umgekehrter Reihenfolge) umgewandelt werden. Es ergeben sich dann mit Induktion von 16 nach 1 für i, $1 \leq i \leq 16$,

$$L_i = R_{i-1}$$

und

$$\begin{aligned} R_i \oplus f(L_i, K_i) &= L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(L_i, K_i) \\ &= L_{i-1} \oplus f(L_i, K_i) \oplus f(L_i, K_i) \end{aligned}$$

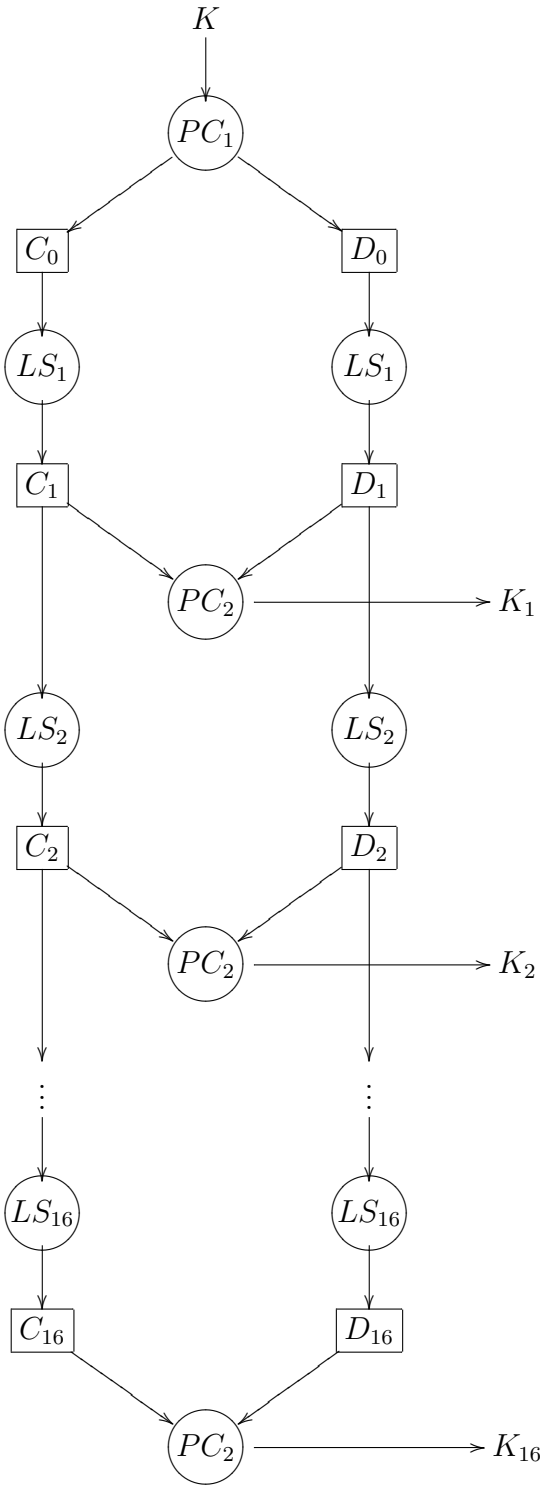


Abbildung 5.10: Berechnung der K_i, $1 \leq i \leq 16$

$$= \quad L_{i-1}$$

Abschließend wird auf L_0R_0 noch IP^{-1} angewendet, wodurch der Buchstabe des Eingabetextes entsteht, der in a_K überführt wurde.

Damit gibt es auch für die Dechiffrierung eine gute Methode.

Der Kryptoanalyst hat kaum andere Möglichkeiten als alle denkbaren Schlüsselwörter K der Länge 56 zu testen, um bei der Entschlüsselung einen sinnvollen (dann hoffentlich richtigen) Klartext zu erhalten. Dies erfordert einen enorm hohen Aufwand.

