OSNABRÜCKER SCHRIFTEN ZUR MATHEMATIK

Reihe V Vorlesungsskripten

EHeft 9 Wintersemester 2001/02

Codierungstheorie und Kryptographie

W. Bruns

Fachbereich Mathematik/Informatik
Universität Osnabrück

OSM Osnabrücker Schriften zur Mathematik

März 2002

Herausgeber	Selbstverlag der Universität Osnabrück Fachbereich Mathematik/Informatik 49069 Osnabrück
Geschäftsführer	Prof. Dr. W. Bruns
Berater:	Prof. Dr. P. Brucker (Angew. Mathematik) Prof. Dr. E. Cohors-Fresenborg (Didaktik der Mathematik) Prof. Dr. V. Sperschneider (Informatik) Prof. Dr. R. Vogt (Reine Mathematik)
Druck	Hausdruckerei der Universität Osnabrück

Weitere Reihen der OSM:

Reihe D	Mathematisch-didaktische Manuskripte
Reihe I	Manuskripte der Informatik
Reihe M	Mathematische Manuskripte
Reihe P	Preprints
Reihe U	Materialien zum Mathematikunterricht

Codierungstheorie und Kryptographie

Winfried Bruns

Skript zur Vorlesung WS 2001/2002

Inhaltsverzeichnis

Vorwort

Der vorliegende Text ist die Niederschrift einer Vorlesung im WS 2001/02 an der Universität Osnabrück. Sie geht davon aus, daß die Hörer neben der Vorlesung „Lineare Algebra“ auch eine „Einführung in die Algebra“ absolviert haben, in der die wichtigsten Sätze und Methoden der elementaren Zahlentheorie, wie die Existenz und Bestimmung des größten gemeinsamen Teilers, der chinesische Restsatz und die Existenz von Primitivwurzeln modulo Primzahlen diskutiert worden sind. Ebenso werden Kenntnisse über Polynome und endliche Körper vorausgesetzt. Die wichtigsten Sätze werden aber zumindest zitiert, wenn sie wirklich gebraucht werden.

Einige Übungsaufgaben verlangen die Implementation von Algorithmen in `Aribas`, das von Otto Forster geschaffen wurde und per

`http://www.mathematik.uni-muenchen.de/~forster/sw/aribas.html`

erhältlich ist.

Ich danke Anja Kipp für ihre engagierte Mithilfe bei der Erstellung der LaTeX-Version des Manuskripts.

Osnabrück, Februar 2002 *Winfried Bruns*

ABSCHNITT 1

Nachrichtenübertragungssysteme

Beispiele für Nachrichtenübertragungssysteme sind uns allen aus dem täglichen Leben vertraut:

Rundfunk, Fernsehen, Telefon, ...,
Datenspeichersysteme,
Nervensysteme in Lebewesen.

Aufgabe der Codierungstheorie ist es herauszufinden, wie Nachrichten möglichst effizient und möglichst fehlerfrei übertragen (oder gespeichert) werden können. Effizienz und Fehlerfreiheit sind dabei einander widersprechende Ziele, so daß raffinierte mathematische Methoden erforderlich sind, um sie dennoch so gut wie möglich zu erreichen. Spektakuläre Anwendungen der Codierungstheorie sind die Datenspeicherung auf CDs und die Bilder der Planeten, die von Satelliten zur Erde übermittelt worden sind.

Das einfachste Modell eines Nachrichtenübertragungssystems unterscheidet die *Nachrichtenquelle*, den *Übertragungskanal* und den *Empfänger*:

Beispiele für Quellen sind die menschliche Stimme, ein Speichermedium mit digitalen Daten, Meßinstrumente in einem Satelliten oder Sensoren in Sinnesorganen. Beispiele für Kanäle sind Telefonverbindungen, Richtfunkstrecken, Nervenbahnen und Datenleitungen in Computern oder Netzwerken. Es ist häufig aber nicht einfach, Quelle und Kanal gegeneinander abzugrenzen.

Fast alle Kanäle unterliegen Störungen, die man üblicherweise als *Rauschen* bezeichnet. Diese Bezeichnung erklärt sich von selbst, wenn man einmal alte Schallplatten hört oder versucht, Radiosender auf der Kurzwelle zu empfangen. Das Rauschen stört die Übertragung der Information und verfälscht diese. Häufig stellt man die Störungen als zusätzliche Quelle dar, die das Rauschen in den (zunächst idealen) Kanal einspeist.

Viele Quellen besitzen *Redundanz*, d. h. eine gewisse Weitschweifigkeit oder Überbestimmtheit. Ein einfaches Beispiel sind Datumsangaben wie „Montag, der 15.10.2001“. Wenn keine Zweifel über den verwendeten Kalender bestehen, ist der Zusatz „Montag“ überflüssig. Er ist aber dennoch sinnvoll, weil er die Datumsangabe absichert und verläßlicher macht. Redundanz in der menschlichen Sprache

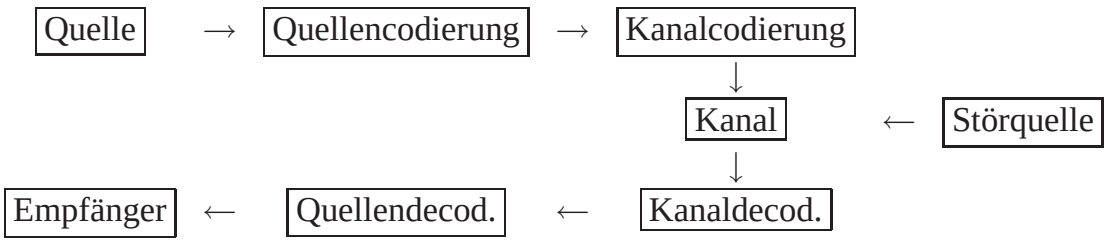
entsteht auch daraus, daß nur ein extrem geringer Anteil der aller möglichen Buchstabenfolgen Wörter der deutschen (oder einer anderen) Sprache sind, und wiederum nur ein geringer Anteil der Folgen von Wörtern syntaktisch korrekte und semantisch sinnvolle Sätze ergeben. Deshalb können wir Texte trotz gelegentlicher Dreckfehler einwandfrei lesen. Kurzum, *Redundanz sichert vor Übertragungsfehlern*.

Die Redundanz vieler Quellen läßt sich technisch schwer nutzen. Darum ist es häufig sinnvoll, die Redundanz der Quelle durch eine *Datenkompression* oder *Quellencodierung* zu eliminieren. Datenkompression ist uns aus der Computerwelt durch Programme wie `zip` bekannt, mit deren Hilfe redundante Daten auf kleinem Raum gespeichert werden können.

In einem zweiten Schritt, der *Kanalcodierung*, wird gezielt Redundanz hinzugefügt, um die Datenübertragung gegen Störungen zu sichern. Die Kanalcodierung ist der eigentliche Gegenstand der Codierungstheorie. Wir werden uns aber zunächst mit den Grundbegriffen der Informationstheorie und der Quellencodierung befassen.

Natürlich müssen Kanal- und Quellencodierung nach der Übertragung wieder rückgängig gemacht werden, wobei die Kanaldecodierung ein schwieriges Problem ist.

Wir können nun das obige Modell folgendermaßen verfeinern:



Man kann die Quellen nach der Art der von ihnen erzeugten Signale einteilen in kontinuierliche und diskrete. Wir werden uns nur für *diskrete Quellen mit endlichem Alphabet* interessieren. Beschrieben werden diese Quellen allein durch die statistischen Eigenschaften der von ihnen erzeugten Signalfolgen.

Ein Kanal wird beschrieben durch sein *Eingangs*- und sein *Ausgangsalphabet* – diese stimmen sehr häufig überein – und die statistischen Zusammenhänge zwischen den eingegebenen und ausgegebenen Zeichenreihen. Wie die Alphabete der Quellen werden auch die Kanalalphabete im folgenden stets endlich sein.

Der Begriff der Information hat Ähnlichkeit zu dem der Energie. Genau so wenig wie die Physik versucht, das „Wesen“ der Energie zu beschreiben, versucht die Informationstheorie zu erklären, was Information eigentlich ist. Man kann in beiden Fällen aber die Transformation zwischen verschiedenen Formen quantitativ

exakt erfassen. Bei der Übertragung von Information über gestörte Kanäle treten Verluste auf, die den Verlusten bei der Energieübertragung durch Reibung oder Abwärme entsprechen.

Wir haben berets von *Alphabeten* gesprochen. Dies sind für uns endliche Mengen A. Ihre Elemente nennen wir auch *Zeichen*, *Symbole* oder *Buchstaben*. Ein *Wort* oder eine *Zeichenreihe* über A ist ein n-Tupel

$$a_1 \cdots a_n = (a_1, \ldots, a_n)$$

mit $n \in \mathbb{N}$ und $a_i \in A$ für $i = 1, \ldots, n$. In der Regel lassen wir dabei die Klammern und Kommata weg, wie auf der linken Seite bereits angezeigt. Im Fall $n = 0$ erhalten wir das *leere Wort*. Wörter kann man zusammensetzen oder verketten :

$$a_1 \cdots a_m * b_1 \cdots b_n = (a_1, \ldots, a_m, b_1, \ldots, b_n)$$

(Meistens werden wir das Zeichen $*$ für die Verkettung weglassen.) Ein Wort a ist *Präfix* des Wortes b, wenn es ein Wort c mit $b = ac$ gibt. Existiert eine Zerlegung $b = ca$, so heißt a *Suffix* von b. Wörter die aus zwei Buchstaben bestehen, heißen *Bigramme*, solche aus drei Buchstaben *Trigramme* usw.

ABSCHNITT 2

Entropie endlicher Wahrscheinlichkeitsräume

Endliche Wahrscheinlichkeitsräume dienen uns als mathematische Modelle für Informationsquellen und Übertragungskanäle.

Definition. Eine Wahrscheinlichkeitsverteilung auf der endlichen Menge $A \neq \emptyset$ ist eine Abbildung $p : A \to \mathbb{R}$ mit folgenden Eigenschaften:

$$\text{(a)} \quad p(a) \geq 0 \quad \text{für alle } a \in S, \qquad \text{(a)} \quad \sum_{a \in A} p(a) = 1.$$

Ein *endlicher Wahrscheinlichkeitsraum* ist ein Paar (A, p), wobei p eine Wahrscheinlichkeitsverteilung auf A ist.

Jede Teilmenge $A' \subset A$ nennt man ein *Ereignis*. Die *Wahrscheinlichkeit* von A' ist

$$p(A') = \sum_{a \in A'} p(a).$$

Beispiel 2.1. Wir wählen $A = \{1,2,3,4,5,6\}$ als die Menge der Augenzahlen eines Würfels und zwei Wahrscheinlichkeitsverteilungen auf A:

(a)

a	1	2	3	4	5	6
$p(a)$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

(b)

a	1	2	3	4	5	6
$p(a)$	$1/9$	$1/9$	$1/9$	$1/6$	$1/6$	$1/3$

Ein Beispiel eines Ereignisses ist $A' = \{2,4,6\}$, das das Werfen einer geraden Zahl darstellt. Mit der Verteilung (a) ist $p(A') = 1/2$, bei der Verteilung (b) hingegen ist $p(A') = 11/18$.

Die Ungewißheit vor Empfang einer Nachricht ist vergleichbar der Ungewißheit über den Ausgang eines Versuchs. Der Informationsgehalt einer Nachricht ist umso größer, je mehr Unsicherheit sie beseitigt, mit anderen Worten: je schwerer es ist, ihren Inhalt vorherzusagen. Beispiel (a) können wir als Modell eines ungezinkten Würfels betrachten, (b) als Modell eines gezinkten Würfels. Offensichtlich ist die Ungewißheit größer beim ungezinkten Würfel.

Wir wollen nun ein Maß für diese Ungewißheit definieren:

Definition. Als *Entropie* (oder *Selbstinformation, Informationsgehalt*) eines endlichen Wahrscheinlichkeitsraums $S = (A, p)$ bezeichnen wir die Zahl

$$H(S) := \sum_{a \in A} -p(a) \log p(a) \qquad (\text{„bit“}).$$

Wenn $p_i = 0$ ist, werde dabei $p_i \log p_i = 0$ gesetzt (dies ist sinnvoll wegen $\lim_{x\to 0} x\log x = 0$).) Dabei ist hier wie im folgenden der Logarithmus zur Basis 2 gewählt. Dies ist durch die „Maßeinheit“ bit gekennzeichnet.

In obigem Beispiel hat der durch (a) gegebene Wahrscheinlichkeitsraum die Entropie $-6 \cdot 1/6 \cdot \log(1/6) = \log 6 \approx 2.585$, während sich für (b) die Entropie 2.447 ergibt. Allgemeiner sehen wir sofort, daß der *gleichverteilte* Wahrscheinlichkeitsraum $S = (A, p)$ mit $|A| = n$, $p(a) = 1/n$ für alle $a \in A$, die Entropie $\log n$ hat.

Der Begriff Entropie stammt aus der Thermodynamik. Er dient dort (üblicherweise mit entgegengesetztem Vorzeichen) als Maß für die Abweichung von der Gleichverteilung einer physikalischen Größe. Wir werden in Satz 2.3 sehen, daß diese Interpretation auch hier sinnvoll ist.

Die Definition der Entropie scheint auf den ersten Blick etwas willkürlich. Ihre endgültige Rechtfertigung erfährt sie durch den *Quellen-Codierungssatz*. Er besagt grob gesprochen, daß sich genügend lange Zeichenfolgen, in denen das Zeichen $a \in A$ mit Wahrscheinlichkeit $p(a)$ vorkommt, binär (also mit Wörtern aus dem Alphabet $\{0,1\}$) so codieren lassen, daß im Mittel fast nur $H(A)$ Binärzeichen pro Element der Folge benötigt werden. Dabei muß man dann aber zunächst „Blöcke“ der Zeichen in A bilden und diese dann durch binäre Codeworte mit möglicherweise unterschiedlichen Längen darstellen. Wir werden das noch sehr präzise diskutieren, können uns aber schon am folgenden Beispiel orientieren.

Beispiel 2.2. (a) Wir betrachten zunächst die Gleichverteilung auf der n-elementigen Menge A. Wir wählen eine rationale Zahl $r/s > H(A)$, $r,s \in \mathbb{N}$. Dann ist $2^r > n^s$, und man findet eine injektive Abbildung $\varphi : A^s \to \{0,1\}^r$.

Um Nachrichten der Quelle A zu codieren, fassen wir dann immer s Zeichen zu einem Block $a = a_1 \cdots a_s$ zusammen und „codieren“ ihn durch $\varphi(a)$. Für s Zeichen der Nachricht sind dann r Binärzeichen verwendet worden, also r/s Binärzeichen pro Zeichen aus A.

Im Fall $n = 6$, $r = 8$, $s = 3$, erhalten wir so $r/s \approx 2.667$, was der Entropie 2.585 deutlich näher kommt als 3, was man bei einer binären Codierung von einzelnen Zeichen aus $\{1,\ldots,6\}$ durch Binärworte fester Länge erhält.

(b) Dies kann man verbessern durch einen Code mit variabler Wortlänge. Wir betrachten die folgende Codierung von einzelnen Zeichen aus $A = \{1,\ldots,6\}$:

1	000
2	001
3	010
4	011
5	10
6	11

Im Fall der Gleichverteilung beträgt die mittlere Anzahl von Binärzeichen pro Zeichen aus A jetzt 2.667. Im Fall der Wahrscheinlichkeitsverteilung ($1/9, 1/9, 1/9, 1/6, 1/6 1/3$) nur 2.5, weil die häufiger vorkommenden Zeichen durch kürzere Codeworte dargestellt werden. (Dieses Prinzip benutzt schon das *Morse-Alphabet*.)

Man beachte, daß bei der obigen Codierung der Empfänger aus der Folge der empfangenen Binärzeichen das gesendete Wort aus A wieder rekonstruieren kann, und zwar deshalb, weil kein Codewort Präfix eines anderen Codewortes ist.

Die systematische Ausnutzung der an den Beispielen erläuterten Ideen wird uns zum Quellen-Codierungssatz führen.

Nach diesen Beispielen wollen wir uns den mathematischen Eigenschaften der Entropie widmen. Ist der Wahrscheinlichkeitsraum S gegeben durch

a	A_1	$\dots$	A_n
$p(a)$	p_1	$\dots$	p_n,

so schreiben wir auch $H(p_1,\dots,p_n)$ statt $H(S)$.

Satz 2.3.

(a) $H(p,1-p)$ *ist eine stetige Funktion auf* $[0,1]$.

(b) $H(p_1,\dots,p_n)$ *ist eine symmetrische Funktion der Argumente.*

(c) (Gruppierungseigenschaft) *Falls* $p_n = q_1 + q_2 > 0$, *so ist*

$$H(p_1,\dots,p_{n-1},q_1,q_2) = H(p_1,\dots,p_n) + p_n H\left(\frac{q_1}{p_n},\frac{q_2}{p_n}\right).$$

(d) $H(p_1,\dots,p_n) = 0 \iff$ *es existiert ein* i *mit* $p_i = 1$.

(e) $H(p_1,\dots,p_n) \le \log n$ *und*

$$H(p_1,\dots,p_n) = \log n \iff p_i = \frac{1}{n}, \quad i = 1,\dots,n.$$

Die Gruppierungseigenschaft (c) ist eine Art „Additivitätsgesetz“ für Unbestimmtheiten, das man folgendermaßen interpretieren kann: Ein Versuch besitze n Ausgänge $A_1,\dots,A_n$. Falls A_n eintritt, werde ein weiterer Versuch ausgeführt mit den möglichen Ausgängen B_1 und B_2. Die Ungewißheit über den Ausgang des Gesamtversuchs ist die Summe aus der Ungewißheit über den 1. Versuch und dem Produkt der Ungewißheit über den 2. Versuch und der Wahrscheinlichkeit, daß der 2. Versuch überhaupt durchgeführt wird.

Man kann zeigen, daß die Funktion H durch ihre Eigenschaften (a), (b) und (c) bis auf einen konstanten Faktor eindeutig bestimmt ist (Satz von Faddeev; siehe [**AD**] zu dieser und anderen axiomatischen Charakterisierungen der Entropie). Sieht man (a), (b) und (c) als vernünftige Forderungen für ein Maß der Ungewißheit an, so ist unser Maß offensichtlich vernünftig und im wesentlichen auch das einzig vernünftige.

Teil (d) besagt, daß die Ungewißheit genau dann verschwindet, wenn der Versuchsausgang (mit Wahrscheinlichkeit 1 vorher bekannt ist. Auch Teil (e) bestätigt eine Erwartung, die wir an ein Maß der Unsicherheit stellen: sie ist dann am größten, wenn alle Versuchsausgänge gleichwahrscheinlich sind und mißt damit die Abweichung von S von der Gleichverteilung.

Die Aussagen (a) und (b) von Satz 2.3 folgen unmittelbar aus der Definition der Entropie, wobei für die Stetigkeit in 0 und 1 zu berücksichtigen ist, daß $\lim_{x\to 0} x\log x = 0$ ist. Teil (c) ist eine einfache Rechenaufgabe. Aussage (d) folgt wieder unmittelbar aus der Definition von H. Für (e) beweisen wir zunächst einen Hilfssatz, dessen Teil (b) ein Spezialfall der *Jensenschen Ungleichung* ist.

Satz 2.4. (a) *Für $x > 0$ gilt $\log x \le (x-1)\log e$ und $\log x = (x-1)\log e$ nur für $x = 1$.*

(b) *Seien $(x_1,\dots,x_n),(y_1,\dots,y_n)$ n-Tupel positiver Zahlen mit $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$. Dann ist*

$$-\sum_{i=1}^n x_i \log x_i \le -\sum_{i=1}^n x_i \log y_i,$$

und Gleichheit tritt genau dann ein, wenn $x_i = y_i$ für $i = 1,\dots,n$.

Beweis. Zu (a): Sei $f(x) := (x-1)\log e - \log x$. Die Funktion f ist (stetig) differenzierbar mit Ableitung $f'(x) = \log e - \frac{\log e}{x}$, und es gilt

$$f'(x) \begin{cases} > 0 & \text{für} \quad x > 1 \\ = 0 & \text{für} \quad x = 1 \\ < 0 & \text{für} \quad x < 1. \end{cases}$$

Also ist f' streng monoton fallend auf $(0,1]$, streng monoton wachsend auf $[1,\infty)$. Mit $f(1) = 0$ folgt die Behauptung.

(b) Wegen (a) ist

$$\sum_{i=1}^n x_i \log y_i - \sum_{i=1}^n x_i \log x_i = \sum_{i=1}^n x_i \log \frac{y_i}{x_i} \le \sum_{i=1}^n x_i \left(\frac{y_i}{x_i} - 1\right) \log e = 0.$$

Gleichheit tritt nur dann ein, wenn $x_i \log \frac{y_i}{x_i} = x_i(\frac{y_i}{x_i} - 1)\log e$ für $i = 1,\dots,n$ ist, was gemäß (a) nur bei $x_i = y_i$ möglich ist. □

Nun folgt Teil (e) von Satz 2.3 ganz einfach:

$$H(p_1,\dots,p_n) = -\sum_{i=1}^n p_i \log p_i \le -\sum_{i=1}^n p_i \log \frac{1}{n} = \log n$$

und Gleichheit gilt genau dann, wenn $p_i = 1/n$ für $i = 1,\dots,n$.

Entropie von Zufallsvariablen. Wir wollen nun den Begriff der Entropie etwas verallgemeinern, indem wir auch Zufallsvariablen eine Entropie zuordnen.

Definition. Sei (A, p) ein endlicher Wahrscheinlichkeitsraum. Eine Abbildung $X : A \to B$ von A in eine Menge B ist eine *Zufallsvariable* (oder zufällige Variable).

Häufig verlangt man, daß $B = \mathbb{R}$ ist, aber diese Einschränkung ist für unsere Anwendungen zu eng. Zufallsvariable modellieren zum Beispiel „Messungen“ auf einem Wahrscheinlichkeitsraum oder, in unserem Fall, die Symbole, die eine Nachrichtenquelle erzeugt.

Die Zufallsvariable X macht $\mathrm{Bild}(X)$ in natürlicher Weise zu einem Wahrscheinlichkeitsraum. Wir dürfen der Einfachheit halber $B = \mathrm{Bild}(X)$ voraussetzen. Sei $b \in B$. Dann können wir das Ereignis

$$A' = X^{-1}(B') = \{a \in A : X(a) = b\}$$

betrachten und

$$q(b) = p(A')$$

setzen. Da offensichtlich $\sum_{b\in B} q(b) = \sum_{a\in A} p(a)$ ist, erhalten wir wirklich eine Wahrscheinlichkeitsverteilung q auf B. Man schreibt nun aber viel suggestiver

$$q(b) = p(X = b).$$

Man nennt die auf $\mathrm{Bild}(X)$ definierte Verteilung q die *Verteilung von* X.

Die Entropie von (B, q) heißt auch *Entropie* $H(X)$ *von* X. Mit anderen Worten,

$$H(X) = -\sum_{b\in B} p(X = b) \log p(X = x).$$

Sind X, Y Zufallsvariable auf A so bezeichnet „$X = x,\ Y = y$“ das Ereignis aller $a \in A$, für die $X(a) = x$, $Y(a) = y$ gilt. Die eingeführten Bezeichnungen können natürlich in vielfältiger Weise variiert werden; sie sollten stets selbsterklärend sein.

Zur Modellierung von Kanälen benutzen wir zusammengesetzte Wahrscheinlichkeitsräume. Dies sind Wahrscheinlichkeitsräume $U = (C, r)$ deren Trägermenge C ein kartesisches Produkt $A \times B$ ist. Seien A, B endliche Mengen, $C := A \times B$ und $U = (C, p)$ ein Wahrscheinlichkeitsraum. Dann sind die Projektionen

$$X : C \to A, \quad X(a, b) = a, \qquad Y : C \to B, \quad Y(a, b) = b$$

typische Beispiele von Zufallsvariablen. Die auf A und B induzierten Verteilungen p und q heißen *Randfelder* von (C, r).

Beispiel 2.5. Die Tabelle links stellt die Verteilung r auf $C = A \times B$ dar und die Tabellen rechts geben die Randverteilungen p und q an.

r	b_1	b_2	
a_1	$1/3$	$1/9$	$4/9$
a_2	$1/9$	$1/6$	$5/18$
a_3	$1/6$	$1/9$	$5/18$
	$11/18$	$7/18$	

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ 4/9 & 5/18 & 5/18 \end{pmatrix}$$
$$\begin{pmatrix} b_1 & b_2 \\ 2/3 & 1/3 \end{pmatrix}$$

Umgekehrt können wir mit Zufallsvariablen X und Y auf einem Wahrscheinlichkeitsraum (A,p) den zusammengesetzten Wahrscheinlichkeitsraum $\text{Bild}(X) \times \text{Bild}(Y)$ betrachten, auf dem

$$r(x,y) = p(X = x, Y = y)$$

ist. Wir bezeichnen ihn kurz mit $X \times Y$.

Mit Hilfe von bedingten Wahrscheinlichkeiten kann man die Abhängigkeit von Zufallsvariablen von einander beschreiben.

Definition. Seien (A,p) ein endlicher Wahrscheinlichkeitsraum und A' ein Ereignis in A mit $p(A') > 0$. Für jedes Ereignis $A'' \subset A$ nennt man

$$p(A'' \mid A') = \frac{p(A' \cap A'')}{p(A')}$$

die *bedingte Wahrscheinlichkeit* von A'' unter der Bedingung A'.

Im Fall $p(A') = 0$ setzt man $p(A'' \mid A') = p(A'')$.

Damit sollte auch klar sein, was $p(X = x \mid Y = y)$ ist, nämlich

$$p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)},$$

vorausgesetzt $p(Y = y) > 0$.

Für zusammengesetzte Wahrscheinlichkeitsräume $(A \times B, r)$ vereinfachen wir die Schreibweise noch etwas, indem wir

$$r(a \mid b) = r(X = a \mid Y = b)$$

setzen, wobei X und Y die Projektionen auf die erste bzw. zweite Komponente sind. Entsprechend ist $r(b \mid a)$ definiert.

Definition. Seien X und Y Zufallsvariable auf (A,p). Die *bedingte Entropie von X unter der Bedingung $Y = y$* ist

$$H(X \mid Y = y) := -\sum_{x \in \text{Bild}(X)} p(X = x \mid Y = y) \log p(X = x \mid Y = y).$$

Wir können $H(X \mid Y = y)$ als Maß für die Ungewißheit des über die Größe X verstehen, wenn bekannt ist, daß Y den Wert y hat. Wir mitteln diese Zahlen noch über $y \in \text{Bild}(Y)$:

Definition. Die (mittlere) *bedingte Entropie von X unter Y* ist

$$H(X \mid Y) := \sum_{y \in \mathrm{Bild}(Y)} p(Y=y) H(X \mid Y=y).$$

Die Interpretation liegt auf der Hand: $H(X \mid Y)$ ist die Ungewißheit, die im Mittel über den Wert von X noch besteht, wenn der Wert von Y bekannt ist. Die Entropien von $X \times Y$ und X unterscheiden sich gerade um die bedingte Entropie von Y unter X, wie der folgende Satz zeigt. Wir können ihn so interpretieren: Die Ungewißheit über den Wert von $X \times Y$ ist die Summe aus der Ungewißheit über den Wert von X und der Ungewißheit über den von Y, sobald X bekannt ist.

Satz 2.6. $H(X \times Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$.

Beweis. Wir beweisen die erste Gleichung; die zweite folgt analog. Zur Vereinfachung der Schreibweise setzen wir $U = \mathrm{Bild}(X)$, $V = \mathrm{Bild}(Y)$, $p(x) = p(X=x)$, $p(y) = p(Y=y)$, $p(x,y) = p(X=x, Y=y)$ usw.

$$\begin{aligned}
H(X \times Y) &= -\sum_{x \in U} \sum_{b \in V} p(x,y) \log p(x,y) \\
&= -\sum_{x \in U} \sum_{y \in V} p(x) p(y \mid x) \log \big(p(x) p(y \mid x) \big) \\
&= -\sum_{x \in U} \sum_{y \in V} p(x) p(y \mid x) \log p(x) \\
&\qquad\qquad - \sum_{x \in U} \sum_{y \in V} p(x) p(y \mid x) \log p(y \mid x) \\
&= -\sum_{x \in U} \left(\sum_{y \in V} p(y \mid x) \right) p(x) \log p(x) \\
&\qquad\qquad - \sum_{x \in U} p(x) \left(\sum_{y \in V} p(y \mid x) \log p(y \mid x) \right) \\
&= H(X) + \sum_{x \in U} p(x) H(Y \mid x) \\
&= H(X) + H(Y \mid X).
\end{aligned}$$

□

Die Ungewißheit über den Wert von X sollte durch Kenntnis des Wertes von Y nicht größer werden, und dies ist auch nicht der Fall:

Satz 2.7.

(a) $H(X \mid Y) \le H(X)$.
(b) $H(X \times Y) \le H(X) + H(Y)$.
(c) *Bei* (a) *und* (b) *gilt Gleichheit genau dann, wenn*

$$p(X=x, Y=y) = p(X=x) p(Y=y)$$

für alle $x \in \mathrm{Bild}(X)$, $y \in \mathrm{Bild}(Y)$.

Beweis. Mit Hilfe der Jensenschen Ungleichung zeigt man (a) und den (a) betreffenden Teil von (c). Dann folgt der Rest mit Satz 2.6. □

Wenn die in (c) angegebene Eigenschaft erfüllt ist, heißen X und Y *unabhängig*, und die Unabhängigkeit von X und Y ist genau dann gegeben, wenn X im Sinne der folgenden Definition keine Information über Y liefert.

Definition. Der *Informationsgehalt von X über Y* ist

$$I(X,Y) := H(Y) - H(Y \mid X).$$

Die Information von X über Y ist also die Größe, um die die Ungewißheit über Y durch die Kenntnis von X vermindert wird. Sie kann nicht größer sein als die Ungewißheit über Y selbst (und nicht größer als die Ungewißheit über X). Ferner gibt X über Y die gleiche Information wie Y über X:

Satz 2.8.

(a) $I(X,Y) \leq \min(H(X),H(Y)$
(b) $I(X,Y) = I(Y,X)$.

Beweis. Gemäß Definition ist $I(X,Y) \leq H(Y)$. Nach Satz 2.6 gilt

$$H(X) + H(Y \mid X) = H(X \times Y) = H(Y) + H(X \mid Y)$$

Daraus folgt

$$I(X,Y) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y) = I(Y,X)$$

Das aber ist (b), und damit gilt auch die zweite Ungleichung in (a). □

Im Spezialfall, daß X und Y die Projektionen auf die Randfelder eines zusammengesetzten Wahrscheinlichkeitsraums $C = A \times B$ sind, schreiben wir auch $H(A \mid B)$, $I(A,B)$ usw.

Anhang. Weitere Begriffe der elementaren Wahrscheinlichkeitsrechnung. Wir wollen nun noch einige Grundbegriffe der Wahrscheinlichkeitsrechnung einführen, die wir hin und wieder benutzen werden. Im folgenden ist $S = (A, p)$ stets ein endlicher Wahrscheinlichkeitsraum.

Die wichtigsten numerischen Invarianten einer zufälligen Variablen mit Werten in $\mathbb{R}$ sind ihr Erwartungswert und ihre Varianz:

Definition. Sei $X : A \to \mathbb{R}$ eine zufällige Variable. Man nennt

$$E(X) = \sum_{a \in A} X(a)p(a)$$

den *Erwartungswert* von X und

$$V(X) = E\left((X - E(X))^2\right)$$

die *Varianz* von X.

Die Varianz mißt, wie stark die Werte von X um den Erwartungswert „streuen“. Man nennt $\sigma(X) = \sqrt{V(X)}$ die *Standardabweichung* oder *Streuung* von X. (Statt $E(X)$ schreibt man oft $\mu(X)$ und statt $V(X)$ auch $\sigma^2(X)$.)

Die Entropie von A ist damit gerade der Erwartungswert der zufälligen Variablen $X(a) = -\log p(a)$, $a \in A$.

Auf zufällige reellwertige Variable X, Y kann man alle Operationen anwenden, die für reellwertige Funktionen auf einer Menge erklärt sind. Insbesondere kann man Summen $X+Y$ und Produkt XY bilden, und reellwertige Funktionen mit X komponieren, also zum Beispiel $\exp(X)$ bilden.

Für den Erwartungswert gilt offensichtlich

$$E(X) = \sum_{x \in X(A)} x\, p(X = x).$$

Man nennt die auf $X(A)$ definierte Funktion p_X die *Verteilung von* X.

Das wichtigste Beispiel einer Verteilung (einer zufälligen Variablen) ist die *Bernoulli-Verteilung* mit den Parametern $n \in \mathbb{N}$ und p, $0 < p < 1$. Wir betrachten die Menge $A = \{0,1\}^n$ aller n-Tupel, deren Komponenten 1 oder 0 sind. (Bei Anwendungen auf Glücksspiele steht 0 zum Beispiel für Rot oder Kopf und 1 für Schwarz oder Zahl.) Die Wahrscheinlichkeitsverteilung auf A ist gegeben durch

$$p(a_1, \ldots, a_n) = p^k q^{n-k},$$

wobei k die Anzahl derjenigen Indizes i ist, für die $a_i = 1$. Nun setzen wir

$$X(a_1, \ldots, a_n) = \{i : a_i = 1\}|.$$

Offensichtlich nimmt X die Werte $0, \ldots, n$ an.

Für $p = 1/2$ steht die zufällige Variable X also zum Beispiel dafür, wie oft man beim n-maligen Werfen einer „fairen“ Münze das Resultat ‘Kopf erhalten hat.

Satz 2.9. *Mit den oben eingeführten Bezeichnungen gilt:*

$$\text{(a)} \quad p(X = k) = \binom{n}{k} p^k q^{n-k}, \qquad \text{(b)} \quad E(X) = np, \qquad \text{(c)} \quad V(X) = npq.$$

Beweis. Für (a) müssen wir zählen, in wievielen n-Tupeln mit den Komponenten 0 und 1 genau k-mal die 1 auftritt, denn jedes dieser Elementarereignisse hat die Wahrscheinlichkeit $p^k q^{n-k}$. Diese Anzahl ist aber gerade der Binomialkoeffizient $\binom{n}{k}$.

Für (b) und (c) benutzen wir einen Trick, der sich auf alle Verteilungen mit der Wertemenge $\{0, \ldots, n\}$ (oder sogar $\mathbb{N}$ bei unendlichen Wahrscheinlichkeitsräumen) anwenden läßt. Wir definieren das Polynom $f \in \mathbb{R}[T]$ durch

$$f = \sum_{k=0}^{n} p(X = k) T^k.$$

(Man nennt f die *erzeugende Funktion* der Folge $p(X=k)$, $k=0,\dots,n$.) Nach Definition ist

$$E(X)=\sum_{k=0}^{n} kp(X=k)=f'(1).$$

In unserem Fall ist

$$f=\sum_{k=0}^{n}\binom{n}{k}p^k q^{n-k}T^k=(pT+q)^n.$$

Mithin $E(X)=np$ für die Bernoulli-Verteilung.

Für die Varianz ist

$$\sum_{k=0}^{n}(k-E(X))^2 p(X=x)=\sum_{k=0}^{n}\big(k(k-1)+k-2kE(X)+E(X)^2\big)p(X=x)$$

auszuwerten. Wie oben sieht man

$$\begin{aligned}V(X)&=f''(1)+f'(1)-2f'(1)E(X)+E(X)^2 f(1)\\&=f''(1)+E(X)-E(X)^2,\end{aligned}$$

denn $f(1)=1$ (bei Verteilungen zufälliger Variabler) und $f'1)=E(X)$. Auf die Bernoulli-Verteilung angewandt, ergibt sich $V(X)=npq$ wie gewünscht. □

Wir notieren noch die Tschebyschevsche Ungleichung:

Satz 2.10. *Für eine reellwertige zufällige Variable X auf einem endlichen Wahrscheinlichkeitsraum gilt*

$$p\big(|X-E(X)|>\varepsilon\big)\le\frac{V(X)}{\varepsilon^2}$$

für alle $\varepsilon>0$.

Der Beweis ist sehr einfach:

$$\begin{aligned}V(X)&=E\big((X-E(X))^2\big)\\&=\sum_{|x-E(X)|<\varepsilon}(x-E(X))^2p(X=x)\quad+\sum_{|x-E(X)|\ge\varepsilon}(x-E(X))^2p(X=x)\\&\ge\varepsilon^2 p(|x-E(X)|\ge\varepsilon).\end{aligned}$$

Die allgemeine *Jensensche Ungleichung* betrifft konkave (oder konvexe) Funktionen. Sei $I\subset\mathbb{R}$ ein Intervall und $f:I\to\mathbb{R}$ eine Funktion auf I. Man nennt f *konkav*, wenn $f(\alpha x+\beta y)\ge\alpha f(x)+(1-\alpha)f(y)$ für alle $x,y\in I$ und alle $\alpha\in[0,1]$ gilt. Mit anderen Worten: f ist konkav, wenn für alle $x,y\in I$ der Graph von f zwischen x und y oberhalb seiner Sekante durch $(x,f(x))$ und $(y,f(y))$ verläuft. Per Induktion folgt dann

$$f\left(\sum_{i=1}^{n}\alpha_i x_i\right)\ge\sum_{i=1}^{n}\alpha_i f(x_i)$$

für alle $x_1, \ldots, x_n \in I$ und alle $\alpha_1, \ldots, \alpha_n \in [0.1]$ mit $\sum_{i=1}^n \alpha_i = 1$. Diese Ungleichung können wir als eine Ungleichung für die Erwartungswerte der zufälligen Variablen X und $f \circ X$ interpretieren, wenn X den Wert x_i mit Wahrscheinlichkeit α_i annimmt:

$$f(E(X)) \geq E(f \circ X).$$

Satz 2.4 ergibt sich unter der unwesentlichen Einschränkung $\sum_{i=1}^n x_i = 1$ daraus, wenn wir als konkave Funktion den Logarithmus wählen und X die Werte y_i/x_i mit Wahrscheinlichkeit x_i annehmen lassen. Dann ist

$$0 = \log(1) = \log(E(X)) \geq E(\log \circ X)) = \sum_{i=1}^n x_i \log \frac{y_i}{x_i},$$

mithin

$$\sum_{i=1}^n x_i \log x_i \geq \sum_{i=1}^n x_i \log y_i.$$

Bei konvexen Funktionen sind alle Ungleichungen umzukehren.

Übungen

2.11. (a) Bestimme die Entropie H des folgenden Wahrscheinlichkeitsraums:

a	1	2	3	4	5
$p(a)$	1/5	1/3	1/15	2/15	4/15.

(b) Gegeben sei der folgende zusammengesetzte Wahrscheinlichkeitsraum $C = (A \times B, r)$:

	B_1	B_2
A_1	1/3	1/9
A_2	1/9	1/6
A_3	1/6	1/9

Bestimme $H(C)$, $H(A)$, $H(B)$, $H(A \mid B)$ und $I(A,B)$.

2.12. (a) Erweitere die Gruppierungseigenschaft der Entropie zu einer Aussage

$$H(p_1, \ldots, p_n, q_1, \ldots, q_m) = H(p_1, \ldots, p_n, q) + \cdots$$

wobei $q = q_1 + \cdots + q_m$.

(b) Zeige

$$H(1/mn, \ldots, 1/mn) = H(1/m, \ldots, 1/m) + H(1/n, \ldots, 1/n).$$

2.13. Finde zwei verschiedene Folgen $0 < p_1 \leq p_2 \leq \cdots \leq p_n$ und $0 < q_1 \leq q_2 \leq \cdots \leq q_n$ mit $H(p_1, \ldots, p_n) = H(q_1, \ldots, q_n)$. Geht das schon mit $n = 2$?

2.14. Wir betrachten allgemeiner als in der Vorlesung diskrete Wahrscheinlichkeitsräume $W = (A, p)$, wobei A eine abzählbare Menge ist und $p : A \to \mathbb{R}$ eine Funktion mit $p(a) \geq 0$ und $\sum_{a \in A} p(a) = 1$. Man kann ohne Einschränkung $A = \mathbb{N}$ annehmen. Alle hier betrachteten Reihen haben nichtnegative Glieder, so daß die Summationsreihenfolge keine Rolle spielt. Wir schreiben p_k für $p(k)$.

Wenn $\sum_{k=0}^{\infty} p_k \log(1/p_k)$ konvergiert, so heißt der Grenzwert $H(W)$ *Entropie* von W. Andernfalls setzen wir $H(W) = \infty$.

(a) Bestimme die Entropie der geometrischen Verteilung $p_k = q^k p$, $k \in \mathbb{N}$, mit dem Parameter p, $0 < p < 1$, und $q = 1 - p$.

(b) Hat die Poisson-Verteilung $p_k = e^{-\lambda} \lambda^k / k!$, $k \in \mathbb{N}$, mit dem Parameter $\lambda > 0$ endliche Entropie?

(c) Finde eine Verteilung mit unendlicher Entropie.

2.15. Beweise Satz 2.7.

In der folgenden Aufgabe benutzen wir die im Beweis von Satz 2.6 eingeführten Schreibweisen.

2.16. Seien X, Y zufällige Variable auf dem Wahrscheinlichkeitsraum (A, p). Wir betrachten $-\log p(x \mid y) = -\log p(X = x \mid Y = y)$ als zufällige Variable auf $\mathrm{Bild}(X) \times \mathrm{Bild}(Y)$. Zeige

$$H(X \mid Y) = E(-\log p(x \mid y),$$

$$I(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x \mid y)}{p(x)} = E\left(\log \frac{p(x \mid y)}{p(x)} \right).$$

ABSCHNITT 3

Diskrete endliche Informationsquellen und Kanäle ohne Gedächtnis

Wir verwenden nun die im Abschnitt 2 entwickelten Begriffe, um mathematische Modelle für Informationsquellen und Kanäle zu formulieren. Für unsere Zwecke genügt es, die einfachsten Typen zu betrachten.

Definition. Eine *diskrete endliche Informationsquelle Q ohne Gedächtnis* mit dem Alphabet A ist ein Wahrscheinlichkeitsraum $Q = (A, p)$.

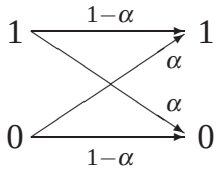
Dazu stellen wir uns vor: Eine Informationsquelle sendet zu diskreten Zeitpunkten $t = 0, 1, \ldots$ Signale aus, die dem Alphabet A entnommen sind. Die Wahrscheinlichkeit für das Auftreten von $a \in A$ zur Zeit t ist gleich $p(a)$, unabhängig von t und unabhängig von den zu den Zeiten $0, \ldots, t-1$ gesendeten Signalen: die Quelle ist ohne Gedächtnis. Viele der realen Informationsquellen erfüllen diese Bedingung nicht, aber für unsere Zwecke genügen diese einfachen Quellen. Für die Verallgemeinerungen verweisen wir auf die Literatur.

Definition. Ein *diskreter endlicher Kanal ohne Gedächtnis* mit Eingabealphabet A und Ausgabealphabet B ist eine Tripel $K = (A, B, R)$, bei dem A, B endliche Mengen und R eine Familie $(r_a)_{a \in A}$ von Wahrscheinlichkeitsverteilungen auf B ist.

Die Zahl $r_a(b)$ gibt die Wahrscheinlichkeit dafür an, daß b empfangen wird, wenn a gesendet wird. Diese Wahrscheinlichkeit hängt nicht von der „Vorgeschichte“ des Kanals ab: auch er ist ohne Gedächtnis.

Beispiel 3.1. Das wichtigste Beispiel eines Kanals ist der *binäre symmetrische Kanal BSC*(α) mit Irrtumswahrscheinlichkeit α, $0 < \alpha < 1$. Er hat das Eingabealphabet $A = \{0, 1\}$, Ausgabealphabet $B = A$ und „Kanalmatrix“

	0	1
0	$1-\alpha$	α
1	α	$1-\alpha$



Das Diagramm rechts beschreibt den Kanal sehr anschaulich.

Im obigen Beispiel (bei dem es allerdings keine Rolle spielt) und im weiteren sollen die Zeilen der Matrix den Zeichen von A und die Spalten denen von B entsprechen. Wir sprechen im folgenden kurz von „Quellen“ und „Kanälen“.

Sei nun eine Quelle $Q = (A, p)$ und ein Kanal $K = (A, B, R)$ gegeben. Wir setzen

$$r(a,b) := p(a) \cdot r_a(b).$$

Dann ist $(A \times B, r)$ ein zusammengesetzter Wahrscheinlichkeitsraum. Die zu A gehörige Randverteilung ist p, und es ist $r(b \mid a) = r_a(b)$. Die Randverteilung (B, q) gibt an, mit welcher Wahrscheinlichkeit die Elemente von B empfangen werden, während $r(a \mid b)$ als Wahrscheinlichkeit dafür gedeutet werden kann, daß a gesendet wurde, wenn b empfangen wird.

Wir können $H(A \mid B)$ ist das Maß der Ungewißheit über das gesendete Zeichen nach dem Empfang, $I(A,B)$ als Maß der Information über das gesendete Zeichen deuten.

Beispiel 3.2. Wir betrachten den Kanal BSC(α) und die Quelle $Q = (A, p) = (\{0,1\}, p(0) = w, p(1) = 1 - w)$. Zur Abkürzung setzen wir $\alpha' := 1 - \alpha$, $w' := 1 - w$. Es gilt

$$r(b \mid a) = \begin{cases} \alpha' & b = a, \\ \alpha & b \neq a. \end{cases}$$

Somit ergibt sich

$$H(B \mid a) = \alpha \log(1/\alpha) + \alpha' \log(1/\alpha') = H(\alpha, \alpha').$$

für $a = 0, 1$. Folglich ist auch $H(B \mid A) = H(\alpha, \alpha')$. Da

$$q(0) = \alpha' w + \alpha w', \qquad q(1) = 1 - q(0)$$

folgt

$$\begin{aligned} I_r(A,B) &= H(B,q) - H_r(B \mid A) \\ &= H\big(\alpha w' + \alpha' w, 1 - (\alpha w' + \alpha' w)\big) - H(\alpha, \alpha') \end{aligned}$$

Also ist $I_r(A,B)$ genau dann maximal, wenn $\alpha w' + \alpha' w = 1/2$ ist, und das ist genau dann der Fall, wenn $w = 1/2$ ist. Dann ist

$$I_r(A,B) = H(1/2, 1/2) - H(\alpha, \alpha') = 1 + \alpha \log \alpha + \alpha' \log \alpha'.$$

Eine wichtige Größe in der Informationstheorie ist die Kapazität eines Kanals:

Definition. Die Zahl

$$\varkappa(K) := \max\{I_r(A,B) : (A,p) \text{ Quelle mit Alphabet } A\}$$

heißt *Kapazität* des Kanals $K = (A, B, R)$.

Der Kanal kann also maximal soviel Information übertragen, wie seine Kapazität angibt. Dies entspricht dem intuitiven Kapazitätsbegriff. Die von uns eingeführte Kapazität wird endgültig durch den Kanal-Codierungssatz gerechtfertigt werden.

Beispiel 3.3. Wie wir oben gesehen haben, hat der Kanal BSC(α) die Kapazität $1+\alpha\log\alpha+\alpha'\log\alpha'$. Sie ist <1, wenn $0<\alpha<1$.

Man kann aus einfachen Quellen kompliziertere zusammensetzen. Die einfachste Konstruktion ist das direkte Produkt von $Q=(A,p)$ und $Q'=(A',p')$ nämlich

$$Q\times Q'=(A\times A',p''),\qquad p''(a,a')=p(a)p'(a').$$

Das n-fache direkte Produkt von Q mit sich selber bezeichnen wir durch Q^n. Mittels einer einfachen Rechnung (vergleichbar dem Beweis von Satz 2.6) sieht man

Satz 3.4.

$$H(Q\times Q')=H(Q)+H(Q')\quad \textit{und}\quad H(Q^n)=nH(Q).$$

Man kann sich $Q\times Q'$ als eine „Parallelschaltung“ von Q und Q' vorstellen, bei denen Q und Q' gleichzeitig jeweils ein Zeichen ausstoßen. Da unsere Quellen ohne Gedächtnis sind, stellt Q^n aber auch diejenige Quelle dar, die man erhält, wenn man in der von Q hervorgebrachten Zeichenfolge jeweils n aufeinander folgende Zeichen zu einem Zeichen von Q^n zusammenfaßt. Daher nennen wir Q^n auch die *n-te Erweiterung* von Q.

Entsprechend kann man auch das direkte Produkt zweier Kanäle definieren, das wie bei Quellen als Parallelschaltung verstanden werden kann:

Definition. Seien $K=(A,B,R)$ und $(A'B',R')$ Kanäle. Dann definieren wir das direkte Produkt $K\times K'$ durch

$$K\times K'=(A\times A',B\times B',R''),\qquad r\big(b,b'\big)\mid(a,a'))=R(a\mid b)r(a'\mid b').$$

Die *n-te Erweiterung von K* ist das n-fache direkte Produkt von K mit sich selbst.

In Analogie zu Satz 3.4 gilt

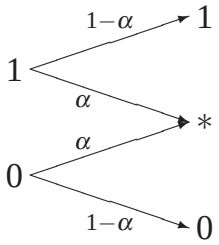
Satz 3.5.

$$\varkappa(K\times K')=\varkappa(K)+\varkappa(K')\quad \textit{und}\quad \varkappa(K^n)=n\varkappa(K).$$

Da wir den Satz nicht wirklich benötigen, überlassen wir den Beweis einer (etwas schwierigeren) Übungsaufgabe. Da auch unsere Kanäle ohne Gedächtnis sind, können wir K^n interpretieren als den Kanal, der entsteht, wenn man sowohl die Eingabe als auch die Ausgabe von K in Blöcke der Länge n zerschneidet.

Übungen

3.6. Bestimme die Kapazität des binären Löschkanals, der durch folgendes Diagramm beschrieben wird:



Jedes der Zeichen 0 und 1 wird also mit Wahrscheinlichkeit α ausgelöscht und mit Wahrscheinlichkeit $1-\alpha$ richtig übertragen.

3.7. Wir verallgemeinern den binären symmetrischen Kanal zum *symmetrischen Kanal K mit q Symbolen* und Fehlerwahrscheinlichkeit α; Für jedes der Symbole $b_1,\dots,b_q$, die zugleich Ein- und Ausgabealphabet bilden, sei

$$r(b_i \mid b_j) = \begin{cases} 1-\alpha, & i = j \\ \alpha/(q-1), & i \neq j. \end{cases}$$

Beweise

$$\varkappa(K) = \log q + \alpha \log \frac{\alpha}{p-1} + (1-\alpha)\log(1-\alpha).$$

3.8. Beweise Satz 3.5 in folgenden Schritten.

(a) Seien (A,p), (A',p') optimale Quellen für K und K', und die Wahrscheinlichkeitsverteilung u auf $A\times A'$ sei gegeben durch $u(a,a') = p(a)p'(a')$. Dann ist

$$I\big((A\times A',u),B\times B'\big) = I(A,B) + I(A',B').$$

Folgere $\varkappa(K\times K') \geq \varkappa(A) + \varkappa(A')$.

(b) Sei nun eine beliebige Wahrscheinlichkeitsverteilung v auf $A\times A'$ gegeben. Die zugehörigen Randverteilungen seien p und p'. Wir definieren eine neue Verteilung u auf $A\times A'$, indem wir

$$u(a,a') = p(a)p'(a')$$

setzen. Zeige nun, daß

$$I(A,B) + I(A',B') = I\big((A\times A',u),B\times B'\big) \geq I\big((A\times A',v),B\times B'\big)$$

ist. Die Quelle $(A\times A',u)$ ist also besser an $K\times K'$ angepaßt ist als $(A\times A',v)$. (Hier muß natürlich die Jensensche Ungleichung in geeigneter Form zum Einsatz kommen.)

(c) Schließe, daß $\varkappa(K\times K') = \varkappa(K) + \varkappa(K')$.

3.9. Der Kanal K_n enstehe durch die Serienschaltung von n Kanälen des gleichen Typs BSC(α):

$$\boxed{\text{BSC}(\alpha)} \to \cdots \to \boxed{\text{BSC}(\alpha)}$$

Zeige, daß die Serienschaltung zum Kanal BSC(β) mit

$$\beta = \frac{1}{2}\big(1-(1-2\alpha)^n\big)$$

äquivalent ist und $\lim_{n\to\infty} \varkappa(K_n) = 0$ ist, wenn $0 < \alpha < 1$.

ABSCHNITT 4

Codes mit variabler Wortlänge

Aufgabe der Quellencodierung ist es, eine Informationsquelle möglichst „effizient“ zu codieren, von Redundanz zu befreien. Ist der Übertragungskanal störungsfrei, so ist damit auch eine „optimale“ Kanalcodierung erreicht. Quellencodierung ist gleichbedeutend mit der Codierung für störungsfreie Kanäle; die Wahl des Codes ist nur von der Quelle abhängig. „Effizient“ bedeutet physikalisch, daß die Nachricht in möglichst kurzer Zeit oder mit möglichst geringem Energieaufwand übertragen wird. Bei der Datenspeicherung heißt ëffizient“‘, daß möglichst wenig Speicherplatz benötigt wird.

Man kann die Quellencodierung so interpretieren, daß man aus der Quelle und dem Codierer eine neue Quelle konstruiert, deren Entropie möglichst groß ist.

zusammengesetzte Quelle Q_1

Allerdings ist hier eine gewisse Vorsicht geboten, denn die neue Quelle kann oft nicht als Quelle ohne Gedächtnis angesehen werden. (Um dies zu präzisieren, muß man natürlich einen allgemeineren Begriff von Quelle einführen.)

Wir untersuchen zunächst quellenunabhängige Eigenschaften von Codes. Für eine Menge M bezeichnet M^* die Menge der Wörter über M, also

$$M^* = \bigcup \{M^n : n \in \mathbb{N},\ n \geq 0\}.$$

Definition. A, B seien endliche Alphabete. Eine *Codierung* von A durch B ist eine injektive Abbildung $\varphi : A \to B^*$. Man nennt $\varphi(A)$ den zu dieser Codierung gehörigen *Code*. Die Elemente von $\varphi(A)$ heißen *Codewörter*.

Ist $B = \{0,1\}$, so heißt die Codierung *binär*.

Beispiel 4.1. Sei $A = \{a_1, a_2, a_3, a_4\}$ und $B = \{0,1\}$. Wir definieren Codierungen von A durch B mit folgenden Tabellen:

(a)

a_1	00
a_2	01
a_3	10
a_4	11

(b)

a_1	0
a_2	10
a_3	110
a_4	111

(c)

a_1	0
a_2	01
a_3	10
a_4	11.

(d)

a_1	010
a_2	11000
a_3	01000
a_4	00110

Im ersten Fall besitzen alle Codewörter gleiche Länge, in den anderen Fällen nicht.

Jeder Codierung $\varphi : A \to B^*$ wird auf folgende Weise eine Abbildung $\varphi^* : A^* \to B^*$ zugeordnet:

$$\varphi^*(a_1 \cdots a_n) := \varphi(a_1) \cdots \varphi(a_n)$$

Einer Folge von Symbolen aus A wird die entsprechende Folge der Codewörter zugeordnet.

