# Kryptologie

Ernst Günter Giessmann
Dirk Verworner
u. a.

Berlin, den 18. Januar 1994

**Zusammenfassung**

Diese Ausarbeitung ist auf der Grundlage einer von mir im Wintersemester 1992/1993 gehaltenen vierstündigen Vorlesung durch die Studenten P. Heitsch, D. Verworner, R. Magnus, und P. Wildenhain und eifrige Lektoren angefertigt worden. Sie haben eigenes hinzugefügt und aus der Literatur ergänzt. Anderes fehlt in dem Skript oder hätte ausführlicher dargestellt sein sollen. So fehlt der ganze Abschnitt über Quantenkryptographie und das Kapitel 7, das Problemen des Datenschutzes und der Datensicherheit gewidmet war. Die Autoren sehen deshalb diesen Preprint als einen ersten Entwurf an, sie nehmen Anregungen der Leser gern entgegen und versprechen, den zweiten Entwurf besser zu machen.

E. G. Giessmann

**Vorbemerkung zur 2. Auflage**

Aufgrund der Fortführung der Vorlesung Kryptologie als Seminar wurde dieses Skript nochmals überarbeitet und einzelne Vorträge als Anhang aufgenommen.

Die einzelnen Anhänge stehen dabei in keinem Zusammenhang.

So kann der Anhang A von Ulrich Kunitz als eine Erweiterung der in Kapitel 3 vorgestellten Verfahren der Kryptoanalyse betrachtet werden.

Die Anhänge B von Robert Magnus und C von mir stellen weitere Anwendungen kryptographischer Verfahren im Bezug auf Public-Key-Systeme und Authentisierung vor, und bilden somit eine logische Fortsetzung des Kapitels 6.

Der Anhang D von Michael Weber stellt einen Überblick über rechtliche Probleme des Datenschutzes und der Datensicherheit dar.

Die Datenkompression, ein der Kryptologie verwandtes Thema, wird in Anhang E von Uwe Erpel beschrieben. Der Anhang enstand im Rahmen einer Studienarbeit und gibt einen einführenden Überblick über einige Verfahren.

D. Verworner

# Inhaltsverzeichnis


**1 Einführung und das Anliegen der Kryptologie** **1**

**2 Einfache kryptographische Systeme** **3**

2.1 Cäsar-Chiffren . . . 3

2.2 Lineare Verschlüsselungen . . . 5

2.3 Lineare Verschlüsselungen größerer Blocklänge . . . 6

2.4 Nichtlineare Verschlüsselungen . . . 6

**3 Blockchiffren** **8**

3.1 Die Vigenère-Chiffre . . . 8

3.2 Die Hill-Chiffre . . . 10

3.3 Die Playfair-Chiffre . . . 11

3.4 Die Dellastelle-Chiffre . . . 12

3.5 Transpositionschiffren . . . 12

3.6 Rotormaschinen – die ENIGMA . . . 13

3.7 Der Data Encryption Standard . . . 14

**4 Flußchiffren** **20**

4.1 Blockchiffren . . . 20

4.2 Flüsse aus Zufallsfolgen . . . 21

4.3 Pseudozufallszahlen – Lineare Schieberegister . . . 22

4.4 Pseudozufallszahlen – Nichtlineare Verfahren . . . 24

**5 Systeme mit öffentlichen Schlüsseln** **25**

5.1 Grundlagen . . . 25

5.2 Der RSA-Algorithmus . . . 26

5.3 Diskreter Logarithmus . . . 28

5.3.1 Der Diffie-Hellman-Schlüsselaustausch . . . 29

5.3.2 Das Massey-Omura-Kryptosystem . . . 29

5.3.3 Das ElGamal-Kryptosystem . . . 30

5.4 Das Knapsack-Problem . . . . . 30
5.5 Elliptische Kurven . . . . . 31

**6 Anwendungen und benachbarte Fragestellungen** **34**
6.1 Digitale Unterschriften . . . . . 35
6.2 Authentisierung von Personen – *das Kerberos-System* . . . . . 36
6.3 Kartenspiele über Telefon . . . . . 38
6.4 Münzwurf . . . . . 39
6.5 Bitdepot . . . . . 39
6.6 Null-Wissen-Beweise . . . . . 40

**A Ausgewählte Verfahren der Kryptoanalyse** **44**
A.1 Kappa und Chi . . . . . 44
A.2 Periodenanalyse . . . . . 51
A.3 Zurechtrücken begleitender Alphabete . . . . . 53

**B Authentisierung durch Schiedsrichter** **56**
B.1 Das Verfahren . . . . . 56
B.2 Betrügen . . . . . 57
B.3 Mehrfache Schiedsrichter . . . . . 58

**C Geheime Wahl per Telefon** **60**
C.1 Vorbemerkungen . . . . . 60
C.2 Ballot Issuing Protocol . . . . . 61
C.3 Die Wahl . . . . . 62
C.4 Nachbetrachtungen . . . . . 63

**D Datenschutz** **64**
D.1 Geschichte . . . . . 64
D.2 Gegenstand des Datenschutzes . . . . . 65
D.3 Bundesverfassungsgericht (BVfG) zur Volkszählung . . . . . 66
D.4 Grundsätze des Datenschutzes . . . . . 67
D.5 Datensicherung . . . . . 68

**E Datenkomprimierung** **69**
E.1 Kodierung einer endlichen Menge von gleich häufigen Symbolen . . . . . 70
E.2 Häufigkeitsabhängige Kodierung . . . . . 71
E.3 Kontextabhängige Kodierung . . . . . 75
E.4 LZW - Datenkomprimierung . . . . . 76

# Kapitel 1

# Einführung und das Anliegen der Kryptologie

*„Die besten Traktate über Kryptographie sind die Werke ungläubiger Gelehrter."*

(Umberto Eco, Der Name der Rose)

*„…denn merk dir: Es gibt keine Geheimschrift, die sich nicht mit ein wenig Geduld entziffern ließe."*

(desgl.)

**Kryptologie** ist der Teil der Mathematik, der einerseits Verfahren ersinnt, wie man Daten vor Unberufenen verbergen kann und der andererseits auch versucht, diese Daten ohne Erlaubnis zu lesen. Der Entwurf dieser Verfahren wird als **Kryptographie**, der unberufene Eingriff als **Kryptoanalyse** bezeichnet.

Für einen erfolgreichen Datenschutz und eine dauerhafte Sicherung der Daten muß man die Stärken und Schwächen der benutzten Verfahren kennen, denn ein absolut sicheres System wird man nicht erstellen können. Immer wird es Schwachpunkte geben, die man nicht verhindern kann. Der Hauptschwachpunkt ist dabei der Mensch, der mit diesen Daten umgeht. Da die Nachrichten zu einem gewissen Zeitpunkt im Klartext vorhanden sind, beispielsweise bei der Eingabe, können sie schon gelesen werden. Zur Kontrolle fertigt man sich oft noch einen Ausdruck des Klartextes an, der nicht ordnungsgemäß vernichtet wird und dann irgendwo liegen bleibt. Solche Unachtsamkeiten beim Umgang mit sensiblen Daten können durch die beste Verschlüsselung nicht verhindert werden.

Die Kryptographie muß Verfahren entwerfen, die leicht zu implementieren sind, ein schnelles Ver- und Entschlüsseln gewährleisten und das unberufene Lesen, Modifizieren und Simulieren der Daten verhindern. Das sind einander widersprechende Forderungen, und man muß in jeder Situation nach einem Kompromiß zwischen den Sicherheitsanforderungen und den vorhandenen Möglichkeiten suchen. Die dann entworfenen Verfahren sollte man durch weitere Sicherungen, wie Zugangskontrollen, Ausweiskontrollen, Schreibschutz oder andere physische Barrieren ergänzen. Nicht immer ist die Verhinderung des Lesens der Daten die alleinige Forderung. Manche Daten müssen sicher gegenüber Verfälschungen, andere gegenüber Vernichtung sein. Bei einer Kreditkarte kann die Kontonummer öffentlich, jedoch nicht veränderbar sein. Der zur Verfügung stehende Kreditbetrag sollte nicht lesbar, eine Überziehung aber trotzdem unmöglich sein.

Die Vorlesung soll nicht der Kryptoanalyse gewidmet sein. Vielleicht ist es an dieser Stelle

zweckmäßig darauf hinzuweisen, daß die aktive Kryptoanalyse sowohl gegen staatliche Stellen, als auch gegenüber Privatpersonen in Deutschland verboten ist. Trotzdem werden wir uns bei jedem Verfahren überlegen, mit welchen Mitteln sich ein unberufener Entschlüsseler die Klartexte verschaffen kann. Wir werden lernen, daß die Geduld der unberufenen Entzifferer oft auf eine arge Probe gestellt werden wird.

**Definition 1** *Im folgenden werden wir unter einem kryptographischen System eine Menge $M$ von Klartexten, eine Menge $C$ von Chiffren, eine Menge $K$ von Schlüsseln und Chiffrier- und Dechiffrierfunktionen $E : M \times K \to C$ und $D : C \times K \to M$ verstehen, so daß*

$$D(E(m, k'), k'') = m$$

*für alle Nachrichten $m \in M$ ist.*

Die Schlüssel $k'$ und $k''$ heißen auch Schlüssel und Gegenschlüssel. Sind die Schlüssel $k'$ und $k''$ identisch oder leicht voneinander abzuleiten, so spricht man von einem **symmetrischen Kryptosystem**. Ist im Gegenteil aus der Kenntnis des Schlüssels $k'$ der Gegenschlüssel $k''$ nicht oder nur schwer zu erhalten bzw. eine chiffrierte Nachricht zu entschlüsseln, so spricht man von einem **asymmetrischen Kryptosystem**. Was es heißt, daß es *schwer* ist, den Gegenschlüssel zu bestimmen, werden wir genauer definieren, wenn wir die Systeme mit öffentlichen Schlüsseln behandeln.

Über die Funktionen $E$ und $D$ setzt man normalerweise voraus, daß sie eindeutig sind. Das kann man auch immer erreichen, wenn man die Mehrdeutigkeiten in der Schlüsselmenge verwaltet. Der Einfachheit halber wählt man auch für die Menge der Nachrichten die Menge der nichtleeren Wörter über einem gewissen Alphabet, obwohl man oft sinnlose Texte nicht zu verschlüsseln braucht.

Bei der Beurteilung der Sicherheit eines Verschlüsselungsverfahrens sollte man immer davon ausgehen, daß der unberufene Entschlüsseler das Verfahren des Ver- und Entschlüsselns, das wir verwenden, in allen Einzelheiten kennt. Wir wollen ihn auch nicht unterschätzen und trauen ihm die Kenntnis der besten Algorithmen und der schnellsten Rechentechnik zu. Wir betrachten ihn aber nicht als allmächtig, eine exponentielle Laufzeithürde kann auch er/sie nicht überspringen.

Wir unterscheiden beim Eindringen des unberufenen Entschlüsselers drei Stufen:

Erstens, der **Angriff mit der Chiffre** (*ciphertext only attack*) allein. Hier gehen wir davon aus, daß der oder die unberufene Entschlüsseler/in nur den chiffrierten Text kennt.

Zweitens, der **Angriff mit bekanntem Klartext** (*known plaintext attack*). Jetzt gehen wir davon aus, daß der unberufene Entschlüsseler sowohl Original als auch Chiffre kennt und Klartextstellen und Chiffrestellen zuordnen kann. Das ist durchaus realistisch, denn bei vielen Chiffren kennt man die Originaltexte, oder zumindest eine Menge möglicher Klartexte. So sind die Wörter, die als Paßwörter in Rechenanlagen verwendet werden, oft leicht durch Probieren zu finden.

Drittens schließlich, der stärkste Eingriff in die Datenverschlüsselung, der **Angriff mit gewähltem Klartext** (*chosen plaintext attack*). So kann man eine Bank mit den verschiedensten Überweisungsaufträgen traktieren, oder einem Rechtsanwalt die unterschiedlichsten Verträge zum Verschlüsseln anbieten.

# Kapitel 2

# Einfache kryptographische Systeme und ihre Kryptoanalyse

In diesem Kapitel beschäftigen wir uns mit einfachen kryptographischen Systemen. Dazu wollen wir als erstes ein wenig in der Geschichte blättern.

## 2.1 Cäsar-Chiffren

Bereits JULIUS CAESAR benutzte (angeblich) zum sicheren Datenaustausch ein kryptographisches System, die sogenannten Cäsar-Chiffren. Dabei benutzte er als Klartexte Wörter (bzw. Sätze) über dem 26-buchstabigen Alphabet. Es entstanden als Chiffren wiederum Wörter des 26-buchstabigen Alphabets, wobei jeder Buchstabe des Klartextes innerhalb des Alphabets um drei Buchstaben verschoben ist. Allgemeiner kann man natürlich das Alphabet auch um eine beliebige Anzahl von Buchstaben verschieben, so daß als Schlüssel Buchstaben des Alphabets fungieren, d.h.

$$\begin{array}{rcll} M & := & \mathbb{Z}_{26}^{\star} & \text{entspricht der Menge der Klartexte,} \\ C & := & \mathbb{Z}_{26}^{\star} & \text{entspricht der Menge der Chiffren und} \\ K & := & \mathbb{Z}_{26} & \text{entspricht der Menge der Schlüssel.} \end{array}$$

Die Chiffrierfunktion $E : M \times K \rightarrow C$ bildet demnach den Klartext $m = m_1 m_2 \cdots m_n \in M$ auf die Chiffre $c = c_1 c_2 \cdots c_n \in C$ wie folgt ab :

$$E(m,k) = c \quad \text{mit} \quad c_i = m_i + k \bmod 26.$$

Für die Dechiffrierfunktion gilt dadurch :

$$D(c,k) = m \quad \text{mit} \quad m_i = c_i - k \bmod 26.$$

Eine weitere Variante ist die Chiffrierfunktion $E$ mit $c_i = k - m_i \bmod 26$. Sie zeichnet sich gegenüber der anderen Variante dadurch aus, daß bei ihr die Chiffrier- und Dechiffrierfunktionen, sowie deren Schlüssel übereinstimmen. Es handelt sich hierbei also um ein symmetrisches System.

Zur weiteren Klassifizierung der Cäsar-Chiffren führen wir im folgenden einen neuen Begriff ein.

**Definition 2** *Eine* **Blockchiffre** *ist ein kryptographisches System mit der Menge $M = \mathcal{A}^\star$ der Klartexte, wobei $\mathcal{A}$ ein Alphabet ist, einer endlichen Schlüsselmenge $K$, der Menge $C$ der Chiffren und einer Chiffrierfunktion $E : \mathcal{A}^t \times K \longrightarrow C$.*

Dabei wird eine Nachricht $m \in \mathcal{A}^n$ so verschlüsselt, daß sie erst in $\mathcal{A}^{rt}$ eingebettet wird, wobei $n \leq rt$ und $r$ minimal ist, und dann auf den Blöcken der Größe $t$ $r$-mal die Chiffrierfunktion $E$ angewendet wird.

Nach dieser Definition ist die Cäsar-Chiffre eine Blockchiffre der Länge $t = 1$. Daraus ergibt sich ein entscheidender Nachteil dieser Chiffre. Der besteht darin, daß der Anfang der Nachricht keinen Einfluß auf die Verschlüsselung des Endes der Nachricht hat. Mit **Diffusion** bezeichnen wir das Maß des Einflusses einer Textstelle auf den gesamten chiffrierten Text. Je größer die Diffusion, desto mehr wird beispielsweise die Redundanz eines sprachlichen Textes gleichmäßig über die Chiffre verteilt. Dem unberufenen Entschlüsseler wird damit eines seiner wichtigsten Werkzeuge entzogen. Die Cäsar-Chiffre hat in diesem Sinne eine geringe Diffusion.

Neben der Diffusion ist ein weiteres Maß für ein Kryptosystem die **Konfusion**. Mit ihr beschreibt man den Zusammenhang zwischen dem Schlüssel und dem chiffrierten Text. Je weniger Rückschlüsse sich aus dem chiffrierten Text auf den Schlüssel ziehen lassen, desto größer ist die Konfusion. Auch nach diesem Maß ist die Cäsar-Chiffre nachteilig. Der häufigste Buchstabe des chiffrierten Textes ist in der Regel die Chiffre des Buchstabens E und ihr Abstand zu E ist der Schlüssel. Große Konfusion bedeutet also unter anderem, daß der gleiche Buchstabe in verschiedene Chiffren übersetzt wird.

Durch diese Schwächen ist die Analyse einer Cäsar-Chiffre meist sehr leicht.

Da bei ihr gleiche Buchstaben im Klartext die gleichen Bilder haben, spiegeln sich in der Chiffre die Häufigkeiten einzelner Buchstaben in der natürlichen Sprache wider. Mit einer statistischen Analyse der Buchstabenhäufigkeiten sind demnach leicht die einzelnen Buchstaben zu erhalten.

Die relativen Häufigkeiten der Buchstaben im deutschen und englischen Text sind[1] :

| Buchstabe | Deutsch | Englisch | Buchstabe | Deutsch | Englisch |
|---|---|---|---|---|---|
| A | 5,40% | 8,17% | N | 10,42% | 6,75% |
| B | 1,89% | 1,49% | O | 2,24% | 7,51% |
| C | 3,15% | 2,78% | P | 0,59% | 1,93% |
| D | 5,17% | 4,25% | Q | 0,01% | 0,10% |
| E | 18,10% | 12,70% | R | 8,08% | 5,99% |
| F | 1,60% | 2,23% | S | 6,35% | 6.33% |
| G | 3,15% | 2,02% | T | 5,57% | 9,06% |
| H | 5,14% | 6,09% | U | 4,10% | 2,76% |
| I | 7,52% | 6,97% | V | 0,87% | 0,98% |
| J | 0,19% | 0,15% | W | 1,67% | 2,36% |
| K | 1,13% | 0,77% | X | 0,01% | 0,15% |
| L | 3,45% | 4,03% | Y | 0,02% | 1,97% |
| M | 2,51% | 2,41% | Z | 1,67% | 0,07% |

[1] Nach [FR88]. Ein von Zitaten strotzender zoologischer Text über den Einfluß von Ozon im Zentrum von Zaire wird dabei natürlich eine andere Häufigkeitsverteilung haben als ein Traktat über die amorösen Aventüren des Balthasar Matzbach am Rande des Panamakanals.

Von Vorteil aber ist, daß die Cäsar-Chiffre ein schnelles Ver- und Entschlüsseln ermöglicht und recht fehlerstabil ist.

**Übungsaufgabe 1** *Entschlüsseln Sie folgenden Textausschnitt aus englischer Quelle !*

```
PXPXKXENVDRUXVTNLX
HYMXGMAXYKXJNXGVRF
XMAHWGXXWLEHGZXKVB
IAXKMXQM
```

## 2.2 Lineare Verschlüsselungen

**Definition 3** *Sei* $M := \mathbb{Z}_n^\star$ *die Menge der Klartexte,* $K := \mathbb{Z}_n^2$ *die Menge der Schlüssel und* $C := \mathbb{Z}_n^\star$ *die Menge der Chiffren. Dann heißt* $E_{(a,b)}$ **lineare Verschlüsselung**, *falls*

$$E_{(a,b)} : \mathbb{Z}_n \to \mathbb{Z}_n \quad \textit{und} \quad c = E_{(a,b)}(m) \equiv am + b \bmod n \quad \textit{mit } ggT(a,n) = 1.$$

Da $a$ und $n$ teilerfremd sind, können wir nun die Dechiffrierfunktion $D_{(a,b)}$ berechnen. Es gilt :

$$\begin{aligned} c &\equiv am + b \bmod n \qquad \text{bzw.} \\ am &\equiv c - b \bmod n. \end{aligned}$$

Wir bestimmen das arithmetische Inverse $u \in \mathbb{Z}_n$ von $a$ modulo $n$ mit dem Euklidischen Algorithmus, d.h. es gilt $au \equiv 1 \bmod n$. Das Inverse existiert, da $a$ und $n$ teilerfremd sind. Wir erhalten aus obiger Äquivalenz durch Multiplikation mit $u$ :

$$\begin{aligned} uam &\equiv uc - ub \bmod n \qquad \text{bzw.} \\ m &\equiv uc - ub \bmod n. \end{aligned}$$

Daraus folgt für die Dechiffrierfunktion

$$m = D_{(a,b)}(c) \equiv uc - ub \bmod n \equiv a'c + b' \bmod n \equiv E_{(a',b')}(c),$$

wobei $a' = u$ und $b' = -ub$ sind.

Bei der betrachteten Art der Verschlüsselung ist es für die Analyse nicht notwendig $a$ und $b$ zu kennen, sondern es genügt $a'$ und $b'$ zu bestimmen. Bei längeren Texten sind $a'$ und $b'$ relativ leicht durch eine Häufigkeitsanalyse zu ermitteln. Das System hat also den Nachteil, daß es bei längeren Texten nicht besonders sicher ist. Vorteilhaft ist aber, daß das System einen kurzen Schlüssel $(a, b)$ und recht einfache Chiffrier- und Dechiffrierfunktionen besitzt. Desweiteren ist der Schlüsselraum der linearen Verschlüsselungen größer, als der Schlüsselraum der Cäsar-Chiffren. So erhält man für $n = 26$ nur 26 verschiedene Schlüssel der Cäsar-Chiffre, aber insgesamt $12 \cdot 26 = 312$ mögliche Schlüssel für lineare Verschlüsselungen.

Wir wollen im folgenden Verknüpfungen kryptographischer Systeme untersuchen.

**Definition 4** *Die* **Komposition** *zweier kryptographischer Systeme* $(M, C, K, E, D)$ *und* $(C, C', K', E', D')$ *ist das System* $(M, C', K \times K', E \circ E', D \circ D')$.

**Definition 5** *Unter dem* **Produkt** *zweier kryptographischer Systeme* $(M, C, K, E, D)$ *und* $(M', C', K', E', D')$ *verstehen wir das kryptographische System* $(M \times M', C \times C', K \times K', E \times E', D \times D')$.

In diesem Sinne ist eine Blockchiffre die Ausdehnung des Kryptosystems, das erst auf Nachrichten der Länge $t$ definiert ist auf beliebige Nachrichten durch Produktbildung.

Durch Verwendung unterschiedlicher Kryptosysteme auf den einzelnen Teilen erhöht man die kryptographische Sicherheit, aber leider auch den Aufwand zum Entschlüsseln.

Die Komposition zweier Cäsar-Chiffren hat zwei Schlüssel $(k_1, k_2)$. Der unberufene Entschlüsseler braucht aber nur $-(k_1 + k_2)$ zu bestimmen. Analog erschwert auch die Komposition linearer Verschlüsselungen nicht die unberufene Dechiffrierung. Anders sieht es dagegen aus, wenn man die Komposition zweier linearer Systeme bezüglich unterschiedlicher Moduln untersucht, d.h. zum Beispiel mit $n_1 = 26$ und $n_2 = 29$.

**Übungsaufgabe 2** *Entschlüsseln Sie den linear chiffrierten Text „*`QAOOYQQEVHEQV`*“ ! Häufigkeitsanalysen ergaben für die Buchstaben Y 13,1% und V 9,0%.*

## 2.3 Lineare Verschlüsselungen größerer Blocklänge

Wir betrachten nun Buchstabenpaare $(a, b) = a \cdot 26 + b$, d.h. wir berechnen die Chiffren in $\mathbb{Z}_{676}$. Damit ist durch einfache Betrachtung der Buchstabenhäufigkeit die Dechiffrierfunktion nicht mehr zu bestimmen, denn es diffundiert der erste Buchstabe in den zweiten. Für jeden zweiten Buchstaben aber ist es genau die gleiche lineare Verschlüsselung mit kleiner Blocklänge wie vorher. Und ein Text wird selten unverständlich, wenn man jeden zweiten Buchstaben wegläßt. .a. k.n. n.c.t .e.n![2]

Danach kann man die Häufigkeiten unter den ersten Buchstaben der Paare mit gleichem zweiten Buchstaben untersuchen. Wieder ist es nur eine Verschlüsselung mit kurzer Blocklänge.

Die zehn häufigsten Bi- und Trigramme der deutschen und englischen Sprache sind :

| Rang | Deutsch | Englisch | Deutsch | Englisch |
|---|---|---|---|---|
| 1 | EN | TH | EIN | THE |
| 2 | ER | HE | ICH | ING |
| 3 | CH | IN | NDE | AND |
| 4 | ND | ER | DIE | HER |
| 5 | EI | AN | UND | ERE |
| 6 | DE | RE | DER | ENT |
| 7 | IN | ED | CHE | THA |
| 8 | ES | ON | END | NTH |
| 9 | TE | ES | GEN | WAS |
| 10 | IE | ST | SCH | ETH |

## 2.4 Nichtlineare Verschlüsselungen

Wir betrachten nichtlineare Verschlüsselungen der Blocklänge 1.

[2] Das kann nicht sein!

Die Verschlüsselung sei eine Permutation der 26 Buchstaben des Alphabets. Als Schlüssel muß die Permutation selbst angegeben werden. Er hat also die Länge 26 und die Anzahl der „sinnvollen“ Schlüssel beträgt $\binom{26!}{2} \approx 2 \cdot 10^{26}$. Es ist also ein sehr großer Schlüsselraum vorhanden. Aber auch dieses System widersteht nicht der Häufigkeitsanalyse über genügend lange Texte. Der Aufwand eines 26 Zeichen langen Schlüsselwortes ist also nicht gerechtfertigt.

# Kapitel 3

# Blockchiffren

Die bis jetzt kennengelernten kryptographischen Verfahren waren Blockchiffren mit geringer Blocklänge und geringer Sicherheit. Zur Erschwerung eines Angriffs bedarf es größerer Blocklängen und besserer Verfahren. In diesem Kapitel werden deshalb einige neue Verfahren und ihre möglichen Schwächen vorgestellt. Zum besseren Verständnis sei hier noch eine Definition nachgereicht.

**Definition 6**

- *Eine Substitution heißt* **monoalphabetisch**, *wenn jedem Zeichen oder jeder Zeichenfolge über einem Alphabet $\mathcal{A}$ eineindeutig ein Zeichen oder eine Zeichenfolge über einem Alphabet $\mathcal{B}$ zugeordnet ist.*
- *Eine Substitution heißt* **polyalphabetisch**, *wenn jedem Zeichen oder jeder Zeichenfolge über einem Alphabet $\mathcal{A}$ ein Zeichen oder eine Zeichenfolge über Alphabeten $\mathcal{B}_1, \ldots, \mathcal{B}_n$ zugeordnet ist.*
- *Eine Substitution heißt* **monographisch**, *wenn sie Einzelzeichen ersetzt.*
- *Eine Substitution heißt* **polygraphisch**, *wenn sie Zeichenfolgen ersetzt.*

Diese Definitionen unterscheiden Kryptosysteme nur bedingt. So läßt sich eine polyalphabetische Substitution der Länge $n$ über einem Alphabet $\mathcal{A}$ als monoalphabetische Substitution über dem Alphabet $\mathcal{A}^n$ auffassen. Die Auswahl der *Zeichen* hängt auch immer von den vorhandenen Gegebenheiten ab, so sind für uns Zeichen manchmal Bits, manchmal Bytes.

## 3.1 Die Vigenère-Chiffre

Eine der wohl bekanntesten polyalphabetischen Substitutionschiffren, die den Prototyp für viele bis in unser Jahrhundert verwendete Algorithmen darstellt, ist die Vigenère-Chiffre. Sie wurde im Jahre 1586 durch den französischen Diplomaten BLAISE DE VIGENÈRE veröffentlicht.

Bei ihr wird eine Nachricht $m = m_1 m_2 \cdots m_n$ über dem Alphabet $\mathcal{A}$ mittels eines Schlüsselwortes $k = k_1 k_2 \cdots k_t$ der Länge $t$ wie folgt verschlüsselt :

$$c_i := E(m_i, k_j) = m_i + k_j \quad \text{mit} \quad j \equiv i \bmod t.$$

Die Addition ist dabei die Addition in der Gruppe des Alphabets $\mathcal{A}$. Die Diffusion der Vigenère-Chiffre ist nicht sehr groß. Allerdings ist dafür die Häufigkeit der einzelnen Buchstaben in der Chiffre viel gleichmäßiger verteilt, d.h. die Konfusion ist größer.

So erhält man für $\mathcal{A} = \mathbb{Z}_{26}$ mit dem Schlüsselwort „Blume“ :

| D | i | e | s | | i | s | t | | d | e | r | | K | l | a | r | t | e | x | t |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *B* | *L* | *U* | *M* | | *E* | *B* | *L* | | *U* | *M* | *E* | | *B* | *L* | *U* | *M* | *E* | *B* | *L* | *U* |
| E | T | Y | E | | M | T | E | | X | Q | V | | L | W | U | D | X | F | I | N |

| e | i | n | e | r | | V | i | g | e | n | e | r | e | - | C | h | i | f | f | r | e |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *M* | *E* | *B* | *L* | *U* | | *M* | *E* | *B* | *L* | *U* | *M* | *E* | *B* | | *L* | *U* | *M* | *E* | *B* | *L* | *U* |
| Q | M | O | P | L | | H | M | H | P | H | Q | V | F | | N | B | U | J | G | C | Y |

Die Analyse der Vigenère-Chiffre beginnt mit gründlichem Textstudium. Dabei werden längere gleiche Textstellen gesucht. Der Abstand dieser gleichen Textstellen ist somit vermutlich ein Vielfaches der Länge $t$ des Schlüssels. Gesucht ist also der größte gemeinsame Teiler mehrerer dieser möglichen Abstände. Eine statistische Analyse der $(i+nt)$-ten Buchstaben liefert damit die verschiedenen Buchstaben. Die Vigenère-Chiffre ist demnach für größere Texte ungeeignet, da diese leicht zu analysieren sind.

**Übungsaufgabe 3** *Führen Sie eine Kryptoanalyse der folgenden Vigenère-Chiffre durch !*

```
BFQTK  CDSXW  WMJHV  IIHWM  CRWVD  OLLVA  SXZYV  IISAR
VFKIZ  HKZVF  XEYFW  RVFKM  EBLAZ  SEEER  SXVBN  SIIFF
VFMJV  BWRMS  JFVFY  IEYSR  EIFHU  SJEER  WVVFK  IZHKE
LQZZV  FKYTV  LHZSK  IUOLI  ECZRV  SJPRI  TRZGR  YCSKI
ERWVV  BLALF  XHZSK  IIJWV  WOZVV  BOMIR  SPJYJ  CGHGK
IOHLZ  SVIII  FFVFM  JVBWI  ZBYVZ  TXECG  CVPDL  SRBSP
PGWFV  NWMTV  FIKTM  IISAR  VBWVW  CDKIS  AGYSF  HRHWR
JQZYK  NMRUS  ARVRS  YVFZE  WHWWZ  QZIII  FKUSJ  HRHWR
DIKWD  OFHZS  KXRSJ  OVBMR  UGULN  OWGYS  FHVFT  IEILD
KSFZV  FXEYF  WRBSF  RVBVI  EBWME  OTWFZ  MXJWU  LVFWW
JMKXV  AOMIR  EEEBA  GYHWV  JHWPC  SFOFS  FRVBA  QDSJA
ZFVIJ  GULNO  ULGIF  OKSYI  SSFHZ  SEEEB  AGYHN  IIVAR
USJRB  OFRUS  JLRIH  XJQZA  RQZTL  BCXZG  LHRPW  MUSJQ
VBKGY  RWVDW  LHZSK  IERSX  VBMQX  SZXUO  VMVBS  GYFAG
YHWRQ  IWMES  EKVKA  WJSFD  VWLTL  BCXZA  CPRFL  IOHNS
IVSRU  SFWZB  VFVWV  IISAR  XOTIS  SAWGW  WPJKW  MJSCS
VBFIE  RAIUO  LIEGU  LFBYI  CSKIE  KWVUS  FDLFC  SEHJS
CZWJV  FLMXH  EEEGA  GYCXX  ECULV  WFIEO  MWUFM  GBRWW
BZSVK  SPXVG  VIIBA  GYHGV  UBMRX  GYIDO  WWJJW  VEWUL
KSLAZ  FVYER  AVXSF  HNCDM  VUWRS  ZWMSH  VMVIF  ETVLW
RACIZ  HWRSS  AQLAY  EEUEM  KGWRJ  WTPVB  VEKSF  OFSFR
VBVYI  QZHZS  TIJHW  ZVFKG  YZMIJ  GWPLB  YRZQZ  XMSJL
ZBVII  HOIIR  WRUWW  OIMHX  FUJEG  VAIDI  KWJCD  GYSNI
ITSLI  SFIEH  OIITW  RUWWP  VWULK  NMMDD  DIDSF  XZSJI
EGARU  SARJQ  ZRVZD  IJJWV  JQZPL  SKWVZ  FYERW  RKGUL
CIWWJ  SDRXS  OEVVJ  PVWKX  VBMRU  RAIUO  KYEPW  VLTWR
VZWWV  BESUW  XMQWW  VVBMR  UGAQL  ZAIIS  FHVFV  EKSFZ
VFZME  RWVER  SWJWF  HVWFE  ERWVN  WWHVF  KTISU  LVBVI
WCJHV  FMRXS  FYERE  EEAMW  JBSGY  SARVA  CSDDJ  SDWKW
QKAWT  VWRUS  FWZQZ  IIVWM  KGSRW  CJHVF  MRXSF  YERVI
EAGIX  ZAGYY  WMKSF  WLQZI  EKWVS  WKLZS  JLZBS  PCSKI
EHKGY  ZMIJG  WPKVS  XYOWX  KSFYI  RAIVW  FPVWL  YEUJM
TVLMX  ZWWVB  KSCZW  R
```

## 3.2 Die Hill-Chiffre

Eine andere Möglichkeit die normale Häufigkeitsverteilung der einzelnen Buchstaben im Chiffrentext zu verändern, ist die Anwendung einer polygraphischen Substitutionschiffre. Bei ihr werden Zeichenketten des Klartextes komplett durch Zeichenketten der Chiffre ersetzt. Ein Vertreter ist die von LESTER HILL 1929 veröffentlichte Chiffre. Die Blocklänge sei weiterhin $t$. Falls nun das Alphabet $\mathcal{A}$ ein Körper ist, so ist $\mathcal{A}^t$ ein Vektorraum über $\mathcal{A}$. Die lineare Verschlüsselungsfunktion sei nun

$$E_t(m,k) = \boldsymbol{A}m + b \quad \text{mit} \quad k = (\boldsymbol{A}, b)\,.$$

Dabei ist $\boldsymbol{A}$ eine $(t \times t)$-Matrix und $b$ ein $t$-dimensionaler Spaltenvektor.

