# Kryptologie

Vorlesungsskript

Dietmar Wätjen

Institut für Theoretische Informatik

Technische Universität Braunschweig

Oktober 1998

# Vorwort

Kryptographie und Kryptologie stellen, wie man vielleicht anschaulich sagen kann, die Lehre vom geheimen Schreiben dar. Eine genauere Begriffsbestimmung werden wir später vornehmen. Die Kryptographie hat schon seit langem die Menschen interessiert. Erste Anfänge gab es bereits in Ägypten und Mesopotamien. Damals wurde Lesen und Schreiben ohnehin nur von wenigen beherrscht, und so war die Kryptographie zunächst wohl mehr eine Spielerei, um z.B. irgendwelche Aussagen okkulten Inhalts vor den anderen Lesern zu verbergen und damit als noch wichtiger erscheinen zu lassen. Aber bald fand auch schon aus geschäftlichem Interesse eine Anwendung der Kryptographie statt. In Mesopotamien hat man z.B. auf einer kleinen Schreibtafel eine verschlüsselte Anweisung entdeckt, wie man die Glasur von Töpfen macht.

Auch von Staaten wurde das geheime Schreiben früh benutzt, vor allem für diplomatische und militärische Zwecke. Aus Griechenland und Rom sind dazu einige Beispiele bekannt. Nach dem Zerfall des Römischen Reichs wird in Europa erst seit der Renaissance-Zeit die Kryptologie weiter entwickelt. Es geht dabei immer um das Verschlüsseln von Nachrichten und ihre Entschlüsselung, die in ständigem Wettstreit liegen. Wir wollen hier nicht weiter auf die Geschichte der Kryptographie eingehen, sondern auf das sehr interessante Buch von *David Kahn* [43] verweisen, in der sie, unter anderem auch im Hinblick auf die beiden Weltkriege, sehr ausführlich dargestellt ist.

Die Kryptographie wurde in der Vergangenheit hauptsächlich zu militärischen, geheimdienstlichen und diplomatischen Zwecken verwendet. Heute kommt ihr durch die weite Verbreitung von Computern und ihre Vernetzung eine neue Bedeutung zu. Ein Computersystem hat viele Benutzer. Es ist aber verständlich, daß man seine persönlichen Daten nicht allen Benutzern zugänglich machen will. So ist zunächst einmal ein Mechanismus zur Autorisierung nötig, etwa durch ein Paßwort. Für Leute, die sich im System auskennen, dürfte es allerdings nicht schwer sein, Paßwörter zu umgehen. Durch eine schwer zu brechende Verschlüsselungstechnik wird jedoch Schutz gegenüber unautorisierten Benutzern gewährleistet.

Ein weiteres Problem ergibt sich, wenn man einem anderen Benutzer einige private Daten übermitteln will, die kein anderer, also auch kein elektronischer Lauscher, erfahren soll. Dabei will der Empfänger auch sicher sein, daß die Daten von dem behaupteten Absender stammen. Gerade bei diesen Kommunikationsproblemen ist der Bedarf an Kryptographie enorm gestiegen.

Im Bankwesen und ganz allgemein im Geschäftswesen gibt es ein weites Feld von Anwendungen der Kryptographie. Jeder hat schon bei einem Geldautomaten Geld von seinem Konto abgehoben. Wie kann man sicher sein, daß der richtige Betrag abgebucht wird und keine anderen Personen das Konto plündern? In der letzten Zeit sind doch einige Mißbräuche bekannt geworden, die deutlich machen, daß hier die kryptographischen Sicherheitsanforderungen noch erhöht werden müssen.