Beispiel 4.2. Bei den in Besipiel 4.1 eingeführten Codierungen ist

(a) $\varphi^*(a_1a_2a_1a_4) = 00010011$
(b) $\varphi^*(a_1a_2a_1a_4) = 0100111$
(c) $\varphi^*(a_1a_3a_2a_1) = 010010$ und $\varphi^*(a_2a_1a_1a_3) = 010010$

Beim Code (c) ist es offensichtlich nicht möglich, einen aus Codeworten zusammengesetzten Wort das codierte Wort zu rekonstruieren. Solche Codes sind offensichtlich nicht sehr brauchbar.

Definition. Eine Codierung φ heißt *eindeutig decodierbar*, wenn φ^* injektiv ist.

Die eindeutige Decodierung hängt nur vom Code ab. Wir sprechen daher von eindeutig decodierbaren Codes.

Dies trifft bei den bisher behandelten Beispielen zu auf (a) und (b), nicht aber auf (c). Beispiel (a) ist eindeutig decodierbar, da alle Codewörter gleiche Länge haben. Auf (b) kommen wir noch zurück; auch bei (b) ist die eindeutige Decodierbarkeit leicht einzusehen.

Wir betrachten nun die Codierung (d). Beim Empfänger laufe die Folge

$$01000\vdots11000$$

ein. Sie ist, wie angedeutet, aus Codeworten zusammengesetzt. Kann er schließen, daß a_3a_4 codiert wurde? Nein, denn mögliche Fortsetzungen sind

(1) $01000\vdots11000|010\ldots$

(2) $010\vdots00110\vdots00|110\ldots$

Im Fall (1) kann jetzt decodiert werden, im Fall (2) immer noch nicht. Bei der Decodierung tritt also eine Verzögerung auf. Bei diesem Beispiel gibt es nicht einmal eine obere Schranke für die Länge der Verzögerung, wie man durch Fortsetzung von (2) sehen kann. Es ist aber leicht zu sehen, daß auch (d) eindeutig decodierbar ist: kein Codewort ist Suffix eines anderen Codewortes, so daß man nach Erhalt der vollständigen Nachricht „von hinten“ decodieren kann.

Bemerkung 4.3. Es ist jedoch entscheidbar, ob ein Code eindeutig decodierbar ist (Algorithmus von Sardinas und Patterson). Es gibt auch einen Algorithmus mit dem entschieden werden kann, ob die Verzögerung beschränkt ist (siehe etwa [**KW**]).

Diese Probleme treten nicht auf bei einer Klasse von Codes, der Beispiel (b) oben angehört.

Definition. Ein Code heißt *sofort decodierbar*oder *Präfix-Code,* wenn kein Codewort Anfangsstück eines anderen Codewortes ist.

Offensichtlich gilt

Satz 4.4. *Jeder Präfix-Code ist eindeutig decodierbar.*

Sei B ein Alphabet mit D Symbolen, und seien $n_1,\dots,n_m$ natürliche Zahlen ≥ 1. Wir wollen die Frage untersuchen, wann ein eindeutig decodierbarer Code aus m Codeworten existiert, die Längen $n_1,\dots,n_m$ besitzen. Eine notwendige Bedingung nennt die *Ungleichung von Kraft-MacMillan*.

Satz 4.5. *$n_1,\dots,n_m$ seien die Längen der Codewörter eines eindeutig decodierbaren Codes über einem Codealphabet mit D Zeichen. Dann gilt*

$$\sum_{i=1}^{m} D^{-n_i} \leq 1$$

Beweis. Wir dürfen annehmen, daß $n_1 \leq \cdots \leq n_m$. Sei

$$S := \sum_{i=1}^{m} D^{-n_i}.$$

Dann ist nach dem allgemeinen Distributivgesetz

$$S^N = \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \dots \sum_{i_N=1}^{m} D^{-(n_{i_1}+\cdots+n_{i_N})}. \tag{1}$$

Wir ordnen die Summe nach Potenzen von D:

$$S^N = \sum_{k=Nn_1}^{Nn_m} f(N,k)D^{-k}, \qquad f(N,k) := \sum_{i_1+\cdots+i_N=k} 1$$

Die Wörter $c_{i_1}\dots c_{i_N}$, $(i_1,\dots,i_N) \in \{1,\dots,m\}^N$, $i_1+\cdots+i_N = k$, sind paarweise verschieden, da der Code eindeutig decodierbar, und ihre Anzahl ist gerade $f(N,k)$. Da es genau D^k Wörter der Länge k über B gibt, gilt

$$f(N,k) \leq D^k.$$

Folglich:

$$S^N \leq \sum_{k=Nn_1}^{Nn_m} D^k \cdot D^{-k} = \sum_{k=Nn_1}^{Nn_m} 1 = N(n_m-n_1)+1 \leq N(n_m-n_1+1).$$

Da $\sqrt[N]{N}\,\sqrt[N]{n_m-n_1+1} \to 1$ für $N \to \infty$, folgt die Behauptung. □

Triviale Beispiele zeigen, daß nicht jeder Code, der die Ungleichung in Satz 4.5 erfüllt, eindeutig decodierbar it. Aber es gilt wenigstens eine partielle Umkehrung, in der „eindeutig“ sogar durch „sofort“ ersetzt werden kann. Dies wird später von Bedeutung sein.

Satz 4.6. *D sei die Anzahl der Symbole über dem Codealphabet B, $n_1, \ldots, n_m$ natürliche Zahlen ≥ 1. Es gelte $\sum_{i=1}^{m} D^{-n_i} \leq 1$. Dann gibt es einen Präfix-Code $C = \{c_1, \ldots, c_m\}$ aus m Codeworten, deren Längen gerade $n_1, \ldots, n_m$ sind.*

Beweis. Wir beweisen den Satz durch Induktion über m. Der Fall $m = 1$ ist trivial. Sei $m > 1$. Wir können wieder $n_1 \leq \cdots \leq n_m$ annehmen.

Da $\sum_{i=1}^{m-1} D^{-n_i} < 1$, existiert nach Induktionsvoraussetzung ein Code $\{c_1, \ldots, c_{m-1}\}$ mit den Wortlängen $n_1, \ldots, n_{m-1}$.

Sei

$$S(c_i) := \{b_1 \cdots b_{n_m} : c_i \text{ ist Anfangsstück von } b_1 \cdots b_{n_m}\}.$$

Es ist $|S(c_i)| = D^{n_m - n_i}$ und wir erhalten

$$\sum_{i=1}^{m-1} |S(c_i)| = D^{n_m} \sum_{i=1}^{m-1} D^{-n_i} < D^{n_m}.$$

Folglich existiert ein Wort c_m der Länge n_m derart, daß kein c_i Anfangsstück von c_m ist. □

Wir wollen noch eine Folgerung aus 4.6 und 4.7 ziehen, die besagt, daß die Klasse der eindeutig decodierbaren Codes nicht wesentlich größer ist als die Klasse der Präfix-Codes.

Satz 4.7. *C sei ein eindeutig decodierbarer Code mit den Wortlängen $n_1, \ldots, n_m$. Dann existiert ein Präfix-Code mit den gleichen Wortlängen.*

Übungen

4.8. Konstruiere einen eindeutig decodierbaren binären Code, der weder Präfix- noch Suffix-Code ist.

4.9. Sei A ein Alphabet mit q Symbolen. Wieviele Wörter kann ein Präfixcode in A^* höchstens haben, wenn alle seine Wörter höchstens die Länge n haben?

ABSCHNITT 5

Codierung von Quellen

Ziel der Quellencodierung ist es, Codes mit möglichst kleiner mittlerer Wortlänge zu finden.

Definition. $Q = (A, p)$ sei eine Quelle, $\varphi : A \to B^*$ eine Codierung von A in B. Dann heißt

$$\bar{\lambda}(\varphi, Q) := \sum_{a \in A} p(a) \cdot \lambda(\varphi(a))$$

mittlere Codewortlänge von φ (bezüglich Q). (Für $b = b_1 \cdots b_n \in B^*$ ist $\lambda(b) = n$ die Wortlänge.)

Zunächst zeigt sich, daß $\bar{\lambda}(\varphi, Q)$ nicht kleiner sein kann als $H(Q)/\log D$, wenn φ eindeutig decodierbar ist:

Satz 5.1. *$Q = (A, p)$ sei eine Quelle, $\varphi : A \to B^*$ eine eindeutig decodierbare Codierung mit $|B| = D$.. Dann gilt:*

(a) $\bar{\lambda}(\varphi, Q) \geq H(Q)/\log D$

(b) $\bar{\lambda}(\varphi, Q) = H(Q)/\log D \iff p(a) = D^{-\lambda(\varphi(a))}$ *für alle $a \in A$.*

Beweis. Sei $S := \sum_{a \in A} D^{-\lambda(\varphi(a))}$ und $q(a) := D^{-\lambda(\varphi(a))} S^{-1}$. Gemäß Satz 2.4 ist

$$H(Q) = \sum_{a \in A} p(a) \log \frac{1}{p(a)} \geq \sum_{a \in A} p(a) \log \frac{1}{q(a)},$$

da $\sum q(a) = \sum p(a) = 1$. Unter Berücksichtigung der in Satz 4.5 bewiesenen Ungleichung folgt

$$\begin{aligned} \sum_{a \in A} p(a) \log \frac{1}{q(a)} &= \sum_{a \in A} p(a) \log S D^{\lambda(\varphi(a))} \geq \sum_{a \in A} p(a) \log D^{\lambda(\varphi(a))} \\ &= \sum_{a \in A} p(a) \lambda(\varphi(a)) \log D = \bar{\lambda}(\varphi, Q) \log D. \end{aligned}$$

Gleichheit gilt genau dann ein, wenn $S = 1$ und $p(a) = q(a)$, also $p(a) = D^{-\lambda(\varphi(a))}$. Für die letzte Aussage haben wir wieder Satz 2.4 benutzt. □

Wir müssen durch $\log D$ teilen, weil wir zur Bestimmung der Entropie den Logarithmus zur Basis 2 gewählt haben. Für $D = 2$ ist $\log D = 1$.

Wir wollen nun zeigen, daß es Präfix-Codes gibt, deren mittlere Codewortlänge $H(Q)(\log D)^{-1} + 1$ nicht überschreitet.

Satz 5.2. *Q sei eine Quelle, B ein Codealphabet, $D := |B|$. Dann existiert eine Präfix-Codierung $\varphi : A \to B*$ mit*

$$\frac{H(Q)}{\log D} \leq \bar{\lambda}(\varphi, Q) < \frac{H(Q)}{\log D} + 1.$$

Beweis. Zu jedem p_i wird n_i so bestimmt, daß

$$\frac{\log p_i}{\log D} \leq n_i < \frac{\log p_i}{\log D} + 1. \qquad (*)$$

Die linke Ungleichung garantiert mit Satz 4.6, daß eine Präfix-Codierung mit den Wortlängen n_i existiert. Die rechte Ungleichung ergibt dann

$$\bar{\lambda}(\varphi, Q) < \frac{H(Q)}{\log D} + 1. \qquad \square$$

Allerdings garantiert die Wahl von n_i gemäß $(*)$ nicht, daß für den so gefundenen Code die mittlere Codewortlänge minimal wird.

Beispiel 5.3. Wir wählen $Q = (A, p)$, $A = \{a_1, \ldots, a_6\}$, $B = \{0, 1\}$. Die Zahlen n_i werden gemäß Gleichung $(*)$ bestimmt:

i	$p(a_i)$	n_i	$\varphi(a_i)$
1	0,25	2	10
2	0,25	2	01
3	0,2	3	00(0)
4	0,15	3	111
5	0,10	4	1101
6	0,05	5	1100(0)

Die eingeklammerten Symbole können, ohne die sofortige Decodierbarkeit zu zerstören, weggelassen werden, und so kann die mittlere Codewortlänge reduziert werden.

Das wichtigste Ergebnis dieses Abschnitts ist der folgende Spezialfall des *Quellen-Codierungssatzes* von Shannon.

Satz 5.4. *Q sei eine Quelle. B ein Codealphabet, $D := |B|$. Dann existiert zu jedem $n \geq 1$ eine Präfix-Codierung φ_n der n-ten Erweiterung Q^n von Q derart, daß*

$$\lim_{n\to\infty} \frac{\bar{\lambda}(\varphi_n, Q^n)}{n} = \frac{H(Q)}{\log D}.$$

Beweis. Nach Satz 5.2 existiert eine Präfix-Codierung φ_n mit

$$\frac{H(Q^n)}{\log D} \leq \bar{\lambda}(\varphi_n, Q^n) < \frac{H(Q^n)}{\log D} + 1.$$

Wie in Abschnitt 3 festgestellt, ist $H(Q^n) = n \cdot H(Q)$. Es folgt

$$\frac{H(Q)}{\log D} \leq \frac{\overline{\lambda}(\varphi_n, Q^n)}{n} < \frac{H(Q)}{\log D} + \frac{1}{n}$$

Durch Übergang zum Grenzwert folgt die Behauptung. □

Dieser Satz besitzt eine weitreichende Konsequenz: Indem man für genügend hohes n die n-te Erweiterung von Q gemäß Satz 5.2 codiert, findet man Codierungen, deren mittlere Codewortlänge *pro Symbol von Q* beliebig nahe an $H(Q)/\log D$ liegt. Satz 5.4 zeigt, daß das von uns verwendete Maß für die Ungewißheit mit dem durch binäre Codierungen definierten Informationsgehalt übereinstimmt: es gilt

$$H(Q) = \inf\left\{\overline{\lambda}(\varphi, Q^n) : n \geq 1, \varphi \text{ ist eindeutige binäre Codierung von } Q^n\right\}.$$

Dies rechtfertigt die Definition der Entropie endgültig.

Beispiel 5.5. Sei $Q = (A, p)$, $A = \{s_1, s_2\}$ und $p(s_1) = 3/4$, $p(s_2) = 1/4$. Dann ist $H(Q) = 0.8113$. Wir codieren nacheinander $Q_1 = Q$, Q_2 und Q_3:

$Q_1 = Q$

a	$p(a)$	$\varphi(a)$
s_1	$\frac{3}{4}$	0
s_2	$\frac{1}{4}$	1

Q_2

a	$p(a)$	$\varphi(a)$
s_1s_1	$9/16$	0
s_1s_2	$3/16$	10
s_2s_1	$3/16$	110
s_2s_2	$1/16$	111

Q^3

a	$p(a)$	$\varphi(a)$	a	$p(a)$	$\varphi(a)$
$s_1s_1s_1$	$27/64$	0	$s_1s_2s_2$	$3/64$	11111
$s_1s_1s_2$	$9/64$	110	$s_2s_1s_2$	$3/64$	11110
$s_1s_2s_1$	$9/64$	101	$s_2s_2s_1$	$3/64$	11101
$s_2s_1s_1$	$9/64$	100	$s_2s_2s_2$	$1/64$	11100

Die mittleren Codewortlängen sind

$$\lambda(\varphi_1, Q) = 1, \qquad \overline{\lambda}(\varphi_2, Q^2) = 1.6875, \qquad \overline{\lambda}(\varphi_3, Q^3) = 2.46875,$$

so daß

$$\frac{\overline{\lambda}(\varphi_2, Q^2)}{2} = 0.84375, \qquad \frac{\overline{\lambda}(\varphi_3, Q^3)}{3} = 0.82292.$$

Zwei Begriffe, mit denen die Güte einer Codierung gemessen wird, sind *Effizienz* und *Redundanz*.

Definition. Sei Q eine Quelle, φ eine Codierung von Q in B und $D := |B|$. Dann heißt

$$E(\varphi, Q) := \frac{H(Q)}{(\log D)\lambda(\varphi, Q)}$$

die *Effizienz von* φ (bezüglich Q) und

$$R(\varphi, Q) := 1 - E(\varphi, Q).$$

heißt *Redundanz von* φ (bezüglich Q). Es gilt stets $0 \le E(\varphi, Q) \le 1$.

Für die gerade betrachteten Codierungen ergibt sich

$$E(\varphi^1, Q^1) = 0.8113, \qquad E(\varphi^2, Q^2) = 0.9615, \qquad E(\varphi^3, Q^3) = 0.9859.$$

Für die aus Quelle Q und Quellencodierer zusammengesetzte „Quelle“ Q_1 bedeutet $E(\varphi, Q) = 1$:

$$\log D = \frac{H(Q)}{\overline{\lambda}(\varphi, Q)}.$$

Wir können $H(Q)/\lambda(\varphi, Q)$ deuten als Entropie der „Quelle“ Q_1. Mit einem allgemeineren Begriff von „Quelle“ läßt sich dies präzisieren. Dabei ergibt sich dann, daß $H(Q_1) = \log D$ äquivalent dazu ist, daß Q_1 eine Quelle ohne Gedächtnis und die Wahrscheinlichkeit aller D Symbole gleich $1/D$ ist.

Huffman-Codierung. Einzig unbefriedigend an Satz 5.4 ist der wenig konstruktive Beweis. Es gibt aber einen Algorithmus, der für jede Quelle eine optimale Codierung findet:

Definition. Sei $Q = (A, p)$ eine Quelle, B ein Codealphabet und X die Menge der eindeutig decodierbaren Codierungen von A in B. Die Codierung $\varphi \in X$ heißt *optimal*, wenn

$$\overline{\lambda}(\varphi, Q) \le \overline{\lambda}(\psi, Q)$$

für alle $\psi \in X$ ist.

Wegen Satz 4.7 können wir uns bei der Suche nach optimalen Codierungen auf die Präfix-Codierungen beschränken.

Satz 5.6. *Unter den optimalen Codierungen einer Quelle Q befindet sich mindestens eine sofort decodierbare.*

Wir formulieren nun den *Algorithmus von Huffman* für binäre Codierungen. Er läßt sich auf beliebige Codealphabete übertragen; dabei wird aber der Nachweis der Optimalität um einiges mühevoller (siehe Aufgabe 5.11) .

(a) Besitzt Q genau zwei Symbole a und b, so codiert man a durch 0 und b durch 1.

(b) Besitzt Q die Symbole $a_1, \dots a_{n+1}$ mit $p(a_1) \ge \cdots \ge p(a_{n+1})$, so bildet man die Quelle $Q' = (A', p')$, $A' = \{a'_1, \dots, a'_n\}$, mit

$$p'(a'_i) := p(a_i) \text{ für } i = 1, \dots, n-1,$$
$$p'(a'_n) := p(a_n) + p(a_{n+1}).$$

(c) Man konstruiert eine optimale Präfix-Codierung φ' für Q' und setzt $c_i = \varphi'(a'_i)$.

(d) Nun definiert man $\varphi : A \to \{0,1\}^*$ durch

$$\begin{aligned} \varphi(a_i) &:= c_i \qquad i = 1,\ldots,n-1, \\ \varphi(a_n) &:= c_n 0, \\ \varphi(a_{n+1}) &:= c_n 1. \end{aligned}$$

Auf diese Weise wird also rekursiv eine Codierung der gegebenen Quelle konstruiert: Q' hat ein Symbol weniger als Q.

Für die Anwendung auf Q' muß das „neue“ Symbol so eingeordnet werden, daß die Voraussetzung in (b) eingehalten wird. Der Platz von a'_n ist aber möglicherweise nicht eindeutig bestimmt, und so kann man bei manchen Quellen optimale Präfix-Codierungen konstruieren, die sich wesentlich unterscheiden (vergleiche Aufgabe 5.9.

Satz 5.7. *Eine nach dem Verfahren von Huffman konstruierte Codierung ist sofort decodierbar und optimal.*

Beweis. Offensichtlich ist φ sofort decodierbar.

Für den Beweis der Optimalität nehmen wir wie im Algorithmus an, daß Q die Symbole $a_1,\ldots,a_{n+1}$ hat und $p(a_1) \geq \cdots \geq p(a_{n+1})$ ist. Wir beweisen zunächst, daß es zu jeder Präfix-Codierung ψ von Q eine ebenfalls sofort decodierbare Codierung χ gibt mit folgenden Eigenschaften:

(a) $\lambda(\chi,Q) \leq \lambda(\varphi,Q))$,
(b) $\lambda(\chi(a_1)) \geq \cdots \geq \lambda(\chi(a_{n+1}))$,
(c) $\lambda(\chi(a_n)) = \lambda(\chi(a_{n+1}))$,
(d) $\chi(a_n)$ und $\chi(a_{n+1})$ unterscheiden sich nur im letzten Symbol.

Die Bedingung (ii) können wir durch eventuelle Permutation der Codewörter erreichen. Durch die Zuordung der Codewörter in einer solchen Reihenfolge kann die mittlere Codewortlänge nicht zunehmen. Wenn $\chi(a_{n+1})$ dann länger ist als $\chi(a_n)$, können wir $\chi(a_{n+1})$ sogar um das letzte Symbol kürzen! Das so entstehende Wort muß von allen Codeworten $\chi(a_1),\ldots,\chi(a_n)$ verschieden sein, denn sonst wäre χ nicht eindeutig decodierbar. Es kann aber auch nicht echtes Anfangsstück eines dieser Wörter sein, denn dann wäre die Ungleichungskette (ii) verletzt.

Schließlich dürfen wir auch noch annehmen, daß $\chi(a_n)$ und $\chi(a_{n+1})$ sich nur im letzten Symbol unterscheiden. Wenn nämlich $\chi(a_n) = b0$ ist und $b1$ unter den Codeworten vorkommt, so vertauschen wir einfach dieses Codewort mit $\chi(a_{n+1})$. Andernfalls ersetzen wir $\chi(a_{n+1})$ durch $b1$.

Wir führen nun den Induktionsbeweis. Für $n = 1$ ist die Codierung offensichtlich optimal. Sei also $n \geq 2$. Wie im Algorithmus sei Q' aus Q konstruiert. Die für Q' konstruierte Codierung φ' ist nach Induktionsvoraussetzung optimal.

Wir nehmen an, φ sei nicht optimal. Dann existiert eine Präfix-Codierung ψ von Q mit

$$\overline{\lambda}(\psi, Q) < \overline{\lambda}(\varphi, Q).$$

Wir dürfen annehmen, daß ψ die Bedingungen (1) – (iv) oben erfüllt. Aus ψ konstruieren eine Präfix-Codierung ψ' von Q' mittels

$$\psi'(a'_i) := \psi(a_i) \qquad i = 1, \ldots, n-1$$
$$\psi'(a'_n) := \psi(a_n) \text{ gekürzt um das letzte Symbol.}$$

Der Code ψ' ist sofort decodierbar: Wegen der Eigenschaften (ii) und (iv) und weil wir binäre Codierungen betrachten, kann $\psi'(a_n)'$ nicht Anfangsstück eines der Wörter $\psi'(a'_i)$ sein für $i = 1, \ldots, n-1$.

Wir erhalten:

$$\begin{aligned}
\overline{\lambda}(\varphi', Q') &= \sum_{i=1}^{n} p(a'_i)\lambda(\varphi'(a'_i)) + p'(a'_n)\lambda(\varphi'(a'_n)) \\
&= \sum_{i=1}^{n} p(a_i)\lambda(\varphi(a_i)) + p(a_n)(\lambda(\varphi(a_n)) - 1) + p(a_{n+1})(\lambda(\varphi(a_{n+1})) - 1) \\
&= \overline{\lambda}(\varphi, Q) - (p(a_n) + p(a_{n+1})) \\
&> \overline{\lambda}(\psi, Q) - (p(a_n)) + p(a_{n+1}) = \overline{\lambda}(\psi', Q)
\end{aligned}$$

Dies ist ein Widerspruch zur Optimalität von φ'. (Bei der Rechnung haben wir Gleichung (iii) ausgenutzt.) □

Beispiel 5.8. Im folgenden Beispiel sind die im Rekursionsschritt entstehenden Werte $P'(a'_n)$ fett gedruckt. Die Indizes deuten an, welche Symbole vom Übergang von φ' zu φ an die schon vorhandenen Codeworte angehängt werden.

a_1	0.42	0.42	0.42	0.42	$\mathbf{0.58}_0$
a_2	0.3	0.3	0.3	0.3_0	0.42_1
a_3	0.1	0.1	$\mathbf{0.18}_0$	$\mathbf{0.28}_1$	
a_4	0.08	$\mathbf{0.1}_0$	0.1_1		
a_5	0.06_0	0.08_1			
a_6	0.04_1				

Zur Übersicht noch einmal die entstandenen Codewörter:

1	1	1	1	0
00	00	00	00	1
011	011	010	01	
0101	0100	011		
01000	0101			
01001				

Die linke Spalte gibt eine optimale Codierung von Q.

Übungen

5.9. Eine Quelle habe 4 Symbole, deren Wahrscheinlichkeiten durch $1/3, 1/3, 1/6, 1/6$ gegeben seien. Zeige, daß unterschiedliche Einordnungen des nach der ersten „Verkürzung“ entstandenen Symbols mit Wahrscheinlichkeit $1/3$ zu wesentlich verschiedenen binären Huffman-Codierungen führen.

5.10. Sei $Q = (A, p)$ eine Quelle mit den n Symbolen $a_1, \ldots, a_n$ und den Wahrscheinlichkeiten $p_1 \geq \cdots \geq p_n$. Wir setzen $r_1 = 0$ und $r_i = \sum_{j=1}^{i-1} p_j$ für $i \geq 1$ und wählen die natürlichen Zahlen $m_1, \ldots, m_n$, so daß

$$2^{-m_i} \leq p_i < 2^{-m_i+1}.$$

Nun betrachtet man die Dualentwicklung

$$r_i = 0, b_{i1} b_{i2} \ldots, \qquad i = 1, \ldots, n,$$

und ordnet a_i das Codewort $b_{i1} \ldots b_{im_i}$ zu. Zeige, daß die so erhaltene binäre Codierung φ eindeutig decodierbar ist und die Ungleichung

$$H(Q) \leq \overline{\lambda}(\varphi, Q) < H(Q) + 1$$

erfüllt, im allgemeinen aber nicht optimal ist.

5.11. Für die Quelle $Q = (A, p)$ mit den n Symbolen $a_1, \ldots, a_n$ und den Wahrscheinlichkeiten $p_1 \geq \cdots \geq p_n > 0$ und das Codealphabet $B = \{b_1, \ldots, b_D\}$ mit $D \geq 2$ Zeichen arbeitet der Algorithmus von Huffman wie folgt:

(i) Wenn die Quelle höchstens D Symbole hat, ordnen wir ihnen Codeworte der Länge 1 über B zu.

(ii) Andernfalls definieren wir die Zahl s durch die Bedingungen

$$s \in \{2, \ldots, D\}, \quad s \equiv n \mod D - 1.$$

Im ersten Schritt werden die s Symbole mit den kleinsten Wahrscheinlichkeiten zusammengefaßt.

In allen folgenden Schritten werden D Symbole mit den kleinsten Wahrscheinlichkeiten zusammmengefaßt.

(iii) Alles andere ist analog zum binären Fall.

Nach Wahl von s hat die „letzte“ zu codierende Quelle D Symbole (?), so daß der Algorithmus wenigstens eine Präfix-Codierung von Q liefert. Zeige, daß die so erhaltene Codierung optimal ist.

Anleitung: Es genügt, daß bei einer optimalen Codierung von Q mit den Wortlängen $m_1 \leq m_2 \leq \cdots \leq m_n$ die letzten s Worte alle gleiche Länge haben. Dann folgt die Optimalität der Huffman-Codierung wie im binären Fall.

Wir setzen

$$u = |\{j : M_j = m_n\}|$$

und haben $u \geq s$ zu beweisen. Der entscheidende Punkt ist die Kraftsch-MacMillansche Ungleichung, die man im binären Fall gar nicht braucht. Wir setzen dazu

$$\delta = D^{m_1} - \sum_{i=1}^{n} D^{m_n - m_i}.$$

Dann ist $\delta \geq 0$ eine ganze Zahl. Zeige:

(a) Falls $\delta \geq D-1$ ist, existiert eine Codierung mit den Wortlängen $m_1, \ldots, m_{n-1}$, $m_n - 1$.

(b) $\delta \equiv -u \mod D$ und $\delta \equiv 1-n \mod D-1$.

Daraus und aus der Definition von s kann man $u \geq s$ herleiten.

5.12. (a) Schreibe eine `Aribas`-Funktion, die für eine Quelle die binäre Huffman-Codierung durchführt. Eingabe für die Funktion sind die Zahl n der Zeichen von Q und Zahlen $m_1, \ldots, m_n$, die die Wahrscheinlichkeiten p_i durch $p_i = m_i/N$ definieren, $N = m_1 + \cdots + m_n$. Ergebnis der Funktion sei ein `array` C mit Komponenten des Typs `string`, wobei C_i das Codewort für a_i enthält.

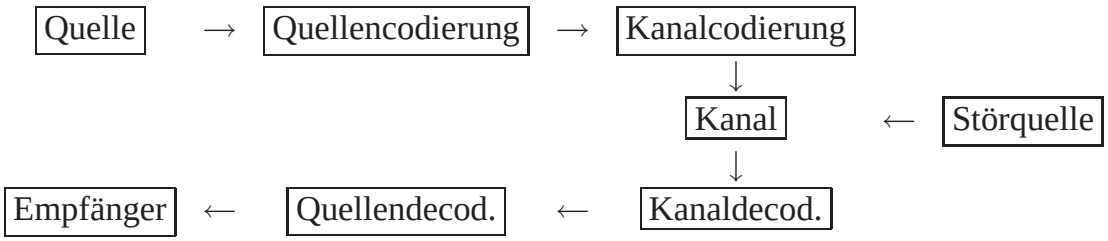
(b) Zähle die Häufigkeiten der ASCII-Codes $0, \ldots, 255$ in einem längeren Text-File und konstruiere einen Huffman-Code gemäß (a). Wie viele Bits erfordert die Kompression der Quelle mittels dieses Codes? Wie schneidet diese Codierung im Vergleich zum Beispiel mit `Winzip` ab?

Hinweis: Viele der ASCII-Zeichen kommen im File vermutlich nicht vor. Diese sollten bei der Ermittlung der Codierung nicht berücksichtigt werden.

ABSCHNITT 6

Fehlerkorrigierende Codes

In Abschnitt 1 haben wir das folgende Modell für die Nachrichtenübertragung entwickelt:



Die Quellencodierung wurde in vorangegangenen Abschnitten diskutiert. Dabei haben wir mit dem Quellen-Codierungssatz und dem Huffman-Algorithmus sehr befriedigende Ergebnisse erhalten. Aufgabe der Kanalcodierung ist es, die im Kanal übertragenen Nachrichten gegenüber Störungen zu sichern. Der Kanalcodierer fügt dabei gezielt Redundanz zu, die vom Decodierer dazu ausgenutzt wird, Übertragungsfehler zu erkennen und zu korrigieren.

Ein einfaches Beispiel für fehlersichernde Codierungen ist der Einsatz von Prüfziffern. Dieses Verfahren war schon lange in Gebrauch, bevor die Codierungstheorie als mathematische Disziplin entstanden ist.

Beispiel 6.1. Jedes Buch hat eine ISBN-Nummer (*International Standard Book Number*). Die eigentliche Information (wie Sprache, Verlag, Titel) ist in den ersten 9 Ziffern $a_1, \ldots, a_9$ enthalten. Die zehnte Ziffer ist durch die Kongruenz

$$a_{10} = \sum_{i=1}^{9} i \cdot a_i \mod 11$$

definierte Prüfziffer (wobei 10 mod 11 durch X dargestellt wird). Diese Prüfziffer sichert, daß zumindest folgende Fehler erkannt werden:

(i) die Veränderung einer einzigen Ziffer,
(ii) die Vertauschung zweier benachbarter Ziffern.

Gerade der Fehlertyp (ii) passiert dem „Kanal“ Mensch sehr häufig. Wenn ein Fehler erkannt wird, kann man notfalls beim Besteller rückfragen.

Ähnliche Prüfziffer-Verfahren kommen bei vielen anderen durch Zahlen repräsentierten Informationen zum Einsatz. Siehe [**Sch**] für weitere interessante Beispiele.

Immer dann, wenn eine Rückfrage möglich ist, ist die Fehlererkennung wichtiger als die Korrektur und wesentlich einfacher zu realisieren. Dies gilt insbesondere auch für die Datenübertragung in Computer-Netzwerken. Wir kommen darauf später zurück.

Wenn eine Rückfrage nicht möglich ist, wie zum Beispiel beim Abspielen einer CD oder der Übertragung von Bildern, die ein Satellit vom Saturn gemacht hat, müssen Codes eingesetzt werden, mit denen Fehler nicht nur erkannt, sondern sogar korrigiert werden können. Da im folgenden zum ersten Mal nicht nur die Menge $\{0,1\}$ zum Einsatz kommt, sondern algebraische Eigenschaften des Körpers aus zwei Elementen benutzt werden, führen wir die Bezeichnung $\mathbb{F}_2 = \{0,1\}$ ein.

Beispiel 6.2. Wir stellen uns (grob vereinfacht) vor, daß ein Satellit Schwarzweiß-Bilder zur Erde übertragen soll. Die Bilder bestehen aus einer Folge von Grauwerten. Jedem „Pixel“ wird dabei eine Zahl zwischen 0 und 7 zugeordnet, die binär durch ein 3-Tupel $(x_1,x_2,x_3) \in \mathbb{F}_2^3$ dargestellt wird. Die Information soll auf einem Kanal BSC(α) übertragen werden mit $\alpha = 0.01$. Die Wahrscheinlichkeit, daß ein 3-Tupel korrekt übertragen wird, ist dann

$$(1-\alpha)^3 \approx 0.97$$

Circa 3% aller Daten erreichen dann verfälscht den Empfänger.

Wir nehmen nun an, daß der Kanal die Zeichen doppelt so schnell übertragen kann, wie sie von der Kamera geliefert werden. Die einfachste Idee ist dann, jedes Bit zu wiederholen. Das erhöht die Wahrscheinlichkeit, Fehler zu erkennen, zwar schon spürbar, nutzt aber für die Korrektur nichts!

Stattdessen erfolgt die Codierung wie folgt. An die Informationsstellen x_1,x_2,x_3 werden drei Kontrollstellen x_4,x_5,x_6 angehängt, die mittels der Addition in $\mathbb{F}_2$ gebildet werden:

$$x_4 = x_1 + x_2, \quad x_5 = x_1 + x_3, \quad x_6 = x_2 + x_3.$$

Dies führt zu der Codierungstabelle

(0, 0, 0)	$\longrightarrow$	(0, 0, 0, 0, 0, 0)	(0, 1, 1)	$\longrightarrow$	(0, 1, 1, 1, 1, 0)
(0, 0, 1)	$\longrightarrow$	(0, 0, 1, 0, 1, 1)	(1, 0, 1)	$\longrightarrow$	(1, 0, 1, 1, 0, 1)
(0, 1, 0)	$\longrightarrow$	(0, 1, 0, 1, 0, 1)	(1, 1, 0)	$\longrightarrow$	(1, 1, 0, 0, 1, 1)
(1, 0, 0)	$\longrightarrow$	(1, 0, 0, 1, 1, 0)	(1, 1, 1)	$\longrightarrow$	(1, 1, 1, 0, 0, 0)

Der Vektor x wird gesendet und y empfangen. Dann bezeichnet man

$$e = y - x$$

als den *Fehlervektor* oder kurz *Fehler*. Aufgabe des Decodierers ist es den Fehler e aus dem empfangenen Signal zu bestimmen und dann $x = y - e$ zu rekonstruieren. Da jeder Vektor x in jedes y durch Übertragungsfehler überführt werden kann, ist man sich über den eingetretenen Fehler nie völlig sicher. Man ordnet y daher den Fehler e zu, der am wahrscheinlichsten aufgetreten ist, sofern dieser eindeutig bestimmt ist. Andernfalls muß man eine willkürliche Auswahl treffen oder auf die Decodierung von y verzichten.

In unserem Beispiel gilt für das Codeword x:

$$\begin{aligned} x_1 + x_2 + x_4 &= 0, \\ x_1 + x_3 + x_5 &= 0, \\ x_2 + x_3 + x_6 &= 0, \end{aligned}$$

und genau diejenigen Elemente $x \in \mathbb{F}_2^6$, die dieses homogene lineare Gleichungssystem erfüllen, gehören zum Code. Der Code ist also ein Untervektorraum von $\mathbb{F}_2^6$, und die obigen 3 Gleichungen fungieren als *Kontrollgleichungen* des Codes.

Also erfüllen y und e folgendes Gleichungssystem, durch das wir gleichzeitig das *Syndrom* (s_1, s_2, s_3) definieren:

$$\begin{aligned} e_1 + e_2 + e_4 &= y_1 + y_2 + y_4 =: s_1 \\ e_1 + e_3 + e_5 &= y_1 + y_3 + y_5 =: s_2 \\ e_2 + e_3 + e_6 &= y_2 + y_3 + y_6 =: s_3 \end{aligned}$$

Damit ist dem Decodierer das Syndrom (s_1, s_2, s_3) bekannt.´Zu ihm bestimmt er den Fehler, der für das Syndrom am wahrscheinlichsten ist, und dies ist bei einem Kanal BSC(α) mit $\alpha < 0.5$ der Fehlervektor mit der kleinsten Anzahl von Null verschiedener Einträge. Mit Ausnahme von $(s_1, s_2, s_3) = (1,1,1)$ ist dieser Fehlervektor stets eindeutig bestimmt, und zwar durch folgende Tabelle:

s		e		s		e
(0, 0, 0)	$\longrightarrow$	(0, 0, 0, 0, 0, 0)		(1, 1, 0)	$\longrightarrow$	(1, 0, 0, 0, 0, 0)
(1, 0, 0)	$\longrightarrow$	(0, 0, 0, 1, 0, 0)		(1, 0, 1)	$\longrightarrow$	(0, 1, 0, 0, 0, 0)
(0, 1, 0)	$\longrightarrow$	(0, 0, 0, 0, 1, 0)		(0, 1, 1)	$\longrightarrow$	(0, 0, 1, 0, 0, 0)
(0, 0, 1)	$\longrightarrow$	(0, 0, 0, 0, 0, 1)				

Das Syndrom 0 zeigt, daß kein Fehler aufgetreten ist, während jedes der anderen 6 Syndrome genau einem einfachen Fehler entspricht. Für das Syndrom $(s_1, s_2, s_3) = (1,1,1)$ erhält man als gleichwahrscheinliche Fehlervektoren:

$$\begin{aligned} e &= (1,\ 0,\ 0,\ 0,\ 0,\ 1) \\ e' &= (0,\ 1,\ 0,\ 0,\ 1,\ 0) \\ e'' &= (0,\ 0,\ 1,\ 1,\ 0,\ 0). \end{aligned}$$

Damit gilt: Jeder empfangene Vektor y mit nur einem Übertragungsfehler wird korrekt decodiert. Von den doppelten Fehlern wird noch einer richtig decodiert (je nach Auswahl von e, e' oder e''). Die Wahrscheinlichkeit einer korrekten Decodierung der gesendeten Nachricht ist also mindestens

$$(1-p)^6 + 6(1-p)^5 p + (1-p)^4 p^2 \approx 0.998.$$

Dies ist eine signifikante Verbesserung gegenüber der Situation ohne Codierung.

Bei der praktischen Beurteilung muß allerdings noch ein Aspekt erwähnt werden, der gerade bei der Nachrichtenübertragung von Satelliten eine Rolle spielt: Wenn nur drei Bits pro Codewort übertragen werden müssen, steht für jedes Bit doppelt soviel Energie zu Verfügung wie bei der Übertragung von 6 Bits. Dies ist bei der Ermittlung der Fehlerwahrscheinlichkeit zu berücksichtigen. Dennoch lohnen sich fehlerkorrigierende Codes. Man vergleiche hierzu [**vL**], §2.3.

Das gerade diskutierte Beispiel nutzt die lineare Algebra über $\mathbb{F}_2$. Wir werden die Theorie der *linearen Codes* in Abschnitt 8 systematisch einführen.

Blockcodes. Wir wollen die Grundsätze, die wir bei Beispiel 6.2 beachtet haben, nun allgemein für Codes mit fester Wortlänge formulieren.

Definition. Ein *Code* oder *Blockcode der Länge n* über einem Alphabet A ist eine Teilmenge von A^n.

Sei $K = (A, B, R)$ ein Kanal, $C \subset A^n$ ein Code der Länge n über A und C' eine Teilmenge von B^n. Dann heißt eine Abbildung $\Phi : C' \to C$ *Decodierungsschema* von C. Man nennt $\Phi_a := B^n - C'$ den *Decodierungsausfall*. Falls $C' = B^n$, heißt Φ *vollständig*.

Wir sagen, Φ sei ein *Maximum-Likelihood-Decodierungsschema*, wenn gilt:

$$r(w|\Phi(w)) = \max\{r(w|v) : v \in C\}.$$

Ein Maximum-Likelihood-Decodierungsschema ordnet also einem empfangenen Wort w dasjenige Wort $\Phi(w)$ zu, das mit größter Wahrscheinlichkeit im Kanal zu w wurde.

Man beachte, daß sich die Wahrscheinlichkeiten $r(w|v)$ auf den Kanal K^n beziehen. Das in Beispiel 6.2 betrachtete Decodierungsschema ist vollständig, vorausgesetzt man ordnet auch dem Syndrom $(1,1,1)$ einen Fehler zu.

Wegen der Möglichkeit der Quellencodierung können wir davon ausgehen, daß die Quelle gleichverteilt ist. Der Kanalcodierer bildet dann die Nachrichten der Quelle bijektiv auf C ab. Nach der Decodierung des empfangenen Wortes zu $c \in C$, muß die Kanalcodierung wieder rückgängig gemacht werden. Die dabei verwendeten bijektiven Abbildungen sind unter dem Gesichtspunkt der Fehlerkorrektur irrelevant und in der Praxis einfach zu realisieren.

Deshalb dürfen wir C selbst als Quelle mit Gleichverteilung ansehen. Das Maximum-Likelihood-Decodierungsschema erfüllt dann die Forderung nach größter mittlerer Wahrscheinlichkeit für richtige Decodierung. Ob man ein vollständiges Decodierungsschema wählt, hängt davon ab, ob es wichtiger ist, alle empfangenen Worte zu decodieren, oder bei machen Worten lieber auf eine Decodierung zu verzichten und auf diese Weise Decodierungsfehler zu vermeiden.

Wir setzen $\Psi(v) := \Phi^{-1}(\{v\})$ für alle $v \in C$ (dabei kann $\Phi^{-1}(\{v\}) = \emptyset$ sein). Dann ist $\Psi(v)$ die Menge derjenigen empfangenen Wörter, die zu v decodiert werden. Die Wahrscheinlichkeit für richtige Decodierung beim Senden von v ist

$$\sum_{w \in \Psi(v)} r(w|v),$$

und die *mittlere Wahrscheinlichkeit für richtige Decodierung* beträgt

$$\frac{1}{|C|} \sum_{v \in C} \sum_{w \in \Psi(v)} r(w|v) = \sum_{w \in C'} p(\Phi(w)) r(w|\Phi(w)).$$

Die Wahrscheinlichkeit für falsche Decodierung beim Senden von v ist

$$\sum_{w \in C' - \Psi(v)} r(w|v),$$

im Mittel also

$$e(C, \Psi) = \frac{1}{|C|} \sum_{v \in C} \sum_{w \in C' - \Psi(v)} r(w|v).$$

Sei $K = (B, B, R)$ ein Kanal mit gleichem Ein- und Ausgabealphabet (was für die Praxis keine Einschränkung bedeutet) aus $q = |B|$ Zeichen, der folgende Bedingung erfüllt:

$$r(b'|b) = \begin{cases} \alpha/(q-1) & b' \neq b, \\ 1-\alpha & b = b'. \end{cases}$$

Dies ist der *symmetrische Kanal* mit q Zeichen und Fehlerwahrscheinlichkeit α, den wir im Spezialfall $q = 2$ schon betrachtet haben. Man benutzt den Buchstaben q für die Anzahl der Codesymbole, weil $|B|$ meistens eine Potenz einer Primzahl p ist.

Der Hamming-Abstand. Die Wahrscheinlichkeit dafür, daß ein Wort v aus B^n bei der Übertragung über K^n in das Wort w verfälscht wird, beträgt

$$\left(\frac{\alpha}{q-1}\right)^d (1-\alpha)^{n-d},$$

wobei d die Zahl der Stellen ist, an denen v und w sich unterscheiden. Unter der Annahme $1-\alpha > \alpha/(q-1)$ ist, fällt diese Wahrscheinlichkeit monoton mit wachsendem d. Damit ist d ein gutes Maß für den „Abstand“ von v und w:

Definition. Sei B eine Menge. Der *Hamming-Abstand* von $v, w \in B^n$ ist gegeben durch

$$d(v,w) = |\{i : v_i \neq w_i\}|.$$

Offensichtlich ist B^n ein metrischer Raum mit dem Hamming-Abstand. Wir können daher viele für metrische Räume gebräuchlichen Begriffe einfach übernehmen. Zum Beispiel gehören zur *Kugel des Radius e um v* alle $w \in B^n$, die sich von v an höchstens e Stellen unterscheiden.

Unsere obige Überlegung zeigt:

Satz 6.3. *Sei $K = (B,B,R)$ ein symmetrischer Kanal mit $q = |B|$ und Irrtumswahrscheinlichkeit α, für den $1-\alpha > \alpha/(q-1)$ gilt. Ferner sei $C \subset B^n$ ein Code. Dann sind äquivalent:*

(a) *Φ ist ein Maximum-Likelihood-Decodierungsschema für C.*
(b) *Es gilt*

$$d(w,\Phi(w)) = \min\{d(w,v) : v \in C\}.$$

Damit bestimmt der Hamming-Abstand über die Fehlerkorrektur.

Definition. Man nennt

$$d_{\min}(C) = \min\{d(v,w) : v,w \in C,\ v \neq w\}$$

den *Minimalabstand* von C.

Der Minimalabstand des Codes in Beispiel 6.2 beträgt 3. Am Minimalabstand können wir ablesen, welche Fehler korrigiert werden können:

Satz 6.4. *Mit den Bezeichnungen von Satz* 6.3 *sei $C \subset B^n$ ein Code mit Minimalabstand d. Dann gilt:*

(a) *Ein vollständiges Maximum-Likelihood-Decodierungsschema für C korrigiert alle e-fachen Fehler mit $2e < d$.*
(b) *Ein Decodierungsschema mit dem Ausfall $B^n \setminus C$ erkennt alle e-fachen Fehler mit $e < d$.*

Beweis. (a) Sei v das gesendete, w das empfangene Wort, das sich von v an e Stellen unterscheidet. Nach Satz 6.3 wird w mit Sicherheit genau dann zu v decodiert, wenn es kein $v' \in C$, $v \neq v'$, mit $d(w,v') < e$ gibt. Dies folgt aus der Dreickecksungleichung:

$$d(v',w) \leq d(v,v') - d(v,w) \geq d - e > e$$

für alle $v' \in C$, $v' \neq v$.

(b) Ein Codewort kann nur mit mindestens d Fehlern in ein anderes Codewort verfälscht werden. □

Etwas verkürzt können wir Satz 6.4 so ausdrücken: (a) C korrigiert alle e-fachen Fehler mit $2e < d$; (b) C erkennt alle e-fachen Fehler mit $e < d$. Freilich kann man nicht beide Ziele gleichzeitig erreichen, sondern wird oft einen Kompromiß eingehen: Für $e < c \leq d - e$ kann man *gleichzeitig* alle e-fachen Fehler korrigieren und alle c-fachen Fehler korrigieren.

In geometrischer Terminologie besagt der obige Satz: Der Code C korrigiert alle e-fachen Fehler, wenn sich die Kugeln mit Radius e um zwei verschiedene Codeworte nicht schneiden, und er erkennt alle e-fachen Fehler, wenn die Kugel mit Radius e um ein beliebiges Codewort kein anderes Codewort enthält.

Einen Code C der Länge n mit M Worten über dem Alphabet A ist ein

$$(n, M, d)\text{-Code}$$

wenn C den Minimalabstand d hat.

Schranken für Codes. Es ist klar, daß wir bei festem n die Anzahl der Codeworte in der Regel um so kleiner machen müssen, je größer d sein soll. Eine erste Beziehung zwischen M und d liefert die *Kugelpackungs-* oder *Hamming-Schranke*.

Satz 6.5. *Sei C ein (n, M, d)-Code über dem Alphabet B mit q Symbolen. Falls $d \geq 2e+1$ für $e \in \mathbb{N}$, so ist*

$$q^n \geq M \sum_{k=0}^{e} \binom{n}{k} (q-1)^k.$$

Beweis. Sei $v \in C$, und $w \in B^n$ habe Abstand k von v. Dann unterscheiden sich w und v an genau k Stellen, und diese k Stellen können auf genau $\binom{n}{k}$ Weisen ausgesucht werden. Wenn diese Stellen festliegen, kommen für w genau $(q-1)^k$ Elemente in B^n in Frage. Daher hat die „Sphäre“ des Radius k um v genau $\binom{n}{k}(q-1)^k$ Elemente, und die Kugel des Radius e hat dann

$$\sum_{k=0}^{e} \binom{n}{k} (q-1)^k$$

Elemente. Nach Voraussetzung sind die Kugeln des Radius e um zwei verschiedene Elemente von C disjunkt, und die Anzahl der Elemente in der Vereinigung der Kugeln gibt die rechte Seite der Ungleichung. Sie kann höchstens q^n Elemente betragen. □

Man nennt einen Code C aus gutem Grund *perfekt*, falls in Satz 6.5 die Gleichheit gilt. Triviale Beispiele für perfekte Codes sind der aus einem einzigen Codewort bestehende *triviale Code*, der Code $C = B^n$ und – fast ebenso trivial – der *Wiederholungscode*

$$\{(0,\ldots,0),\ (1,\ldots,1)\} \subset \mathbb{F}_2^n$$

bei ungeradem n. Perfekte Codes definieren offensichtlich optimale Kugelpackungen von B^n.

Nichttriviale Beispiele für perfekte Codes sind die Hamming-Codes mit den Parametern $(n = (q^k - 1)/(q-1), q^{n-k}, 3)$, wobei das Codealphabet B der Körper $\mathbb{F}_q$ mit q Elementen ist, der binäre $(23, 2^{12}, 7)$ und der ternäre $(11, 3^6, 5)$-Golay-Code. Die Hamming-Codes werden wir noch diskutieren; siehe [**vL**] oder [**Wi**] zu den Golay-Codes.

Es gilt folgender bemerkenswerte Satz, der 1973 von Tietäväinen und unabhängig von Zinov'ev und Leont'ev bewiesen wurde: *Ein nicht trivialer perfekter Code über einem Alphabet, dessen Elementezahl eine Primzahlpotenz q ist, hat die gleichen Parameter wie ein Hamming- oder Golay-Code.* Später wurde gezeigt: *Für $d \geq 7$ ist der binäre $(23, 2^{12}, 7)$-Golay Code der einzige nichttriviale perfekte Code.*

Bemerkung 6.6. Sei $\varepsilon : B^n \to B^n$ eine Abbildung, die die Bedingung

$$d(\varepsilon(v), \varepsilon(w)) = d(v, w)$$

für alle $v, w \in B^n$ erfüllt; ε ist also eine Isometrie bezüglich des Hamming-Abstands. Dann sind die Codes C und $\varepsilon(C)$ in jeder Hinsicht äquivalent und haben speziell die gleichen Parameter. Diesen Äquivalenz-Begriff haben wir gerade verwendet. Beispiele von Isometrien ε sind offensichtlich Permutationen der Komponenten oder Permutationen von B, die komponentenweise auf B^n fortgesetzt werden.

Es gibt noch eine weitere sehr einfache Schranke für Codes, die *Singleton*-Schranke:

Satz 6.7. *Sei C ein (n, M, d)-Code über dem Alphabet B mit q Symbolen. Dann gilt*

$$d \leq n - \log_q M + 1.$$

Beweis. Sei $\pi : B^n \to B^{n-d+1}$ die Projektion, die jedes Wort um seine letzten $d-1$ Komponenten kürzt. Die Beschränkung von π auf C ist injektiv, denn zwei verschiedene Codeworte müssen sich in mehr als $d-1$ Stellen unterscheiden. Also ist $M \leq q^{n-d+1}$, und Logarithmieren ergibt die Behauptung. □

Codes, die denen in der Singleton-Schranke Gleichheit gilt, heißen *MDS-Codes* (maximum distance separable).

Neben den Blockcodes werden bei Anwendung häufig auch sogenannte *convolutional codes* eingesetzt; siehe dazu [**vL**].

Übungen

6.8. Wir betrachten den Körper $F = \mathbb{F}_q$ mit q Elementen und natürliche Zahlen $m, n \geq 1$. Es sollen Vektoren v der Länge mn über F gesendet werden. Wir betrachten einen solchen Vektor als eine $m \times n$-Matrix. Um sie gegen Übertragungsfehler

zu schützen, erweitern wir v zu einer $(m+1)\times(n+1)$-Matrix v', indem wir

$$v'_{ij} = v_{ij}, \qquad i,j = 1\ldots,n,$$
$$v'_{in+1} = -\sum_{j=1}^{n} v_{ij}, \quad i = 1\ldots,m,$$
$$v'_{m+1j} = -\sum_{i=1}^{m} v_{ij}, \quad j = 1\ldots,n+1$$

setzen. Sei $C = \{v' : v \in F^{mn}\}$.

(a) Zeige, daß C ein Untervektorraum von $F^{(m+1)(n+1)}$ ist.

(b) Welche Dimension, wieviele Elemente und welchen Minimalabstand hat C?

(c) Konstruiere ein Decodierungsschema, das alle einfachen Fehler korrigiert. Welche Wahrscheinlichkeit für richtige Decodierung wird im Fall $q = 2$ beim Kanal $\mathrm{BSC}(\alpha)$ erreicht?

Im Fall $q = 2$ haben wir v zeilen- und spaltenweise *Paritätsbits* hinzugefügt.

6.9. Zeige, daß ein perfekter Code mit mindestens zwei Codeworten ungeraden Minimalabstand hat.

6.10. (a) Für $q \le 27$, $n \le 100$ und die ungeraden d, $3 \le d \le n-1$ bestimme man alle Tripel (q,n,d) für die gilt:

$$\sum_{k=0}^{e} \binom{n}{k} (q-1)^k \quad \text{teilt} \quad q^n \qquad (e = (d-1)/2),$$

zum Beispiel mit einem `Aribas`-Programm. Diese Bedíngung ist offensichtlich notwendig für die Existenz eines perfekten Codes der Lönge n und der Minimaldistanz d über einem Alphabet mit q Elementen.

(b) Zur Kontrolle: Dabei muß auch $q = 2$, $n = 90$, $d = 5$ auftreten. Ein perfekter Code C mit diesen Parametern (der dann 2^{70} Worte hätte) existiert aber nicht, was bewiesen werden soll.

Wir nehmen das Gegenteil an. Man darf dann voraussetzen, daß C den Nullvektor enthält (weshalb?). Jetzt betrachte man

$$U = \{v \in \mathbb{F}_2^{90} : v_1 = v_2 = 1,\ d(v,0) = 3\},$$
$$D = \{c \in C : c_1 = c_2 = 1,\ d(v,0) = 5\}.$$

und bilde

$$E = \{(v,c) : v \in U,\ c \in D,\ d(v,c) = 2\}.$$

Zeige nun:

(i) Zu jedem $v \in U$ gibt es genau ein Element $c \in D$ mit $(v,c) \in E$.
(ii) Zu jedem $c \in D$ gibt es genau 3 Elemente $v \in U$ mit $(v,c) \in E$.

Dies führt auf einen Widerspruch!