Sei zum Beispiel $t = 2, \boldsymbol{A} = \binom{2\ 3}{7\ 8}, b = \binom{0}{0}$ und die Nachricht $m =$ „NO“ $= \binom{13}{14}$. Man erhält

$$c = \boldsymbol{A}m = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix} \cdot \begin{pmatrix} 13 \\ 14 \end{pmatrix} = \begin{pmatrix} 16 \\ 21 \end{pmatrix} = \text{„QV“}.$$

Die Diffusion vergrößert sich dadurch etwas, aber es beeinflussen sich jeweils nur die $t$ Buchstaben in einem Block. Zum Dechiffrieren der Hill-Chiffre benötigt man die zu $\boldsymbol{A}$ inverse Matrix $\boldsymbol{A}^{-1}$ und es gilt

$$m = D(c,k) = \boldsymbol{A}^{-1}(c-b) \quad \text{mit} \quad k = (\boldsymbol{A}, b)\,.$$

Dadurch kommt es zu folgenden Bedingungen :

- Falls das Alphabet $\mathcal{A}$ ein Körper ist, so ist die Matrix $\boldsymbol{A}$ genau dann invertierbar, wenn $\det\boldsymbol{A} \neq 0$.
- Falls das Alphabet $\mathcal{A} = \mathbb{Z}_n$ kein Körper ist, so muß $\text{ggT}(\det\boldsymbol{A}, n) = 1$ sein, damit ein arithmetisches Inverses $\boldsymbol{A}^{-1}$ existiert.

Bei der Analyse hat man im Fall des gewählten Klartextes (*chosen plaintext attack*) keine Probleme. Man braucht sich nur die Einheitsvektoren verschlüsseln zu lassen. Im Fall des bekannten Klartextes (*known plaintext attack*) gilt für $t = 2$ :

$$\begin{aligned} c_1 &= \boldsymbol{A}m_1 + b, \\ c_2 &= \boldsymbol{A}m_2 + b \qquad \text{und} \\ c_3 &= \boldsymbol{A}m_3 + b\,. \end{aligned}$$

Daraus ergibt sich

$$C := \begin{pmatrix} c_1 - c_3 & c_2 - c_3 \end{pmatrix} = \boldsymbol{A} \cdot \begin{pmatrix} m_1 - m_3 & m_2 - m_3 \end{pmatrix} =: \boldsymbol{A} \cdot M$$

und man erhält $\boldsymbol{A} = C \cdot M^{-1}$.

**Übungsaufgabe 4** *Entschlüsseln Sie „*`FWMDIQ`*“ mit der Matrix $\boldsymbol{A}$ aus dem obigen Beispiel! Man beachte, daß die inverse Matrix $\boldsymbol{A}^{-1}$ im Restklassensystem modulo $26$ zu berechnen ist!*

## 3.3 Die Playfair-Chiffre

Eine weitere polygraphische Substitutionschiffre, die nach Lord LYON PLAYFAIR benannte Playfair-Chiffre, wurde 1854 von dem britischen Wissenschaftler CHARLES WHEATSTONE entwickelt. Diese einfache Blockchiffre wurde von den Engländern während des ersten Weltkrieges eingesetzt. Bei ihr werden jeweils Buchstabenpaare aus einem Alphabet von 25 Buchstaben (z. B. wird statt des Buchstabens 'J' ein 'I' geschrieben) verschlüsselt.

Der Schlüssel ist eine $5 \times 5$-Buchstabenmatrix, in der die 25 Buchstaben des Alphabetes beliebig angeordnet sind. Man kann sich die Matrix durch eine Schlüsselphrase (z. B. „Blumen sind schoen") leicht merken.

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

Die Chiffrierung erfolgt dabei nach folgenden Regeln :

1. Liegen die beiden zu verschlüsselnden Buchstaben in einer Zeile, so sind die jeweils rechten Nachbarn ihre Kryptogramme (als die rechten Nachbarn der rechten Spalte gelten die Buchstaben der linken Spalte). So ist zum Beispiel $E($„KP"$) =$ „PQ".

2. Liegen die beiden Buchstaben in einer Spalte, so sind die jeweils unteren Nachbarn ihre Kryptogramme (als die unteren Nachbarn der untersten Zeile gelten die Buchstaben der obersten Zeile). So ist zum Beispiel $E($„ET"$) =$ „CZ".

3. Liegen die beiden Buchstaben in den diagonalen Ecken eines Rechtecks, so sind die anderen Ecken in der jeweiligen Zeile die Kryptogramme. So ist zum Beispiel $E($„DT"$) =$ „CR".

4. Sind beide Buchstaben gleich, wird zwischen ihnen ein vereinbarter Zwischenbuchstabe (z. B. 'X') eingefügt und verschlüsselt. ( Dieser Fall kann entfallen, wenn Doppelbuchstaben grundsätzlich vermieden werden; z. B. „Sume" statt „Summe".)

5. Ist nur noch ein Buchstabe zu verschlüsseln, so wird ein vereinbarter Buchstabe (z. B. 'X') angefügt und dann verschlüsselt.

Zur Dechiffrierung werden diese Regeln umgekehrt angewandt. Insgesamt gibt es $25!$ verschiedene Schlüsselmatrizen, da aber dieselbe Playfair-Chiffre durch 25 unterschiedliche Matrizen beschrieben werden kann, gibt es nur $24!$ (immerhin wenigstens $6{,}2 \cdot 10^{23}$) verschiedene Chiffrierfunktionen. Zur Vergrößerung dieser Anzahl kann das Alphabet mit den Ziffern $2, 3, 4$ und $5$ auf 30 Zeichen erweitert und eine $(5 \times 6)$-Schlüsselmatrix angewandt werden.

Einige Eigenschaften der Playfair-Chiffre sind :

- Gilt $E(m_1 m_2) = c_1 c_2$, so ist $E(m_2 m_1) = c_2 c_1$.
- Jedes Klartextzeichen besitzt nur 5 verschiedene Bilder, die Buchstaben derselben Zeile und seinen unteren Nachbarn.
- Ein Zeichen wird dabei mit größerer Wahrscheinlichkeit auf seinen rechten Nachbarn abgebildet, als auf die anderen Buchstaben.
- Haben zwei Zeichen sich gegenseitig als Bilder, so liegen sie in einer Zeile.

Auf Grund dieser Eigenschaften kann die Playfair-Chiffre relativ leicht analysiert werden.

**Übungsaufgabe 5** *Entschlüsseln Sie „*`AE ZT BE GI GK`*" mit obiger Schlüsselmatrix!*

## 3.4 Die Dellastelle-Chiffre

Eine ähnliche Chiffre ist die Dellastelle-Chiffre. Das Alphabet umfasse wieder 25 Buchstaben. Der Schlüssel sind diesmal vier $(5 \times 5)$-Matrizen $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ und $\boldsymbol{M}_4$, in denen die 25 Buchstaben des Alphabetes willkürlich verteilt sind. Die Matrizen werden dabei folgendermaßen angeordnet :

$$K = \begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_3 \\ \boldsymbol{M}_4 & \boldsymbol{M}_2 \end{pmatrix}.$$

Zum Chiffrieren bilden die beiden zu verschlüsselnden Zeichen die diagonalen Ecken eines Rechtecks. Dabei sind das erste Zeichen in Matrix $\boldsymbol{M}_1$ und das zweite Zeichen in Matrix $\boldsymbol{M}_2$ zu suchen. Die Kryptogramme ergeben sich dann aus den anderen Ecken des Rechtecks in Matrix $\boldsymbol{M}_3$ und Matrix $\boldsymbol{M}_4$.

Zur Vereinfachung des Verfahrens seien nur die Matrizen $\boldsymbol{M}_1$ und $\boldsymbol{M}_2$ gegeben und die beiden anderen Matrizen $\boldsymbol{M}_3$ und $\boldsymbol{M}_4$ ergeben sich durch Zeilen- und Spaltenvertauschung aus den gegebenen Matrizen. Sei $\boldsymbol{M}_i = (m_{kl}^{(i)}),\ \ 1 \leq k, l \leq 5$, so gelte

$$m_{kl}^{(3)} = m_{z_1(k)s_1(l)}^{(1)} \quad \text{und} \quad m_{kl}^{(4)} = m_{z_2(k)s_2(l)}^{(2)} \quad \text{mit } 1 \leq k, l \leq 5.$$

Sind die Permutationen $z_1, z_2, s_1$ und $s_2$ so gewählt, daß $z_1^2 = z_2^2 = s_1 \cdot s_2 = \text{id.}$, so ist die Chiffrierfunktion $E$ involutorisch, d.h. Verschlüsseln und Entschlüsseln sind gleich. Der Schlüssel besteht dann nur noch aus zwei Matrizen (z. B. gegeben durch Schlüsselphrasen „Blumen sind schoen“ und „Chrysanthemen erst recht“) und drei Permutationen (z. B. $z_1 = (1\,3)(2\,5), z_2 = (2\,3)(4\,5)$ und $s_1 = (1\,2\,3\,5\,4) = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \end{smallmatrix}\right)$).

Dadurch erhält man einen Schlüsselraum von $\frac{1}{5!}\left(\frac{26!}{5!}\right)^2 \approx 9{,}4 \cdot 10^{46}$ nichtäquivalenten involutorischen Dellastelle-Chiffren.

**Übungsaufgabe 6** *Entschlüsseln Sie „*`ZBHWUSTGHWGP`*“ mit den oben als Beispiel gegebenen Schlüsselphrasen „Blumen sind schoen“ bzw. „Chrysanthemen erst recht“ und den Permutationen $z_1$, $z_2$ und $s_1$!*

## 3.5 Transpositionschiffren

Das wohl älteste Chiffrierverfahren, die von den Griechen schon vor ca. 2400 Jahren benutzte *Skytala*, ist eine einfache Transpositionschiffre. Im Gegensatz zu Substitutionschiffren werden bei Transpositionschiffren die Klartextzeichen nicht modifiziert, sondern nur in ihrer Anordnung verändert.

So schrieben die Griechen ihre Nachricht auf mit Papyrusstreifen umwickelte Zylinder (Skytala) und „übermittelten“ die abgewickelten Streifen, auf denen die Nachricht in permutierter Reihenfolge nur mit einem Zylinder gleichen Durchmessers zu lesen war. Es bleiben zwar die statistischen Häufigkeiten einzelner Zeichen erhalten, nicht aber die der Buchstabenpaare oder Buchstabentripel.

Transpositionschiffren sind Blockchiffren der Länge $t$, bei denen jeder Block einer Schlüsselpermutation $\pi$ unterworfen wird, die jedem Zeichen eine neue Position im Block zuweist.

Die Chiffrierfunktion ist demnach

$$E(m) = m_{\pi(1)} \cdots m_{\pi(t)} \qquad \text{mit } m = m_1 \cdots m_t.$$

Das Dechiffrieren eines Blockes geschieht dann mit der Permutation $\pi^{-1}$.

Es wird im allgemeinen von der Permutation erwartet, daß sie den Klartext möglichst gut mischt. Deshalb gelten folgende Kriterien zur Wahl der Permutation $\pi$ :

1. Bigramme dürfen nicht erhalten bleiben.
2. Bigramme sollten auch nicht invertiert in der Chiffre auftauchen.
3. Der mittlere Abstand von Bild und Urbild soll möglichst groß sein. So ist $|i - \pi(i)| \sim \frac{t}{2}$ erstrebenswert.
4. Der mittlere Abstand benachbarter Urbilder sollte ebenfalls ca. der halben Blocklänge entsprechen.

Schließlich sollen noch einige Transformationen vorgestellt werden.

So kann als Schlüssel eine Permutation $\pi = (1\ 5\ 2\ 7)(0\ 9\ 3)(4\ 8\ 6)$ gegeben sein und man erhält aus dem Klartext „Mathematik“ die Chiffre „`HTMKAAITEM`“.

Man kann aber auch den Klartext zeilenweise in eine Matrix mit vorher vereinbarter Größe schreiben und ihn spaltenweise als Chiffre wieder auslesen. So erhält man mit einer $(5 \times 5)$-Matrix das Kryptogramm „`KOET SRL MVPYO AI APGCES TIHLS`“. Beim Eintragen des Klartextes und beim Auslesen der Chiffre sind dabei der Phantasie keine Grenzen gesetzt.

So kann der Text auch im Zickzack geschrieben werden, und man erhält

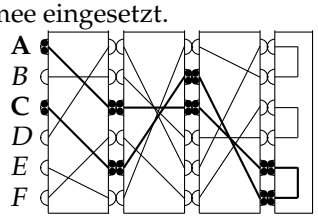
```
T   S   I   N   I   E
 R N P S T O S H F R   ⟹   TSINIERNPSTOSHFRAOICF
  A   O   I   C   F
```

Durch Kombination von Transpositions- und Substitutionschiffren kann die Sicherheit der Chiffre weiter erhöht werden. Im allgemeinen wächst dabei aber auch deutlich der Aufwand, so daß man sich dann besser nach einem einfacher zu handhabenden System umsieht, das die gleiche Sicherheit bietet.

## 3.6 Rotormaschinen – die ENIGMA

Die industrielle Entwicklung machte, gefördert durch die Geheimdienste, auch nicht vor der Kryptographie halt. So wurde sehr früh auch schon versucht kryptographische Verfahren durch Maschinen zu implementieren.

Zur Realisierung leistungsfähiger elektromechanischer Chiffriergeräte werden sogenannte Rotoren benutzt. Ein Rotor ist eine Zylinderscheibe mit meist 26 Kontakten an jeder Seite, die eine einfache Substitutionschiffre realisieren. Durch die bewegliche Anordnung mehrerer Rotoren erhält man eine polyalphabetische Substitutionschiffre. Diese Rotormaschinen wurden im zweiten Weltkrieg in großem Umfang eingesetzt. So wurden und werden vielleicht noch heute, die von BORIS HAGELIN entwickelten Chiffriermaschinen Hagelin C-36 (auch M-209) und Hagelin C-48, in der US-amerikanischen Armee eingesetzt.




Eine weitere Rotormaschine ist die 1923 von dem Deutschen ARTHUR SCHERBIUS erfundene ENIGMA. Sie arbeitet mit drei (später vier) Rotoren, die zwischen zwei feststehenden Scheiben montiert sind. Die eine Scheibe dient zur Ein- und Ausgabe, während die andere ein ankommendes Signal über einen anderen Pfad „zurückschickt“. Durch diese Konstruktion wird

ein Buchstabe nie auf sich selbst abgebildet und das Ver- bzw. Entschlüsseln ist gleich. Wurde ein Buchstabe verschlüsselt, so wird die Walzenstellung geändert. Dies geschieht nach dem „Kilometerzählerprinzip“, wobei die zweite Scheibe bei einem Übertrag einen zusätzlichen Schritt ausführt. Damit hat die polyalphabetische Substitution der ENIGMA eine Periode von $26 \cdot 25 \cdot 26 = 16900$.

Der Schlüsselraum der Rotormaschinen ergibt sich somit aus

- der Auswahl der benutzten Rotoren (Die ENIGMA hat ca. 10 verschiedene Rotoren.),
- der Stellung der Rotoren zu Beginn der Verschlüsselung,
- der Anordnung der Rotoren in der Rotormaschine und
- der initialen Permutation.

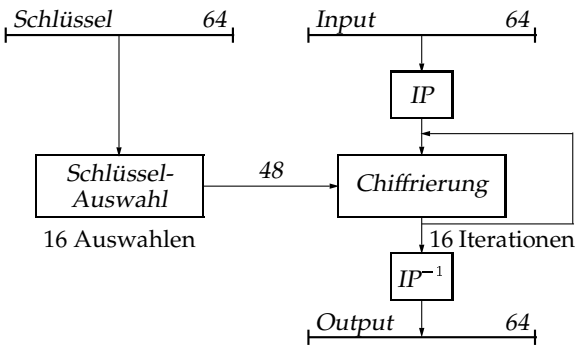
So ergibt sich für die ENIGMA ein Schlüsselraum von ungefähr $2 \cdot 10^{20}$ verschiedenen Schlüsseln. Daß die ENIGMA trotz dieses großen Schlüsselraums und der langen Periode einer Analyse der Polen, Franzosen und Engländer im zweiten Weltkrieg nicht stand hielt, verdankt sie auch den verschiedenen Schwachstellen beim Gebrauch der Maschine. So wurden lange Zeit die Rotorpositionen vor jeder Nachricht gesendet und die kurzen, typisch militärischen Nachrichten erhöhten auch nicht gerade die Sicherheit.

## 3.7 Der Data Encryption Standard

Der DES ist eine Blockchiffre, die im Jahre 1977 als US-Verschlüsselungsstandard vom National Bureau of Standards (NBS) genormt wurde. Bei ihm werden binär dargestellte Nachrichten durch eine Folge von Permutationen, nichtlinearen Substitutionen und logischen Produktbildungen mit einem individuellen Schlüssel verschlüsselt.

Der Klartext ist in 64-Bit Blöcken gegeben. Des weiteren wird noch ein 64-Bit langer Schlüssel benötigt, bei dem 56 Bit beliebig sind und die 8 weiteren Bits je acht 7-Bit Blöcke auf die ungerade Parität ergänzen. Somit stehen zirka $7{,}2 \cdot 10^{16}$ unterschiedliche Schlüssel zur Verfügung.

Zuerst wird der 64-Bit Inputblock einer Eingangspermutation *IP* unterzogen. Anschließend durchläuft er 16 schüsselabhängige Iterationen. Die dabei benötigten 48-Bit langen Teilschlüssel werden durch ein Schlüsselauswahlverfahren aus dem 64-Bit-Schlüsssel generiert. Nach den Iterationen wird dann noch die zur Eingangspermutation inverse Ausgangspermutation $IP^{-1}$ angewendet.




Prinzipieller Aufbau des DES

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Initialpermutation IP

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Permutation $IP^{-1}$

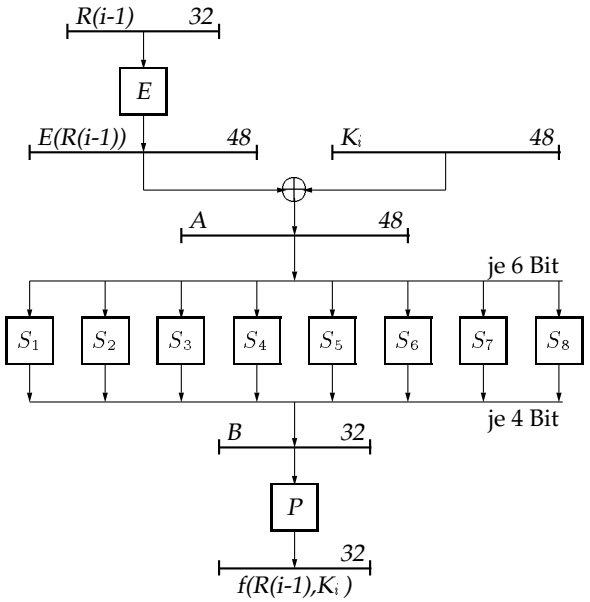
Die Entschlüsselung des so erhaltenen Blockes geschieht analog der Verschlüsselung, wobei dazu nur die 16 Teilschlüssel in der umgekehrten Reihenfolge benötigt werden. In diesem Sinne ist die DES-Chiffre ein symmetrisches Kryptosystem. Die Eingangspermutation *IP* erscheint überflüssig, sie dient ausschließlich dazu, Software-Lösungen zu erschweren, denn eine Permutation läßt sich ohne wesentliche Zeitverzögerung zwar verdrahten, jedoch nicht implementieren (auf einer sequentiellen Maschine).

Vor der Verschlüsselungsiteration wird der 64-Bit Block in einen linken Block $L$ und einen rechten Block $R$ zu je 32 Bit unterteilt. Danach wird darauf die Verschlüsselungsiteration angewendet:

$$\begin{array}{llll} L(0)=L & , & R(0)=R & \text{und} \\ L(i)=R(i-1), & & R(i)=L(i-1)\oplus f(R(i-1),K_i) & \text{, für } i=1,\ldots,16; \end{array}$$

wobei $\oplus$ die Addition modulo 2 bzw. die logische Verknüpfung XOR ist.

Die Funktion $f$ ist der Kern des gesamten Verfahrens. Bei ihr wird der 32-Bit Inputblock $R(i-1)$ der $i$-ten Iteration zuerst mit der linearen Expansionsabbildung $E$ auf einen 48-Bit Block erweitert, indem ausgewählte Bits von $R(i-1)$ dupliziert werden. Danach werden der 48-Bit Block $E(R(i-1))$ und der durch die Schlüsselauswahl erhaltene 48-Bit Teilschlüssel bitweise modulo 2 addiert.




Die Funktion $f(R(i-1), K_i)$

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Expansion $E$

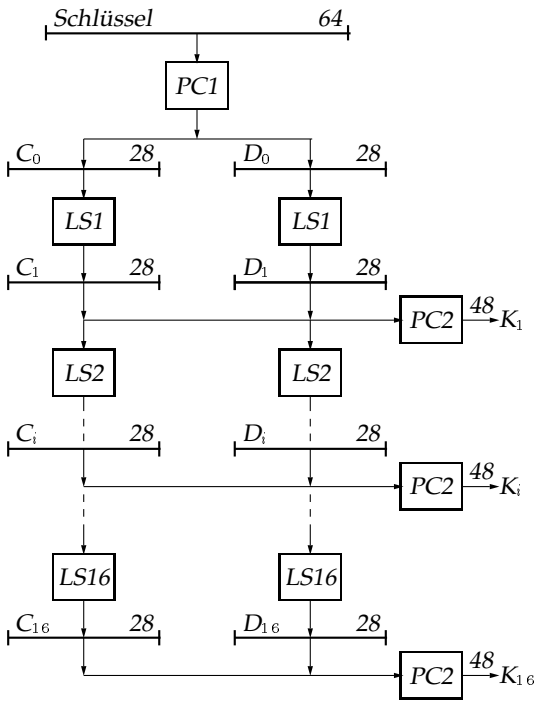
| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

Permutation $P$

Der so erhaltene Block $A$ wird in acht 6-Bit Blöcke zerlegt, die den Input für die S-Boxen $S_1, \ldots, S_8$ bilden. Jede der acht S-Boxen des DES substituiert nichtlinear den 6-Bit Inputblock $(i_1, i_2, i_3, i_4, i_5, i_6)$ durch einen 4-Bit Outputblock $(o_1, o_2, o_3, o_4)$. Dabei werden die durch die Expansion redundanten Bits $i_1$ und $i_6$ zur Auswahl einer bijektiven 4-Bit Substitution benutzt, die auf die verbleibenden Bits $(i_2, i_3, i_4, i_5)$ angewendet wird.

Bei der Schreibweise einer $4 \times 16$ - Matrix für die S-Boxen bedeutet somit der Input $(0, 1, 1, 0, 1, 1)$, daß das Ergebnis der S-Box in Zeile $(0, 1)$ und Spalte $(1, 1, 0, 1)$ zu suchen ist. Dies ergibt für die erste S-Box den Output $(0, 1, 0, 1)$. Der so erhaltene 32-Bit Block $B$ wird nun noch der Permutation $P$ unterworfen und ergibt den Funktionswert $f(R(i-1), K_i) = P(B)$ der $i$-ten Iteration.

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




Schlüsselauswahlfunktion

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 |

Permuted Choice 1

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Permuted Choice 2

Nun bleibt nur noch zu zeigen, wie die 16 Teilschlüssel für die Iterationen aus dem gegebenen Schlüssel $K$ generiert werden. Dazu wählt die Funktion $PC1$ (*permuted choice 1*) aus dem Schlüssel $K$ die kryptographisch relevanten Bits aus und permutiert sie.

Das erhaltene Ergebnis wird in zwei 28-Bit Blöcke $C_0$ und $D_0$ unterteilt. Die einzelnen Blöcke werden dann zyklisch um 1 oder 2 Bit verschoben. Dabei wird bei $LS1, LS2, LS9$ und $LS16$ um 1 und sonst um 2 nach links geshiftet. Anschließend bestimmt die Funktion $PC2$ (*permuted choice 2*) den Teilschlüssel $K_i$ aus den Blöcken $C_i$ und $D_i$ durch Entfernen der Bits an den Positionen $9, 18, 22, 25, 35, 38, 43$ und $56$, sowie der Permutation der verbleibenden 48 Bits.

Damit ist der DES vollständig beschrieben. Bleiben noch einige Betrachtungen über seine Sicherheit.

Durch die Expansion $E$, die Permutation $P$ und die acht Substitutionsboxen $S_1, \ldots, S_8$ hängt bereits nach 5 Iterationen jedes Bit der Blöcke $R$ und $L$ von jedem Bit des Inputs und von jedem Bit des Schlüssels ab. So kommt es, daß eine Bitänderung im 64-Bit Inputblock, schon nach 6 Iterationen eine Änderung von rund 32 Bit erreicht, d.h. „kleine“ Änderungen im Klartext oder im Schlüssel führen zu „großen“ Änderungen in der Chiffre.

Durch das verwendete Schlüsselauswahlverfahren gibt es aber 4 sogenannte „schwache“ Schlüssel mit der Eigenschaft $K_i = K_j$, für alle $i, j \in \{1, \ldots, 16\}$. Diese lauten

`0101010101010101, FEFEFEFEFEFEFEFE, 1F1F1F1F1F1F1F1F, E0E0E0E0E0E0E0E0`

und für sie gilt

$$\text{DES}(\text{DES}(X, K), K) = X \quad .$$

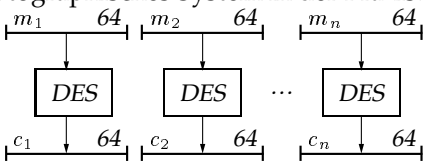
Neben diesen schwachen Schlüsseln existieren noch 6 weitere „semi-schwache“ Schlüsselpaare $(K, K')$, für die gilt

$$\text{DES}(\text{DES}(X, K), K') = X \quad .$$

Diese Schlüssel sind ungeeignet und sollten deshalb gemieden werden. Allgemeiner benutze man keine Schlüssel mit Periode.

Der DES kann in verschiedenen Betriebsarten eingesetzt werden. Vier von ihnen wurden 1980 im Zusammenhang mit der Standardisierung des DES normiert. Diese vier Betriebsarten können ohne weiteres mit jeder symetrischen Blockchiffre eingesetzt werden und gewährleisten die meisten Anforderungen an ein kryptographisches System in der Praxis.

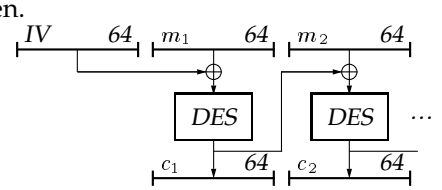
Im **ECB-Modus** (*electronic code book mode*) wird der Klartext in 64-Bit Blöcke zerlegt und jeder einzelne Block nacheinander und unabhängig voneinander mit dem gleichen Schlüssel verschlüsselt. Dabei kommt es aber zu mindestens zwei Nachteilen:




Gleiche Klartextblöcke liefern gleiche Chiffren und können somit Informationen über den Klartext liefern (Textstruktur, statistische Analyse).

Der zweite Nachteil ist, daß der unberufene Entschlüssler die Chiffre ändern kann, ohne daß der Empfänger der Nachricht dies bemerkt. Der ECB-Modus sollte deshalb nur beim Verschlüsseln kurzer Nachrichten Verwendung finden.

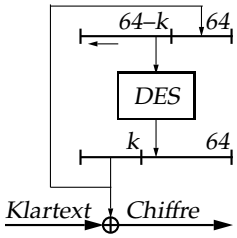
Im **CBC-Modus** (*cipher block chaining mode*) wird der Klartext, genau wie im ECB-Modus, in 64-Bit Blöcke zerlegt. Danach wird jeder Block mit dem Resultat der Verschlüsselung des vorhergehenden Blockes bitweise modulo 2 addiert und selbst verschlüsselt. Für den ersten Block wird, da er keinen Vorgänger besitzt, dabei ein Initialisierungsvektor



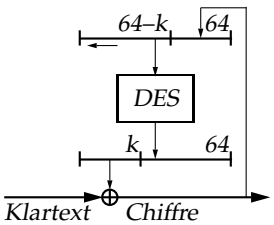
$IV$ benutzt. Somit werden identische Klartextblöcke im allgemeinen auf unterschiedliche Chiffrenblöcke abgebildet. Zum anderen hat der Emfänger die Möglichkeit, Änderungen der Chiffre zu erkennen.

Bei der Entschlüsselung wird analog das Resultat der Entschlüsselung eines Chiffrenblockes mit dem vorhergehenden Chiffrenblock bzw. dem Initialisierungsvektor bitweise modulo 2 addiert. Somit wirken sich Übertragungsfehler eines Blockes bei der Entschlüsselung nur auf diesen und den nachfolgenden Block aus.

Für die Verschlüsselung sequentiell generierter Klartexte, d.h. falls nur einzelne Bits oder Zeichen ohne Verzögerung übertragen werden müssen, sind die beschriebenen blockorientierten Betriebsarten weniger gut geeignet.




Im **OFB-Modus** (*output feedback mode*) generiert der DES aus einem Initialisierungsvektor und einem Schlüssel durch mehrmalige Anwendung eine pseudo-zufällige Bitfolge, die modulo 2 zu den Klartextzeichen bzw. den Chiffrenzeichen addiert wird. Der OFB-Modus hat aber mehrere Nachteile. So bietet diese Betriebsart nur wenig Schutz gegen eine Änderung des Chiffrentextes durch Unbefugte. Außerdem können Synchronisationsprobleme auftreten. Besonders wichtig ist, daß zur Verschlüsselung unterschiedlicher Nachrichten nicht der gleiche Initialisierungsvektor benutzt wird, da unter Umständen durch bitweise Addition modulo 2 die Nachrichten zurückgewonnen werden können.