5.4 Steganographie

Steganographie ist eine Methode, bei der der Klartext in einem umfangreichen Text versteckt wird und nur durch den Schlüssel lesbar gemacht werden kann. Da hierbei keine Verschlüsselung vorgenommen wird, kann dieses Verfahren nur bedingt der Kryptologie zugeordnet werden. Wir wollen hier nur ein solches Beispiel behandeln, das vom französischen Kardinal RICHELIEU (1585 – 1642) vielfach angewendet wurde. Wir betrachten den folgenden Text:

D E R I N D E R K A M V O R D E M

L E T T E N E R S A H D E N U N G

A R N U N D B E Z A H L T E D A S

G E L D I M K L E I N E N R A U M

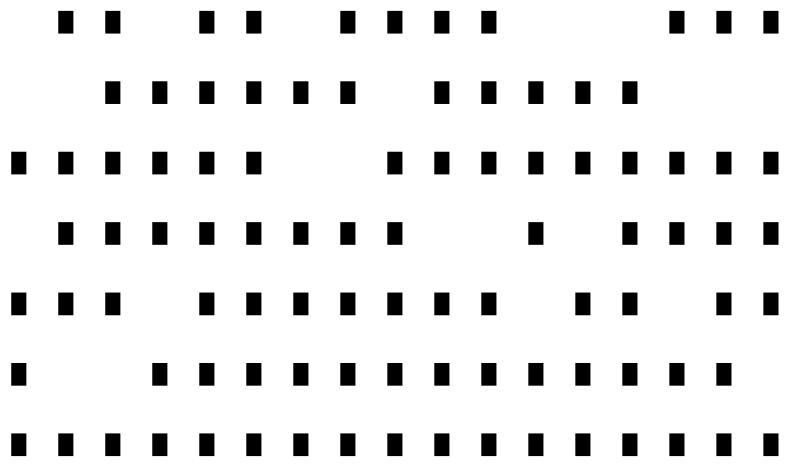
D E S T U R M E S A U F D E R B L

A U E N L I E G E S A S S D E R H

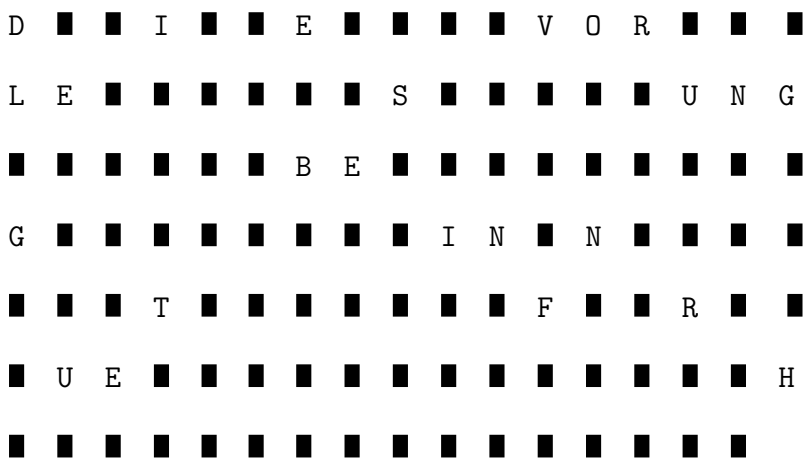
E S S E G E L A N G W E I L T

Die damit übermittelte Nachricht wird nur dann lesbar, wenn man im Besitz einer Schablone ist, die alle überflüssigen Buchstaben abdeckt. Die verbleibenden sichtbaren Buchstaben ergeben dann den Text.

In unserem Beispiel ist folgende Schablone zu benutzen. Dabei haben wir die abzudeckenden Buchstaben durch ■ markiert.



Es ergibt sich dann folgende Situation



und damit der Klartext DIE VORLESUNG BEGINNT FRÜH.

Kapitel 6

Perfekte Sicherheit

Wir haben bei den monoalphabetischen Chiffren gesehen, dass der Kryptoanalyst wichtige Informationen, wie z. B. die Häufigkeit der Buchstaben im Kryptotext ausnutzen kann, um die Schlüsselinformation zu erhalten und den Kryptotext zu entschlüsseln. In diesem Abschnitt wollen wir solche Chiffriersysteme behandeln, bei denen der Kryptograph keine Erkenntnisse aus der Kenntnis des Kryptotextes gewinnen kann.