6.11. Seien $e, a \in \mathbb{N}$ und sei $C \subset B^n$ ein Code des Minimalabstands $d \geq 2e + a + 1$. Zeige: Der Code C kann gleichzeitig e Fehler und a Auslöschungen korrigieren.

ABSCHNITT 7

Der Fundamentalsatz der Informationstheorie

Wir haben im vorangegangenen Abschnitt an einem Beispiel gesehen, daß man mit Hilfe von Blockcodes $C \subset B^n$ Nachrichten sehr wirksam gegen Übertragungsfehler schützen kann. Andererseits können wir dann nicht alle Elemente von B^n als Nachrichten zu nutzen, sondern müssen eine Auswahl treffen. Je kleiner C im Vergleich zu B^n ist, desto weniger Information transportieren die Codeworte. Das präzisieren wir mit folgender

Definition. Die *Informationsrate* eines Blockcodes $C \subset B^n$ ist

$$\rho(C) = \frac{\log|C|}{n \log q} \qquad (q = |B|).$$

Eine gleichverteilte Quelle mit $|C|$ Symbolen hat Entropie $\log|C|$. Andererseits kann mit den Worten aus B^n eine Quelle der Entropie $n \log q$ codiert werden. Die Rate mißt also, wie gut die maximale Entropie von B^n durch C ausgenutzt wird.

Eine passende Vergleichsgröße zur Infornmationsrate von C ist die relative Kapazität eines Kanals $K = (B, B, R)$, die wir durch

$$\varkappa_{\text{rel}}(K) = \frac{\varkappa(K)}{\log q}$$

definieren. Die maximale Kapazität eines solchen Kanals ist ja $\log q$ (bei Störungsfreiheit), und die relative Kapazität mißt, welcher Anteil davon nutzbar ist.

Durch einen Code C mit M Worten ist eine Quelle $Q = (B^n, p)$ für den Kanal K^n gegeben, nämlich die mit

$$p(c) = \begin{cases} 1/M, & c \in C, \\ 0, & c \notin C. \end{cases}$$

Die Entropie dieser Quelle ist $\log M$. Die Signale dieser Quelle werden über den Kanal K^n übertragen, und es ist klar, daß wir höchstens dann einen Übertragungsverlust vermeiden können, wenn $H(Q) < \varkappa(K^n) = n\varkappa(K)$ ist; nach Definition der Kapazität überträgt K^n ja höchstens die Information $\varkappa(K^n) = \varkappa(K)$. Es gilt

$$H(Q) < \varkappa(K^n) \iff \frac{H(Q)}{n \log q} < \frac{\varkappa(K)}{\log q},$$

so daß

$$H(Q) < \varkappa(K^n) \iff \rho(C) < \varkappa_{\text{rel}}(K).$$

Die Hamming- und die Singleton-Schranke zeigen, daß ein hohe Übertragungssicherheit gewährleistender Minimalabstand (bei festem n) mit einer kleinen Informationsrate erkauft werden muß. Umso überraschender ist es, daß sich durch

(i) Verwendung von Blockcodes C genügend großer Länge n,
(ii) der Übertragung mittels der n-ten Erweiterung eines Kanals K mit Eingabealphabet B, und
(iii) Verwendung eines Maximum-Likelihood-Decodierungsschemas

zwei Ziele *gleichzeitig* erreichen lassen:

(a) Eine Informationsrate beliebig wenig unterhalb der relativen Kanalkapazität $\varkappa_{\mathrm{rel}}(K)$ und
(b) eine beliebig kleine Fehlerwahrscheinlichkeit (>0).

Dies ist die Aussage des *Kanal-Codierungssatzes* von Shannon, den man mit Fug und Recht auch als Fundamentalsatz der Informationstheorie bezeichnen kann. Er rechtfertigt die Definition der Kanalkapazität auf nachdrückliche Weise. Der Einfachheit halber beschränken wir uns auf den Kanal $\mathrm{BSC}(\alpha)$. In diesem Fall ist $\log q = 1$, so daß Kapazität und relative Kapazität als Zahlenwerte übereinstimmen.

Für den Beweis des Satzes brauchen wir die *Ungleichung von Rand*, mit deren Hilfe wir die Anzahl der Elemente in einer Kugel bezüglich des Hamming-Abstands in $\{0,1\}^n$ abschätzen können.

Satz 7.1. *Für alle λ mit $0 \le \lambda \le 1/2$ gilt*

$$\sum_{k=0}^{\lfloor \lambda n \rfloor} \binom{n}{k} \le 2^{nh(\lambda)},$$

wobei $h(\lambda) = H(\lambda, 1-\lambda)$.

Beweis. Sei $\lambda' = \lfloor \lambda n \rfloor / n$. Dann gilt $\lambda' \le \lambda$, und daher ist $h(\lambda') \le h(\lambda)$. Wir dürfen also $\lambda = \lambda'$ annehmen. In diesem ist λn eine ganze Zahl, und es gilt

$$\begin{aligned} 1 = [\lambda + (1-\lambda)]^n &\ge \sum_{k=0}^{\lambda n} \binom{n}{k} \lambda^k (1-\lambda)^{n-k} \\ &\ge (1-\lambda)^n \sum_{k=0}^{\lambda n} \binom{n}{k} \left(\frac{\lambda}{1-\lambda}\right)^{\lambda n} \\ &= \lambda^{\lambda n} (1-\lambda)^{n(1-\lambda)} \sum_{k=0}^{\lambda n} \binom{n}{k}. \end{aligned}$$

Folglich ist

$$\sum_{k=0}^{\lambda n} \binom{n}{k} \le \lambda^{-\lambda n} (1-\lambda)^{-n(1-\lambda)}.$$

Wenn wir die Logarithmen zur Basis 2 bilden, liefert dies die Behauptung. □

Für den Beweis des Satzes von Shannon erweitern wir den Begriff des Blockcodes etwas. Wir gehen davon aus, daß M verschiedene Nachrichten $A_1,\dots,A_m$ gesendet werden sollen und ordnen jeder Nachricht ein Codewort $x_i \in \{0,1\}^n$ zu, wobei es nicht notwendig ist, daß die x_i paarweise verschieden sind. Der Code ist also eine Folge der Länge M in $\{0,1\}^n$. Das Wort x_i wird über den Kanal $\mathrm{BSC}(\alpha)$ gesendet und dann nach dem Maximum-Likelihood-Schema decodiert. Zu jedem i betrachten wir das Ereignis E_i, das eintritt, wenn das empfangene Wort bei Senden von x_i nicht zu A_i decodiert wird, und die zugehörige Wahrscheinlichkeit

$$e(A_i) = p(E_i),$$

Die mittlere Fehlerwahrscheinlichkeit des Codes ist dann

$$e(C) = \frac{1}{M}\sum_{i=1}^{M} e(A_i)$$

und die maximale Fehlerwahrscheinlichkeit ist

$$e_{\max}(C) = \max_i e(A_i).$$

Satz 7.2. *Sei $\varkappa(\alpha) = 1 - H(\alpha, 1-\alpha)$ die Kapazität des binären symmetrischen Kanals* $\mathrm{BSC}(\alpha)$*, $0 \le \alpha < 1/2$, und ρ eine reelle Zahl, $0 < \rho < \varkappa(\alpha)$. Ferner sei eine Folge (M_n) ganzer Zahlen mit $M_n \ge 1$ für alle n und $M_n \le 2^{n\rho}$ für $n \gg 0$ gegeben. Dann existiert eine Folge (C_n) von binären Blockcodes der Länge n mit M_n Codeworten, so daß*

$$\lim_{n\to\infty} e_{\max}(C_n) = 0.$$

Beweis. Der Beweis benutzt Prinzipien der Wahrscheinlichkeitsrechnung, und deshalb ist es sinnvoll, wenn wir uns ihrer Sprache bedienen.

Wir betrachten zunächst ein festes n, setzen $M = M_n$ und bilden einen Code C, indem wir M Codeworte $x_1,\dots,x_M$ aus $\{0,1\}^n$ zufällig auswählen. Ferner fixieren wir ein $r \in \mathbb{N}$, $r > 0$ und betrachten folgendes Decodierungsschema Φ: Ein empfangenes Wort y wird zu A_i decodiert, wenn es genau ein i gibt, für das x_i in der Hamming-Kugel

$$K_r(y) = \{z \in \{0,1\}^n : d(z,y) \le r\}$$

ist. Andernfalls decodieren wir y zu A_1. Es ist klar, daß ein Maximum-Likelihood-Decodierungsschema für kein gesendetes Codewort eine schlechtere Wahrscheinlichkeit für richtige Decodierung liefert. Daher reicht es aus, die Fehlerwahrscheinlichkeit das Schema Φ (für geeignetes r) abzuschätzen.

Das Codewort x_i werde gesendet und y empfangen. Das Ereignis E_i eines Decodierungsfehlers kann nur eintreten, wenn

$$d(x_i,y) > r \tag{2}$$

gilt oder

$$d(x_j, y) \leq r \text{ für ein } j \neq i \tag{3}$$

ist. Wir bezeichnen die Ereignisse, die durch (2) und (3) beschrieben werden, mit F und G, so daß $E_i \subset F \cup G$. Es gilt

$$p(E_i) \leq p(F \cup G) \leq p(F) + p(G).$$

Für das Ereignis G ist

$$p(G) = p(x_j \in K_r(y) \text{ für ein } j \neq i).$$

Da die Codewörter zufällig ausgewählt werden, ist die Wahrscheinlichkeit, daß $x_j \in K_r(x)$ liegt, gleich

$$\frac{1}{2^n}|K_r(x)| = \frac{1}{2^n}\sum_{k=0}^{r}\binom{n}{k}.$$

Damit kann $p(G)$ durch

$$p(G) \leq \frac{(M-1)}{2^n}\sum_{k=0}^{r}\binom{n}{k} \tag{4}$$

abgeschätzt werden.

Sei $\varepsilon > 0$ gegeben. Nach eventuellem Verkleinern von ε können wir annehmen, daß

$$\rho < \varkappa(\alpha+\varepsilon) < \varkappa(\alpha) \tag{5}$$

ist. Für

$$r = \lfloor n\alpha + n\varepsilon \rfloor$$

ergibt sich mit Ungleichung (4) und Satz 7.1, daß

$$p(G) \leq \frac{M}{2^n}2^{nh(\alpha+\varepsilon)} = M2^{-n(1-h(\alpha+\varepsilon))} = M2^{-n\varkappa(\alpha+\varepsilon)}.$$

Nun schätzen wir $p(F)$ ab. Sei U die zufällige Variable, die die Anzahl der Übertragungsfehler zählt. Sie ist binomial verteilt mit den Parametern n und p. Nach Definition von F ist

$$p(F) = p(U > r).$$

Unter Berücksichtigung von $V(U) = n\alpha(1-\alpha)$ erhalten wir mit der Tschebyschewschen Ungleichung:

$$p(F) = p(U > n\alpha + n\varepsilon) \leq p(|U - n\alpha| > n\varepsilon) \leq \frac{V(U)}{n^2\varepsilon^2} = \frac{\alpha(1-\alpha)}{n\varepsilon^2}$$

Die Fehlerwahrscheinlichkeit erfüllt also die Ungleichung

$$p(E_i) \leq p(F) + p(G) \leq \frac{\alpha(1-\alpha)}{n\varepsilon^2} + M2^{-n(1-h(\alpha+\varepsilon))}.$$

Für $M \leq 2^{n\rho}$ folgt

$$p(E_i) \leq \frac{\alpha(1-\alpha)}{n\varepsilon^2} + M2^{-n\varkappa(\alpha+\varepsilon)} \leq \frac{\alpha(1-\alpha)}{n\varepsilon^2} + 2^{-n(\varkappa(\alpha+\varepsilon)-\rho)}$$

wenn ist. Gemäß (5) ist $\varkappa(\alpha+\varepsilon) > \rho$, so daß $p(E_i) < \varepsilon$ für hinreichend großes $n \geq N(\varepsilon)$.

Wir haben nun folgendes gezeigt: Die Wahrscheinlichkeit für falsche Decodierung, *gemittelt über alle Codes C mit M_n Worten und alle Codeworte $x \in C$*, ist $< \varepsilon$ für alle $n \geq N(\varepsilon)$. Daraus folgt, daß $e(C_n) < \varepsilon$ für mindestens einen Code C_n mit $|C| = M_n$, sobald $n \geq N(\varepsilon)$.

Im Satz wird behauptet, daß wir für hinreichend großes n einen Code C_n finden können, für den sogar $e_{\max}(C_n) < \varepsilon$ ist. Das geht jetzt mit einem kleinen Trick.

Sei $\varepsilon' = \varepsilon/2$, $M_n' = 2M_n$ und $\rho < \rho' < \varkappa(p)$. Für genügend großes n ist dann $M_n' \leq 2^{n\rho'}$ und wir finden einen Code C_n' der Länge M_n', dessen mittlere Fehlerwahrscheinlichkeit $< \varepsilon'$ ist. Für mindestens die Hälfte der Codeworte in C_n' muß dann die individuelle Fehlerwahrscheinlichkeit $< \varepsilon$ sein, und wir wählen dementsprechend C_n aus C_n' aus. □

Bemerkung 7.3. (a) Der Satz besitzt auch eine Umkehrung, der unsere anfangs gemachte Beobachtung hinsichtlich $\rho(C)$ und $\varkappa_{\text{rel}}(K)$ präzisiert: Ist $M_n \geq 2^{\sigma n}$ für ein $\sigma > \varkappa(\alpha)$ und alle n, so gibt es ein $\varepsilon > 0$ mit $e(C_n) \geq \varepsilon$ für jede Folge (C_n) von Blockcodes der Längen M_n, $n \in \mathbb{N}$. Wir verweisen dazu auf [**We**] oder [**McE**]. In [**McE**] wird der Satz von Shannon für allgemeinere Kanäle bewiesen.

(b) Das gröbste Argument in unserem Beweis des Satzes ist die Tschebyschewsche Ungleichung. Wenn man sie durch bessere Abschätzungen ersetzt, kann man zeigen, daß die maximale Fehlerwahrscheinlichkeit mit wachsendem n sogar exponentiell gegen 0 geht.

(c) Der Beweis des Satzes von Shannon gibt keinerlei Ansatz für die Konstruktion guter Codes. Selbst wenn der Beweis Konstruktionshinweise gäbe, würde dies vermutlich nichts nützen. Denn es bleibt das Problem, die Decodierung durchzuführen. Besitzt der Code keine Systematik, so ist dies nur mittels einer Zuordnungstabellen möglich, die jedoch etwa bei $|B| = 2^8$, $n = 32$ in einem CD-Spieler nicht realisierbar ist. Man muß den Codes also eine „Struktur“ geben. Dies werden wir in den nächsten Abschnitten verfolgen.

Die Bedeutung des Satzes für die Praxis wird natürlich auch dadurch geschmälert, daß bei jeder Anwendung die Länge des Codes von vornherein physikalisch begrenzt ist.

Übungen

7.4. Ein binärer symmetrischer Kanal K kann 300 Kbit pro Sekunde mit einer Fehlerwahrscheinlichkeit von 0.01 pro Bit übertragen. Wieviele Bits pro Sekunde darf eine Informationsquelle Q höchstens liefern, wenn die Daten mit beliebig großer Sicherheit den Empfänger erreichen sollen? (Dem Aufwand für Codierung und Decodierung seien dabei keine Grenzen gesetzt.)

7.5. Gegeben sei der Kanal $K = \mathrm{BSC}(0.01)$. Man bestimme ein n, für das ein Blockcode der Länge n mit Informationsrate $\rho > \varkappa(K) - 10^{-3}$ und mittlerer Fehlerwahrscheinlichkeit $< 10^{-6}$ existiert. (Es wird nicht verlangt, das kleinstmögliche n zu finden.)

ABSCHNITT 8

Lineare Codes

In diesem Abschnitt beginnen wir, Codes mit algebraischer Struktur zu betrachten.

Definition. Sei $\mathbb{F} = \mathbb{F}_q$ der Körper mit q Elementen. Ein Untervektorrraum von $\mathbb{F}^n$ heißt *linearer Code der Länge n*. Sind k die Dimension und d der Minimalabstand, so sagen wir, C sei ein $[n,k]$-Code oder $[n,k,d]$-Code über $\mathbb{F}$ oder, noch kürzer, ein $[n,k,d]_q$-Code.

Ältere Bezeichnungen für lineare Codes sind (insbesondere im Fall $q = 2$) *Gruppencode* oder *verallgemeinerter Paritätskontrollcode*.

Bereits bei der Bestimmung des Minimalabstands zeigt sich ein kleiner Vorteil linearer Codes. Dazu benutzen wir das Gewicht der Codewörter:

Definition. Für $v \in \mathbb{F}^n$ sei

$$\mathrm{Tr}(v) = \{i : v_i \neq 0\}$$

der *Träger* von v und

$$\mathrm{wt}(v) = d(v,0) = |\mathrm{Tr}(v)|$$

das *Hamming-Gewicht* von v.

Bei linearen Codes stimmen Minimalgewicht und Minimalabstand überein:

Satz 8.1. *Sei C sei linearer Code. Dann gilt:*

$$d_{\min}(C) = \min\{\mathrm{wt}(v) : v \in C\}.$$

Beweis. Es gilt $d(v,w) = d(v-w,0)$ und mit v und w liegt auch $v-w$ in C. (Wir haben gerade die Translationsinvarianz der Hamming-Metrik ausgenutzt.) □

Untervektorräume kann man zwei Arten beschreiben: (i) durch Angabe einer Basis, (ii) als Lösungsraum eines homogenen linearen Gleichungssystems. Wir betrachten zunächst die Beschreibung mit Hilfe einer Basis.

Definition. Sei C sei ein $[n,k]$-Code. $v_1 = (v_{11},\ldots,v_{1n}),\ldots,v_k = (v_{k1},\ldots,v_{kn})$ eine Basis von C. Dann heißt die Matrix

$$\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{k1} & \cdots & v_{kn} \end{pmatrix}$$

eine *erzeugende Matrix* von C.

Es ist klar, daß ein Code im allgemeinen verschiedene erzeugende Matrizen besitzt.

Satz 8.2. *Sei C ein $[n,k]$-Code, G eine erzeugende Matrix von C. Dann ist $\varphi : \mathbb{F}^k \to \mathbb{F}_n$*

$$\varphi(v) = vG$$

eine lineare Abbildung und $\varphi(\mathbb{F}^k) = C$.

Dies ist uns aus der linearen Algebra bekannt.
Wir definieren $\pi_{i_1 \ldots i_j} : \mathbb{F}^n \to \mathbb{F}^j$ durch

$$\pi_{i_1 \ldots i_j}(v) := (v_{i_1}, \ldots, v_{i_j}).$$

Sei C ein $[n,k]$-Code und $i_1, \ldots, i_k \in \{1, \ldots, n\}$. Gilt $\pi_{i_1 \ldots, i_k}(\mathbb{F}^n) = \mathbb{F}^k$, so heißen $i_1, \ldots, i_k$ *Informationsstellen* des Codes und $\{1, \ldots, n\} - \{i_1, \ldots, i_k\}$ *Kontrollstellen*. Sind $1, \ldots, k$ Informationsstellen, so heißt C *systematisch*. Die erzeugende Matrix hat Rang k. Daher gibt es k Spalten, die linear unabhängig sind, und die zugehörigen Indizes $i_1, \ldots, i_k$ bilden dann Informationsstellen. Also besitzt jeder $[n,k]$-Code Informationsstellen. Durch Permutation der Komponenten von K^n kann man immer erreichen, daß der Code systematisch wird. Ein systematischer Code besitzt eine erzeugende Matrix der Form

$$G = (E_k \mid G') \qquad (E_k = (k \times k)\text{-Einheitsmatrix}).$$

Die durch G gegebene Codierung $\varphi : \mathbb{F}^k \to \mathbb{F}^n$ heißt ebenfalls *systematisch*.

Definition. Sei C sei ein $[n,k]$-Code. Eine $m \times n$-Matrix H heißt *Kontrollmatrix* von C, wenn gilt:

$$v \in C \iff vH^\top = 0.$$

Während die erzeugende Matrix für die Codierung gebraucht wird, ist die Kontrollmatrix für die Decodierung wichtig. Bereits die Existenz der Kontrollmatrix zeigt, daß die Fehlererkennung bei einem linearen Code einfach durch Ausführen einer linearen Abbildung realisiert werden kann.

Offensichtlich gilt

Satz 8.3. *Sei C ein $[n,k]$-Code. Eine $m \times n$-Matrix H ist genau dann Kontrollmatrix von C, wenn sie den Rang $n-k$ besitzt und $GH^\top = 0$ für eine (und damit für jede) erzeugende Matrix G von C ist.*

Der Lösungsraum des homogenen linearen Gleichungssystems $xH^\top = 0$ hat ja die Dimension $n - (n-k) = k$ und enthält C. Also stimmt er mit C überein. Ist $G = (E_k \mid G')$ erzeugende Matrix von C, so ist

$$H = ((-G')^\top \mid E_{n-k})$$

Kontrollmatrix von C, denn H erfüllt die Bedingung von Satz 8.3 und ist daher eine Kontrollmatrix.

Mit Hilfe der Kontrollmatrix können wir den Minimalabstand eines linearen Codes bestimmen.

Satz 8.4. *Sei H eine Kontrollmatrix des linearen Codes C. Genau dann C besitzt C Minimalabstand d, wenn je $d-1$ Spalten von H linear unabhängig sind, aber d Spalten existieren, die linear abhängig sind.*

Beweis. Sei $v \in C$ und seien $i_1, \ldots, i_u$ die Indizes der von 0 verschiedenen Komponenten von v. Dann sind die Spalten $i_1, \ldots, i_u$ von H linear abhängig. Wenn wir für v einen Vektor des Gewichts d wählen, erhalten wir d linear abhängige Spalten von H. Umgekehrt muß $u \geq d$ sein, wenn je $d-1$ Spalten linear unabhängig sind. □

Klassische Codes. Wir beschreiben nun zwei Typen klassischer Codes.

Beispiel 8.5. Aus jedem 1-dimensionalen Untervektorraum von $\mathbb{F}^k$ wählen wir einen Vektor $v \neq 0$. Da es genau $n = (q^k-1)/(q-1)$ solche Untervektorräume gibt, können wir mit diesen Vektoren als Spalten eine $k \times n$-Matrix H bilden. Im Fall $q = 2$ gibt es in jedem 1-dimensionalen Untervektorraum genau ein $v \neq 0$, und im Fall $q = 2$, $k = 3$ ist

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

Die Spaltenvektoren sind so gewählt, daß keine zwei linear abhängig sind. Andererseits gibt es (bei $k \geq 2$) drei linear abhängige Spalten, denn aus jedem Untervektorraum der Dimension 2 von $\mathbb{F}^k$ sind mindestens 3 Vektoren vertreten. Also hat der Code C mit Kontrollmatrix H den Minimalabstand 3. Da H den Rang k hat, hat C die Dimension $n-k$.

Der so konstruierte Code C ist ein $[n, n-k, 3]_q$-*Hamming-Code*. Man prüft unmittelbar nach, daß er die Kugelpackungsgleichung erfüllt. Daher ist C perfekt.

Die Codes des folgenden Beispiels waren insbesondere in der Frühzeit der Codierungstheorie im praktischen Einsatz, da es für sie ein sehr effizientes Decodierungsverfahren gibt.

Beispiel 8.6. Sei $\mathbb{F} = \mathbb{F}_2$. Zu $0 \leq r \leq m$ definiert man die *binären Reed-Muller-Codes* $\mathrm{RM}(r,m)$ unter Verwendung der in einer Aufgabe 8.12 besprochenen Konstruktion

$$C_1 \ltimes C_2 = \{(v_1, v_1+v_2) : v_1 \in C_1,\ v_2 \in C_2\}.$$

für lineare $[2n, k_i, d_i]$-Codes C_i, $i = 1,2$. Wir wissen, daß $C_1 \ltimes C_2$ ein $[n, k_1+k_2, \min(2d_1, d_2)]$-Code ist.

Man wählt für $\mathrm{RM}(0,m)$ den $[2^m,1,2^m]$-Wiederholungscode und setzt $\mathrm{RM}(m,m)=\mathbb{F}^{2^m}$. Dann definiert man $\mathrm{RM}(r,m)$ für $m\geq 1$ und $r=1,\ldots,m-1$ rekursiv durch

$$\mathrm{RM}(r,m)=\mathrm{RM}(r,m-1)\ltimes\mathrm{RM}(r-1,m-1).$$

Per Induktion folgt unmittelbar, daß $\mathrm{RM}(r,m)$ die Länge 2^m hat und den Minimalabstand

$$d=\min(2\cdot 2^{m-1-r},2^{m-1-(r-1)})=2^{m-r}.$$

Daß

$$\dim\mathrm{RM}(r,m)=\sum_{i=0}^{r}\binom{m}{i}$$

ist folgt ebenfalls leicht per Induktion unter Benutzung des Additionstheorems der Binomialkoeffizienten.

Der Reed-Muller-Code $\mathrm{RM}(1,5)$ ist ein $[32,6,16]_2$-Code. Er wurde berühmt durch seine Verwendung bei der Mariner-Expedition 1969–1976 für die Übermittlung von Bildern zur Erde.

Man kann Reed-Muller über allen endlichen Körpern definieren. Dies erfordert aber etwas mehr Aufwand.

Neue Codes aus alten Codes Wichtige Operationen für Codes sind das Erweitern, Punktieren und Kürzen:

Definition. Sei C ein linearer Code der Länge n. Dann heißt

$$\bar{C}=\{(v,a)\in\mathbb{F}^{n+1}:\sum_{i=1}^{n}v_i+a=0\}$$

Erweiterung von C. Man nennt

$$\dot{C}=\dot{C}_i=\pi_{1,\ldots,i-1,i+1,\ldots,n}(C)$$

die *Punktierung* von C (an der Stelle i) und

$$\check{C}=\check{C}_i=\pi_{1,\ldots,i-1,i+1,\ldots,n}(\{v\in C:v_i=0\})$$

die *Verkürzung* von C (um die Stelle i).

Es ist nicht schwer zu sehen, daß für die Parameter der modifizierten Codes gilt:

$$\begin{array}{llll}\hat{n}=n+1, & \hat{k}=k, & & d\leq\hat{d}\leq d+1,\\ \dot{n}=n-1, & \dot{k}=k, & d-1\leq\dot{d}\leq d, & \\ \check{n}=n-1, & \check{k}=k-1, & \check{d}=d. & \end{array}$$

Dabei haben wir bei $\dot{C}$ vorausgesetzt, daß $C\cap\mathrm{Kern}(\pi_{1,\ldots,i-1,i+1,\ldots,n})=\{0\}$ ist (was bei $d\geq 2$ sicherlich zutrifft), und bei $\check{C}$, daß C Codeworte v mit $v_i\neq 0$ enthält. In einem wichtigen Spezialfall gilt $\hat{d}=d+1$, nämlich dann, wenn C ein binärer Code

mit ungeradem Minimalabstand ist. Insbesondere besitzen die erweiterten binären Hamming-Codes den Minimalabstand 4.

Man kann die obigen Operationen manchmal verwenden, um die Existenz gewisser Codes mit Hilfe der bekannten Schranken auszuschließen. Zum Beispiel schließt die Schranke von Plotkin (vergleiche Aufgabe 8.13) die Existenz eines $[14,6,7]_2$-Codes C nicht aus, wohl aber die Existenz eines $[13,5,7]_2$-Codes. Da man einen $[13,5,7]_2$-Code durch Verkürzen aus C gewinnen könnte, kann C nicht existieren. (Mit der Hamming-Schranke kann man allerdings direkt sehen, daß kein $[14,6,7]_2$-Code existiert.)

Bemerkung 8.7. Für die Lineare Algebra sind Untervektorräume U_1, U_2 von V mit gleicher Dimension gleichwertig, da es einen Automorphismus α von V mit $\alpha(U_1) = U_2$ gibt. In der Codierungstheorie ist neben der Dimension auch der Minimalabstand eine bestimmende Größe, und daher sind in der Codierungstheorie nur solche Automorphismen von Interesse, die den Hamming-Abstand unverändert lassen. Man vergleiche dazu Bemerkung 6.6. Es ist nicht schwer zu sehen, daß solche Automorphismen bezüglich der kanonischen Basis gerade durch *monomiale Matrizen* gegeben werden: dies sind Matrizen, die in jeder Spalte und jeder Zeile genau ein Element $\neq 0$ haben. Mit anderen Worten: die linearen Isometrien bezüglich der Hamming-Metrik sind gerade diejenigen Automorphismen α, zu denen es eine Permutation π von $\{1,\ldots,n\}$ gibt und $a_i \in \mathbb{F}$, $a_i \neq 0$, so daß

$$\alpha(v) = (a_1 v_{\pi(1)}, \ldots, a_n v_{\pi(n)}).$$

Lineare Codes $C_1, C_2 \subset \mathbb{F}^n$ heißen *äquivalent*, wenn ein solcher Automorphismus α mit $\alpha(C_1) = C_2$ existiert.

Bei $q > 2$ ist die Wahl der Kontrollmatrix für einen Hamming-Code durch die Parameter nicht mehr eindeutig bestimmt, nicht einmal bis auf die Reihenfolge der Spalten. Aber alle Hamming-Codes mit den gleichen Parametern sind im obigen Sinn äquivalent.

Bemerkung 8.8. Die Betrachtung von Codes über den Körpern $\mathbb{F}_q$ für $q > 2$ ist keineswegs nur von theoretischem Interesse. Zumindest Codes über Körpern $\mathbb{F}_{2^m}$ werden in der Praxis eingesetzt, so etwa (mit $m = 8$) bei CDs.

Elemente aus $\mathbb{F}_{2^m}$ werden durch binäre m-Tupel dargestellt, und jeder $\mathbb{F}_{2^m}$-Untervektorraum der Dimension k von $\mathbb{F}_{2^m}^n$ hat als $\mathbb{F}_2$-Vektorraum die Dimension mk. Ein $[n,k]_{2^m,d}$-Code ist daher ein $[mn,mk]_2$-Code mit Minimaldistanz $\geq d$. Die Minimaldistanz über $\mathbb{F}_2$ zu ermitteln, ist allerdings ein schwieriges Problem.

Der technische Vorteil von Codes über $\mathbb{F}_{2^m}$ liegt in ihrer Fähigkeit, lange binäre Fehlerbündel erkennen oder sogar korrigieren zu können. Ein Fehlervektor e heißt dabei *Fehlerbündel der Länge* f, wenn

$$f = \max \mathrm{Tr}(e) - \min \mathrm{Tr}(e) + 1$$

ist. Fehlerbündel entstehen dann, wenn die (zeitliche oder räumliche) Ausdehnung einer Störung länger ist als die eines einzelnen Signals. Dies ist bei vielen Funkverbindungen der Fall, aber gilt auch für Kratzer auf der CD.

Ein binäres Fehlerbündel der Länge $f \leq gm+1$ kann nur maximal $g+1$ aufeinander folgende Symbole aus $\mathbb{F}_{2^m}$ treffen, also ein Fehlerbündel der Länge höchstens $g+1$ über $\mathbb{F}_{2^m}$ auslösen. Wenn der $[n,k,d]_{2^m}$-Code C Fehlerbündel der Länge $g+1$ erkennen kann (und dies ist sicher für $g+1 \leq d-1$ erkennen kann, so erkennt er binäre Fehlerbündel der Länge $\leq m(g+1)+1$. Eine analoge Aussage gilt für die Korrektur.

Eine weitere wirksame Maßnahme zur Behandlung von Fehlerbündeln ist die *Codeverschachtelung* (englisch *Interleaving*). Dabei schreibt man die zu sendenden Codeworte der Länge n zeilenweise in eine Matrix mit t Zeilen und sendet die Matrix dann spaltenweise. Man nennt t die *Tiefe* der Verschachtelung. Ein Fehlerbündel der Länge $\leq t$ kann dann in jedem Codewort höchstens einen Fehler verursachen. Bereits bei $d \geq 3$ kann ein Fehlerbündel der Länge $\leq m$ korrigiert werden.

Decodierung mittels Restklassenführer. Wir besprechen jetzt ein einfaches Decodierungsverfahren für lineare Codes, das allerdings nur kleine Werte von $n-\dim C$ praktikabel ist. Dazu charakterisieren wir Maximum-Likelihood-Decodierungsschemata zunächst in der Sprache der Linearen Algebra. Die Voraussetzungen über den Kanal K sind die gleichen wie bei Satz 6.3: K ist symmetrisch mit q Symbolen. Zu einem empfangenen Wort y betrachten wir dabei den *Fehler* $e_y = y - \Phi(x)$.

Satz 8.9. *Sei $C \subset \mathbb{F}^n$ ein linearer Code und Φ ein Decodierungsschema, das jedem $y \in \mathbb{F}^n$ den Fehler e_y zuordnet. Dann sind äquivalent:*

(a) *Φ ist ein Maximum-Likelihood-Decodierungsschema.*

(b) *Es gilt*

$$e_y \in y+C \quad \textit{und} \quad \mathrm{wt}(e_y) = \min\{wt(z) : z \in y+C\}.$$

Beweis. Da $d(y,x) = \mathrm{wt}(y-x)$ ist, gilt

$$d(\Phi(y),y) = \min\{d(x,y) : x \in C\} \iff \mathrm{wt}(y-\Phi(y)) = \min\{\mathrm{wt}(y-x) : x \in C\}.$$

Wenn x durch C läuft, dann läuft $y-x$ gerade durch $y-C = y+C$. Dies zeigt sofort die Äquivalenz von (a) und (b), wenn man berücksichtigt, daß ein Maximum-Likelihood-Decodierungsschema nach Satz 6.3 gerade durch die Bedingung auf der linken Seite der Äquivalenz gekennzeichnet ist. □

Ein Element e_y, das innerhalb der Restklasse $y+C$ Minimalgewicht hat, nennt man einen *Restklassenführer* von C. Dieser ist im allgemeinen nicht eindeutig bestimmt. Eine negative Erkenntnis aus Satz 8.9 ist, daß es man in jeder Restklasse von C höchstens einen Fehler korrigieren kann: Gleichgültig, welches Wort

$z \in y+C$ empfangen wird, ihm wird als Fehler stets der gewählte Restklassenführer zugeordnet.

Beispiel 8.10. Wir betrachten den $[4,2,2]_2$-Code

$$C = \{(0,0,0,0),\ (0,0,1,1),\ (1,1,0,0),\ (1,1,1,1)\}.$$

In der folgenden Tabelle sind die Zeilen die Restklassen von C, und die Elemente in der ersten Spalte die jeweiligen Restklassenführer (die hier außer für C selbst) nicht eindeutig bestimmt sind.

0000	0011	1100	1111
0001	0010	1101	1110
0100	0111	1000	1011
1010	1001	0110	0101

Man braucht, aber gar keine Tabelle, in der alle Elemente von $\mathbb{F}^n$ vorkommen, denn die Restklassen von C werden durch die Syndrome repräsentiert. Sei dazu H eine Kontrollmatrix von C. Dann nennen wir

$$\sigma(y) = yH^\top$$

das *Syndrom* von $y \in \mathbb{F}^n$ (bezüglich H). Es gilt:

$$z \in y+C \iff z-y \in C \iff \sigma(z-y) = 0 \iff \sigma(y) = \sigma(z).$$

Also entsprechen sich Restklassen von C und die Syndrome $\sigma(y) \in \mathbb{F}^{n-k}$ ($k = \dim C$) eindeutig, und man kann nach einem Maximum-Likelihood-Schema decodieren, wenn man eine Tabelle hat, die jedem Syndrom den Führer der entsprechenden Restklasse zuordnet.

Im obigen Beispiel erhält man mit der Kontrollmatrix H folgende Zuordnung von Restklassenführern und Syndromen:

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{array}{c|c} 0000 & 00 \\ \hline 0001 & 01 \\ 0100 & 10 \\ 1010 & 11 \end{array}$$

Bei der Bestimmung einer solchen Tabelle durchläuft man zweckmäßig die Elemente von $\mathbb{F}^n$ mit steigendem Gewicht und hört auf, sobald alle Syndrome getroffen sind (oder alle Fehler bis zu dem Gewicht, zu dem man korrigieren möchte, abgearbeitet sind).

Schranken für lineare Codes. Die in Abschnitt 6 diskutierten Schranken von Hamming und Singleton gelten natürlich speziell für lineare Codes, und die Singleton-Schranke kann man für $[n,k,d]$-Codes speziell so formulieren:

$$d \leq n-k+1$$

Wir wollen diese Abschnitt mit der *unteren* Schranke von Varshamov-Gilbert für den bei linearen Codes erreichbaren Minimalabstand beschließen.

Satz 8.11. *Falls*

$$\sum_{i=0}^{t-1} \binom{n-1}{i} (q-1)^i < q^{n-k+1},$$

so existiert ein $[n,k,d]_q$ *Code* C *mit* $d \geq t$.

Beweis. Wir beweisen dies durch Induktion über k. Für $k = 0$ ist die Behauptung trivial. Wir können also davon ausgehen, daß es einen $[n,k-1,d']$-Code C_k mit $d' \geq t$ gibt. Die Hamming-Kugeln vom Radius $t-1$ um die Codeworte $v \in C_{k-1}$ enthalten höchstens

$$g^{k-1} \sum_{i=0}^{t-1} \binom{n-1}{i} (q-1)^i < q^n$$

Worte in $\mathbb{F}^n$. Es gibt also ein Wort w, das zu allen $v \in C_{k-1}$ mindestens den Abstand t hat. Wir setzen nun

$$C_k = C_{k-1} + \mathbb{F}w.$$

Sei $x = v + aw \in C_k$, $v \in C_{k-1}$, $a \in \mathbb{F}$, $a \neq 0$. Dann ist

$$\mathrm{wt}(x) = \mathrm{wt}(-a^{-1}x) = \mathrm{wt}(a^{-1}v - w) = d(w, -a^{-1}v) \geq t,$$

denn $-a^{-1}v \in C$. □

Übungen

8.12. Für $i = 1,2$ sei C_i ein $[n,k_i,d_i]$-Code über dem Körper K. Zeige:

$$C_1 \ltimes C_2 = \{(v_1, v_1 + v_2) \in K^{2n} : v_i \in C_i\}$$

ist ein $[2n, k_1 + k_2, \min(2d_1, d_2)]$-Code über K.

8.13. Sei C ein $[n,k,d]_q$-Code. Beweise die *Schranke von Plotkin:*

$$d \leq \frac{n(q-1)q^{k-1}}{q^k - 1}.$$

Anleitung: Wir betrachten die i-te Komponente v_i aller Wörter $v \in C$. Wenn $v_i \neq 0$ für mindestens ein $v \in C$, dann gilt $v_i \neq 0$ für genau $(q-1)q^{k-1}$ Wörter $v \in C$. Vergleiche nun Minimalgewicht und durchschnittliches Gewicht der Wörter in $C \setminus \{0\}$.

8.14. Sei C ein $[n,k]$-Code über $\mathbb{F} = \mathbb{F}_q$. Wir definieren den *zu* C *dualen Code* $C^\perp$ durch

$$C^\perp = \{w \in \mathbb{F}^n : \sum_{i=1}^{n} w_i v_i = 0 \text{ für alle } v \in C\}.$$

(a) Zeige, daß $C^\perp$ ein $[n, n-k]$-Code ist.

(b) Wie stehen Erzeuger- und Kontrollmatrizen von C und $C^\perp$ in Zusammenhang?

(c) Zeige $C = C^{\perp}$ für den erweiterten binären $[8,4,4]$-Hamming-Code.

8.15. Sei C ein $[n,k]_q$-Code.

(a) Zeige, daß die folgenden Aussagen äquivalent sind:

(i) C ist ein MDS-Code.
(ii) Je k Spalten einer erzeugenden Matrix G von C sind linear unabhängig.

(b) Sei $q = 2$, $k \geq 2$ und C MDS-Code. Zeige, daß $d \leq 2$, und daß $d = 2$ bei geeigneter Wahl von C erreicht wird.

Anleitung: Man darf annehmen, daß die ersten k Spalten von G die Einheitsmatrix bilden. Dann läuft die Behauptung darauf hinaus, daß man überhaupt nur noch eine Spalte anfügen kann, ohne die Bedingung in (ii) zu verletzen, nämlich welche?

(c) Zeige allgemeiner $d \leq q$, falls C ein MDS-Code mit $k \geq 2$ ist.

8.16. Sei C der $[n, n-k, 3]_q$-Hamming-Code aus Beispiel 8.5 mit $n = (q^k - 1)/(q - 1)$. Wir betrachten den $[n,k]$-*Simplex-Code* S der von den Spalten der Kontrollmatrix von H erzeugt wird. Zeige, daß jedes von 0 verschiedene Codewort in S das Gewicht q^{k-1} hat. Insbesondere ist S ein $[n,k,q^{k-1}]$-Code.

Anleitung: Seien $g_1, \ldots, g_k$ die Zeilen von H, $a_1, \ldots, a_k \in \mathbb{F}$ und $v = \sum_{i=1}^{k} a_i g_i \in S$. Wir haben zu zählen, wieviele Komponenten $v_j = 0$ sind, wenn $v \neq 0$.

Wir betrachten die Spalte des Index j von H. Dann ist $v_j = 0$ genau dann, wenn diese Spalte zu

$$U = \{b \in \mathbb{F}^k : \sum_{i=1}^{k} b_i a_i = 0\}$$

gehört. Welche Dimension hat U?

8.17. Schreibe ein `Aribas`-Programm, das für den erweiterten $[8,4,4]$-Hamming-Code alle 1-fachen Fehler korrigiert und alle doppelten Fehler erkennt. Auszugehen ist von der Kontrollmatrix des $[7,4,3]$-Hamming-Codes, die in Beispiel 8.5 angegeben ist. Das Paritätsbit wird „rechts“ angehängt.

Die Datei `empfangen.txt` enthält in der ersten Zeile die Anzahl der noch folgenden Zeilen, die jeweils ein empfangenes Wort enthalten. Dabei sind die einzelnen Bits durch ein Leerzeichen getrennt.

Lösung in einer E-Mail an den Korrekteur. Jede Zeile soll dabei das decodierte Codewort enthalten (Bits durch Leerzeichen getrennt) oder ein `A` (für „Ausfall“, falls, das empfangene Wort nicht decodierbar ist.

8.18. (a) Sei C ein $[n,k,d]_2$- Code und v ein Wort des Gewichts d in C. Wir dürfen annehmen, daß $v_1 = \cdots = v_d = 1$, und betrachten den Code $C' \subset \mathbb{F}_2^{n-d}$, der durch Projektion auf die letzten $n-d$ Komponenten aus C entsteht. Zeige, daß jedes von

v und 0 verschiedene Wort in C an mindestens $\lceil d/2 \rceil$ unter den Stellen $n-d+1,\ldots,n$ die Komponente 1 hat.

Welche Parameter hat C'?

(b) Beweise unter Ausnutzung des Resultats in (a) die *Griesmer-Schranke* (für $q=2$)

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

durch Induktion über k.

(c) Zeige, daß (b) für $[n,k,d]_q$-Codes gilt, wenn man 2 durch q ersetzt.

8.19. Seien C_i, $i=1,2$, Codes mit den Parametern $[n_i,k_i,d_i]_q$. Wir definieren das *Tensorprodukt* $C_1 \otimes C_2$ von C_1 und C_2 als Menge der $n_1 \times n_2$-Matrizen, deren Zeilen zu C_2 und deren Spalten zu C_1 gehören. Zeige: $C_1 \otimes C_2$ ist ein $[n_1 n_2, k_1 k_2, d_1 d_2]$-Code.

Hinweise: (1) Unter der Annahme, daß $1,\ldots,k_i$ Informationsstellen von C_i sind, $i=1,2$, kann man jede $k_1 \times k_2$-Matrix zu einem Element von $C_1 \otimes C_2$ auffüllen.

(2) $d_{\min}(C_1 \otimes C_2) \geq d_1 d_2$ ist leicht zu sehen. Für die Umkehrung betrachte man zu Elementen $v \in C_1$, $w \in C_2$ die Matrix $(v_i w_j)_{i,j}$.

Diese Konstruktion verallgemeinert Aufgabe 6.8.

ABSCHNITT 9

Zyklische Codes

Wir betrachten in diesem Paragraphen Codes, die nicht nur Untervektorräume sind, sondern sogar von der zyklischen Verschiebung in sich selbst überführt werden.

Definition. Die Abbildung $Z : \mathbb{F}^n \to \mathbb{F}^n$

$$Z(x_{n-1},\ldots,x_0) := (x_{n-2},\ldots,x_1,x_0,x_{n-1})$$

heißt *zyklische Verschiebung*. (Es wird später klar, weshalb wir Indexierung der Komponenten in dieser Weise gewählt haben.)

Ein Code $C \subset \mathbb{F}^n$ heißt *zyklisch*, wenn $Z(C) \subset C$ ist.

Offensichtlich gilt

Satz 9.1. *Die zyklische Verschiebung Z ist ein* $\mathbb{F}$*-Vektorraum-Automorphismus von* $\mathbb{F}^n$.

Beispiel 9.2. Wir betrachten den [7,3]-Code C mit der erzeugenden Matrix

$$G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}.$$

Dieser Code ist zyklisch. Da Z nach 9.1 eine lineare Abbildung ist, genügt es für diese Behauptung zu zeigen, daß $Z(w_i) \in C,\ i = 1,2,3$. In der Tat ist

$$Z(w_3) = w_2 \in C, \quad Z(w_2) = w_1, \quad Z(w_1) = w_1 + w_3 \in C.$$

Wir wollen nun die zyklischen Codes als ausgezeichnete Objekte in einer bestimmten algebraischen Struktur erfassen. Dazu betrachten wir den Polynomring $\mathbb{F}[X]$. Wir setzen folgende Begriffe als bekannt voraus: Ring, Polynomring, Teilbarkeit bei Polynomen, Division von Polynomen mit Rest, Grad eines Polynoms, normiertes Polynom.

Sei R ein Ring. Wir nehmen immer an, daß die Multiplikation kommutativ ist und ein Einselement $1 \in R$ existiert. Eine Teilmenge $I \subset R$ ist ein Ideal, wenn $I \neq \emptyset$, $a+b \in I$ für alle $a,b \in I$ und $ra \in I$ für alle $a \in I$ und $r \in R$. Jedes Ideal ist speziell eine Untergruppe von R bezüglich der Addition. Die Menge $R/I = \{r+I : r \in R\}$ der Restklassen modulo I macht man zu einem Ring, indem man

$$(r+I)+(s+I) = (r+s)+I, \quad (r+I)(s+I) = rs+I$$

setzt. Dazu ist nur zu zeigen, daß diese Operationen repräsentantenunabhängig definiert sind, und dies folgt leicht aus den Eigenschaften eines Ideals. Die Restklassenabbildung $\pi : R \to R/I$, $\pi(r) = r + I$, ist ein Ringhomomorphismus. Wir lassen R auf R/I operieren, indem wir

$$r\pi(s) = \pi(s)r = \pi(r)\pi(s)$$

setzen. Allgemein wollen wir vereinbaren, daß ein Produkt $a_1 \cdots a_n$, bei dem die Faktoren aus R oder R/I stammen, in R/I zu bilden ist, wenn einer der Faktoren zu R/I gehört.

Jedes Ideal in $\mathbb{F}[X]$ ist von der Form $(f) = f\mathbb{F}[X]$. Dies sieht man sofort, wenn man f als ein normiertes Polynom kleinsten Grades in I wählt. Ein Ideal ist speziell ein $\mathbb{F}$-Untervektorraum von $\mathbb{F}[X]$. Die Restklassen modulo I werden durch die Divisionsreste modulo f eindeutig repräsentiert. Dies sind gerade die Polynome r mit $\operatorname{grad} r < \operatorname{grad} f$ und die Zuordung $r \to r + I$ bildet den von ihnen gebildeten Untervektorraum von $\mathbb{F}[X]$ $\mathbb{F}$-linear und bijektiv auf $\mathbb{F}[X]/I$ ab. Dies nutzen wir gleich aus.

Wir betten nun $\mathbb{F}^n$ in den Polynomring $\mathbb{F}[X]$ mit Hilfe der Abbildung ε ein, indem wir

$$\varepsilon(a_{n-1}, \ldots, a_0) := a_{n-1}X^{n-1} + \cdots + a_1X + a_0$$

setzen. Offensichtlich ist ε ein injektiver K-Vektorraum-Homomorphismus, und $\varepsilon(\mathbb{F}^n)$ ist der von den Polynomen höchsten $(n-1)$-ten Grades gebildete Untervektorraum.

Sei $f \in \mathbb{F}[X]$ ein Polynom mit $\operatorname{grad} f \leq n-1$,

$$f = a_{n-1}X^{n-1} + \cdots + a_1X + a_0.$$

Dann gilt

$$\begin{aligned} Xf &= a_{n-1}X^n + a_{n-2}X^{n-1} + \cdots + a_1X^2 + a_0X \\ &= a_{n-1}(X^n - 1) + a_{n-2}X^{n-1} + \cdots + a_1X^2 + a_0X + a_{n-1}. \end{aligned}$$

Dies zeigt für $w = (a_{n-1}, \ldots, a_0)$:

$$\varepsilon(Z(w)) \text{ ist der Rest von } X\varepsilon(w) \text{ bei Division durch } X^n - 1. \tag{6}$$

Es ist zweckmäßig, diese Aussage mit Hilfe des Restklassenrings $\mathbb{F}[X]/(X^n - 1)$ zu interpretieren.

Satz 9.3. *Sei $R_n = \mathbb{F}[X]/(X^n - 1)$. Die Einbettung $\varepsilon : \mathbb{F}^n \to \mathbb{F}[X]$ und die Restklassenabbildung $\pi : \mathbb{F}[X] \to R_n$ ergeben einen $\mathbb{F}$-linearen Isomorphismus $\bar{\varepsilon} = \pi \circ \varepsilon : \mathbb{F}^n \to R_n$. Für die Restklasse x von X und alle $w \in \mathbb{F}^n$ ist:*

$$\bar{\varepsilon}(Z(W)) = x(\bar{\varepsilon}(w)).$$

Beweis. Beide Abbildungen ε und π sind $\mathbb{F}$-linear, und damit ist es auch $\overline{\varepsilon}$. Die Restklassen entsprechen bijektiv den Divisionsresten bei Division durch $X^n - 1$, und dies sind gerade die Polynome des Grades höchstens $n-1$. Damit ist klar, daß $\overline{\varepsilon}$ bijektiv ist. Die behauptete Gleichung ist nichts anderes als die Aussage (6). □

Satz 9.3 erlaubt es uns, $\mathbb{F}^n$ mit dem Restklassenring zu identifizieren, und die zyklische Verschiebung als Multiplikation mit (der Restklasse von) X anzusehen. Es ist daher zweckmäßig, $\mathbb{F}^n$ nun durch R_n zu ersetzen und die zu betrachtenden Codes als Teilmengen von R_n zu betrachten.

Satz 9.4. *Sei $C \subset R_n$ ein linearer Code. Dann sind äquivalent:*

(a) *C ist ein zyklischer Code.*
(b) *C ist ein Ideal in R_n.*

Beweis. Sei zunächst C ein Ideal. Dann ist C speziell ein Untervektorraum, also ein linearer Code. Ferner gilt $xC \subset C$, und somit ist C zyklisch.

Zur Umkehrung: Als linearer Code ist C ein Untervektorraum. Es genügt also zu zeigen, daß

$$(a_{n-1}x^{n-1} + \cdots + a_1 x + a_0)v \in C$$

für alle $v \in C$ und $a_{n-1}, \ldots, a_0 \in \mathbb{F}$. Da C eine Untergruppe bezüglich der Addition ist, genügt es hierfür, daß $a_i x^i v \in C$ für alle $a_i \in \mathbb{F}$ und $i \in \mathbb{N}$. Da C sogar ein Untervektorraum ist, reduziert sich dies auf $x^i v \in C$, was wiederum aus $xC \subset C$ durch Induktion über i folgt. □

Man kann sich guten Grundes fragen, weshalb man gerade die zyklischen Codes betrachtet und damit die Restklassenbildung nach $X^n - 1$ (und nicht nach anderen Polynomen). Der praktische Grund dafür ist, daß sich die Multiplikation modulo $X^n - 1$ sehr einfach mit Hilfe linearer Schieberegister instrumentieren läßt. Einfacher ginge es wohl nur noch für X^n selbst, aber die Linksverschiebung vernichtet in n Schritten jedes Element von $\mathbb{F}^n$, und ist daher unbrauchbar. Mathematisch hat $X^n - 1$ den Vorteil, daß es als Faktoren die sehr gut verstandenen Kreisteilungspolynome hat.

Generator- und Kontrollpolynom. Im Beispiel 9.2 wurde C als Untervektorraum erzeugt durch w_3, xw_3, x^2w_3. Wir wollen nun zeigen, daß dies keine spezielle Eigenschaft des Beispiels, sondern stets möglich ist. Wir setzen im folgenden stets $C \neq \{0\}$ voraus.

Satz 9.5. *Sei $C \subset R_n$ ein zyklischer Code. Dann existiert ein eindeutiges Polynom $g \in \mathbb{F}[X]$ mit folgenden Eigenschaften:*

(a) *$C = gR_n$,*
(b) *g ist normiert,*
(c) *$g \mid X^n - 1$.*

Die Elemente $x^{k-1}g,\ldots,xg,g$, $k = n - \operatorname{grad} g$, bilden eine Basis von C. Speziell ist C ein $[n,k]$-Code.

Beweis. Wir betrachten das Urbild $I = \pi^{-1}(C)$ im Polynomring $\mathbb{F}[X]$. Dann gibt es genau ein normiertes Polynom g mit $I = g\mathbb{F}[X]$. Durch Anwenden von π folgt $C = \pi(I) = \pi(g\mathbb{F}[X]) = \pi(g)R_n = gR_n$. Der Kern $(X^n - 1)$ von π ist in I enthalten, und speziell gilt $X^n - 1 \in I$. Also ist $X^n - 1$ Vielfaches von g.

Damit haben wir ein Polynom g gefunden, das die drei geforderten Eigenschaften hat. Ein solches Polynom ist aber eindeutig bestimmt, denn es gilt $(X^n - 1) \subset g\mathbb{F}[X]$ wegen (c), und (a) impliziert dann $I = g\mathbb{F}[X]$. Unter den erzeugenden Elementen von I gibt es genau ein normiertes Polynom.

Die Elemente $x^{k-1}g,\ldots,xg,g$ sind sicherlich linear unabhängig, wie man sofort durch Hinschreiben der Vektoren in R_n sieht. Ferner ist auch klar, daß man durch Subtraktion einer geeigneten Linearkombination aus jedem Codewort $a_{n-1}x^{n-1} + \cdots + a_0$ durch Abziehen einer geeigneten Linearkombination von$x^{k-1}g,\ldots,xg,g$ ein Codewort $r' = a'_u x^u + \cdots + a'_0$ mit $u < k-1$ erhält: $r = a'_u X^u + \cdots + a'_0$ ist gerade der Rest von $a_{n-1}X^{n-1} + \cdots + a_0$ bei Division durch g. Da g das Urbild von C in $\mathbb{F}[X]$ erzeugt, muß $r = 0$ und damit auch $r' = 0$ sein. Also erzeugen $x^{k-1}g,\ldots,xg,g$ den Code C. □

Definition. Das gemäß Satz 9.5 bestimmte Polynom heißt *Generatorpolynom* oder *erzeugendes Polynom* von C.