Im **CFB-Modus** (*cipher feedback mode*) wird analog dem OFB-Modus eine pseudo-zufällige Bitfolge generiert, nur daß das Chiffrenzeichen diesmal als Eingabe für die nächste Anwendung des DES rückgekoppelt wird. Dadurch wird eine Verkettung der Nachrichten, wie im CBC-Modus erreicht. Analog ist es damit möglich, auch Änderungen der Chiffre zu erkennen.

Mittlerweile existieren viele Implementierungen des DES in Form von Schaltkreisen, fertigen Leiterplatten oder Programmen. So findet der DES, trotz aller Kritik gegen die kleine Schlüssellänge und der nicht veröffentlichten Entwurfskriterien bzw. Sicherheitstests, eine breite Anwendung.

# Kapitel 4

# Flußchiffren

Eine Flußchiffre wird dadurch gebildet, daß eine Folge von Zeichen, der Fluß, die Folge der Nachrichtenzeichen überlagert.

Die exakte Definition ist die folgende:

**Definition 7** *Gegeben ist eine Menge $M$ von Klartexten, als Menge der endlichen Wörter über einem Alphabet $\mathcal{A}$, eine Menge $C$ von Chiffren als Menge endlicher Wörter über einem Alphabet $\mathcal{B}$, und ein Kryptosystem von $\mathcal{A}$ nach $\mathcal{B}$ mit der Schlüsselmenge $\mathcal{K}$.*
*Wir bilden daraus ein Flußchiffrensystem von $M$ nach $C$ mit der Schlüsselmenge $K$, einer Teilmenge der Folgen über dem Alphabet $\mathcal{K}$.*
*Die Verschlüsselungsfunktion $E$ ist*

$$E(m,k)_i = e(m_i, k_i),$$

*und die Entschlüsselungsfunktion ist*

$$D(c,k)_i = d(c_i, k_i),$$

*wobei jeweils $e$ und $d$ die Ver- und Entschlüsselungsfunktionen aus einem „kleinen" Kryptosystem sind.*

Der Witz bei der Flußchiffre besteht nicht in der Komplexität des kleinen Kryptosystems, das kann eine einfache Cäsarchiffre sein, sondern in der Auswahl des Schlüsselflusses. Selbstverständlich kann man in die Ver- und Entschlüsselung eine Rückkopplung einbauen, etwa $E(m,k)_i = e(m_i, m_{i-1}, k_i)$. Wir wollen aber der Einfachheit halber nur solche Flußchiffren betrachten, bei denen die Flüsse nachrichtenunabhängig sind. Im Übrigen ist das vielleicht unsicherer, wenn man an den unberufenen Entschlüsseler denkt. Bei einer Übertragung auf einem instabilen, fehlerbehafteten Kanal ist es meist zweckmäßiger.

Wir werden im folgenden mehrere Möglichkeiten dafür betrachten.

## 4.1 Blockchiffren

Dieser Abschnitt dient ausschließlich der eigenen Überprüfung. Erkennen Sie, daß jede Blockchiffre eine Flußchiffre mit periodischem Fluß ist? Wenn ja, so lesen Sie im nächsten Abschnitt weiter, wenn nein, so schlagen Sie das Kapitel Blockchiffren nach und wiederholen Sie es.

## 4.2 Flüsse aus Zufallsfolgen

An dieser Stelle finden sich die einzigen wahrhaft sicheren Kryptosysteme. Wir gehen dabei davon aus, daß die gesendeten Nachrichten aus einer bestimmten Quelle stammen und für die Wahrscheinlichkeiten der gesendeten Nachrichten gewisse Werte gelten, die dem Kryptoanalytiker bekannt sind. Einen in einer ihm unbekannten Sprache geschriebenen und verschlüsselten Text wird er nie verstehen können. Aber wir wollen davon ausgehen, daß er die Sprache kennt und aus den vorangegangenem Nachrichtenaustausch auch über Häufigkeiten der von uns verwendeten Wörter Bescheid weiß. Für ihn hat also jede Nachricht $m$ eine bestimmte Wahrscheinlichkeit $P\{m\} = p_m$ mit der sie auftritt. Im einfachsten Fall gehe man davon aus, daß jeder Buchstabe $m_i$ eine bestimmte Häufigkeit $p_i$ hat (siehe Abschnitt 2.1), und ein Wort $m_1 m_2 \dots m_k$ mit der Wahrscheinlichkeit $p_1 p_2 \cdots p_k$ auftritt. Das ist natürlich in einer gesprochenen Sprache nicht so, da spielen, wie wir wissen die Paar- und Tripelhäufigkeiten eine nicht zu unterschätzende Rolle und man müßte eigentlich $P\{m_i|m_1, m_2, \dots, m_{i-1}\}$, die Wahrscheinlichkeiten für den Buchstaben $m_i$ unter der Voraussetzung, daß vorher die Buchstaben $m_1, m_2, \dots$ auftraten, multiplizieren. Bei zu großem Abstand wird aber die Abhängigkeit der Buchstaben von den Vorgängern geringer und man wird sich in der Praxis auf Paare oder Tripel einstellen.

Für den Kryptoanlytiker sind also die Nachrichten $m$ mit einer ihm bekannten Wahrscheinlichkeit $P\{m\}$ des Auftretens behaftet.

**Definition 8** *Nach* SHANNON *definieren wir ein Kryptosystem als sicher, wenn für alle möglichen Nachrichten $m$ die Wahrscheinlichkeit $P\{m\}$ genauso so groß wie die bedingte Wahrscheinlichkeit $P\{m|c\}$ ist, wobei $c$ die Chiffre der Nachricht $m$ ist.*

Für den Kryptoanalytiker ändert sich also nichts an seinem Wissen über die Quelle und die Nachricht, wenn er die Chiffre der Nachricht erfährt.

Wenn ein kryptographisches System perfekt ist, dann muß es wenigstens genauso viele Schlüssel wie Nachrichten haben. Erfährt der unberufene Entschlüsseler die Chiffre $c$ und sind vorher wenigstens zwei verschiedene Nachrichten $m_1$ und $m_2$ möglich gewesen, dann müssen sie auch nach Kenntnis von $c$ noch mögliche Originaltexte sein. Das heißt, es gibt zwei Schlüssel $k_1$ und $k_2$, so daß $E(k_1, m_1) = c = E(k_2, m_2)$ ist. Dann müssen auch beide Schlüssel verschieden sein.

Andererseits kann man mit genügend vielen Schlüsseln auch ein sicheres System aufbauen.

Wir wählen uns ein lineares Kryptosystem der Blocklänge $k$ über einem Alphabet $\mathcal{A}$, als Nachrichten $k$-Folgen und als Schlüssel die Menge $\mathcal{A}^k$. Wir nehmen wieder an, daß die Elemente aus $\mathcal{A}^k$ einen $\mathcal{A}$-Vektorraum bilden, sie man also addieren und subtrahieren kann. Wählt man alle Schlüssel gleichwahrscheinlich aus, dann sind auch die alle Chiffren einer Nachricht gleichwahrscheinlich, mehr noch, es sind alle Elemente aus $\mathcal{A}^k$ als Chiffren möglich. Die Kenntnis der Chiffre ändert nichts an der Wahrscheinlichkeit der Nachricht $m$.

Warum ist hiermit der Kurs über Kryptologie noch nicht beendet? Es gibt ein sicheres System und der unberufene Entschlüsseler hat keine Chance. Die Nachteile dieser Chiffriertechnik liegen auf der Hand. Erstens benötigt man gleichwahrscheinliche und sehr große Schlüssel, die genauso lang sind, wie die Nachricht selbst. Diese Schlüssel dürfen nicht bekannt werden, man muß also den Schlüsselaustausch auf sicheren Kanälen durchführen. Meist kann man dann auch die Nachrichten selbst austauschen. Jeder Schlüssel kann nur einmal verwendet werden (Wegwerfschlüssel ohne grünen Punkt) und schließlich lassen sich mit diesem System Nachrichten nicht authentisieren, ein Problem, auf das wir später noch einmal zurückkommen.

## 4.3 Pseudozufallszahlen – Lineare Schieberegister

Wir haben im vorigen Abschnitt gesehen, daß eine echte Sicherheit nur durch sehr lange Schlüssel erreicht werden kann. Das Ziel besteht nun darin, aus einer relativ kurzen Beschreibung eine unendlich lange Folge zu generieren, die als Schlüsselfolge für unsere Flußchiffre dienen kann. Die so erzeugte Folge sollte möglichst nahe einer Zufallsfolge sein. Daß sie irgendwann einmal periodisch wird, stört nicht, wenn die Periode groß genug ist.

Wenn man aber schon von einer nicht zufällig erzeugten Folge ausgeht, dann muß man gewisse Forderungen an die Folge stellen. So sollte man verhindern, daß es sehr lange eintönige Stücke gibt, die würden nämlich die Chiffre stückweise in eine Cäsar-Chiffre verwandeln, mit katastrophalen Folgen für die kryptographische Sicherheit.

Im folgenden gehen wir davon aus, daß unser Alphabet der endliche Körper mit zwei Elementen $\mathbb{Z}_2$ ist.

Auf S. W. GOLOMB gehen die folgenden Pseudozufallskriterien zurück, die wir für die Schlüsselfolge fordern wollen:

**G1** In einer Periode unterscheiden sich die Anzahl der Nullen $0$ von der Anzahl der Einsen $1$ höchstens um eins.

**G2** Der Anteil der eintönigen Folgen (ausschließlich 0 oder ausschließlich 1) der Länge $k$ an der Menge aller eintönigen Folgen beträgt $2^{-k}$, wobei jeweils die Hälfte 0-tönig und 1-tönig sind.

**G3** Verschiebt man die Folge einer Periode zyklisch um $r$ Elemente, dann bezeichnet man mit $A(r)$ die Anzahl aller Übereinstimmungen ($s_i = s_{i+r}$) und mit $D(r)$ die Anzahl der unterschiedlichen Elemente an gleichen Stellen. Die Differenz $A(r) - D(r)$ muß dann für alle $r \neq 0$ gleich sein.

**Beispiel.** Man betrachte die Folge

$$1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0.$$

Sie hat die Länge $15$, enthält $7$-mal die Null, $8$-mal die Eins, eintönige Folgen sind, in der Reihenfolge ihres Auftretens:

$$1\,000\;1111\;0\;1\;0\;11\,00.$$

Von diesen $8$ Folgen sind vier von der Länge $1$, zwei der Länge $2$ und je eine der Länge $3$ und $4$. Die Bedingungen G1-3 sind erfüllt.

Wir werden nun ein Verfahren der Erzeugung von Pseudozufallsfolgen angeben, für das wir diese drei Eigenschaften nachweisen.

Es sei $(s_i)$ eine Folge von Bits.

**Definition 9** *Wir sagen, daß die Zufallsfolge $(s_i)$ durch ein lineares rückgekoppeltes Schieberegister gebildet wird, wenn*

$$s_{k+n} = f(s_k, \ldots, s_{k+n-1})$$

*für alle $k$ ist, wobei $f$ eine lineare boolesche Funktion ist*

$$f(s_0, s_1, \ldots, s_{n-1}) = s_0 c_0 \oplus \ldots \oplus s_{n-1} c_{n-1}.$$

Eine Folge $(s_i)$ ist periodisch, wenn es eine Zahl $r$ gibt, so daß $s_{n+r} = s_n$ ist. Wir nehmen an, daß $r$ die kleinste positive Zahl mit dieser Eigenschaft ist.

Wir betrachten nun einmal die Folgen $(s_i)$ als Folgen von Polynomen $(s_N(x))$, mit

$$s_N(x) = s_0 + s_1 x + \ldots + s_N x^N.$$

Wenn wir diese Polynome mit dem Polynom $f^\star(x) = 1 - c_{n-1}x - \ldots c_0 x^n$ multiplizieren, so erhalten wir ein Polynom, dessen Grad bis auf einen Bestandteil hinter $N$ kleiner als $n$ ist. Diesen kleinen Bestandteil bezeichnen wir mit $g(x)$.

Ist die Folge periodisch, mit der Periode $r$, dann ist $(x^r - 1)s_N(x) = h(x)$, ein Polynom, dessen Grad kleiner als $r$ ist.

Aus den beiden Zerlegungen folgt, daß unabhängig von $N$ daß

$$f^\star(x)h(x) = g(x)(x^r - 1)$$

gelten muß.

Andererseits folgt aus jeder solchen Zerlegung, daß die Periode der durch $f^\star$ erzeugten Pseudozufallsfolge ein Teiler von $r$ ist. Aus der Zerlegung ergibt sich nämlich die Periodizität der Folge $s_i$. Eine Folge, die durch Schiebregister der Größe $n$ erzeugt wird, hat dann maximale Periodenlänge, wenn $f^\star$ ein unzerlegbares Polynom ist.

Wie man aus der Algebra der endlichen Körper weiß, ist ein solches Polynom aber erzeugendes Polynom eines endlichen Körpers und wird durch das lineare Schieberegister an der Ausgabe jede $0-1$-Folge der Länge $n$ ausgegeben, wenn mit dem erzeugenden Element des Körpers begonnen wird.

An der Ausgabe wird aber jedes Körperelement innerhalb der Periode nur einmal ausgegeben, ausgenommen das Nullpolynom. Deshalb wird $1 * * \ldots *$ genau $2^{n-1}$ mal ausgegeben, die Folge $0 * * \ldots *$ genau $2^{n-1} - 1$ mal. An Stelle des $*$ kann jedes beliebige Bit stehen. Damit ist die Gültigkeit der ersten Golombschen Forderung gezeigt. Die beiden anderen Forderungen erhält man, wenn man nun die Blöcke

$$011\ldots10 * * \ldots *$$

zählt. Dabei sollen die Einsen eine eintönige $k$-Folge bilden. Da alle Elemente auftreten, gibt es genau $2^{n-k-2}$ eintönige Folgen aus Einsen und entsprechend bei der Betrachtung von

$$100\ldots01 * * \ldots *$$

sieht man die $2^{n-k-2}$ eintönige Folgen aus Nullen. Zählen wir jetzt alle eintönigen Folgen, so erhalten wir

$$2 \cdot 2^{n-3} + \ldots + 2 \cdot 1 + 1 + 1 = 2^{n-1}.$$

Die letzte beiden Folgen sind die eintönige $(n-1)$-Folge $10\ldots0$ und die eintönige $n$-Folge $11\ldots1$. Von den $2^{n-1}$ eintönigen Folgen sind $2 \cdot 2^{n-k-2}$ eintönige $k$-Folgen. Damit ist auch das zweite Golombsche Kriterium erfüllt.

Für das dritte Kriterium beobachten wir, daß sowohl die Folge $s_i$ als auch die Folge $s_{i+r}$ die Rekursionsgleichung des Schieberegisters erfüllen. Damit ist auch die Summe $s_i + s_{i+r}$ eine Folge, die der Gleichung genügt und die Golombschen Kriterien erfüllt. Die Anzahl der Nullen ist aber in dieser Folge die Anzahl der Übereinstimmungen der beiden Folge, die Anzahl der Einsen die Anzahl der unterschiedlichen Werte. Der Autokorrelationskoeffizient $(A(r) - D(r))/(2^n - 1)$ ist damit konstant und gleich $-(2^n - 1)$.

## 4.4 Pseudozufallszahlen – Nichtlineare Verfahren

Nichtlineare Verfahren haben den Nachteil, daß so gut wie nichts über sie bewiesen werden kann. Ein Vertreter dieses Verfahrens ist der uns schon bekannte DES-Algorithmus. Startet man mit einem beliebigen Startwert im OFB-Modus so erhält man eine Pseudozufallsfolge. Oder man wähle sich einen Startwert $s_0$, verschlüssele ihn und erhöhe dann $s_0$ um eins. Auch so erhält man eine Folge. Diese hat eine Periode, wenn man alle $2^{64}$ Blöcke verwendet hat. Das ist offensichtlich. Nicht klar ist dagegen, ob es eine kleinere Periode gibt. Niemand wird das ernsthaft glauben, die nicht-linearen *S-boxes* werden das verhindern. Einen Beweisansatz hat man so lange nicht, wie die Konstruktionsprinzipien der Permutationsblöcke unklar sind.

Andere Beispiele für nichtlineare Verfahren entstehen durch die erzeugenden Funktionen

$$s_n := 997 * s_{n-1} - \boldsymbol{INT}(997 * s_{n-1}) \qquad \text{Hewlett Packard HP-25}$$

oder

$$s_n := \boldsymbol{FRAC}(\boldsymbol{FRAC}(100\,000 * s_{n-1}) + 3 * s_{n-1})$$

oder schließlich

```
var x, y, z: integer;
function rnd: real;
begin x:= 170*(x mod 178) - 63*(x div 178);
      if x<0 then x:=x+30323;
      y:= 171*(y mod 177) -  2*(y div 177);
      if y<0 then y:=y+30269;
      z:= 172*(z mod 176) - 35*(z div 176);
      if z<0 then z:=z+30307;
      rnd:= frac(x/30323.0 + y/30269.0 + z/30307.0);
end;
```

Die erste Folge hat für die Startzahl $0.5284163$ die Periode $5 \cdot 10^5$, die zweite Folge die Periode $5 \cdot 10^7$ und die dritte hat Startwerte, die zu einer Periode von $6 \cdot 10^{12}$ führen können. Ein Nachweis, daß diese Folgen irgendeine der Golombschen Bedingungen erfüllen, gibt es nicht. Ihre Sicherheit ist damit nicht unumstritten.

# Kapitel 5

# Systeme mit öffentlichen Schlüsseln

Bei allen bisher behandelten Kryptosystemen war es unumgänglich, daß sowohl Sender als auch Empfänger über die Ver- und Entschlüsselungsfunktion verfügten. Aus der Kenntnis der ersteren konnte die letztere abgeleitet werden. Damit haben wir das Problem, daß sich Sender und Empfänger vor einer Kommunikationsaufnahme kennen und eventuell abgesprochen haben müssen. Es wäre aber wünschenswert, wenn jeder an eine beliebige Person eine Nachricht schicken und sich sicher sein könnte, daß auch wirklich nur dieser die Nachricht lesen kann.

Diese Aufgabe erfüllen Kryptosysteme mit öffentlichen Schlüsseln. Bei ihnen werden die Verschlüsselungsfunktion $E$ und der Schlüssel $k'$ veröffentlicht. Die Entschlüsselungsfunktion $D$ und der Gegenschlüssel $k''$ bleiben jedoch geheim.

## 5.1 Grundlagen

Einen möglichen Ansatz für solche Systeme bilden die Einwegfunktionen.

**Definition 10** *Eine Funktion $f : X \to Y$ heißt* **Einwegfunktion**, *wenn $y = f(x)$ für jedes $x \in X$ leicht zu berechnen ist, und es für die meisten $y \in Y$ schwer ist, ein $x$ zu bestimmen, so daß $x = f^{-1}(y)$ ist.*

Anzumerken bleibt jedoch, daß „schwer“ bzw. „leicht“ den Aufwand zur Berechnung der Funktionen beschreibt. Ein mögliches Beispiel für eine solche Einwegfunktion ist die Multiplikation von Zahlen. Das kann selbst nach der Schulmethode in quadratischer Zeit geschafft werden, die inverse Operation jedoch, die Zerlegung einer Zahl in Faktoren, ist bedeutend schwieriger. Bisher wurde ein Algorithmus gefunden, der zwar in Polynomialzeit beweist, daß n zerlegbar ist, jedoch nicht die Primfaktoren liefert[1] Ein weiteres Beispiel ist der login-Prozess. Dabei wird das Paßwort als Eingabe einer Funktion aufgefaßt und der berechnete Wert mit dem abgespeicherten Wert verglichen.

[1] Dieser Algorithmus geht von einer noch unbewiesenen zahlentheoretischen Aussage, der verallgemeinerten RIEMANNschen Hypothese aus. Im übrigen sind aber die Konstanten aber so schlecht, daß man mit subexponentiellen Algorithmen bei hundertstelligen Zahlen noch schneller ist. Entscheidend ist vielmehr, daß man für das Finden eines Faktors immer noch unerträglich lange braucht.

Ob es überhaupt Einwegfunktionen gibt, ist genauso schwer wie die Frage, ob **P** = **NP** ist. Im übrigen sind solche Funktionen für einen Nachrichtenaustausch nicht geeignet, der *berufene* Entschlüsseler hat mit dem Entschlüsseln ja die gleichen Probleme.

**Definition 11** *Eine Funktion $f : X \to Y$ heißt* **Falltürfunktion** *(oder besser Einweg-Falltürfunktion), wenn gilt :*

- *$f(x)$ ist leicht mit einem Algorithmus $E$ zu berechnen und*
- *$f^{-1}(y)$ ist leicht mit einem Algorithmus $D$ zu berechnen.*

*Die Bestimmung von $D$ aus $E$ soll jedoch schwer sein.*

Wir setzen hier Funktion und Algorithmus gleich, oder anders ausgedrückt, wir betrachten nur solche Funktionen, für die es auch einen effektiven Algorithmus gibt[2].

Mit Hilfe dieser Funktionen lassen sich nun sehr einfach Kryptosysteme mit öffentlichen Schlüsseln aufbauen.

Sei wie bisher $E: M \times K \to C$ die Verschlüsselungsfunktion, $D: C \times K \to M$ die Entschlüsselungsfunktion, und $D \circ (E(m,k),k) = m$ die identische Abbildung. Zusätzlich soll man jetzt aus der Kenntnis des Algorithmus $E(\cdot,k)$ nichts für den Algorithmus $D(\cdot,k)$ gewinnen können.

Weil nicht der Schlüssel $k$ und die Verschlüsselungsfunktion $E$, sondern die Funktion $D(\cdot,k)$ verborgen bleiben, nennt man diese Systeme auch **Public-Key-Systeme** oder **Systeme mit öffentlichen Schlüsseln**.

Besonders gesucht sind solche Systeme, die bei Bedarf (neuere Algorithmen, schnellere Rechentechnik) leicht vergrößert werden können, um gleiche oder größere Sicherheit gegen Angriffe zu bieten.

Grundsätzlich sind aber diese Systeme nach unserem bisherigen Verständnis unsicher. Sie können bei bekannter Chiffre gebrochen werden. Man verschlüsselt der Reihe nach alle möglichen Texte, bis deren Chiffre mit der zu brechenden übereinstimmt. Dann hat man die Nachricht gefunden.

Wichtige Vorteile sind aber das Wegfallen des geheimen Schlüsselaustausches und das Schlüsselmanagment bei sehr vielen Partnern. Für 100 Personen, die untereinander Daten austauschen wollen, bräuchte man 4950 Schlüssel in Secret-Key-Systemen, da jede Person mit jedem der 99 Partner einen speziellen geheimen Schlüssel vereinbaren muß. Dem gegenüber stehen 100 Schlüssel bei Public-Key-Systemen, wo jede Person nur einen öffentlichen Schlüssel benötigt. Hierbei können die Schlüssel (oder die Verschlüsselungsfunktionen) eines Teilnehmers in der Art eines Telefonverzeichnisses jedem anderen zugänglich gemacht werden.

Weitere Probleme, die bei der Benutzung solcher Systeme auftauchen, sind Autorisierung, Authentisierung und Nachrichtenbescheinigung. Darauf werden wir aber in späteren Kapiteln noch näher eingehen.

## 5.2 Der RSA-Algorithmus

Der RSA-Algorithmus basiert auf der Idee, daß Multiplikation einfach, aber Faktorisieren schwer ist. Er wurde 1978 von den Kryptologen RONALD RIVEST, ADI SHAMIR und LEONARD

[2] Manche meinen sogar, daß nur diese sich Funktionen nennen dürften, aber dann wäre $\pi_2(x)$, die Anzahl aller Primzahlzwillinge kleiner als $x$, keine Funktion, die Anzahl $\pi(x)$ der Primzahlen jedoch wäre eine.

ADLEMAN entwickelt und nach Ihnen benannt. Das hier angewendete Prinzip gilt zur Zeit als die beste Methode, was sowohl die Sicherheit als auch die Erweiterbarkeit betrifft. Sein entscheidender Mangel ist der fehlende Beweis seiner kryptographischen Sicherheit. Ist aber nicht zehnjähriger Widerstand gegen die vereinten Kryptoanalytiker nicht schon Beweis genug?

Benötigt werden zwei große Primzahlen $p$ und $q$. Daraus wird eine Zahl $n = p \cdot q$ berechnet. Zusätzlich bestimmen wir die Eulersche $\varphi$-Funktion, die die Anzahl der zu $n$ teilerfremden Zahlen ($< n$) angibt. Für Primzahlen $p$ gilt $\varphi(p) = (p-1)$ und entsprechend

$$\varphi(n) = (p-1)(q-1).$$

Wir wählen ein beliebiges $k$ mit ggT$(\varphi(n), k) = 1$ und berechnen $l$, so daß gilt

$$k \cdot l \equiv 1 \bmod \varphi(n).$$

Daraus ergeben sich die Kodierungsfunktion

$$E(m) \equiv m^k \bmod n$$

und die Dekodierungsfunktion

$$D(c) \equiv c^l \bmod n.$$

Dabei ist $(k, n)$ der öffentliche und $l$ der geheime Schlüssel.

Sind alle Primteiler von $n$ bekannt, so kann $\varphi(n)$ leicht bestimmt werden. Für die Berechnung ohne Primteiler ist (noch?) kein effektiver Algorithmus bekannt. Kennt man allerdings $\varphi(n)$ und weiß, daß $n$ das Produkt von zwei Primzahlen ist, so erhält man leicht $p$ und $q$. Es gilt demnach

$$\begin{aligned} \varphi(n) &= (p-1)(q-1) \\ &= pq - p - q + 1 \\ &= n - (p+q) + 1. \end{aligned}$$

Folglich sind $p$ und $q$ nach dem Satz von Vieta die Lösungen der Gleichung

$$x^2 - (n - \varphi(n) + 1) \cdot x + n = 0.$$

Kennt man $\varphi(n)$, so kann das geheime Inverse $l$ leicht aus $k$ bestimmt werden. Die Umkehrung, also ob die Kenntniss von $l$ und $k$ für die Berechnnug von $\varphi(n)$ reicht, ist noch nicht geklärt. Es gibt bisher keinen Algorithmus, der aus einer Menge von Klartexten und ihren Chiffren die Werte $k$ und $l$ berechnet bzw. aus einer weiteren Chiffre die zugehörige Nachricht ermittelt.

Es gilt also, wer $n$ faktorisieren kann, ist in der Lage, die Chiffre zu brechen. Die für die kryptographische Sicherheit entscheidende Frage, ob der, der die Chiffre brechen kann, auch schnell faktorisieren kann, ist unklar. Daß das Faktorisieren aller Wahrscheinlichkeit nach ein für immer schwieriges Problem bleibt, sollte uns nicht in trügerische Sicherheit wiegen. Das Brechen der RSA-Chiffre kann leicht sein, das heißt in Polynomialzeit lösbar sein, und trotzdem bleibt das Faktorisieren noch schwer.

Es ergibt sich natürlich die Frage, wie man große Primzahlen generiert. Der „kleine Fermat“ besagt: Ist $n$ eine Primzahl, so gilt

$$a^{n-1} \equiv 1 \bmod n, \quad \text{für alle } a \text{ mit } \text{ggT}(a, n) = 1.$$

Die Umkehrung ist jedoch falsch[3].

Wir müssen also einen anderen Weg suchen. Eine Möglichkeit wäre der Rabin-Test. Aus dem „kleinen Fermat“ folgt:

$$a^{\frac{n-1}{2}} \equiv +1 \bmod n \quad \text{oder} \quad a^{\frac{n-1}{2}} \equiv -1 \bmod n.$$

Im ersten Fall können wir sukzessive fortfahren:

$$a^{\frac{n-1}{4}} \equiv +1 \bmod n \quad \text{oder} \quad a^{\frac{n-1}{4}} \equiv -1 \bmod n \quad \text{usw. ...}$$

Ist $n$ eine Primzahl und sind $n-1, \frac{n-1}{2}, \ldots$ ganze Zahlen, so ist $a^{n-1}, a^{\frac{n-1}{2}}, \ldots$ kongruent zu der Folge $+1, +1, \ldots, +1, -1, \ldots$. Im anderen Fall gibt es einen „Zeugen“ für die Zerlegbarkeit, d.h. es existiert ein $a$, so daß sich die obige Folge nicht ergibt. Das Verhältnis von Nichtzeugen zu Zeugen ist jedoch wenigstens 1:3 für eine Nichtprimzahl. Wenn wir also diesen Test $r$-mal mit beliebig gewähltem $a$ wiederholen, so sinkt die Wahrscheinlichkeit unter $\frac{1}{4^r}$, daß wir stets nur Nichtzeugen erwischen und folglich eine zerlegbare Zahl für eine Primzahl halten.

Als Ergänzung kann noch der Lehmer-Test (auch Solovay-Strassen-Test genannt) benutzt werden. Dabei wird ausgenutzt, daß das Legendresymbol

$$\left(\frac{a}{n}\right) = \begin{cases} +1 & \text{falls a quadratischer Rest ist,} \\ -1 & \text{falls a kein quadratischer Rest ist.} \end{cases}$$

Man wählt sich ein $a$ mit ggT$(a, n) = 1$ und berechnet das Legendresymbol. Dann ist $n$ keine Primzahl, wenn gilt

$$\left(\frac{a}{n}\right) \not\equiv a^{\frac{n-1}{2}} \bmod n.$$

Für die kryptographische Sicherheit ist noch darauf zu achten, daß $p$ und $q$ nicht allzu dicht beeinanderliegen. Es kann sonst leicht eine Zerlegung gefunden werden. Wir berechnen $s = \lceil \sqrt{n} + 1 \rceil$ und prüfen, ob $s^2 - n$ eine Quadratzahl ist. Wenn nicht, erhöhen wir $s$ um eins und prüfen erneut. Andernfalls gilt

$$n = s^2 - t^2 = (s+t)(s-t),$$

womit wir eine Zerlegung für $n$ gefunden haben.

Es gibt bereits mehrere Implementierungen des RSA-Algorithmus' mit 512 Bit langem $n$. Einige RSA-Chips arbeiten sogar mit einer Länge von 660 Bit und einer Kodierungsgeschwindigkeit von 29 kBit/sec. Trotzdem sind die RSA-Chips den schnellen DES-Chips (20 MBit/sec) deutlich unterlegen. Ihr Vorteil ist die vermutlich grössere Sicherheit.

**Übungsaufgabe 7** *Überlegen Sie sich eine Kombination von RSA und DES, die schnell und sicher ist !*

## 5.3 Diskreter Logarithmus

Ein ähnlich schwieriges Problem, wie das Faktorisieren einer Zahl, ist das Problem des diskreten Logarithmus. Dabei gehen wir davon aus, daß man über einem gegebenen endlichen Körper $K$ (z.B. $K = \mathbb{Z}_p$ mit Primzahl $p$) Exponential- und Logarithmusfunktionen

[3] Gegenbeispiele sind die sogenannten Carmichaelzahlen. Die kleinste ist $561 = 3 \cdot 11 \cdot 17$, eine andere $32\,150\,317\,751 = 151 \cdot 751 \cdot 28351$, die übrigens die kleinste zerlegbare Zahl ist, die dem Rabin-Test mit 2, 3, 5 und 7 widersteht.

definieren kann, die zueinander invers sind. Die Berechnung der Potenzen einer gegebenen Zahl $g \in K \setminus \{0\}$ ist leicht. So ist $5^4 \equiv 13 \bmod 17$, aber das Bestimmen des Exponenten $x$ mit $5^x \equiv 20 \bmod 23$ ist außerordentlich schwer. Bisher ist nur ein Algorithmus verfügbar, falls sich $p-1$ in relativ kleine Primfaktoren zerlegern läßt.

### 5.3.1 Der Diffie-Hellman-Schlüsselaustausch

In der ersten Arbeit über Public-Key-Systeme wurde 1976 von W. DIFFIE und M. E. HELLMAN ein Verfahren vorgeschlagen, das es ermöglicht, über einen öffentlichen Kanal geheime Schlüssel auszutauschen. Dabei werden der Körper $K$, d.h. die Primzahl $p$ bei $K = \mathbb{Z}_p$, und eine beliebige Zahl $s < p$ veröffentlicht[4].