Wir führen dazu folgende Bezeichnungen ein. Mit $\mathcal{T}$ bezeichnen wir die Menge der (möglichen) Klartexte. $\mathcal{S}$ sei eine Menge von Verschlüsselungsfunktionen, bei denen jede Verschlüsselungsfunktion durch einen Schlüssel bestimmt sei. Mit $\mathcal{K}$ bezeichnen wir die Menge der Kryptotexte, die aus Klartexten durch Verschlüsselungen aus $\mathcal{S}$ entstehen können. Es gilt also

$$\mathcal{K} = \{\tau(t) \mid t \in \mathcal{T}, \tau \in \mathcal{S}\}.$$

Wir wollen davon ausgehen, dass es für jede Verschlüsselung $\tau \in \mathcal{S}$ und jeden Kryptotext $k \in \mathcal{K}$ höchstens einen Klartext $t \in \mathcal{T}$ mit $\tau(t) = k$ gibt (ansonsten könnte auch der befugte Empfänger den Klartext nicht eindeutig aus dem Kryptotext rekonstruieren). Damit gilt offensichtlich

$$\#(\mathcal{T}) \leq \#(\mathcal{K}). \tag{6.1}$$

Wir bezeichnen noch mit $p(t)$ die Wahrscheinlichkeit für das Auftreten des Klartextes $t \in \mathcal{T}$. Außerdem sei $p_k(t)$ die Wahrscheinlichkeit dafür, dass der Kryptotext $k \in \mathcal{K}$ durch Chiffrierung des Textes $t \in \mathcal{T}$ entstanden ist.

Definition 6.1 *Wir sagen, dass die Verschlüsselungen aus $\mathcal{S}$ (bzw. die Schlüssel zu $\mathcal{S}$) hinsichtlich $\mathcal{T}$ und $\mathcal{K}$ perfekte Sicherheit bieten, falls*

$$p_k(t) = p(t) \text{ für jeden Kryptotext } k \in \mathcal{K} \text{ und jeden Klartext } t \in \mathcal{T}$$

gilt.

Intuitiv bedeutet dies, dass die Kenntnis des Textes k dem Kryptoanalysten nichts hilft, da sich dadurch sein Kenntnisstand hinsichtlich der Frage, ob t das Urbild des Textes ist, nicht ändert.

Ein einfaches Beispiel für ein System mit perfekter Sicherheit bilden die Verschiebechiffren hinsichtlich der Mengen $\mathcal{T}$ und $\mathcal{K}$, die beide jeweils nur aus den Buchstaben aus *alph* bestehen. Offensichtlich gilt dabei $p(t) = \frac{1}{26}$ für jeden Text aus $t \in \mathcal{T}$ (Buchstaben t

aus *alph*). Liegt dem Kryptoanalysten ein Text $k \in \mathcal{K}$ vor, so ist jeder Klarbuchstabe gleichwahrscheinlich als Urbild von k. Somit gilt auch $p_k(t) = \frac{1}{26}$.

Sei nun $\mathcal{S}$ ein Schlüsselsystem mit perfekter Sicherheit bez. gewisser Mengen $\mathcal{T}$ und $\mathcal{K}$ von Klar- und Kryptotexten. Dann gilt für jeden Kryptotext $k \in \mathcal{K}$ und für jeden Klartext $t \in \mathcal{T}$, $p_k(t) = p(t)$. Weiterhin ist $p(t) > 0$ für jeden Text $t \in T$ (wäre die Wahrscheinlichkeit $p(t) = 0$, so könnten wir diesen Text einfach aus $\mathcal{T}$ streichen). Damit ist $p_k(t) > 0$. Dies bedeutet, dass folgende Aussage gilt.

Fakt 1 *Bietet das Schlüsselsystem $\mathcal{S}$ perfekte Sicherheit bez. $\mathcal{T}$ und $\mathcal{K}$, so gibt es zu jedem Klartext $t \in \mathcal{T}$ und jedem Kryptotext $k \in \mathcal{K}$ eine Verschlüsselung $\tau \in \mathcal{S}$ so, dass $\tau(t) = k$ gilt.*

Wir betrachten nun zu einem Klartext t alle Verschlüsselungen, d.h. die Menge $U(t) = \{\tau(t) \mid \tau \in \mathcal{S}\}$. Nach Definition gilt $\#(U(t)) = \#(\mathcal{S})$. Außerdem gilt wegen Fakt 1 aber noch $\#(U(t)) \geq \#(\mathcal{K})$. Damit erhalten wir unter Beachtung von (6.1) den nächsten Fakt.