Als erzeugende Matrix von C ergibt sich mit $m = \operatorname{grad} g$ gemäß 9.5 die $k \times n$-Matrix

$$G = \begin{pmatrix} 0 & \cdots & \cdots & 0 & 1 & g_{m-1} & \cdots & \cdots & g_1 & g_0 \\ 0 & \cdots & 0 & 1 & g_{m-1} & \cdots & \cdots & g_1 & g_0 & 0 \\ \vdots & \iddots & \iddots & \iddots & & & \iddots & \iddots & \iddots & \vdots \\ 0 & \iddots & \iddots & & & \iddots & \iddots & \iddots & & \vdots \\ 1 & g_{m-1} & \cdots & \cdots & g_1 & g_0 & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

Wir können R_k als Untervektorraum von R_n auffassen, indem wir einfach die Elemente $a_{k-1}x^{k-1} + \cdots + a_0$ in R_k und R_n identifizieren. Dann gilt für die lineare Codierung $\varphi : R_k \to R_n$, mittels G:

$$\varphi(u) = uG = ug.$$

Diese Codierung ist im allgemeinen nicht systematisch. Eine systematische Codierung erhält man auf folgende Weise. Für $m \le j \le n-1$ ergebe die Division mit Rest in $\mathbb{F}[X]$:

$$X^j = q^{(j)} \cdot g + r^{(j)}.$$

Wir setzen $g^{(j)} := q^{(j)}g$. Dann ist $\operatorname{grad} g^{(j)} = j$ und die Bilder von $g^{(m)},\ldots,g^{(n-1)}$ in R_n sind linear unabhängig. Die mit ihnen gebildete erzeugende Matrix G hat die

gewünschte Form

$$G = \left(\begin{array}{cccc|ccc} 1 & 0 & \cdots & 0 & -r_{m-1}^{(n-1)} & \cdots & -r_0^{(n-1)} \\ 0 & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & -r_{m-1}^{(m)} & \cdots & -r_0^{(m)} \end{array}\right)$$

Die systematische Codierung kann aber auch ohne Bestimmung dieser Matrix erfolgen: $u \in \mathbb{F}^k$ wird codiert zu qg mit

$$x^m u - r = qg, \qquad \operatorname{grad} r < \operatorname{grad} g.$$

Man überprüft leicht, daß dies genau die Abbildung $u \to uG$ ist.

Zyklische Codes werden auch durch ein Polynom kontrolliert:

Definition. Sei $g \in \mathbb{F}[X]$ Generatorpolynom des zyklischen Codes $C \subset \mathbb{F}^n$. Es gelte $X^n - 1 = g \cdot n$. Dann heißt *h Kontrollpolynom* von C.

Diese Bezeichnung wird gerechtfertigt durch

Satz 9.6. *Sei $h = X^k + h_{h-1}x^{k-1} + \cdots + h_0$ Kontrollpolynom des zyklischen Codes $C \subset R_n$. Dann gilt*

$$v \in C \iff vh = 0$$

für alle $v \in R_n$. Ferner ist

$$H = \begin{pmatrix} 0 & \cdots & \cdots & 0 & h_0 & h_1 & \cdots & \cdots & h_{k-1} & 1 \\ 0 & \cdots & 0 & h_0 & h_1 & \cdots & \cdots & h_{k-1} & 1 & 0 \\ \vdots & \iddots & \iddots & \iddots & & & \iddots & \iddots & \iddots & \vdots \\ 0 & \iddots & \iddots & & & \iddots & \iddots & \iddots & & \vdots \\ h_0 & h_1 & \cdots & \cdots & h_{k-1} & 1 & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

eine Kontrollmatrix von C.

Beweis. Sei g das Generatorpolynom von C. Zu $v \in C$ existiert ein $w \in R_n$ mit $v = wg$. Dann ist $vh = w(gh) = w(X^n - 1) = 0$.

Ist umgekehrt $vh = 0$, so gilt für ein Polynom $\tilde{v}$ mit $v = \pi(\tilde{v})$, daß $\pi(\tilde{v}h) = vh = 0$. Dies bedeutet $\tilde{v}h \in (X^n - 1)$. Es existiert also ein Polynom f mit

$$\tilde{v}h = f(X^n - 1) = fgh$$

Im Polynomring $\mathbb{F}[X]$ können wir den Faktor h kürzen und erhalten $\tilde{v} = fg$. Anwenden von π liefert $v = \pi(\tilde{v}) = \pi(f)g \in C$.

Wir haben noch zu zeigen, daß H die Kontrollmatrix ist. Da $gh = X^n - 1$, sind $h_k, h_0 \neq 0$. Schon aus $h_0 \neq 0$ folgt $\operatorname{rang} H = m$. Es bleibt zu zeigen, daß $GH^\top = 0$

ist. Die Multiplikation der i-ten Zeile von G mit der j-ten Spalte von $H^\top$ ergibt

$$\sum_{q=0}^{n} G_{iq}H_{jq} = g_{n-i-q+1}h_{q-1-m+j} = \sum_r g_r h_{m+j-i-r}.$$

(Dabei sind die Koeffizienten zu Indizes < 0 und $> \operatorname{grad} g$ bzw. *gradh* gleich 0 zu setzen.) Der Eintrag der Produktmatrix an der Stelle (i,j) ist also der Koeffizient von gh im Grad $m+j-i$. Für $i = 1,\ldots,k$ und $j = 1,\ldots,m$ ergeben sich gerade die Koeffizienten von $X^n - 1$ in den Graden $1,\ldots,n-1$, und diese sind sämtlich 0. □

Beispiel 9.7. Sei $q = 2$, $n = 7$. Es gilt

$$X^7 - 1 = (X+1)(X^3+X+1)(X^3+X^2+1),$$

Der $[7,3]$-Code in Beispiel 9.2 wird von $g_1 = X^4+X^3+X^2+1 = (X+1)(X^3+X+1)$ erzeugt. Das Kontrollpolynom ist $h_1 = X^3+X^2+1$.

Wir betrachten jetzt den von $g_2 = h_1 = X^3+X^2+1$ erzeugten $[7,4]$-Code C mit Kontrollpolynom $h_2 = g_1$. Zur Ermittlung der erzeugenden Matrix für systematische Codierung bestimmen wir:

$$\begin{aligned}
X^3 &= 1\cdot(X^3+X^2+1)+(X^2\quad+1)\\
X^4 &= (X+1)(X^3+X^2+1)+(X^2+X+1)\\
X^5 &= (X^2+X+1)(X^3+X^2+1)+(\quad X+1)\\
X^6 &= (X^3+X^2+X\quad)(X^3+X^2+1)+(X^2+X\quad)
\end{aligned}$$

Damit ist

$$G = \left(\begin{array}{cccc|ccc} 1&0&0&0&1&1&0\\ 0&1&0&0&0&1&1\\ 0&0&1&0&1&1&1\\ 0&0&0&1&1&0&1 \end{array}\right)$$

erzeugende Matrix in systematischer Form. Als Kontrollmatrix von C erhalten wir aus G:

$$H = \left(\begin{array}{cccc|ccc} 1&0&1&1&1&0&0\\ 1&1&1&0&0&1&0\\ 0&1&1&1&0&0&1 \end{array}\right)$$

Man sieht, daß H die Kontrollmatrix eines $[7,4]$-Hamming-Codes ist. Damit erweist sich dieser Code als zyklisch. Wir werden später noch zeigen, daß alle binären (und viele andere) Hamming-Codes zyklisch sind.

Zyklische Codes sind auch deshalb interessant, weil mit ihnen lange Fehlerbündel erkannt werden können. Ein zyklischer Code mit Generatorpolynom g erkennt alle Fehlerbündel der Länge $< \operatorname{grad} g$.

Zyklische Codes werden zur Fehlererkennung in Computer-Netzwerken verwendet. In diesen ist bei Erkennung eines Fehlers eine Rückfrage beim Sender

möglich, der dann das fehlerhafte Datenpaket noch einmal sendet – das macht Sinn, denn Fehler kommen relativ selten vor. Man benutzt dafür sogenannte binäre CRC-Codes. Das sind zyklische Codes, deren Generatorpolynom von der Form $(X+1)\mu$ ist, wobei μ Minimalpolynom eines erzeugenden Elementes der Einheitengruppe eines Erweiterungskörpers von $\mathbb{F}_2$ ist. Wir kommen auf die hier verwendeten Begriffe im nächsten Abschnitt zurück.

Übungen

9.8. Zerlege die Polynome X^n-1 für $n=1,\ldots,8$ über $\mathbb{F}_2$ und $\mathbb{F}_3$ in irreduzible Faktoren.

9.9. Zeige, daß ein zyklischer $[n,k]$-Code Fehlerbündel einer Länge $\leq n-k$ erkennen kann.

9.10. Sei C ein zyklischer Code mit Generatorpolynom g, und C' entstehe aus C durch Verschachtelung zur Tiefe t. Zeige, daß auch C' zyklisch ist und das Generatorpolynom $g(X^t)$ hat.

9.11. Der chinesische Restsatz gilt im Polynomring $K[X]$ über einem Körper K ebenso wie in $\mathbb{Z}$: Ist $f=p_1^{e_1}\cdots p_m^{e_m}$ die Zerlegung von f mit paarweise teilerfremden irreduziblen Polynomen $p_1,\ldots,p_m$, so gilt

$$R=K[X]/(f)\cong K[X]/(p_1^{e_1})\times\cdots\times K[X]/(p_m^{e_m}).$$

Wenn f nur einfache Nullstellen in seinem Zerfällungskörper besitzt, sind die Exponenten $e_i=1$, und $K[X]/(f)$ ist das direkte Produkt der Körper $K_i=K[X]/(p_i)$. Für $i=1,\ldots,m$ sei

$$e_i=(0,\ldots 0,1,0,\ldots,0)\quad\text{mit}\quad 1\in K_i$$

gebildet. Jedes Ideal in R ist von der Form $K_{i_1}\times\cdots\times K_{i_k}\subset R$ mit $i_1<\cdots<i_k$, und wird von $e=e_{i_1}+\cdots+e_{i_m}$ erzeugt. Das Element e ist *idempotent*, d. h. es gilt $ee=e$.

Wir betrachten nun speziell $R_n=K[X]/(X^n-1)$.

(a) Zeige, daß X^n-1 nur einfache Nullstellen in seinem Zerfällungskörper von $\mathbb{F}_q$ hat, wenn q und n teilerfremd sind, und daß jeder zyklische Code der Länge n von einem eindeutig bestimmten idempotenten Element erzeugt wird. (Im allgemeinen ist dies *nicht* das Generatorpolynom!)

(b) Beweise die Aussage in (a) direkt ohne Benutzung des chinesischen Restsatzes: Wenn q teilerfremd zu n ist und C das Generatorpolynom g und Kontrollpolynom h hat, so ist

$$e=\frac{1}{n}\,gh'\mod X^n-1$$

ein idempotentes Element, das C als Ideal erzeugt. Hinweis: Differenziere $X^n - 1 = gh$.

ABSCHNITT 10

Beispiele zyklischer Codes

Die sehr wichtige Klasse von Codes sind die *BCH-Codes*, benannt nach ihren Entdeckern Bose, Ray-Chaudury (1960) und Hackendem (1959). Einerseits besitzen diese Codes relativ hohe Minimalabstände, zum anderen kann man für sie effiziente Decodierungsverfahren angeben. In diesem Abschnitt brauchen wir mehr Kenntnisse über endliche Körper als bisher. Deshalb stellen wir einige wichtige Aussagen zusammen.

Sei K ein Körper, $L \supset K$ ein Erweiterungskörper und x in L ein Element, zu dem es ein Polynom $f \in K[X]$, $f \neq 0$, mit $f(x) = 0$ gibt. Man nennt x dann *algebraisch* über K. Unter allen Polynomen $f \neq 0$ mit $f(x) = 0$ gibt es ein eindeutig bestimmtes normiertes Polynom μ kleinsten Grades, das *Minimalpolynom von x*, und jedes Polynom g mit $g(x) = 0$ ist ein Vielfaches von μ. Mit anderen Worten, μ erzeugt den Kern des Einsetzungshomomorphismus

$$K[X] \to L, \quad g \mapsto g(x).$$

Das Minimalpolynom ist irreduzibel, denn aus $g(x)h(x) = 0$ folgt $g(x) = 0$ oder $h(x) = 0$. Es ist dann leicht zu sehen, daß $K(x) = \{g(x) : g \in K[X]\}$ ein Körper ist, und er ist offensichtlich der kleinste Teilkörper von L, der x enthält. Die Dimension von $K(x)$ als K-Vektorraum ist der Grad von μ. Dies sieht man wie bei 9.3.

Ist umgekehrt μ ein irreduzibles Polynom in $K[X]$, so findet man stets einen Erweiterungskörper $L \supset K$ und ein Element $x \in L$ mit $\mu(x) = 0$. Man nimmt für L einfach den Restklassenring $K[X]/(\mu)$ und wählt x als die Restklasse von x. Durch sukzessive Konstruktion von Erweiterungskörpern kann man zu jedem Polynom $f \in K[X]$ einen Körper L finden, in dem f vollständig in Linearfaktoren zerfällt. In L gibt es einen kleinsten Körper $L_0 \supset K$, der alle Nullstellen von f enthält. Dieser *Zerfällungskörper* von f ist bis auf Isomorphie über K eindeutig bestimmt.

Zu jeder Primzahl p existiert ein Körper mit p Elementen, nämlich $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Aber auch zu jeder Primzahlpotenz p^n gibt es einen Körper mit dieser Elementezahl.

Satz 10.1. *Sei p eine Primzahl.*

(a) *Dann gibt es zu jedem Exponenten $n \geq 1$ einen Körper $\mathbb{F}_q$ mit $q = p^n$ Elementen. Er ist der Zerfällungskörper von $X^{q-1} - 1$ über $\mathbb{F}_p$ und daher bis auf Isomorphie eindeutig bestimmt.*

(b) *Die Gruppe* $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ *(bezüglich der Multiplikation) ist zyklisch, das heißt es existiert ein* $\alpha \in \mathbb{F}_q$ *mit*

$$\mathbb{F}_q^* = \{\alpha^i : i \in \mathbb{Z}\}.$$

(c) *Speziell ist* $\mathbb{F}_q = \mathbb{F}_p(\alpha)$ *und das Minimalpolynom von* α *über* $\mathbb{F}_p$ *hat den Grad n.*

(d) *Genau dann ist* $\mathbb{F}_{p^n}$ *in* $\mathbb{F}_{p^m}$ *enthalten, wenn* $n \mid m$*. Die endlichen Erweiterungskörper von* $\mathbb{F}_q$ *sind also die Körper* $\mathbb{F}_{q^r}$.

Teil (b) folgt einfach daraus, daß die endlichen Untergruppen von K^* für jeden Körper K zyklisch sind. Teil (c) sieht man so: Wenn $\mathbb{F}_{p^n}$ in $\mathbb{F}_{p^m}$ enthalten ist, muß die Elementezahl von p^m eine Potenz von p^n sein, denn $\mathbb{F}_{p^m}$ ist dann ein $\mathbb{F}_{p^n}$-Vektorraum. Ist umgekehrt n ein Teiler von m, so ist $X^{p^n-1}-1$ ein Teiler von $X^{p^m-1}-1$, und der Zerfällungskörper von $X^{p^m-1}-1$ enthält alle Nullstellen von $X^{p^n-1}-1$.

Man nennt ein Element α wie in Teil (b) des Satzes ein *primitives Element* von $\mathbb{F}_q$ und sein Minimalpolynom ein *primitives Polynom* über $\mathbb{F}_p$. Beispiele primitiver Polynome über $\mathbb{F}_2$ der Grade $n = 1,\ldots,7$ sind

$$1+X,\ 1+X+X^2,\ 1+X+X^3,\ 1+X+X^4,\ 1+X^2+X^5,\ 1+X+X^6,\ 1+X^3+X^7.$$

Da $\mathbb{F}_q$ zyklisch ist, gibt es zu jedem Teiler d von $q-1$ ein Element der Ordnung d in der Gruppe $\mathbb{F}_q^*$.

Wir benutzen nun Erweiterungskörper K von $\mathbb{F}_q$, um zyklische Codes über $\mathbb{F}_q$ zu definieren. Sei $\alpha \in K$ ein Element mit $\alpha^n = 1$ und $f \in R_n = \mathbb{F}_q[X]/(X^n-1)$. Wir können dann jedem $g \in R_n$ den Wert $g(\alpha)$ zuordnen. Dazu wählen wir uns einen Repräsentanten $\tilde{g}$ in $\mathbb{F}[X]$ und setzen

$$g(\alpha) = \tilde{g}(\alpha).$$

Diese Definition ist repräsentantenunabhängig, denn je zwei Repräsentanten von g unterscheiden sich um ein Vielfaches von X^n-1, und $\alpha^n - 1 = 0$.

Satz 10.2. *Sei K Erweiterungskörper von* $\mathbb{F} = \mathbb{F}_q$ *und* $\alpha \in K$ *ein Element mit* $\alpha^n = 1$*. Dann ist*

$$C(\alpha) = \{f \in R_n : f(\alpha) = 0\}$$

ein zyklischer Code über $\mathbb{F}$*. Sein Generatorpolynom ist das Minimalpolynom* μ *von* α.

Beweis. Zunächst ist zu beachten, daß X^n-1 von μ geteilt wird. Daher definiert μ als Generatorpolynom einen zyklischen Code C' der Länge n über $\mathbb{F}$. Offensichtlich ist $C' \subset C$, denn mit μ verschwinden auch alle Vielfachen von μ auf α.

Ist umgekehrt $f(\alpha) = 0$, so gilt $\tilde{f}(\alpha) = 0$ für jeden Repräsentanten $\tilde{f}$ von f. Mithin ist $\tilde{f}$ Vielfaches von μ und damit $f \in C'$. □

Satz 10.3. *Seien $C_1,\dots,C_m$ zyklische Codes in R_n und $g_1,\dots,g_m$ ihre Generatorpolynome. Dann ist das kleinste gemeinsame Vielfache von $g_1,\dots,g_n$ das Generatorpolynom des zyklischen Codes $C_1\cap\dots\cap C_m$.*

Beweis. Der Durchschnitt von Idealen eines Ring ist ein Ideal. Somit ist $C_1\cap\dots\cap C_m$ ein zyklischer Code. Das Element f gehört zu C_i, wenn ein (und damit jeder) Repräsentant $\tilde{f}$ von g_i geteilt wird. Somit gehört f zum Durchschnitt der Codes, wenn es vom kleinsten gemeinsamen Vielfachen der g_i geteilt wird. □

Damit sind unsere Vorbereitungen zur Definition der BCH-Codes abgeschlossen.

Definition. Sei $n\in\mathbb{N}$, $n\geq 1$ und β Element eines Erweiterungskörpers K von $\mathbb{F}_q$, das in K^* die Ordnung n hat. Ferner sei $\delta\in\mathbb{N}$, $1\leq\delta\leq n$. Dann heißt

$$C=\mathrm{BCH}_q(\beta,\delta)=\bigcap_{i=1}^{\delta-1}C(\beta^i)\subset R_n$$

ein *BCH-Code* (im engeren Sinn) über $\mathbb{F}_q$. Ist $n=q^m-1$ für ein geeignetes m, und damit β primitives Element von $\mathbb{F}_q(\beta)\cong\mathbb{F}_{q^n}$, so heißt C *primitiver BCH-Code*. Man nennt δ die *Entwurfsdistanz* von C.

Man beachte, daß der Fall $K=\mathbb{F}_q$ keineswegs ausgeschlossen ist und weiter unten sogar explizit vorkommen wird.

Den Begriff „Entwurfsdistanz“ rechtfertigt der folgende Satz:

Satz 10.4. *Mit den Bezeichnungen der Definition gilt:*

(a) *C ist zyklisch, sein Generatorpolynom ist das kleinste gemeinsame Vielfache der Minimalpolynome der Elemente β^i, $i=1,\dots\delta-1$.*
(b) *C hat Minimalabstand $d\geq\delta$.*

Beweis. Wir betrachten die folgende Matrix über K:

$$H=\begin{pmatrix}\beta^{n-1} & \dots & \beta & 1\\ \beta^{2(n-1} & \dots & \beta^2 & 1\\ \vdots & & \vdots & \vdots\\ \beta^{(\delta-1)(n-1)} & \dots & \beta^{\delta-1} & 1\end{pmatrix}$$

Nach Definition von C gilt $v\in C$ genau dann, wenn $vH^\top=0$ ist. Es genügt daher zu zeigen, daß je $\delta-1$ Spalten der Matrix H linear unabhängig sind. Dies gilt sogar über K und damit erst recht über $\mathbb{F}_q$. Wir betrachten dazu die Spalten, in deren

erster Zeile die Elemente $\beta^{j_1}, \ldots, \beta^{j_{\delta-1}}$ stehen:

$$\det \begin{pmatrix} \beta^{j_1} & \cdots & \beta^{j_{\delta-1}} \\ (\beta^{j_1})^2 & \cdots & (\beta^{j_{\delta-1}})^2 \\ \vdots & & \vdots \\ (\beta^{j_1})^{\delta-1} & \cdots & (\beta^{j_{\delta-1}})^{\delta-1} \end{pmatrix}$$

$$= \beta^{j1} \cdots \beta^{j_{\delta-1}} \det \begin{pmatrix} 1 & \cdots & 1 \\ (\beta^{j_1}) & \cdots & (\beta^{j_{\delta-1}}) \\ \vdots & & \vdots \\ (\beta^{j_1})^{\delta-2} & \cdots & (\beta^{j_{\delta-1}})^{\delta-2} \end{pmatrix} \neq 0,$$

denn die β^{j_i} sind paarweise verschieden, und die Vandermonde-Matrix hat Determinante $\neq 0$. □

Diese Beweis zeigt, daß der von β über $\mathbb{F}_q(\beta)$ definierte BCH-Code wirklich Abstand d hat. Er hat außerdem die Dimension $n-d+1$ und ist daher ein MDS-Code. Die primitiven unter diesen Codes haben einen speziellen Namen:

Definition. Sei β primitives Element von $\mathbb{F}_q$. Dann heißt der von β definierte BCH-Code der Länge $q-1$ ein *Reed-Solomon-Code* (im engeren Sinn).

Die Reed-Solomon-Codes sind also MDS-Codes. Die Bestimmung der Minimaldistanz d allgemeiner BCH-Codes ist ein schwieriges Problem, das wir hier nicht verfolgen wollen. Man findet dazu Ergebnisse in [**vL**] und [**Wi**]. Zum Beispiel ist d ungerade und $d \leq 2\delta - 1$ für primitive binäre BCH-Codes. Da die effizienten Decodierungsalgorithmen für BCH-Codes aber in der Regel nur höchstens $(\delta-1)/2$ Fehler korrigieren können, ist δ eine wichtigere Größe als d.

Beispiel 10.5. Viele Hamming-Codes sind (äquivalent zu) BCH-Codes und damit zyklisch. Die Hamming-Codes $C \subset \mathbb{F}^n$ haben die Parameter

$$n = \frac{q^k-1}{q-1}, \quad n-k, \quad d = 3.$$

Wir betrachten den Fall, in dem $\mathrm{ggT}(k, q-1) = 1$ ist, und wählen β in $\mathbb{F}_{q^k}^*$ als ein Element der Ordnung n. (Dies ist möglich, weil n ein Teiler der Ordnung q^k-1 von $\mathbb{F}_{q^k}^*$ ist, und diese Gruppe zyklisch ist.) Sei $C' = \mathrm{BCH}(\beta, 2)$. Die Matrix H im Beweis von Satz 10.4 ist dann einfach

$$H = \begin{pmatrix} \beta^{n-1} & \cdots & \beta & 1 \end{pmatrix}.$$

Nach Wahl einer $\mathbb{F}$-Basis von $\mathbb{F}_{q^k}$ können wir diese Matrix als k-zeilige Matrix über $\mathbb{F}$ betrachten. Nach Wahl von n hat sie hat genau so viele Spalten wie $\mathbb{F}^k$

1-dimensionale Untervektorräume hat. Es genügt also zu zeigen, daß je zwei Spalten von H linear unabhängig über $\mathbb{F}$ sind. Dann ist nämlich jeder 1-dimensionale Untervektorraum von $\mathbb{F}^k$ genau einmal vertreten.

Der entscheidende Punkt ist nun, daß nicht nur $\mathrm{ggT}(k, q-1) = 1$, sondern sogar $\mathrm{ggT}(n, q-1) = 1$. Es gilt nämlich

$$n = q^{k-1} + \cdots + q + 1 = (q-1)(q^{k-2} + 2q^{k-3} + \cdots + (k-1)) + k,$$

so daß $n \equiv k \mod q-1$.

Wir betrachten zwei Spalten β^i und β^j, $0 \le i < j \le n-1$. Dann ist $\gamma = \beta^j/\beta^i = \beta^{j-i} \neq 1$. Die Ordnung von γ ist ein Teiler von n aber $\neq 1$. Daß β^i und β^j über $\mathbb{F}_q$ linear abhängig sind, bedeutet nichts anderes als $\gamma \in \mathbb{F}^*$. Damit müßte die Ordnung von γ aber auch ein Teiler von $q-1$ sein, was wegen $\mathrm{ggT}(n, q-1)$ ausgeschlossen ist.

Obwohl die Entwurfsdistanz nur 2 ist, haben die Hamming-Codes Minimalabstand 3. Man beachte, daß die Voraussetzung $\mathrm{ggT}(k, q-1)$ im Fall $q = 2$ aus trivialen Gründen erfüllt ist.

Binäre BCH-Codes. Wir wollen den Fall $q = 2$ noch etwas näher betrachten, und eine Existenzaussage für BCH-Codes mit guten Eigenschaften machen.

Satz 10.6. *Sei $q = 2$ und δ ungerade, $\delta = 2t+1$. Dann ist*

$$\mathrm{BCH}_2(\beta, \delta) = ((\beta) \cap C(\beta^3) \cap \cdots \cap C(\beta^{2t-1}).$$

Beweis. Es genügt zu zeigen, daß $C(\beta^j) = C(\beta^{2j})$ oder allgemeiner $C(\gamma) = C(\gamma^2)$ ist. Dies folgt unmittelbar aus

$$\sum_{i=0}^{m-1} x_i (\gamma^2)^i = \sum_{i=0}^{m-1} (x_i \gamma^i)^2 = \left(\sum_{i=0}^{m-1} x_i \gamma^i \right)^2.$$

für alle m und $x_i \in \mathbb{F}_2$. □

Satz 10.7. *Zu jedem m und t gibt es einen binären BCH-Code der Länge 2^{m-1}, der alle höchstens t-fachen Fehler korrigiert und höchstens mt Kontrollstellen besitzt.*

Beweis. Wir wählen β als primitives Element von $\mathbb{F}_{2^m}$. Das Generatorpolynom von $\mathrm{BCH}(\beta, 2t+1)$ ist gemäß Satz 10.6 das kleinste gemeinsame Vielfache der Minimalpolynome von $\beta, \beta^3, \ldots, \beta^{2t-1}$. (Die Minimalpolynome sind teilerfremd; siehe Aufgabe 10.9.) Jedes dieser Minimalpolynome besitzt höchstens den Grad m, denn sonst würden die Potenzen von β^i einen mehr als m-dimensionalen Unterraum von $\mathbb{F}_{2^m}$ erzeugen. Daraus ergibt sich unmittelbar die Behauptung. □

Die Bedeutung der BCH-Codes beruht vor allem auf effizienten Decodierungsalgorithmen. Wir wollen dies wenigstens an einem kleinen Beispiel studieren. Für die systematische Behandlung dieses Themas verweisen wir auf [**Wi**]. Das Beispiel

zeigt aber schon das Prinzip: Zur Decodierung hat man algebraische Gleichungssysteme zu lösen.

Beispiel 10.8. Sei $n = 15$, $m = 4$, β primitives Element von $\mathbb{F}_{2^4}$ mit Minimalpolynom $X^4 + X + 1$ über $\mathbb{F}_2$, $t = 2$ und $\delta = 5$. Als Kontrollmatrix des zugehörigen. BCH-Codes können wir gemäß 10.6

$$H = \begin{pmatrix} \beta^{14} & \beta^{13} & \cdots & \beta & 1 \\ (\beta^3)^{14} & (\beta^3)^{13} & \cdots & \beta^3 & 1 \end{pmatrix}$$

wählen. Das Minimalpolynom von β^3 ist

$$p_2 = X^4 + X^3 + X + 1.$$

Die Polynome p_1, p_2 sind irreduzibel, normiert und verschieden. Ihr kleinstes gemeinsames Vielfaches ist

$$g = p_1 p_2 = (X^4 + X + 1)(X^4 + X^3 + X^2 + X + 1).$$

Damit ist C ein $[15, 7]_2$-Code.

Sei $v \in R_{15}$, $v = \sum_{j=0}^{14} v_j x^j$. Als Syndrom ergibt sich

$$vH = (S_1(v), S_3(v))$$

mit

$$S_i(v) = \sum_{j=0}^{14} v_j (\beta^i)^j, \qquad i = 1, 3.$$

Sei $e \in F_{15}$ ein Fehlervektor mit $\mathrm{wt}(e) = 2$, etwa

$$e = x^{a_1} + x^{a_2}, \qquad 0 \le a_1, a_2 \le 14, \quad a_1 \ne a_2.$$

Dann ist

$$\begin{aligned} S_1(e) &= \beta^{a_1} + \beta^{a_2}, \\ S_3(e) &= (\beta^{a_1})^3 + (\beta^{a_2})^3. \end{aligned}$$

Um a_1 und a_2 zu bestimmen, genügt es, die „Fehlerpositionen“ $\beta_1 = \beta^{a_1}$, $\beta_2 = \beta^{a_2}$ zu bestimmen, d.h. wir müssen das Gleichungssystem nach β_1,β_2 auflösen. Sei $S_1 := S_1(e)$, $S_2 := S_2(e)$. Wir formen um:

$$\begin{aligned} \beta_2 &= S_1 + \beta_1, \\ \beta_2^3 &= S_1^3 + S_1^2\beta_1 + S_1\beta_1^2 + \beta_1^3, \\ \beta_1^3 + \beta_2^3 &= S_1^3 + S_1^2\beta_1 + S_1\beta_1^2, \end{aligned}$$

also $S_3 = S_1^3 + S_1^2\beta_1 + S_1\beta_1^2$, mithin

$$1 + \beta_1^{-1} S_1 + \beta_1^{-2}\left(S_1^2 + \frac{S_3}{S_1}\right) = 0.$$

Genauso erhält man

$$1+\beta_2^{-1}S_1+\beta_2^{-1}\left(S_1^2+\frac{S_3}{S_1}\right)=0.$$

Somit sind β_1^{-1} und β_2^{-1} die Nullstellen von

$$\sigma(X)=1+S_1X+\left(S_1^2+\frac{S_3}{S_1}\right)X^2\in\mathbb{F}_{2^4}[X]$$

in $\mathbb{F}_{2^4}$. Liegt genau ein Fehler vor, so ist $S_3=S_1^3$ und

$$\sigma(X)=1+\beta_1^{-1}S_1.$$

Zur Fehlerkorrektur bestimmt man also zunächst S_1 und S_3. Falls $S_1=S_3=0$, gehört das empfangene Wort zum Code. Falls $S_3=S_1^3\neq 0$, liegt genau ein Fehler vor, und man bestimmt β_1 als Nullstelle von σ. Im $S_3\neq S_1^3$, bestimmt man wieder die Nullstellen von σ (etwa durch Ausprobieren).

(a) Falls σ eine (und damit 2) Nullstelle(n) in $\mathbb{F}_{2^4}$ besitzt, liegen genau 2 Fehler vor.
(b) Falls σ irreduzibel ist, liegt ein zwar erkennbarer, aber nicht korrigierbarer Fehler vor.

Wir wollen zum Abschluß des codierungstheoretischen Teils der Vorlesung kurz die Codierung von Musik-CDs diskutieren, auf denen ja die Musik – im Gegensatz zu Schallplatte und klassischem Magnetband – digital gespeichert ist. Fabrikneue CDs enthalten infolge von Fabrikationsfehlern und Materialunreinheiten bereits ca. 500 000 Einzelfehler. Durch Abnutzung beim Gebrauch, Verschmutzung oder gar Kratzer (die das Abhören einer Schallplatte zur Qual machen können) kommen Bündelfehler hinzu, die Hunderte oder Tausende aufeinander folgender Bits löschen oder verfälschen. Davon hören wir beim Abspielen der CD erstaunlicherweise nichts.

Bei der Codierung werden 2 Codes C_1 und C_2 über $\mathbb{F}=\mathbb{F}_{2^8}$ verwendet. Beide werden durch mehrfaches Verkürzen aus einem $[255,251,5]$-RS-Code über $\mathbb{F}$ durch mehrfache Verkürzung gewonnen: der sogenannte innere Code C_1 hat die Parameter $[32,28,5]$, der äußere Code C_2 ist ein $[28,24,5]$-Code.

Die beiden Stereokanäle werden mit einer Frequenz von $44,1$ kHz abgetastet. Mittels eines Analog-Digital-Wandlers wird dabei jedem Kanal ein 16bit-Wort zugeordnet. Diese 16bit Worte repräsentieren zwei Elemente des Körpers $\mathbb{F}$. Pro Abtastung entstehen also 4 Elemente aus $\mathbb{F}$. Man faßt dann 4 Abtastungen zu einem Audiowort der Länge 24 über $\mathbb{F}$ zusammen. Diese Wörter werden mittels C_2 zu Wörtern der Länge 28 codiert. Dann erfolgt zunächst eine Codeverschachtelung zur Tiefe 4, und die so entstehenden Worte der Länge 112 werden einer weiteren

Verschachtelung zur Tiefe 28 unterworfen. Insgesamt entsteht dann eine Matrix

$$V = \begin{pmatrix} v_{11} & v_{21} & v_{31} & v_{41} & v_{12} & \cdots & v_{4,28} \\ v_{51} & v_{61} & v_{71} & v_{81} & v_{52} & \cdots & v_{8,28} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ v_{109,1} & v_{110,1} & v_{111,1} & v_{112,1} & v_{109,2} & \cdots & v_{112,28} \end{pmatrix}.$$

Die Spalten der Länge 28 werden dann mit dem inneren Code C_1 zu Wörtern der Länge 32 über $\mathbb{F}$ codiert. Allerdings kann diese Information nicht direkt auf die CD übertragen werden. Es werden vielmehr noch Display-Information hinzugefügt, die der CD-Spieler beim Abspielen anzeigt, und die Aufzeichnung auf der CD muß Nebenbedingungen erfüllen, die für die Spurführung des Laserstrahls und die Synchronisation zwischen CD und CD-Spieler erforderlich sind. Wir verweisen dazu auf [**Wi**] und die dort angegebene Literatur.

Beim Abspielen der CD korrigiert der Decodierer zunächst Einzelfehler des inneren Codes C_1. Dies ist möglich, weil C_1 den Minimalabstand 5 hat. Werden mindestens 2 Fehler erkannt, wird das ganze Wort als Auslöschung deklariert. Der äußere Code kann wegen seiner Minimaldistanz 5 bis zu 4 Auslöschungen korrigieren. Insgesamt erlaubt dieses Schema Fehlerbündel einer Länge von 4096 Bit. Diese entsprechen 512 Elementen über $\mathbb{F}$ und damit 16 aufeinander folgenden C_1-Worten. Damit sind 16 Spalten der Matrix V betroffen, und von denen wird jedes C_2-Wort maximal an 4 Stellen ausgelöscht.

Neuere CD-Spieler nutzen die Fehlerkorrektur von C_1 voll aus, markieren aber Spalten mit mehr als einem Fehler als „verdächtig“. Man kann über die Fehlerkorrektur hinaus noch Interpolationsverfahren benutzen, um nicht korrigierbare Fehler so gut wie möglich zu überspielen.

Algebraisch-geometrische Codes. BCH-Codes kann man als spezielle *Goppa-Codes* verstehen. Die klassischen Goppa-Codes kann man mit Hilfe der algebraischen Geometrie erheblich erweitern. Dabei erhält man die zur Zeit besten bekannten Codes, zumindest hinsichtlich Minimalabstand und Informationsrate. Solche Codes werden zusammen mit den notwendigen Hilfsmitteln der algebraischen Geometrie in [**Stn**] diskutiert.

Übungen

10.9. Sei q Potenz einer Primzahl p und $r = q^m$.

(a) Zeige, daß $X^{r-1} - 1$ Produkt aller irreduziblen Polynome über $\mathbb{F} = \mathbb{F}_q$ ist, deren Grad m teilt.

(b) Ein irreduzibles Polynom über $\mathbb{F}$ hat in einem Erweiterungskörper von $\mathbb{F}$ nur einfache Nullstellen.

(c) Ist β algebraisch über $\mathbb{F}$ mit Minimalpolynom μ vom Grad m, so sind $\beta, \beta^q, \ldots, \beta^r$ die paarweise verschiedenen Nullstellen von μ.

10.10. (a) Konstruiere einen binären BCH-Code C der Länge 31 und der Entwurfsdistanz 7. Welche Dimension läßt sich erreichen?

(b) Wende auf die Parameter des Codes in (a) alle bekannten Schranken an, um eine Abschätzung für den Minimalabstand zu finden.

ABSCHNITT 11

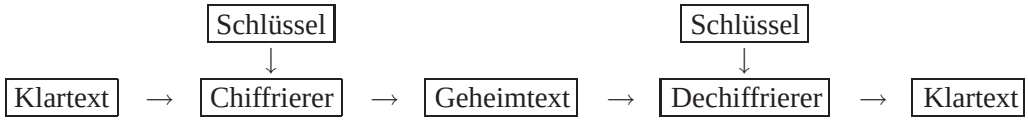
Kryptosysteme

In der Codierungstheorie haben wir Verfahren diskutiert, die Information gegen Übertragungsfehler auf störanfälligen Kanälen absichern. Die Kryptographie entwickelt *Kryptosysteme* zum Zweck der Geheimhaltung von Information. Ein Kryptosystem wird immer dann benötigt, wenn Nachrichten auf *unsicheren* Kanälen übermittelt werden. Ein unsicherer Kanal ermöglicht es unbefugten Dritten, die übermittelten Nachrichten ebenfalls zu empfangen, vielleicht sogar abzufangen und verfälscht weiterzuleiten. Ein Kanal ist zum Beispiel auch dann unsicher, wenn er es Dritten ermöglicht, dem Absender oder dem Empfänger eine falsche Identität vorzuspiegeln.

Die klassischen Anwendungsbereiche von Kryptosystemen sind Militär, Diplomatie und Geheimdienste. Aber bereits im 19. Jahrhundert, als auch Nachrichten von wirtschaftlicher Bedeutung von Telegraphisten im Morse-Zeichen übertragen und relativ offen über Telegraphenleitungen gesendet wurden, spielte die Kryptographie auch für die Wirtschaft eine erhebliche Rolle.

Inzwischen sind wir alle zu Anwendern von Kryptosystemen geworden, und wir können dafür beispielhaft den Einkauf im Internet, das Login in einem Computer-Netzwerk, die Benutzung von Geldautomaten, Electronic Banking und das Telefonieren über mobile Telefone nennen. In all diesen Fällen ist zumindest eine Feststellung der Identität des Nutzers von entscheidender Bedeutung und, soweit Daten übertragen werden, auch deren Geheimhaltung.

Das folgende Diagramm beschreibt das Zusammenwirken der Komponenten eines Kryptosystems (oder einer *Chiffre*):



Formal können wir ein Kryptosystem als ein Quintupel

$$(\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$$

beschreiben, bei dem $\mathscr{P}$ die Menge der *Klartexte* und $\mathscr{C}$ die Menge der *Geheim-* oder *Chiffretexte* ist. Diese sind Wörter über dem *Klartext-* bzw. *Geheimtextalphabet*. Mit $\mathscr{K}$ bezeichnen wir die Menge der *Schlüssel* oder den *Schlüsselraum*, und

die *Chiffrierung* (oder *Verschlüsselung*) $\mathscr{E}$ ist eine Familie von Abbildungen

$$E_K : \mathscr{P} \to \mathscr{C}, \quad K \in \mathscr{K},$$

während die *Dechiffrierung* (oder *Entschlüsselung*) eine Familie von Abbildungen

$$D_K : \mathscr{C} \to \mathscr{P}, \quad K \in \mathscr{K},$$

ist mit

$$D_K \circ E_K = \mathrm{id}_{\mathscr{P}}.$$

Chiffrierung und Dechiffrierung werden also durch den Schlüssel gesteuert. Die Zuordnungen von E_K und D_K zu $K \in \mathscr{K}$ stellen den Chiffrier- und den Dechiffrieralgorithmus dar. (Bei der Wahl der Bezeichnungen sind wir den englischen Namen *plaintext*, *cyphertext*, *key*, *enciphering*, *deciphering* gefolgt.)

Um insbesondere einigen klassischen Kryptosystemen gerecht zu werden, müßte man die Bedingungen an die Chiffrierung etwas auflockern: Es genügt, daß E_K eine linkseindeutige Relation ist. Dies läßt die Verschlüsselung eines Klartextes durch verschiedene Geheimtexte zu.

Eine wichtige Maxime der Kryptographie lautet: *Der Gegner kennt das Verfahren*. Man darf sich also nicht auf die Geheimhaltung des Chiffrierverfahrens $\mathscr{E}$ und des Dechiffrierverfahrens $\mathscr{D}$ verlassen. Für die Sicherheit ist es mindestens erforderlich, daß es dem Gegner nicht gelingt, bei bekanntem Geheimtext den Schlüssel zu finden. Dies setzt offensichtlich einen sehr großen Schlüsselraum voraus, insbesondere seit es möglich ist, mit Computerhilfe sehr viele Schlüssel in kurzer Zeit zu probieren. An moderne Kryptosysteme sind aber noch strengere Anforderungen zu stellen, die wir noch diskutieren werden.

Die klassischen Kryptosysteme sind *symmetrisch*: bei ihnen ist es sehr „leicht“, aus der Verschlüsselung E_K die Umkehrabbildung D_K zu bestimmen. Dies ist zum Beispiel der Fall bei allen Verfahren, die auf Permutationen des Klartextalphabets (oder des Klartextes) beruhen. Die Verwendung eines solchen symmetrischen Systems macht den sicheren Schlüsseltausch zwischen Sender und Empfänger zwingend erforderlich: In früheren Zeiten nahm das Kriegsschiff vor dem Auslaufen ein Codebuch an Bord, das auf keinen Fall dem Feind in die Hände fallen durfte, und der Geheimagent verwahrte seine Schlüssel im hohlen Schuhabsatz.

Der sichere Schlüsseltausch erfordert einen zweiten, sicheren Kanal zwischen Sender und Empfänger. Bei dem Einkauf via Internet ist ein vorheriger Schlüsseltausch aber nicht realisierbar. Diese Schwierigkeit läßt sich durch ein *asymmetrisches* Kryptosystem lösen. Es kommt ja nicht eigentlich darauf an, den Gegner daran zu hindern, Klartexte zu verschlüsseln. Man muß es ihm vielmehr unmöglich machen, Geheimtexte zu entschlüsseln. Wenn es nicht möglich ist, ohne Zusatzinformation die Umkehrabbildung D_K allein aus E_K zu bestimmen, darf ja E_K durchaus bekannt sein. Der Empfänger R kann dem Sender offen mitteilen, welchen Schlüssel K er für die Verschlüsselung von Nachrichten an R benutzen soll. Nur

der Empfänger hat die Zusatzinformation, die ihm die Dechiffrierung ermöglicht. Da der Schlüssel öffentlich bekannt ist, heißen solche Systeme *Public-Key-Kryptosysteme* oder *Kryptosysteme mit öffentlichem Schlüssel*.

Die Forderung, es solle unmöglich sein, D_K allein aus E_K zu bestimmen, ist mathematisch natürlich unsinnig. Wir müssen sie so verstehen: Die Ermittlung von D_K allein aus E_K muß einen Aufwand erfordern, der den durch das Brechen der Verschlüsselung zu erzielenden Gewinn bei weitem überwiegt, oder eine so lange Zeit benötigen, daß nach ihrem Ablauf jegliches Interesse an der zu erlangenden Information erloschen ist. Funktionen E_K, die dieser Forderung genügen, heißen *Einweg-Funktionen*. Da die Dechiffrierung aber möglich sein soll, muß es eine (geheime) *Falltür* geben, eben die Zusatzinformation, die die Berechnung von D_K ermöglicht.

In der Kryptographie sind auch Einweg-Funktionen ohne Falltür von Interesse, Sie werden zum Beispiel für die Überprüfung von Paßwörtern von Betriebssystemen wie Windows oder Unix benutzt.

Man kann den Aufwand, den die Umkehrung von E_K erfordert, mit Mitteln der *Komplexitätstheorie* messen. Wir verweisen auf [**Ko2**] für eine Einführung in die Komplexitätstheorie.

In der Praxis verwendet man heute oft *Hybridverfahren*, bei denen ein aufwendiges asymmetrisches Verfahren genutzt wird, um Schlüssel für ein effizientes symmetrisches Verfahren auszutauschen.

Blockchiffren sind den Blockcodes vergleichbar. Die Symbole des Klartextalphabets A werden zu Blöcken $a_1 \dots a_k$ einer festen Länge k zusammengefaßt. Jeder solche Klartextblock wird dann durch einen Geheimtextblock $b_1 \dots b_n$ mit ebenfalls festem n über dem Geheimtextalphabet B chiffriert. Die Chiffrierung erfolgt also durch Abbildungen

$$E_K : A^k \to B^n.$$

Dabei wird für jeden Block der gleiche Schlüssel verwendet. Meistens ist dabei $A = B$ und $k = n$. Im Fall $n > k$ spricht man von einer *gespreizten* Chiffre. (Der Fall $k < n$ ist bei $A = B$ natürlich ausgeschlossen.)

Stromchiffren hingegen verwenden für jedes Zeichen des Klartexts (oder jeden Block des Klartexts) eine andere Abbildung E_K. Sie müssen daher eine fortlaufende Schlüsselfolge erzeugen. Die Abgrenzung von Block- und Stromchiffren ist allerdings nicht immer einfach.

Man unterscheidet verschiedene Arten von Angriffen gegen Kryptosysteme:

(a) *Geheimtext-Angriff:* Dem Angreifer ist nur der Geheimtext bekannt. Er versucht, daraus Klartext und Schlüssel zu bestimmen. Wenn ein Geheimtext-Angriff erfolgreich sein kann, ist ein Kryptosystem wertlos. Wie wir im folgenden Abschnitt sehen werden, sind viele klassische Chiffren mit Geheimtext-Angriffen zu brechen.

Auch den *Brute-Force-Angriff* durch Exhaustion des Schlüsselraums kann man zu den Geheimtext-Angriffen rechnen.

(b) *Klartext-Angriff:* Man muß häufig unterstellen, daß der Angreifer nicht nur den Geheimtext, sondern zu einem Teil auch den zugehörigen Klartext kennt. Wenn es dann möglich ist, den Schlüssel zu finden, ist das System gebrochen. Von modernen symmetrischen Verfahren wird erwartet, daß sie Klartext-Angriffen widerstehen.

Für die Sicherheit gegen Klartext-Angriffe ist es erforderlich, daß die Zuordung des Paares $(P, E_K(P))$ zum Schlüssel K eine Einweg-Funktion ohne Falltür ist.

(c) *Angriff mit gewähltem Klartext:* Der Angreifer kann die Geheimtexte zu von ihm gewählten Klartexten erzeugen. Dies ist geradezu das Kennzeichen von Public-Key-Verfahren, und diese müssen natürlich gegen alle Angriffe mit gewähltem Klartext sicher sein.

Aber auch bei symmetrischen Systemen macht die Forderung nach Sicherheit gegen Angriffe mit gwähltem Klartext Sinn. Wenn dem Angreifer ein Chiffriergerät in die Hände fällt, ist er zumindest nicht in der Lage, den Schlüssel zu extrahieren und empfangene Geheimtexte zu dechiffrieren.

Insbesondere bei Public-Key-Verfahren ist die Vorspiegelung falscher Identitäten eine große Gefahr. Man muß Methoden zur sicheren Identifizierung des Senders und des Empfängers entwickeln. Dazu dienen kryptographische *Protokolle*. Für die sichere Verwahrung und Verteilung öffentlicher Schlüssel benötigt man eine gut organisierte *Public-Key-Infrastruktur*.

Der Begriff „Kryptographie“ wird in der Literatur meistens so verstanden, daß diese Wissenschaft Kryptosysteme konstruiert, während die *Kryptanalyse* Methoden entwickelt, mit denen sich Kryptosysteme brechen lassen. Zusammen bilden Kryptographie und Kryptanalyse die *Kryptologie*.

Zur Kryptologie gehört auch die *Steganographie*. Bei ihr geht es um Methoden, geheime Informationen in öffentlich zugänglichen so zu verstecken, daß sie nur der eingeweihte Beobachter entdecken kann. Auch die Entzifferung historischer Schriften, deren Bedeutung nicht mehr bekannt ist, kann man zur Kryptologie rechnen.

Das Cryptool. Ein ausgezeichnets Hilfsmittel zur Demonstration vieler klassischer und moderner Kryptosysteme ist das *Cryptool*, das man über [**Cry**] beziehen kann.

ABSCHNITT 12

Klassische Chiffren

Caesar-Addition. Eine der ältesten Chiffren wird Gaius Julius Caesar zugeschrieben und deshalb oft „Caesar-Addition“, „Caesar-Verschiebung“ oder noch kürzer „Caesar“ genannt. Man faßt dazu die 26 Buchstaben des (deutschen) Alphabets als Repräsentanten der Elemente von $\mathbb{Z}_{26}$ auf, wobei (zum Beispiel) *A* der Restklasse von 0, *B* der Restklasse von 1 usw. entspricht (und *Z* der von 25). Nun wählt man als Schlüssel einen Buchstaben aus und „addiert“ ihn zu jedem Buchstaben des Klartexts, um den Geheimtext zu erhalten:

Klartext	`veni vidi vici`
Schlüssel	`EEEE EEEE EEEE`
Geheimtext	`ZIRM ZMHM ZMGM`

Diese Chiffre mag genügen, den Klartext vor den Augen eines zufälligen Mitlesers zu verbergen, aber die antiken Kryptanalytiker werden kein Problem damit gehabt haben: Der Schlüsselraum enthält nur 26 Schlüssel (bei Verwendung des deutschen Alphabets), und man kann diese auch per Hand leicht durchprobieren. Diesen “Angriff“ nennt man *Exhaustion des Schlüsselraums*. Die Grundidee ist aber dennoch ausgezeichnet und wird in vielen anderen Verfahren angewandt. Im Usenet wird `ROT13`, die zu sich selbst inverse Caesar-Verschiebung des englischen Alphabets um 13 Buchstaben, verwendet, um die Lösungen von Rätseln zu verbergen oder möglicherweise beleidigenden Text zu kaschieren.

Monalphabetische Substitution. Die Caesar-Addition können wir als eine spezielle Permutation des Alphabets betrachten, nämlich als zyklische Verschiebung um mehrere Stellen. Der Schlüsselraum vergrößert sich gigantisch, wenn wir beliebige Permutationen des Alphabets als Schlüssel zulassen, nämlich auf $26! \approx 4 \cdot 10^{26}$ mögliche Schlüssel. Es ist aber enorm schwierig, sich eine Permutation zu merken, und deshalb benutzt man oft ein *Schlüsselwort*, um sie festzulegen. Das Schlüsselwort wird unter den Anfang des Alphabets geschrieben, wobei mehrfach auftretende Buchstaben nur bei ihrem ersten Erscheinen berücksichtigt werden. Sodann schreibt man die im Schlüsselwort nicht vorkommenden Buchstaben der Reihe nach unter die restlichen Buchstaben des Alphabets, wobei wir mit dem Buchstaben beginnen, der auf den letzten im Schlüsselwort vorkommenden Buchstaben des Alphabets folgt. (Das kann man variieren.) Bei Verwendung des Schlüsselworts POMPEIUS ergibt sich folgende Permutation:

Klartextalphabet	`abcdefghijklmnopqrstuvwxyz`
Geheimtextalphabet	`POMEIUSVWXYZABCDFGHJKLNQRT`

Diese Chiffre führt dann also zu folgender Verschlüsselung:

Klartext	`gallia est omnis divisa in partes tres`
Geheimtext	`SPZZWS IJH CABWH EWLWHS WB DPGJIH JGIH`

Chiffren dieser Art heißen *monalphabetische Substitutionen*, weil man dem Klartextalphabet ein einziges Geheimtextalphabet gegenüber stellt. (Wir fassen hier ein Alphabet nicht als Menge, sondern als Folge seiner Buchstaben auf.) In vielen historischen Chiffren wurden dabei die Buchstaben des Geheimtextalphabets phantasievoll gestaltet. Ein Beispiel ist Maria Stuarts Nomenklator (Abbildung 1, entnommen aus [**Si**], p. 57) Er weist nicht nur einzelnen Buchstaben Zeichen des Ge-

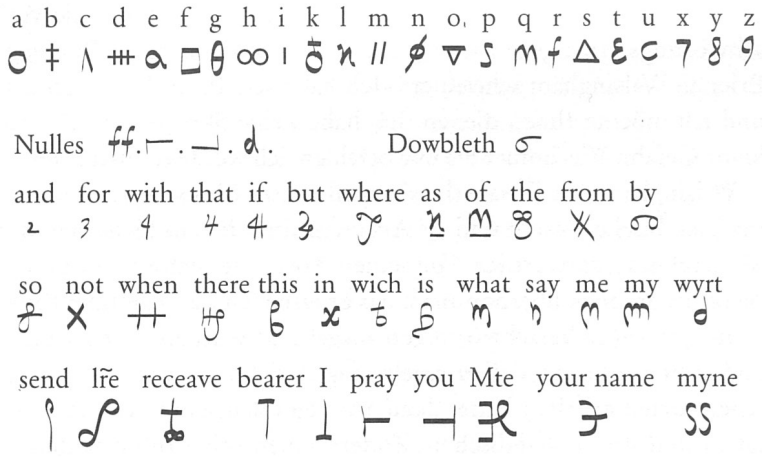


ABBILDUNG 1. Maria Stuarts Nomenklator

heimtextalphabets zu, sondern auch noch einigen häufig vorkommenden Wörtern. (Die Ersetzung ganzer Wörter durch andere nennt man häufig einen *Code*.) Die mit dem Nomenklator chiffrierte Korrespondenz Maria Stuarts mit dem Verschwörer Babington wurde nicht nur dechiffriert. Der Geheimdienst der Königin Elisabeth, deren Ermordung Babington und Maria planten, fügte einem der Briefe Marias sogar noch ein Postskriptum an, in dem Babington aufgefordert wurde, die Namen der Mitverschwörer zu nennen, was dieser dann in seiner Antwort auch prompt tat. Am Ende rollten Köpfe, am 8. Februar 1587 auch der Marias.

Häufigkeitsanalyse. Monalphabetische Substitutionen haben einen entscheidenden Schwachpunkt: Sie übertragen alle statistischen Merkmale des Klartexts auf den Geheimtext, insbesondere die Verteilung der Häufigkeit der einzelnen Buchstaben, die gegenüber dem Klartext nur permutiert wird. Die folgende Tabelle

(entnommen [**Si**], p. 36) gibt die Häufigkeit der einzelnen Buchstaben im Deutschen und im Englischen an. Man findet in der Literatur auch geringfügig andere Angaben, da verschiedene Texte ausgezählt worden sind.