Seien Anne und Bob zwei Personen[5], die untereinander Nachrichten austauschen wollen und dazu einen geheimen Schlüssel benötigen.

Unsere zwei Partner Anne und Bob wählen sich je eine Zahl $a$ und $b$, die sie geheim halten. Anne berechnet $\alpha \equiv s^a \bmod p$ und Bob berechnet $\beta \equiv s^b \bmod p$. Danach schickt jeder sein Ergebnis an den Partner. Der berechnet nun $k \equiv \beta^a \bmod p$ bzw. $k \equiv \alpha^b \bmod p$. Wenn alles korrekt erfolgte, sind beide $k$ gleich und können jetzt als Schlüssel benutzt werden. Ein Eindringling sieht zwar $s^a$ und $s^b$, kann aber $a$ und $b$ und somit $k = s^{a \cdot b}$ nicht bestimmen.

### 5.3.2 Das Massey-Omura-Kryptosystem

Ein besonders originelles System, das ohne Schlüsselaustausch arbeitet, wurde auf der ersten EUROCRYPT-Tagung 1983 in Udine vorgestellt. Die Idee dazu stammt ursprünglich von ADI SHAMIR und wurde von JIM MASSEY und JIM OMURA für die diskreten Logarithmen umgesetzt.

Seien wie oben Anne und Bob unsere zwei Personen und $p$ eine öffentliche Primzahl. Privat wählt sich nun Anne ein $a$ und das arithmetische Inverse $a'$ modulo $p$. Analog bestimmt Bob seine geheimen Zahlen $b$ und $b'$. Aufgrund des Satzes von Euler gilt für natürliche Zahlen $n < p$ die Gleichung

$$n^{a \cdot a'} \bmod p \equiv n^{b \cdot b'} \bmod p \equiv n\,.$$

Sei $m$ nun die Nachricht von Bob für Anne.

- Zuerst verschlüsselt Bob $m$ mit $b$ und schickt das Ergebnis $c_1 \equiv m^b \bmod p$ an Anne.
- Sie verschlüsselt dann $c_1$ mit ihrem $a$ und schickt das berechnete $c_2 \equiv {c_1}^a \bmod p$ an Bob zurück.
- Bob entschlüsselt nun $c_2$ mit $b'$ und erhält

  $$c_3 \equiv {c_2}^{b'} \bmod p \equiv {c_1}^{ab'} \bmod p \equiv m^{bab'} \bmod p \equiv m^a \bmod p.$$

  Dieses $c_3$ geht wieder an Anne.
- Sie erhält damit ohne Mühe $m \equiv {c_3}^{a'} \bmod p \equiv m^{aa'} \bmod p$.

[4] Günstig ist, wenn $s$ ein erzeugendes Element der zyklischen Gruppe $\mathbb{Z}_p$ ist, da $s^x$ für $x = 1, \ldots, p-1$ dann die ganze Gruppe „überstreicht".

[5] Trotz der Tatsache, daß 75 % der Kommunikation zwischen Frauen stattfindet, haben wir uns aus Gründen der Gleichberechtigung für einen gemischten Dialog entschieden.

Der Nachteil dieses System liegt nun offensichtlich darin, daß eine Nachricht dreimal ausgetauscht werden muß, bis sie ihren Empfänger erreicht. Außerdem werden wir hier besonders mit dem Problem der Identifizierung konfrontiert. Klinkt sich nämlich Clara in diesen Datenaustausch ein, fängt sie Annes Nachrichten ab, und ersetzt sie durch ihre, dann kann sie alles mitlesen, ohne daß Anne und Bob davon erfahren.

**Übungsaufgabe 8** *Zeigen Sie den beiden die Schwachstelle ihres Systems !*

### 5.3.3 Das ElGamal-Kryptosystem

Auch das von TAHER ELGAMAL entwickelte Kryptosystem basiert auf der Verschlüsselung mit Hilfe diskreter Logarithmen. Seien wieder eine Primzahl $p$ und ein $s$ öffentlich vereinbart. Jeder Teilnehmer, in unserem Beispiel Bob, hat einen geheimen Schlüssel $b$ und veröffentlicht $s^b$. Will jetzt Anne die Nachricht $m$ an Bob schicken, wählt sie eine beliebige Zahl $k$ und bestimmt $s^k$ sowie $(s^b)^k$ mit Bob's öffentlichen Schlüssel $s^b$. An Bob übermittelt sie das Paar

$$(s^k,\ m \cdot s^{bk}).$$

Bob kann nun aus dem erhaltenen Päckchen $(n_1, n_2)$ mit seinem geheimen Schlüssel $b$ leicht die Nachricht

$$m \equiv n_2 \cdot n_1^{-b} \bmod p \equiv m \cdot s^{bk} \cdot s^{-bk} \bmod p$$

entschlüsseln.

Als Verallgemeinerung kann eine beliebige Verschlüsselungsfunktion $E$ veröffentlicht und das Paar $(s^k,\ E(m, s^{bk}))$ verschickt werden. Bob braucht dabei nur die zu $E$ inverse Entschlüsselungsfunktion mit dem Schlüssel $(s^k)^b$ zu berechnen und erhält die Nachricht.

## 5.4 Das Knapsack-Problem – das Merkle-Hellman-Kryptosystem

Ein NP-vollständiges Problem ist das Knapsackproblem. Dabei besteht die Aufgabe darin, ein Gefäß, zum Beispiel einen Rucksack (*knapsack*), möglichst optimal mit Gegenständen unterschiedlichen Volumens (oder Gewichtes) zu packen.

Hat man zum Beispiel fünf Gegenstände der Gewichte $2, 3, 5, 15, 31$ und einen Aufzug mit maximaler Last von 41, so stellt man am besten die Gegenstände mit den Gewichten 31, 7 und 2 in den Aufzug, um ihn möglichst gut auszulasten.

Anscheinend ist das Problem nicht allzu kompliziert. So wird jedoch jeder, der in seinem Leben schon sehr viel Zeit mit Koffer packen zugebracht hat, bestätigen, daß dieser Eindruck gewaltig täuscht. Deshalb hier noch einige Vorbetrachtungen.

**Definition 12** *Das* **Zahlenrucksackproblem** *ist ein Optimierungsproblem folgender Art:*

*Zu gegeben $n$ Zahlen $a_1, \ldots, a_n$ und einem Gesamtgewicht $m$ wird eine Belegung $k_i \in \{0, 1\}$, für $i = 1, \ldots, n$ , gesucht, so daß gilt:*

$$\sum_{i=0}^{n} a_i \cdot k_i = m.$$

**Definition 13** *Eine Zahlenfolge $a_1, \ldots, a_k, \ldots$ heißt* **schnell wachsend**, *wenn für alle $k$ gilt*

$$a_{k+1} > \sum_{i=1}^{k} a_i.$$

Für schnellwachsende Zahlenfolgen ist das Rucksackproblem leicht zu lösen. Die größte passende Zahl wird zuerst „verstaut", und der nun „geschrumpfte" Rucksack wird mit den verbliebenen Zahlen weiter gepackt.

Ist dagegen die Zahlenfolge für den Rucksack nicht schnell wachsend, ist das Zahlenrucksackproblem nicht nur schwer, sondern vielleicht (man weiß es noch nicht) sogar noch schwerer als die Primzahlenfaktorisierung.

Damit haben wir einen weiteren Kandidaten für eine Verschlüsselungsmethode. So wurde 1978 von R. C. MERKLE und M. E. HELLMAN ein Public-Key-System vorgestellt, dessen Trick darin besteht, ein leichtes Rucksackproblem in ein schweres zu verwandeln und dieses als öffentlichen Schlüssel zu benutzen.

- Zuerst wählt sich Anne ein leichtes Rucksackproblem, zum Beispiel die Folge $(a_i) = 2, 3, 7, 15, 31$, eine Zahl $n > \sum a_i$, ein $k$ mit ggT$(n, k) = 1$ und bestimmt das arithmetische Inverse $l$ von $k$ modulo $n$. Seien $n = 61$ und $k = 17$, so erhält sie $l = 18$. Anne berechnet dann $b_i \equiv a_i \cdot k \bmod n$ und veröffentlicht die Folge $(b_i) = 34, 51, 58, 11, 39$. Diese Folge repräsentiert jetzt ein schwieriges Rucksackproblem.
- Möchte Bob nun die binäre Nachricht $m = (1, 1, 0, 1, 1)$ an Anne senden, ermittelt er $c = \sum m_i \cdot b_i$ und schickt $c = 34 + 51 + 11 + 39 = 135$.
- Anne kann, durch Kenntnis von $k$ und $l$, den schweren in einen leichten Rucksack umwandeln. Sie berechnet

  $$c \cdot l = \sum m_i \cdot a_i \cdot k \cdot l \equiv (\sum m_i \cdot a_i) \bmod n = \sum m_i \cdot a_i, \quad \text{da } n > \sum a_i.$$

  In unserem Beispiel erhält Anne das leichte Rucksackproblem $51 = \sum m_i \cdot a_i$ und somit die Nachricht $m = (1, 1, 0, 1, 1)$.

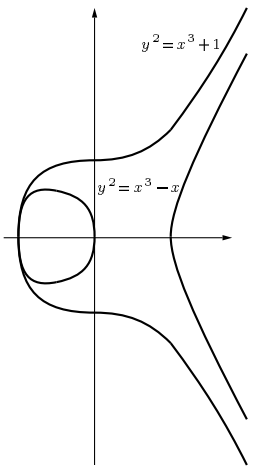
Zum Erschweren der Analyse kann man iterativ mit $n, n', \ldots$ und $k, k', \ldots$ mehrere Rucksackprobleme ineinander verschachteln.

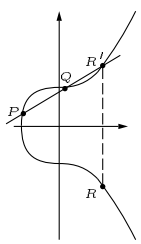
## 5.5 Elliptische Kurven

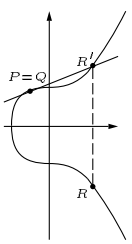
**Definition 14** *Eine Kurve wird elliptisch genannt, wenn sie folgender Gleichung genügt:*

$$y^2 = x^3 + ax + b.$$

Eine elliptische Kurve in Normalform wird durch zwei Punkte eindeutig definiert. Im weiteren gelte die Voraussetzung, daß die Kurve keine mehrfachen Punkte enthalte.



Betrachten wir $P = (x_1, y_1)$ ,$Q = (x_2, y_2)$ und $R = (x_3, y_3)$ als drei Punkte auf der Kurve. Eine Addition von Punkten mit $R = P + Q$ wird so definiert, daß $P$, $Q$ und $R'$ auf einer Geraden liegen, wobei $R'$, der an der $x$-Achse gespiegelte Punkt $R$ ist.

Die Koordinaten $(x_3, y_3)$ können durch folgende Formeln berechnet werden:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \qquad -y_3 = \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1) + y_1.$$

Ist $P = Q = (x, y)$, so haben natürlich diese Formeln keinen Sinn und es gilt

$$x_3 = \left(\frac{3x^2 + a}{2y}\right)^2 - 2x, \qquad y_3 = -y + \frac{3x^2 + a}{2y}(x - x_3).$$

In diesem Fall wird im Punkt $P = (x, y)$ eine Tangente angelegt. $R$ ist der an der $x$-Achse gespiegelte Schnittpunkt der Tangente mit der Kurve.

Ist $P = (x, y)$ und $Q = (x, -y)$ so wird $R = \mathbf{0}$ definiert.

Die Punkte einer elliptischen Kurve bilden eine Gruppe. Die Addition der Punkte ist kommutativ und assoziativ. Es existiert ein neutrales Element $\mathbf{0}$ und zu jedem beliebigen Element $P$ findet man ein inverses $-P$.

Man kann die Kurven über den rationalen Zahlen oder auch modulo einer Primzahl $p$ betrachten und erhält dann Punkte über $\boldsymbol{Q}$ oder $\mathbf{F}_p$.

Sei eine Logarithmusfunktion wie folgt definiert:

$$\text{Wenn gilt } n \cdot B = \underbrace{B + B + \cdots + B}_{n\text{-mal}} = P, \text{ so ist } n = \log_B P.$$

Die Bestimmung von $n$ aus $P$ und $B$ ist schwer.

Es lassen sich nun die Kryptosysteme mit diskreten Logarithmen nach Belieben auf elliptischen Kurven umstellen. Für den Diffie-Hellman-Schlüsselaustausch ergibt sich mit $B$ als öffentlichen Punkt einer Kurve $K$ folgendes Protokoll.

- Anne berechnet mit ihrem geheimen $a$ das Produkt $aB$ und Bob entsprechend $bB$ mit seinem $b$.
- Beide tauschen ihre Ergenisse aus, multiplizieren sie mit ihrem geheimen Schlüssel und erhalten gemeinsam einen neuen Punkt $P = abB$ auf der Kurve $K$, der im weiteren als geheimer Schlüssel verwendet werden kann.

Auch das ElGamal-Verfahren läßt sich mit elliptischen Kurven realisieren. Seien die Kurve $K$ und ein Punkt $B$ der Kurve öffentlich. Anne und Bob berechnen mit ihren geheimen Schlüsseln $a$ und $b$ die Punkte $aB$ und $bB$ und veröffentlichen sie.

- Um eine Nachricht an Bob zu schicken, wählt sich Anne ein zufälliges $k$ und sendet ihm das Tupel $(kB, m + kbB)$.
- Bob erhält zwei Punkte $(P_1, P_2)$ und berechnet sich die Nachricht $m = -(bP_1) + P_2$.

*Beispiel:* Wir betrachten die elliptische Kurve $K$ mit $y^2 + y = x^3 - x$ in $\mathbf{F}_{751}$, $B = (0, 0) \in K$ und das Alphabet $0, \ldots, 9, A, \ldots, Z$.

Die zu übermittelnde Nachricht sei $\boldsymbol{STOP} = (28, 29, 24, 25) = (m_1, m_2, m_3, m_4)$ und $bB = (201, 380)$ Bob's öffentlicher Punkt.

- Anne expandiert die Nachricht um einen vorher vereinbarten Expansionsfaktor und sucht die nächstgelegenden Punkte auf der Kurve[6].

$$\begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{pmatrix} \cdot 20 = \begin{pmatrix} 560 \\ 580 \\ 480 \\ 500 \end{pmatrix} \longrightarrow \begin{pmatrix} P_1(562, 576) \\ P_2(581, 395) \\ P_3(484, 214) \\ P_4(501, 220) \end{pmatrix}$$

- Anne generiert zufällig $k_1, \ldots, k_4 = 386, 209, 118, 589$ und berechnet $(k_i B, m_i + k_i bB)$.

$$c_1 = E(m_1) = [386(0,0), (562,576) + 386(201,380)] = [(676,182), (385,703)]$$

Analog ergeben sich

$$\begin{aligned} c_2 &= [(595,454), (212,625)], \\ c_3 &= [(261,87), (77,369)] \quad \text{und} \\ c_4 &= [(126,100), (66,589)]. \end{aligned}$$

- Bob entschlüsselt

$$\left\lfloor \frac{(-b(676,182) + (385,703))_{x\text{-Koordinate}}}{20} \right\rfloor = 28 = \boldsymbol{S}.$$

**Übungsaufgabe 9** *Finden Sie weitere Kandidaten für Falltürfunktionen und überprüfen Sie deren Tauglichkeit für Public-Key-Systeme! Verwenden Sie dabei Ihre Kenntnisse aus anderen Vorlesungen, wie „Berechnungstheorie“ und „Algebra“ !*
*Denken Sie an das Lösen quadratischer Gleichungen in Restklassenringen.*

[6] Ohne Expansion lassen sich meistens nicht genügend unterschiedliche Punkte finden. So sei in unserem Beispiel der Expansionsfaktor 20 vorher vereinbart worden.

# Kapitel 6

# Anwendungen und benachbarte Fragestellungen

Nicht immer geht es darum, den Inhalt einer Nachricht zu verbergen. Oft ist es wichtig, das Datum einer Nachricht (Aktienkauf), den Empfänger einer Nachricht (Überweisung), den Absender einer Nachricht (Bestellung) zweifelsfrei feststellen zu können und auch eine unberufene Verfälschung nicht zuzulassen. Eine unberufene Einmischung geschieht

- durch Verleugnen einer Nachricht,
- durch Behaupten des Empfangs einer Nachricht und
- durch Verändern (Verfälschen) einer Nachricht.

Besonders beliebt ist beim Verfälschen, nicht die Veränderung der Nachricht, sondern die des Absenders. Man versucht sich als andere Person zu maskieren und erwirkt in ihrem Namen eigene Vorteile, wie zusätzliche Rechenzeit. Um das zu verhindern, *authentisiert* man die Nachrichten, um sie danach jederzeit *authentifizieren* zu können[1]. Wir werden deshalb immer von der Authentisierung sprechen, auch wenn im deutschen Sprachgebrauch sehr oft Authentifizierung verwendet wird.

**Definition 15** *Ein Authentisierungssystem besteht aus einer Schlüsselmenge $K$, einer Nachrichtenmenge $M$, einer Etikettenmenge $T$ und einer Authentisierungsfunktion $A_k : M \to T$, die für jeden Schlüssel $k \in K$ jeder Nachricht ein Etikett zuordnet.*

Man kann wie bei den Kryptosystemen vorsehen, daß zu jedem Schlüssel eine eigene Nachrichtenmenge und eine eigene Etikettenmenge gegeben ist, das ist aber durch eine entsprechende Vereinigung auf den obigen Fall zurückführbar.

Für die Authentisierungsfunktion müssen wir fordern, daß sie für jeden Schlüssel $k$, effizient berechenbar ist. Das Authentisierungverfahren zwischen Alice und Bob läuft dann wie folgt:

- Alice und Bob vereinbaren einen geheimen gemeinsamen Schlüssel ($k$),
- will Alice eine Nachricht $m$ authentisieren, so berechnet sie $t = A_k(m)$ und sendet $m$ zusammen mit $t$,
- Bob prüft die empfangene Nachricht (authentifiziert sie), indem er das Etikett der empfangenen Nachricht berechnet und mit $t$ vergleicht.

[1] authentikós bedeutet im griechischen 'Urheber', 'Ausführer'; authentisieren bedeutet rechtgültig, glaubwürdig machen; authentifzieren die Echtheit bezeugen, beglaubigen, vgl. im englischen authentication vs. authentification

Für die Sicherheitsanforderungen an dieses Verfahren gibt es drei Stufen. So sollte der unberufene Entschlüsseler

1. aus einer Nachricht $m$ und dem Etikett $t$ der Nachricht nicht den Schlüssel $k$ bestimmen können, oder
2. aus gegebenen Nachrichten $m_i$ mit den entsprechenden Etiketten $t_i$ nicht das Etikett $t$ einer beliebigen Nachricht $m$ bestimmen können, oder als maximale Forderung,
3. sollte er nicht das Etikett irgendeiner Nachricht aus einer möglichen Teilmenge von Nachrichten bestimmen können.

Die Forderung, daß der unberufene Entschlüsseler zu keiner möglichen Nachricht das Etikett bestimmen kann, ist nicht durchzuhalten. Aber auch nicht notwendig, denn es gibt viele mögliche jedoch uninteressante Nachrichten, deren Schutz nicht erforderlich ist. Deshalb muß die schutzwürdige Teilmenge von Fall zu Fall konkret festgelegt werden, so sind es für eine unberufene Gutschrift auf dem Konto des Entschlüsselers alle positiven Nachrichten. Denn dem ist es nämlich „fast“ egal, ob sie 193856.73 DM oder 214534.26 DM beträgt.

Für erhöhte Sicherheitsanforderungen muß man auch hier vom Schlimmsten ausgehen, daß nämlich der unberufene Entschlüsseler die Nachrichten eine gewisse Zeit selbst wählen könnte.

Wie auch schon in den vorigen Kapiteln werden wir nicht genauer darauf eingehen, was es bedeuten soll, daß der unberufene Entschlüsseler das Etikett nicht bestimmen kann. Es soll darunter nicht ausschließlich die beweisbare Unmöglichkeit verstanden werden, sondern auch die praktische Unmöglichkeit, die durch die polynomiale Rechenfähigkeit des unberufenen Entschlüsselers vorgegeben ist.

## 6.1 Digitale Unterschriften

Digitale Unterschriften sind eine Möglichkeit des Authensierens von Dokumenten. Wir unterscheiden dabei zwischen universellen und nicht universellen Unterschriften. Eine universelle Unterschrift soll von jedermann nachprüfbar sein. Eine nicht universelle erfordert einen unabhängigen Richter, der von allen akzeptiert wird. Wir haben universelle Unterschriften bereits im Fall der Systeme mit öffentlichen Schlüsseln behandelt. Dabei ist es wichtig, daß für die Ver- und Entschlüsselungfunktionen nicht nur

$$D_k(E_k(m)) = m, \text{ sondern auch } E_k(D_k(c)) = c \text{ gilt.}$$

Wir wollen hier ein mögliches nicht universelles System vorstellen. Zu Beginn müssen (nach deutscher Rechtsauffassung) beide Partner schriftlich vereinbaren, daß sie digitale Unterschriften zulassen wollen. Unter Umständen muß das sogar notariell beglaubigt werden.

Alice und Bob vereinbaren eine Liste von Schlüsseln $k_1, k_2, \ldots, k_N$ die der Authentisierung der Nachrichten von Alice dienen sollen. Genauer gesagt, sie vereinbaren eine Liste von Etiketten einer Standardnachricht und behalten die zugehörigen Schlüssel für sich.

Um eine Nachricht unterschreiben zu können, wählt Alice sich $n$ noch unbenutzte Schlüssel und sendet eine Nachricht $m$ zusammen mit $n$ Etiketten $(m, t_{k_{i_1}}(m), \ldots, t_{k_{i_n}}(m))$. Danach wählt Bob zufällig $s$ Etiketten aus, und fordert Alice auf, ihm dafür die Schlüssel zu senden.

Alice sendet sie und Bob prüft, ob die entsprechenden Etiketten der Standardnachricht auf der vereinbarten Liste stehen und ob die ausgewählten Etiketten der unterschriebenen Nachricht übereinstimmen.

Ein Richter kann überprüfen, ob der Vertrag gültig ist. Er gibt Bob recht, wenn der $n$ Unterschriften präsentieren kann, Alice kann dem Richter die Mittel in die Hand geben, mit der geprüft werden kann, daß diese korrekt sind. Fällt diese Prüfung negativ aus, sind es etwa nur $s$ Etiketten, so gilt der Vertrag als nicht gültig. Sind es dagegen wenigstens $s+1$ korrekte Etiketten, so ist der Vertrag gültig. Das Risko, das Alice eingehen müßte, um einen Betrug zu begehen, also $s$ korrekte Etiketten und $n-s$ falsche Etiketten zu senden, ist sehr groß. Bob müßte gerade die richtigen Etiketten auswählen, und das ist so wahrscheinlich wie ein Lottogewinn (keine schlechte Alternative: gültiger Vertrag oder sechs Richtige im Lotto).

Bob könnte allerdings betrügen, wenn er genügend Schlüssel besitzen würde, könnte er sich selbst einen Vertrag zurecht machen, der angeblich von Alice unterschrieben ist. Um das zu verhindern, kann man vorsehen, daß Alice die benutzten Schlüssel mit den Verträgen beim Richter hinterlegt, oder aber Bob unterschreibt seinerseits einen Vertrag, in dem steht, daß die aufgeführten Schlüssel zu dem von Alice genannten Vertrag gehören. Dann kann Bob diese Schlüssel nicht an andere Veträge koppeln. Alice hätte dafür den Beweis.

Wenn die Schlüssel irgendwann einmal ausgehen sollten, dann unterschreibt man digital einfach einen Vertrag, in dem die Schlüssel für die nächste Zeit vereinbart werden.

## 6.2 Authentisierung von Personen – *das Kerberos-System*

Die Authentisierung von Personen ist besonders wichtig in Rechnernetzen. Hier besteht eine große Gefahr des Mitlesens, Maskierens und Erschleichens von Dienstleistungen. Am sichersten ist es, wenn man von Beginn an davon ausgeht, daß es unberufene Entschlüsseler gibt, die Zugang zu einem beliebigen Terminal haben. Diesen Zugang zu verhindern, ist zwar prinzipiell auch möglich, durch Kontrolle personeneigener Eigenschaften (Fingerabdruck, Augenhintergrund, Sprache) oder fälschungssicherer Dokumente, aber erstens ist deren Sicherheit auch nicht hundertprozentig und zweitens können sie gestohlen sein. Wir werden also die Personen nicht durch Besitz authentifizieren, sondern durch diesen Personen eigenes Wissen. Das sind die Paßwörter, deren Diebstahl und unberufene Weitergabe wir (unkorrekterweise) ausschließen. Durch eigene Auswahlkriterien, für die wir Beispiele an anderer Stelle behandeln werden, ist das auch möglich, wenngleich es auch nicht immer durchgesetzt wird.

Wir gehen von einem Netz mit unsicheren Terminals aus, setzen aber voraus, daß es ein Terminal $K$ gibt, dessen Sicherheit absolut ist, und deren Entscheidungen von allen akzeptiert werden. Dieses Terminal verfügt sicher über alle Nutzernamen und die von ihnen benutzten Schlüssel.

Anne setzt sich an das Terminal mit der Netzadresse $n$ und möchte mit Bob $B$, dem Mailserver $M$ und weiteren Nutzern in Verbindung treten. Dazu beginnt sie zunächst mit **login** und der Eingabe ihres Nutzernames $u$. Der wird an $K$ gesendet, der mit einer zufälligen Nachricht $m$ antwortet. Diese zufällige Nachricht wird in Annes Terminal mit ihrem Schlüssel verschlüsselt und an $K$ zurückgesandt. $K$ kann testen, ob die Nachricht wirklich von Anne verschlüsselt wurde, es hat den Zugang zu den Schlüsseln der Nutzer. Genauer gesagt, bekommt Anne von $K$ mit der zufälligen Nachricht $m$ gleichzeitig einen Sitzungsschlüssel $k$ und ein Bewilligungszertifikat $b$, mit dem sie sich dann für jede Verbindung ein Verbindungszertifikat besorgen kann. Da diese Nachricht mit Annes Gegenschlüssel

verschlüsselt wird, kann nur Anne sie lesen, denn nur sie hat (hoffentlich) ihren privaten Schlüssel.

Das Verbindungszertifikat ist zusammen mit einem Gültigkeitsetikett für jede Verbindung zu präsentieren. Also, für den Austausch mit Bob besorgt sie sich ein $B$-Zertifikat, für den mit $M$ ein $M$-Zertifikat. Bei jeder Anfrage an Bob, präsentiert Anne zugleich mit der Anfrage ihr $B$-Zertifikat und ein Gültigkeitsetikett für dieses Zertifikat. Stellen wir uns für einen Moment einen Geldautomat der X-Bank vor. Das Verbindungszertifikat ist die Kundenkarte der Bank, das Gültigkeitsetikett – die Geheimzahl, die man eintippt.

Unser Ziel ist es zu verhindern, daß andere unter Annes Namen mit Bob oder dem Mailer korrespondieren. Dazu gestalten wir die einzelnen Zertifikate und Etiketten folgendermaßen.

Das Bewilligungszertifikat für Anne besteht aus einer Folge

$$(k : u : n : l : t),$$

wobei $k$ ein Sitzungsschlüssel für Anne, $u$ der Nutzername „ANNE“, $n$ die Netzadresse ihres Terminals, $l$ die Gültigkeitsdauer des Zertifikats und $t$ die aktuelle Zeit ist. Diese Folge wird vom Kerberos-Server mit dem Schlüssel des Zertifikat-Servers verschlüsselt und ergibt das Bewilligungszertifikat $b$, das mit Annes Schlüssel verschlüsselt und an Anne gesandt wird.

Das Bewilligungszertifikat für die Verbindung mit Bob beantragt Anne beim Zertifikat-Server mit der Mitteilung

$$(u : n : b : s),$$

eigener Name, eigene Netzadresse, Bewilligungszertifikat, gewünschter Service $s$ und einem Gültigkeitsetikett

$$f_k(u : n : l : t),$$

das, mit dem Sitzungsschlüssel verschlüsselt, den eigenen Namen, die eigene Adresse, die Gültigkeitsdauer des Etiketts und die aktuelle Zeit enthält. Der Zertifikat-Server berechnet aus dem Bewilligungstikett den Sitzungsschlüssel, überprüft die Übereinstimmung der Namen und Adressen, die Gültigkeit und gibt das Verbindungszertifikat zurück. Das besteht aus:

$$(k_s : u : n : s : l : t),$$

Verbindungsschlüssel $k_s$, Nutzernamen $u$, Netzadresse des Nutzers $n$, Servicename $s$ =BOB, Gültigkeitsdauer $l$ und aktueller Zeit $t$. Verschlüsselt mit dem Schlüssel von Bob ergibt es das Verbindungszertifikat zu Bob. Zusammen mit dem Verbindungsschlüssel $k_s$ bildet es die Nachricht, die der Zertifikat-Server an Anne sendet, natürlich verschlüsselt mit Annes privatem Schlüssel. Der Zertifikat-Server und der Server Kerberos sind in diesem Fall die gleiche Einheit, die Unterscheidung dient nur dem besseren Verständnis.

Die Verbindungsaufnahme zu Bob bedarf nun stets des Verbindungszertifikats und eines Gültigkeitsetiketts $f_{k_s}(u : n : l : t)$, das wie oben gebildet wird. Aus dem Verbindungszertifikat kann sich Bob immer den Sitzungsschlüssel $k_s$ berechnen und das Gültigkeitsetikett prüfen. Wer weiß, mit wem er noch Verbindungen hat und die vielen Verbindungsschlüssel kann sich Bob gewiß nicht merken. Der Verbindungsschlüssel ist übrigens auch gut geeignet, die zu übermittelnden Nachrichten selbst zu verschlüsseln.

Die Lebensdauer eines Bewilligungszertifikats wird auf die durchschnittliche Länge einer Terminalsitzung berechnet, die des Gültigkeitsetiketts auf die Dauer der Nachrichtenübertragung im System.

**Übungsaufgabe 10** *Versuchen Sie alles, um die Verbindungen von Anne ins Netz abzuhören und zeigen Sie, daß Sie keine Chance haben, sich als Anne auszugeben !*

## 6.3 Kartenspiele über Telefon

Wir nehmen einmal an, daß eine Gruppe von Personen am Telefon Kartenspielen will. Man kann sich nun leicht vorstellen, daß dieses Unterfangen durch zahlreiche Betrugsversuche schnell zunichte gemacht werden kann. Es muß besonders darauf geachtet werden, daß die Karten korrekt gemischt, verteilt und keine zusätzlichen Karten ins Spiel gebracht werden. Der Fall, daß sich zwei Spieler zu Ungunsten eines Dritten untereinander absprechen, können wir mit diesem Verfahren allerdings (wie im richtigen Leben) nicht ausschließen !

Wir setzen voraus, daß jeder Spieler eine persönliche Ver- und Entschlüsselungsfunktion besitzt, die zueinander invers sind. Außerdem mögen für je zwei Spieler die entsprechenden Funktionen miteinander vertauschbar sein.

Diese Bedingungen erfüllt zum Beispiel das RSA-System. In jeder Spielrunde werden zwei ausgezeichnete Spieler vereinbart: der Mischer $M$ und der Geber $G$.

- $M$ mischt zunächst die Karten, chiffriert sie und sendet die „verdeckten“ Karten $c_i = E_M(m_{\pi(i)})$ an $G$.
- $G$ mischt die Karten noch einmal und verteilt sie an alle Mitspieler. Er behält eventuell den Talon.
- Jeder Spieler $A$ hat nun $k$ Karten $c_{A_1}, \ldots, c_{A_k}$. Diese verschlüsselt er mit seiner persönlichen Kodierungsfunktion $d_{A_i} = E_A(c_{A_i})$ und sendet die nun „2-fach verdeckten“ Karten an der Mischer. Dieser „dreht“ den Kartenstapel durch Anwendung seiner Entschlüsselungsfunktion einmal „um“ und sendet $d'_{A_i} = D_M(d_{A_i})$ an $A$ zurück. Weil sich $E_M$ und $D_M$ gegenseitig aufheben, sind diese gleich $E_A(m_{A_i})$, und $A$ kann sie nun schnell endgültig entschlüsseln und so sein tatsächliches Blatt ansehen.