Verwunderlich ist allerdings auch nicht, daß die Verschlüsselung der in den Computernetzen versendeten Daten manchen staatlichen Stellen, insbesondere Polizei und Geheimdiensten, nicht gefällt. Verbrecherische Organisationen bedienen sich leider sehr viel schneller technologisch fortschrittlichen Methoden, als es die häufig sehr schwerfälligen staatlichen Organisationen tun. In den USA ist es z.B. nur erlaubt, ein Verschlüsselungssystem zu

benutzen, das von der Regierung genehmigt ist und von dem man annehmen darf, daß die National Security Agency mitlesen kann. Auch in der Bundesrepublik Deutschland gibt es Versuche in dieser Richtung, die Benutzung der Kryptologie einzuschränken. Da andererseits in allen Computernetzen Krypto-Systeme als freie Software erhältlich sind, ist es kaum anzunehmen, daß die staatlichen Stellen sich mit ihren restriktiven Forderungen durchsetzen können. Selbst bei einem Verbot der Verwendung sicherer Schlüssel (d.h. genügend langer Schlüssel) gibt es Verfahren, wie man trotzdem Sicherheit erreichen kann. Die bekannteste solche Methode ist die Steganographie, bei der die geheime Information in digitalen Bildern versteckt wird. Wir werden hier auf diese Technik nicht näher eingehen. Eine erste Einführung dazu findet sich z.B. in [24].

Wir wollen in der Vorlesung nach einführenden Begriffen erst einige ältere Verfahren der Kryptographie besprechen, um dann später auf neuere Verfahren einzugehen. Besonders interessant sind in diesem Zusammenhang die Public-Key-Kryptosysteme, bei denen jeder Benutzer einen Verschlüsselungsalgorithmus öffentlich bekanntmacht, mit dem ihm jeder andere Teilnehmer Nachrichten schicken kann. Nur der Benutzer selbst kann jedoch diese Nachrichten in angemessener Zeit entschlüsseln.

Dieses Skript ist aus einer Vorlesung entstanden, die ich in den vergangenen Jahren als zweistündige Vorlesung im Wintersemester gehalten habe. Als weitere Lehrbücher zur Kryptologie möchte ich die Bücher von *Dorothy E. Denning* [15], *Arto Salomaa* [83], *Patrick Horster* [42], *Fumy/Rieß* [27] und *Stinson* [99] erwähnen. Das Buch von *Wobst* [103] gibt in leicht lesbarer Form eine Einführung in viele Bereiche der Kryptologie und in ihre Anwendungen. Eine umfangreiche Referenz auf fast alle Themen der Kryptologie liefert das Buch von *Bruce Schneier* [86]. Darüberhinaus sind im Literaturverzeichnis viele Originalarbeiten angegeben, die zur Vertiefung der im Skript besprochenen oder teilweise nur angeschnittenen Probleme dienen können. In dem von *Gustavus J. Simmons* herausgegebenen Sammelband [97] befinden sich übersichtliche Beiträge zu der Entwicklung der Kryptologie der letzten Jahre.

Sicher sind im Skript noch Fehler vorhanden. Ich würde mich über entsprechende Hinweise freuen.

Braunschweig, im Oktober 1998 Dietmar Wätjen

# Inhaltsverzeichnis


**1 Grundlagen** **1**
- 1.1 Erste Grundbegriffe . . . 1
- 1.2 Kryptographische Systeme . . . 2
- 1.3 Informationstheoretische Grundlagen . . . 4

**2 Klassische kryptographische Verfahren** **15**
- 2.1 Transpositionschiffren . . . 15
- 2.2 Chiffren mit einfacher Substitution . . . 16
- 2.3 Chiffren mit homophoner Substitution . . . 21
- 2.4 Polyalphabetische Substitution . . . 22
- 2.5 Chiffren mit Polygramm-Substitution . . . 32
- 2.6 Produktchiffren und der DES . . . 34

**3 Zahlentheoretische Grundlagen** **45**

**4 Das RSA-Verfahren** **57**
- 4.1 Public-Key-Kryptosysteme . . . 57
- 4.2 Exponentiationschiffren . . . 60

**5 Kryptographische Protokolle** **71**
- 5.1 Mental Poker (Kopf-Poker) . . . 71
- 5.2 Vergeßliche Übertragung . . . 75
- 5.3 Schlüsselverwaltung . . . 88
- 5.4 Protokoll zum Altersvergleich . . . 93
- 5.5 Protokolle für Auktionen und Geschäfte . . . 95
- 5.6 Ein Protokoll für Wahlen . . . 98