Buchstabe	Häufigkeit in %	
	Deutsch	Englisch
a	6,5	8,2
b	1,9	1,5
c	3,1	2,8
d	5,1	4,3
e	17,4	12,7
f	1,7	2,2
g	3,0	2,0
h	4,8	6,1
i	7,5	7,0
j	0,3	0,2
k	1,2	0,8
l	3,4	4,0
m	2,5	2,4

Buchstabe	Häufigkeit in %	
	Deutsch	Englisch
n	9,8	6,7
o	2,5	7,5
p	0,8	1,9
q	0,02	0,1
r	7,0	6,0
s	7,3	6,3
t	6,2	9,1
u	4,4	2,8
v	0,7	1,0
w	1,9	2,4
x	0,03	0,2
y	0,04	2,0
z	1,1	0,1

TABELLE 1. Buchstabenhäufigkeit in deutschen und englischen Texten

Die statistischen Merkmale von Sprachen wurden erstmals von arabischen Gelehrten erforscht, die mit ihrer Hilfe die Entstehung der Koran-Texte chronologisch ordnen wollten. Bei der Kryptanalyse einer monalphabetischen Substitution wird man also zunächst eine Häufigkeitstabelle der Einzelzeichen des Geheimtextes anfertigen und diesen dann ihrer Häufigkeit gemäß Buchstaben des Klartextes zuweisen. Bei kurzen Texten treten natürlich Abweichungen zwischen den erwarteten und den beobachteten Häufigkeiten auf, so daß die Zuordnung nicht immer auf Anhieb möglich ist. Man muß dann experimentieren, und nach probeweiser Zuordnung der häufigsten Buchstaben Wörter ergänzen und so weitere Zuordnungen finden. Von einem bestimmten Punkt ab tritt dann der Klartext hervor, und der Rest ist sehr einfach.

Wir studieren diese Methode an folgendem Geheimtext:

```
QWIVOQVOHIJAXOPRWSVHPYJJDJIOQJIOXIOJCFOWZOSSNJPWAXICHQROJIOPVWOJAXHOSOKJAXPOWVO
PWPWPIOKPBIWNPBSOPSOWJIHPYJGOKYSOWAXOPQOWJIQBOJJWYRWJQWJOKBROSBRVWOBRWIHKWOPIOP
LHNIOSWOYIPWOVKWYOKBSJWPBPVOKOPSBOPVOKPHPVVWOXNAXJAXHSOPGOKCOWAXPOPBIOQROKBHROP
VOBRRKOAXOKCBXSOPHPVBHJHZOKPVOJIHVWOPCOWIOPPBAXCFOWUBOXKWYOKBKROWIXBIVBJZNKHQRW
SVHPYOQMZOXSHPYOPGNKYOSOYIVWOOJWPJWAXXBROPGNQTWPVOKYBKIOPRWJCHKXNAXJAXHSOFNSSOP
RHPVHPVSBOPVOKYOQOWPJBQKOZNKQOPBPMBATOP
```

Die Auszählung der Buchstaben ergibt folgende Häufigkeiten:

A	B	C	D	E	F	G	H	I	J	K	L	M
14	25	7	1	0	3	4	21	22	27	25	1	2

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
10	76	47	14	15	20	2	1	22	36	21	13	5

Wir ersetzen nun die groß geschriebenen Geheimtext-Buchstaben versuchsweise durch klein geschriebene Klartext-Buchstaben. Im ersten Versuch führen wir folgende Ersetzungen durch, wobei wir den in obiger Tabelle gegebenen Häufigkeiten folgen:

```
OPWJKBIVX
enirsatdh
```

Wir erhalten die partielle Dechiffrierung

```
QitdeQdeHtrAhenRiSdHnYrrDrteQrtehterCFeiZeSSNrniAhtCHQRertendierAhHeSesrAhneide
nininteSnatiNnaSenSeirtHnYrGesYSeiAhenQeirtQaerriYRirQiresaReSaRdieaRitHsienten
LHNteSieYtniedsiYesaSrinandesenSaendesnHnddiehNAhrAhHSenGesCeiAhnenateQResaHRen
deaRRseAhesCahSenHndaHrHZesndertHdienCeitennaAhCFeiUaehsiYesasReithatdarZNsHQRi
SdHnYeQMZehSHnYenGNsYeSeYtdieerinriAhhaRenGNQTindesYastenRirCHshNAhrAhHSeFNSSen
RHndHndSaendesYeQeinraQseZNsQenanMaATen
```

Auffällig ist die Folge `inintesnatiNnaSen`, die man wohl nur dann erklären kann, wenn das erste Vorkommen von `in` das Wort „in“ ist, und `intesnatiNnaSen` muß für „internationalen“ stehen, wobei klar wird, daß wir `s` und `r` falsch zugeordnet haben. Wir vertauschen die beiden Buchstaben und nutzen aus, daß `N` vermutlich für `o` und `S` für `l` steht. Damit ergibt sich

```
QitdeQdeHtsAhenRildHnYssDsteQstehtesCFeiZellosniAhtCHQRestendiesAhHelersAhneide
nininternationalenleistHnYsGerYleiAhenQeistQaessiYRisQiseraRelaRdieaRitHrienten
LHotelieYtniedriYeralsinanderenlaendernHnddiehoAhsAhHlenGerCeiAhnenateQReraHRen
deaRRreAherCahlenHndaHsHZerndestHdienCeitennaAhCFeiUaehriYerarReithatdasZorHQRi
ldHnYeQMZehlHnYenGorYeleYtdieesinsiAhhaRenGoQTinderYartenRisCHrhoAhsAhHleFollen
RHndHndlaenderYeQeinsaQreZorQenanMaATen
```

`dieaRitHrienten` sind wohl die „die Abiturienten“, und nach dieser Erkenntnis kann man den Text schon fast flüssig lesen. Der Rest ist ein Kinderspiel. Wir erhalten die endgültige Zuordnung

```
OPWJKBIVXNSRHQAYDCFZGLMLTU
enisratdholbumcgyzwfvqpqkj
```

und als Klartext

```
mitdemdeutschenbildungssystemstehteszweifellosnichtzumbestendieschuelerschneide
nininternationalenleistungsvergleichenmeistmaessigbismiserabelabdieabiturienten
quoteliegtniedrigeralsinanderenlaendernunddiehochschulenverzeichnenatemberauben
deabbrecherzahlenundausuferndestudienzeitennachzweijaehrigerarbeithatdasforumbi
ldungempfehlungenvorgelegtdieesinsichhabenvomkindergartenbiszurhochschulewollen
bundundlaendergemeinsamreformenanpacken
```

(Der Text wurde einer Seite von `www.spiegel.de` entnommen.)

Als Ergänzung der Häufigkeitsanalyse kann man die Methode des *wahrscheinlichen Wortes* heranziehen. Die Koinzidenzmuster von Wörtern bleiben unter bijektiven Abbildungen des Alphabets erhalten. Natürlich kann man nicht auf ein so einzigartiges Wort wie

```
                    Mississippi
Koinzidenzmuster    12332332442
```

hoffen, aber auch schon „Division“ hat in einem militärischen Kontext ein verräterisches Muster, von Wortmonstern wie „Generalfeldmarschall“ ganz abgesehen. Entdeckt man das Koinzidenzmuster eines wahrscheinlichen Wortes in einem Geheimtext, so wird man es probeweise zu einem Ansatz für die Dechiffrierung ausnutzen. Gerade in militärischen Nachrichten werden oft Dienstgrade und die Namen von Truppenteilen auftauchen, so daß es reichlich Ansatzpunkte der Kryptanalyse gibt. Mitunter fordert der Gegner sogar durch gezielte Aktionen Mitteilungen heraus, die dann mit großer Sicherheit bestimmte Wörter enthalten.

Die Häufigkeitsanalyse der einzelnen Buchstaben kann und muß oft durch die Bigramm-Statistik ergänzt werden, bei der man auch die Häufigkeit von Buchstabenpaaren heranzieht. Insbesondere bei kurzen Geheimtexten kann es schwierig sein, Buchstaben mit annähernd gleicher Häufigkeit richtig zuzuordnen – beim obigen Beispiel haben wir Glück gehabt.

Homophone und Blender. Eine schon recht wirksamer Ansatz, die Häufigkeitsanalyse und Koinzidenzmuster zu konterkarieren, besteht in der Verwendung von *Homophonen* und *Blendern*. Man verwendet für jeden Buchstaben des Klartextes so viele Homophone im Geheimtextalphabet, wie es seiner Häufigkeit entspricht. Dabei können die Buchstabenmengen von Klartextalphabet und Geheimtextalphabet natürlich nicht mehr identisch sein. Auf diese Weise erreicht man annähernd Gleichverteilung im Geheimtext. Blender sind bedeutungslose Zeichen, die wahllos in den Text eingestreut werden. (Sie treten in Maria Stuarts Nomenklator als „Nulles“ auf; „Dowbleth“ dient dazu Doppelbuchstaben anzuzeigen.) Ein berühmtes Beispiel einer (im wesentlichen) monalphabetischen Substitution ist die „Grande Chiffre“ Ludwig des XIV., die erst Ende des 19. Jahrhunderts gebrochen wurde. Eine homophone Chiffre wurde im Abendland erstmals 1401 vom Herzog von Mantua verwendet; siehe Abbildung 2 (entnommen [**Ka1**], p. 107). In ihr sind Homophone für die Vokale außer „i“ vorgesehen.

Wir wollen es uns hier ersparen, eine Liste von Chiffrierregeln anzugeben, die man befolgen sollte, um dem Kryptanalytiker die Arbeit so schwer wie möglich zu machen. Dieses Thema wird in [**Ba**] eingehend diskutiert. Klar ist aber, daß man auf keinen Fall den Wortzwischenraum anzeigen darf.

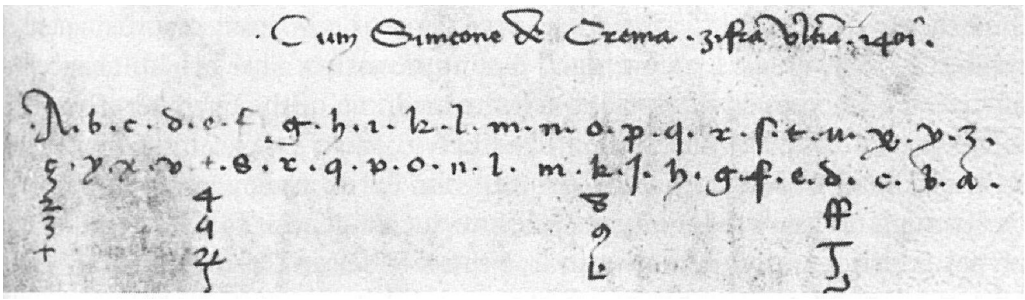


ABBILDUNG 2. Homophone Chiffre des Herzogs von Mantua

Vigenère-Chiffre und Periodenanalyse. Eine entscheidende Verbesserung gegenüber monalphabetischer Substitution ist die *polyalphabetische Substitution*, deren klassische Form die *Vigenère-Chiffre* ist. Sie verwendet mehrere Caesar-Additionen, die periodisch, durch ein Schlüsselwort gesteuert abgewechselt werden:

Klartext	`galliaestomnisdivisainpartestres`
Schlüssel	`BRUTUSBRUTUSBRUTUSBRUTUSBRUTUSBR`
Geheimtext	`HRFECSFJNHGFJJXBPATRCGJSSKYLNJFJ`

Die Vigenère-Chiffre galt über einige Jahrhunderte als „chiffre indechiffrable“. Sie ebnet Häufigkeitsverteilungen ein und bricht Koinzidenzmuster. Vigenère hat sein Hauptwerk 1586, ein Jahr vor Maria Stuarts Tod, veröffentlicht.

Will man eine Vigenère-Chiffre brechen, so muß man zuerst die Länge des Schlüsselworts herausfinden. Kennt man diese, so kann man den Text in Teile zerlegen, wobei wir in jedem Teil die Buchstaben sammeln, deren Position im Geheimtext zu einer festen Restklasse nach der Schlüssellänge gehören. In jedem „Restklassen-Teiltext“ kann man die Häufigkeit der einzelnen Buchstaben ermitteln, und so eine Dechiffrierung versuchen.

Der kryptanalytische Durchbruch gelang dem pensionierten preußischen Offizier F. W. Kasiski im Jahr 1863. (Die Entdeckung dieses Verfahrens wird oft auch Ch. Babbage zugeschrieben – mit Sicherheit haben beide nichts voneinander gewußt.) In einem genügend langem Geheimtext lassen sich *Parallelstellen* finden, also doppelt auftauchende Buchstabenfolgen. Dies können „unecht“ sein, also zufällig durch die Chiffre erzeugt, oder „echt“, wenn nämlich der Klartext Parallelstellen enthält, deren Abstand ein Vielfaches der Schlüssellänge ist. Parallelen von einzelnen Buchstaben sind natürlich wertlos, und solche von Bigrammen meistens auch. Mit wachsender Länge der Buchstabenfolge, die mehrfach auftritt, wird die zufällige Erzeugung jedoch unwahrscheinlicher. Man wird also Parallelstellen im Geheimtext suchen, und den größten gemeinsamen Teiler der Abstände der wahrscheinlich echten Parallelstellen als Schlüssellänge annehmen. Die Crux

des Verfahrens ist natürlich die Unterscheidung von echten und unechten Parallelen, die nicht immer zuverlässig gelingt. Ist der Text lang genug, gibt es jedoch meistens auch sehr lange Parallelstellen. Wir betrachten den folgenden (zugegebenermaßen schon recht langen) Geheimtext:

```
DMYRVYVXUNJGMSENAEDOEKXGPELXMMKIMHVERPECWIQZFEFBCBKDZASQKMEHUMJGTTMXLYIWHVEQAWE
HZRNBKQJGANZSSOCQFEECJXZBXENXRACINQYQFFECJXROVEKBGVZWRWJQJTBYCEGRZQSUINLVNSEFWG
QOFXJZZQYMNCVHWWXQJTLMZRFBUQJXNFRISRVDFNNXUMJVFOZLCBLPJBMQJSECTLSSEMLXMVVVFISQF
WEUSFWSTTWKZUYPJBLZVTUMLJJFEPWLTOUMJBQQAMEHEEHVQIWBJUVLWWXQJTRVVMYVRFVTSZFVZASU
DWUHXIRDWQZEUHXISJFDYXLYXXIWVQKBNMZGMVRNWGVIDONBUQJZALKISPZERNRBFGMGTTMEEQFPQSE
NMGDOEHQOVZVXRAVQJWEESFRYWSWAVZSGPUTOJBJQDMEHNEWSEEAVHMFDNOCPWFOEIEYSEGFWUHZSSG
GADBTCBIWGFQAGIANMJODYAMTQFGMOCEKBEXRWUOGUWKVIIWYSCXLXNUDJTFLYKBNXEIGSEBGEINZOJ
FEMMVHQZWXSEEUAAZKPJFRDTXINXIGSISWPELBWHVRRLXNEZVHVVZKHWCVZJFKDWMELMSSSCFWKNOEH
XQYGWEELSIYSZXAZTZRPQGUUWXMJWIMZLZYXNODKJGVFRMWYIHJBNMJMENVMSSIMVBKUCOZFRGXWAMS
MQRLZYLSSJXJAJOZHNCDONBUQJZALKISGFXDXNXZIPZVUFLTYEJWSDPKIRUTLJBCQJGEHLRIPVEKXRU
CWGWJTWKAOWHNSJOZNLYMSWPVDWBTYKAJFUQFYUYIONBUQJMAAVWJWEDAVHNLRLSEEGELYEHNSVXLXR
HDSJUCUUASNBINBVSWUUYYVJBQMZEEHDYJGJQF
```

Die Parallelstellen-Suche ergibt folgende Parallelstellen von Länge mindestens 4.

```
[GTTM] Position 64 Abstand 320     [UHXI] Position 319 Abstand 10
[ECJX] Position 99 Abstand 20      [DONBUQJZALKIS] Position 361 Abstand 370
[WWXQJT] Position 173 Abstand 120  [ONBUQJ] Position 732 Abstand 100
[BUQJ] Position 184 Abstand 180
```

Damit liegt die Schlüssellänge 10 offen. Da nur Caesar-Verschiebungen benutzt werden, genügt es, in jeder Restklasse nach der Schlüssellänge einen einzigen Buchstaben richtig zuzuordnen. Dann ist das Schlüsselwort bekannt und der Klartext gewonnen. Die häufigsten Buchstaben in jeder Restklasse (wobei in einer Restklasse `E` und `V` gleich oft vorkommen):

```
ZIJSE(V)QWXEY
```

Die probeweise Chiffrierung von `E` zu diesen Buchstaben führt bei Caesar-Verschiebung zu folgendem Schlüsselwort:

```
VEFOAMSTAU
```

Mit der Alternative `V` statt `E` an der 5. Stelle erhalten wir

```
VEFORMSTAU
```

Hier stört irgendwie das `V` an der 1. Stelle. Wenn wir dann noch einmal in die Häufigkeitstabelle sehen, finden wir `V` in der Restklasse von 1 als zweithäufigsten Buchstaben. Wenn wir ihn als Chiffrat von `E` annehmen, ergibt sich als mögliches Schlüsselwort

```
REFORMSTAU
```

und mit dem sind wir sehr zufrieden, wenn auch nicht mit dem politischen Sachverhalt, den es beschreibt. (Man kann natürlich feinere Techniken benutzen, um

das Schlüsselwort zu finden.) Die Entschlüsselung ist

```
mitdemdeutschenbildungssystemstehteszweifellosnichtzumbestendieschuelerschneide
nininternationalenleistungsvergleichenmeistmaessigbismiserabelabdieabiturienten
quoteliegtniedrigeralsinanderenlaendernunddiehochschulenverzeichnenatemberauben
deabbrecherzahlenundausuferndestudienzeitennachzweijaehrigerarbeithatdasforumbi
ldungempfehlungenvorgelegtdieesinsichhabenvomkindergartenbiszurhochschulewollen
bundundlaendergemeinsamreformenanpackenseltenwarensichsozialdemokratenundunions
politikersoeinigwieammittwochalssiedaspapiervorstelltenamforumsindnebenpolitike
rnauchwissenschaftlerarbeitgebergewerkschaftenkirchensowievertretervonelternund
schuelerbeteiligtfallsdieempfehlungenumgesetztwerdenwarteteineradikalkuraufdasb
ildungssystemschonimkindergartensollendiekleinstenfremdsprachenlernenundbessera
lsbisheraufdieschulevorbereitetwerdenfuerkindertageseinrichtungensollendieelter
nmoeglichstkeinegebuehrenzahlenmuessen
```

(Den Anfang des Textes haben wir oben schon gesehen.) Natürlich ist der Angriff um einiges schwieriger, wenn man statt der Caesar-Verschiebungen beliebige Permutationen des Alphabets verwendet. Man muß dann aber zum Beispiel 10 solche Permutationen fixieren, was bereits einen gewissen Aufwand erfordert und ohne maschinelle Unterstützung recht fehleranfällig ist.

Autokorrelation. Es gibt eine viel zuverlässigere, auf *W. F. Friedman* zurückgehende Methode, die Schlüssellänge zu finden. Dazu vergleicht man den Text T mit einer um t Stellen verschobenen Kopie und definiert

$$\varkappa(T,t) = \frac{\iota(T,t)}{(L-t)}$$

wobei L die Länge von T ist und $\iota(T,t)$ die Anzahl der Positionen i, $1 \le i \le L-t$, für die $T_i = T_{i+t}$. Wir nennen $\varkappa(T,t)$ die *Autokorrelation* von T zur Verschiebung t. Für großes t besteht kein statistischer Zusammenhang zwischen den Buchstaben an den Stellen j und $j+t$. Daher sollte

$$\varkappa(T,t) \approx \sum_{i=1}^{n} p_i^2$$

für hinreichend große t sein, wenn das Alphabet n Buchstaben hat, die mit den Wahrscheinlichkeiten $p_1,\ldots,p_n$ vorkommen. Für die deutsche Sprache mit den in Tabelle 1 gegebenen relativen Häufigkeiten der Buchstaben erwarten wir $\varkappa(T,t) \approx 0,076$.

Der Vergleich des Alphabets mit einer durch τ permutierten Version ergibt als Wahrscheinlichkeit für übereinstimmende Zeichen

$$\varkappa_\tau = \sum_{i=1}^{n} p_i p_{\tau^{-1}(i)}.$$

Wenn wir dies über alle Permutationen τ mitteln, ist

$$\overline{\varkappa} = \frac{1}{n}$$

der Erwartungswert. (Siehe Aufgabe 12.6.) Bei $n = 26$ ist $\overline{\varkappa} = 1/26 \approx 0,038$.

Sei nun s die Schlüssellänge. Für den Geheimtext G zum Klartext T gilt dann $\varkappa(G,ks) = \varkappa(T,ks)$ für alle $k \geq 0$. Ist dagegen die Verschiebung t kein Vielfaches von s, so wird das Alphabet mit permutierten Versionen verglichen, und dabei wird auch noch über viele Permutationen gemittelt. Wir erwarten $\varkappa(G,t) \approx 1/n$ für Verschiebungen t, die kein Vielfaches der Schlüssellänge sind.

Die Vielfachen der Schlüssellänge sollten sich also in den Maxima der Funktion $\varkappa(G,t)$ klar abzeichnen. Für den obigen Klartext T und den zugehörigen Geheimtext G ergibt sich

t	1	2	3	4	5	6	7	8	9	10	11	...	18	19	20	21
$\varkappa(T,t)$	24	47	89	59	75	94	63	76	93	62	80	...	69	66	82	77
$\varkappa(G,t)$	46	37	50	31	31	35	44	26	32	62	35	...	40	25	82	36

Wir haben in der zweiten und dritten Zeile die Einheit 10^{-3} gewählt. Die Maxima bei 10 und 20 sind klar erkennbar, und damit ist die Schlüssellänge gefunden. Noch deutlicher wird sie in der graphischen Darstellung (Abbildung 3). Tabelle

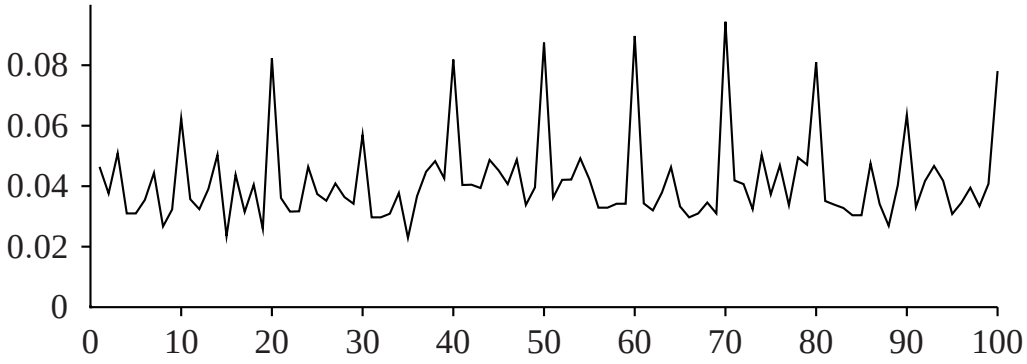


ABBILDUNG 3. Autokorrelation des Vigenère-chiffrierten Geheimtextes

und graphische Darstellung bestätigen unsere Erwartung von $\varkappa(T,ks) \approx 0,076$ und $\varkappa(T,t) \approx 1/26 \approx 0,38$ für $t \neq ks$, $k \in \mathbb{N}$.

Playfair-Chiffre. Eine bekannte Bigramm-Substitution ist die Playfair-Chiffre, die 1854 von Ch. Wheatstone erfunden wurde, aber nach seinem Freund L. Playfair benannt ist. Man füllt dazu eine 5×5-Matrix mit den Buchstaben des Alphabets (unter Auslassung von J, das mit I identifiziert wird). Man kann diese Matrix mittels eines Schlüsselworts festlegen, z. B.

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

Die Verschlüsselung der Buchstaben erfolgt paarweise so:

(a) Stehen die Buchstaben eines Paares in einer Zeile, so gehen wir von beiden Buchstaben einen Schritt zyklisch nach rechts und setzen die so erhaltenen Buchstaben ein. Zum Beispiel wird `VZ` durch `WV` verschlüsselt.

Stehen die Buchstaben in einer Spalte, so gehen wir eine Stelle zyklisch nach unten.

(b) Andernfalls verschlüsseln wir das Paar mit den Positionen $(z_1, s_1), (z_2, s_2)$ durch $(z_2, s_1), (z_1, s_2)$, also zum Beispiel `WB` durch `SZ`.

(c) Doppelbuchstaben werden ausgelassen oder durch einen Blender wie `Q` oder `X` gebrochen.

An der Playfair-Chiffre besticht die kompakte Form, in der die Bigramm-Verschlüsselung dargestellt ist. Sie ist eine „Feldchiffre“ par excellence und als solche noch im 2. Weltkrieg eingesetzt worden. Kahn [**Ka1**], p. 592 berichtet, daß der Untergang des vom späteren US-Präsidenten J. F. Kennedy befehligten Patrouillenboots PT109 am 02.08.43 mittels einer Playfair-chiffrierten Nachricht mitgeteilt wurde. Gegen die Playfair-Chiffre kann man Bigramm-Statistik einsetzen und die Topologie des Verfahrens ausnutzen.

Transposition. Alle bisher besprochenen Chiffren beruhen auf Substitution: sie verarbeiten die Buchstaben des Klartexts in der Reihenfolge, in der sie im Text stehen. Demgegenüber permutieren *Transpositions-Chiffren* den Klartext. Das älteste bekannte militärische Kryptosystem, die *Skytale* des klassischen Griechenlands, beruht auf Transposition. Man wickelt dazu einen Lederstreifen auf einen Stab, beschreibt diesen zeilenweise und wickelt dann den Streifen ab. Zum Dechiffrieren hat man dann den Streifen nur wieder auf einen geeigneten Stab zu wickeln und kann den Klartext ablesen. Abbildung 4 (entnommen aus [**Si**], p. 23) erklärt diese Chiffre. Bei vielen Transpositions-Chiffren schreibt man den Text zeilenweise

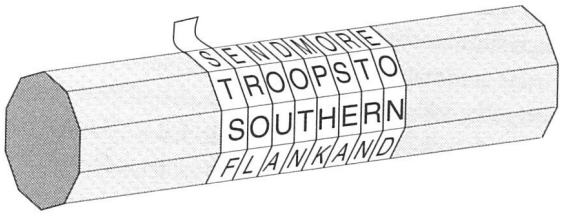


ABBILDUNG 4. Skytale

in eine Matrix und liest ihn dann nach Permutation der Spalten spaltenweise wieder aus. Transpositions-Chiffren werden üblicherweise mit Substitutions-Chiffren kombiniert. Zum Beispiel kann man bei Verwendung der Playfair-Chiffre den Text in zwei Zeilen schreiben und die Chiffre dann spaltenweise verwenden.

Ein Beispiel für die Kombination von Substitution und Transposition ist das ADFGVX-System (von *F. Nebel* erfunden), das vom deutschen Heer im Jahr 1918

an der Westfront verwendet wurde. Dabei wird zunächst eine monalphabetische Substitution eingesetzt, für die die 26 Buchstaben und 10 Ziffern in ein 6×6-Quadrat eingetragen werden. Zeilen und Spalten des Quadrats sind jeweils mit den sechs Buchstaben A, D, F, G, V und X markiert. Ein Beispiel:

	A	D	F	G	V	X
A	8	p	3	d	1	n
D	I	t	4	o	a	h
F	7	k	b	c	5	z
G	j	u	6	w	g	m
V	x	s	v	i	r	2
X	9	e	y	0	f	q

Im ersten Schritt der Chiffrierung ersetzt man jedes Klartextzeichen durch das Buchstabenpaar, das seine Position im Quadrat kennzeichnet.

Klartext	m	u	n	i	t	i	o	n	v	e	r	s	c	h	o	s	s	e	n
Substitution	GX	GD	AX	VG	DD	VG	DG	AX	VF	XD	VV	VD	FG	DX	DG	VD	VD	XD	AX

Im zweiten Schritt wird eine Transposition angewandt, die durch ein Schlüsselwort gesteuert wird. In unserem Beispiel wählen wir ein Schlüsselwort der Länge 4 (im kaiserlichen Heer hatte es die Länge $12 - -23$). Der Text wird zeilenweise in eine Tabelle der Breite des Schlüsselworts eingetragen. Dann werden die Spalten so permutiert, daß die Buchstaben des Schlüsselworts dabei in aufsteigende Reihenfolge geraten und der Geheimtext zeilenweise ausgelesen.

L	O	R	E
G	X	G	D
A	X	V	G
D	D	V	G
D	G	A	X
V	F	X	D
V	V	V	D
F	G	D	X
D	G	V	D
V	D	X	D
A	X	X	X

E	L	O	R
D	G	X	G
G	A	X	V
G	D	D	V
X	D	G	A
D	V	F	X
D	V	V	V
X	F	G	D
D	D	G	V
D	V	D	X
X	A	X	X

Der Geheimtext ist dann

DGXGGAXVGDDVXDGADVFXDVVVXFGDDDGVDVDXXAXX

Der Geheimtext wird im Morsecode gefunkt. Dies erklärt die Wahl der Buchstaben ADFGVX: ihre Morsezeichen lassen sich gut unterscheiden. Das ADFGVX-System wurde von den Franzosen gebrochen. Das Brechen der Chiffrierung dauerte mitunter aber mehrere Tage.

Transpositions-Chiffren ließen sich vor dem Computer-Zeitalter maschinell kaum realisieren, weil vor Anwendung der Transposition die Zeichen eines zu chiffrierenden Blocks gespeichert werden müssen.

Hill-Chiffren. Caesar-Addition und Transposition lassen sich gemeinsam verallgemeinern zu Hill-Chiffren, vorgeschlagen von L. S. Hill im Jahr 1929. Das sind injektive affin-lineare Abbildungen

$$\varphi : \mathbb{Z}_{26}^k \to \mathbb{Z}_{26}^n, \quad \varphi(v) = vA + b,$$

wobei $b \in \mathbb{Z}_{26}^n$ und A eine $k \times n$-Matrix über $\mathbb{Z}_{26}$ ist, die eine injektive Abbildung vermittelt. Injektivität ist äquivalent dazu, daß die Matrix A sowohl modulo 2 als auch modulo 13 den Rang k hat. (Hill-Chiffren kann man natürlich auch für andere Alphabete definieren.) Die Hill-Chiffre läßt sich bei hinreichender Länge durch Geheimtext-Angriffe kaum brechen. Dennoch ist sie für moderne Kryptosysteme ungeeignet, weil sie anfällig gegen Klartext-Angriffe ist.

Die Enigma. Es ist klar, daß man zur Anwendung von Hill-Chiffren maschinelle Hilfe benötigt. Vor dem zweiten Weltkrieg waren solche Maschinen jedoch schwer zu realisieren. Statt dessen benutzte man Rotor-Maschinen, deren berühmteste die *Enigma* der deutschen Wehrmacht war. Diese Maschine wurde ab 1918 von *A. Scherbius* entwickelt und in vielen Versionen kommerziell und militärisch eingesetzt. (Die Abbildungen zur Enigma sind [**Ba**] und [**Ka3**] entnommen.)

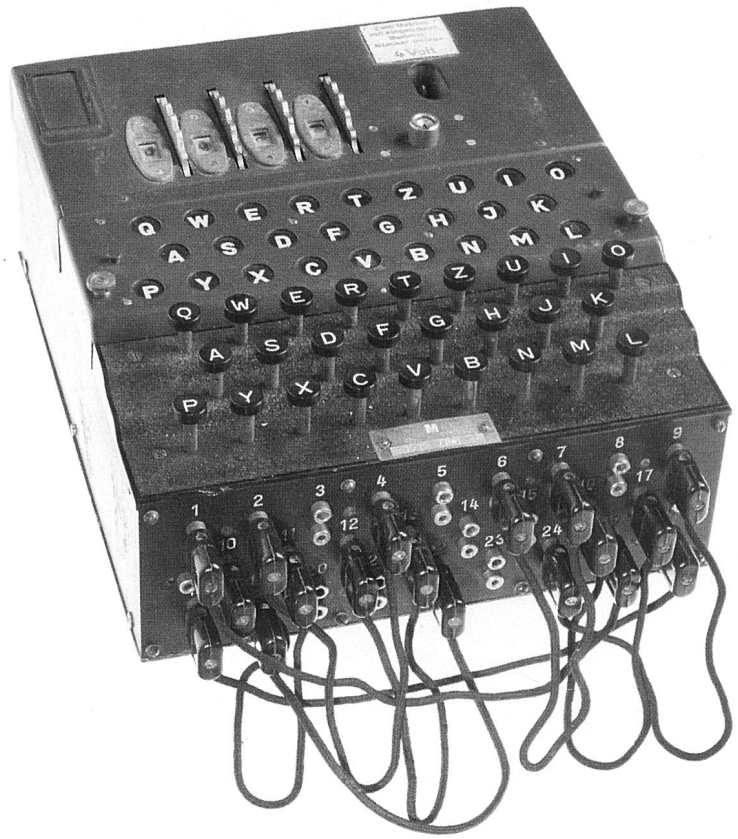


ABBILDUNG 5. Enigma mit 4 Rotoren

Ein *Rotor* (oder Walze) R ist dabei eine Scheibe, die auf beiden Seiten elektrische Kontakte trägt. Die Verbindung der Kontakte von der einen zur anderen Seite folgt dabei einer Permutation π_R des Alphabets, die bei der Konstruktion des Rotors festgelegt wird. Auf einer Achse werden nun mehrere Rotoren hinter

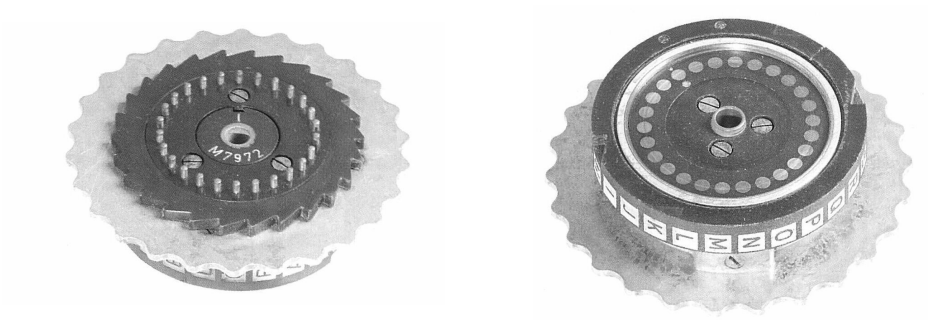


ABBILDUNG 6. Rotor der Enigma

einander angeordnet. Bei Eingabe eines Buchstabens durch Drücken einer Taste durchläuft der Strom die durch die Rotorstellung festgelegte Leitung. Der erste

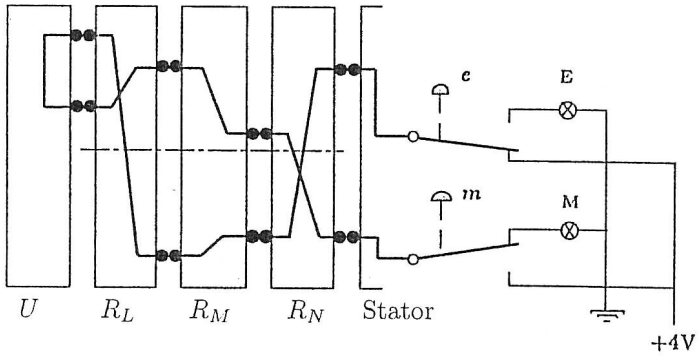


ABBILDUNG 7. Schematischer Stromlaufplan

Rotor übernimmt dabei den Strom vom Eingabe-Stator. Chiffriert wird das eingegebene Zeichen durch den Buchstaben, bei dem der Strom den letzten Rotor zum Ausgabe-Stator hin verläßt. Soweit ist dies nur eine monalphabetische Substitution. Aber nach (oder vor) jedem Zeichen wird der erste Rotor einen Schritt gedreht, und wenn eine bestimmte Stellung erreicht ist, schaltet der erste Rotor den zweiten um einen Schritt fort usw. Nachdem auch der letzte Rotor eine volle Umdrehung ausgeführt hat und der erste Rotor sich wieder in der Ausgangsstellung befindet, ist die ganze Maschine in der Ausgangsstellung. (Wir vereinfachen hier etwas; siehe [**DK**] zur „Anomalie“ der Rotorfortschaltung bei der Enigma.) Bei einem Alphabet mit 26 Zeichen ergibt sich mit der Rotor-Zahl n die Periode 26^n. Wenn ein Rotor in

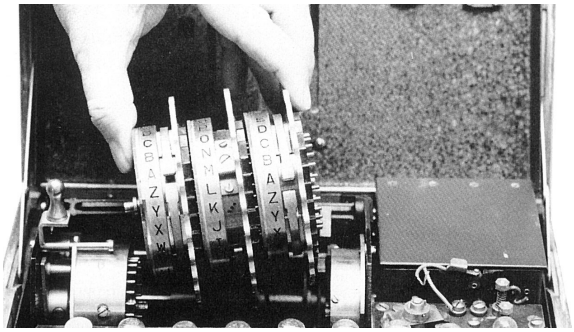


ABBILDUNG 8. Rotorsatz einer Enigma

der Ausgangsstellung die Permutation π_R bewirkt, so führt er nach einer Drehung von i Schritten die Permutation

$$\zeta^i \pi_R \zeta^{-i}$$

aus, wobei ζ eine zyklische Verschiebung um eine Stelle ist, abhängig von der Drehrichtung.

Bei der Enigma kamen zunächst 3, später auch 4 Walzen zum Einsatz. Statt des Ausgabestators wurde ein Reflektor (oder Umkehrwalze) verwendet, der wiederum eine Permutation des Alphabets darstellt. Der Reflektor leitete den Strom zurück in den letzten Rotor, von wo aus die Rotoren in umgekehrter Reihenfolge durchlaufen wurden. Das Chiffre-Zeichen wurde durch eine Lampe angezeigt (siehe den schematischen Stromlaufplan in Abbildung 7). Die Rotoren konnten aus einem Satz von (maximal) 8 ausgewählt werden und in beliebiger Reihenfolge in das Gerät eingesetzt werden. Auch der Reflektor war auswechselbar.

Die Rotoren der Enigma verfügten zusätzlich noch über einen Einstellring, der geben den Rotorkörper verdreht und dann fixiert wurde. Dieser beeinflußte letzten Endes nur die Rotorfortschaltung, weil die Nut zur Fortschaltung an ihm angebracht war.

Als zusätzliche Komplikation hatte die Enigma ein Steckerbrett, das noch eine Permutation σ des Alphabets durchführte. Diese war allerdings (überflüssigerweise) involutorisch: es galt $\sigma^2 = \mathrm{id}$. Zunächst wurden die Buchstaben von 6 ausgewählten Paaren gegeneinander ausgetauscht, später die von 10.

Zum Einsatz der Enigma wurde eine Grundstellung festgelegt (Steckerbrett, Walzenauswahl, Ringstellungen, Ausgangsstellung der Walzen). Diese wechselte nach einem festen Rhythmus, ab 1936 täglich, ab Mitte 1942 alle 8 Stunden. Jeder Funkspruch begann dann mit der Mitteilung des Spruchschlüssels. Er bestand aus drei Buchstaben und legte die Walzenstellung für die dann folgende Nachricht fest. Auf diese Weise wurde vermieden, daß zuviel Text mit der gleichen Chiffre verschlüsselt und diese damit der Häufigkeitsanalyse zugänglich wurde. (Der Modus der Schlüsselvereinbarung hat mehrfach gewechselt.)

17. Gültiger Tagesschlüssel:
(Ausschnitt aus der für die Verschlüsselung des Klartextes in Betracht kommenden Schlüsseltafel, z. B. ».....« Maschinenschlüssel für Monat Mai«)

Datum	Walzenlage	Ringstellung	Grundstellung
4.	I III II	16 11 13	01 12 22

Steckerverbindung	Kenngruppen-Einsatzstelle ..... Gruppe	Kenngruppen
CO DI FR HU JW LS TX	2	adq nuz opw vxz

ABBILDUNG 9. Dienstvorschrift für die Enigma

Die Enigma war eine ausgezeichnete Maschine. Während des Krieges konnten die Engländer (nach polnischen Vorarbeiten) die Enigma-Chiffrierung dennoch brechen. Der Einbruch in das U-Boot-Netz „Triton“ gelang nachhaltig aber erst nach der Erbeutung von Schlüsselunterlagen vom aufgebrachten U-Boot U 110 im Mai 1941 (siehe [**Ka3**]). Ein führender Kopf der Codebrecher war *A. Turing*. Für die Analyse der Enigma-Chiffrierung wurden die ersten elektronischen Rechengeräte gebaut.

Zum Brechen der Enigma-Chiffrierung haben aber auch wesentlich Fehler beim Einsatz der Maschine beigetragen, von denen wir hier nur einen nennen wollen: Bis Mai 1940 wurde der Spruchschlüssel zweimal nacheinander gesendet, und zwar mit zwei verschiedenen Chiffren, da sich die Maschine ja laufend weiterdreht. Dies ist eine sogenannte *Geheimtext-Geheimtext-Kompromittierung*, wenn auch eine extrem kurze. (Senden des gleichen Klartexts mit verschiedenen Schlüsseln gilt als ein schwerer Chiffrierfehler.)

Die Enigma hatte allerdings auch einen Schwachpunkt in ihrer Konstruktion. Die Verwendung der Umkehrwalze schloß aus, daß ein Buchstabe durch sich selbst verschlüsselt wurde. Ist nämlich ρ die Permutation, die durch die Rotorstellung bewirkt wird, σ die des Steckerbretts und τ die des Reflektors, so wird insgesamt die Permutation

$$\alpha = \sigma^{-1}\rho^{-1}\tau\rho\sigma$$

ausgeführt. Diese hat keinen Fixpunkt, weil τ keinen hat. Da die Reflektor-Permutation τ involutorisch war, ist auch α involutorisch. Dies hat den Vorteil, daß Chiffrierung und Dechiffrierung mit der gleichen Einstellung der Maschine durchgeführt werden konnten.

Man findet im Internet zahlreiche Simulationen der Enigma. Sehr schön gestaltet ist die Simulation der 3-Rotor-Marine-Enigma, die man über [**Eni**] beziehen kann.

Kryptanalyse mittels Invarianz von Konjugationsklassen. Wir wollen kurz erläutern, wie es den polnischen Kryptanalytikern, allen voran *M. Rejewski*, gelang, die doppelte Übertragung des Spruchschlüssels auszunutzen. Rejewski hatte die geniale Idee, nicht nur die hergebrachten Methoden der Häufigkeitsanalyse und der wahrscheinlichen Wörter anzuwenden, sondern auch die Invarianz der Zykelzerlegung von Permutationen unter Konjugation. Wir gehen im folgenden davon aus, daß sich während der Verschlüsselung der ersten 6 Zeichen nur der schnelle Rotor dreht. Dann hängt deren Verschlüsselung nicht von der Ringstellung, sondern nur von der Grundstellung der Rotorkörper und vom Steckerbrett ab. Für deren Wahl gibt es dann die gigantische Zahl von

$$6 \cdot 26^3 \cdot \binom{26}{2}\binom{24}{2}\cdots\binom{16}{2} \approx 7.6 \cdot 10^{18}$$

Möglichkeiten, wenn das Steckerbrett 6 Buchstabenpaare vertauscht und die (zunächst nur benutzten) 3 Rotoren in 6 verschiedenen Anordnungen eingesetzt werden können. Die Einstellung des Steckerbretts wirkt sich aber, wie wir gerade gesehen haben, nur als Konjugation der durch die Rotoren gegebenen Permutation des Alphabets aus.

Wir betrachten die Verschlüsselungen π_1 und π_4 des ersten und des vierten Buchstabens. Der erste und der vierte Klartextbuchstabe stimmen überein. Wir nennen ihn x. Die Verschlüsselungen $y = \pi_1(x)$ und $z = \pi_4(x)$ sind im abgehörten Funkspruch enthalten, und es gilt

$$z = (\pi_4 \circ \pi_1^{-1})(y) = (\pi_4 \circ \pi_1)(y).$$

(Beachte,daß die Permutationen π_i involutorisch sind.) Liegen hinreichend viele Funksprüche vor, so kann man also $\alpha_1 = \pi_4 \circ \pi_1$ vollständig konstruieren. Gleiches gilt für $\alpha_2 = \pi_5 \circ \pi_2$ und $\alpha_3 = \pi_6 \circ \pi_3$. Die Konjugationsklasse ist durch den Typ der Zykelzerlegung eindeutig bestimmt und sofort zu ermitteln.

Wie wir gerade gesehen haben, sind die Konjugationsklassen $\overline{\alpha}_i$ unabhängig vom Steckerbrett. Für $(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3)$ gibt es höchstens

$$6 \cdot 26^3 = 105456$$

Möglichkeiten. Die polnischen Kryptanalytiker haben daher einen Atlas der Konjugationsklassen der drei Permutationen in Abhängigkeit von der Grundstellung der Rotorkörper angelegt. Sobald also $\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3$ bekannt waren, konnte die Grundstellung im Atlas nachgeschlagen werden. Die zunächst unüberwindlich erscheinende Komplexität der Enigma war somit entscheidend vereinfacht.

Die Zykelzerlegungen der α_j erfüllen noch eine gewisse Nebenbedingung; siehe Aufgabe 12.9. Diese hat geholfen, die π_i aus den α_j zu bestimmen. Dafür und für weitere Einzelheiten verweisen wir auf [**Ba**] oder [**DK**].

Negative Mustersuche. Nachdem die Spruchschlüssel-Verdoppelung abgeschafft war, war dieser (nach langen Vorarbeiten) bequeme Einstieg verschüttet. Einen Ersatz bot dafür die Lokalisierung wahrscheinlicher Wörter, *cribs* genannt, im Geheimtext. Wie bereits erwähnt, konnte die Enigma einen Buchstaben nicht durch sich selbst verschlüsseln. Viele Permutations-Chiffren schließen Fixpunkte aus, was auf den ersten Blick die Sicherheit zu erhöhen scheint. Aber es handelt sich um eine *complication illusoire*, denn sie erlaubt eine *negative Mustersuche* mit wahrscheinlichen Wörtern. Tabelle 2 ist ein [**Ba**], p. 253 entnommenes Beispiel. Es spielt auf einen Vorfall im 2. Weltkrieg an, bei dem die englische Luftwaffe eine Leuchttonne in der Fahrrinne zum Hafen Calais nur deshalb versenkte, um eine entsprechende Meldung der deutschen Seite herauszufordern, in der unweigerlich das Wort „Leuchttonne“ vorkommen mußte. Großbuchstaben zeigen eine Kollision an; nur die mit • markierten Positionen sind möglich.

Weitere Rotormaschinen. Eine andere berühmte Rotor-Maschine war die *C-36* von *B. Hagelin*, die unter dem Namen *M-209* von der U.S. Armee bis in die 50er Jahre eingesetzt wurde. Im Gegensatz zur Enigma war sie eine rein mechanische Konstruktion. Diese Maschine und ihre Kryptanalyse werden detailliert in [**BP**] und [**Sa**] behandelt. In [**DK**] werden neben der Enigma und der M-209 weitere Rotormaschinen analysiert.

Rotor-Chiffrierenden sind mit der Enigma und der M-209 nicht ausgestorben: Die Funktion `crypt` des Unix-Betriebssystems verschlüsselt Files mit einer „Rotor-Maschine“, die einen Rotor des Umfangs 256 und einen Reflektor hat. Rotor- und Reflektor-Permutation werden aus einem Schlüsselwort erzeugt.

Geschichte der Kryptographie. Ein ausgezeichnetes Buch zur Geschichte der Kryptographie ist [**Si**]. Eine umfassendere Darstellung findet man in [**Ka1**]. Auch das eher mathematisch orientierte Buch [**Ba**] enthält viele historische Details, insbesondere zur Enigma und zu den Techniken der Kryptanalyse.

Kryptographie spielt auch in vielen Werken der Literatur eine wesentliche Rolle. Eine ausführliche Übersicht findet man in [**Ka1**], Kap. 21. Hier sollen nur drei Beispiele genannt werden. In „Der Goldkäfer“ von *E. A. Poe* dechiffriert der Held William Legrand eine geheime Mitteilung von Captain Kidd mit klassischer Häufigkeitsanalyse. Sherlock Holmes, der von *A. C. Doyle* geschaffene Prototyp aller Privatdetektive, hat in seiner Laufbahn mehrfach Geheimschriften zu entziffern, unter anderem in „Die tanzenden Männchen“; siehe Abbildung 10. Schließlich knackt Lord Peter Wimsey in *D. Sayers'* „Mein Hobby: Mord“ auf elegante Weise eine Playfair-Chiffre. Im Roman „Enigma“ von *R. Harris* erfährt man zwar

```
YOAQUTHNCHWSYTIWHTOJQMTCFKUSLZVSMFNGTDUQNYAVH
erloscHenistleucHttonne
 erloscheniStleuchttonne
  erloschenisTleuchttonne
   erlosCHenistleuchttonne
    erloschenistleuchtTonne
     erloschenIstleuchTtonne
      erloschenisTleuchttonne
       erloSchenistleuchttonne
        erloschenistleuChttonne
        •erloschenistleuchttonne
          erloscHenistleuchttonne
           erloschenisTleuchttonne
            erloschenistleUchttonne
             erloschenistleuchttonNe
              erloschenistleuchttoNne
               erlOschenistleuchttonne
                erloschenistLeuchttonne
                 erloscheniStleuchttonne
                  erloschenistleuchtTonne
                   erloschenistleuchTtonNe
                    erloschenistleuchttoNne
                     erloscheniStleuchttonne
                     •erloschenistleuchttonne
```

TABELLE 2. Negative Mustersuche

nichts über die Enigma, kann aber die Geschehnisse in Bletchley Park, dem Ort, an dem die sie gebrochen wurde, auf spannende Weise miterleben.

ABBILDUNG 10. Aus *Die tanzenden Männchen* von A. C. Doyle

Übungen

12.1. (a) Chiffriere den folgenden Text mittels der Caesar-Verschiebung zum Kennbuchstaben `H`:

```
alea iacta est
```

(b) Dechiffriere den folgenden Caesar-chiffrierten Text:

```
IQDPU QEZUO TFWMZ ZUEFW QUZWD KBFMZ MXKFU WQD
```

Die 5er-Gruppen dienen der Erleichterung des Lesens.

12.2. (a) Chiffriere den folgenden Text mit monalphabetischer Substitution zum Kennwort KRYPTOGRAPHIE:

```
codie rungs theorie undkr yptog raphie sindt eiled ermat hemat ik
```

(b) Dechiffriere den folgenden, mit einer monalphabetischen Substitutions-Chiffre verschlüsselten Text in deutscher Sprache:

```
JVPSTQYBZOMSCCRZFYLBMKXNVSCDRQLBSQOSTFGRZKVRZDMZOISCPXYEVCQRSASTFVGSTZKMSZLBSZJ
YTSRZSTFSIVSBTMZAQXTYFSFSRSRZSKOTRPPCRARQPSZORSJYTDSRLBSZNMSTORSVLBPFSASAZMZASZ
OSQONFXYEVCQJSTBSRQQSZSRASZPCRLBXVLESZOSLMXEZMSCCSTVFSTFVGSTPTVRZSTECVMQPYXXKYS
CCSTQYTAPQRLBJYTOSTZVLBBYCXVTPRSOSTDISRPSZTMZOSFSRKDISRPCRARQPSZJNCFYLBMKMKORSE
YSTXSTCRLBSJSTNVQQMZAQSRZSQXSTQYZVCQOSTVEEMRQPCSSTSFSZQYIRSFVGSTZPTVRZSTYPPKVTB
RPDNSCOOSTFSRKJNCYQZVFTMSLEQSRZSZQMSOVKSTREVQPMTKARYJVZSSCFSTMZOLCVMORYXRDDVTYQ
LBYZSZOMSTNPSJSTDRLBPSPPYXXROVBSTASASZQSRZSZSBSKVCRASZJSTSRZMZOQYBZORZYIYBCVMNO
RSPYTUVSASTYCRJSTZSMJRCCSMZOMCNERTQPSZQYIRSGRCORTVGFVQPMSTEQLBCRSQQCRLBQPSBSZZV
LBOSKXYEVCBVSTPSPSQPJYTISRBZVLBPSZZYLBDISRFMZOSQCRAVQXRSCPVASVMNOSKXTYATVKK
```

12.3. Dechiffriere den folgenden Geheimtext aus E. A. Poes „Goldkäfer“. Die Unterschrift *Captain Kidd* legt die Vermutung nahe, daß es sich um einen Text englischer Sprache handelt.

```
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶
60))85;1‡(;:‡*8†83(88)5*†;46(;88*96*
?;8)*‡(;485);5*†2:*‡(;4956*2(5*−4)8¶
8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
1;48†85;4)485†528806*81(‡9;48;(88;4(
‡?34;48)4‡;161;:188;‡?;
```

12.4. (a) Chiffriere den folgenden Text mit der Vigenère-Chiffre zum Kennwort ALGEBRA

```
diere ellen zahle nbild enein enkoe rper
```

(b) Dechiffriere den folgenden Text deutscher Sprache. Er ist Vigenère-chiffriert.

```
RIKHRNIEDVUBGOHNKFHGSHLLOLMUJDZMEVNTMEEHSUTGYZVYOGAZGSHJHWEOOIJRNSAMPKIIWTAVYWR
RNSEXHOMHKRZYERIEWAPLHYJVQZHNSQESKPKPLKCTRMVGBLQWLXDBSPKZVJGRISWALLYAUAZMEJRRXX
ZVTYFGYCMEHRRQEDVFHNQTCELGIOWFPVJNRGYNELGGEWRFQVYTGYYMVUEUJRPQXCWULIHZSQAXOLOJZ
GTUDWTVRLFROGFMVGBOWTVRNJREHSNSKUZVJIZGJFPQECWBDZMXCGTXGSDMEZWUYIDCTLNYYGCUDWCJ
RUSEEIPPQSUKULYIIDLRJRLHFHTGYBLLBQEMEMHOHAEOOKVXNUQXHHEYJCBNJISHDJRZFIUMUCZVQKR
IKPFQHQWKSWIZRRNLSEWTWZCSFWVWTESXWLDBGJUZLZBGAJKWLDBWVDVWESHEXDFEFEDCLAJVBVSYWP
LUYAPPBIEKBCMIYLHHGTPZVVBQIJLFQEYAJYZWRIQAIMPVPQAGZJELGRISIXDOXWTLIXFDSSHLWDCVW
TAVYTVTOYXDFIUDMZZRKSETFMYPFHLMVGPVURGJSCJFWDQVIIPWFTFYQVTWZYLDRXSXORQPQNUPDLIE
EBGEJVODTNAGYZWQORHQXHLFAGVANGYOYKXWALLYRWKZRMWRTSEXHTCKEOZRYORNLIMDVWZUJCAVWAE
SZZQEYFGUZMESXAQMQRSHAUJCIQIRCMXPUJHYCFGIJDRAWWTPKUZTYPRUNJENHFWAYFFHWWVHMTBITO
TYAPMVGYDHTEMRHUCWTJCIEGVNIATHTCWUHBX
```

12.5. Konstruiere eine Mini-Enigma mit einem Rotor, der die Buchstaben zwischen Eingangs-Stator und Ausgangs-Stator permutiert. Die Verdrahtung des Rotors ist durch folgende Permutation gegeben:

```
KMUIO LGDWY NRSZT FACBE JVXHQ P
```

Dies ist so zu interpretieren: In der Stellung 0 wird das eingegebene Zeichen `A` als `K` ausgegeben, `B` als `M` usw. Die Kontakte von Ein-und Ausgabe-Stator sind den Buchstaben in der Folge des Alphabets zyklisch so zugeordnet, daß bei direkter Verbindung sich gegenüber liegender Kontakte die identische Permutation entsteht.