Nicht gelöst ist damit jedoch das Problem, daß tatsächlich nur authentische Karten ausgespielt werden. Ein möglicher Weg wäre vielleicht, daß der jeweilige Mischer nicht mitspielt, sondern als Richter fungiert.

Wenn dann im Laufe des Spieles 2 Parteien dieselbe Karte ausspielen, werden sie aufgefordert, die entsprechende ursprünglich ausgeteilte „verdeckte“ Karte $c_i$ getrennt an den Mischer zu schicken.

Dieser prüft, welcher von beiden Spielern die richtige Chiffre für die umstrittene Karte vorweisen kann und entscheidet somit über Sieg oder Niederlage. Das ist genauso wie im Leben, wenn zwei gleiche Trumpf-Asse auf dem Tisch liegen, vergleicht man die Rückseiten miteinander.

Die Auswahl der „Karten“, genauer der Nachrichten $m_i$, darf man nicht dem Geber überlassen! Er kann sich durch gezielte Auswahl einen Vorteil verschaffen.

So kann er, falls das RSA-System verwendet wird, alle „Buben“-Nachrichten als quadratische Reste auswählen, eventuell durch Anfügen von Skatregeln oder klugen Kiebitzsprüchen. Beim Verschlüsseln durch den Mischer sind die Chiffren auch wieder quadratische Reste und der Geber kann durch Berechnen des Jacobi-Symbols die Buben lokalisieren.

Ganz genau gesagt, kann er viele Nicht-Buben aussondern, denn das Jacobi-Symbol ist für alle quadratischen Reste $+1$, und wenn es $-1$ ist, kann es kein Rest sein. Die absolute Sicherheit, daß sich hinter einer $+1$ auch ein Bube verbirgt, hat er nicht.

Am besten ist es in diesem Fall, wo so viel betrogen wird, in jeder Runde ein neues Kartenspiel zu verwenden, so könnte der Mischer immer eine Zufallsfolge an jede Karte anhängen.

Wenn sich dann aber Geber und Mischer absprechen, …

## 6.4 Münzwurf

Anne und Bob wollen ein schwieriges Entscheidungsproblem auf bewährte Art und Weise durch ein binäres Zufallsereignis lösen. Der konventionelle Münzwurf ist jedoch unpraktikabel, wenn beide nur über Telefon miteinander in Verbindung stehen. Wir wollen darum im folgenden ein System vorstellen, welches eine „elektronische" Version des Münzwurfes darstellt.

Grundlage bildet wieder das Problem der Primzahlzerlegung.

- Anne bildet eine Zahl $n$ aus zwei großen teilerfremden Primzahlen $n = p \cdot q$ und sendet diese Zahl an Bob.
- Bob wählt eine zu $n$ teilerfremde Zahl $x$, berechnet

  $$a \equiv x^2 \bmod n$$

  und schickt $a$ an Anne zurück.
- Anne bestimmt die 4 Lösungen $x_1, -x_1, x_2$ und $-x_2$ der Gleichung $x_i^2 \equiv a \bmod n$, was ihr aufgrund ihrer Kenntnis von $p$ und $q$ unter Anwendung des Chinesischen Restsatzes möglich ist.
- Sie entscheidet sich nun für $x_1$ oder $x_2$ und sendet diese Antwort an Bob.
- Wenn die Antwort von Anne mit Bob's ursprünglichen $x$ oder $-x$ übereinstimmt, so hat er verloren, anderenfalls gewonnen.

Im letzteren Fall muß Bob seinen Gewinn belegen. Er tut dies, in dem er die Primzahlzerlegung von n präsentiert. Sei nämlich o.B.d.A. die Antwort von Anne $x_1$ gewesen, dann gilt

$$x_1^2 \equiv x^2 \bmod n \qquad \text{und} \qquad x_1 \not\equiv x \bmod n \quad \text{bzw.} \quad x_1 \not\equiv -x \bmod n.$$

Daraus folgt

$$\begin{aligned} (x_1 - x)(x_1 + x) &\equiv 0 \pmod n \qquad \text{und} \\ p \cdot q &\mid (x_1 - x)(x_1 + x), \end{aligned}$$

d.h. $p$ und $q$ teilen entweder $(x_1 - x)$ oder $(x_1 + x)$, aber keinesfalls beide gleichzeitig einen der Faktoren. Mit dem Euklidischen Algorithmus bestimmt Bob dann ggT$(x_1 - x, n)$. Er erhält damit einen der Faktoren von $n$ und durch Division danach den anderen.

Zwar kann Bob mit diesem System seinen Verlust nicht beweisen, sollte dies aber einmal notwendig sein, so ist dieses Problem leicht durch geeignete Modifizierung der Entscheidungsfrage zu lösen.

## 6.5 Bitdepot

Ein Bitdepot ist eine Möglichkeit, bei einer Person / Institution (im folgenden „Bank" genannt) eine Information zu hinterlegen, die weder von der Bank noch von Dritten interpretiert werden kann. Allerdings soll die eingelagerte Information auch für den „Kontoinhaber" verbindlich sein, d.h. er darf nicht in der Lage sein, die Information durch Präsentation verschiedener Schlüssel beim Öffnen des Depots nachträglich zu ändern.

Grundlage des hier verwendeten Systems ist die Schwierigkeit der Konstruktion eines Isomorphismus zu zwei vorgegebenen isomorphen Graphen.

**Definition 16** *Zwei Graphen $G = [E_G, V_G]$ und $H = [E_H, V_H]$ sind isomorph gdw. eine bijektive Abbildung $f : G \longrightarrow H$ existiert mit*

- $f(V_G) = V_H$ *und*
- $(f(g_1), f(g_2)) \in E_H \iff e = (g_1, g_2) \in E_G$.

Anne möchte also bei ihrer Bank ein Bitdepot eröffnen und dort eine 0 oder 1 hinterlegen. Dazu muß die Bank zwei möglichst große isomorphe Graphen $G$ und $H$ veröffentlichen. Anne wählt ihre 0 oder 1, indem sie entsprechend einen zu $G$ oder $H$ isomorphen Graphen $K$ konstruiert[2]. Dieser Graph $K$ bildet ihre Depot-Einlage bei der Bank.

Möchte sie ihr Depot wieder öffnen, muß Anne ihren Isomorphismus $K \simeq G$ oder $K \simeq H$ präsentieren.

Wollte Anne betrügen, d.h. ein 0-Depot als ein 1-Depot eröffnen, müßte sie einen Isomorphismus zwischen $G$ und $H$ kennen. Und umgekehrt, wäre sie in der Lage, ein 0-Depot als 1-Depot zu öffnen, dann könnte sie auch sofort einen Isomorphismus präsentieren. Das heißt, daß das 0-1-Problem äquivalent zum Graphisomorphie-Problem ist.

Die Bank wiederum kann den drei in ihrem Besitz befindlichen untereinander isomorphen Graphen nichts entnehmen, solange Anne ihren Isomorphismus nicht offenlegt. Aber auch hinterher hat die Bank nichts Neues erfahren. Daß sie isomorph sind, wußte Bank vorher auch schon.

## 6.6 Null-Wissen-Beweise

In diesem Abschnitt wollen wir zeigen, daß es Anne möglich ist, Bob zu überzeugen, daß sie etwas weiß, ohne daß Bob erfährt, was es ist. Das erscheint nicht möglich, denn wir sind es gewohnt unser Wissen unter Beweis zustellen, indem wir es darlegen. Welche Folgen es für eine Prüfung hat, wenn man nur noch glaubwürdig machen muß, daß man etwas weiß, ist gar nicht auszudenken.

Wir behandeln dazu noch einmal das Beispiel der zwei isomorphen Graphen aus dem Bit-Depot. Anne kennt $G$ und $H$ und einen Isomorphismus zwischen ihnen. Sie will Bob davon überzeugen, ohne das Bob diesen Isomorphismus erfährt. Genau in dieser Situation ist damals die Bank gewesen, die zwei Graphen veröffentlichen sollte, die isomorph sind. Da mußte man glauben, daß die Bank nicht betrügt und zwei nichtisomorphe Graphen veröffentlicht. Jetzt könnten wir auch zu einer betrügerischen Bank gehen.

Anne wählt sich einen beliebigen dritten Graphen $K$ aus, der zu $G$ und $H$ isomorph ist und sendet ihn an Bob. Der wählt nun aus und verlangt von Anne entweder der Isomorphismus $G \simeq K$ oder $H \simeq K$. Wenn Anne wirklich einen Isomorphismus $G \simeq H$ kennt, dann kann sie auf beide Anfragen korrekt antworten. Andererseits lernt Bob nichts über den Isomorphismus zwischen $G$ und $H$ dazu. Nach $n$ Runden dieses *cut-and-choose* Spiels, ist die Wahrscheinlichkeit, daß Anne betrügt, nur noch $1/(2^n)$.

Natürlich ist Bob noch nicht restlos überzeugt, aber das wäre er gewiß auch nicht, wenn Anne ihm den Isomorphismus direkt angeben würde. Bei der erforderlichen Größe wäre eine Nachprüfung durch Bob auch fehlerbehaftet. Er könnte sich ja verhört haben!

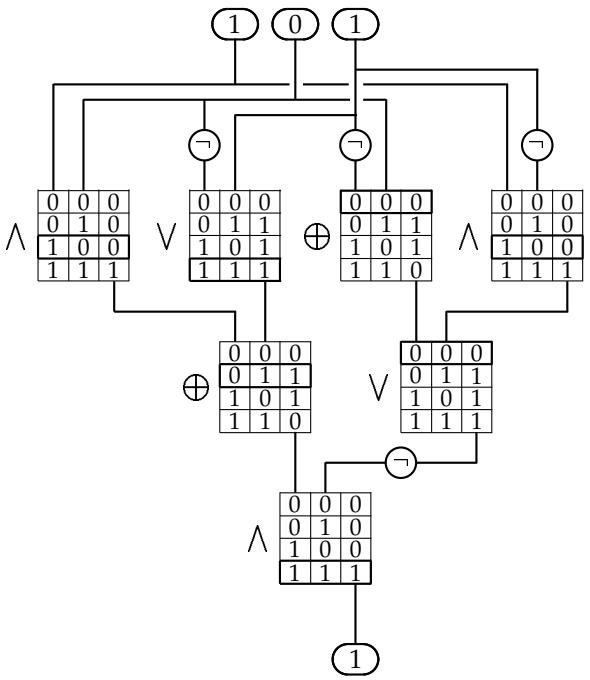
Der Nachweis, daß Bob überhaupt nichts über Annes Isomorphismus erfährt, ist etwas komplizierter, denn in der Tat erfährt Bob in jeder Runde von einem neuen Graphen, der zu einem der beiden isomorph ist. Hoffentlich wählt Anne nicht einmal $G = K$ aus.

[2] Die Konstruktion eines zu einem anderen Graphen isomorphen Graphen ist leicht (Umbenennung der Knoten), im Gegensatz dazu ist das Auffinden eines Isomorphismus zwischen zwei Graphen, selbst wenn man weiß, daß es ihn gibt, schwer!

Wir wollen noch ein weiteres Beispiel behandeln und werden zeigen, daß man jedes Wissen über ein **NP**-vollständiges Problem beweisen kann. Dazu wähle man sich eine komplizierte Boolesche Formel:

$$((p \wedge q) \oplus (\neg q \vee r)) \wedge \neg ((\neg r \oplus q) \vee (p \wedge \neg r)) .$$

Diese Formel kann man durch einen Booleschen Schaltkreis berechnen, über den sich Anne und Bob einigen. Für Anne, die eine erfüllende Belegung kennt, $101$, stellt sich die Berechnung in dem Schaltkreis wie folgt dar:




Die Belegung der Signalpegel an der Oberseite des Schaltkreises führt zu einer eindeutig bestimmten Auswahl in der Wahrheitstafel jeder Elementarzelle und schließlich zur $1$ an der Unterseite.

Wie im Fall der isomorphen Graphen teilt Anne ihr Wissen auf, so daß einerseits aus der Kenntnis jedes Teils für sich Bob keine neue Information gewinnen kann, und daß andererseits Bob sicher sein kann, daß nur der Besitzer des vollen Wissens die Aufteilung vornehmen kann.

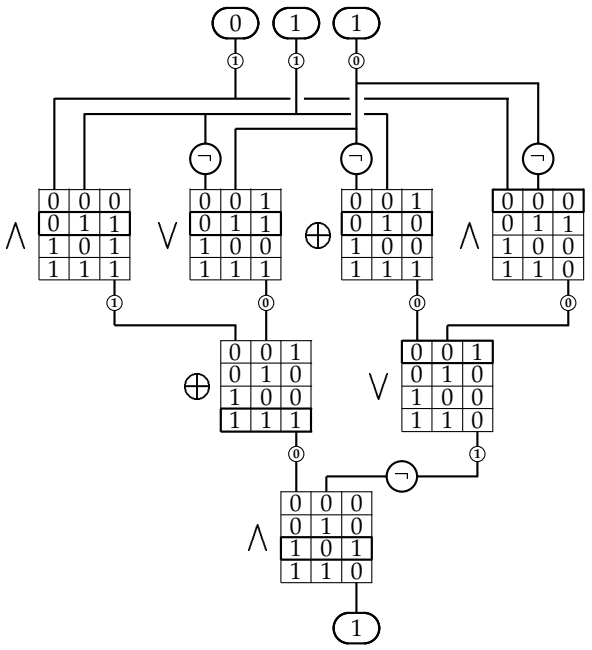
Dazu belegt Anne jeden Kommunikationsweg im Schaltkreis mit einem Bit, ausgenommen der Pfad vom letzten Gatter zur Ausgabe **1**. Auf jedem Weg addiert sie dieses Bit ($\oplus$-Addition) zu den durch den Weg verbundenen Spalten der Wahrheitstafeln.

So wird beim Bit $0$ nichts verändert. Beim Bit $1$ werden zum Beispiel auf der Verbindung zwischen dem $\wedge$- und dem $\oplus$-Gatter die Spalten der Gatter bitweise invertiert.

So entstehen an jeder Stelle neue Wahrheitstafeln, die natürlich nicht mehr Tafeln der entsprechenden Booleschen Funktionen sind. Anne sortiert diese Tafeln noch nach den ersten beiden Stellen, die ja trotz partiellem Kippen der Bits immer noch $00$, $01$, $10$ und $11$ sind.

Mit ihrem Wissen, der korrekten erfüllenden Belegung, kann Anne jederzeit auch in dem veränderten Schaltkreis einen Weg von der Eingabe dieser Werte bis zur $1$ am Ausgang

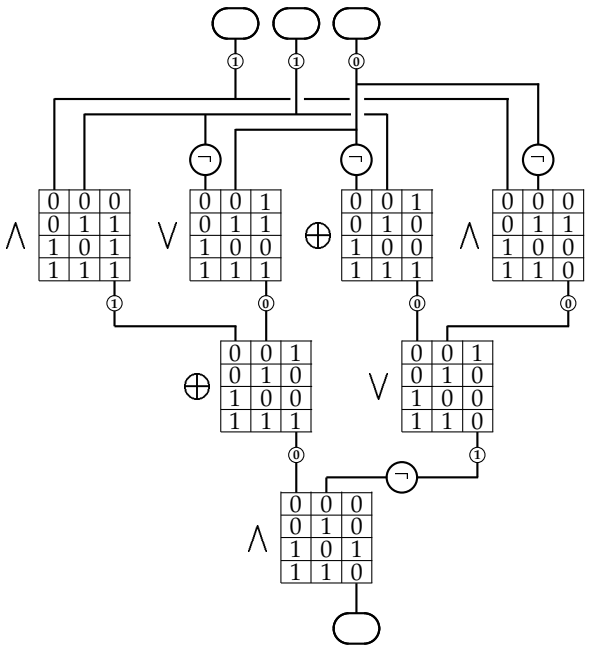
angeben. Ein „kippendes“ Bit auf einem Kommunikationsweg bedeutet die Veränderung der ursprünglichen Eingabe. An den Funktionselementen wird entsprechend der neuen Wahrheitstafeln gerechnet.



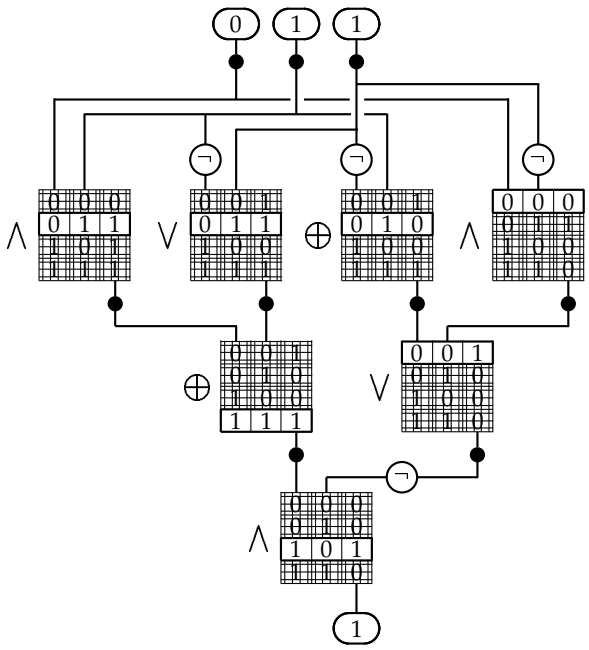
Anne teilt ihr Wissen auf. Sie sendet Bob ein verdecktes Exemplar des veränderten Schaltkreises.

Bob hat die Wahl, sich entweder eine Wahrheitstafel mit den umliegenden Kommunikationsbits, oder aber den Weg der teilweise „gekippten“ erfüllenden Belegung durch den Schaltkreis zeigen zu lassen.

Im ersten Fall öffnet Anne ihm das folgende Bild:



im anderen Fall:




Im ersten Fall überprüft Bob, ob die veränderten Wahrheitstafeln wirklich mit den angegebenen Kommunikationsbits aus den ursprünglichen Tafeln entstanden sind. Im zweiten Fall verfolgt Bob eine erfüllende Belegung.

Man sieht leicht, daß nur Anne, die eine erfüllende Belegung kannte, beide Möglichkeiten eröffnen kann. Bob hat keine Möglichkeiten, sich irgendein Wissen über die richtige erfüllende Belegung zu verschaffen.

# Anhang A

# Ausgewählte Verfahren der Kryptoanalyse

**Zusammenfassung**

Dieses Kapitel basiert auf dem Buch [Bau93], wobei Passagen fast wörtlich übernommen wurden. Die Lektüre dieses Buches ist nur zu empfehlen. Es wird das Kappa-Chi-Theorem als auch das Kappa-Phi-Theorem erläutert. Des weiteren wird auf die Periodenanalyse und auf die Methode des Zurechtrückens begleitender Alphabete eingegangen.

Ulrich Kunitz

## A.1 Kappa und Chi

Man kann von einem monoalphabetisch chiffrierten Text leichter herausfinden in welcher Sprache er verfaßt ist, als ihn entschlüsseln. Für Klartexte gilt dies natürlich auch. Es gibt auch ein einfaches Verfahren Herauszufinden ob zwei Texte in der selben Sprache abgefaßt sind.

### A.1.1 Definition und Invarianz von Kappa

Gegeben seien zwei Texte $T' = (t'_1, t'_2, t'_3, \ldots t'_M)$, $T'' = (t''_1, t''_2, T''_3, \ldots t''_M)$ gleicher Länge M über dem selben Zeichenvorrat. Die relative Häufigkeit der zeichenweise Übereinstimmung in den übereinanderliegenden Texten (die **Zeichenkoinzidenz**) wird als *Kappa*der beiden Texte bezeichnet. Es ist

$$\mathit{Kappa}(T', T'') = \sum_{\mu=1}^{M} \kappa(t'_\mu, t''_\mu)/M$$

mit der **Indikatorfunktion**

$$\kappa(x, y) = \begin{cases} 1, & \text{falls } x = y \\ 0, & \text{sonst.} \end{cases}$$

Es gilt

$$\begin{array}{ll} \mathit{Kappa}(T', T'') \leq 1, & \text{wobei} \\ \mathit{Kappa}(T', T'') = 1 & \text{genau dann, wenn } T' = T''. \end{array}$$

Durch Untersuchung von langen Texten der selben Sprache kann man finden, daß die Werte $\mathit{Kappa}(T', T'')_S = \kappa_S$ ziemlich übereinstimmen, sich aber von Sprache zu Sprache ändern. Folgende Werte wurden gefunden:

| S | $N$ | *Kullback* 1976 | *Eyraud* 1953 |
|---|---|---|---|
| Englisch | 26 | 6.61% | 6.75% |
| Deutsch | 26 | 7.62% | 8.20% |
| Französisch | 26 | 7.78% | 8.00% |
| Italienisch | 26 | 7.38% | 7.54% |
| Spanisch | 26 | 8.19% | 7.69% |
| Japanisch (Romaji) | 26 | 8.19% | |
| Russisch | 32 | 5.29% | 4.70% |

Die Werte für $\kappa_S$ spiegelt in etwa die Redundanz einer Sprache wieder. Die Übersetzung des Markus-Evangeliums, die im Englischen etwa $29\,000$ Silben hat, hat (nach *Mencken*) in den germanischen Sprachen im Mittel $32\,650$ Silben, in den romanischen Sprachen $40\,200$ und in den slawischen Sprachen $36\,500$. Es besteht kein strenger Zusammenhang.

Man hat also:

**Invarianzsatz 1:**

> Für alle polyalphabetischen, funktionalen monographischen Substitutionen, insbesondere für alle linearen polyalphabetischen monographischen Substitutionen gilt:
> *Das Kappa zweier (gleichlanger) Texte, die mit dem gleichen Schlüssel chiffriert werden, bleibt erhalten.*

**Invarianzsatz 2:**

> Für alle Transpositionen gilt:
> *Das Kappa zweier (gleichlanger) Texte, die mit dem gleichen Schlüssel chiffriert werden, bleibt erhalten.*

Man kann also aus dem *Kappa*die Sprache des Chiffrats erkennen.

Der Erwartungswert für das *Kappa*zweier Sprachen (mit gleichem Alphabet) errechnet sich aus den Wahrscheinlichkeiten $p'_i$, $p''_i$ des Auftretens der Zeichen in den „stochastischen Quellen“ $Q'$, $Q''$ der Sprachen: der Erwartungswert für das Auftreten des Zeichen $\chi_i$ an der $\mu$-ten Stelle beider Texte ist $p'_i \cdot p''_i$; es ergibt sich der Erwartungswert für $\mathit{Kappa}(T', T'')$

$$\langle \mathit{Kappa}(T', T'')\rangle_{Q'Q''} = \sum_{i=1}^{N} p'_i \cdot p''_i .$$

Sind beide Quellen identisch, $p_i = p'_i = p''_i$, so ist

$$\langle \mathit{Kappa}(T', T'')\rangle_{Q} = \sum_{i=1}^{N} p_i^2 .$$

**Satz 1** *Für identische Quellen $Q = Q' = Q''$ gilt*

$$\frac{1}{N} \leq \langle \mathit{Kappa}(T', T'')\rangle_{Q} \leq 1,$$

*wobei die linke Grenze angenommen wird für den Fall der Gleichchverteilung $Q_R : p_i = \frac{1}{N}$, und nur dafür; die rechte für jede deterministische Verteilung $Q_j : p_j = 1$, $p_i = 0$ für $i \neq j$, und für keine andere Verteilung.*

Es ergibt sich

$$\langle \mathit{Kappa}(T', T'') \rangle_{Deutsch} = 0.07619 = \kappa_d,$$
$$\langle \mathit{Kappa}(T', T'') \rangle_{Englisch} = 0.06577 = \kappa_e.$$

Für die Quelle mit Gleichverteilung $Q_R$ erhält man ($N = 26$)

$$\langle \mathit{Kappa}(T', T'') \rangle_R = \kappa_R = 0.03846 = \frac{1}{26}.$$

Der *Kappa*-Test unterscheidet deutsche und englische Quellen von einer gleichverteilten Quelle. Als Faustregel mag gelten:

$$\textit{Das Verhältnis} \quad \frac{\langle \mathit{Kappa}(T', T'') \rangle_S}{\langle \mathit{Kappa}(T', T'') \rangle_R} \quad \textit{liegt nahe bei zwei.}$$

Es ist

$$\frac{\kappa_d}{\kappa_R} = N \cdot \kappa_d = 1.98, \qquad \frac{\kappa_e}{\kappa_R} = N \cdot \kappa_e = 1.71.$$

### A.1.2 Definition und Invarianz von Chi

Es seien wieder zwei Texte gleicher Länge $M$ über dem gleichen Zeichenvorrat von $N$ Zeichen, $T' = (t'_1, t'_2, t'_3, \ldots t'_M)$, $T'' = (t''_1, t''_2, t''_3, \ldots t''_M)$. Die Häufigkeit $\chi_i$ des Vorkommens des $i$-ten Zeichens im Text $T'$ sei mit $m'_i$, im Text $T''$ mit $m''_i$ bezeichnet. Es ist

$$\sum_{i=1}^{N} m'_i = M, \qquad \sum_{i=1}^{N} m''_i = M.$$

Als *Chi*wird

$$\mathit{Chi}(T', T'') = \frac{\sum_{i=1}^{N} m'_i \cdot m''_i}{M^2}$$

bezeichnet. Anders geschrieben

$$\mathit{Chi}(T', T'') = \frac{\sum_{i=1}^{N} m'_i \cdot m''_i}{(\sum_{i=1}^{N} m'_i) \cdot (\sum_{i=1}^{N} m''_i)}.$$

Ähnlich wie für *Kappa*gilt

| | |
|---|---|
| $\mathit{Chi}(T', T'') \leq 1,$ | wobei |
| $\mathit{Chi}(T', T'') = 1$ | genau dann, wenn $T'$ und $T''$ aus einem einzigen Zeichen aufgebaut sind. |

Sind alle $m'_i$ gleich, $m'_i = \frac{M}{N}$, so ist, (bei beliebigen $m''_i$)

$$\mathit{Chi}(T', T'') = \frac{1}{N} = \kappa_R.$$

Empirisch findet man wieder, daß Texte aus ein und der selben Sprache $S$ nicht nur übereinstimmende Werte von *Chi*haben, sondern sie liegen noch bei den *Kappa*-Werten für $S$.

Wichtig ist der Spezialfall $T' = T'' = T$, $m_i' = m_i'' = m_i$. Sei

$$\mathit{Sigma}(T) = \mathit{Chi}(T,T) = \sum_{i=1}^{N} \frac{m_i^2}{M^2}.$$

Es gilt der Steinersche Satz

$$\sum_{i=1}^{N} \frac{(m_i - \frac{M}{N})^2}{M^2} = \sum_{i=1}^{N} \frac{m_i^2}{M^2} - \frac{1}{N}.$$

Somit gilt

| | |
|---|---|
| $\frac{1}{N} \leq \mathit{Sigma}(T) \leq 1,$ | wobei[1] |
| $\mathit{Sigma}(T) = \frac{1}{N} = \kappa_R$ | genau dann, wenn alle $m_i$ gleich sind, |
| $\mathit{Sigma}(T) = 1$ | genau dann, wenn $T$ aus einem einzigen Zeichen aufgebaut ist. |

Nun seien die schwächeren Invarianzeigenschaften für *Chi*und *Sigma*formuliert:

**Invarianzsatz $3^\chi$:**

> Für alle monoalphabetischen, funktionalen monographischen Substititutionen, insbesondere auch für alle linearen monoalphabetischen monographischen Substitutionen gilt:
> *Das Chi zweier (gleichlanger) Texte, die mit dem gleichen Schlüssel chiffriert werden, wie auch das Sigma eines Textes, bleiben erhalten.*

**Invarianzsatz $4^\chi$:**

> Für alle Transpositionen gilt:
> *Das Chi zweier (gleichlanger) Texte, die mit dem gleichen Schlüssel chiffriert werden, wie auch das Sigma eines Textes, bleiben erhalten.*

Die Sprache kann also mit *Chi*und *Sigma*aus dem Chiffrat erkannt werden.

Der Erwartungswert für das *Chi*zweier Sprachen (mit gleichem Alphabet) errechnet sich direkt aus den Wahrscheinlichkeiten $p_i'$, $p_i''$ des Auftretens der Zeichen in den „stochastischen Quellen" $Q'$, $Q''$ der Sprachen: der Erwartungswert für die Anzahl der Zeichen $\chi_i$ ist $p_i' \cdot M$ bzw. $p_i'' \cdot M$; es ergibt sich

$$\langle \mathit{Chi}(T', T'') \rangle_{Q'Q''} = \sum_{i=1}^{N} p_i' \cdot p_i''.$$

Sind die beiden Quellen identisch, $p_i = p_i' = p_i''$, so ist

$$\langle \mathit{Chi}(T', T'') \rangle_{Q} = \sum_{i=1}^{N} p_i^2.$$

[1] Für den extremen Fall $M < N$ gilt schärfer $1/M \leq \mathit{Sigma}(T)$, wobei $1/M = \mathit{Sigma}(T)$ genau dann, wenn $0 \leq m_i \leq 1$.

Insbesondere gilt

$$\langle \mathit{Sigma}(T)\rangle_Q = \sum_{i=1}^{N} p_i^2.$$

**Satz 2** *Für identische Quellen $Q = Q' = Q''$ gilt*

$$\frac{1}{N} \leq \langle \mathit{Chi}(T', T'')\rangle_Q \leq 1,$$

*und speziell*

$$\frac{1}{N} \leq \langle \mathit{Sigma}(T)\rangle_Q \leq 1,$$

*wobei die linke Grenze angenommen wird für den Fall der Gleichverteilung $Q_R : p_i = \frac{1}{N} = \kappa_R$, und nur dafür; die rechte für jede deterministische Verteilung $Q_j : p_j = 1, p_i = 0$ für alle $i \neq j$, und für keine andere Verteilung.*

Man kann erstaunt darüber sein, daß die beiden Erwartungswerte $\langle \mathit{Kappa}(T', T'')\rangle_Q$ und $\langle \mathit{Chi}(T', T'')\rangle_Q$ zusammenfallen. Es besteht aber sogar ein Zusammenhang zwischen $\mathit{Kappa}(T', T'')$ und $\mathit{Chi}(T', T'')$.

### A.1.3 Das Kappa-Chi-Theorem

Wir benötigen zwei Hilfsfunktionen $g'_{i,\mu}$, $g''_{i,\mu}$. Sei

$$g'_{i,\mu} = \begin{cases} 1, & \text{falls } t'_\mu \text{ (das } \mu\text{-te Zeichen von } T') = \chi_i \\ 0, & \text{sonst} \end{cases};$$

entsprechend sei $g''_{i,\mu}$ für $T''$ definiert. Dann ist $\kappa(t'_\mu, t''_\nu) = \sum_{i=1}^{N} g'_{i,\mu} \cdot g''_{i,\nu}$ und $m'_i = \sum_{\mu=1}^{M} g'_{i,\mu}$, $m''_i = \sum_{\nu=1}^{M} g''_{i,\nu}$.
Sei

$$\mathit{Kappa}(T', T'')^r = \sum_{\mu=1}^{M} \frac{\kappa(t'_\mu, t''_{(\mu+r) \bmod M})}{M}.$$

Damit ist $\mathit{Kappa}(T', T'')^0 = \mathit{Kappa}(T', T'')$. $\mathit{Kappa}(T', T'')^r$ ist die Anzahl der Koinzidenzen zwischen dem $T'$ und dem um $r$ Plätze zyklisch verschobenen $T''$.

**Kappa-Chi-Theorem:**

$$\frac{1}{M} \sum_{\rho=0}^{M-1} \mathit{Kappa}(T', T'')^\rho = \mathit{Chi}(T', T'').$$

$\mathit{Chi}(T', T'')$ ist also das arithmetische Mittel aller $\mathit{Kappa}(T', T'')^r$.

**Korollar:**

$$\frac{1}{M} \sum_{\rho=0}^{M-1} \mathit{Kappa}(T, T)^\rho = \mathit{Sigma}(T).$$

Der Beweis erfolgt durch Termumformungen und bleibt dem Leser überlassen.