Fakt 2 *$\#(\mathcal{S}) \geq \#(\mathcal{K}) \geq \#(\mathcal{T})$ gilt für jedes Schlüsselsystem $\mathcal{S}$ mit perfekter Sicherheit bez. $\mathcal{T}$ und $\mathcal{K}$.*

Der nächste Satz kann als Umkehrung der beiden Fakten aufgefasst werden.

Satz 6.1 *Es sei $\mathcal{S}$ ein Schlüsselsystem mit $\#(\mathcal{T}) = \#(\mathcal{K}) = \#(\mathcal{S})$, in dem alle Schlüssel mit der gleichen Wahrscheinlichkeit vorkommen und in dem es zu jedem Klartext t und jedem Kryptotext k genau eine Transformation $\tau \in \mathcal{S}$ gibt, für die $\tau(t) = k$ gilt. Dann bietet $\mathcal{S}$ perfekte Sicherheit bez. $\mathcal{T}$ und $\mathcal{K}$.*

Beweis. Aufgrund des Bayesschen Satzes gilt

$$p_k(t) = \frac{p(t) \cdot p_t(k)}{p(k)}\,, \tag{6.2}$$

wobei $p(k)$ die Wahrscheinlichkeit dafür ist, dass ein Kryptotext mit k übereinstimmt, und $p_t(k)$ die Wahrscheinlichkeit dafür ist, dass t in k überführt wird. Aufgrund unserer Voraussetzung ist jeder Schlüssel gleichwahrscheinlich, womit jeder Kryptotext aus einem Klartext mit gleicher Wahrscheinlichkeit entsteht. Dies liefert $p_t(k) = \frac{1}{\#(\mathcal{S})}$. Außerdem ist jeder Kryptotext gleichwahrscheinlich, denn jeder Text entsteht mit gleicher Wahrscheinlichkeit aus einem Klartext unter Verwendung gleichwahrscheinlicher Schlüssel. Folglich ist $p(k) = \frac{1}{\#(\mathcal{K})}$. Beachten wir nun die Voraussetzung, dass $\#(\mathcal{S}) = \#(\mathcal{K})$ gilt, so erhalten wir $p_t(k) = p(k)$ und damit aus (6.2) die gewünschte Gleichheit $p_k(t) = p(t)$ für alle $k \in \mathcal{K}$ und alle $t \in \mathcal{T}$. □

Als ein Beispiel behandeln wir das *one-time Pad*. Dabei bestehen die Klartexte aus allen Bit-Folgen der Länge n und als Menge der Schlüssel verwenden wir die gleiche Menge. Die Verschlüsselung erfolgt durch bitweises Addieren modulo 2. Aus dem Klartext $t = t_1 t_2 \dots t_n \in \{0,1\}^n$ und dem Schlüssel $s_1 s_2 \dots s_n \in \{0,1\}^n$ entsteht der Kryptotext $(t_1 \oplus s_1)(t_2 \oplus s_2) \dots (t_n \oplus s_n)$. Damit liegt aber auch jeder Kryptotext in $\{0,1\}^n$. Weiterhin ergibt sich, dass zu gegebenen $t \in \{0,1\}^n$ und gegebenem $k \in \{0,1\}^n$ ein Schlüssel $s = t \oplus k \in \{0,1\}^n$ mit $t \oplus s = k$ existiert. Hieraus ergibt sich als erstes, dass jede

Bit-Folge aus $\{0,1\}^n$ als Kryptotext auftaucht, womit ebenfalls $\mathcal{K} = \{0,1\}^n$ gilt. Da die Mengen $\mathcal{T}$, $\mathcal{S}$ und $\mathcal{K}$ damit identisch sind, stimmen insbesondere ihre Kardinalitäten überein. Auch die anderen Voraussetzungen von Satz 6.1 sind erfüllt. Folglich handelt es sich bei dem System mit $\mathcal{T} = \mathcal{S} = \mathcal{K} = \{0,1\}^n$ um ein System mit perfekter Sicherheit.