**6 Ein Protokoll für Kreditkartenzahlungen** **105**
- 6.1 Hashfunktionen, digitale Signaturen und duale Signaturen . . . 105
- 6.2 Zertifikate . . . 107
- 6.3 SET-Protokolle . . . 108

**7 Zero-Knowledge-Protokolle** **115**
- 7.1 Komplexitätstheorie . . . 115
- 7.2 Zero-Knowledge-Beweise . . . 116

**8 Knapsacksysteme** **125**
- 8.1 Merkle-Hellman-Knapsacks . . . 125
- 8.2 Brechen des Merkle-Hellman-Knapsacks . . . 128
- 8.3 Shamir-Knapsacks für Authentizität . . . 131
- 8.4 Ein brechbarer *NP*-vollständiger Knapsack . . . 134

**9 Ein weiteres Kryptosystem und Ergänzungen** **137**
9.1 Ein Kryptosystem mit formalen Sprachen . . . . . 137
9.2 Blockchiffren . . . . . 140

**Anhang: Häufigkeitstabellen** **147**

**Literaturverzeichnis** **159**

**Index** **166**

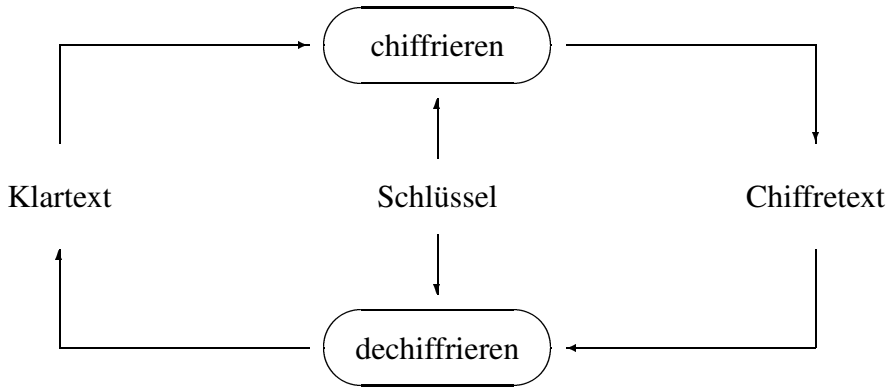
# 1 Grundlagen

## 1.1 Erste Grundbegriffe

Zunächst sollen einige grundlegende Begriffe der Kryptologie aufgeführt und kurz erläutert werden.

- *Kryptographie:* Wissenschaft vom geheimen Schreiben
- *Chiffre (cipher):* eine geheime Methode des Schreibens (Methode des Verschlüsselns)
- *Klartext (plaintext, cleartext):* unverschlüsselter Text
- *Chiffretext (ciphertext, cryptotext, cryptogram):* verschlüsselter Text
- *Chiffrieren:* verschlüsseln (to encode, to encipher, to encrypt)
- *Dechiffrieren:* entschlüsseln (to decode, to decipher, to decrypt)
- *Schlüssel (key):* kontrolliert die Ver- und Entschlüsselung, er ist der Informationsträger für die Verschlüsselung des Klartextes bzw. Entschlüsselung des Chiffretextes

Die Zusammenhänge werden durch das folgende Bild verdeutlicht.




Es gibt zwei prinzipielle Arten des Chiffrierens, und zwar Transposition und Substitution. Eine *Transpositionschiffre* stellt die Bits oder Zeichen des Textes um, eine *Substitutionschiffre* ersetzt Bits, Zeichen oder Blöcke von Zeichen durch andere Symbole.

**Beispiel 1.1.1** Wir betrachten als Beispiel einer Transpositionschiffre einen „Zaun“ der Tiefe 3. Der Klartext lautet DIESISTEINEGUTEVORLESUNG. Das Chiffrierverfahren ist durch die Art des Zauns und dessen Tiefe gegeben, nämlich

```
D   I   I   U   O   S
  I S S E N G T V R E U G
    E   T   E   E   L   N .
```