Nach jedem eingegebenen Zeichen wird der Rotor von der Stellung i in die Stellung $i+1$ (modulo 26) so gedreht, daß der in Stellung i mit dem `A` des Eingangs-Stators verbundene Kontakt des Rotors in Stellung $i+1$ mit `B` verbunden wird.

Mit der Ausgangsstellung 5 verschlüssele man den Text

```
diere ellen zahle nbild enein enkoe rper
```

12.6. Für Texte S,T gleicher Länge m über dem gleichen Alphabet setzen wir

$$\varkappa(S,T) = \frac{\iota(S,T)}{m}$$

wobei $\iota(S,T)$ die Positionen zählt, an denen S und T übereinstimmen.

Die Quelle Q produziere die Buchstaben $b_1,\dots,b_n$ mit den Wahrscheinlichkeiten $p_1,\dots,p_n$. Wir setzen im folgenden voraus, daß die Texte S,T aus Q stammen und an jeder Position voneinander unabhängig sind.

(a) Zeige: Der Erwartungswert von $\varkappa(S,T)$ ist $\varkappa_0 = \sum_{i=1}^{n} p_i^2$.

Welcher Wert für $\varkappa(S,T)$ ergibt sich für Buchstaben-Häufigkeiten gemäß Tabelle 1 (i) im Deutschen, (ii) im Englischen?

(b) Wir unterwerfen nun den Text T einer monalphabetischen Substitution τ. Zeige, daß $\varkappa(T,\tau(T))$ den Erwartungswert

$$\varkappa_\tau = \sum_{i=1}^{n} p_i p_{\tau^{-1}(i)}$$

hat. Welche Werte ergeben sich gemäß Tabelle 1 für die 25 nichttrivialen Cäsar-Verschiebungen?

(c) Zeige $\varkappa_\tau \le \varkappa_0$ für alle $\tau \in S_n$. (Interpretiere $\varkappa_\tau$ als Skalarprodukt und wende die Cauchy-Schwarzsche Ungleichung an.)

(d) Zeige: Der Mittelwert der $\varkappa_\tau$ über alle $\tau \in S_n$ ist

$$\overline{\varkappa} = \frac{1}{n!}\sum_{\tau} \varkappa_\tau = \frac{1}{n}.$$

12.7. Man kann die Funktion $\varkappa$ satt durch den Vergleich von Einzelbuchstaben auch den Vergleich von Bigrammen (oder allgemeiner n-Grammen) definieren. Experimentiere mit Bigramm- und Trigramm-$\varkappa$.

12.8. Eine Hill-Chiffre $\varphi : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$,

$$\varphi(x) = xA + b,$$

soll durch einen Angriff mit gewähltem Klartext gebrochen werden. (Wir stellen uns dies so vor: Die Berechnung von $\varphi(x)$ bleibt uns verborgen. Wir können aber x in eine Chiffriermaschine eingeben und erhalten die Ausgabe $\varphi(x)$.) Man bestimme eine kürzestmögliche Nachricht $x_1 \cdots x_N$ ($x_i \in \mathbb{Z}_q^n$), so daß man A und b aus $x_1 \cdots x_N$ und $\varphi(x_1) \cdots \varphi(x_N)$ bestimmen kann.

Hinweis: Wie wählt man x, um b zu bestimmen? Usw.

12.9. Die Permutationen π und $\tau \in S_n$ seien involutorisch und ohne Fixpunkte (speziell ist n gerade). Zeige, daß in der Zykelzerlegung von $\pi \circ \tau$ jede Zykellänge mit gerader Vielfachheit auftritt. (Es macht Sinn, erst einmal mit kleinen Beispielen zu experimentieren.)

ABSCHNITT 13

Perfekte Sicherheit

Die Schemata der Informationsübertragung, die in der Codierungstheorie und der Kryptographie auftreten, sind sich durchaus ähnlich, vor allem dann, wenn wir den unbefugten Mithörer eines Geheimtextes und den Empfänger einer codierten Nachricht vergleichen. Beide sind im Ungewissen über die gesendete Nachricht N und müssen diese aus der empfangenen Nachricht E erschließen.

Bei diesem Vergleich zeigt sich, daß Codierungstheorie und Kryptographie absolut konträre Ziele verfolgen: Codierung soll sicherstellen, daß E *möglichst viel* Information über N enthält, während Chiffrierung dafür sorgen soll, daß E *möglichst wenig* Information über N preisgibt. Wir haben in Abschnitt 2 die Begriffe Entropie und Information eingeführt. Sie lassen sich sehr gut verwenden, um die informationstheoretischen Aspekte der Kryptographie zu erfassen.

Wir betrachten ein Kryptosystem mit endlichem Klartextraum $\mathscr{P}$ und endlichem Schlüsselraum $\mathscr{K}$. Wir setzen im folgenden voraus, daß jeder Klartext $P \in \mathscr{P}$ mit Wahrscheinlichkeit p_P gesendet wird und dazu (unabhängig von P) ein Schlüssel $K \in \mathscr{K}$ mit Wahrscheinlichkeit q_K ausgewählt wird. Die Wahrscheinlichkeit für das Auftreten des Paares (P,K) ist also $p_P q_K$. Die Chiffrierung E_K ordnet P den Geheimtext C zu. Die Wahrscheinlichkeit für das Auftreten von C ist dann

$$r(C) = \sum_{P \in \mathscr{P}} \sum_{K \in \mathscr{K}, C = E_K(P)} p_P q_K.$$

Die bedingte Wahrscheinlichkeit für einen Klartext P bei gegebenem Geheimtext C ist

$$r(P \mid C) = \frac{1}{r(C)} \sum_{K \in \mathscr{K}, E_K(P) = C} p_P q_K.$$

Damit können wir die *Entropie von $\mathscr{P}$ unter C* definieren durch

$$H(\mathscr{P} \mid C) = -\sum_{P \in \mathscr{P}} r(P \mid C) \log r(P \mid C)$$

und die *mittlere Entropie von $\mathscr{P}$ unter $\mathscr{C}$* ist dann

$$H(\mathscr{P} \mid \mathscr{C}) = \sum_{C \in \mathscr{C}} r(C) H(\mathscr{P} \mid C).$$

Sie mißt die mittlere Ungewißheit über den Klartext bei bekanntem Geheimtext. Schließlich ist

$$I(\mathscr{P},\mathscr{C}) = H(\mathscr{P}) - H(\mathscr{P} \mid \mathscr{C})$$

die *Information von $\mathscr{C}$ über $\mathscr{P}$*.

Als einfaches Beispiel betrachten wir die Vigenère-Chiffre der Länge d des Alphabets $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, mit der wir Klartexte der Länge kd chiffrieren wollen. Auf Klartext- und Schlüsselraum herrsche Gleichverteilung. Zu jedem Geheimtext C gehören dann q^d gleichwahrscheinliche Klartexte. Daraus ergibt sich

$$H(\mathscr{P} \mid \mathscr{C}) = \log q^d = d \log q.$$

Hingegen ist $H(\mathscr{P}) = \log q^{kd} = kd \log q$. Es folgt

$$I(\mathscr{P},\mathscr{C}) = (k-1)d \log q.$$

Definition. Ein Kryptosystem $(\mathscr{P},\mathscr{C},\mathscr{K},\mathscr{E},\mathscr{D})$ (mit endlichen Mengen $\mathscr{P}$, $\mathscr{C}$, $\mathscr{K}$) heißt *perfekt sicher*, wenn $I(\mathscr{P},\mathscr{C}) = 0$ ist.

Wie so vieles in der Informationstheorie stammt auch diese Definition von Shannon.

Satz 13.1. *Ein Kryptosystem ist genau dann perfekt sicher, wenn $p_P = r(P \mid C)$ für alle Klartexte P und Geheimtexte C gilt, mit anderen Worten, wenn die Wahrscheinlichkeitsverteilungen auf $\mathscr{P}$ und $\mathscr{C}$ unabhängig sind.*

Dies ergibt sich unmittelbar aus Satz 2.7. Wir ziehen noch einige Folgerungen.

Satz 13.2. *In einem perfekt sicheren Kryptosystem gibt es mindestens so viele Schlüssel wie Klartexte.*

Beweis. Sei $N = |\mathscr{P}|$. Wir betrachten einen Klartext P und einen Schlüssel S.

Sei $C = E_S(P)$. Dann ist $r(P \mid C) > 0$, und nach Satz 13.1 muß $r(P' \mid C) = r(P \mid C) > 0$ für die anderen $N-1$ Klartexte P' sein. Da der Geheimtext C alle Klartexte repräsentieren kann, brauchen wir mindestens N Schlüssel. □

Satz 13.3. *Ein Kryptosystem erfülle folgende Bedingungen:*

(a) $|\mathscr{P}| = |\mathscr{K}|$

(b) $q_S = |\mathscr{K}|^{-1}$ *für alle* $K \in \mathscr{K}$.

Dann ist es perfekt sicher.

Dies ist nun offensichtlich: Da jeder Schlüssel mit gleicher Wahrscheinlichkeit auftritt, ist $p(P \mid C) = p(P)$ für alle Klartexte P.

Die vorangegangenen Sätze zeigen, daß sich perfekt sichere Systeme nur mit extrem hohem Aufwand realisieren lassen.

One-Time-Pad. Eine mögliche binäre Realisierung ist der One-Time-Pad (von *G. Vernam* ca. 1920 vorgeschlagen). Um einen Klartext $P = k_1 \dots k_n \in \{0,1\}^n$ zu verschlüsseln, wählt man $S = s_1 \dots s_n \in \{0,1\}^n$ und setzt

$$q_i = s_i \oplus k_i, \quad i = 1, \dots, n.$$

Dabei steht das Zeichen $\oplus$ hier (wie in der Kryptographie allgemein üblich) für die Addition in $\mathbb{F}_2$, die in der Schaltalgebra XOR (exklusives oder) genannt wird. Der One-Time-Pad muß so verwendet werden, wie es sein Name vorgibt: nur ein einziges Mal.

Sender und Empfänger müssen also vorab so viele Schlüssel der Länge n erzeugen, wie Nachrichten der Länge n gesandt werden sollen, und diese Schlüssel austauschen. Für einen Einsatz als Public-Key-System ist der One-Time-Pad völlig unbrauchbar. Gerüchten zufolge ist der „heiße Draht“ zwischen Washington und Moskau als One-Time-Pad realisiert. Der sowjetische Spion R. I. Abel hat Nachrichten mit (alphabetischen) One-Time-Pads verschlüsselt. Sie werden niemals zu entschlüsseln sein (siehe dazu [**Ka2**]).

Bereits die zufällige Erzeugung der Schlüssel ist ein Problem. Echte Zufallszahlen-Generatoren (sofern es denn Zufall wirklich gibt) sind nur mit hohem Aufwand zu realisieren. Man kann den radioaktiven Zerfall nutzen, um einen Zufallszahlen-Generator zu bauen. Einfacher ist es zum Beispiel, die Zeit zwischen den Anschlägen einer Computer-Tastatur zu messen. Allerdings wird man diesem Prozeß weniger „Zufälligkeit“ zubilligen als dem radioaktiven Zerfall. Sehr einfach lassen sich Pseudo-Zufallszahlen erzeugen, für die man häufig eine Rekursion der folgenden Art benutzt:

$$x_{i+1} = ax_i + b \mod m,$$

wobei $\mathrm{ggT}(a,m) = 1$. Der Startwert x_0 bestimmt die erzeugte Folge. Wenigstens ihn muß man „zufällig“ wählen.

Übungen

13.4. Sei A ein Alphabet. Die Texte der Länge N über dem Alphabet A werden in zwei Klassen eingeteilt, die Klasse $\mathscr{Z}$ der zulässigen Texte und die Klasse $\mathscr{U}$ der unzulässigen. (Dies kommt einer natürlichen Sprache schon wesentlich näher als die Annahme, alle Wörter über A gehörten zur Sprache.)

Wir nehmen nun an, daß es zu jedem Geheimtext G der Länge N höchstens eine Cäsar-Verschiebung π von A gibt, die den Text G aus einem zulässigen Text erzeugt.

(a) Wieviele Wörter gehören höchstens zu $\mathscr{Z}$?

(b) Welche Entropie ist für $\mathscr{Z}$ höchstens möglich? Welche Entropie pro Zeichen besitzt $\mathscr{Z}$ höchstens?

(c) Diskutiere den Fall $q = 26$, $N = 5$.

Das kleinstmögliche N, das man bei natürlichen Sprachen nur schätzen kann, nennt man die *Unizitätslänge* der Sprache für Cäsar-Verschiebung. Für eine präzise Definition für beliebige Kryptosysteme siehe [**BP**]. Nach Shannon heißen Kryptosysteme ohne Unizitätslänge *ideal sicher*.

ABSCHNITT 14

DES

Ein wichtiges, standardisiertes und immer noch sehr gutes Kryptosystem ist der *Data Encryption Standard* (DES). Man kann die Dokumente in der DES und andere standardisierte Verfahren beschrieben werden, über die Internet-Adresse [**NIST**] beziehen.

Feistel-Chiffre. DES beruht auf dem Prinzip der Feistel-Chiffre, die wir zunächst beschreiben wollen. Klartext- und Geheimtextalphabet ist $\{0,1\}^n$, n gerade. Die Chiffrierung erfolgt in *r Runden*, für die man *Rundenschlüssel* $K_1,\dots,K_r$ festgelegt hat. (Die Natur der Rundenschlüssel spielt zunächst keine Rolle). Sei $t = n/2$. Jeder Rundenschlüssel K_i bestimmt eine Abbildung, die *innere Blockchiffre*

$$f_{K_i} : \{0,1\}^t \to \{0,1\}^t,$$

Jedes Element $P = (b_1,\dots,b_n) \in \{0,1\}^n$ teilt sich in zwei Hälften $L,R \in \{0,1\}^t$,

$$L = (b_1,\dots,b_t), \quad R = (b_{t+1},\dots,b_n).$$

Wir schreiben dann auch $P = (L,R)$.

Für die Feistel-Chiffre startet man mit dem Klartext-Block $P_0 = (L_0,R_0)$ und bildet iterativ

$$P_i = (L_i,R_i) = (R_{i-1}, L_{i-1} \oplus f_{K_i}(R_{i-1})), \qquad i = 1,\dots,r. \tag{7}$$

(Dabei ist $\oplus$ wieder die Addition in $\mathbb{F}_2^t$.) Der Geheimtext ist dann

$$C = (R_r, L_r). \tag{8}$$

Man beachte die Vertauschung von L_r und R_r.

Die Dechiffrierung ist sehr einfach: Man verwendet das gleiche Verfahren, aber mit umgekehrter Schlüsselfolge. Um einzusehen, daß dabei der Klartext zurückgewonnen wird, spalten wir die Runde (7) in 2 Schritte auf:

$$\pi(L,R) = (R,L), \qquad \varphi_i(L,R) = (L, R \oplus f_{K_i}(L))$$

Dann ist

$$P_i = \varphi_i \circ \pi(P_{i-1})$$

Insgesamt erfolgt wegen der abschließenden Vertauschung (8) die Chiffrierung

$$P \mapsto \pi \circ \varphi_{16} \circ \pi \circ \cdots \circ \varphi_1 \circ \pi(P).$$

Da die Abbildungen π und φ_i involutorisch sind, ist

$$\left(\pi \circ \varphi_{16} \circ \pi \circ \cdots \circ \varphi_1 \circ \pi\right)^{-1} = \pi \circ \varphi_1 \circ \pi \circ \cdots \circ \varphi_{16} \circ \pi.$$

DES. Nun zu DES selbst. Dabei ist $n = 64$, $t = 32$. Auch der „Hauptschlüssel“ K ist ein Element von $\{0,1\}^{64}$. Aus ihm werden, wie später noch beschrieben, die Rundenschlüssel gebildet. Allerdings verwendet man nicht alle 2^{64} Schlüssel, sondern nur die, bei denen jedes der 8 Bytes ungerade Parität hat. Die effektive Schlüssellänge ist also 56 Bit, und vom Schlüssel $K = (k_1, \ldots, k_{64})$ werden nur die Bits an den Positionen $j \neq 8, 16, \ldots, 64$ benutzt.

Sei nun $P \in \{0,1\}^{64}$ der zu chiffrierende Klartext-Block. Er wird zunächst einer initialen bitweisen Permutation IP unterzogen, die in Tabelle 1 beschrieben ist. Die Tabelle ist zeilenweise zu lesen, und zwar so:

$$\mathrm{IP}(p_1 \ldots p_{64}) = (p_{58} p_{50} \ldots p_7).$$

Diese Permutation ist kryptographisch wirkungslos. Ihr einziger Sinn kann darin bestehen, daß sie die Hardware Implementation von DES zur Zeit von dessen Entwicklung (1974) einfacher gemacht hat. Auf $\mathrm{IP}(P)$ wird eine 16-Runden-Feistel-

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

TABELLE 1. Die initiale Permutation IP

Chiffre angewendet, und der Geheimtext ist dann

$$\mathrm{IP}^{-1}(R_{16}, L_{16}).$$

Rundenschlüssel von DES. Wir beschreiben nun, wie die Rundenschlüssel K_i aus dem Hauptschlüssel K gebildet werden. Jeder Rundenschlüssel K_i hat Länge die 48. Zunächst sei

$$v_i = \begin{cases} 1 & \text{für } i = 1, 2, 9, 16, \\ 2 & \text{sonst.} \end{cases}$$

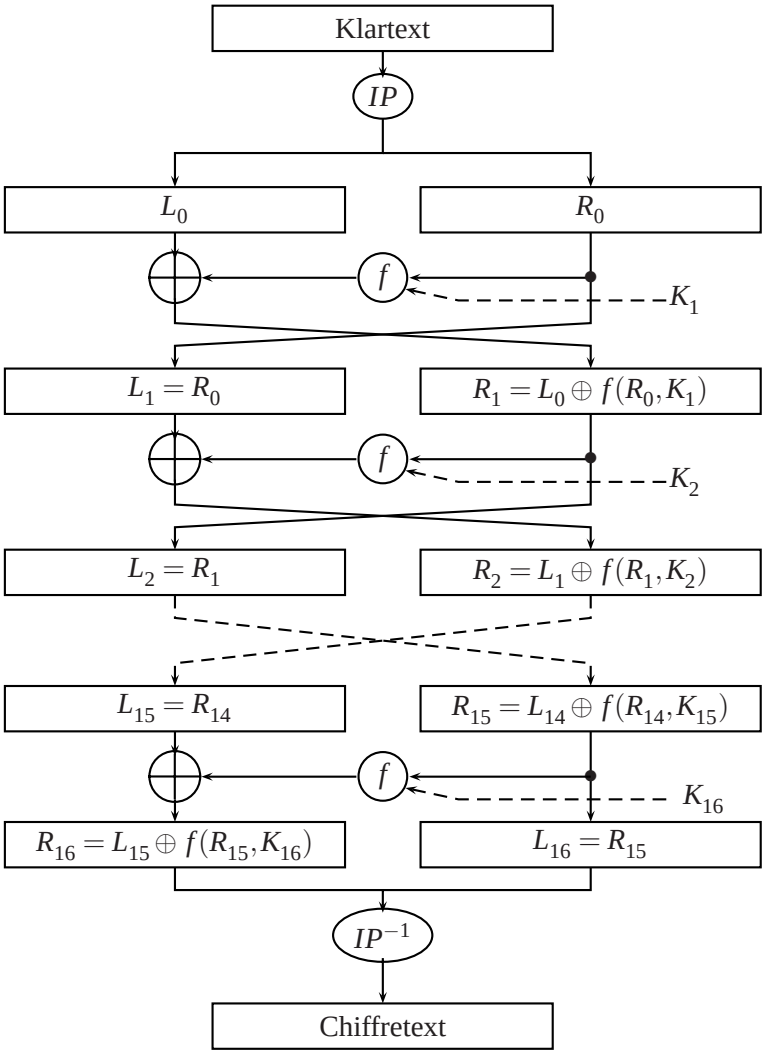


TABELLE 2. Schema des DES

Mit Hilfe der Tabellen PC 1 und PC 2 definiert man Abbildungen

$$\mathrm{PC}\,1 : \{0,1\}^{64} \to \{0,1\}^{28} \times \{0,1\}^{28}$$
$$\mathrm{PC}\,2 : \{0,1\}^{28} \times \{0,1\}^{28} \to \{0,1\}^{48}.$$

Dabei wählt PC 1 aus den Bits von K die in der oberen Hälfte der Tabelle gegebenen Positionen für die linke Komponente des Bildes aus und die Positionen in der unteren Hälfte für die rechte Komponente. (Man beachte, daß die Bits an den Positionen $j = 8, 16, \ldots, 64$ nicht benutzt werden.) Auf

$$(U_0, V_0) = \mathrm{PC}\,1(K)$$

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC2					
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

TABELLE 3. Die Funktionen PC1 und PC2

wendet man ein „Rundenverfahren“ an:

$$U_i = \lambda^{v_i}(U_{i-1}), \quad V_i = \lambda^{v_i}(V_{i-1}),$$

wobei λ die zyklische Linksverschiebung ist, die je nach Wert von v_i ein- oder zweimal angewandt wird. Am Ende ist

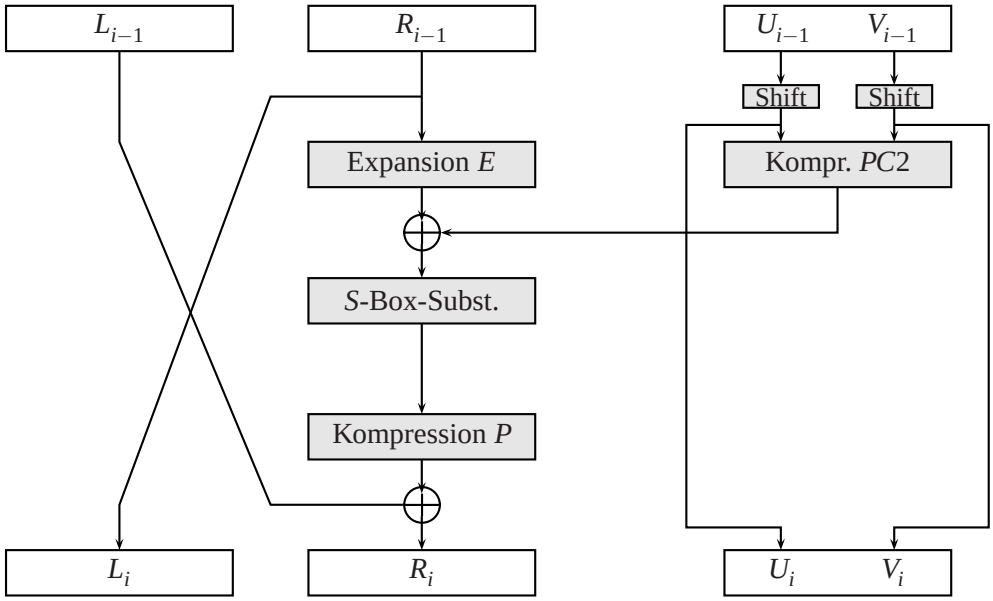


TABELLE 4. Eine Runde des DES

$$K_i = \mathrm{PC}\,2(U_i, V_i), \quad i = 1, \ldots, 16,$$

wobei PC 2 aus den 56 Bit gemäß der Tabelle 48 auswählt. Für den Schlüssel

```
13 34 57 79 9B BC DF F1
```

in hexadezimaler Notation ergeben sich die binären Rundenschlüssel in Tabelle 5.

```
K[ 1]= 000110110000000101110111111111100011100000111001O
K[ 2]= 011110011010111011011001110110111100100111100101
K[ 3]= 010101011111110010001010010000101100111110011001
K[ 4]= 011100101010110111010110110110110011010100011101
K[ 5]= 011111001110110000000111111010110101001110101000
K[ 6]= 011000111010010100111110010100000111101100101111
K[ 7]= 111011001000010010110111111101100001100010111100
K[ 8]= 111101111000101000111010110000010011101111111011
K[ 9]= 111000001101101111101011111011011110011110000001
K[10]= 101100011111001101000111101110100100011001001111
K[11]= 001000010101111111010011110111101101001110000110
K[12]= 011101010111000111110101100101000110011111101001
K[13]= 100101111100010111010001111110101011101001000001
K[14]= 010111110100001110110111111100101110011100111010
K[15]= 101111111001000110001101001111010011111100001010
K[16]= 110010110011110110001011000011100001011111110101
```

TABELLE 5. Rundenschlüssel bei DES

Innere Blockchiffre. Nun fehlt „nur“ noch die Beschreibung der in jeder Feistel-Runde anzuwendenden Chiffrierung

$$f_{K_i}(R_{i-1}), \quad i = 1, \ldots, 16.$$

Sei $R \in \{0,1\}^{32}$ gegeben. Dann wird R zunächst einmal gemäß der Tabelle E durch zweifache Verwendung gewisser Komponenten auf die Länge 48 „expan-

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

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

TABELLE 6. Die Funktionen E und P

diert“. Alsdann wird $E(R) \oplus K_i$ in 8 Komponenten der Länge 6 gespalten:

$$E(R) \oplus K_i = B_1 \ldots B_8.$$

Im letzten Schritt wendet man auf B_i eine Funktion $S_i : \{0,1\}^6 \to \{0,1\}^4$, $i = 1, \ldots, 8$, an und setzt die $C_i = S_i(B_i)$ zur Bitfolge

$$C_1 \ldots C_8 \in \{0,1\}^{32}$$

zusammen. Nun wird noch die Permutation P gemäß ihrer Tabelle angewandt, und man erhält

$$f_{K_i}(R) = P(C_1 \dots C_8).$$

Nachzutragen ist die Beschreibung der S_i. Für sie werden die sogenannten S-Boxen verwendet (siehe Tabelle 7. Sei

$$B = b_1 \dots b_6 \in \{0,1\}^6.$$

Wir setzen $u = (b_1 b_6)_2$ (der Index 2 weist darauf hin, daß wir Zahlen im Dualsystem interpretieren) und $v = (b_2 \dots b_5)_2$. Sodann betrachten wir u als Zeilen- und v als Spaltenindex in der „Box“ S_i $(i = 1, \dots, 8)$. Den entsprechenden Tabelleneintrag wählen wir aus, stellen ihn im Dualsystem dar und füllen mit führenden Nullen auf die Länge 4 auf. Das ergibt $S_i(B)$.

Beispiel 14.1. Wir wollen $S_5(011100)$ bestimmen. Dann ist $u = (00)_2 = 0$, $v = (1110)_2 = 14$. Nach Tabelle ist

$$S_5(0,14) = 14 = (1110)_2.$$

Die nun vollständige Beschreibung von DES ist so kompliziert, daß man sie kaum behalten kann. Man sollte aber beachten, daß sie nur sehr einfache Operationen erfordert, die sich leicht in Hardware realisieren lassen: XOR, zyklische Verschiebung, Auswahl und Zusammensetzung von Bits. Bereits 1993 gab es DES-Chips, die 200 MByte pro Sekunde verarbeitet haben (siehe [**Scn**]). Die Sicherheit des Verfahrens beruht auf der Konstruktion der S-Boxen. Alle anderen Schritte sind lineare Abbildungen, die keine Sicherheit bieten.

Entstehung und Sicherheit von DES. DES wurde von der IBM im Auftrag des NBS (National Bureau of Standards, heute NIST) entwickelt und 1977 veröffentlicht. Bis heute ist die Beteiligung der NSA (National Security Agency) an der Entwicklung nicht völlig geklärt. Da zwar der Algorithmus, nicht aber (zunächst) die Kriterien des Entwurfs veröffentlicht wurden, konnten sich Gerüchte halten, die NSA habe in DES geheime Hintertüren eingebaut, die die unbefugte Dechiffrierung „von Amts wegen“ ermöglichen könnte. Andererseits wurde das Verfahren von vielen Kryptanalytikern intensiv geprüft, ohne daß Schwachstellen gefunden wurden. Die einzige wesentliche Schwachstelle ist die rasante Entwicklung der Computer-Technik, mit deren Hilfe man heute einen Brute-Force-Angriff angesichts der „kleinen“ Schlüssellänge von 56 Bit in weniger als einem Tag vornehmen kann. Man probiert schlichtweg alle vorhandenen Schlüssel aus und dechiffriert den Geheimtext. Dann bestimmt man die Entropie des zum jeweiligen Schlüssels gehörigen Klartextes und erklärt denjenigen unter ihnen als wahrscheinlichsten Klartext, der die niedrigste Entropie liefert. (Gibt es einige eng bei einander liegende Kandidaten, so kann man ja auch noch einen menschlichen Leser als Hilfe

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

TABELLE 7. *S*-Boxen des DES

benutzen.) Die 1997 in [**Ba**], p. 172 ausgesprochene Hoffnung „Für private Initiativen sollte DES unangreifbar sein und ist es höchstwahrscheinlich auch, wenn es diszipliniert gebraucht wird“ ist inzwischen von der Wirklichkeit überholt.

Gegen DES (und verwandte Verfahren) sind noch andere Angriffsarten erfunden worden, nämlich die *differentielle* und die *lineare Kryptanalyse* (siehe dazu [**Mv**] und [**Sti**]). Es handelt sich dabei um Angriffe mit gewähltem Klartext, bei denen man versucht, die Abweichungen zwischen den Geheimtexten benachbarter Klartexte für die Rekonstruktion des Schlüssels auszunutzen. Die lineare Kryptanalyse versucht, die einzelnen Runden möglichst gut linear zu approximieren. Es ist später eingeräumt worden, daß die *S*-Boxen so konstruiert worden sind, daß sie der differentiellen Kryptanalyse möglichst gut widerstehen.

Beim Entwurf von Kryptosystemen muß man nach Shannon zwei Prinzipien berücksichtigen, nämlich *Konfusion* und *Diffusion*. Die Konfusion zerreißt den Zusammenhang zwischen Klartext und Geheimtext Zeichen für Zeichen, während Diffusion dafür sorgt, daß sich Änderungen an einzelnen Stellen des Klartextes (oder des Schlüssels) sich mit annähernd gleicher Wahrscheinlichkeit auf alle Stellen des Geheimtextes auswirken. Die Konstruktion der *S*-Boxen sorgt für Diffusion im DES-Algorithmus. Um dies zu demonstrieren, chiffrieren wir mit dem Schlüssel K die Klartexte P_1 und P_2 zu den Geheimtexten C_1 und C_2 (hexandezimale Notation):

```
K = 13 34 57 79 9B BC DF F1
P1= 00 00 00 00 00 00 00 00
P2= 00 00 00 00 00 00 00 01
C1= 94 8A 43 F9 8A 83 4F 7E
C2= 5D 59 D4 46 07 49 5A 7A
```

Tabelle 8, in der die Daten (L_i, R_i) für $i = 0, 1, 2, 3, 6$ für P_1 und P_2 in binärer Form jeweils untereinander stehen, zeigen den „Lawineneffekt“: die Hamming-Abstände d vergrößern sich schnell von Runde zu Runde.

i	L_i	R_i	d
0	00000000000000000000000000000000	00000000000000000000000000000000	
	00000000000000000000000010000000	00000000000000000000000000000000	1
1	00000000000000000000000000000000	00100010001100110100000011110011	
	00000000000000000000000000000000	00100010001100110100000001110011	1
2	00100010001100110100000011110011	01111001110110100001111001100101	
	00100010001100110100000001110011	01111011111010100011101001100100	7
3	01111001110110100001111001100101	00100100101010110001111011000110	
	01111011111010100011101001100100	11111001110010110101010110011100	22
6	01001110100101101000101000010010	01000000010001001111000101000111	
	00000011101110001011111111000100	01001101010001010000000101101000	30

TABELLE 8. Lawineneffekt bei DES

Triple-DES. Wie oben bereits dargelegt, sind heute Brute-Force-Angriffe gegen DES möglich. Das Verfahren kann aber leicht so modifiziert werden, daß ein solcher Angriff zumindest auf absehbare Zeit keine Chance hat (und auch kein anderer erfolgversprechender Angriff bekannt ist). Für die Variante *Triple-DES* (und andere Varianten) ist es wichtig, daß die 2^{56} Permutationen DES_K von $\{0,1\}^{64}$, die den Schlüsseln K des DES zugeordnet sind, *keine* Untergruppe der Permutationsgruppe bilden und im allgemeinen

$$\mathrm{DES}_K \circ \mathrm{DES}_L \neq \mathrm{DES}_M, \qquad \mathrm{DES}_K^{-1} \neq \mathrm{DES}_M$$

für alle Schlüssel M ist. (Es ist bekannt, daß die DES-Permutationen eine Gruppe der Ordnung $\geq 10^{2499}$ erzeugen.)

Für Triple-DES verwenden wir drei Schlüssel K_1, K_2, K_3 und die Chiffrierung

$$P \mapsto \mathrm{DES}_{K_3} \circ \mathrm{DES}_{K_2}^{-1} \circ \mathrm{DES}_{K_1}(P).$$

Für weitere Varianten verweisen wir auf [**Scn**]. Der Internet-Explorer von Microsoft verwendet DES mit 56 Bit Schlüssellänge „international“ und Triple-DES mit 128 Bit Schlüssellänge ($K_3 = K_1$) für „Hochsicherheits-Arbeitsumgebungen in Nordamerika“.

Man kann sich natürlich fragen, weshalb Triple-DES verwendet wird und nicht eine weniger aufwendige Doppel-Chiffrierung

$$P \mapsto C = \mathrm{DES}_L(\mathrm{DES}_K(P)).$$

Der Grund dafür ist der sogenannte „Meet-in-the-middle“-Angriff, der zeigt, daß ein Klartext-Angriff gegen die Doppel-Chiffrierung weniger Aufwand erfordert, als man auf den ersten Blick annehmen würde. Es sieht ja zunächst so aus, als müßte man zur Ermittlung der Schlüssel K und L alle 2^{112} Paare (K,L) durchprobieren. Stattdessen geht man so vor: Man chiffriert P mit allen Schlüsseln K, sortiert die Ergebnisse und speichert sie. Dann dechiffriert man C mit allen Schlüsseln L und vergleicht $\mathrm{DES}_L^{-1}(C)$ mit der gespeicherten Liste. Dabei stößt man auf Paare (K,L), die sich in der Mitte treffen, also die Gleichung

$$\mathrm{DES}_L^{-1}(C) = \mathrm{DES}_K(P).$$

erfüllen. Ob dieses Schlüssel-Paar schon das gesuchte ist, prüft man an einem zweiten Klartext-Geheimtext-Paar (P',C') und gegebenenfalls mit weiteren solchen bekannten Paaren. Bereits bei zwei Übereinstimmungen ist das Schlüssel-Paar mit sehr hoher Wahrscheinlichkeit gefunden. Dieser Angriff erfordert nur 2^{57} Ausführungen des DES-Algorithmus, aber natürlich einen enormen Speicherbedarf. Er ist daher zur Zeit wohl nur von theoretischem Interesse, aber dennoch der Grund, gleich auf Dreifach-Chiffrierung überzugehen. Ein Meet-in-the-middle-Angriff erfordert dann tatsächlich $2^{112} + 2^{56}$ Ausführungen von DES und ist daher jenseits aller realisierbaren Rechenzeiten.

Schwache Schlüssel. Beim DES-Algorithmus (und verwandten Verfahren) gibt es einige schwache Schlüssel (siehe Aufgabe 14.3), die man vermeiden muß. Ferner existieren 6 Paare (K, K') *halbschwacher Schlüssel*, für die

$$\mathrm{DES}_K = \mathrm{DES}_{K'}^{-1}.$$

Aus jedem dieser 12 Schlüssel entstehen nur jeweils zwei verschiedene Rundenschlüssel. Hinzu kommen 48 *möglicherweise schwache Schlüssel*, die nur zu je 4 Rundenschlüsseln führen. Insgesamt sind also 64 Schlüssel zu vermeiden.

DES und Paßwörter. Sei P ein Klartext-Block. Unter Vertauschung der Rollen von Klartext und Schlüssel können wir die Abbildung

$$K \mapsto \mathrm{DES}_K(P)$$

betrachten. Nach allem was wir wissen, ist sie eine Einweg-Funktion ohne Falltür (und das sichert DES gegen Klartext-Angriffe). Daher kann man diese Funktion für die Speicherung und Überprüfung von Paßwörtern im Rechner-Systemen nutzen. (Für diesen Zweck wurden Einweg-Funktionen bereits 1968 vorgeschlagen; siehe [**Ko1**], p. 86.) Man wählt das vom Nutzer vergebene *Paßwort als Schlüssel* K und chiffriert damit einen fest gewählten Klartext. Nur der erzeugte Geheimtext wird in der Paßwort-Datei gespeichert. Bei der erneuten Anmeldung wird zu dem eingegebenen Paßwort wieder der Geheimtext erzeugt und mit dem gespeicherten Geheimtext verglichen.

Auch wenn die Paßwort-Datei gestohlen wird, ist die Sicherheit des Systems nicht gefährdet, da DES Klartext-Angriffen standhält. Allerdings kann man „offline“ versuchen, „naheliegende“ Paßwörter und zugehörige Geheimtexte zu erzeugen und dieses „Wörterbuch“ mit den Einträgen in der Paßwort-Datei zu vergleichen. Dies hat oft Erfolg, weil viele Nutzer ihr Paßwort sorglos wählen.

Das klassische Unix-Paßwort dient als Schlüssel für eine 25-fach iterierte Verschlüsselung des 0-Worts mit DES. Das Endresultat wird in (druckbarer Form) in der Paßwort-Datei abgelegt. Um das Knacken der Paßwörter mit spezieller DES-Hardware zu verhindern, wurde der DES-Algorithmus aber leicht modifiziert: Statt der festen Expansion E wird abhängig von einer Prise „salt“ eine von 4096 Permutationen gewählt. Das gewählte „salt“ wird mit dem Paßwort gespeichert. Die Verwendung von „salt“ schützt auch gegen Wörterbuch-Angriffe. Jedes Paßwort hat nun 4096 mögliche Verschlüsselungen, was den Umfang des benötigten Wörterbuchs entsprechend vervielfacht.

Ein Schutz gegen schlecht gewählte Paßwörter ist das aber letzten Endes auch nicht. Da man dem Sicherheitsbewußtsein der Nutzer nicht trauen kann, muß die Systemverwaltung bei besonders hohen Anforderungen an die Sicherheit Paßwörter zufällig erzeugte Paßwörter vergeben.

AES (Adavanced Encryption Standard). Im Jahr 1997 hat das NIST einen Wettbewerb für einen Nachfolger von DES als standardisierte Blockchiffre ausgeschrieben. Als Sieger ging daraus im Jahr 2000 das System *Rijndael* hervor. Es kann zum Beispiel für die Blocklänge 128 Bit und Schlüssel der Länge 128 Bit konfiguriert werden. Wie DES arbeitet es in Runden, deren Zahl aber nur 10 beträgt. Aus dem Schlüssel werden Rundenschlüssel erzeugt. Die einzelnen Runden sind aber keine Feistel-Runden, und statt Permutationen kommen (auch) arithmetische Operationen im Körper $\mathbb{F}_{2^8}$ zum Einsatz.

Übungen

14.2. Für $x \in \{0,1\}^n$ sein $\overline{x}$ das bitweise bebildete Komplement, d.h. $\overline{x}_i = 1 - x_i$. Zeige:: $\mathrm{DES}(\overline{P},\overline{K}) = \overline{\mathrm{DES}(P,K)}$. (Dies ist die einzige bekannte Symmetrie-Eigenschaft von DES.)

14.3. (a) Zeige daß U_{16} und V_{16} aus U_1 und V_1 durch eine zyklische Rechtsverschiebung um 1 Stelle hervorgehen.

(b) Zeige daß jeweils alle Bits in U_1 und alle Bits in V_1 übereinstimmen, wenn alle Rundenschlüssel K_i, $i = 1,\ldots,16$ gleich sind.

(c) Zeige, daß es genau 4 Schlüssel für DES gibt, für die alle Rundenschlüssel übereinstimmen. Welche sind diese *schwachen* Schlüssel?

14.4. Eine bekannte Stromchiffre ist RC4. Sie kommt auch in den gängigen Browsern zum Einsatz. Der Internet Explorer von Microsoft verwendet für RC4 „international“ die Schlüssellängen 40 und 56 Bit, „in Nordamerika“ die Schlüssellänge 128 Bit. RC4 war ursprünglich ein Geschäftsgeheimnis der Firma RSA Data Security Inc., wurde jedoch 1994 von den *Cypherpunks* in einer Mailing-Liste offengelegt. RSA verlangt eine Lizenz für kommerzielle Nutzung.

Der Schlüssel wird mit einer S-Box $(S_0,\ldots,S_{255})$ von 256 Bytes erzeugt. Diese wird mit $S_i = i$ für alle i initialisiert. Sei L ein maximal 256 Bytes langer Schlüssel. Man füllt nun ein Feld $(K_0,\ldots,K_{255})$ von links mit den Bytes von L und wiederholt dies so lange, bis alle 256 Bytes K_i gefüllt sind. (Es ist nicht erforderlich, daß die Länge von L ein Teiler von 256 ist.)

Nun setzt man $j = 0$ und iteriert

Für $i = 0$ bis 255
    $j = (j + S_i + K_i) \mod 256$,
    vertausche S_i und S_j,
Ende Schleife

Dann werden i und j auf 0 gesetzt und die Bytes B der Stromchiffre sukzessiv durch folgende Iteration erzeugt:

$$
\begin{aligned}
&i := (i+1) \mod 256\\
&j := (j+S_i) \mod 256\\
&\text{vertausche } S_i \text{ und } S_j\\
&k := (S_i+S_j) \mod 256\\
&B := S_k
\end{aligned}
$$

Für die Verschlüsselung (und ebenso die Entschlüsselung) verknüpft man das t-te Byte des Klartexts per XOR mit dem t-ten Byte des Schlüsselstroms.

Implementiere RC4, etwa in `Aribas`, und verschlüssele den Text

```
mitdemdeutschenbildungssystemstehteszweifellosnichtzumbestendieschuelerschneide
nininternationalenleistungsvergleichenmeistmaessigbismiserabelabdieabiturienten
quoteliegtniedrigeralsinanderenlaendernunddiehochschulenverzeichnenatemberauben
deabbrecherzahlenundausuferndestudienzeitennachzweijaehrigerarbeithatdasforumbi
ldungempfehlungenvorgelegtdieesinsichhabenvomkindergartenbiszurhochschulewollen
bundundlaendergemeinsamreformenanpacken
```

mit dem in ASCII-Form gegebenen Schlüssel `Kryptologie`.

ABSCHNITT 15

Betriebsmodi für Blockchiffren

Zusätzlich zu DES sind auch sogenannte Betriebsmodi standardisiert worden. Diese lassen sich aber auch mit jeder anderen (binären) Blockchiffre einsetzen. Wir gehen im folgenden davon aus, daß eine solche Blockchiffre $E : \{0,1\}^n \to \{0,1\}^n$ eingesetzt wird.

ECB. Der einfachste Betriebsmodus ist ECB (*electronic code book*). Bei ihm wird jeder Klartextblock p_i der Länge n zum Geheimtextblock $c_i = E(p_i)$ verschlüsselt und die Folge (c_i) wird gesendet. Abbildung 1 stellt dies in einem „Blockschaltbild“ dar.

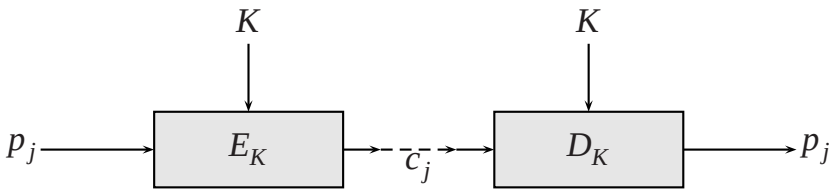


ABBILDUNG 1. Der ECB-Modus

Da man nur ganze Blöcke verschlüsseln kann, entsteht am Ende des Klartexts möglicherweise ein Problem: der Empfänger kann nicht entscheiden, wo der Klartext wirklich aufhört. Man kann dies so lösen: Das allerletzte Byte des Klartexts enthält die Zahl derjenigen Bytes des letzten Blocks, die nicht mehr zur eigentlichen Nachricht gehören. Der Nachteil dieses Verfahrens ist, daß man man einen ganzen, eigentlich überflüssigen Block anhängen muß, wenn die Nachricht mit einem Block genau abschließt. Es gibt eine elegante Lösung dieses Problems, die es einem sogar ermöglicht, den Geheimtext immer genauso lang zu halten wie den Klartext. Der Klartext ende in

$$p_{N-1}, p_N = (p'_N, p''_N),$$

wobei p'_N noch Information enthält und p''_N nur dazu dient, den Block aufzufüllen. Nun chiffriert man $c_{N-1} = E_K(p_{N-1}$ und zerlegt

$$c_{N-1} = (c'_{N-1}, c''_{N-1}),$$

wobei c'_{N-1} die gleiche Länge wie p'_N hat. Übertragen wird schließlich die Folge

$$E_K(p'_N, c''_{N-1}), c'_{N-1},$$

wobei am Ende eventuell nur ein Teilblock zu senden ist. Diese Technik heißt *ciphertext stealing*, weil man etwas Chiffretext „stiehlt“, um den Klartext aufzufüllen. Den gesparten Platz hat man sich mit der Unbequemlichkeit erkauft, daß die Dechiffrierung verzögert wird, und man erst dann weiß, ob ein dechiffrierter Block wirklich Klartext ist, wenn noch zwei weitere Blöcke übertragen worden sind oder die Übertragung beendet ist.

Ein wesentlicher Nachteil dieses Modus ist seine Empfindlichkeit gegen Attacken des „man in the middle“, der sich zwischen Sender und Empfänger einschaltet und von diesen unbemerkt Geheimtextblöcke entfernen, vertauschen oder zusätzlich einschleusen kann. Da jeder Geheimtextblock c_i einzeln dechiffriert werden kann und direkt Klartext liefert, ist nicht ohne weiteres erkennbar, ob Geheimtextblöcke entfernt oder hinzugefügt worden sind.

Wir diskutieren dazu ein kleines Szenario, den Angriff mittels *block replay*. Die A-Bank sendet Nachrichten an die B-Bank, die Informationen über Überweisungen enthalten. Der böse Mallory (so heißt der „man in the middle“ in der kryptographischen Literatur üblicherweise) hat Konten bei der A-Bank und der B-Bank. Mallory überweist 100 Euro von A nach B und wiederholt dies gleich noch einmal. In dem von ihm abgehörten Datenverkehr entdeckt er dann zwei identische Nachrichten, und kann mit einigem Recht annehmen, daß er seine Überweisung identifiziert hat. (Eventuell sendet er seinen Auftrag noch einige Male.) Nun sendet er diese Nachricht immer wieder an die B-Bank, die Mallory unaufhörlich Geld gutschreibt. Wenn die Konten erst am Abend abgeglichen werden, ist Mallory mit dem Geld schon in der Südsee. Die Banken kennen die Gefahr und fügen einen Zeitstempel-Block in ihre Nachrichten ein. Dies aber kümmert Mallory nicht im geringsten. Er fängt beliebige Nachrichten von A an B ab, extrahiert den Zeitstempel-Block und nutzt ihn für seine Geldvermehrungsmaschine.

CBC. Der gerade geschilderten Gefahr kann man wirksam durch den Betriebsmodus CBC (*cipher block chaining*) begegnen. Man startet mit einem *Initialisierungs-Block* (oder *Initialisierungs-Vektor*) p_0, auf den die Blöcke $p_1, \ldots, p_N$ des eigentlichen Klartextes folgen. Man bildet nun

$$\begin{aligned} c_0 &= E(p_0) \\ c_i &= E(p_i \oplus c_{i-1}), \qquad i \geq 1, \end{aligned}$$

und überträgt die Blöcke $c_0, \ldots, c_N$. Beim Dechiffrieren ist

$$\begin{aligned} p_0 &= D(c_0), \\ p_i &= D(c_i) \oplus c_{i-1}, \qquad i \geq 1. \end{aligned}$$

Auf diese Weise werden die Geheimtextblöcke verkettet. Wenn die Reihenfolge in einer Übertragung durch Einfügen, Entfernen oder Vertauschen von Blöcken

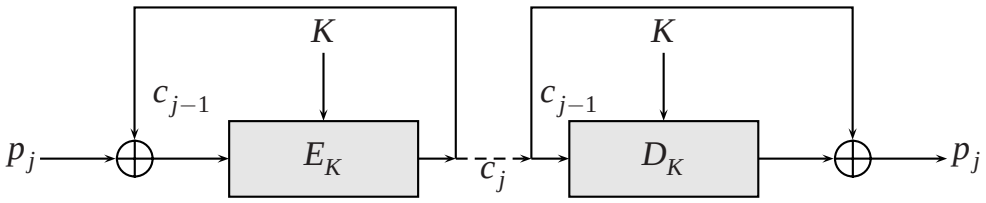


ABBILDUNG 2. Der CBC-Modus

verändert wird, entsteht unlesbarer „Klartext“. (Man kann die Methoden der Codierungstheorie benutzen um dem Klartext gezielt Redundanz hinzuzufügen und so unzulässigen Klartext erkennbar zu machen.) Man beachte, daß auch in diesem Modus gleiche Klartexte zu gleichen Chiffretexten verschlüsselt werden, wenn der gleiche Initialisierungs-Block verwendet wird. Daher wählt man ihn zufällig, um gleichen Klartexten verschiedene Geheimtexte zuzuordnen.

Im ECB-Modus stört ein Fehler bei der Übertragung eines Geheimtextblocks nur einen Block des Klartextes. Im CBC-Modus werden bei einem Fehler in c_{i-1} die Klartextblöcke p_{i-1} und p_i gestört – alle nach c_i folgenden Geheimtextblöcke hängen von c_{i-1} nicht mehr ab.

Der Empfänger muß den Initialisierungs-Block kennen, wenn der erste Chiffretext-Block richtig dechiffriert werden soll. Wenn dies aus irgendwelchen Gründen nicht realisierbar ist. darf die eigentliche Nachricht erst mit dem dritten gesendeten Block beginnen. Der Initialisierungs-Block braucht allerdings nicht geheimgehalten werden. Wenn man ihn bekannt gibt, ist aber ein Paar $(p, E(p))$ offengelegt.

Es besteht immer noch eine gewisse Gefahr, daß an den gesendeten Geheimtext Blöcke angefügt werden. Diese werden zwar zu keinem sinnvollen Klartext führen, aber man sollte dennoch dafür sorgen, daß das Ende des eigentlichen Klartextes sicher vom Dechiffrierer erkannt werden kann.

Ein Nachteil des CBC-Modus (und ebenso des ECB-Modus) ist, daß die Übertragung mindestens einen vollen Klartextblock erfordert (daran ändert auch „ciphertext stealing“ nichts). Dies ist bei manchen Anwendungen sehr nachteilig, zum Beispiel beim Datenverkehr zwischen einem Computer-Terminal und dem Server, weil hier auch Klartexte von der Länge 1 Byte ohne Verzögerung übertragen werden müssen.

CFB. Die Verzögerung kann man mittels des CFB-Modus (*cipher feedback*) vermeiden. Dabei wird eine Blockchiffre benutzt, um eine Stromchiffre zu erzeugen. Vor der Übertragung füllen Sender und Empfänger ein Schieberegister mit dem gleichen Initialisierungs-Block (der Länge der Blockchiffre). Wir können beispielsweise annehmen, daß die Blockchiffre die binäre Länge 64 hat und der Klartext byteweise anfällt. Das Schieberegister wird mittels E_K zu S chiffriert und dieser Schritt wird nach jedem Klartext-Byte wiederholt. Ein eintreffendes Klartext-Byte

p_j wird nun zu

$$c_j = p_j \oplus s_j$$

verschlüsselt, wobei s_j das linke Byte von S zum Zeitpunkt j ist. Das Geheimtext-

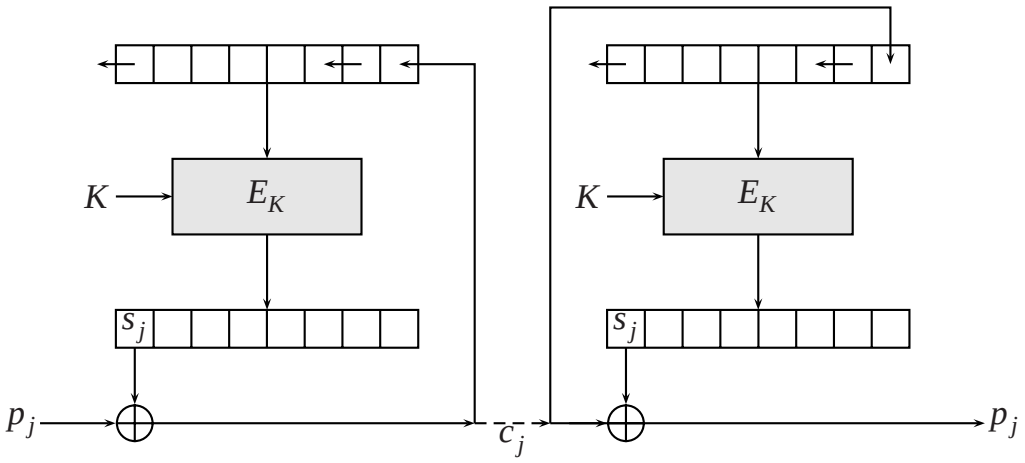


ABBILDUNG 3. Der CFB-Modus

zeichen c_j wird gesendet und außerdem über eine Rückkopplung (feedback) von rechts in das Schieberegister geschoben, dessen Bytes alle um ein Byte nach links wandern – das linke Byte fällt heraus. Die Dechiffrierung erfolgt mit dem gleichen Algorithmus (siehe Abbildung 3).

Wie auch bei CBC sollte jede Nachricht mit einem individuellen Initialisierungs-Block gestartet werden, solange K nicht wechselt.

Ein falsch übertragenes Chiffretext-Byte führt zu einem falschen Klartext-Byte und stört die nachfolgenden Klartext-Bytes so lange, bis es nach links aus dem Schieberegister des Dechiffrierers herausgefallen ist. CFB verhält sich in diesem Sinn auch gutmütig gegenüber Synchronisationsfehlern: fällt ein Chiffretext-Byte ganz aus dem Übertragungsstrom heraus, so setzt die korrekte Dechiffrierung wieder ein, sobald das Schieberegister mit korrekt übertragenen Chiffretext-Bytes gefüllt ist. Man sagt, daß die Chiffre *selbstsynchronisierend* ist.

OFB. Der letzte standardisierte Betriebsmodus ist OFB (*output feedback*). Bei diesem wird nicht c_j rückgekoppelt, sondern s_j. Der wesentliche Vorteil ist, daß die Folge (s_j) vorab berechnet werden kann, da sie unabhängig vom Klartext ist. Ferner schädigt ein falsches Chiffretext-Byte nur das entsprechende Klartext-Byte. Allerdings wirken sich Synchronisationsfehler fatal aus: von einem ausgelassenen Chiffretext-Byte ab ist der folgende Klartext unbrauchbar.

Der OFB-Modus ist eine Stromchiffre, bei dem die zugrunde liegende Blockchiffre nur verwendet wird, um den Schlüsselstrom der Stromchiffre zu erzeugen.