Wir merken an, dass wegen $(t \oplus s) \oplus s = t$ die Dechiffrierung mit der Chiffrierung übereinstimmt.

Auf den ersten Blick ist dies natürlich kein brauchbares Kryptosystem, denn der geheimzuhaltende Schlüssel ist von gleicher Länge wie der Klartext. Wenn der Schlüssel nun problemlos vom Sender an den Empfänger übermittelt werden kann, so wäre dies ja auch für den Klartext zutreffend, und man könnte gleich diesen sicher senden. Jedoch gibt es einen kleinen Unterschied. Der Klartext ist in der Regel zu einem bestimmten Moment zu verschicken, während der Schlüsselaustausch früher zu einem passenden Moment oder auch über einem anderen Kanal verschickt werden kann. So könnten z.B. die Schlüssel durch einen Diplomaten überbracht werden, während der Klartext eine wichtige Information zu einem bestimmten Zeitpunkt innerhalb von Verhandlungen sein kann. Dabei kann dann der früher übergebene Schlüssel benutzt werden.

Für die perfekte Sicherheit ist es erforderlich, dass die Schlüssel gleichwahrscheinlich sind. Dies bedeutet in der Praxis, dass es sich beim Schlüssel um eine möglichst zufällige Folge handelt. Dies erfordert dann aber bei der Schlüsselübermittlung die Übergabe des gesamten Wortes der Länge n. Einfacher wäre es Folgen zu finden, die den Eindruck einer zufällig erzeugten Folge machen, aber durch einen (einfach zu merkenden und einfach zu realisierenden) Algorithmus gewonnen werden können.

Wir behandeln hier eine derartige Möglichkeit, bei der wir die Folge durch ein Schieberegister erzeugen.

Definition 6.2 *i) Unter einem Schieberegister der Länge m verstehen wir*

- *m Speicherelemente k_1, k_2, $\ldots k_m$, von denen jedes zu einem Zeitpunkt t, $t \geq 0$, genau ein Element $k_i(t) \in \{0,1\}$ enthält,*
- *m Konstanten $c_1, c_2, \ldots, c_m$ aus $\{0,1\}$ und*
- *m Werte $x_1, x_2, \ldots, x_m$ aus $\{0,1\}$.*

ii) Die Speicherelemente sind initial mit den Werten belegt, d.h. für $1 \leq i \leq m$ gilt $k_i(0) = x_i$.

Die Veränderung in einem Speicherelement erfolgt taktweise entsprechend den folgenden Formeln:

$$\begin{aligned} k_1(t+1) &= c_1 k_1(t) \oplus c_2 k_2(t) \oplus \ldots \oplus c_m k_m(t)\,, \\ k_i(t+1) &= k_{i-1}(t) \text{ für } 2 \leq i \leq m,\ t \geq 0\,. \end{aligned}$$

iii) Die von einem Schieberegister ausgegebene Folge ist $k_t(0)k_t(1)k_t(2)\ldots$

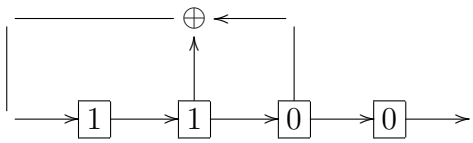


Abbildung 6.1: Veranschaulichung eines Schieberegisters

Die Speicherelemente werden auch Register genannt.

Als Beispiel betrachten wir das Schieberegister der Länge 4 mit

$$c_1 = 0,\ c_2 = 1,\ c_3 = 1,\ c_4 = 0 \quad \text{und} \quad x_1 = x_2 = 1,\ x_3 = x_4 = 0\,.$$

Das Schieberegister lässt sich wie in Abbildung 6.1 veranschaulichen. Dabei sind in der unteren Zeile die Register mit der Anfangsbelegung angegeben. Ferner gehen wir davon aus, dass die Addition (im Allgemeinen die Additionen) ohne Verzögerung arbeiten. Wir haben für die Berechnung des Wertes im Register k_1 nur die Addition von den Inhalten der Register k_2 und k_3 benutzt, da nach der Wahl der c_i nur diese Register einen Beitrag zur Summe $c_1k_1(t) \oplus c_2k_2(t) \oplus c_3k_3(t) \oplus c_4k_4(t)$ leisten. Es ergeben sich dann in den Takten die folgenden Werte:

Takt t	$k_1(t)$	$k_2(t)$	$k_3(t)$	$k_4(t)$	Ausgabe
0	1	1	0	0	0
1	1	1	1	0	0
2	0	1	1	1	1
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	0	0	0
6	1	0	1	0	0
7	1	1	0	1	1
8	0	1	1	0	0
9	0	0	1	1	1
10	1	0	0	1	1

Wir erkennen, dass die Register in den Takten 3 und 9 die gleichen Inhalte haben. Folglich stimmen auch die Inhalte in den Takten 4 und 10 überein, da die Berechnung der Inhalte im nächsten Takt eindeutig ist. Somit haben wir ein periodisches Verhalten, nach jeweils 6 Takten erhalten wir das gleiche Resultat, wobei der Ausgangstakt ≥ 3 sein muss. Dieses Verhalten überträgt sich selbstverständlich auch auf die Ausgabefolge. Folglich erzeugt unser Schieberegister die Ausgabefolge

$$001(110010)^*\,.$$

Dieses periodische Verhalten muss sich bei jedem Schieberegister der Länge m einstellen, denn die Tupel der Registerinhalte sind in $\{0,1\}^n$ enthalten, und diese Menge hat nur 2^m Elemente. Es ist beim Entwurf des Kryptosystems, das eine Bit-Folge verwendet, die von einem Schieberegister erzeugt wird, die Länge m möglichst groß und die Koeffizienten c_i und die Anfangsbelegung x_i, $1 \leq i \leq m$ so zu wählen, dass eine möglichst große Periode entsteht. Wir geben hier ohne Beweis an, dass man bei gegebenem m die Parameter

stets so wählen kann, dass sich eine Periode der Länge $2^m - 1$ ergibt. Dadurch wirkt die erzeugte Folge relativ zufällig.

Kommen wir nun zur Situation des Kryptoanalysten. Kennt er ein Paar *(Klartext, Kryptotext)* der Länge $2m$, wobei m die Länge des Schieberegisters ist, so kann er die Gleichungen, die das Schieberegister beschreiben, in einfacher Weise umformen. Sei $(t_1 t_2 \ldots t_{2m}, v_1 v_2 \ldots v_{2m})$ das gegebene Paar. Dann wird daraus zuerst die Ausgabefolge $y_1 y_2 \ldots y_{2m}$ des Schieberegisters ermittelt. Offensichtlich muss $y_i = t_i \oplus v_i$ für $1 \leq i \leq 2m$ gelten. Als Erstes stellt der Kryptoanalyst fest, dass die Ausgabe in den ersten m Takten, ihm gerade die Anfangsbelegung liefert, d.h. $y_i = d_{m-i}$, da in den ersten m Takten der Reihe nach die Werte $k_m(0), k_{m-1}(0), \ldots, k_1(0)$ ausgegeben werden. Ferner gilt noch $y_j = k_{m-s}(j+s+1))$. Damit erhalten wir aus der Gleichung

$$k_1(t+1) = c_1 k_1(t) \oplus c_2 k_2(t) \oplus \ldots \oplus c_m k_m(t)\,,$$

die das Verhalten des ersten Registers beschreibt, die Gleichung

$$y(m+t+1) = c_1 y(m+t) \oplus c_2 y(m+t-1) \oplus \ldots \oplus c_m y(t+1)$$

für $0 \leq t \leq m-1$. Der Kryptoanalyst erhält somit ein lineares Gleichungssystem zur Bestimmung der c_i, $1 \leq i \leq m$. Die Lösungen bilden einen Vektorraum der Dimension $m - r$, wobei r der Rang der Koeffizientenmatrix ist. Ist $r = m$, so hat das Gleichungssystem eine eindeutige Lösung, aus der er die volle Kenntnis des Schieberegisters gewinnt und damit in der Lage ist, jeden Kryptotext zu entschlüsseln. Ist das Gleichungssystem nicht eindeutig lösbar, so gibt es 2^{n-r} Lösungen, d.h. der Kryptoanalyst hat in der Regel durch sein Vorgehen die Zahl der möglichen Schieberegister stark eingeschränkt.