### A.1.4 Das Kappa-Phi-Theorem

Der Fall $T = T' = T''$ bietet eine Besonderheit: Es ist *Kappa*$(T,T) =$
*Kappa*$(T,T)^0 = 1$. Dagegen ist für $r \neq 0$ mit einem bedeutend kleineren Wert für *Kappa*$(T,T)^r$ zu rechnen. Somit fällt bei der Mittelbildung der Fall $r = 0$ aus dem Rahmen, und es empfiehlt sich eine Mittelbildung über $M-1$ Werte, nämlich

$$\frac{1}{M-1}\sum_{\rho=1}^{M-1} \mathit{Kappa}(T,T)^\rho$$

zu betrachten.

Nun ist

$$\begin{aligned}
&\tfrac{1}{M-1}\textstyle\sum_{\rho=1}^{M-1} \mathit{Kappa}(T,T)^\rho &&= \\
&\tfrac{1}{M-1}(\textstyle\sum_{\rho=0}^{M-1} \mathit{Kappa}(T,T)^\rho - 1) &&= \\
&\tfrac{1}{M-1}(M \cdot \mathit{Sigma}(T) - 1) &&= \\
&\tfrac{1}{M-1}(\textstyle\sum_{i=1}^{N} \frac{m_i^2}{M} - 1) &&= \\
&\tfrac{1}{M-1} \cdot \tfrac{1}{M} \cdot (\textstyle\sum_{i=1}^{N} m_i^2 - M) &&= \\
&\tfrac{1}{M-1} \cdot \tfrac{1}{M} \cdot (\textstyle\sum_{i=1}^{N} (m_i^2 - m_i) &&= \\
&\tfrac{1}{M-1} \cdot \tfrac{1}{M} \cdot (\textstyle\sum_{i=1}^{N} m_i \cdot (m_i - 1))
\end{aligned}$$

Man definiert

$$\mathit{Phi}(T) = \frac{\sum_{i=1}^{N} m_i \cdot (m_i - 1)}{M \cdot (M-1)}$$

und haben nun das **Kappa-Phi-Theorem**:

$$\frac{1}{M-1}\sum_{\rho=1}^{M-1} \mathit{Kappa}(T,T)^\rho = \mathit{Phi}(T).$$

Der Vorteil der Berechnung von *Phi*$(T)$ gegenüber der von *Sigma*$(T)$ besteht darin, daß für $m_i = 0$ wie für $m_i = 1$ kein Beitrag zur Summe anfällt. Dies ist bei kurzen Texten für seltene Buchstaben nützlich.

*Phi*$(T)$ ist von *Sigma*$(T)$ nicht sehr verschieden: es gilt

$$\begin{aligned}
&\mathit{Phi}(T) = \tfrac{M}{M-1}\mathit{Sigma}(T) - \tfrac{1}{M-1}, \mathit{Sigma}(T) = \tfrac{M-1}{M}\mathit{Phi}(T) + \tfrac{1}{M} \quad \text{und} \\
&\mathit{Sigma}(T) - \mathit{Phi}(T) = \tfrac{1-\mathit{Phi}(T)}{M} = \tfrac{1-\mathit{Sigma}(T)}{M-1}, \quad \text{also} \\
&\mathit{Phi}(T) \leq \mathit{Sigma}(T).
\end{aligned}$$

*Chi*hat die gleichen Invarianzeigenschaften wie *Sigma*:

**Invarianzsatz** $3^\varphi$**:**

> Für alle monoalphabetischen, funktionalen monographischen Substititutionen, insbesondere auch für alle linearen monoalphabetischen monographischen Substitutionen gilt:
> *Das Phi eines Textes bleibt erhalten.*

**Invarianzsatz** $4^{\varphi}$**:**

Für alle Transpositionen gilt:
*Das Phi eines Textes bleibt erhalten.*

Der Erwartungswert für das *Chi*eines Textes $T$ der Länge $M$ errechnet sich ebenfalls aus der Wahrscheinlichkeit $p_i$ des Auftretens der Zeichen in der „stochastischen Quelle“ der Sprache: er ergibt sich zu

$$\langle \mathit{Phi}(T)\rangle_Q^{(M)} = \frac{M}{M-1}\cdot\Big(\sum_{i=1}^{N} p_i\cdot\big(p_i-\frac{1}{M}\big)\Big),\quad \text{wobei}$$

$$\langle \mathit{Phi}(T)\rangle_Q^{(M)} \geq \begin{cases} \frac{M}{M-1}\cdot\big(\frac{1}{N}-\frac{1}{M}\big)=\frac{1}{N}\cdot\frac{M-N}{M-1}, & \text{falls } M>N \\ 0, & \text{falls } M\leq N \end{cases}.$$

Für größer werdendes M nähert sich der Erwartungswert für $\mathit{Phi}(T)$ dem Erwartungswert für $\mathit{Sigma}(T)$, nämlich

$$\langle \mathit{Phi}(T)\rangle_Q^{(\infty)} = \sum_{i=1}^{N} p_i^2.$$

### A.1.5 Symmetrische Funktionen der Zeichenhäufigkeiten

Die Invarianz von Satz $4$ und $4^{\varphi}$ gilt für alle symmetrischen Funktionen der Zeichenhäufigkeiten. Die einfachste nichtkonstante polynomiale ist $\sum_{i=1}^{N} m_i^2$. Eine interessante Klasse ist die folgende:

$$\mathit{Sigma}_a(T) = \begin{cases} \big(\sum_{i=1}^{N}\big(\frac{m_i}{M}\big)^a\big)^{\frac{1}{a-1}} & \text{falls } 1<a<\infty \\ \exp\big(\sum_{i=1}^{N}\frac{m_i}{M}\cdot\ln\frac{m_i}{M}\big) & \text{falls } a=1 \\ \max_{i=1}^{N}\frac{m_i}{M} & \text{falls } a=\infty \end{cases}$$

mit der Normierung $\sum_{i=1}^{N}\frac{m_i}{M}=1$.

$\mathit{Sigma}_1$ und $\mathit{Sigma}_\infty$ sind stetige Grenzfunktionen, $\mathit{Sigma}_2$ ergibt Sigma von oben. Es gilt für $1\leq a\leq\infty$

$$\mathit{Sigma}_a(T)=\frac{1}{N}\quad \text{genau dann, wenn alle } m_i \text{ gleich sind.}$$

Die Größe $-\operatorname{ld}\mathit{Sigma}_a(T)$ heißt **Renyi-a-Entropie** von $T$; man erhält

$$-\operatorname{ld}\mathit{Sigma}_a(T) = \begin{cases} -\frac{1}{a-1}\cdot\operatorname{ld}\big(\sum_{i=1}^{N}\big(\frac{m_i}{M}\big)^a\big) & \text{falls } 1<a<\infty \\ -\big(\sum_{i=1}^{N}\big(\frac{m_i}{M}\cdot\operatorname{ld}\frac{m_i}{M}\big)\big) & \text{falls } a=1 \\ -\max_{i=1}^{N}\operatorname{ld}\frac{m_i}{M} & \text{falls } a=\infty \end{cases}$$

Die *Renyi*-1-Entropie ist die **Shannon-Entropie** (*Claude E. Shannon*, 1946).

Für einen englischen Text ergibt sich

$$\begin{array}{llll} \mathit{Sigma}_1(T) &= 5.851\% & -\operatorname{ld}\mathit{Sigma}_1(T) &= 4.095 \\ \mathit{Sigma}_2(T) &= 6.936\% & -\operatorname{ld}\mathit{Sigma}_2(T) &= 3.850 \\ \mathit{Sigma}_4(T) &= 8.295\% & -\operatorname{ld}\mathit{Sigma}_4(T) &= 3.592 \\ \mathit{Sigma}_\infty(T) &= 12.857\% & -\operatorname{ld}\mathit{Sigma}_\infty(T) &= 2.959 \end{array}$$

## A.2 Periodenanalyse

Periodische polyalphabetische Chiffrierung enthält trotz einer Fülle von möglicherweise unabhängiger Alphabete ein schwer zu versteckendes Element: die Periode.

**Satz 3** *Ist $d$ die Periode einer periodischen polyalphabetischen Chiffrierung, so stammt das Chiffrat $T$ eines Textes $P$ und das um $k \cdot d$ Plätze zyklisch verschobene Chiffrat $T^{k \cdot d}$ aus der selben stochastischen Quelle $Q$,*

$$\langle \mathit{Kappa}(T, T^{k \cdot d}) \rangle_Q = \sum_{i=1}^{N} p_i^2.$$

Über $\langle \mathit{Kappa}(T, T^u) \rangle_Q$ mit $d \not| u$ kann man diese Aussage nicht machen. In der Regel stammen $T$ und $T^u$ aus voneinander unabhängigen stochastischen Quellen, in diesem Fall ist:

$$\langle \mathit{Kappa}(T, T^u) \rangle_Q = \langle \mathit{Kappa}(T', T'') \rangle_R = \frac{1}{N} \quad \text{für } d \not| u.$$

### A.2.1 Friedmanns Periodenbestimmung durch Kappa-Verlauf

Man berechnet nun für eine gegebene Chiffre den Verlauf von *Kappa*$(T, T^u)$ für $u = 1, 2, \ldots$. An den Stellen $k \cdot d$ müssen nach obigem Satz die Werte nah bei dem *Kappa*der Sprache des Originaltextes liegen. Für längere Texte wird die Periode viel deutlicher. Für Texte mit ca. 3000 Zeichen springt einem die Periode förmlich ins Auge.

Man braucht die Diskussion nicht nur auf Einzelzeichen beschränken Bi- und Multigramme können auch als Zeichen aufgefaßt werden, wobei allerdings der Zeichenvorrat stark ansteigt. Für Bigramme ist $\kappa_R^{(2)} = \frac{1}{N^2} = 14.8\%\%$, für Trigramme $\kappa_R^{(3)} = \frac{1}{N^3} = 0.569\%\%$.

$\kappa_S^{(i)}$ hebt sich deutlicher von $\kappa_R^{(i)}$ ab als bei der Einzelzeichenbetrachtung (Faktor 4.5 bis 7.5). Nach *Kullback* liegt $\kappa_e^{(2)}$ bei 69%%, für das Deutsche nach *Kullback* und *Bauer* bei 112%%. Allerdings werden die Schwankungen auch bei 3000 Zeichen sehr beträchtlich.

Man kann davon ausgehen, daß diese Methode im 2. Weltkrieg von Amerikanern, Deutschen und Japanern verwendet wurde.

### A.2.2 Parallelstellensuche nach Kasiski

Als Grenzfall des *Kappa*-Verlaufs für Multigramme bestimmt man alle echten Multigramme, die sich wiederholen. Diese von *F. W. Kasiski* 1863 publizierte „Parallelstellensuche“ brach den bis dahin weit verbreiteten Glauben an die Unbrechbarkeit polyalphabetischer Chiffren zumindest im periodischen Fall.

Das Verfahren läuft auf die Bestimmung der Abstände von gleichen Multigrammen in der Chiffre hinaus. Der größte gemeinsame Teiler dieser Abstände ergibt die Periode. Allerdings kann es vorkommen, daß diese Multigramme zufällig entstanden sind. Man muß also echte von unechten Multigrammen unterscheiden. Sehr viele unechte Multigramme entstehen, wenn das Schlüselwort aus dem Genre des Textes entnommen wird, die erschwert die Analyse nach *Kasiski* als auch *Kappa*- und *Chi*-Test.

Während des zweiten Weltkriegs wurde dieses Verfahren von den Deutschen und vermutlich auch von den Briten mit COLLOSSUS verwendet.

### A.2.3 Kolonnenbildung und Phi-Test nach Kullback

Man unterteilt den Geheimtext in Kollonnen $T_\rho$ aller Zeichen mit dem Abstand $\rho$. Wenn die Periode getroffen wurde, ist jede Kolonne mit dem selben Alphabet chffriert, andernfalls mit verschiedenen Alphabeten.

Bildet man also *Phi*$(T_\rho)$ für $\rho = 1, 2, \ldots, u-1, u$, so sollte man für alle *Phi*$(T_\rho)$ im ersten Fall $(u = r \cdot d)$ Werte nahe $\kappa_S$ erwarten, im zweiten Falle aber nahe an $\kappa_R = \frac{1}{N}$.

Am besten ist es zu mitteln. Man bildet für $\rho = 1, 2, \ldots, u-1, u$

$$\phi_\rho = \sum_{i=1}^{N} m_i^\rho \cdot (m_i^\rho - 1)$$

und summiert dann;

$$Phi^{(u)}(T) = u \cdot \frac{\sum_{\rho=1}^{u} \phi_\rho}{M \cdot (M-1)}.$$

Der Faktor u dient zur Normalisierung.

**Kappa-Phi$^{(u)}$-Theorem:**

$$\frac{1}{M-1} \sum_{\rho=1}^{M-1} Kappa(T,T)^{u\cdot\rho} = Phi^{(u)}(T).$$

*Phi*$^{(u)}$ ist also der Mittelwert aller *Kappa*-Werte im Abstand $u$ und erweist sich damit als sehr scharfes Instrument.

Man betrachtet den Werteverlauf von *Phi*$^{(u)}$ für $u = 2, 3, 4, \ldots$. Die Periode müßte dann sehr deutlich zu Tage treten.

### A.2.4 Eine Abschätzung der Periodenlänge

Aus dem *Kappa-Chi*-Theorem ergibt sich für die Erwartungswerte

$$\langle Phi(T)\rangle_Q^{(M)} = \frac{1}{M-1} \sum_{\rho=1}^{M-1} \langle Kappa(T,T)^\rho\rangle_Q .$$

Es wurde schon darauf hingewiesen, daß

$$\langle Kappa(T, T^{k\cdot d})\rangle_Q = \kappa_S ,$$

und des weiteren, daß „in der Regel“

$$\langle Kappa(T, T^u)\rangle_Q = \kappa_R = \frac{1}{N} \quad \text{für } d \not| u.$$

Wird der Einfachheit halber angenommen, $M$ sein ein Vielfaches der Periode $d$, so ergibt sich in der Summe $M/d-1$ mal $\kappa_S$, die übrigen Male $\kappa_R$; also ist $\langle Phi(T)\rangle_Q^{(M)}$ eine Mittelung aus $\kappa_S$ und $\kappa_R$,

$$(M-1) \cdot \langle Phi(T)\rangle_Q^{(M)} = (\frac{M}{d} - 1) \cdot \kappa_S + ((M-1) - (\frac{M}{d} - 1)) \cdot \kappa_R .$$

Unter der Annahme, daß das beobachtete *Phi*$(T)$ den Erwartungswert approximiert, gilt also (*Abraham Sinkov*, um 1935)

$$(M-1)\cdot \mathit{Phi}(T) \approx \frac{1}{d}\cdot\kappa_S + ((M-1) - (\frac{M}{d}-1))\cdot\kappa_R.$$

Diese Beziehung zeigt, wie bei gleichbleibender Quelle mit wachsender Periode einer polyalphabetischen Chiffrierung der Wert von *Chi*sich ändert. Für große $M$ und $d \ll M$ kann man mit

$$\mathit{Phi}(T) \approx \frac{1}{d}\cdot\kappa_S + (1-\frac{1}{d})\cdot\kappa_R$$

fast ebenso gut arbeiten.

Die Beziehung von *Sinkov* läßt sich auch nach $d$ auflösen:

$$\begin{aligned} (\frac{M}{d}-1) &\approx \frac{(M-1)\cdot \mathit{Phi}(T) - \kappa_R}{\kappa_S - \kappa_R}, \qquad \text{d.h.} \\ d &\approx \frac{\kappa_S - \kappa_R}{\frac{\kappa_S-\kappa_R}{M} + (\mathit{Phi}(T) - \kappa_R)}. \end{aligned}$$

Zur Unterstützung einer Periodenanalyse nach *Kasiski*, *Friedman* oder *Kullback* ist die Abschätzungsformel – insbesondere für kleinere Perioden – brauchbar, allein wäre sie wegen ihrer großen Streuung ohne Nutzen.

## A.3 Zurechtrücken begleitender Alphabete

Wenn man die Periode für eine Chiffre hat, sind bei polyalphabetischen Verschlüsselungen die Alphabete zurechtzurücken.

### A.3.1 Durchdecken der Häufigkeitsgebirge

Für die VIGENÉRE-Chiffrierung muß man bei einer Periode $d$ nur die $d$ Häufigkeitsgebirge für die einzelnen Kolonnen untersuchen. Es empfiehlt sich hier besonders für große Perioden die graphische Darstellung.

### A.3.2 Zurechtrücken gegen bekanntes Alphabet

Rechnerische Methoden des „Zurechtrückens“ erweisen sich als schärfere Methoden. Es bietet sich an, die Verschiebung einer bestimmten Kolonne gegen das Referenzalphabet durch die Berechnung des *Chi*zu prüfen.

Man braucht bloß aus den Verschiebungen mit den maximalen *Chi*-Werten durch Ausschlußverfahren (Bigramme zwischen benachbarten Kolonnen) die richtige bestimmen.

Diese Methode ist für eine beliebige polyalphabetische Chiffrierung mit beliebigen unabhängigen Alphabeten brauchbar, wenn all Alphabete in unberufene Hände gefallen sind.

Es gibt auch eine Reihe von mechanischen Verfahren, die darauf hinauslaufen die Anzahl der häufigsten Zeichen einer Sprache bei einer bestimmten Verschiebung der Kolonne zu zählen.

### A.3.3 Gegenseitiges Zurechtrücken begleitender Alphabete

Ist das Referenzalphabet nicht das Standardalphabet, so verbleibt immer noch die Möglichkeit, die einzelnen Alphabete gegenseitig zurechtzurücken und damit den polyalphabetisch chiffrierten Geheimtext in einen monoalphabetischen zu überführen und diesen mit bekannten Verfahren zu brechen. Dieses Vorgehen ist auch dann brauchbar, wenn man nicht weiß oder erkennt, daß das Referenzalphabet das Standardalphabet ist.

Dazu berechnet man die *Chi*-Werte gegeneinander verschobener Kolonnen. „In der Regel" zeichet sich ein Wert aus, der in der Nähe von $\kappa_S$ liegt und die anderen sind in der Nähe von $\kappa_R$ zu finden. Es gibt nun verschiedene Strategien der Ausrichtung. Es ist zu entscheiden, welche Kolonne gegen welche im nächsten Schritt auszurichten ist. Bei kurzen Texten muß man wohl zur Methode des gegenseitigen Ausschlusses greifen.

Wenn man die Häufigkeitsgebirge durchgedeckt hat, kann es sein, daß man das Schlüsselwort erkennen kann.

### A.3.4 Wiedergewinnung des Referenzalphabets

Man wandelt den Text in ein monoalphabetisch verschlüsselten Text um (möglicherweise nach dem Alphabet der ersten Kolonne) und ermittelt die Häufigkeitsverteilung. Aus dieser schließt man dann auf die Verschlüsselung der häufigstens Buchstaben. Jetzt entschlüsselt man den Text mit bekannten Methoden (z.B. Suchen nach Wortmustern aus dem Textgenre). Damit kann man einen Teil des Referenzalphabets rekonstruieren. Da der Schlüssel häufig nicht aus einer vollständigen Permutation des Alphabets besteht, sondern kürzer ist, kann man wahrscheinlich auch den fehlenden Rest rekonstruieren.

### A.3.5 Kerckhoffs' symétrie de position

Häufig gibt es Anhaltspunkte für wahrscheinliche Worte und es besteht die Möglichkeit einer Mustererkennung. Dabei können für jedes der begleitenden Alphabete auch gewisse seltenere Klartext-Buchstaben entziffert werden. *Kerckhoffs* erkannte 1883, daß es in geeignet Fällen möglich ist, eine solche Entzifferung auch in andere Kolonnen zu übertragen. Er nannte die diesbezügliche auf der Kommutativität der Addition in $\mathbf{Z}_N$ (ganze Zahlen $0, 1, \ldots, N$) beruhende Eigenschaft der verschobenenen Alphabete *symétrie de position*.

Aus einem polyalphabetischen Text wurde durch Kombination von Häufigkeitsanalyse und Mustererkennung folgender Text gewonnen:

```
R B N B J  J H G T S  P T A B G  J X Z B G  J I C E M  Q A M U W
l e g e n e r a l w o l s e l  e y t e l  e g r a p  h i e * *
```

Bisher hat man folgendes entziffert:

```
     a b c d e f g h i j k l m n o p q r s t u v w x y z
(1)        J     Q     R     P
(2)        B  I     A  T           H              X
(3)  G     M  N                    C A Z
(4)  E     B           T
(5)        Z           G  J  M
```

Es fällt auf das das 2. und 4. Zeile identisch sein muß. Die 1. gegenüber der 5. um 9 Stellen verschoben ist, aber auch die 3. gegenüber der 5. um 11 Stellen. So kann man das Alphabet in den einzelnen Zeilen ergänzen. Damit kann man zwar nicht auf Anhieb das ganze Alphabet erkennen, aber in diesem Beispiel dürften 17 erkannte von 26 Zeichen ausreichend sein.

Die *symétrie de position* ist auch bei einer VIGÉNERE-Chiffrierung von Nutzen. Man ermittelt die Differenzen aller Chiffrenpaare in einer Kolonne, dabei müßten eine Reihe von Werten wesentlich häufiger auftreten. Man bildet dann Tripel für jede Kolonne. Das könnte dann zum Beispiel so aussehen:

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | | B | M | | W | L | M | | M | X | L |
| +7 | | | | | | | | | | | | |
| | | N | I | T | E | D | S | | A | T | E | S |
| +4 | | | | | | | | | | | | |
| | Y | R | M | X | I | H | W | X | E | X | | W |
| | *0* | *19* | *14* | *25* | *10* | *9* | *24* | *25* | *6* | *25* | *10* | *24* |

Die zur Deckung notwendigen Verschiebungen der einzelnen Kolonnen gegen die erste sind in einer Fußzeile eingetragen, sie ergeben einen Schlüssel. Den reduzierten monoalphabetischen chiffrierten Zwischentext erhält man durch Subtraktion dieses Schlüssels. Die Entzifferung durch Häufigkeitsanalyse sollte kein Problem mehr darstellen.

### A.3.6 Abstreifen einer Überchiffrierung: Differenzenmethode

Das gegenseitige Zurechtrücken begleitender Alphabete nimmt auf den Klartext keinen Bezug; er wird erst aus dem monoalphabetisch chiffrierten Zwischentext wiedergewonnen. Damit ist diese Technik auch für überchiffrierten Code verwendbar, also eine Kombination von Codierung und nachfolgendem VIGÉNERE über $\mathbb{Z}_{26}$ oder $\mathbb{Z}_{10}$. Unter Annahme einer gewissen Codelänge und einer gewissen Periode werden wieder Kolonnen von gleich überchiffrierten Codewörtern gebildet („Latten"). Häufig vorkommende Klartextworte und -begriffe führen in jeder dieser Latten zu Häufigkeitsunterschieden, bei genügend Material heben sich in jeder Latte gewisse Codegruppen heraus. Die reine Häufigkeitsuntersuchung, etwa durch Bildung von gegenseitigen *Chi*mag erfolgreich sein; oft wird sie aber kein gegenseitiges Ausrichten zustande bringen.

Die *symétrie de position* hilft jedoch häufig weiter. Die sich in jeder Latte heraushebenden Codegruppen gehören genau dann zum selben Placode, wenn ihre Differenz die Differenz der zu den betreffenden Latten gehörenden Überchiffrierungszahlen ergibt. Differenzen sind nach *Shannon* die 'residue classes' der linearen polygraphischen Substitution. Die Entzifferung der nach Abstreifen der Überchiffrierung erhaltenen Codes ist durch philologische Arbeit zu leisten.

# Anhang B

# Nachrichtenaustausch mit Authentisierung durch Schiedsrichter

Im dem hier benutzten Modell für Nachrichtenauthentisierung werden vier Parteien unterschieden, der Sender, der Empfänger, der Gegner und der Schiedsrichter. Der Sender hat eine Information, in unserem Beispiel das Ergebniss eines Münzwurfes, die er einem Empfänger über einen öffentlichen Kanal mitteilen will. Der Empfänger will nicht nur die Information kennenlernen, sondern auch sicher sein, daß die Nachricht nicht von Gegnern stammt oder verfälscht wurde. Der Gegner will erreichen, daß er den Empfänger täuschen kann, indem er sich als Sender ausgibt oder eine bestehende Nachricht fälscht. Der Schiedsrichter hat die Aufgabe, eine ihm vorgelegte Nachricht zu bestätigen oder zu verwerfen. Dazu besorgt er sich jeweils von Sender und Empfänger geheime Informationen, die in die Übertragung eingebunden werden.

## B.1 Das Verfahren

Ein kleines Beispiel ist folgendes: Die Information ist das Ergebnis eines Münzwurfes, also entweder $K$ für Kopf oder $Z$ für Zahl. Das System soll ohne Schiedsrichter arbeiten. Dann nehmen wir vier Kodierungsregeln, mit der in Abhängigkeit vom Wurf vier mögliche Nachrichten verschlüsselt werden.

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ |
|---|---|---|---|---|
| $e_1$ | $K$ | $-$ | $Z$ | $-$ |
| $e_2$ | $K$ | $-$ | $-$ | $Z$ |
| $e_3$ | $-$ | $K$ | $Z$ | $-$ |
| $e_4$ | $-$ | $K$ | $-$ | $Z$ |

Jede Regel nutzt nur zwei der vier möglichen Nachrichten und jede Nachricht kommt nur in zwei Regeln vor. Daraus ergibt sich die Wahrscheinlichkeit, daß eine zufällig gewählte Nachricht in einer bestimmten Regel enthalten ist, als die Hälfte der Wahrscheinlichkeit für die Identifizierung der Regel, wenn die Nachricht gegeben ist. Wenn der Gegner die Nachricht $m_1$ abfängt, weiß er, daß entweder Regel $e_1$ oder $e_2$ verwendet wird. Im ersten Fall

wird der Empfänger die Nachricht $m_3$ akzeptieren, $m_4$ jedoch als unauthorisiert verwerfen. Natürlich müssen sowohl Sender als auch Empfänger die gewählte Kodierungsregel kennen, um ihre Nachrichten austauschen zu können.

Wir erweitern nun das Protokoll, um die Übertragung sowohl gegen äußere als auch gegen innere Angriffe zu sichern. Wir nutzen dieselbe Information (Münzwurf) wie vorhin. Der Empfänger bildet das folgende Produkt und wählt aus der entstehenden Matrix eine Kodierungsregel.

$$\begin{pmatrix} K & K & - & - \\ K & - & K & - \\ - & K & - & K \\ - & - & K & K \end{pmatrix} \times \begin{pmatrix} Z & Z & - & - \\ Z & - & Z & - \\ - & Z & - & Z \\ - & - & Z & Z \end{pmatrix}$$

Die gewählte Zeile habe in unserem Beispiel folgende Form

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ | $m_7$ | $m_8$ |
|---|---|---|---|---|---|---|---|---|
| $\vdots$ | | | | | | | | |
| $e_i$ | $K$ | $K$ | $-$ | $-$ | $Z$ | $Z$ | $-$ | $-$ |
| $\vdots$ | | | | | | | | |

Der Empfänger schickt seine Wahl an den Schiedsrichter. Seine Arbeit ist damit beendet. Er braucht jetzt nur Nachrichten zu akzeptieren, die in der gewählten Regel auch erlaubt sind.

Der Schiedsrichter wählt im nächsten Schritt einen der vier Vektoren des Produkts

$$\begin{pmatrix} 1 & - & - & 1 \\ - & 1 & 1 & - \end{pmatrix} \times \begin{pmatrix} 1 & - & - & 1 \\ - & 1 & 1 & - \end{pmatrix}$$

und verknüpft[1] ihn mit der Regel, die er vom Empfänger bekam. Die fertige Matrix hat dann folgende Gestalt.

$$\begin{matrix} K & - & - & - & Z & - & - & - \\ K & - & - & - & - & Z & - & - \\ - & K & - & - & Z & - & - & - \\ - & K & - & - & - & Z & - & - \end{matrix}$$

Der Schiedsrichter wählt sich eine Zeile (eine Kodierungsregel) aus und übermittelt sie dem Sender. Die Vorbereitungen sind nun beendet und die Übertragung kann beginnen. Wenn der Schiedsrichter die dritte Zeile wählt, hat die Kodierungsregel folgende Gestalt

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ | $m_7$ | $m_8$ |
|---|---|---|---|---|---|---|---|---|
| $e$ | $-$ | $K$ | $-$ | $-$ | $Z$ | $-$ | $-$ | $-$ |

Sollte der Münzwurf das Ergebnis *Kopf* liefern, wird die Nachricht $m_2$ übertragen und bei *Zahl* logischerweise $m_5$.

## B.2 Betrügen

Wie groß ist nun die Wahrscheinlichkeit für eine Beeinträchtigung der Übertragung. Es werden folgende Fälle betrachtet:

[1] Er bildet das Schur-Produkt. Für zwei gegebene Vektoren $A = (a_i)$ und $B = (b_i)$ ist das Schur-Produkt der Vektor $C = (a_i, b_i)$.

**Gegner**
- *Fälschung:* Ohne Kenntnis der bisherigen Kommunikation sendet der Gegner dem Empfänger eine Nachricht. Er gewinnt, wenn sie als authentisch akzeptiert wird.
- *Substitution:* Der Gegner empfängt eine Nachricht und sendet eine andere. Er gewinnt, wenn die veränderte Nachricht akzeptiert wird.

**Empfänger**
- *Fälschung:* Der Empfänger versucht, den Schiedrichter vom Erhalt einer nichtgesendeten Nachricht zu überzeugen.
- *Substitution:* Der Empfänger versucht den Schiedsrichter nach Erhalt einer Nachricht vom Erhalt einer anderen zu überzeugen.

**Sender**
- Der Sender verleugnet eine empfangene und akzeptierte Nachricht

Der erweiterte Code wird als eine $m \times n$ Matrix $\boldsymbol{A}$ dargestellt. Sie ist ein $k$-faches kartesisches Produkt von Matrizen, deren Zeilen $r$ Elemente und Spalten $t = mr/n$ Elemente enthalten. $k$ ist die Anzahl der möglichen Informationszustände, in unserem Beispiel 2.

Dem Gegner sei der Matrix $\boldsymbol{A}$ bekannt. Dann beträgt seine Wahrscheinlichkeit für eine Fälschung

$$P_{\text{Gegner}}(\textit{Fälschung}) = \frac{kr}{kn} = \frac{r}{n} = \frac{t}{m}.$$

Im anderen Fall ergibt sich für die Generierung aus bisher bekannten Nachrichten eine Wahrscheinlichkeit von

$$P_{\text{Gegner}}(\textit{Substitution}) = \frac{(k-1)r}{(k-1)n} = \frac{r}{n} = \frac{t}{m}.$$

Die Chancen für den Gegner sind in beiden Fällen gleich.

Es kann natürlich auch der Fall auftreten, daß der Empfänger den Erhalt einer Nachricht fälschen will. Er weiß, daß der Sender nur die Nachrichten benutzen kann, die er selbst dem Schiedsrichter gegeben hat. Daraus folgt eine Wahrscheinlichkeit von

$$P_{\text{Empfänger}}(\textit{Fälschung}) = \frac{k}{kr} = P_{\text{Empfänger}}(\textit{Substitution}) = \frac{(k-1)}{(k-1)r} = \frac{1}{r}$$

für seinen Erfolg. Wenn wir eine Gleichverteilung der Chancen annehmen, setzen wir beide Wahrscheinlichkeiten (Gegner und Sender) gleich und erhalten $n = r^2$.

## B.3 Mehrfache Schiedsrichter

Bis jetzt wurde vorausgesetzt, daß der Schiedsrichter nicht betrügt, da sonst das System einfach zu umgehen ist, wenn Schiedsrichter und Betrüger zusammenarbeiten. Wenn wir mehrere Schiedsrichter haben, ist die Möglichkeit eines einzelnen zum Betrügen deutlich eingeschränkt.