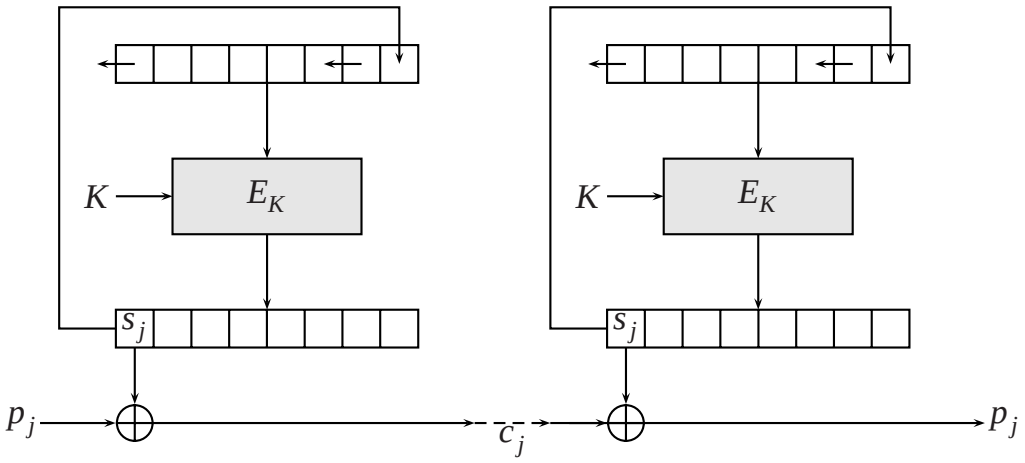


ABBILDUNG 4. Der OFB-Modus

Der Schlüsselstrom ist vom Klartext unabhängig. Daher ist es unbedingt erforderlich, für jede Nachricht einen individuellen Initialisierungsblock zu verwenden. Sonst werden verschiedene Klartexte mit der gleichen Schlüsselstrom chiffriert und man kann Teile dieser Texte gegeneinander austauschen, ohne daß unsinnige Klartexte entstehen.

Wir sind bei CFB und OFB davon ausgegangen, daß die zu sendenden Einheiten Bytes sind, während die Blockchiffre Länge 64 hat. Diese Annahme diente nur der Illustration. Beide Verfahren lassen sich bei Blockchiffren der binären Länge n auf zu sendende Einheiten der Länge $m \leq n$ anwenden. Die Aussagen über Fehlerfortpflanzung und Synchronisation sind dann aber entsprechend zu modifizieren.

Man kann sich gegen das Verfälschen von Nachrichten durch Einfügen, Vertauschen oder Entfernen von Blöcken auch durch das Einfügen von Signaturen schützen. Wie solche verallgemeinerten Prüfsummen gebildet werden können, werden wir noch besprechen.

Übungen

15.1. Zwei weitere Betriebsmodi für Blockchiffren sind BC (block chaining) und PCB (plaintext block chaining).

Bei BC startet man mit einem Initialisierungs-Block F_1 und chiffriert die Blöcke $P_1, P_2 \ldots$ des Klartexts mittels

$$C_i = E_K(P_i \oplus F_i), \quad F_{i+1} = F_i \oplus C_i.$$

Bei PBC startet man mit einem Initialisierungs-Block P_0 und chiffriert die Blöcke $P_1, P_2 \ldots$ des Klartexts mittels

$$C_i = E_K(P_i \oplus P_{i-1}).$$

Diskutiere diese Modi. Welchen Vorzug hat CBC ihnen gegenüber?

15.2. Wie kann man „cipher text stealing“ im CBC-Modus realisieren?

ABSCHNITT 16

RSA

Das Grundprinzip eines Kryptosystems mit öffentlichem Schlüssel, das wir in Abschnitt 11 schon kurz angedeutet haben, kann man im wesentlichen so beschreiben:

(1) Jeder Teilnehmer A konstruiert eine Einwegfunktion E_A mit Falltür; die in die Konstruktion einfließende, nur ihm bekannte Zusatzinformation liefert $D_A = E_A^{-1}$.
(2) Er gibt E_A bekannt, hält aber die Entschlüsselung D_A natürlich geheim.
(3) Will nun ein anderer Teilnehmer B eine Nachricht P an A senden, so verschlüsselt er sie mittels E_A und sendet den Chiffretext $C = E_A(P)$ an A.
(4) Wenn A den Chiffretext C empfängt, wendet er die nur ihm bekannte Entschlüsselung D_A an und erhält $P = D_A(E_A(P))$ zurück.

Die Idee des Public-Key-Kryptosystems wurde von Diffie und Hellman ca. 1975 formuliert. Sie wird ihnen gemeinhin zugeschrieben. Nach [**Si**] haben Public-Key-Kryptosystem aber auch eine „geheime“ Geschichte: Mitarbeiter des GCHQ, der britischen Variante der NSA, haben zur etwa gleichen Zeit ebenfalls die Grundlagen für Public-Key-Kryptosysteme entwickelt.

Zahlentheorie von RSA. Das eleganteste Public-Key-Kryptosystem ist das nach seinen Entdeckern Rivest, Adleman und Shamir (1980) benannte RSA-Kryptosystem. Wir rekapitulieren zunächst die Aussagen der elementaren Zahlentheorie, die zu seiner Beschreibung nötig sind.

Satz 16.1. *Sei $m > 0$ eine natürliche Zahl und $a \in \mathbb{Z}$. Genau dann existiert ein $b \in \mathbb{Z}$ mit $ab \equiv 1 \ (m)$, wenn a und m teilerfremd sind.*

Man kann ein geeignetes b mit dem erweiterten euklidischen Algorithmus bestimmen, denn dieser führt auf eine Darstellung

$$1 = \mathrm{ggT}(a,m) = ax + my$$

mit $x, y \in \mathbb{Z}$. Offensichtlich können wir $b = x$ wählen. Die zu m teilerfremden Restklassen bilden also die Gruppe $\mathbb{Z}_m^*$ der Einheiten des Restklassenrings $\mathbb{Z}_m$. Man setzt

$$\varphi(m) = |\mathbb{Z}_m^*|$$

und nennt die so definierte Funktion *Eulersche φ-Funktion*. Mit ihrer Hilfe formulieren wir die Eulersche Verallgemeinerung des „kleinen“ Satzes von Fermat:

Satz 16.2. *Sei $m > 0$ eine natürliche Zahl und $a \in \mathbb{Z}$ teilerfremd zu m. Dann ist $a^{\varphi(m)} \equiv 1 \ (m)$.*

Speziell gilt $a^{p-1} \equiv 1 \ (p)$, wenn p eine Primzahl und a nicht durch p teilbar ist.

Dies folgt einfach daraus, daß die Ordnung der Restklasse von a in $\mathbb{Z}_m^*$ die Ordnung $\varphi(m)$ dieser Gruppe teilen muß. Für Primzahlen p ist $\varphi(p) = p-1$.

Wir benötigen auch den chinesischen Restsatz. Er zeigt, daß eine Zerlegung $m = uv$ in teilerfremde Faktoren u und v zu einer entsprechenden Aufspaltung des Restklassenrings $\mathbb{Z}_m^*$ führt:

Satz 16.3. *Sei $m > 0$ eine natürliche Zahl, $m = uv$ mit teilerfremden $u, v \in \mathbb{Z}$. Dann ist die Abbildung*

$$\mathbb{Z}_m \to \mathbb{Z}_u \times \mathbb{Z}_v, \quad (a \mod m) \mapsto (a \mod u, \ a \mod v)$$

ein Isomorphismus $\mathbb{Z}_m \cong \mathbb{Z}_u \times \mathbb{Z}_v$.

Speziell gilt $\mathbb{Z}_m^* \cong \mathbb{Z}_u^* \times \mathbb{Z}_v^*$ und damit $\varphi(m) = \varphi(u)\varphi(v)$.

Für den Beweis des Satzes konstruiert man zweckmäßig die Umkehrabbildung $\mathbb{Z}_u \times \mathbb{Z}_v \to \mathbb{Z}_m$. Dies läuft darauf hinaus, das System

$$\begin{aligned} x &\equiv a \ (u) \\ x &\equiv b \ (v) \end{aligned}$$

für alle $a, b \in \mathbb{Z}$ zu lösen. Dazu bestimmt man eine Darstellung $1 = uy + vz$ mit dem erweiterten euklidischen Algorithmus und setzt $x = a(vz) + b(uy)$.

Der grundlegende Satz für das RSA-Kryptosystem ist

Satz 16.4. *Genau dann ist $m \in \mathbb{Z}$, $m > 1$, Produkt paarweise verschiedener Primzahlen, wenn für alle Zahlen a gilt:*

$$a^{\varphi(m)+1} \equiv a \ (m).$$

Beweis. Wir benötigen nur die Implikation „$\Longrightarrow$“. Sei $m = p_1 \cdots p_n$ mit Primzahlen p_i. Wegen des chinesischen Restsatzes genügt es zu zeigen, daß $a^{\varphi(m)+1} \equiv a \ (p_i)$ für alle i gilt. Der chinesische Restsatz zeigt außerdem

$$\varphi(m) = \varphi(p_1) \cdots \varphi(p_n) = (p_1 - 1) \cdots (p_n - 1).$$

Wir setzen $e_i = \varphi(m)/(p_i - 1)$.

Falls $a \equiv 0 \ (p_i)$ ist, ist auch $a^{\varphi(m)+1} \equiv 0 \equiv a \ (p_i)$. Andernfalls erhalten wir aus dem kleinen Satz von Fermat

$$a^{\varphi(m)+1} = a a^{\varphi(m)} = a(a^{p_i-1})^{e_i} \equiv a \cdot 1 \equiv a \ (p_i). \qquad \square$$

Das RSA-Schema. Das RSA-Kryptosystem arbeitet nach folgendem Schema. Die in ihm verwendeten Namen „Alice“ und „Bob“ werden in der kryptographischen Literatur gern für die Bezeichnung von Personen A und B benutzt.

(1) Alice wählt zufällig zwei große Primzahlen p und q. Gängige Wahlen sind Primzahlen, deren Darstellung im Dualsystem eine Länge ≥ 256 oder ≥ 512 hat.
(2) Sie bildet das Produkt $m_A = pq$ und $\varphi(m_A) = (p-1)(q-1)$. Man nennt m_A den *RSA-Modul* von Alice.
(3) Sie wählt ferner eine zu $\varphi(m)$ teilerfremde Zahl e_A, ihren *Verschlüsselungs-Exponenten*, zwischen 1 und $\varphi(m)-1$.
(4) Sie bestimmt als *Entschlüsselungs-Exponenten* die Zahl d_A, $1 \leq d_A < \varphi(m_A)$ mit $e_A d_A \equiv 1 \ \ (\varphi(m_A))$.
(5) Sie gibt m_A und e_A öffentlich bekannt, hält aber d_A, p, q und $\varphi(m_A)$ geheim. Damit sind ihre Vorbereitungen abgeschlossen.
(6) Wenn Bob eine Nachricht an Alice senden will, so stellt er den Klartext zunächst als eine Folge von Zahlen $n_1, \ldots, n_r$ zwischen 0 und $m_A - 1$ dar. Dann bestimmt er zu jeder dieser Zahlen n_i den Geheimtext

$$c_i \equiv n_i^{e_A} \ \ (m_A)$$

und sendet die Folge $c_1, \ldots, c_r$ an A. (Jede der Restklassen wird dabei natürlich durch c_i mit $0 \leq c_i \leq m_A - 1$ repräsentiert.)
(7) Alice empfängt $c_1, \ldots, c_r$, bildet die Potenzen

$$c_i^{d_A} \equiv n_i \ \ (m_A)$$

und erhält $n_1, \ldots, n_r$ zurück (die dann noch in die eigentliche Nachricht zu transformieren sind).

Wir haben im RSA-Schema Kongruenzen als Wertzuweisungen benutzt und werden dies von nun an oft tun. Dies ist immer so zu verstehen, daß der zugewiesene Wert der kleinste nichtnegative Repräsentant der Restklasse ist. (Es gibt später Stellen, bei denen es auf die Wahl des Repräsentanten ankommt.)

Beispiel 16.5. Wir wählen $p = 3$, $q = 11$, also $m = 33$. Da $(p-1)(q-1) = 20$ ist, ist $e = 3$ als Verschlüsselungs-Exponent brauchbar. Wegen $3 \cdot 7 \equiv 1 \ \ (20)$, ergibt sich der Verschlüsselungs-Exponent $d = 7$. Für den Klartext $n = 13$ erhalten wir den Geheimtext $c \equiv 13^3 \equiv 19 \ \ (33)$, und wie gewünscht ist $c^7 \equiv 13 \ \ (33)$.

Zunächst sollten wir uns klar machen daß das Verfahren wirklich allgemein funktioniert. Nach Wahl von d_A gilt ja

$$e_A d_A \equiv 1 \ \ (\varphi(m_A)),$$

also

$$e_A d_A = t\,\varphi(m_A) + 1$$

mit einer Zahl $t \geq 0$. Dann ist mit $m = m_A$

$$c_1^{d_A} \equiv n_i^{e_A d_A} \equiv n_i^{t\varphi(m)+1} \equiv (n_i^{\varphi(m)})^t n_i \equiv (n_i^{\varphi(m)})^{(t-1)} n_i^{\varphi(m)} n_i \\ \equiv n_i^{\varphi(m)(t-1)} n_i \equiv \cdots \equiv n_i \ \ (m)$$

nach Satz 16.4.

Für die Durchführbarkeit des RSA-Verfahrens ist es notwendig, daß man große Primzahlen schnell finden kann und Potenzen mit sehr großen Exponenten modulo m_A berechnen kann, abgesehen davon, daß am eine „Langzahl-Arithmetik“ braucht, die das genaue Rechnen modulo m_A gestattet.

Schnelle Potenzierung. Um die Potenzen a^n modulo m schnell zu berechnen, stellt man n im Dualsystem dar (falls es nicht ohnehin so gegeben ist),

$$n = \sum_{k=0}^{u} \delta_k 2^k, \quad \delta_k \in \{0,1\},$$

und bestimmt $a^n \mod n$ so: Sei $a_{u+1} := 1$ und mit absteigender Rekursion

$$a_k := \begin{cases} a_{k+1}^2 \cdot a \ \ (m) & \text{wenn } \delta_k = 1 \\ a_{k+1}^2 \ \ (m) & \text{wenn } \delta_k = 0. \end{cases}$$

Dann ist $a_0 \equiv a^n \ \ (m)$. Auf diese Weise brauchen wir höchstens $(2\log_2 m) - 1$ Multiplikationen modulo m.

Beispiel 16.6. Sei etwa $m = 1903$, $a = 2$, und

$$n = 1902 = (11101101110)_2.$$

Dann ist $a_{11} = 1$, $a_{10} = 2$, $a_9 = 8$, $a_8 = 128$, $a_7 = 1160$, $a_6 = 358$, $a_5 = 1326$, $a_4 = 1807$, $a_3 = 1305$, $a_2 = 1583$, $a_1 = 1179$, $a_0 = 851$.

Primzahltests. Den einfachsten Primzahl-Test, der nicht auf der Probedivision beruht, gewinnt man aus dem Satz von Fermat:

> Ist $a^{n-1} \not\equiv 1 \ \ (n)$ für eine Zahl $a \in \mathbb{Z}$, die von n nicht geteilt wird, so ist n nicht prim.

Wir haben gerade gesehen, daß $2^{1902} \not\equiv 1 \ \ (1903)$ ist. Also ist 1903 keine Primzahl. Der Fermat-Test ist sehr schnell, aber nicht sehr zuverlässig, weil ihm zu viele zusammengesetzte Zahlen durch die Maschen schlüpfen, und zwar auch noch dann, wenn man ihn zu verschiedenen Basen a wiederholt. Besonders drastisch wird dies durch die *Carmichael-Zahlen* demonstriert: Dies sind die Zahlen Zahl m, bei denen $a^{m-1} \equiv 1 \ \ (m)$ für alle zu mv teilerfremden a gilt. Die kleinste Carmichael-Zahl ist 561, und es ist seit einigen Jahren bekannt, daß es unendlich viele Carmichael-Zahlen gibt.

Ein wesentlich besserer Test, der kaum mehr Aufwand erfordert, ist der *Rabin-Miller-Test*. Man führt ihn für die Zahl n folgendermaßen durch:

(i) Man zerlegt $n-1 = q \cdot 2^u$, wobei q ungerade ist.
(ii) Man bestimmt für ein $a \in \mathbb{Z}$, das von n nicht geteilt wird, die Potenzen $a^q, (a^q)^2, \ldots, (a^q)^{2^u}$, wobei man aber stoppt, wenn $(a^q)^{2^v} \equiv 1 \ (n)$.
(iii) n ist keine Primzahl, wenn einer der folgenden Fälle eintritt:

$$(1) \quad a^{n-1} \not\equiv 1 \ (n), \qquad (2) \quad v \geq 1, \ (a^q)^{2^{(v-1)}} \not\equiv -1 \ (n).$$

Im Fall (1) besteht n ja nicht einmal den Fermat-Test zur Basis a. Im Fall (2) gilt für $x = (a^q)^{2^{(v-1)}}$, daß $x^2 \equiv 1 \mod n$, aber $x \not\equiv \pm 1 \ (n)$ ist. Dies ist aber unmöglich, wenn n eine Primzahl ist:

$$x^2 - 1 \equiv (x-1)(x+1) \equiv 0 \ (n)$$

impliziert, daß $x-1$ oder $x+1$ von n geteilt wird.

Es gilt folgender Satz, auf dessen Beweis wir hier verzichten (siehe [**Fo**]):

Satz 16.7. *Ist n keine Primzahl, so beträgt die Anzahl der Restklassen a, zu denen n den Rabin-Miller-Test übersteht, höchstens $n/4$.*

Man kann sich also mit an Sicherheit grenzender Wahrscheinlichkeit annehmen, daß n prim ist, wenn es den Test zu 10 Basen zufällig gewählten Basen a besteht. Es gibt jedoch auch einen etwas aufwendigeren, aber immer noch „schnellen" Test (Adleman-Pomerance-Rumely, vereinfacht von Cohen und Lenstra), mit dem man wirklich entscheiden kann, ob n prim ist. (Wir werden gleich noch präzisieren, was hier mit „schnell" gemeint ist.) Man kann Primzahlen p dann finden, indem man mit einer zufällig gewählten Zahl in der geforderten Größenordnung startet und diese systematisch erhöht, bis man eine Zahl erreicht hat, die den Test besteht. Der Primzahlsatz zeigt, daß die Primzahlen dicht genug liegen: die Dichte der Primzahlen „bei x" beträgt $1/lnx$. Da $\ln 2^{1024} \approx 710$, ist etwa jede 710-te Zahl in der Nähe von 2^{1024} prim. Wenn man sich auf die ungeraden Zahlen beschränkt, muß man also etwa 355 Zahlen ausprobieren, bis man auf eine Primzahl gestoßen ist. Viele der Kandidaten scheitern schon an der Probedivision durch „kleine" Primzahlen (etwa $< 10^6$), und auf die verbleibenden wendet man den Rabin-Miller-Test (oder sogar den Adleman-Pomerance-Rumely-Test) an. Diese Überlegung zeigt zudem, daß es genug Primzahlen für RSA gibt: die Gefahr, daß zwei Teilnehmer am System den gleichen RSA-Modul bilden, ist verschwindend gering.

RSA als Blockchiffre. Es ist nicht schwer, das RSA-System als eine gespreizte Blockchiffre zu implementieren. Wir gehen davon aus, daß der zu chiffrierende Klartext als eine Folge von Bytes anfällt. Sei m der gewählte RSA-Modul. Dann setzen wir

$$n = \lfloor \log_{256} m \rfloor = \left\lfloor \frac{1}{8} \log_2 m \right\rfloor.$$

Jede Folge $B_1, \ldots, B_n$ von Bytes stellt dann im Dualsystem eine Zahl a mit $0 \leq a \leq m-1$ dar, und umgekehrt kann jede solche Zahl durch eine Folge von $n+1$ Bytes

repräsentiert werden. Speziell gilt das für die Potenz a^e mit dem Verschlüsselungs-Exponenten e. Damit realisiert RSA also eine gespreizte Blockchiffre

$$\{0,\ldots,255\}^n \to \{0,\ldots,255\}^{n+1}$$

(Hat das Alphabet des Klartexts eine andere Länge N, so ist 256 überall durch N zu ersetzen.) Insbesondere lassen sich die Betriebsmodi ECB und (mit einfachen Änderungen auch) CBC für RSA einsetzen. CFB und OFB sind für Public-Key-Systeme natürlich ungeeignet, da sie für die Entschlüsselung das öffentlich bekannte Verschlüsselungsverfahren benutzen.

Effizienz von RSA. Ein Nachteil von RSA ist der hohe Rechenaufwand, der für die Ver- und die Entschlüsselung notwendig ist. Für beide ist jeweils eine Exponentiation modulo des RSA-Moduls m notwendig. Man kann den Aufwand bei der Verschlüsselung reduzieren, indem man einen kleinen Exponenten e wählt. Dabei ist jedoch einige Vorsicht geboten, wie wir gleich noch sehen werden. Als ein guter Wert für die Praxis gilt die (prime!) Zahl $e = 2^{16}+1$. Dann werden 16 Quadrierungen modulo m benötigt und 1 zusätzliche Multiplikation.

Der Entschlüsselungs-Exponent d hat aber unvermeidlich die Größenordung von m. (Es wäre gefährlich, ihn auf Kosten von e klein zu machen.) Wenn k Bits für die Darstellung von m erforderlich sind, werden in der Darstellung von d etwa $k/2$ Bits den Wert 1 haben. Damit sind k Quadrierungen und $k/2$ Multiplikationen modulo m erforderlich. Man kann die Entschlüsselung mit dem chinesischen Restsatz beschleunigen, indem man simultan modulo p und modulo q rechnet. Da Multiplikationen modulo der (im Vergleich zu m) etwa halb so langen Zahlen p und q bei Verwendung der „Schulmultiplikation“ nur etwa $1/4$ des Aufwands einer Multiplikation modulo m erfordern, beschleunigt sich die Entschlüsselung um etwa den Faktor 2. (Die für die Lösung der simultanen Kongruenz notwendigen Produkte py, qz mit $1 = py + qz$ können vorab berechnet werden.)

Sind große Datenmengen zu chiffrieren, so verwendet man ein gutes symmetrisches Verfahren und benutzt das RSA-System, um mit ihm die Schlüssel des symmetrischen Verfahrens auszutauschen. Public-Key-Systeme sind auch zunächst (als die Computer noch wesentlich langsamer waren) für den Schlüsseltausch entwickelt worden. Zum Beispiel nutzt die bekannte PGP-Chiffrierung von E-Mail das RSA-system, um mit ihm Schlüssel für die symmetrische IDEA-Chiffre (siehe dazu [**Scn**]) zu tauschen. Die IDEA-Chiffre wird dann im CFB-Modus eingesetzt.

Schlüsselverwaltung. Wie bei allen Public-Key-Systemen muß die Zuordnung von öffentlichem Schlüssel zu dessen Eigner fälschungssicher sein. Dies wird über sogenannte „Trust-Center“ oder „Key-Distribution-Center“ organisiert. Wir verweisen auf [**Bu**] oder [**Scn**] für eine eingehende Diskussion der damit verbundenen logistischen und Sicherheitsprobleme.

Neben diesem öffentlich organisierten Teils des Kryptosystems muß jeder Nutzer seine geheimen Schlüssel zuverlässig ablegen. Für die Verschlüsselung von E-Mail mag es genügen, die geheimen Schlüssel auf der eigenen Festplatte zu speichern, wo sie natürlich wiederum verschlüsselt liegen und nur bei Benutzung mittels eines Kennworts aktiviert werden. Bei höheren Ansprüchen an die Sicherheit wird der gesamte Umgang mit dem Public-Key-system auf Chipkarten abgewickelt, die sich von außen nur sehr schwer manipulieren lassen. Die Chipkarten dienen dann nur der Schlüsselvereinbarung für eine symmetrische Chiffre, und die Arbeit mit der symmetrischen Chiffre wird auf dem PC oder einem anderen leistungsfähigen Rechner durchgeführt.

Faktorisierung des RSA-Moduls. Das RSA-System ist gebrochen, wenn es jemandem gelingt, die Zahl m_A in ihre Teiler p und q zu zerlegen. Um die Gefahr abzuschätzen, die von den bekannten Faktorisierungsverfahren ausgeht, müssen wir den Rechenaufwand, den diese erfordern präzisieren. Für zwei reellwertige Funktionen f und g, die auf einem Intervall $[t_0,\infty)$ definiert sind, schreiben wir

$$f = O(g) \quad \text{oder} \quad f(x) = O(g(x))$$

wenn es eine Konstante C gibt mit $f(x) \le Cg(x)$ für alle genügend großen x. Die naive Probedivision muß (bis auf einen konstanten Faktor) alle Zahlen bis $\sqrt{m}$ ausprobieren, um mit Sicherheit einen Faktor von n zu finden. Sie hat Laufzeit $O(\sqrt{m})$.

Man beachte bei dieser Schreibweise, daß das Gleicheitszeichen hier für eine Ungleichung mißbraucht wird. Es gilt zwar $x = O(x^2)$, aber nicht $x^2 = O(x)$.

Die übliche Bezugsgröße für Faktorisierungsverfahren und Primzahltests ist aber nicht m selbst, sondern die Anzahl der Ziffern von m, also $\ln m$ (bis auf eine Konstante). Algorithmen gelten als effizient, wenn ihre Laufzeit polynomial von $\ln m$ abhängt, d. h. von der Größenordnung

$$O\big((\ln m)^n\big)$$

für ein $n \in \mathbb{N}$ ist. In der Praxis darf man natürlich die Rolle des Grades n und der beteiligten Konstanten nicht unterschätzen.

Algorithmen gelten als ineffizient, wenn ihre Laufzeit exponentiell von $\ln m$ abhängt, d. h. von der Größenordnung

$$O\big(e^{c\ln m}\big)$$

für eine Konstante $c > 0$ ist. Die Probedivision hat exponentielle Laufzeit mit $c = 1/2$.

Die besten bekannten Faktorisierungsverfahren, wie etwa das subexponentielle Laufzeit

$$O\Big(e^{v(\ln m)^u(\ln\ln m)^{1-u}}\Big)$$

mit $0 < u < 1$ und einer Konstanten v. (Für $u = 0$ wird die Laufzeit polynomial, für $u = 1$ exponentiell.)Beim quadratischen Sieb ist $u = 1/3$,beim Zahlkörpersieb $u = 1/3$. Diese Abschätzungen beruhen allerdings auf unbewiesenen, wenn auch sehr plausiblen zahlentheoretischen Aussagen.

Manche der Algorithmen, wie die Faktorisierung mit elliptischen Kurven, sind probabilistisch: sie finden in der angegebenen Laufzeit fast immer, aber nicht mit Sicherheit einen Faktor von m. Für Einzelheiten zu Faktorisierungsverfahren verweisen wir auf [**Co**], [**Ko1**] und [**Fo**]. Ein Beispiel eines probabilistischen Algorithmus werden wir im nächsten Abschnitt kennenlernen.

Man darf bei der Bestimmung der Laufzeit nicht nur die Verfahrensschritte zählen, sondern muß deren Zahl mit der Laufzeit jeden Schritts multiplizieren. Eine Addition modulo m erfordert den Aufwand $O(\ln m)$. Der Aufwand für eine Multiplikation nach der „Schulmethode“ ist $O((\ln m)^2)$. Ist aber die Zahl der Verfahrensschritte subexponentiell oder exponentiell, so spielen polynomiale Faktoren für die Größenordnung der Laufzeit keine wesentliche Rolle.

Zum Vergleich die Laufzeiten der Primzahltests: Die einmalige Ausführung des Rabin-Miller-Tests hat Laufzeit $O((\ln m)^3)$, der Adleman-Pomerance-Rumely-Test hat Laufzeit $O\big((\ln m)^{c \ln\ln\ln m}\big)$ mit einer Konstanten c und ist damit auch fast polynomial.

Um RSA-Moduln m gegen Faktorisierungsverfahren abzusichern, sollten ihre Primfaktoren noch gewisse Nebenbedingungen erfüllen. Zum Beispiel sollten $p-1$ und $q-1$ einen möglichst kleinen ggT und jeweils mindestens einen großen Primfaktoren besitzen. Sehr gut ist die Wahl von “sicheren“ Primzahlen p und q, für die auch $p' = (p-1)/2$ und $q' = (q-1)/2$ prim sind.

Man beachte, daß jeder Klartext n mit $p \mid n$ oder $q \mid n$ zu einem Geheimtext c verschlüsselt wird, der ebenfalls von p oder q geteilt wird. Durch Bilden von $\mathrm{ggT}(m,c)$ erhält man dann einen Faktor von m. Die Gefahr, daß solche Klartext getroffen wird, ist natürlich mikroskopisch klein. Man kann sie aber systematisch vermeiden und rechnet dann in $\mathbb{Z}_m^*$. Insofern spielt Satz 16.4 keine wesentliche Rolle, denn für a teilerfremd zu m ist die Kongruenz $a^{\varphi(m)+1} \equiv a \ \ (m)$ immer richtig. Die Wahl von m als Produkt von *zwei* Primzahlen ist aber der beste Schutz vor der Faktorisierung.

Man kann allerdings bisher nicht beweisen, ob die Dechiffrierung wirklich die Zerlegung des RSA-Moduls erfordert. Daher ist nicht bekannt, ob das Brechen des RSA-Systems ebenso schwierig ist wie die Primfaktorzerlegung. Man kann hingegen beweisen, daß man bei Kenntnis des RSA-Moduls m und der beiden Exponenten e und d die Primfaktorzerlegung von m rasch finden kann (siehe [**Bu**], 7.2.4.) Für das Brechen des Systems ist das aber bei Kenntnis von d nicht mehr notwendig.

Spezielle Angriffe. Die von einem kleinen Verschlüsselungs-Exponenten e ausgehende Gefahr belegt das folgende Szenario. Die A-Bank verschickt an k Kunden den Klartext n und chiffriert ihn mit dem gleichen Verschlüsselungs-Exponenten e. Die RSA-Moduln $m_1, \ldots, m_k$ der Kunden sind mit hoher Wahrscheinlichkeit teilerfremd. Ist $k \geq e$, so läßt sich nun der Klartext n aus den Chiffretexten c_i bestimmen. Mittels des chinesischen Restsatzes löst man nämlich das System

$$x \equiv c_i \ (m_i), \quad i = 1, \ldots, e$$

Es hat genau eine Lösung c mit $0 \leq c \leq m = m_1 \cdots m_e$. Aber auch n^e erfüllt alle diese Bedingungen, und deshalb ist $x = n^e$. Man kann nun aus x die e-te Wurzel ziehen, und hat den Klartext n gewonnen. Bei großem e tritt die Bedingung $k \geq e$ seltener auf und außerdem wird die Zahl x astronomisch groß. Man kann ferner dafür sorgen, daß der gleiche Klartext nicht zweimal chiffriert wird, indem man der eigentlichen Nachricht noch zufällig gewählte Zeichen hinzufügt.

Eine zweite (sehr kleine) Gefahr besteht in der Existenz von Zahlen n kleiner Periode modulo m. Dabei heiße die kleinste Zahl k, für den $n^{k+1} \equiv n \mod m$ die *Periode* von n modulo m. (Da $\mathbb{Z}_m$ mit der Multiplikation keine Gruppe ist, sollten wir nicht von Ordnung sprechen.) Die Potenzen von m durchlaufen also nur k verschiedene Restklassen modulo n. Da c eine Potenz von n und n eine Potenz von c modulo m ist, hat auch c die gleiche Periode k. Es ist nicht schwer zu sehen, daß k ein Teiler von $(p-1)(q-1)$ ist.

Insbesondere ist e teilerfremd zu k und daher existiert ein w mit $e^w \equiv 1 \mod k$. Das kleinstmögliche w erfüllt $w \leq \varphi(k) \leq k-1$. Nun wenden wir die Verschlüsselung w-mal auf $c = m^e$ an und erhalten die Potenzen

$$c, c^e, \ldots, c^{e^w} \quad \text{mit} \quad c^{e^w} \equiv c^{vk+1} \equiv c \ (n).$$

Da $\left(c^{e^{w-1}}\right)^e \equiv c \ (n)$ und weil das Potenzieren mit e eine injektive Abbildung auf $\mathbb{Z}_m$ ist, folgt $m = c^{e^{w-1}}$. Wir haben also durch fortgesetzte Verschlüsselung aus c den Klartext gewonnen. Man sollte deshalb dafür sorgen, daß es möglichst wenige Elemente kleiner Periode in $\mathbb{Z}^m$ gibt. Grob gesprochen läuft dies darauf hinaus, daß es nur wenige kleine Teiler von $(p-1)(q-1)$ gibt.

Ein letzter „zahlentheoretischer“ Sicherheitsaspekt ist die Gleichung

$$(m_1 m_2)^e = m_1^e m_2^e.$$

Wer den Geheimtext zu m_1 und m_2 kennt, kann den Geheimtext zu $m_1 m_2$ erzeugen (ohne daß ihm diese Klartexte bekannt sein müßten). Man kann dem leicht dadurch begegnen, daß man nur bestimmte Klartexte zuläßt, so daß das Produkt $m_1 m_2$ mit großer Wahrscheinlichkeit kein zugelassener Klartext ist.

Übertragbarkeit von RSA. Sollte eines Tages ein Faktorisierungsverfahren gefunden werden, das das RSA-Kryptosystem wertlos macht, so könnte die Idee möglicherweise auf andere Monoide als $(\mathbb{Z}_m, \cdot)$ übertragen werden. RSA nutzt aus,

daß $\mathbb{Z}_m$ Vereinigung von Gruppen ist, deren Ordnungen alle $(p-1)(q-1)$ teilen. Wenn wir nämlich $\mathbb{Z}_m$ gemäß dem chinesischen Restsatz mit $\mathbb{Z}_p \times \mathbb{Z}_q$ identifizieren, ist

$$\mathbb{Z}_m = \left(\mathbb{Z}_p^* \times \mathbb{Z}_q^*\right) \cup \left(\mathbb{Z}_p^* \times \{0\}\right) \cup \left(\{0\} \times \mathbb{Z}_q^*\right) \cup \{(0,0)\}.$$

(Diese Gruppen haben allerdings die paarweise verschiedenen Einselemente $(1,1)$, $(1,0)$, $(0,1)$ und $(0,0)$.) Das Verfahren beruht letzten Endes darauf, daß wir zwar $\mathbb{Z}_m$ durch den öffentlichen Schlüssel kennen, die Zerlegung aber daraus nicht effektiv bestimmen können, und speziell nicht das kleinste gemeinsame Vielfache der Ordnungen. Zum Rechnen in $\mathbb{Z}_m$ nutzen wir aus, daß $\mathbb{Z}_m$ hinsichtlich der Addition eine zyklische Gruppe ist.

Übungen

16.8. Wir wählen den RSA-Modul $m = 13 \cdot 23$ und den Verschlüsselungsexponenten 5.

(a) Bestimme den Entschlüsselungsexponenten.

(b) Chiffriere die Nachricht „Mathematik ist schön.“, wobei jeder Buchstabe (inklusive des Leerzeichens) durch seinen ASCII-Code eine Zahl zwischen 0 und 255 darstellt. Der Geheimtext ist als Folge von Zahlen zwischen 0 und $m-1$ darzustellen.

16.9. (a) Breche das RSA-System mit dem öffentlichen Schlüssel $m = 536813567$, $e = 3602561$.

(b) Dechiffriere den Geheimtext BEHJCQAALSTFOV. Dieser ist in Blöcken der Länge 7 gegeben, und ein solcher Block repräsentiert eine im 26er-System geschriebene Zahl, wobei A die Ziffer 0, B die 1 usw. darstellt. Der Klartext hat entsprechend Blöcke der Länge 6.

16.10. Der *Exponent* einer endlichen Gruppe G ist das kleinste $n \in \mathbb{N}$ mit $g^n = e$ für alle $g \in G$. Offensichtlich ist $\exp(G)$ das kleinste gemeinsame Vielfache der Ordnungen der Elemente von G.

(a) Man kann zeigen, daß es in abelschen Gruppen sogar ein Element gibt, dessen Ordnung der Exponent der Gruppe ist. (Das braucht hier nicht bewiesen zu werden.) Gilt dies auch in nichtabelschen Gruppen?

(b) G und H seien endliche Gruppen mit den Exponenten m und n. Zeige: Der Exponent von $G \times H$ ist $\mathrm{kgV}(m,n)$.

(c) Welchen Exponenten hat $\mathbb{Z}_m^*$, wenn $m = pq$ Produkt von Primzahlen $p \neq q$ ist? (Dazu muß man natürlich den Exponenten von $\mathbb{Z}_p^*$ kennen.)

(d) Welches ist die kleinste Zahl $v > 0$ mit $z^{v+1} = z$ für alle $z \in \mathbb{Z}_m$? (Bezeichnungen wie in (c)).

16.11. Man darf im RSA-System den Klartext n nicht zum gleichen RSA-Modul m mit teilerfremden Exponenten e und f verschlüsseln. Der *Common-Modulus-Angriff* bestimmt dann nämlich n aus den Geheimtexten n^e und n^f. Wie?

16.12. Wieviele Primzahlen, sichere oder sogar sehr sichere Primzahlen gibt es zwischen 10^6 und $2 \cdot 10^6$?

Bei Zahlen dieser Größenordnung kann man mit Probedivision arbeiten. In `Aribas` gibt es die Funktion `prime32test(x: integer)`, die für jede Zahl bis 2^{32} genau dann den Wert `true` liefert, wenn x prim ist.

ABSCHNITT 17

Diskrete Logarithmen

Das RSA-System beruht darauf, daß die Kongruenz

$$c = n^e \ \ (m)$$

sich ohne Kenntnis der Primfaktorzerlegung von m nur sehr schwer nach n auflösen läßt. Aber selbst wenn $m = p$ eine Primzahl ist, kann man nur sehr schwer aus n und c den diskreten Logarithmus e von $c \bmod m$ bezüglich n finden. Daher ist auch der diskrete Logarithmus kryptographisch nutzbar. Er läßt sich allgemein in Gruppen definieren.

Definition. Sei G eine Gruppe und $g \in G$. Man nennt dann e den *(diskreten) Logarithmus* von g^e bezüglich e.

Hat g endliche Ordnung n, so gilt $g^e = g^r$ für jedes r mit $r \equiv e(n)$, und wir verlangen zur Normierung, daß $0 \leq e < n$. Wir werden im folgenden Kryptosysteme diskutieren, bei denen $G = \mathbb{Z}_p^*$ für eine Primzahl p ist. Man kann und hat diese Systeme aber auch auf andere Gruppen übertragen. Wir kommen darauf noch zurück.

Primitivwurzeln. Sei nun p eine Primzahl. Die Struktur der Gruppe $\mathbb{Z}_p^*$ ist gut bekannt:

Satz 17.1. *Für jede Primzahl p ist $\mathbb{Z}_p^*$ eine zyklische Gruppe. Es gibt also ein $g \in \mathbb{Z}$, für das zu jedem $a \in \mathbb{Z}$, $p \nmid a$, ein e mit $a \equiv g^e \ \ (p)$ existiert.*

Man nennt jedes g, dessen Restklasse $\mathbb{Z}_p^*$ erzeugt, eine *Primitivwurzel* modulo p. Satz 17.1 ist nur ein Spezialfall des allgemeineren Satzes, nach dem endliche Untergruppen der Einheitengruppen K^* von Körpern K stets zyklisch sind.

Um zu entscheiden, ob g eine Primitivwurzel modulo p ist, kann man folgendes Kriterium anwenden:

Satz 17.2. *Sei p eine Primzahl. Genau dann ist $g \in \mathbb{Z}$, $p \nmid g$, eine Primitivwurzel modulo p, wenn*

$$g^{(p-1)/q} \not\equiv 1 \ \ (p)$$

für alle Primteiler q von $p-1$.

Dies ist leicht einzusehen. Es gilt ja

$$\operatorname{ord}_{\mathbb{Z}_p^*} q \mid p-1,$$

und wenn die Bedingung von Satz 17.2 erfüllt ist, kann die Ordnung kein echter Teiler von $p-1$ sein. Man braucht also die Primfaktoren von $p-1$, um Primitivwurzeln modulo p zu finden. Im allgemeinen wird die Faktorisierung von $p-1$ jedoch nicht möglich sein. Es bieten sich zwei Auswege:

(a) Man verzichtet darauf, eine Primitivwurzel zu finden. Dann arbeiten die im folgenden diskutierten Verfahren immer noch korrekt – man verkleinert aber den Schlüsselraum.
(b) Man sucht eine Primzahl p, bei der man $p-1$ faktorisieren kann. Das ist zum Beispiel der Fall, wenn $p-1$ außer einem „großen“ Primfaktor nur „kleine“ Primfaktoren (etwa $< 10^6$) besitzt. Letztere kann man aus $p-1$ herausziehen und den komplementären Teiler einem Primzahltest unterwerfen.

Die Zahl der Primitivwurzeln modulo p beträgt $\varphi(p-1)$. Ihr Anteil an den Restklassen modulo p ist also groß genug, so daß man durch Ausprobieren von Kandidaten bald eine Primitivwurzel findet.

Diffie-Hellman-Schlüsseltausch. Dieses wohl erste veröffentlichte Public-Key-Verfahren überhaupt, erfolgt so:

(1) Die Teilnehmerin Alice (ein Standard-Name in der kryptographischen Literatur) wählt eine Primzahl p, eine Primitivwurzel $g \mod p$ und eine Zahl a, $0 \le a < p-1$. Sie bestimmt $A = g^a \mod p$.
(2) Um einem Schlüssel mit Bob zu vereinbaren, sendet sie ihm die Daten

$$(p, g, A).$$

(3) Bob wählt nun eine Zahl b, $0 \le b < p-1$ bestimmt $B = g^b$, und sendet B an Alice zurück.
(4) Der vereinbarte Schlüssel ist

$$K = B^a = A^b.$$

Dabei können p und g vorab zum öffentlich zugänglichen Schlüssel von Alice gemacht werden. Dann braucht nur A an Bob übermittelt werden. Die Zahl a muß Alice natürlich geheimhalten – sie öffnet die Falltür für ihre Bestimmung von K. Gleiches gilt für b auf Bobs Seite. Wenn g keine Primitivwurzel ist, verringert sich der effektive Schlüsselraum auf die Restklassen modulo $\operatorname{ord}_{\mathbb{Z}_p^*} g$.

Die Sicherheit des Verfahrens beruht darauf, daß sich der diskrete Logarithmus von A bezüglich g nur schwer bestimmen läßt. Es wäre allerdings auch möglich, daß ein Weg gefunden wird, g^{ab} aus g^a und g^b zu bestimmen, ohne a oder b zu berechnen. Die Äquivalenz dieses „Diffie-Hellman-Problems“ zu dem der Bestimmung des diskreten Logarithmus ist nicht bekannt.

Das ElGamal-Kryptosystem. Mit einem Zusatz kann man den Diffie-Hellman-Schlüsseltausch zum ElGamal-Kryptosystem ausbauen. Die Schritte (1) und (2) verlaufen wie oben. Danach ist

$$(p, g, A)$$

der öffentliche Schlüssel von Alice. Nun möchte Bob einen Klartext m, $b \leq m \leq p-1$, an Alice schicken. Statt nur B an Alice zu senden, mutipliziert er den Klartext n mit dem Schlüssel K und schickt das Paar (B, c) mit $c \equiv Kn \ (p)$ an Alice. Zum Entschlüsseln muß Alice den Chiffretext mit $K^{-1} \mod p$ multiplizieren. Das Inverse von K kann man einfach ducrh Potenzieren von B finden, denn es gilt:

$$B^{p-1-a}K \equiv g^{(p-1)b-ab}g^{ab} \equiv 1 \ (p).$$

Die Schritte (3) und (4) sind also wie folgt zu erweitern:

(3') Bob wählt wieder ein b mit $0 \leq b < p-1$ und bestimmt $B = g^b \ (p)$. Dann berechnet er $K = A^b \ (p)$ und

$$c = Kn \ (p)$$

und sendet (B, c) an Alice, die folgendermaßen entschlüsselt:

(4') Sie setzt $u = p-1-a$ und bestimmt

$$B^u c = g^{b(p-1-a)}g^{ab}n = g^{b(p-1)}n \equiv m \ (p).$$

Auch die Sicherheit des ElGamal-Verfahrens beruht auf der Schwierigkeit, diskrete Logarithmen zu bestimmen: Es reicht aus, a zu bestimmen, wenn man das System brechen will.

Ein Nachteil gegenüber RSA ist, daß der Geheimtext (B, c) doppelt so lang ist, wie der Klartext m. Beim Chiffrieren benötigt ElGamal wie RSA eine modulare Exponentiation. Allerdings ist g^b von Klartext unabhängig und kann vorab berechnet werden. Dann braucht die eigentliche Chiffrierung nur eine Multiplikation. Beim Dechiffrieren hat hingegen RSA die Nase vorn: Auch ElGamal benötigt eine Exponentiation modulo p, die nun aber nicht mittels des chinesischen Restsatzes vereinfacht werden kann.

Als Vorteil von ElGamal ist zu sehen, daß die zufällige Wahl von b dafür sorgt, daß der gleiche Klartext nicht zweimal auf die gleiche Weise verschlüsselt wird. (Eine solche „Randomisierung“ kann man RSA natürlich auch noch hinzufügen.)

Übertragbarkeit von ElGamal. Der entscheidende Vorteil von ElGamal gegenüber RSA besteht allerdings in seiner Übertragbarkeit auf andere Gruppen G. Das setzt nur voraus, daß man in G Elemente g finden kann, für die die Bestimmung des diskreten Logarithmus bezüglich g „schwer“ ist. Man benötigt nicht einmal, daß die Gruppe G abelsch ist. Der Schlüsselraum besteht dann aus den Potenzen von G, sowohl für Diffie-Hellman als auch für ElGamal. Als Kandidaten für G kommen in Frage:

(1) Einheitengruppen endlicher Körper (sie sind zyklisch),
(2) Idealklassengruppen imaginär-quadratischer Zahlkörper,
(3) elliptische Kurven über endlichen Körpern (oder die Jacobi-Varietäten hyperelliptischer Kurven).

Alle diese Gruppen sind abelsch. Die Kryptographie mit elliptischen Kurven wird in [**Bu**] und [**Ko1**] diskutiert.

Das Geburtstagsparadoxon. Es soll nun wenigstens ein Algorithmus besprochen werden, mit dem man diskrete Logarithmen bestimmen kann, das *ρ-Verfahren von Pollard*.

Es beruht darauf, daß man in einer zufällig gewählten Folge

$$x_1, \ldots, x_k, x_{k+1}, \ldots$$

in einer endlichen Menge M mit n Elementen überraschend schnell Indizes $i < j$ findet, für die $x_i = x_j$ ist. Eine gängige Einkleidung dieses Sachverhalts ist das „Geburtstagsparadoxon“: In einem Raum mögen sich k Personen befinden; wie groß ist die Wahrscheinlichkeit, daß wenigstens zwei von ihnen am gleichen Tag Geburtstag haben? In diesem Fall ist $n = 365$ und die Wahrscheinlichkeit dafür, daß alle k Geburtstage verschieden sind, ist

$$p_{365}(k) = \frac{365 \cdot 364 \ldots (365-k+1)}{365^k}.$$

Die Wahrscheinlichkeit dafür, daß zwei Geburtstage übereinstimmen, beträgt also

$$1 - p(k),$$

und es gilt $1 - p(k) \geq 1/2$ für alle $k \geq 23$!

Allgemein ist

$$p_n(k) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right).$$

Logarithmieren ergibt

$$p_n(k) = \sum_{i=1}^{k-1} \ln\left(1 - \frac{i}{n}\right)$$

und mit der Abschätzung

$$\ln 1 - x \approx -x$$

folgt

$$\ln p_n(k) \approx -\sum_{i=1}^{k-1} \frac{i}{n} = \frac{k(k-1)}{2n}.$$

Also ist

$$p_n(k) \approx \exp\left(-\frac{k(k-1)}{2n}\right)$$

und $p_n(k) \leq 1/2$, wenn

$$\frac{k(k-1)}{2n} \geq \ln 2.$$

Dies führt auf die Bedingung

$$k \geq k_0 \approx 1,2\sqrt{n}.$$

Da $-x \geq \ln 1 - x$, sind wir mit der Abschätzung auf der sicheren Seite. Man kann auch die mittlere Wartezeit bis zur ersten Wiederholung asymptotisch bestimmen und erhält dafür

$$\sqrt{\frac{\pi}{2}}\sqrt{n} \approx 1.25\sqrt{n}$$

(siehe [**Co**], 8.5.4).

Wiederholungen in iterativen Folgen. Wir betrachten nun eine Funktion $f : M \to M$ und bestimmen ausgehend von einem Startwert x_0 die Folge

$$x_{k+1} = f(x_k), \quad k \geq 1.$$

Seien nun u und v minimal gewählt, so daß $u < v$ und

$$x_u = x_v.$$

Dann bilden $x_0, \ldots, x_{u-1}$ die *Vorperiode* und von x_u wiederholt sich die *Periode* $x_u, \ldots, x_{v-1}$ immer wieder. Wenn man dieses Verhalten durch einen Polygonzug in der Ebene repräsentiert, erhält man eine Figur ähnlich dem griechischen Buchstaben ρ. Daher der Name des Verfahrens.

Man kann nicht beweisen, daß die im folgenden benutzten Folgen $x_{k+1} = f(x_k)$ wirklich Zufallsfolgen sind, so daß die Anwendung der obigen Überlegung auf wackligen Füßen steht. Die empirischen Resultate rechtfertigen aber unser Vorgehen.

Will man wirklich das erste Auftreten einer Wiederholung finden, so muß man die gesamte Folge speichern und x_{k+1} mit $x_0, \ldots, x_k$ vergleichen. Der Speicheraufwand dafür würde das Verfahren unbrauchbar machen. Statt dessen bildet man zwei Folgen

$$\begin{aligned} x_0, &\quad x_{k+1} = f(x_k), \\ y = x_0, &\quad y_{k+1} = f(f(y_k)), \qquad k \geq 1 \end{aligned}$$

und vergleicht x_k mit y_k. Auch so findet man mit Sicherheit eine Wiederholung (wieso?), wenn auch nicht unbedingt die erste und um den Preis von drei Auswertungen von f pro Verfahrensschritt.

Eine andere Methode, Wiederholungen zu finden, besteht darin, daß man jeweils die Werte x_{2^j} speichert und dann die x_k, $2^j < k \leq 2^{j+1}$ mit ihnen vergleicht. Sobald $j = 2^{k+1}$ erreicht ist, ersetzt man den gespeicherten Wert durch $x_{2^{j+1}}$. Man braucht dann nur eine Auswertung von f pro Verfahrensschritt, muß im Mittel

aber länger auf die erste erkannte Wiederholung warten. (Für eine effizientere Implementation siehe [**Co**], 8.5.)

Das ρ-Verfahren. Es gibt das ρ-Verfahren von Pollard für die Faktorisierung und für den diskreten Logarithmus. Letzteres soll nun diskutiert werden. (Siehe Aufgabe 17.8 für das Faktorisierungsverfahren.)

Gesucht ist der diskrete Logarithmus a von $z \mod p$ bezüglich des Elementes $g \mod p$. Dabei sei p eine Primzahl. (Das Verfahren läßt sich leicht auf andere Gruppen G statt $\mathbb{Z}_p^*$ übertragen.) Wir teilen $\mathbb{Z}_p^*$ in drei disjunkte Teilmengen G_1, G_2, G_3 ein, zum Beispiel

$$\begin{aligned} G_1 &= \{x \mod p : 1 \le x < p/3\}, \\ G_2 &= \{x \mod p : p/3 \le x < 2p/3\}, \\ G_3 &= \mathbb{Z}_p^* \setminus (G_1 \cup G_2). \end{aligned}$$

Mit $r_0 \in \{1, \ldots, p-1\}$ und $b_0 = g^{r_0}$ starten wir die Iteration

$$b_{k+1} = f(b_k),$$

wobei

$$f(b) = \begin{cases} gb, & b \in G_1, \\ b^2, & b \in G_2, \\ zb, & b \in G_3. \end{cases}$$

Es gilt

$$b_k = g^{r_k} z^{s_k}$$

mit

$$r_0 \text{ wie gegeben}, \qquad r_{k+a} = \begin{cases} r_k + 1 \mod p-1, & b_k \in G_1, \\ 2r_k \mod p-1, & b_k \in G_2, \\ r_k \mod p-1, & b_k \in G_3, \end{cases}$$

$$s_0 = 0, \qquad s_{k+1} = \begin{cases} s_k \mod p-1, & b_k \in G_1, \\ 2s_k \mod p-1, & b_k \in G_2, \\ s_k + 1 \mod p-1, & b_k \in G_3. \end{cases}$$

Es gelte nun $b_u = b_v$. Dann ist

$$g^{r_u} z^{s_u} \equiv g^{r_v} z^{s_v} \ (p),$$

und nach Einsetzen von $a = g^z$ erhält man

$$g^{r_u + a s_u} \equiv g^{r_v + a s_v} \ (p).$$

Folglich muß

$$r_u + a s_u \equiv r_v + a s_v \ (p-1),$$

also

$$a(s_u - s_v) \equiv r_v - r_u \ \ (p-1)$$

sein.

Diese Kongruenz löst man folgendermaßen nach a auf. Wir setzen $r = s_u - s_v$, $s = r_v - r_u$ und $d = \mathrm{ggT}(r, p-1)$. Mit dem erweiterten euklidischen Algorithmus erhält man Eine Darstellung

$$d = xr + y(p-1).$$

Da unsere Kongruenz $ar \equiv s \ \ (p-1)$ eine Lösung a besitzt, ist d ein Teiler von s, etwa $s = kd$, und wir erhalten

$$a = kx$$

als eine Lösung. Die weiteren Lösungen sind

$$kx + i\frac{p-1}{d} \mod p-1, \quad i \in \mathbb{Z},$$

denn genau die Vielfachen von $(p-1)/d$ annullieren $r \mod p-1$. Es gibt also exakt d Lösungen der Kongruenz modulo $p-1$.

Im Fall $d = 1$ ist der diskrete Logarithmus gefunden. Wenn $d > 1$, aber nicht zu groß ist, kann man die Kandidaten für a ausprobieren. Andernfalls kann man das Verfahren mit einem anderen Startwert x_0 wiederholen (und/oder G_1, G_2, G_3 anders wählen). Die Wahl der Mengen G_i soll nur dafür sorgen, daß die Kongruenz, aus der a bestimmt wird, möglichst wenig Lösungen hat. Man kann auch die Iterationsvorschrift noch abändern, aber die obige Wahl ist insofern optimal, als die Bestimmung von $f(b)$ nur eine Multiplikation erfordert.

Die erwartete Laufzeit des ρ-Verfahrens ist $O(\sqrt{p})$, wobei wir unterstellen, daß sich die iterativ gebildete Folge zufällig verhält. Damit ist das ρ-verfahren sicher keine Gefahr für Primzahlen p der Länge 512 Bit, aber es zeigt, daß es für die Bestimmung des diskreten Logarithmus wesentlich bessere Methoden gibt, als das naive Ausprobieren.