Es seien $n$ Schiedrichter gegeben, von denen $t$ betrügen. Der Empfänger verteilt seine $n$ unabhängigen Regeln an alle Schiedsrichter und diese senden die bearbeitete Regel an den Sender weiter. Jede Übertragung muß von allen Schiedsrichtern beglaubigt werden. Ein Gegner kann mit den $t$ Schiedsrichtern zusammenarbeiten und die verwendeten Nachrichten in Erfahrung bringen. Er muß aber noch alle übrigen Regeln entschlüsseln, was eine Wahrscheinlichkeit von

$$P_{\text{Gegner}}(\textit{Fälschung}) = P_{\text{Gegner}}(\textit{Substitution}) = \left(\frac{1}{r}\right)^{n-t}$$

ergibt. Der Empfänger kann ebenfalls mit den Betrügern zusammenarbeiten, und so die $t$ bearbeiteten Regeln in Erfahrung bringen. Die Wahrscheinlichkeit für den Betrug mit genau $i$ Regeln beträgt

$$P_{\text{Empfänger}} = \sum_{i=t}^{n-t} \binom{n-t}{i} \left(\frac{1}{r}\right)^{i} \binom{n-t}{i}^{n-t-i} \quad .$$

Abschließend kann gesagt werden, daß das vorgestellte Verfahren einen sicheren Austausch von Nachrichten ermöglicht. Der Nachteil aber ist, daß eine größere Menge von Matrizen kartesisch multipliziert werden müssen. Zusätzlich ist die Anwesenheit eines Schiedsrichter zu akzeptieren.

Wenn diese Einschränkungen aber egal sind oder ein Schiedsrichter sogar nötig ist, dann bildet das obige Protokoll ein gutes Verfahren.

# Anhang C

# Wer die Wahl hat …
# oder
# Geheime Wahl per Telefon

**Zusammenfassung**

In diesem Abschnitt werden Probleme sowie Lösungsmöglichkeiten bei der Durchführung geheimer Wahlen diskutiert und ein mögliches Verfahren genauer vorgestellt.

Dabei handelt es sich um ein 1989 von DAVID CHAUM in [Cha89] vorgestelltes Wahlprotokoll.

Dirk Verworner

## C.1 Vorbemerkungen

Schon von Anbeginn der Zeit wurden Menschen vor Entscheidungen gestellt, deren Schwierigkeit mit der Anzahl der Entscheidungsträger zunahm, und nicht immer endeten diese Entscheidungsprozesse unblutig.

Ein Mittel zur Abhilfe sind geheime Wahlen. Bei ihnen muß gewährleistet sein, daß:

- die Stimmen der einzelnen Parteien (Wähler) geheim bleiben,
- die Wähler aber sicher sein können, daß ihre Stimme gezählt wird bzw. das Ergebnis der Wahl korrekt ist, und
- einzelne Wähler die Wahl nicht beeinflussen können.

Ein auftretendes Problem, daß uns hier nicht weiter beschäftigen wird, ist die *Registrierung* der Wähler, also der Personen, die die Möglichkeit und das Recht haben, abzustimmen. Nehmen wir also der Einfachheit halber im folgenden an, daß eine einzelne Organisation $B$ (die Behörde) bevollmächtigt ist, zu entscheiden, wer in das Wahlregister eingetragen wird und wer nicht.

Nun wird sich der geneigte Leser nach dieser Einleitung irgendwann fragen, warum wir nicht einfach Papier und Bleistift zur Hand nehmen, unseren Favoriten darauf schreiben

und den gefalteten Zettel einfach in eine *Wahlurne* werfen, deren Inhalt danach ausgezählt wird. Zum einen kann das jeder, und zum anderen tauchen vielleicht längst verstorbene Verwande wieder auf, deren Stimme natürlich den Ausschlag am Gewinn der Gegenseite gab.

Das Schlagwort heißt auch hier wieder *Authentisierung*. Unsere Behörde $B$ wird demnach auch den Auftrag erhalten, die Wahl auszurichten. Somit muß unser Verfahren für einen Wähler $A$ und die Behörde $B$ folgende Eigenschaften besitzen:

| Wähler $A$ | Behörde $B$ |
|---|---|
| • Die Stimme bleibt geheim, d.h. sie kann nur unter Mitwirkung aller Wähler bestimmt werden. | • $B$ kann Stimme eines Wählers nicht erfahren. |
| • Die Stimme wird gezählt. | • $B$ kann abgegebene Stimmen nicht verfälschen. |
| • $A$ kann die Wahl nicht stören. | • $B$ zählt richtig aus. |

Einige der sich daraus ergebenden Probleme können gleich von vornherein ausgeschlossen werden. So erhält man die unbedingte Anonymität der Stimme eines Wählers durch Einführen eines geeigneten Protokolls zur Verhinderung der Rückverfolgung von Stimmen. Dieses Protokoll wird hier nicht näher beschrieben und somit der Phantasie des Lesers überlassen. (Im Notfall kann der Wähler aus einer Telefonzelle eine von der Behörde veröffentlichte Telefonnummer anrufen, und mit verstellter Stimme eben diese abgeben.)

Ein anderes Problem ist die Störung der Wahl durch Einzelne. Dies wird im folgenden Verfahren durch Anwendung des RSA gelöst. Damit ist die Störung der Wahl durch Einzelne so *schwer*, wie das Brechen der RSA.

Desweiteren nehmen wir vorerst an, daß $B$ zuverlässig arbeitet. (Diese Annahme werden wir später fallen lassen.) Auch wenn $B$ mit unendlich viel Rechenleistung ausgestattet ist, darf er nicht erfahren, wer wie abstimmt.

Somit können wir jetzt zur Wahl schreiten und ein Wahlprotokoll besichtigen, welches die oben genannten Eigenschaften besitzt.

## C.2 Ballot Issuing Protocol

Das Herzstück des hier vorgestellten Verfahrens der Durchführung einer geheimen Wahl mit kryptographischen Mitteln ist das *ballot issuing protocol* (BIP). Es erlaubt im Wesentlichen die Stimmabgabe eines Teilnehmers $A$ mit sehr hoher Gewißheit für $B$, daß dies in einer Form erfolgte, die es nur $A$ ermöglichte, diese Stimme abzugeben.

Betrachten wir also das folgende Protokoll zwischen

**Teilnehmer A** und **Behörde B**.

---

**Schritt 1:** Zuerst stellt die Behörde $B$ alle nötigen Informationen öffentlich bereit. Dazu gehören zwei Sicherheitsparameter $z$ und $s$, ein Produkt $n$ zweier Primzahlen, eine Primzahl $p$, und $s$ verschiedene zufällige Einheiten $v_j$ des Restklassenrings modulo $n$.

$$\Leftarrow \begin{cases} z, s \in \mathbf{N} & \text{Sicherheitsparameter,} \\ n \in \mathbf{N} & \text{RSA-Modul,} \\ p > n & \text{Primzahl und} \\ v_j \in \mathbf{Z}_n & \text{zufällig, verschieden.} \end{cases}$$

$$i \in \{1, \ldots, z\}, \quad j \in \{1, \ldots, s\}$$

---

**Schritt 2:** Teilnehmer $A$ wählt sich nun privat $z \cdot s$ zufällige Einheiten $r_{ij}$ modulo $n$ und $z$ zufällige Permutationen $\pi_i$. Danach berechnet er die Matrix $M$ und sendet diese an $B$.

| | |
|---|---|
| $r_{ij} \in \mathbf{Z}_n$ | - zufällig |
| $\pi_i$ | - Permutationen |

$$\left.\begin{array}{l} M = (m_{ij}) \in M_{z,s}(\mathbf{Z}_n); \\ m_{ij} \equiv v_{\pi_i(j)} r_{ij}^p \mod n \end{array}\right\} \Rightarrow$$

---

**Schritt 3:** Darauf übermittelt $B$ eine wiederum zufällig gewählte, echte Teilmenge $C$ der Zahlen 1 bis $z$.

$$\Leftarrow \quad \emptyset \neq C \subset \{1, \ldots, z\}$$

---

**Schritt 4:** Als letzten Schritt sendet dann $A$ ein beliebiges $k$ und die Matrizen $P$ und $Q$.

$$\left.\begin{array}{l} k \in \{1, \ldots, z\} \setminus C, \\ P = (p_{ij}); \\ \quad p_{ij} = \begin{cases} \pi_i(j), & i \in C \\ \pi_k^{-1}(\pi_i(j)), & \text{sonst} \end{cases} \\ Q = (q_{ij}); \\ \quad q_{ij} = \begin{cases} r_{ij} & i \in C \\ r_{k\pi_k^{-1}(\pi_i(j))} r_{ij}^{-1}, & \text{sonst} \end{cases} \end{array}\right\} \Rightarrow$$

---

$B$ erhält nun Gewißheit, indem folgendes nachgeprüft wird:

1. Jede Zeile von $P$ ist Permutation von $\{1, \ldots, s\}$.
2. Es gilt: $m_{ij} \equiv \begin{cases} v_{p_{ij}} q_{ij}^p \mod n, & \text{falls } i \in C \\ m_{kp_{ij}} (q_{ij}^p)^{-1} \mod n, & \text{sonst} \end{cases}$.

## C.3 Die Wahl

Die eigentliche Wahl zerfällt nun in drei Phasen:

**Vorbereitung:** In der Vorbereitungsphase führt $B$ den Schritt 1 des BIP durch und veröffentlicht die dort angegebenen Infomationen. Zusätzlich gibt $B$ auch noch eine Unterteilung der $v_j$ in Äquivalenzklassen bekannt, die jede eine mögliche Wahl repäsentieren. Diese Vorbereitung geschieht aber nur einmal pro Wahl.

**Registrierung und Wahl:** Während der Registrierungsphase kommuniziert jeder Teilnehmer mit $B$. Erlaubt $B$ einem Teilnehmer $A$ die Wahl, führen sie ein BIP durch. Das Resultat des BIP ist ein Tupel von $s$ Elementen $m_{kj}$. Nun sucht sich $A$ daraus, bei Wahl von $v_{\pi_k(l)}$, das Element $m_{kl} = a$ aus und meldet dieses $B$. Das Endergebnis der Registrierung sind alle diese $a$'s, aus denen aber durch einfaches Betrachten noch keine Information über das Ergebnis der Wahl zu erhalten ist.

**Abstimmungsauszählung:** Die Abstimmungsphase beginnt mit Veröffentlichung der $p$-ten Wurzeln der $a$'s durch $B$. ($B$ kennt ja die Interna von $n$ und kann somit die $p$-ten Wurzeln modulo $n$ bestimmen.) Dann nimmt jeder Wähler die $p$-te Wurzel seines $a$, bestimmt einfach

durch Division dieser Wurzel durch das zugehörige $r_{kl}$ die $p$-te Wurzel von $v_{\pi_k(l)}$ und sendet diese *anonym*[1] an $B$. Die Anzahl der Stimmen für einen Kandidaten ist gerade die Anzahl der bei $B$ eingetroffenen $p$-ten Wurzeln der $v_j$ für diesen Kandidaten.

## C.4 Nachbetrachtungen

Die zu Anfang geforderten Eigenschaften eines Wahlprotokolls werden größtenteils durch das BIP abgefangen. So kann ein einzelner Wähler die Wahl nicht stören und seine Stimme bleibt geheim, da der RSA eingesetzt wird.

*Welcher Voraussetzungen bedarf es also noch ?*

Es ist leicht zu erkennen, daß eine der wichtigsten Voraussetzungen des Wahlprotokolls ein *sicherer* Kommunikationskanal ist. Es darf also nicht möglich sein, inkonsistente oder defekte Nachrichten zu erhalten. Dies kann durch Anbringen von digitalen Signaturen und/oder Zeitstempeln an die Nachricht erreicht werden[2].

Eine weitere Vorausetzung ist, daß jeder Teilnehmer, der sich registrieren läßt, auch an der Wahl teilnehmen muß, so daß $B$ nicht neue Stimmen generieren kann.

Zur weiteren Verhinderung eines Betruges von $B$ können die administrativen Aufgaben auch mehreren Parteien zugeteilt werden. So kann eine Partei die Generierung der zufälligen $v_j$ übernehmen, eine andere die Registratur durchführen und wieder eine andere die $a$'s unterschreiben. Desweiteren kann im Anschluß an die Wahl $B$ durch „Münzwerfen" dazu aufgefordert werden, einzelne $p$-te Wurzeln zu offenbaren.

Um die Möglichkeit, daß durch zwei Wähler zufällig die selben $v_j$ (und somit deren $p$-te Wurzeln) gewählt werden, akzeptabel klein zu machen, muß $s$ wesentlich größer als die Anzahl $m$ der Wähler sein, d.h. $s \gg m$.

Dies erfordert mindestens $s = 100m^2$, welches unpraktikabel für große $m$ ist. So bedeutet dies, bei einer Wahl mit 100 Teilnehmern die mehrfache Übertragung von Matrizen mit $1.000.000$ (*lies:* eine Million) Spalten. Durch ein von D. CHAUM und B. DEN BOER vorgeschlagenes *slot reservation*-Protokoll, das ermöglicht, daß die Wähler anonym $v_j$'s reservieren können, kann man auch $s = m$ erreichen.

Das Wahlprotokoll kann auch für andere Authentisierungsprotokolle eingesetzt werden. So steht einer Nutzung des Wahlprotokolls für Smartcards nichts im Wege. Seien die einzelnen $v_j$ die Geldwerte (z.B. 10 DM), so entspricht die Registrierung dem Abheben von einem Bankkonto und es wird mit den $p$-ten Wurzeln der $v_j$ bezahlt.

Eine Variation des Wahlprotokolls kann auch genutzt werden, um einen *Beglaubigungsmechanismus* zu implementieren. Die $v_j$ dienen dann als eindeutige Personenidentifikationen.

Abschließend kann gesagt werden, daß ich es meinem ärgsten Feind nicht wünschen möchte, dieses Wahlprotokoll je anwenden zu müssen. Der Aufwand bei entsprechender Sicherheit ist beachtlich. So sind allein auf Wählerseite $z \cdot s$ Potenzen von einer möglichst großen Primzahl (vielleicht mindestens 300-stellig) und zahlreiche arithmetische Inverse zu berechnen.

Ansonsten stellt dieses Protokoll ein gutes Beispiel für geheime Wahlen und die Probleme, die dabei auftreten können, dar und ich hoffe, mit diesem Kapitel einen Denkanstoß in diese Richtung geleistet zu haben.

[1] Hier schlägt das Protokoll zur Verhinderung der Rückverfolgung von Stimmen zu.
[2] Man denke an die Abschnitte 5.3 und 6.1.

# Anhang D

# Datenschutz

**Zusammenfassung**

Dieser Abschnitt entstand aus einem Seminarvortrag, den ich im Seminar „Mathematische Methoden der Datensicherung“ bei Dr. Gießmann gehalten habe[1].

Ich halte dieses Thema für ein von den Informatikern nur am Rande betrachtetes Gebiet. Es ist aber durchaus notwendig, daß sich auch „Beteiligte“ (und nicht nur Juristen) mit den rechtlichen Fragen des Datenschutzes auseinandersetzen und für bestimmte Probleme sensibilisiert werden. Noch trägt das gegenwärtige Datenschutzrecht zur Sicherung demokratischer Rechte bei; sorgen auch wir dafür, daß es so bleibt!

Michael Weber

## D.1 Geschichte

Wie so vieles auf dem Gebiet neuer Technologien und der Diskussion um ihre Folgen beginnt auch die „Datenschutzdiskussion“ in den USA. Bereits in den 60er Jahren lassen sich Hinweise auf die Gefahren des Computereinsatzes finden. Zu dieser Zeit wurde daran gedacht, eine „Nationale Datenbank“ (vgl. die Personenkennzahl (PKZ) der DDR) einzurichten, in der personenbezogene Daten aller Bürger zusammengeführt werden sollten und zu der einzelne Behörden jederzeit Zugang erhalten würden. Die Frage, inwieweit dadurch Individualrechte verletzt werden, löste heftige Diskussionen und Proteste aus, die letztlich dazu führten, daß das Nationale Datenzentrum nicht eingerichtet wurde. Stattdessen setzten sich in der Öffentlichkeit Forderungen nach einer gesetzlichen Verankerung des Schutzes vor Mißbrauch personenbezogener Daten durch.

Bedeutung für die Datenschutzdiskussion hatte auch das 1948 von George Orwell geschriebene Buch „1984“, in dem auf den radikalen Einsatz von Technik als Mittel der Unterdrückung, massiven Kontrolle und Gleichschaltung hingewiesen wurde. Orwells Horrorvisionen sind zweifelsohne technisch machbar und verdeutlichen umsomehr die Notwendigkeit des Widerstandes gegen jede Form der Unterdrückung mit technischen Hilfsmitteln.

Die aufgezeigten Probleme im Zusammenhang mit dem Datenschutz zeigen, daß eine Kontrolle möglich ist und auch sein muß.

[1] Ich beziehe mich im folgenden vor allem auf [Tub89].

Die Gefahren des Einsatzes von Datenverarbeitungs-(DV-)Anlagen zeigen sich beispielsweise

- im Kontextverlust durch den Einsatz von Personalinformationssystemen, die Anforderungs- und Persönlichkeitsprofile erstellen und vergleichen
- bei (automatisierten) Handschriftengutachten
- bei der Rasterfahndung, der alle zu einem Sachverhalt bereits aufgefallene (auch unbeteiligte) Personen umfaßt
- der Leistungs- und Verhaltenskontrolle in Betrieben, z. B. Protokollierung der Befehlseingabe bei ProgrammiererInnen

Selbstverständlich ist die technische Entwicklung nicht umkehrbar. Sie trägt zu einer schnellen und weltweiten Kommunikation bei; dadurch sind Informationen zu einer wichtigen Ressource geworden und tragen zu einer Produktivkraftsteigerung bei. Außerdem ist es nicht vorstellbar wie die riesige Informationsmenge schon in kleinen (eigentlich überschaubaren) Bereichen ohne Technik beherrscht werden soll.

Bereits vor der Verabschiedung des Bundesdatenschutsgesetzes 1977 gab es in der alten BRD (und der DDR) Datenschutzregelungen, wie Berufs-, Brief-, Post-, Fernmelde-, Bank-, Steuer-, Sozialgeheimnisse etc. Mit den technischen Möglichkeiten wuchs aber auch die Notwendigkeit einer lückenfüllenden Regelung. Daß dies den Mißbrauch nicht ausschließt, zeigen regelmäßige Presseberichte anschaulich.

In der DDR waren durch das Zentrale Personenregister und die PKZ der BürgerInnen Möglichkeiten zum nicht nachvollziebaren „Gebrauch“ von Personendaten durch staatliche Behörden gegeben. Hier gab es nur ein Gesetz zur Datensicherheit.

## D.2 Gegenstand des Datenschutzes

Datenschutz wird abgeleitet vom Schutz der „Persönlichkeit“ bzw. ihrer „Privatsphäre“. Personenbezogene Daten gelten also als Bestandteil einer Person.

**Definition:** Datenschutz ist Schutz der Privatsphäre vor den Gefahren der automatisierten Datenverarbeitung.

Das *allgemeine Recht auf Privatheit*, das durch die Lehre vom Persönlichkeitsrecht postuliert wurde, drückt aus, daß jedeR selbst bestimmen kann, welche Gedanken, Meinungen, Gefühle mitteilenswert sind.

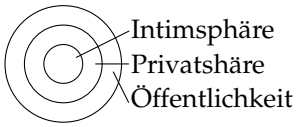
Das *allgemeine Persönlichkeitsrecht* gewann durch Art. 1 Abs. 1 GG[2] und Art. 2 Abs. 2 GG[3] zentrale Bedeutung. Davon ausgehend wurde durch eine Entscheidung des Bundesgerichtshofes (BGH) das allgemeine Persönlichkeitsrecht unter die „sonstigen Rechte“ des §823 Abs. 1 BGB[4] subsumiert.

Zur Anwendung kam dabei das sogenannte Sphärenmodell

[2] „Die Würde des Menschen ist unantastbar. Sie zu achten und zu schützen, ist Verpflichtung aller staatlichen Gewalt“

[3] „Jeder hat das Recht auf die freie Entfaltung seiner Persönlichkeit, soweit er nicht die Rechte anderer verletzt und nicht gegen die verfassungsgemäße Ordnung oder das Sittengesetz verstößt“

[4] „Wer vorsätzlich oder fahrlässig das Leben, den Körper, die Gesundheit, die Freiheit, das Eigentum oder ein sonstiges Recht eines anderen widerrechtlich verletzt, ist dem anderen zum Ersatze des daraus entstehenden Schadens verpflichtet.“




Die Intimsphäre ist dabei völlig unantastbar, während die Privatsphäre als schutzwürdig gilt.

Da allerdings jeder etwas anderes zu seiner Privatsphäre zählt, dieser Bereich obendrein von bestimmten Bedingungen abhängig ist, ist eine genaue Definition von „Privatsphäre“ nicht möglich; das Sphärenmodell ist damit nicht besonders brauchbar. Stattdessen hat jeder (durch sein Selbstverständnis begründet) das Recht auf Selbstdarstellung also ein *informationelles Selbstbestimmungsrecht*.

Dabei treten Probleme auf, wenn berechtigte Interessen miteinander kollidieren. Beispielsweise hat ein Geschäftspartner ein Interesse an der Zahlungsfähigkeit des anderen, während jener seine Kreditdaten verständlicherweise für sich behalten möchte.

Damit ergibt sich die Forderung nach der Beschränkung der Machtfülle des Staates, von Arbeitgebern Geschäftsleuten u. a., die sie durch die Möglichkeiten von DV–Anlagen haben, durch ein *Datenschutzgesetz*. Ziel solch eines Datenschutzgesetzes kann also nicht der „gläserne Mensch“, sondern muß der „gläserne Computer“ sein.

Gegenstand des Bundesdatenschutzgesetzes (BDSG) ist somit:

- Schutz des informationellen Selbstbestimmungsrechts des/der einzelnen
- Schutz der Freiheit, über die eigenen Daten grundsätzlich selbst bestimmen zu dürfen
- rechtsstattliche Absicherung dieses Anspruchs

## D.3 Bundesverfassungsgericht (BVfG) zur Volkszählung

Volkszählungen sind die vollständige und exakte Aufnahme des Bestandes und der Struktur der Bevölkerung eines Landes durch Befragen aller Haushalte und Personen. Sie gilt als Voraussetzung für eine sozialstaatliche Politik.

Hintergrund der Stellungnahme des BVfG war eine Beschwerde anläßlich der Volkszählung 1983 in der damaligen BRD. Die Beschwerde wurde zwar abgelehnt, (die Volkszählung fand einige Jahre später doch statt), jedoch wurden im Urteil des BVfG einige Grundsätze festgeschrieben, an die sich der Gesetzgeber des „Volkszählungsgesetzes 1983“ nicht gehalten hatte. So stehen die Daten, die sich aus der Beantwortung des Fragenkataloges ergeben, nach dem Urteil des BVfG für andere als statistische Zwecke nicht zur Verfügung. Damit soll verhindert werden, daß ein Abgleich mit anderen Dateien sowie die Erstellung eines Persönlichkeitsbildes nicht möglich ist. Das BVfG erkennt den Schutz des Persönlichkeitsrechtes an, hielt es aber auch für notwendig dieses Recht aufgrund von Allgemeininteressen einzuschränken.

Offenbar zeigte der massive Widerstand von Bürgerinnen und Bürgern 1983 Erfolg, denn die 1993 eigentlich anstehende Volkszählung wurde nicht durchgeführt, und auch die Diskussionen, die derzeit geführt werden, zeigen ein solches Ereignis nicht an. Stattdessen werden häufiger sogenannte Mikrozensus–Umfragen durchgeführt, die ebenfalls zur Auskunft verpflichten aber dadurch, daß nur ein geringer Teil der Bevölkerung dafür ausgewählt wird, nicht zu derartigem Widerstand veranlassen. Durch verbesserte Auswahlmethoden (also kaum fehlerträchtig) erachten die Behörden solch ein Verfahren für günstiger als eine ärgerliche Volkszählung.

## D.4 Grundsätze des Datenschutzes

### D.4.1 Öffentlicher Bereich

Wie oben angedeutet floß die oberste Rechtssprechung (des BGH und BVfG) in das BDSG ein. Dabei wurde festgeschrieben, daß für den Datenschutz im öffentlichen Bereich das Gebot der Normenklarheit erfüllt sein muß, das heißt, es muß präzise bestimmbar sein, wofür Daten gesammelt werden. Mit diesem Prinzip stehen weitere Erfordernisse im Zusammenhang:

**Normenklarheit** — wofür?

**Erforderlichkeit** Werden bestimmte Daten zur Erfüllung einer Aufgabe überhaupt benötigt?

**Zweckbestimmtheit** Es ist zu beachten, daß Daten nur für eine und nur diese Aufgabe erhoben werden dürfen.

**Übermaßverbot** Damit soll der geringstmögliche Eingriff in Persönlichkeitsrechte gesichert werden.

**Verhältnismäßigkeit** In welchem Verhältnis stehen Eingriff in Persönlichkeitsrechte und Notwendigkeit der Datenerhebung?

**Durchschaubarkeit** Wer verfügt wo über welche Daten und zu welchem Zweck?

Daraus läßt sich die Notwendigkeit einer Anonymisierung von Daten begründen, damit keine Rückschlüsse auf die Person gezogen werden können und daraus abgeleitet eine soziale oder sonstige Benachteiligung nicht möglich ist. (Die Praxis zeigt nach Meinung des Autors nur die Notwendigkeit einer strengen Handhabung des BDSG!) Auch die Datensicherheit, also die Abschottung von Daten nach außen läßt sich mit diesem Gesetz begründen.

Ein Problem ist beispielsweise das durch das Grundgesetz geforderte Gebot der Freiheit von Lehre und Forschung, das mit dem BDSG kollidiert. Damit machen sich unter Umständen weitere gesetzliche Regelungen notwendig.

Mit dem BDSG ging es dem Gesetzgeber um die Kontrolle staatlicher Macht bzw. um die Verhinderung des unkontrollierten Zusammenfließens gespeicherter personenbezogener Daten. Es soll also ein „registrierter Bürger“ verhindert werden. Einem ähnlichen Ziel dient das bekannte Prinzip der Gewaltenteilung. Damit ist es z. B. dem Sozialamt a priori nicht gestattet auf Daten des Finanzamtes zuzugreifen. Damit wird deutlich, warum es in der BRD keine PKZ gibt. Das Prinzip PKZ beruht darauf, dem Bürger Wege zu ersparen und seine Daten zentral zu speichern; alle Behörden, mit denen der einzelne nun zu tun hat, haben das Recht auf diesen Zentralspeicher zuzugreifen. Für den Bürger ist die Verwendung bzw. der Fluß seiner Daten (also einem Teil seiner Persönlichkeit) nicht mehr nachvollziehbar. (Dieses Negativargument begründet meiner Meinung nach aber nicht die derzeitige ausgeuferte Bürokratie.) In [Tät93] wird auf die bedenkliche Entwicklung eines PKZ-Ersatzes in Gestalt der Stammsatzdatei des Verbandes Deutscher Rentenversicherungsträger hingewiesen, in der ca. 84 Millionen personenbezogene Datensätze gespeichert sind. Jede öffentliche Stelle darf mithin nur das speichern, was sie zur Erfüllung ihrer Aufgaben braucht. Die Datenweitergabe ist nur unter Einhaltung der Übermittlungsvorschriften gestattet.

### D.4.2 Privater Bereich

Grundsätzlich gilt der Datenschutz für alle Bereiche des Rechtes, d. h. für den Datenschutz im privaten Bereich ist entprechendes Zivilrecht anzuwenden. Entsprechend wird im BDSG darauf verwiesen, daß es natürlichen und juristischen Personen gestattet ist, nur die Daten von Kunden, Geschäftspartnern, Arbeitnehmern etc. gespeichert und ausgewertet dürfen, die für legitime Geschäftszwecke erforderlich sind. Nicht angemessen ist es dann, Daten zu speichern, die einmal nützlich sein könnten. Ebenso ist ein ungehinderter Austausch personenbezogener Daten nicht zulässig, da Rechte Dritter verletzt werden könnten. Zum Beispiel ist ein unbeschränkter Datenaustausch zwischen Arbeitgebern unzulässig, da als unzuverlässig diskredidierten Arbeitnehmern ein „Wettbewerbsnachteil“ entstehen könnte. Der Datenschutz im privaten Bereich ist also nur dann gewährleistet, wenn die Verarbeitung und Weitergabe von Daten für die Betroffenen durchschaubar bleiben.

## D.5 Datensicherung

Die für Informatikerinnen und Informatiker wichtigste Frage im Zusammenhang mit dem Datenschutz ist natürlich die technische Absicherung und Kontrolle der Regelungen des Datenschutzrechtes. Insbesondere betrifft das die Fragen

- Zugriff zu Daten nur berechtigten Personen zu gestatten
- unerwünschter bzw. unberechtigter Verarbeitung zu verhindern
- keine Verfälschung bei der Verarbeitung zuzulassen
- Daten reproduzieren zu können

Im Sinne des BDSG ist die Datensicherung die Verhinderung des Mißbrauchs personenbezogener Daten. Der Datenschutz legt fest, *was*, die Datensicherung, *wie* geschützt werden soll.

# Anhang E

# Datenkomprimierung

Es sollen nun einige Theorien und deren mathematische Grundlagen aufgezeigt werden, womit wir unter anderem durch deren Implementierung in C-Programme eine Effizienzprüfung verschiedener Verfahren aus Sicht der Software und Hardware durchführen können. Außerdem soll gezeigt werden, welche Verfahren sich besonders für die Komprimierung welcher Arten von Daten und welcher Größe der Daten eignen.

Die besten Ergebnisse erreichen die Komprimierungsverfahren, die ganz speziell auf die konkret zu verarbeitenden Daten und deren Formate samt Dateiformate angepaßt sind. Beispiele für diese Art der Datenkomprimierung sind insbesondere in den Bereichen digitale Signalspeicherung und Audio-Kompensation zu finden. Als erstes interessieren uns die Verfahren, die auf möglichst allgemeine Daten und Datenformate anwendbar sind und hier gute Ergebnisse liefern. Diese Verfahren besitzen somit keinerlei Kenntnis über den Inhalt der zu komprimierenden Daten, sondern betrachten diese als Folge von Symbolen.

Datenkomprimierung ist eine Form der Kodierung von Daten mit dem Ziel, eine Datenmenge zu verkleinern ohne deren Informationsgehalt zu verringern. Trotz der sinkenden Preise von Speichermedien ist dies sehr nützlich, beispielsweise für die Übertragung von Daten, da der zeitliche Aufwand dafür verringert wird. Dies bringt Vorteile, wenn z.B. für die Leitungsbenutzung bei größeren Netzwerken Geld gezahlt werden muß. Auch bei der Übertragung von bewegten Bildern muß die Übertragung der Bildinformation sehr schnell (z.B. 25 Bilder/s) erfolgen. Auch dabei bietet sich eine Datenkomprimierung an, die aber neben dem Ziel der Verringerung der Datenmenge auch eine sehr schnelle Komprimierung bzw. Dekomprimierung ermöglichen muß. Die Verringerung des Zeitaufwandes kann man dabei durchaus mit der geringfügigen Einbuße von Information bezahlen, d.h. die nach Komprimieren und Dekomprimieren gewonnene Datenmenge ist gegenüber der Originaldatenmenge leicht verändert. Bei Bilddaten kann man diese unscharfe Methode oftmals anwenden. Anders ist es bei Daten, die in Datenbanken gespeichert werden, Produktionsdaten, ausführbare Programme und vieles mehr. Eine unscharfe Dekomprimierung hätte oft fatale Folgen. Im folgenden soll uns die Methode der scharfen Komprimierung beschäftigen. Es gibt drei verschiedene Arten, auf denen eine Komprimierung basieren kann:

- Endlichkeit einer Menge von gleich häufigen Symbolen
- relative Häufigkeit der Symbole
- Kontext, in dem Symbole benutzt werden

Die Symbole sind immer Bestandteil einer Menge von Symbolen. Diese Mengen können endlich oder unendlich sein. Beispiele für diese Mengen sind:

- Menge von Bits : 0, 1
- Menge von Buchstaben und Dezimalzahlen : 0, 1, …, 9, a, b, …, z, A, B, …, Z
- Menge von Farben : rot, blau, grün, gelb, …
- Menge von deutschen Wörtern : aber, bei, durch, …

Diese Beispiele lassen sich unendlich weiterführen. Schauen wir uns einmal einzelne Methoden an.