Beispiel 17.3. Wir wollen den diskreten Logarithmus a mit

$$2^a \equiv 19 \ \ (80945413).$$

bestimmen. Es gilt $80945412 = 2^2 \cdot 3 \cdot 6745451$, und 2 ist Primitivwurzel modulo 80945413. Mit dem Startwert $x_0 = 71972261$ ergibt sich nach 9794 Iterationen ein Treffer. Zu lösen ist dann die Kongruenz

$$-41240980 \cdot a \equiv 73261968 \ \ (80945412).$$

Diese hat 4 Lösungen, weil der Faktor vor a und $p-1$ den größten gemeinsamen Teiler 4 haben, und man erhält $a = 76447284$.

Man vergleiche die Zahl der Iterationen mit $\sqrt{80945413} \approx 8996$.

Es gibt einige weitere Verfahren, für die Bestimmung des diskreten Logarithmus, so das Verfahren von Pohlig-Hellman und die Index-Calculus-Methode (siehe dazu [**Bu**] oder [**Co**]). Das Verfahren von Pohlig-Hellman zeigt, daß Primzahlen p gefährlich sind, wenn $p-1$ nur kleine Primfaktoren besitzt. Wie beim RSA-Verfahren erweist es sich auch hier als günstig, sichere Primzahlen zu benutzen (und für diese kann man Primitivwurzeln besonders schnell finden).

Übungen

17.4. Beweise daß die besprochenen Methoden, Wiederholungen in iterativ gebildeten Folgen $x_{k+1} = f(x_k)$ zu finden, nämlich

(i) Vergleich von x_k mit y_k, wobei $y_0 = x_0$ und $y_{k+1} = f(f(y_k))$, und

(ii) Vergleich von x_i, $2^k < i \leq 2^{k+1}$, mit x_{2^k},

wirklich erfolgreich sind. Vergleiche die Methoden!

17.5. Anna und Bernd wollen ihre amouröse Korrespondenz mittels eine Cäsar-Chiffrierung schützen. Die Länge der Cäsar-Verschiebung soll nach Diffie-Hellman in der Gruppe $\mathbb{Z}_{29}^*$ vereinbart werden. Ihr Alphabet hat nämlich 29 Zeichen, wobei die Zahlen $0,\ldots,25$ wie üblich die Großbuchstaben repräsentieren, und $26,27,28$ für die oft benutzten Wörter „Liebe“, „Herz“ und „Kuß“ eingesetzt werden.

(a) Bestimme die kleinste Primitivwurzel g modulo 29.

(b) Anna wählt ihren Schlüsselanteil $A = g^2 \mod 29$, Bernd den Anteil $B = g^{11} \mod 29$. Auf welchen Schlüssel einigen sie sich?

17.6. Nachdem die Cäsar-Chiffre aus der vorherigen Aufgabe von Annas Mutter gebrochen wurde, beschließen die Liebenden, zum ElGamal-System überzugehen. Die verwendete Primzahl ist $p = 863$, die Primitivwurzel ist $g = 5$.

Verschlüsselt werden Blöcke der Länge 2 im 29-er-System der vorherigen Aufgabe zu Zahlen c, $0 \leq c \leq 862$. Anna wählt ihren öffentlichen Schlüssel $A = 741$, und erhält von Bernd die Nachricht

$$(514, 355).$$

Diese Nachricht wird von Bernds Vater mitgehört, und es gelingt ihm (mit Hilfe seines beim BND arbeitenden Bruders) diese zu entschlüsseln. Wie lautete der Klartext?

17.7. Implementiere mittels `Aribas` das Pollardsche ρ-Verfahren zur Bestimmung des diskreten Logarithmus und bestimme den diskreten Logarithmus von $m = 1001$ bezüglich $g = 3$ modulo $p = 1234567891$.

17.8. Beim ρ-Verfahren von Pollard für Faktorisierung einer zerlegbaren Zahl m bildet man iterativ eine Folge

$$x_{k+1} \equiv f(x_k) \ \ (m)$$

mit einem Polynom f, zum Beispiel $f(x) = x^2 + 1$. Für jeden Primteiler p von m induziert diese eine Folge $(\overline{x}_k)$ modulo p. Im Falle $\overline{x}_i = \overline{x}_j$ gilt

$$\mathrm{ggT}(x_j - x_i, m) \mid m$$

und man kann hoffen, so einen nichttrivialen Teiler von m zu finden. Man bildet also fortlaufend

$$\mathrm{ggT}(y_k - x_k, m),$$

wobei y_k wieder die doppelt so schnell laufende Folge bezeichnet (oder $\mathrm{ggT}(x_k - x_{2^j}, m)$ für $2^j < k \leq 2^{j+1}$).

Die erwartete Laufzeit ist proportional zu $\sqrt{p}$, wobei p der kleinste Primteiler von m ist. Damit ergibt sich ein Rechenaufwand von $O(\sqrt[4]{m})$, da $m \geq p^2$.

Man implementiere diese ρ-Verfahren in `Aribas` und experimentiere damit.

ABSCHNITT 18

Hash-Funktionen

Hash-Funktionen ordnen Wörtern über einem Alphabet A eine „Prüfsumme“ oder einen Fingerabdruck fester‘“ Länge zu. Wenn wir voraussetzen, daß der Hash-Wert über dem gleichen Alphabet A gebildet wird, sind Hash-Funktionen vom Typ

$$h : A^* \to A^n.$$

Die Nützlichkeit von Hash-Funktionen können wir uns bei der Virensuche auf dem heimischen PC klarmachen. Der (erstaunlich schnell arbeitende) Virenscanner bildet bei der Suche nach Viren sukzessiv fortgeschriebene Hash-Werte der Daten in einem File. Wird dabei der Fingerabdruck eines Virus gefunden, kann die Fundstelle einer eingehenden Prüfung unterzogen werden. Ein besonders schlauer Virus könnte sein Auffinden zu verhindern suchen, indem er den Virenscanner oder dessen Datenbank selbst verändert. Dies kann man entdecken, indem man dem Virenscanner selbst einen Hash-Wert zuordnet und diesen an einem sicheren Ort oder verschlüsselt speichert. Findet der Virenscanner bei seinem Selbsttest den richtigen Hash-Wert an, können wir annehmen, daß er nicht verändert wurde.

Um in ähnlicher Weise eine zu übermittelnde Nachricht gegen Verfälschung zu sichern, bestimmt man zunächst ihren Hash-Wert und hängt ihn an die Nachricht an. Der Empfänger bestimmt ebenfalls den Hash-Wert. Stimmt dieser mit dem empfangenen Wort überein, so gilt die Nachricht als integer. Diese Absicherung erinnert uns an die Methoden der Codierungstheorie. Während es dort aber um die Absicherung gegen zufällige Störungen bei der Datenübertragung ging, muß man nun böswilligen Veränderungen oder Fälschungen vorbeugen. Daher wird der Hash-Wert einer Chiffrierung unterzogen. Das Verfahren für die Berechnung des Hash-Wertes zu einer Nachricht wird dabei in der Regel bekannt sein, aber der Schlüssel für die Chiffrierung muß natürlich geheim bleiben. Dann kann der Fälscher zwar Nachrichten und ihre Hash-Werte erzeugen, weiß aber nicht, wie die Hash-Werte zu chiffrieren sind.

Unter strengen mathematischen Maßstäben ist das Vertrauen in die Integrität des Virenscanners allerdings nicht gerechtfertigt. Eine Funktion $h : A^* \to A$ kann bei einem endlichen Alphabet nicht injektiv sein, und der Schluß $h(x) = h(x') \implies x = x'$ ist prinzipiell nicht zulässig. Man steht hier vor dem gleichen Problem wie bei der Identifizierung eines Straftäters anhand eines aller Wahrscheinlichkeit

nach nur einmal vorkommenden Merkmals, wie etwa des Fingerabdrucks oder des DNA-Profils.

Wörter x, x' nennt man eine *Kollision* von h, wenn $x \neq x'$, aber $h(x) = h(x')$ ist. Kollisionsfreiheit von Hash-Funktionen ist per Definition ausgeschlossen, aber man kann folgende Forderungen stellen:

(a) h heißt *schwach kollisionsresistent*, wenn es nur „sehr schwer“ möglich ist, zu einem gegebenen $x \in A^*$ ein $x' \in A^*$, $x \neq x'$, mit $h(x) = h(x')$ zu finden.

(b) h heißt *kollisionsresistent*, wenn es „sehr schwer“ ist, überhaupt Wörter x, $x' \in A^*$, $x \neq x'$, zu finden mit $h(x) = h(x')$.

Ebensowenig wie man beweisen kann, daß es wirklich Einweg-Funktionen gibt, kann man die Existenz von (schwach) kollisionsresistenten Hash-Funktionen nachweisen. Zur Präzisierung des Attributs „schwer“ müßte man (wie schon häufiger in der Vorlesung) die Komplexitätstheorie bemühen.

Die Grenzen der Forderung (b) zeigt das schon im vergangenen Abschnitt diskutierte Geburtstags-Paradoxon auf: Wenn wir die Hash-Werte $h(x_i)$ von $x_1, \ldots,$ $x_m \in A^*$ mit

$$m \geq 1{,}2\sqrt{|A^n|}$$

bilden, so ist die Wahrscheinlichkeit, daß wir unter $x_1, \ldots, x_m$ eine Kollision finden, bereits größer als $1/2$. Im Fall $A = \{0,1\}$, $n = 128$, müßte man also 2^{64} Wörter aus $\{0,1\}^*$ bilden, um mit ziemlicher Gewißheit eine Kollision zu finden. Obwohl dies auch heute noch einen enormen Zeit- und Speicheraufwand erfordert, gilt die Hash-Länge 128 nicht mehr als sicher und die standardisierte Hash-Funktion SHA (Secure Hash Algorithm), die wir noch besprechen werden, hat die Länge 160 Bit. Es wird aber bereits die Erweiterung auf 256, 384 oder gar 512 Bit diskutiert.

Kompressionsfunktionen. Hash-Funktionen werden in der Regel aus Kompressionsfunktionen

$$g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$$

konstruiert. Dazu füllt man ein Wort $x \in \{0,1\}^*$ zunächst mit Nullen auf eine durch n teilbare Länge auf und hängt außerdem die Dualdarstellung der Länge von x, wiederum auf eine durch n teilbare Länge aufgefüllt, an. Der letzte Schritt verhindert, daß zwei unterschiedliche Wörter x durch das Auffüllen identisch werden. Alternativ kann man als „Begrenzer“ der ursprünglichen Nachricht eine 1 anhängen und dann mit Nullen auf die erforderliche Länge auffüllen (oder beides tun).

Wir dürfen also jetzt annehmen, daß

$$x = x_1 \ldots x_m, \quad x_i \in \{0,1\}^n, \qquad i = 1, \ldots, m.$$

Man startet den Hash-Algorithmus mit einem Startwert, etwa $h_0 = (0, \ldots, 0) \in \{0,1\}^n$, und setzt

$$h_i = g(h_{i-1}, x_i), \qquad i = 1, \ldots, m.$$

Der x zugeordnete Hash-Wert ist dann h_m.

Man kann Blockchiffren mit Verschlüsselungsfunktionen E, deren Blocklänge mit der Schlüssellänge übereinstimmt, als Kompressionsfunktionen nutzen, zum Beispiel mittels

$$g(h,x) = E_h(x) \oplus x.$$

Hierfür gibt es auch noch andere Varianten, die als sicher gelten (siehe [**Scn**], p. 511). Hingegen ist die nächstliegende Wahl

$$g(h,x) = E_x(h)$$

unsicher, weil gegen sie ein „Meet-in-the middle"-Angriff möglich ist, dessen Prinzip wir in Abschnitt 14 schon studiert haben. Einer Nachricht aus 2 Blöcken x,y wird dann ja der Hash-Wert

$$h = E_y(E_x(h_0))$$

zugeordnet, und ein Schlüsselpaar (x,y), das dies für bekannte h_0 und h leistet ist mit der Methode aus Abschnitt 14 mit jeweils $2^{n/2}$ Auswertungen von E und der Entschlüsselungsfunktion D aufzufinden.

Secure Hash Algorithm. Als Beispiel einer in der Praxis eingesetzten Hash-Funktion beschreiben wir den standardisierten Secure Hash Algorithm (SHA). Wie bereits erwähnt, haben die erzeugten Hash-Werte eine Länge von 160 Bit.

Zum Auffüllen der Nachricht werden eine 1 angehängt und dann so viele Zeichen 0, daß die Länge der aufgefüllten Nachricht die Form $512m - 64$ hat. Danach wird die Länge der ursprünglichen Nachricht in 64 Bit Länge dual dargestellt und auch noch angehängt. Insgesamt entsteht eine Nachricht, deren Länge ein Vielfaches von 512 ist.

Fünf Variablen der Länge 32 Bit werden mit folgenden Startwerten in Hexadezimaldarstellung belegt:

$$A = \texttt{67452301}, \qquad B = \texttt{efcdab89}, \qquad C = \texttt{98badcfe},$$
$$D = \texttt{10325476}, \qquad E = \texttt{c3d2e1f0}.$$

Der Algorithmus wird nun iterativ auf die Nachricht angewandt, wobei in jedem Iterationsschritt ein Block der Länge 512 Bit abgearbeitet wird.

Die Hauptschleife wird in jedem Iterationsschritt 80-mal durchlaufen, wobei abhängig vom Schleifenzähler t vier verschiedene Funktionen zum Einsatz kommen:

$$f_t(X,Y,Z) = \begin{cases} (X \wedge Y) \vee (\overline{X} \wedge Z), & t = 0,\ldots,19, \\ X \oplus Y \oplus Z, & t = 20,\ldots,39, \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & t = 40,\ldots,59, \\ X \oplus Y \oplus Z, & t = 60,\ldots,79. \end{cases}$$

Dabei steht $\wedge$ für bitweises AND, $\vee$ für bitweises OR und $\overline{X}$ für das bitweise Komplement.

Im Algorithmus werden vier Konstanten benutzt (wiederum in Hexadezimaldarstellung):

$$K_t = \begin{cases} \texttt{5a827999}, & t = 0,\ldots,19, \\ \texttt{6ed9eba1}, & t = 20,\ldots,39, \\ \texttt{8f1bbcdc}, & t = 40,\ldots,59, \\ \texttt{ca62c1d6}, & t = 60,\ldots,79. \end{cases}$$

Diese Konstanten sind aus den ganzzahligen Anteilen gewisser Zahlen gewonnen:

$$\texttt{5a827999} = \left\lfloor 2^{32}\cdot 2^{1/2}/4 \right\rfloor, \qquad \texttt{6ed9eba1} = \left\lfloor 2^{32}\cdot 3^{1/2}/4 \right\rfloor,$$
$$\texttt{8f1bbcdc} = \left\lfloor 2^{32}\cdot 5^{1/2}/4 \right\rfloor, \qquad \texttt{ca62c1d6} = \left\lfloor 2^{32}\cdot 10^{1/2}/4 \right\rfloor.$$

Der Nachrichtenblock der Länge 512 Bit wird in 16 Wörter $M_0,\ldots,M_{15}$ der Länge 32 unterteilt und folgendermaßen in 80 Wörter $W_0,\ldots,W_{79}$ ebenfalls der Länge 32 expandiert:

$$W_t = M_t \qquad t = 0,\ldots,15$$
$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1 \qquad t = 16,\ldots,79.$$

Hier steht $\lll s$ für die zyklische Linksverschiebung um s Stellen. Diese Operation wird auch im folgenden benutzt.

Nun der Iterationsschritt des Algorithmus, in der $+$ für die Addition modulo 2^{32} steht:

Kopiere A,B,C,D,E nach a,b,c,d,e
Für $t = 0$ bis 79
  TEMP $= (a \lll 5) + f_t(b,c,d) + e + W_t + K_t$
  $e = d$
  $d = c$
  $c = b \lll 30$
  $b = a$
  $a =$ TEMP
Ende Schleife
Setze $A = A + a,\ldots,E = E + e.$

Abbildung 1 stellt einen Durchlauf der t-Schleife dar. SHA fährt dann mit dem nächsten Datenblock fort. Die Ausgabe ist die Konkatenierung von A,B,C,D,E zu einem Wort der Länge 160 Bit.

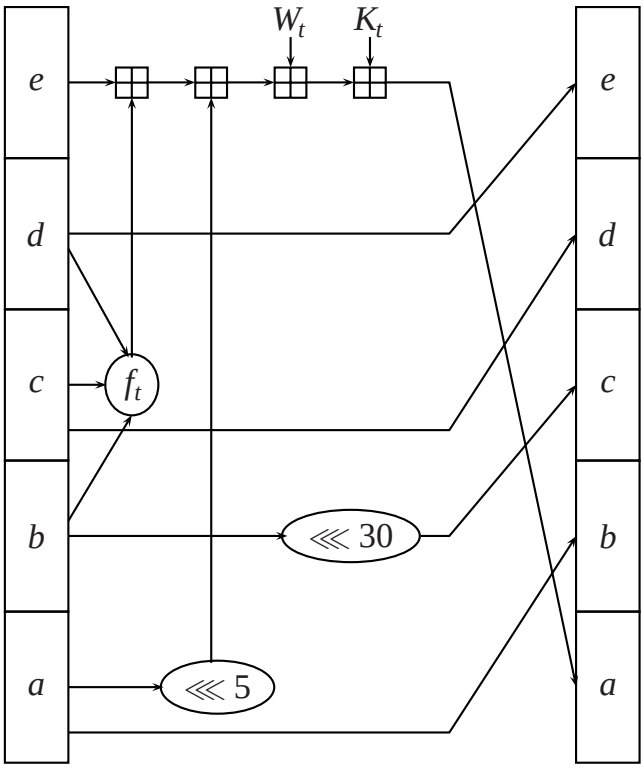


ABBILDUNG 1. Eine Runde von SHA

Eine Kompressionsfunktion mittels diskreter Logarithmen. Wenn man auch nicht nachweisen kann, ob eine „sichere“ Hash-Funktion überhaupt existiert, so kann man doch (nach Chaum, van Heist und Pfitzmann) eine Hash-Funktion angeben, die so sicher ist wie die auf diskreten Logarithmen beruhenden Kryptosysteme.

Wir wählen eine Primzahl p, für die $q = (p-1)/2$ ebenfalls prim ist und konstruieren eine Funktion

$$g : \{0,\ldots,q-1\} \times \{0,\ldots,q-1\} \to \{0,\ldots,p-1\}.$$

Diese ist zwar keine Kompressionsfunktion im strengen Sinn, kommt ihr aber nahe, weil $\log p \approx 1 + \log q$. Die Eingabe der binären Länge $2\log q$ wird also fast auf Länge $\log q$ komprimiert. Allerdings ist die Funktion g nur von theoretischem Interesse, da sie im Vergleich etwa zu SHA sehr langsam ist. Wir setzen

$$g(x,y) \equiv a^x b^y \ \ (p),$$

wobei a Primitivwurzel modulo p und $b \equiv a^e \ \ (p)$ zufällig gewählt ist.

Sei nun (u,v), (x,y) eine Kollision von g

$$a^u b^v \equiv a^x b^y \ (p),$$
$$b^{v-y} \equiv a^{x-u} \ (p),$$

und damit

$$a^{e(v-y)} \equiv a^{x-u} \ (p).$$

Wir sind nun in der gleichen Situation wie beim Pollardschen ρ-Verfahren. Es gilt

$$e(v-y) \equiv x-u \ (p-1)$$

und erst recht

$$e(v-y) \equiv x-u \ (q). \qquad (*)$$

Falls $x \neq u$ ist, folgt $x-u \not\equiv 0 \ (q)$. Dann aber ist auch $r-y \not\equiv 0 \ (q)$, denn q ist prim. Die Kongruenz $(*)$ hat eine eindeutig bestimmte Lösung $e_0 \mod q$, so daß

$$e \equiv e_0 \ (p-1) \quad \text{oder} \quad e \equiv e_0 + q \ (p-1),$$

und wir den richtigen Wert von e ausprobieren können.

Wenn $x = u$ ist, muß $e \equiv 0 \ (q)$ sein, weil dann $v \neq y$ gilt und somit $v-y \not\equiv 0 \ (q)$ ist.

Message Authentication Codes. Man kann schlüsselabhängige Hash-Werte benutzen, um Nachrichten in Klartext zu authentifizieren: Mit der Authentifikation weist der Urheber einer Nachricht seine Identität nach. Professor B. möchte die Klausurergebnisse seiner Studenten an das Prüfungsamt übermitteln. Dies kann im Klartext geschehen, weil die Ergebnisse mit Matrikelnummern im Internet bekanntgegeben worden sind. Das Prüfungsamt muß sich aber vergewissern können, daß die Liste wirklich von Professor B. stammt. Dieser bildet dazu einen Hash-Wert der Liste, verschlüsselt ihn mittels eines vereinbarten Schlüssels, zum Beispiel mit DES, und hängt den verschlüsselten Wert an die Liste an. Derartig verschlüsselte Hash-Werte heißen *Message Authentication Codes* (MAC). Das Prüfungsamt kann den empfangenen Hash-Wert entschlüsseln und mit dem Hash-Wert der empfangenen Liste vergleichen. Da es nicht möglich ist, aus dem Klartext, seinem Hash-Wert und dem verschlüsselten Hash-Wert den Schlüssel zu rekonstruieren, ist die Erzeugung einer gefälschten Nachricht nicht möglich.

In PGP wird mittels der Hash-Funktion MD5 (ähnlich SHA, siehe [**Scn**]) vom öffentlichen Schlüssel des Teilnehmers ein „fingerprint“ hergestellt. Mit seiner Hilfe läßt sich die Integrität des Schlüssels überprüfen.

Übungen

18.1. Wir definieren eine Hash-Funktion $h : \{0,1\}^* \to \{0,1\}^*$ mittels

$$h(x) = \left\lfloor 10000 \cdot f\left(\frac{y(1+\sqrt{5})}{2}\right)\right\rfloor, \quad \text{wobei} \quad f(x) = x - \lfloor x \rfloor$$

ist und ferner jedes $x \in \{0,1\}^*$ mit der von ihm dual dargestellten Zahl identifiziert wird.

(a) Welche (maximale) Länge hat $h(x)$?

(b) Gib eine Kollision von h an.

(c) Weshalb taucht in der definition von h eine irrationale Zahl auf?

ABSCHNITT 19

Signaturen

Signaturen haben in der elektronischen Kommunikation die Funktion von Unterschriften. Sie werden benötigt zum Beispiel bei Zahlungsaufträgen an Banken oder beim Abschluß von Verträgen. Eine Unterschrift muß vor allem fälschungssicher sein. Sie muß ja den Empfänger davon überzeugen, daß der Absender das ausgetauschte Dokument auch wirklich, wissentlich und willentlich unterschrieben hat. Sie muß den Empfänger aber auch in die Lage versetzen, im Streitfall nachzuweisen, daß der Absender das Dokument wirklich unterzeichnet hat. Hat der Kunde A seiner Bank einen Auftrag über den Kauf von Aktien erteilt, so muß die Bank verifizieren, daß der Auftrag wirklich von A stammt: Sitzt sie einer Fälschung auf, dann wird sie gegenüber A schadenersatzpflichtig. Wenn der Kurs der Aktie fällt, könnte A allerdings einfallen zu behaupten, er habe den Auftrag nie erteilt. Dann muß die Bank das Gegenteil beweisen können.

Um Fälschungen zu verhindern, muß man die Unterschrift fest mit dem Dokument verbinden und das Dokument gegen nachträgliche Veränderung schützen. Im täglichen Leben weist man die Identität durch Vorlegen eines Ausweises nach. Der Ausweis ist von einer staatlichen Stelle, etwa der Stadtverwaltung, ausgestellt und das Vertrauen auf die Gültigkeit des Dokuments ist letzten Endes bei ihr verankert. In der elektronischen Kommunikation übernehmen die bereits in Abschnitt 16 erwähnten „Trustcenter“ die Rolle der Stadtverwaltung.

Signaturen haben zunächst nichts mit Geheimhaltung zu tun. Sie sind auch dann notwendig, wenn ein Vertrag im Klartext ausgetauscht wird. Im Datenverkehr mit der Bank ist Vertraulichkeit zwar sehr wichtig, viel entscheidender ist aber die Verläßlichkeit der Signatur.

Signaturen müssen 3 Anforderungen elektronischer Kommunikation gewährleisten: die *Authentizität* der Nachricht, also die Identität des Absenders, die *Integrität* der Nachricht und schließlich ihre *Verbindlichkeit*, die es dem Absender nicht erlaubt, seine Urheberschaft zu verleugnen.

RSA-Signatur. Das RSA-System kann nicht nur zum Chiffrieren, sondern auch zum Erzeugen digitaler Signaturen eingesetzt werden. (Zum Beispiel verwendet PGP die RSA-Signatur.) Alice will Bob eine von ihr signierte Nachricht n schicken. Dazu bildet sie die *Signatur*

$$s \equiv n^{d_A}(m_A)$$

mit ihrem öffentlichen RSA-Modul m_A und ihrem geheimen Entschlüsselungsexponenten d_A. Sie sendet das Paar

$$(n,s)$$

an Bob. Dieser bildet mit dem ihm öffentlichen Verschlüsselungsexponenten e_A den Wert

$$t \equiv s^{e_A}(m_A)$$

und prüft, ob $t = n$ ist. Wenn ja, ist die Signatur positiv verifiziert. Einerseits ist Bob sicher, daß die Nachricht von Alice stammt, andererseits kann er das Paar (n,s) vorzeigen, falls Alice später ihre Urheberschaft bestreitet. Da der Klartext n sich aus der Signatur bestimmen läßt, müssen bei diesem Schema sowohl n als auch s chiffriert werden, wenn Geheimhaltung notwendig ist. Es ist grundsätzlich sicherer, für Signatur und Chiffrierung verschiedene Schlüssel zu verwenden.

wie die RSA-Chiffre ist die RSA-Signatur durch Faktorisierung des RSA-Moduls gefährdet. Bei der Signatur gibt es noch eine wesentliche andere Gefahr, nämlich die *existentielle Fälschung*. Da e_A öffentlich bekannt ist, kann man ja zuerst eine Signatur s wählen, dann die gefälschte Nachricht

$$f \equiv s^{e_A}(m_A)$$

bilden und das Paar (f,s) Alice unterschieben – vorausgesetzt auf diese Weise entsteht ein sinnvoller Klartext f. Es ist Alice unmöglich nachzuweisen, daß sie ihn nicht signiert hat. Gegen diese Gefahr gibt es jedoch wirksame Vorbeugung.

Man kann (zum Beispiel mit Hilfsmitteln der Codierungstheorie) nur Nachrichten zulassen, die in einem wohldefinierten Sinn redundant sind. Eine andere Methode besteht darin, eine Hash-Funktion h einzusetzen und nicht die Nachricht selbst, sondern den Hash-Wert zu signieren. Damit ist auch gleich geklärt, wie Nachrichten größerer Länge signiert werden: man signiert nur den Hash-Wert und nicht jeden einzelnen Block. Alice sendet dann

$$n_1,\ldots,n_k \quad \text{und} \quad s \equiv h(n_1,\ldots,n_k)^{d_A} \mod m_A$$

an Bob. Zur Verifikation bestimmt Bob den Hash-Wert $(h(n_1,\ldots,n_k)$ und prüfe die Kongruenz

$$s^{e_A} \equiv h(n_1,\ldots,n_k) \mod m_A.$$

Zur Anfertigung einer existentiellen Fälschung müßte man zu einem gegebenen Hash-Wert eine passende (und sinnvolle) Nachricht erzeugen, und gerade das lassen Hash-Funktionen nicht zu.

Die ElGamal-Signatur. Das Prinzip der RSA-Signatur läßt sich auf alle Public-Key-Systeme übertragen, bei denen Ver- und Entschlüsselung zueinander inverse Abbildungen sind. Im allgemeinen, und speziell beim ElGamal-Kryptosystem gilt nur

$$D \circ E = \mathrm{id}_{\mathscr{P}}\,.$$

Mit einer Variante des Kryptosystems erhält man aber auch eine ElGamal-Signatur. Sie hat wie das Kryptosystem den Vorteil, auf andere Gruppen als $\mathbb{Z}_p^*$ übertragbar zu sein.

Der öffentliche Schlüssel von Alice ist (wie beim Kryptosystem) ein Tripel

$$(p,g,A),$$

wobei p eine Primzahl, g eine Primitivwurzel modulo p und A mittels Alices geheimen Schlüssels a durch

$$A \equiv g^a \ (p)$$

gebildet wird. Alice bestimmt den Hash-Wert h der Nachricht, wählt zufällig ein zu $p-1$ teilerfremdes k, $0 < k < p-1$, und berechnet

$$r \equiv g^k \ (p), \qquad s \equiv k^{-1}(h-ar) \ (p-1).$$

Dabei ist k^{-1} das Inverse zu k modulo $p-1$. Ferner muß man nicht nur die Restklasse von g^k fixieren, sondern den kleinsten positiven Repräsentanten, denn s hängt vom Repräsentanten von g ab und nicht nur von dessen Restklasse. Die Signatur ist (r,s).

Zum Verifizieren prüft Bob zunächst die Ungleichung

$$1 \le r \le p-1$$

und dann die Gültigkeit der Kongruenz

$$A^r r^s \equiv g^h \ (p).$$

Diese ist ja äquivalent zur Gültigkeit von

$$g^{ar} g^{ks} \equiv g^h \ (p),$$

was auf

$$ar + ks \equiv h \ (p-1)$$

hinausläuft. Nun ist s aber gerade so gewählt, daß diese Kongruenz modulo $p-1$ erfüllt ist. Es ist unbedingt notwendig, außer der Kongruenz auch die Ungleichung für r zu prüfen, wie wir einer Übungsaufgabe sehen werden.

Die Sicherheit der ElGamal-Signatur beruht wie die des Kryptosystems auf der Schwierigkeit der Berechnung von diskreten Logarithmen. Falls jemand diskrete Logarithmen in $\mathbb{Z}_p^*$ (bezüglich g bestimmen kann, so findet er zunächst k als diskreten Logarithmus von r und dann a aus der s bestimmenden Gleichung.

Digital Signature Standard. Der DSS ist eine Variante des ElGamal-Kryptosystems. Bei ihm wird zunächst der 160 Bit lange Hash-Wert h mittels der SHA (siehe Abschnitt 18) gebildet. Ein solcher (oder noch kürzerer) Hash-Wert nutzt die Länge des RSA-Moduls oder der ElGamal-Primzahl gar nicht aus, während der DSS von vornherein an diese Länge angepaßt ist.

Alice wählt dazu eine Primzahl q mit

$$2^{159} < q < 2^{160}.$$

Die Dualdarstellung von q hat also die Länge 160. Dann bestimmt man eine Primzahl p, die folgenden Bedingungen genügt:

$$q \mid p \quad \text{und} \quad 2^{511+64t} < p < 2^{512+64t}, \quad 0 \le t \le 8.$$

Die Dualdarstellung von p hat also eine durch 64 teilbare Länge, die zwischen 512 und 1024 liegt. (In der Änderung des Standards von Oktober 2001 wird allerdings die binäre Länge von p auf 1024 Bit fixiert.) Sodann bestimmt man eine Primitivwurzel w modulo p und setzt

$$g \equiv w^{\,(p-1)/q} \ (p).$$

Offensichtlich erzeugt eine Untergruppe der Ordnung q von $\mathbb{Z}_p^*$. Zuletzt wird der geheime Schlüssel $a \in \{1,\dots,q-1\}$ gewählt und

$$A \equiv g^a \ (p)$$

gebildet. Der öffentliche Schlüssel ist nun (p,q,g,A).

Zum Signieren wird wie bei ElGamal ein zu q teilerfremdes $k \in \{1,\dots,q-1\}$ gewählt. Sodann setzt man (anders als bei ElGamal)

$$r \equiv (g^k \bmod p) \ (q).$$

Wieder ist Vorsicht geboten: $(g^k \bmod p) \bmod q$ hängt von der gewählten Restklasse von g^k modulo p ab, und es kommt darauf an, den kleinsten positiven Repräsentanten von $g^k \bmod p$ zu fixieren. Schließlich ist

$$s \equiv k^{-1}(h+ar) \ (q).$$

Das Paar (r,s) bildet die Signatur.

Zum Verifizieren prüft Bob zunächst die Ungleichungen

$$1 \le r,s \le q-1.$$

Sodann bestimmt er den kleinsten positiven Repräsentanten von

$$g^{s^{-1}h} A^{rs^{-1}} \equiv g^{s^{-1}(h+ar)} \equiv g^k \ (p).$$

Nun ist nur noch zu prüfen, ob der kleinste positive Repräsentant von $g^k \bmod p$ die Kongruenz

$$r \equiv (g^k \bmod p) \ (q)$$

erfüllt.

Die DSS-Signatur besteht aus zwei Zahlen, die die gleiche binäre Länge wie der Hash-Wert haben. Zwar hängt die Sicherheit von DSS „nur“ vom diskreten Logarithmus in der von g erzeugten Untergruppe U von $\mathbb{Z}_p^*$ ab, aber nach allem was man weiß, ist die Berechnung diskreter Logarithmen in U nicht einfacher als in $\mathbb{Z}_p^*$.

Übungen

19.1. Man muß für jede Nachricht n bei ElGamal-Signatur ein neues k wählen. Weshalb? (Die bei gleicher Wahl von k für Hash-Werte h_1, h_2 entstehende Situation haben wir schon mehrfach diskutiert.)

19.2. Die Hash-Werte h und h' seien verschieden, und (r,s) sei die ElGamal-Signatur zu h. Wir nehmen ferner an, daß $\mathrm{ggT}(h, p-1) = 1$ ist, und setzen

$$u \equiv h'h^{-1} \ (p-1), \qquad s' \equiv su \ (p-1).$$

Nach dem chinesischen Restsatz existiert dann ein r' mit

$$r' \equiv ru \ (p-1), \qquad r' \equiv r \ (p).$$

Zeige, daß (r',s') die Verifikationskongruenz erfüllt, aber $r' > p-1$ gilt.

Diese Konstruktion zeigt, daß bei der Verifikation der ElGamal-Signatur die Ungleichung $1 \le r \le p-1$ keinesfalls ausgelassen werden darf.

19.3. Sei $p \equiv 3 \ (4)$ prim. Wir nehmen an, daß die Primitivwurzel g modulo p ein Teiler von $p-1$ ist, also $p-1 = gq$ mit $q \in \mathbb{N}$ gilt. Sei a der geheime Schlüssel von Alice und $A \equiv g^a \ (p-1)$ wie gewohnt. Wenn es möglich ist, ein z mit

$$g^{qz} \equiv A^q \ (p)$$

zu bestimmen, dann kann man leicht gefälschte ElGamal-Signaturen herstellen: Für jeden Hash-Wert h, für den $h - qz$ gerade ist, ist (q,s) mit $s = (p-3)(h-qz)/2$ eine gültige Signatur.

Hinweis: Bestimme $q^{(p-3)/2} \mod p$.

Dies zeigt, daß man g nicht als Teiler von $p-1$ wählen sollte. Insbesondere ist $g = 2$ eine schlechte Wahl, denn mit $g = 2$ oder ähnlich kleinen Werten von g läßt sich die kritische Kongruenz sehr leicht lösen. Wieso?

ABSCHNITT 20

Zero-Knowledge-Beweise und Oblivious Transfer

Der praktische Einsatz von Kryptosystemen verlangt die Fixierung von *Protokollen*. In ihnen ist präzise geregelt, wie die Kommunikationspartner gegenseitig ihre Identität überprüfen, dann Schlüssel und schließlich Geheimtexte austauschen.

Solche Protokolle, wenn auch häufig unter anderem Namen, regeln viele Aspekte des menschlichen Lebens, speziell der Kommunikation. Es ist in Deutschland üblich, daß bei einem Telefongespräch zuerst der Angerufene seinen Namen nennt, dann der Anrufer. In Italien hingegen meldet sich der Angerufene mit „pronto“ und der Anrufer identifiziert sich als erster. Hier kommen zwei unterschiedliche Protokolle für die telephonische Kommunikation zum Einsatz. Abweichungen von einem gewohnten Protokoll führen oft zu erheblicher Verwirrung, was der Verfasser durch Anrufe bei italienischen Kollegen am eigenen Leibe erfahren hat.

Sehr ausgefeilte Protokolle findet man in komplizierten Verträgen, zum Beispiel in Kaufverträgen für Grundstücke: Der Verkäufer muß sicher sein, den Kaufpreis zu erhalten, bevor er sein Einverständnis zur Eigentumsumschreibung im Grundbuch gibt. Der Käufer hingegen will den Kaufpreis erst zahlen, wenn sicher ist, daß die Eigentumsumschreibung nicht mehr verhindert werden kann. Als weitere Komplikation tritt häufig auf, daß der Käufer den Kaufpreis durch einen Kredit finanziert, für den er das Grundstück belasten muß, das sich noch gar nicht in seinem Eigentum befindet. Es muß also durch ein präzises Protokoll festgelegt werden, wie die einzelnen Schritte der Vertragsparteien aufeinander folgen, und ohne einen vertrauenswürdigen Dritten, den Notar, lassen sich Blockaden nicht vermeiden.

Wir erwähnen dieses Beispiel, weil Verträge (wenn auch nicht Kaufverträge für Grundstücke) mittels elektronischer Kommunikation abgeschlossen werden. Ein Protokoll für *elektronisches Bargeld* steht dem eines Grundstücksverkaufs an Komplexität in nichts nach. Protokolle müssen dafür sorgen, daß keine Partei die andere betrügen kann. Eine ausführliche Diskussion von Protokollen für den kryptographischen Einsatz findet man in [**Scn**]. Wir beschränken uns hier auf zwei Aspekte, die auch von mathematischem Interesse sind.

Zero-Knowledge-Beweise. Sie sollen folgende, zunächst unlösbar erscheinende Aufgabe lösen. Bernd, der *Beweiser*, will Vera, die *Verifiziererin*, davon überzeugen, daß er ein Geheimnis kennt, ohne es ihr zu verraten. Die Mitteilung des Geheimnisses würde dieses nämlich entwerten. Es muß sichergestellt sein, daß nicht nur der unbefugte Lauscher, sondern auch Vera nach Durchführung des Beweises nichts über das Geheimnis erfahren hat.

Ein Zero-Knowledge-Beweis (man sollte das tunlichst nicht als „Null-Ahnung-Beweis“ übersetzen) dient zum Identitätsnachweis: Nur Bernd kennt das Geheimnis, und wenn er Vera davon überzeugen kann, daß er es kennt, wird sie sich seiner Identität sicher sein.

Im folgenden Beispiel, dem (patentierten) *Feige-Fiat-Shamir-Verfahren*, ist Bernds Geheimnis die Kenntnis einer Quadratwurzel modulo $m = pq$, wobei p und q Primzahlen sind. Bernd wählt zunächst ein s, $0 \leq s \leq m-1$, und bildet

$$b \equiv s^2 \pmod{m}.$$

Sein öffentlicher Schlüssel ist das Paar (b, m). Diesen kennt Vera, und Bernd muß sie davon überzeugen, daß er eine Quadratwurzel von b modulo m kennt, ohne daß er ihr diese mitteilt. Daß der Besitz der Quadratwurzel wirklich ein Geheimnis ist, diskutieren wir am Ende dieses Abschnitts.

Bernd und Vera führen nun einen Dialog.

(1) Bernd wählt zufällig ein r, $0 \leq r \leq m-1$, und sendet

$$x \equiv r^2 \pmod{m}$$

an Vera.

(2) Vera wählt zufällig ein $e \in \{0,1\}$ und sendet es Bernd.

(3) Wenn Bernd $e = 0$ empfängt, sendet er die zufällig gewählte Zahl r an Vera und diese überprüft, ob tatsächlich $r^2 \equiv x \pmod{m}$.

(4) Wenn Bernd $e = 1$ empfängt, sendet er das Produkt rs modulo m an Vera und diese überprüft, ob

$$(rs)^2 \equiv bx \pmod{m}.$$

Wie groß sind die Chancen von Bernd, Vera hinters Licht zu führen, wenn er r gar nicht kennt? Bernd weiß nach Schritt (1) nicht, welche Herausforderung (*challenge*) Vera ihm stellt. Jemand, der auf beide Herausforderungen die richtige Antwort (*response*) kennt, muß r kennen, denn aus r und rs läßt sich s berechnen. Ein Betrüger kann aber eine der beiden Herausforderungen richtig beantworten. Er kann in Schritt (1) wirklich $x \equiv r^2 \pmod{m}$ an Vera senden, um die Herausforderung $e = 0$ erfolgreich meistern zu können. Er kann stattdessen auch $x' \equiv r^2 b^{-1} \pmod{m}$ an Vera senden. Dann besteht er die Prüfung $e = 1$, indem er mit r antwortet.

Die Wahrscheinlichkeit für den Erfolg des Betruges ist also bei einmaligem Ablauf des Dialoges $1/2$. Bei n-maligem Ablauf beträgt sie nur

$$\left(\frac{1}{2}\right)^n.$$

Vera muß von vornherein eine Zahl n von Wiederholungen festlegen, die ihr für die Absicherung gegen Betrug ausreichend erscheinen.

Bei einem Zero-Knowledge-Beweis wird verlangt, daß Vera auch bei positiver Verifikation nichts über das Geheimnis erfährt. Dies kann man am einfachsten durch eine Simulation nachweisen: Simon, der das Geheimnis ebenfalls nicht kennt, wählt zufällig eine Folge $E = \{e_1, \ldots, e_n\}$ der Länge n aus $\{0,1\}$, bereitet die Antworten entsprechend vor und läßt sich die Herausforderungen $e_1, \ldots, e_n$ von Vera stellen. Obwohl Simon die Quadratwurzel aus b nicht kennt, besteht er alle Herausforderungen.

Man kann mit Hilfe der Komplexitätstheorie formalisieren, was ein Zero-Knowledge-Beweis ist (siehe dazu [**Go**]). Das obige Beispiel sollte aber für die Erklärung des Prinzips ausreichen.

Das Fiat-Shamir-Verfahren beruht letzten Endes auf der Schwierigkeit, die Primfaktorzerlegung von m zu finden. Man kann dafür auch andere „schwierige“ Probleme wählen, wie etwa die Färbung gewisser Graphen, den Nachweis der Isomorphie von Graphen, die Bestimmung von Hamiltonschen Wegen oder auch die Berechnung diskreter Logarithmen (siehe dazu [**Ko1**] und [**Scn**]).

Oblivious Transfer. Das Feige-Fiat-Shamir-Verfahren verlangt die Interaktion von Bernd und Vera. Mit Hilfe von *oblivious transfer*, deutsch etwa *blinde Übertragung*, kann man das Verfahren von der Interaktion befreien. Die blinde Übertragung verlangt folgendes:

(1) Bernd schickt Vera zwei verschlüsselte Nachrichten.
(2) Vera kann genau eine davon entschlüsseln und lesen.
(3) Bernd weiß aber nicht, welche der beiden.
(4) Bernd und Vera sind beide sicher, daß die Eigenschaften (2) und (3) erfüllt sind.

Wir diskutieren eine auf diskreten Logarithmen beruhende Realisierung der blinden Übertragung. Dazu werden zunächst eine Primzahl p und eine Primitivwurzel g modulo p fixiert.

Anschließend wird ein C mit $1 \leq C \leq p-1$ zufällig gewählt, dessen diskreten Logarithmus bezüglich g modulo p *niemand* berechnen kann.

Vera bereitet nun ein Paar (b_1, b_2) von „blinden“ Schlüsseln vor. Sie wählt dazu zufällig ein $i \in \{1,2\}$ und ein x, $1 \leq x \leq p-1$. Sie setzt $j = 3-i$ und

$$b_i \equiv g^x, \qquad b_j \equiv Cg^{-x} \ (p).$$

Das blinde Schlüsselpaar (b_1, b_2) wird veröffentlicht. Man beachte, daß Vera den diskreten Logarithmus x' von b_j nicht kennt – sie könnte sonst auch den von C bestimmen, entgegen unserer generellen Voraussetzung über C.

Bernd hat zwei Nachrichten m_1, m_2, die an Vera zu senden sind. Er wählt zufällig y_1, y_2 mit $1 \le y_1, y_2 \le p-1$ und sendet gemäß dem ElGamal-Kryptosystem

$$g^{y_1}, g^{y_2}, m_1 b_1^{y_1}, m_2 b_2^{y_2} \ \ (p).$$

Vera kennt x und g^{y_i}. Sie kann also

$$b_i^{y_i} \equiv (g^{y_i})^x \ \ (p)$$

bestimmen und somit m_i aus $m_i b_i^{y_i}$ entschlüsseln.

Wenn sie auch m_j gewinnen will, muß sie

$$b_j^{y_j} \equiv (b^{x'})^{y_j} \ \ (m)$$

bestimmen. Sie kennt zwar $C = b^{x'}$ und b^{y_j}, aber niemand weiß, wie man daraus ohne Kenntnis von x' oder y_j die Potenz $b^{x'y_j}$ ausrechnen könnte. (Dies ist das uns schon bekannte Diffie-Hellman-Problem.)

Man beachte, das es in Veras Interesse ist, eine von beiden Antworten lesen zu können. Ebenso ist es in ihrem Interesse, Bernd nicht wissen zu lassen, welche der beiden das ist. Bernd hingegen kann nicht erkennen, welche der Antworten entschlüsselbar ist. Daß es nur eine von beiden ist, kann er prüfen, indem er

$$b_1 b_2 \equiv C \ \ (p)$$

verifiziert. Um zu betrügen, müßte Vera ihm ein Paar (b_1, b_2) schicken, das diese Bedingung nicht erfüllt (oder den diskreten Logarithmus von C kennen).

Wir beschreiben nun, wie man mit Hilfe der blinden Übertragung eine nicht interaktive Version des Fiat-Shamir-Protokolls realisieren kann. Zur Vorbereitung muß Vera eine hinreichend lange Folge von Schlüsseln (b_{1k}, b_{2k}), $k = 1, \ldots, n$, für die blinde Übertragung bereitstellen.

Statt auf Veras Herausforderungen zu warten, bildet Bernd nun eine zufällige Folge (r_k) und sendet

$$r_k^2, r_k, r_k s \mod m, \qquad k = 1, \ldots, n,$$

an Vera, wobei r_k mit dem Schlüssel b_{1k} und $r_k s$ mit dem Schlüssel b_{2k} blind verschlüsselt wird. Vera kann immer nur r_k oder $r_k s$ ermitteln und wie bei der interaktiven Version die entsprechende Kongruenz verifizieren.

Quadratwurzeln modulo m. Wir wollen nun zeigen, daß die Fähigkeit, Qudratwurzeln modulo m zu ziehen, äquivalent dazu ist, m zerlegen zu können. Die Kenntnis einer Quadratwurzel modulo m ist es also ein ebenso sicheres Geheimnis wie die Kenntnis der Zerlegung.

Da $m = pq$ Produkt zweier verschiedener Primzahlen p und q ist, besitzt jedes b, das ein Quadrat modulo m ist, *vier* Quadratwurzeln modulo m. Dann ist b ja auch ein Quadrat modulo p und q, und sind y, z Quadratwurzeln von b modulo p beziehungsweise modulo q, dann liefert jedes der vier Systeme

$$x \equiv \pm y \ \ (p)$$
$$x \equiv \pm z \ \ (q)$$

simultaner Kongruenzen eine Quadratwurzel von b modulo m.

Besitzt jemand einen Algorithmus, der alle vier Quadratwurzeln bestimmen kann, so findet er sofort einen Faktor von m (und damit die Zerlegung). Falls $x^2 \equiv y^2 \ (m)$ gilt ja

$$(x-y)(x+y) \equiv 0 \ \ (m).$$

Wir können x, y so wählen, daß $x \not\equiv \pm y \ \ (m)$. Da $x+y \not\equiv 0 \ \ (m)$, sind $x-y$ und m nicht teilerfremd, aber m scheidet wegen $x-y \not\equiv 0 \ \ (m)$ als größter gemeinsamer Teiler aus. Mithin ist $\mathrm{ggT}(x-y, m)$ ein nichttrivialer Teiler von m.

Besitzt jemand einen Algorithmus, der wenigstens eine Quadratwurzel y liefert, so sind seine Chancen auf Faktorisierung bei einmaliger Anwendung schon $1/2$: Mit Wahrscheinlichkeit $1/2$ erfüllt die gefundene Quadratwurzel y von x^2 modulo m (bei zufällig gewähltem x) die Bedingung $y \not\equiv \pm x \ \ (m)$. Bei n-maliger Wiederholung ist die Wahrscheinlichkeit der Faktorisierung $1-(1/2)^n$.

Umgekehrt ist es nicht schwierig, Quadratwurzeln modulo m zu bestimmen, Wenn man die Zerlegung von $m = pq$ kennt. Wegen des chinesischen Restsatzes braucht man nur Quadratwurzeln modulo p und modulo q. Es genügt also, Quadratwurzeln modulo einer Primzahl p zu berechnen.

Man prüft zunächst, ob

$$a^{(p-1)/2} \equiv 1 \ \ (p)$$

ist. Falls nicht, ist a kein Quadrat modulo p. Andernfalls ist a Quadrat modulo p.

Sei x die noch unbekannte Quadratwurzel von a, $x^2 \equiv a \ \ (p)$. Dann ist

$$a^{(p-1)/2} \equiv x^{p-1} \equiv 1 \ \ (p)$$

und somit

$$a^{(p+1)/2} \equiv a \ \ (p).$$

Im Fall $p \equiv 3(4)$ ist $(p+1)/2$ gerade, und

$$x \equiv a^{(p+1)4} \ \ (p)$$

ist eine Quadratwurzel.

Im Fall $p \equiv 1(4)$ sucht man nun am besten ein c mit

$$(c^2-a)^{)(p-1)/2} \equiv -1 \ \ (p)$$

und setzt $D = c^2 - a$. Da D in $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ keine Quadratwurzel besitzt, ist die Erweiterung

$$\mathbb{F}_p[\sqrt{D}] = \mathbb{F}_p[x]/(X^2 - D)$$

ein Körper (mit p^2 Elementen). Wir behaupten, daß

$$(\sqrt{D})^p = -\sqrt{D}$$

ist. Es gilt ja

$$(\sqrt{D})^p = \sqrt{D}(\sqrt{D})^{p-1} = \sqrt{D}D^{\frac{p-1}{2}} = -\sqrt{D},$$

denn $D^{(p-1)/2} = -1$. Es folgt

$$\begin{aligned}(c+\sqrt{D})^{p+1} &= (c+\sqrt{D})(c+\sqrt{D})^p = (c+\sqrt{D})(c+(\sqrt{D})^p)\\ &= (c+\sqrt{D})(c-\sqrt{D}) = c^2 - D = a.\end{aligned}$$

Man kann zeigen, daß $c^2 - a$ für $(p-1)/2$ Restklassen modulo p kein Quadrat ist und daher durch Ausprobieren schnell ein geeignetes D finden (siehe [**Fo**], 16.6). Eine andere Methode zum Ziehen von Quadratwurzeln (im Fall $p \equiv 1(4)$) wird in [**Ko1**] beschrieben.

Übungen

20.1. Entwickle einen Zero-Knowledge-Beweis für die Kenntnis eines diskreten Logarithmus.

20.2. Man kann das Feige-Fiat-Shamir-Protokoll durch „Parallelisierung“ beschleunigen. Bernds Geheimnis besteht dann aus einer Folge $(s_1, \ldots, s_k)$ von Quadratwurzeln modulo m. Er schickt in Schritt (1) eine Folge $(r_1^2, \ldots, r_k^2)$ an Vera, empfängt die Herausforderung $(e_1, \ldots, e_k) \in \{0,1\}^k$ und antwortet mit $\prod_{i=1}^k r_i s_i^{e_i}$.

Diskutiere dies in allen Einzelheiten und bestimme die Chancen eines Betrügers.

20.3. Vera wählt x, bildet $y \equiv x^2 \mod m$, sendet dies an Bernd und verlangt von ihm, er solle seine Kenntnis der Zerlegung von m dadurch nachweisen, daß er ihr eine Quadratwurzel von $y \mod m$ zurückschickt. Darf Bernd sich darauf einlassen?

Literaturverzeichnis

[AD] Aczél, J., Daróczy, Z.: On measures of information and their chracterizations. Academic Press 1975.

[Ba] Bauer, F.: Entzifferte Geheimnisse. Springer-Verlag 1995.

[BP] Beker, H., Piper, F.: Cipher systems. Northwood Books, 1982.

[Be] Beutelspacher, A., Schwenk, J., Wolfenstetter, K.-D.: Moderne Verfahren der Kryptographie. Vieweg 1998.

[Bu] Buchmann, J.: Einführung in die Kryptographie. Springer 1999.

[Co] Cohen, H.: A course in computational algebraic number theory. Springer 1993.

[DK] Deavours, C. A., Kruh, L.: Machine Cryptography and modern cryptanalysis. Artech House 1985.

[Fo] Forster, O.: Algorithmische Zahlentheorie, Vieweg 1996.

[Go] Goldreich, O.: Modern cryptography, probabilistic proofs and pseudorandomness. Springer 1999.

[HQ] Heise, W., Quattrocchi, P.: Informations- und Codierungstheorie, 3. Auflage. Springer 1995.

[Ka1] Kahn, D.: The Codebreakers: The Story of Secret Writing, 2. Auflage. Scribner 1996.

[Ka2] Kahn, D.: Kahn on Codes. Macmillan 1983.

[Ka3] Kahn, D.: Seizing the Enigma. Houghton Mifflin 1991.

[KW] Kameda, T., Weihrauch, K.: Einführung in die Codierungstheorie. Bibliographisches Inst. 1973.

[Ko1] Koblitz, N.: A course in number theory and cryptography, 2. Auflage. Springer 1994.

[Ko2] Koblitz, N.: Algebraic aspects of cryptography. Springer 1998.

[McE] McEliece, R. J.: The theory of information and coding. Addison-Wesley 1977.

[Mv] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography. CRC Press 1997.

[Re] Rényi, A.: Tagebuch über die Informationstheorie. Birkhäuser 1982.

[Ro] Roman, St.: Codierung and Information Theory. Springer 1992.

[Sa] Salomaa, A.: Public-Key Cryptography, 2. Auflage. Springer-Verlag 1996.

[Scn] Schneier, B.: Angewandte Kryptographie, 2. Auflage. Addison-Wesley 2000.

[Sch] Schulz, R.-H.: Codierungstheorie. Vieweg 1991.

[Si] Singh, S.: Geheime Botschaften. Hanser 2000.

[Sti] Stinson, D.: Cryptography: Theory and Practice. CRC Press 1995.

[Stn] Stichtenoth, H.: Algebraic function fields and codes. Springer 1993.

[vL] van Lint, J. H.: Introduction to Coding Theory, Third Edition. Springer 1999.

[We] Welsh, D.: Codes und Kryptographie. VCH 1991.
[Wi] Willems, W.: Codierungstheorie. De Gruyter 1999.

Einige Internet-Ressourcen zur Kryptographie:

[Cry] `http://www.cryptool.de/`. Von dieser Adresse kann man das Cryptool beziehen, in dem sehr viele klassische und moderne Chiffren implementiert sind.
[NIST] `http://csrc.nist.gov/encryption/`. Über diese Adresse hat man zu den Dokumenten, die die vom NIST (früher NBS) standardisierten Verfahern beschreiben.
[Eni] `http://mad.home.cern.ch/frode/crypto/index.html`. Hier findet man viele Dokumente und Software zur (historischen) Kryptographie, unter anderem eine sehr schön gestaltete Simulation der Marine-Enigma mit 3 Rotoren und Originalarbeiten Alan Turings zur Enigma.