## E.1 Kodierung einer endlichen Menge von gleich häufigen Symbolen

Gegeben sei beispielsweise ein Text bestehend aus ASCII-Zeichen. Hierbei werden nur eine bestimmte Anzahl der 256 möglichen Bitkombinationen, die ein Byte beinhalten kann, verwendet. Das sind in der Regel die großen und kleinen Buchstaben, die Dezimalzahlen und einige Sonderzeichen. Man zählt alle im Text vorkommenden verschiedenen Zeichen. Diese Anzahl beträgt meist weniger als 64 Zeichen. Das heißt, man kann jedes Zeichen mit 6 statt 8 Bit verschlüsseln. Es muß lediglich eine Tabelle bereitgestellt werden, in der jedes Zeichen mit der entsprechenden Kodierung festgehalten wird. Besteht die Menge aus einer endlichen Anzahl $a$ von Symbolen, so ist jedes Symbol darstellbar mit $s$ Bits, wobei gilt $a \leq 2^s$ und daraus folgend $s \geq \log_2 a$. Voraussetzung für diese Methode ist, daß das Symbol mehr Platz benötigt als die Kodierung und der Text lang genug ist, daß die Tabelle mit den Kodierungen mit der kodierten Datenmenge zusammen immer noch eine Einsparung ergibt.

Ein einfaches Beispiel soll das verdeutlichen. Gegeben sei ein Text, der aus drei verschiedenen Symbolen, den Buchstaben a, b und c bestehen soll. Er lautet:

aaabbaaaabaccababbbc

Für die Kodierung eines der drei Symbole benötigen wir zwei Bits. Die Tabelle mit den Kodierungen müßte ungefähr so aussehen:

| Symbol | Kodierung des Symbols |
|---|---|
| a | 00 |
| b | 01 |
| c | 10 |

Der kodierte Text lautet dann als Bitfolge: 00000001010000000001001010000100010101 10

Das sind 40 Bits. Der Ursprungstext bestand aus 20 Bytes, also 160 Bits. Die Einsparung beträgt 75%. In der Praxis sind die erreichbaren Einsparungen nur ähnlich hoch, wenn die Symbole eine relativ hohe Länge und es nicht zu viele verschiedene gibt. Anwenden kann man dieses Verfahren in der Praxis z.B. bei der Verwaltung von Bücherbeständen in Bibliotheken. Für jedes Buch kann man sich ein Kürzel ausdenken, das kürzer ist als der Buchtitel. Wenn der Benutzer eine Karte hat, auf der entliehene Bücher elektronisch festgehalten werden, so ist solch eine Kodierung aufgrund des geringen Speicherplatzes der Karte sogar notwendig.

## E.2 Häufigkeitsabhängige Kodierung

Da in jedem Text bestimmte Symbole häufiger vorkommen als andere, kann man diese kürzer kodieren als solche, die seltener vorkommen. Auf diese Weise bleibt bei der Kodierung eines Textes weniger Redundanz bei den Codes. Ein Beispiel soll das verdeutlichen. Wir benutzen den gleichen Originaltext, wie bei der Kodierung gleich häufiger Symbole. Der Text lautet also:

aaabbaaaabaccababbbc

Bei der bisher angewandten Kodierungsmethode wurde jedes Symbol mit zwei Bits verschlüsselt. Da zwei Bits aber vier verschiedene Codes ergeben, enthält die Kodierung noch eine gewisse Redundanz. In der Tat wurde der Code 11 nicht benutzt. Wir stellen außerdem fest, daß die Symbole nicht mit der exakt gleichen Häufigkeit vorkommen. Der Buchstabe a hat eine Auftrittswahrscheinlichkeit von 0.5, der Buchstabe b von 0.35 und der Buchstabe c von 0.15. Aus diesen Angaben läßt sich der Informationsgehalt eines jeden Symbols ermitteln, d.h. die Anzahl von Bits, die tatsächlich zur Verschlüsselung eines Symbols nötig sind. Dieser Wert, der auch als Entropie eines Symbols bezeichnet wird, läßt sich errechnen nach der Formel

$$\textit{Entropie} = -\sum_{i=1}^{N} P_j \log_2 P_i$$

mit der Anzahl $N$ der Symbole und der Auftrittswahrscheinlichkeit $P_i$ des $i$-ten Symbols.

Wir kommen bei unserem Beispiel auf eine Entropie von ungefähr 1.44. Wir benötigen also ein Minimum von 1.44 Bits, um eines unserer Symbole zu verschlüsseln. Bei der nachfolgenden Methode wird dieses Minimum angestrebt. Betrachten wir die Tabelle mit den Kodierungen.

| Symbol | Häufigkeit des Symbols | Kodierung des Symbols |
|---|---|---|
| a | 0.5 | 0 |
| b | 0.35 | 10 |
| c | 0.15 | 11 |

Der kodierte Text lautet also als Bitfolge: 00010100000100111110010010101011

Das sind dreißig Bits, d.h. gegenüber der ersten Kodierungsmethode haben wir weitere 10 Bits eingespart. Die Anzahl der benötigten Bits pro Symbol beträgt jetzt 1.66. Diese Kodierung entspricht einem *Huffman*-Baum, der den besten Code für einzeln kodierte Symbole liefert.

### E.2.1 Tree-Codes nach Huffman und Shannon-Fano

Eine gute Annäherung an das theoretische Minimum bietet der *Huffman*-Algorithmus. Er funktioniert folgendermaßen:

**1. Schritt.** Für alle in einem Text vorkommenden verschiedenen Symbole wird die Häufigkeit ermittelt, mit der diese auftreten. Es wird ein Binärbaum aufgebaut, der die entsprechenden Symbole als Endknoten besitzt. Die Wertigkeit der Knoten besteht aus der Wahrscheinlichkeit des Auftretens der Symbole.

**2. Schritt.** Die zwei Knoten mit der kleinsten Wertigkeit werden markiert. Es wird ein neuer Knoten hinzugefügt, der mit jedem der beiden markierten Knoten verbunden wird. Der neue Knoten erhält eine Wertigkeit, die der Summe der Wertigkeiten der beiden Knoten entspricht, die mit diesem verbunden sind.

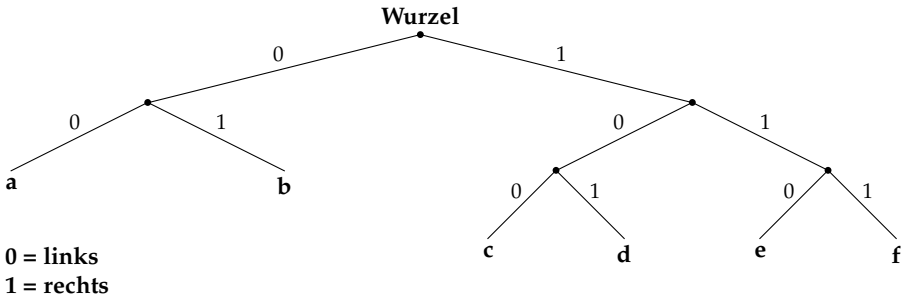
**3. Schritt.** Der Schritt 2 wird wiederholt, bis nur noch ein unmarkierter Knoten übrig bleibt. Dieser hat dann logischerweise die Wertigkeit 1.

**4. Schritt.** Der Code für ein Symbol entspricht dem Pfad des Baumes vom unmarkierten Knoten bis zu dem Endknoten, der das Symbol repräsentiert. Linke Abzweigung heißt 0, rechte Abzweigung heißt 1 (oder umgekehrt).

**Beispiel.** Es gilt die folgende Kodierungstabelle, wobei sich die Codes durch die Haltung der Zeichen $z_i$ in einem binären Baum, wie im Bild dargestellt.

| **Zeichen** $z_i$ | **Codes** | **Länge** $l_i$ |
|---|---|---|
| a | 00 | 2 |
| b | 01 | 2 |
| c | 100 | 3 |
| d | 101 | 3 |
| e | 110 | 3 |
| f | 111 | 3 |

Der Weg von der Wurzel zum Blatt ergibt den Code.




Zum Beispiel wird „acdb“ zur Bitfolge '0010010101', d.h. 10 Bit, kodiert und entsprechend '1001010100' zu „cdba“.

**Tree-Code Bedingung**

Die $k$ Zeichen $z_1, \ldots, z_k$ können genau dann durch einen Tree-Code dargestellt werden, wenn gilt:

$$\sum_{i=0}^{k} 2^{-l_i} \leq 1 \ .$$

### Optimale Kodierung

Die $k$ Zeichen $z_1, z_2, \ldots, z_k$ mögen mit einer relativen Häufigleit $p_1, p_2, \ldots, p_k$ auftreten (mit $p_i \geq 0$ und $p_1 + p_2 + \cdots + p_k = 1$). Die Codelängen mögen dabei $l_1, l_2, \ldots, l_k$ betragen. Dann ist die mittlere Codelänge definiert als

$$\sum_{i=0}^{k} p_i \cdot l_i \quad .$$

Ein optimaler Code minimiert die mittlere Codelänge unter Beachtung der Tree-Code-Bedingung. Man erhält demnach $l_i = -\log_2 p_i$. Nun müssen die Längen natürlich ganzzahlig sein, dabei sollte beim Runden aber die minimierte mittlere Codelänge nicht unnotig ansteigen.

Das Huffmannverfahren liefert die beste ganzzahlige Lösung, indem es sukzessive Teilbäume der Elemente mit niedrigster Häufigkeit aufbaut. Eine ebenfalls noch recht gute, aber viel leichter zu ermittelnde Näherungslösung erhält man, indem man die $l_i$ einfach jeweils zur nächsten Zahl nach oben aufrundet. Dieses ergibt dann den *Shannon-Fano-Code*.

Eine solche Shannon-Fano-Kodierung kann sehr einfach durch Vergabe der Codes in aufsteigender Folge gefunden werden (siehe obige Baumdarstellung). Diese Methode wird beispielsweise beim Nachcodieren im *Imploding*-Verfahren benutzt. An anderer Stelle werden wir diesem Verfahren noch ein anderes beifügen, bei dem der Computer sich bei der Vergabe der Stellen im Baum für Symbole an zuvor aufgenommene Erfahrungswerte hält.

### Statisch, dynamisch und adaptiv

Diese Art von Kodierungen, die von der Häufigkeit der auftretenden Symbole abhängig sind, lassen sich auf verschiedene Weise bestimmen. Man unterscheidet statische, dynamische und adaptive Verfahren.

**Statisch.** Bei den statischen Verfahren wird eine einmal berechnete Häufigkeitsverteilung im Kompressor als auch im Expander benutzt. Die benutzte Häufigkeitsverteilung stimmt also nicht mit der Häufigkeitsverteilung der tatsächlich übertragenen Symbole überein, da eine hinreichend genaue Vorraussagung auftretender Daten nicht möglich ist. Man erhält also niemals eine optimale Komprimierung.

Der Vorteil dieses Verfahrens liegt darin, daß es sich auf Grund seiner Konstanz relativ leicht als Hardwarelösung realisieren lassen kann, welches enorme Geschwindigkeitsvorteile mit sich bringt. Auch ist jeweils nur ein Komprimierungsdurchlauf notwendig.

Aufgrund der geschilderten Nachteile findet das Verfahren in der Praxis höchstens als Hilfsfunktion in Programmen seinen Einsatz. Beispielsweise verwendet das Programm LHARC die statische Huffman-Kodierung nicht zur eigentlichen Datenkomprimierung, sondern nur zur Nachbereitung von entstehenden Offsets und Längenangaben, die bei anderen Verfahren entstanden sind.

**Dynamisch.** Bei den dynamischen Verfahren muß die Häufigkeitsverteilung der Symbole jedesmal für die zu übertragenden Daten neu berechnet werden. Dies geschieht in einem ersten Durchlauf des Kompressors, welcher aus dem Ergebnis die entsprechende Code-Tabelle ermittelt. Im zweiten Durchlauf werden dann die Daten mit Hilfe dieser Code-Tabelle komprimiert.

Da nun für jede Datei eine andere Code-Tabelle gültig ist, muß diese dem Expander mitgeteilt werden. Dabei entsteht ein Overhead. Als Nachteile sind somit doppelter Durchlauf mit anfänglicher Neuberechnung der Codetabelle und die Übertragung dieser zu nennen. Dem gegenüber steht eine optimierte Komprimierung.

In der Praxis findet dieses Verfahren in dem Programm PKZIP Imploding zur Nachkodierung literal übertragener Bytes seinen Einsatz.

**Adaptiv.** Die Vorteile der statischen und dynamischen Verfahren bezüglich Durchlaufanzahl und optimaler Komprimierung werden durch die adaptiven Verfahren zusammengefaßt, dem ein höherer Rechenaufwand bei Kompressor und Expander entgegensteht.

Dabei wird das Kodierungsschema während des Laufes anhand der zuvor gelesenen Symbole verändert. So werden am Anfang der Datei die Symbole statisch komprimiert. Die Codetabelle ist meist so verfasst, daß die Codierung eines Symbols durchschnittlich durch 8-Bit realisiert wird. Nach einer bestimmten Anzahl eingelesener Symbole wird dann eine neue Häufigkeit ermittelt und eine neue Code-Tabelle berechnet. Der Expander expandiert die ersten Symbole mit der festen Code-Tabelle und ermittelt auch nach einer bestimmten Anzahl eine neue Codetabelle mit deren Hilfe er dann die nachsten Symbole expandiert.

Dieses kann sich innerhalb des Prozesses einer Komprimierung oder Expandierung mehrmals wiederholen, womit man der Änderung der Datenstruktur innerhalb der Datei Rechnung trägt. Dabei kann man der Codetabelle des $i$-ten Blockes entweder die Häufigkeitsverteilung der Symbole von Block 0 bis Block $i-1$ verwenden oder jeweils die des Blockes $i-1$, womit man auf sich schnell wechselnde Strukturen, wie zum Beispiel innerhalb einer Datenbank, besonders gut reagieren kann.

### E.2.2 Die arithmetische Kodierung

Diese Kodierung verschlüsselt die Symbole nicht unabhängig voneinander. Es werden auch nicht zwei Bits für die Darstellung von 1.5 Bit benötigt. Jedes Symbol wird als Teil einer reellen Zahlenreihe von 0 bis 1 dargestellt. Als Beispiel nehmen wir die Häufigkeitsverteilung unseres Beispieles:

| Symbol | Häufigkeit des Symbols |
|---|---|
| a | 0.5 |
| b | 0.35 |
| c | 0.15 |

Auf der Zahlengeraden legen wir nun Bereiche für die Symbole a, b und c entsprechend der Auftrittswahrscheinlichkeit fest. Für a legen wir das Intervall $[0, 0.5]$ fest, für b das Intervall $[0.5, 0.85]$ und für c das Intervall $[0.85, 1]$.

Wir werden jetzt mit dieser Verteilung den String „abc“ kodieren. Dazu legen wir unseren Bereich zwischen 0 und 0.5, da a das erste Symbol ist, und in diesem Bereich liegt. Wir erweitern nun diesen Bereich auf Einheitslänge und legen wieder Einteilungen für die einzelnen Symbole fest. Für a legen wir das Intervall $[0, 0.25]$ fest, für b das Intervall $[0.25, 0.425]$ und für c das Intervall $[0.425, 0.5]$. Das zweite Symbol ist ein b, unser Bereich liegt also jetzt zwischen 0.25 und 0.425. Wir bringen diesen Bereich wieder auf Einheitslänge und legen für unsere Symbole Bereiche fest. Für a legen wir das Intervall $[0.25, 0.3375]$ fest, für b das Intervall $[0.3375, 0.39875]$ und für c das Intervall $[0.39875, 0.425]$. Der dritte Buchstabe ist ein c. Das Intervall, in dem die zu übermittelnde Zahl liegt ist also $[0.39875, 0.425]$.

Für den Empfänger ist sofort klar, daß der Text mit einem a beginnen muß, da die Zahl kleiner als 0,5 ist. Natürlich ist dies nur ein einfaches Beispiel, da so wichtige Sachen, wie z.B. die Ende-Kennung vernachlässigt wurden.

## E.3 Kontextabhängige Kodierung

Im Vergleich zu den Verfahren, welche wir in vorigen Kapiteln schon vorgestellt haben, erreichen die symbolisch ausgelegten Kodierungsverfahren niemals das theoretisch mögliche Minimum. Besonders zeigt sich das bei Versuchen textbasierende Dateien zu komprimieren.

Die Einschränkungen ergeben sich hauptsächlich aus der rein byteweisen Betrachtung der Dateien, ohne möglicherweise auf vorhergehende Zeichen zu reagieren. Da beispielsweise ein n wesentlich häufiger als ein u auftritt, ist auch die Wahrscheinlichkeit, daß ein n auf ein q folgt, wesentlich größer als daß ein u auf ein q folgt.

Die Erfahrung lehrt uns, das das unsinnig ist. Bestimmte Zeichenfolgen tauchen auch sehr häufig auf, beispielsweise die Zeichen „en“ am Ende eines Wortes. Dies kann jedoch nur beachtet werden wenn man sogenannte Kontextmodelle aufbaut, die bis zu mehreren 100 000 Zeichen lang sein können. Man betrachtet jetzt neben Symbolen also auch auftretende Zeichenfolgen. Aufgrund der aufwendigen Berechnungen und des erhöhten Speicherplatzverbrauches wird diese Herangehensweise jedoch in der PC Technik sehr selten benutzt.

In der Praxis findet man eine recht einfache kontextabhängige Kodierung innerhalb des PKZIP Programmes im Reducing Verfahren. Im folgenden werden einige Beispiele für kontextabhängige Kodierungen beschrieben.

### E.3.1 Der Telegraphencode von Baudot

Nimmt man als Symbole Buchstaben, so muß für jeden Buchstaben eine Liste mit der Häufigkeit eines möglichen Nachfolgers auf diesen Buchstaben existieren. Dies führt zu einer großen Einsparung. Es müssen jedoch auch entsprechend viele Listen existieren, da neben den 26 Buchstaben einige Sonderzeichen, die Dezimalzahlen und noch einmal 26 Buchstaben (wegen Groß- und Kleinschreibung) in Texten vorkommen. Für jedes dieser Zeichen müßte eine Liste existieren mit den Häufigkeiten der möglichen Nachfolger. Für $k$ Symbole existieren $k$ Listen mit jeweils $k$ Einträgen.

Der Telegraphencode von Baudot ist ein 5-Bit Code. Damit können 32 Symbole verschlüsselt werden. Diese sind die 26 Buchstaben, zwei Symbole für »Leerzeichen« und »neue Zeile« und vier Umschaltsymbole.

Es wird davon ausgegangen, daß nach einem Kleinbuchstaben wieder ein Kleinbuchstabe folgt, nach einer Zahl wieder eine Zahl und so weiter. Das gilt solange, bis das nächste Umschaltzeichen kommt. Die Umschaltsymbole decken folgende vier Fälle ab: Großschreibung, Kleinschreibung, numerische und besondere Angaben sowie Steuerzeichen. Je nach gesetztem Umschaltzeichen werden die nachfolgenden Symbole interpretiert.

Damit können in Wirklichkeit $4 \cdot 28 = 112$ Symbole kodiert werden. Der Baudotcode hat immer die gleiche Länge, was algorithmisch einfacher zu handhaben ist und bei der Datenübertragung nicht so fehleranfällig ist.

### E.3.2 Die Lauflängenkodierung (run-length encoding)

Kommen lange Folgen von Symbolen (Läufe) in einem Text vor, so besteht eine hohe Redundanz. Beispielsweise kann jede Folge sich mehrfach wiederholender Symbole ersetzt werden durch eine Anzahl und das Symbol. In unserem bekannten Beispiel:

aaabbaaaabaccababbbc

würde das so aussehen:

3a2b4abaccaba3bc

Aus 20 Byte sind 16 geworden. Dieses primitive Beispiel läßt sich in viele verschiedene Varianten verfeinern, die sehr gut für Dateien mit langen Folgen von Symbolen geeignet sind (z.B. Bilddaten). Bei Bitketten, die lange Folgen von Nullen enthalten, kann man zum Beispiel alle Anzahlen von Nullen kodieren, die zwischen zwei Einsen auftreten. Im folgenden Beispiel werden die Anzahlen durch jeweils drei Bits dargestellt.

| | | | |
|---|---|---|---|
| Original: | 00001001100000000010000000010001 | → | 30 Bits |
| zu kodieren: | 4 2 0 8 7 3 | | |
| kodierte Bitkette: | 001 010 000 111 100 111 000 110 | → | 24 Bits |
| Einsparung: | 20% | | |

### E.3.3 Nachfolger-Kodierung innerhalb der Reducing Verfahres

Dieses Verfahren stellt den wahrscheinlich einfachsten Gebrauch eines kontextabhängigen Kodierungsverfahrens dar, wobei der betrachtete Kontext nur aus dem jeweils nachfolgenden Zeichen besteht.

Für jedes Byte $x$ bestimmt der Kompressor in seinem ersten Pass eine Liste der 1, 2, 4, 8, 16 oder 32 günstigsten Nachfolger. Alle diese Listen $L(x)$ werden im Kopf der Datei dem Expander mitgeteilt. Im zweiten Pass des Kompressors erfolgt dann die eigentliche Komprimierung nach der im folgenden beschriebenen Art.

Sei $x$ das zuletzt kodierte Zeichen und $y$ das nächste Byte (Ausgangsstellung).

Ist $y$ aus $S(x)$, so wird zunächst ein Bit 0 als Kennzeichen gesendet und anschließend die Position von $y$ innerhalb der Liste $S(x)$, je nach der Länge der Liste werden dazu 1 bis 5 Bit benötigt. Weiter mit 4.

Ist $y$ nicht aus $S(X)$, so wird ein Bit 1 als Kennzeichen gesendet und anschließend $y$ als Literal (8-Bit Wert). $y$ übernimmt die Rolle von $x$ und weiter bei 1.

## E.4 LZW - Datenkomprimierung

Ein völlig anderer Weg der Datenkomprimierung wurde 1977 von LIV und LEMAN vorgestellt und schließlich von WELCH im Jahre 1984 das erstemal technisch realisiert. Das Verfahren wird auch oftmals nach seinen Entdeckern *LZW*-Verfahren genannt.

Dieses völlig andere Verfahren nimmt an, daß die Redundanzen sich in ständig wiederholenden Byte-Ketten wiederspiegeln. Diese sich mehr oder weniger im Eingabestrom wiederholenden Symbolketten werden als Phrasen bezeichnet. Die Komprimierung geschieht dann dadurch, daß diese Phrasen nicht mehrfach gesendet werden, sondern sich

im Ausgabestrom als Rückbezug offenbaren. Ein Rückbezug besteht aus einem Offset- und einem Längenteil.

Wenn ein solcher Rückbezug auftritt, soll dieser nun folgendermaßen interpretiert werden: „Gehe *Offset* Einheiten zurück und kopiere an die derzeit aktuelle Stelle *Länge* Einheiten."

Wenn es nicht möglich ist die Daten als Rückbezug zu übertragen, so werden sie im Original, also literal gesendet. Dieser in den siebziger Jahren von Liv und Lemon entdeckte Weg der Datenkomprimierung hat sich heute weitgehend durchgesetzt und man findet ihn in vielen professionellen Programmen implementiert. Trotzdem gibt es bei der Implementierung teilweise sehr große Unterschiede. Diese Unterschiede resultieren aus der unterschiedlichen Lösung zweier Aufgaben, die uns als Unterscheidungsmerkmal von unterschiedlichen Herangehensweisen innerhalb der LZW-Komprimierungsverfahren dienen sollen.

**Merkmal:**

- Behandelt die Unterscheidungsmöglichkeit von Literal oder als Rückbezug gesendeten Daten. Schrittweise entwickelten sich verschiedene Lösungen:

- Jedem Rückbezug folgt genau ein Literal. Im Falle des Auftretens von mehreren Literalen hintereinander müssen diese durch Null-Rückbezüge getrennt werden. Hierbei handelt es sich um eines der Originalverfahren, die von LIV und LEMON aufgezeigt worden sind. Die Nachteile dieses Verfahrens sind sofort sichtbar und auch in der Praxis wird dieses Verfahren kaum angewendet.

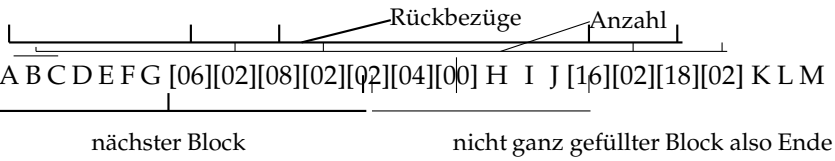
- Rückbezüge und Literale dürfen beliebig wechseln, ihre Unterscheidung trifft ein einzelnes Bit bzw. ein vorher definierter Wertebereich. Wenn man also alle Werte als 12 Bit Wert überträgt, so kann man vorher definieren, daß Werte, die kleiner als 256 sind, als literale Bytes zu interpretieren sind. Diese Verfahren wurden als erstes von STORNER und SZYMANSKI vorgestellt.

- Man erlaubt auch hier verschieden lange literale Byteketten, gibt jedoch vor dem Senden einer solchen ihre Länge an. Dieses Verfahren ist eines der gebräuchlichsten.

- Am Anfang der übertragenen Daten kommt ein Block (mit vorher übertragener Länge), in dem die Positionen der sich im Datenblock befindenen Rückbezüge festgehalten werden, dadurch kann auch hier die Aufeinanderfolge von Rückbezügen und Literal beliebig wechseln. Diese Herangehensweise ist den Autoren bei dem umfangreichen Literaturstudium nicht aufgefallen, so daß es wahrscheinlich als ein neues Verfahren (HeRet) angesehen werden kann. Dieses Verfahren birgt einige Vorteile gegenüber den anderen. So wird per Rückbezug der Speicherplatz von 8 Byte verwendet, wenn Rückbezüge sich über die ganze Datei erstrecken können und die Länge der Blöcke sich auf ffffh begrenzt. Die maximale Lange der Datei wird auf ffffffh festgelegt. Legt man fest, daß die Blöcke nur ffh Byte lang sein dürfen, und die Rückbezüge nur ffffh lang sein dürfen, so verringert sich der Speicherplatzaufwand auf 6 Byte bei einer Dateilange von ffffffh. Das Problem des zweifachen Durchlaufes, der bei diesem Verfahren notwendig wird, da die Inhalte des am Anfang zu sendenden Blockes zu Beginn des Komprimierens noch nicht bekannt sind, wird dadurch umgangen, daß als letzter Eintrag in dem Block ein Verweis auf den nächsten Block zu finden ist. Verweise auf Rückbezüge die den Inhalt Null haben, werden als eine nicht belegte Stelle im Rückbezugblock interpretiert. Soll die Datei

  ABCDEFGABCABCHIJABCABCKLM

nach dem LZW-Verfahren mit der Herangehensweise HeRet komprimiert werden, ergibt sich folgende Ansicht:




Die Länge des Rückbezuges sei 3. Das Beispiel ist willkürlich gewählt und ohne Realitätsbezug. In einem zweiten Pass läßt sich die Anzahl der Blöcke auf einen Anfangsblock reduzieren.

**2. Merkmal:** Dieses Merkmal behandelt die Frage, auf wieviele Phrasen innerhalb des Komprimierungslaufes überhaupt Bezug genommen werden kann. Das Problem liegt dabei in der Zeit, die nötig ist, um eine einzelne Phrase aufzusuchen.

Im weiteren Verlauf werden wir verschiedene Möglichkeiten der Phrasen oder auch Stringsuche kennenlernen, wobei zur Grundlage die nächsten Unterkapitel gelesen werden sollten.

Die Kompensationsrate ist unter anderem auch davon abhängig, wie effizient die einzelne Bestandteile der komprimierten Datei (literale Zeichen, Offset-Werte und Längenangaben) kodiert sind, deshalb sollen an dieser Stelle einige Verfahren vorgestellt werden, die später realisiert und auf ihre Praxistauglichkeit überprüft, optimiert und getestet werden sollen.

### E.4.1 Kodierung von Rückbezügen

Ein Beispiel ist die Kodierung von Rückbezügen. So muß man bei kurzen Rückbezügen nicht immer ganze 2 Byte benutzen, sondern kann auf das von FIALLA und GREENE vorgeschlagene *start-step-stop*-Verfahren zurückgreifen.

Da es sich prinzipiell als ungünstig erwies, feste Formate für die bei Rückbezügen nötigen Längen- und Offset-Werte einzusetzen, setzt man hier verschiedene Bezeichner für verschiedene Wertebereiche ein. In der nachfolgenden Tabelle sind zum Zwecke der Demonstration solche Bezeichner für Wertebereiche aufgelistet.

| Länge | Code |
|---|---|
| … 3 | 0xx |
| … 11 | 10xxx |
| … 27 | 110xxxx |
| … 59 | 1110xxxxx |
| … 123 | 11110xxxxxx |
| … 251 | 111110xxxxxxx |
| … 507 | 1111110xxxxxxxx |
| … 1019 | 11111110xxxxxxxxx |
| … 2043 | 111111110xxxxxxxxxx |

Hierbei gilt, daß die mit „x“ gekennzeichneten Bits die Länge oder die Position binär kodieren.

Dieses Verfahren kann im Prinzip überall da eingesetzt werden, wo Werte, die einen unterschiedlich langen Platz beanspruchen, auftreten.

Eine Kritik sei jedoch jetzt schon angebracht: Natürlich bedeutet eine solche Kodierung eine wesentliche Verlängerung der Berechnung der einzelnen Werte, da hier mit „krummen“ Werten (17 Bit) gerechnet werden muß, was sehr viele Schiebebefehle und Positionsberechnungen zur Folge hat. Ob die geringe Platzersparnis das rechtfertigt, ist fraglich.

# Literaturverzeichnis


[adv] *Advances in Cryptology – EUROCRYPT*nn *und CRYPTO*nn. *Conferences on Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, Berlin. verschiedene Bände.

[Bau93] Friedrich L. Bauer. *Kryptologie : Methoden und Maximen*. Springer-Verlag Heidelberg New York, 1993.

[Beu91] Albrecht Beutelspacher. *Kryptologie*. Vieweg, Braunschweig, 1991. ISBN 3-528-18990-8.

[Bra90] G. Brassard. *Modern Cryptology*, volume 325 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1990. ISBN 3-540-96842-3 (enthält 250 weitere Literaturstellen).

[Cha89] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa. In *Advances in Cryptology: Proceedings of CRYPTO 89*, Lecture Notes in Computer Science, pages 177–182, Berlin, 1989.

[Cur92] D. A. Curry. *UNIX System Security: A Guide for Users and System Administrators*. Addison-Wesley, New York, 1992. ISBN 0-201-56327-4.

[fFugVFeV91] Forum InformatikerInnen für Frieden und gesellschaftliche Verantwortung (F!FF) e. V. *Das Datendschungelbuch*. F!FF, Reuterstr. 44, 53 113 Bonn, 1991.

[FR88] W. Fumy and H. P. Rieß. *Kryptographie: Entwurf und Analyse*. Oldenbourg, München, 1988. ISBN 3-486-20868-3.

[Hor85] Patrick Horster. *Kryptologie*, volume 47 of *Reihe Informatik*. Bibliographisches Institut, Mannheim - Wien - Zürich, 1985. ISBN 3-411-03106-9.

[Kah67] D. Kahn. *The Codebreakers. The Story of Secret Writing*. Macmillan Publishing Co., New York, 1967.

[Kob87] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, Berlin, 1987. ISBN 3-540-96576-9.

[Kra86] E. Kranakis. *Primality and Cryptography*. Wiley-Teubner Series in Computer Science. Wiley-Teubner, London, 1986.

[Lan89] Serge Lang. *Faszination Mathematik: Ein Wissenschaftler stellt sich der Öffentlichkeit*. Teubner-Verlag, Leipzig, 1989. ISBN 3-322-00475-9.

[Orw48] George Orwell. *1984*. London, 1948.

[Tät93] *14. Tätigkeitsbericht des Bundesbeauftragten für den Datenschutz*, Bonn, 1993. (Bundestags–Drucksache 12/4805).

[Tub89] Marie-Theres Tinnefeld / Helga Tubies. *Datenschutzrecht*. Oldenbourg Verlag, München, 1989.