Daraus ergibt sich der Chiffretext DIIUOSISSENGTVREUGETEELN. □

**Beispiel 1.1.2** Wir betrachten die *Cäsar-Chiffre*, die eine Substitutionschiffre ist. Jeder Buchstabe des Alphabets wird dabei um $k$, $0 \leq k \leq 25$, Plätze verschoben (Verschiebechiffre). Mit $k = 3$ (dieser Fall wurde von Cäsar benutzt) wird der Klartext aus Beispiel 1.1.1 zu GLHVLVWHLQHJXWHYRUOHVXQJ. □

Die *Kryptoanalysis* ist die Wissenschaft des „Brechens“ von Chiffren. Eine Chiffre ist dann zu brechen, wenn der Klartext bzw. der Schlüssel vom Chiffretext oder der Schlüssel von Klartext-Chiffretext-Paaren bestimmt werden kann. Der *Kryptoanalytiker*, der z.B. ein elektronischer Lauscher sein kann, kennt i. allg. das Chiffrierverfahren, etwa den Zaun aus Beispiel 1.1.1, aber nicht den Schlüssel dazu, d.h. die Tiefe des Zauns. Es ist nicht abwegig anzunehmen, daß der Kryptoanalytiker das Chiffrierverfahren kennt. Häufig bedingt ein bestimmtes Informationssystem auch ein bestimmtes Chiffrierverfahren.

Es gibt drei typische Anfangssituationen, denen ein Kryptoanalytiker gegenübersteht, und aus denen er seine Schlüsse ziehen muß:

(a) *nur Chiffretext* (nur-Chiffretext-Angriff): Der Kryptoanalytiker kann den Klartext nur aus dem abgefangenen Chiffretext bestimmen. Bei einem Chiffretext, der z.B. den Weg zu einem versteckten Schatz beschreibt, sind Wörter wie „Schatz“, „vergraben“, „südlich“, usw. zu erwarten. Dies geht über zum nächsten Fall.

(b) *bekannter Klartext*: Die Kenntnis von Klartext-Chiffretext-Paaren kann zum Entschlüsseln des ganzen Textes hilfreich sein. So gibt es etwa bei Briefen feststehende Anfangs- und Schlußformeln. Bei verschlüsselten Programmen kann der Kryptoanalytiker eventuell Programmsymbole wie **begin**, **end** sofort erkennen.

(c) *gewählter Klartext*: Er erhält den Chiffretext zu von ihm selbst gewähltem Klartext. Datenbanksysteme sind gegenüber diesen Versuchen anfällig, da ein Benutzer etwas in die Datenbank einfügen und dann beobachten kann, wie sich der gespeicherte Chiffretext ändert.

*Public-Key-Kryptosysteme* erlauben noch andere Vorgehensweisen des Kryptoanalytikers, auf die wir später eingehen werden. Kryptoanalyse wird natürlich auch benutzt, um zu zeigen, daß gewisse Chiffrierverfahren „sicher“ sind. Dabei gilt eine Chiffre als *uneingeschränkt sicher*, wenn unabhängig von der Menge des empfangenen Chiffretextes dieser nicht ausreichend Informationen liefert, um den Klartext eindeutig zu bestimmen, und als *berechnungs-sicher*, wenn sie nicht mit „tatsächlichen“ Zeit- und Rechnermöglichkeiten gebrochen werden kann.

Die Kryptologie umfaßt nun genau die Kryptographie und die Kryptoanalysis.

## 1.2 Kryptographische Systeme

Ein *kryptographisches System* (kurz *Kryptosystem*) besteht aus fünf Komponenten, und zwar

1. einem *Klartextraum* $\mathcal{M}$,
2. einem *Chiffretextraum* $\mathcal{C}$,
3. einem *Schlüsselraum* $\mathcal{K}$,
4. einer Familie von *Chiffriertransformationen* $E_K : \mathcal{M} \longrightarrow \mathcal{C}$ mit $K \in \mathcal{K}$ und
5. einer Familie von *Dechiffriertransformationen* $D_K : \mathcal{C} \longrightarrow \mathcal{M}$ und $K \in \mathcal{K}$.

Dabei sind $\mathcal{M}, \mathcal{C}, \mathcal{K}$ höchstens abzählbar. Jede Chiffriertransformation $E_K$ wird durch einen Schlüssel $K$ und einen Chiffrieralgorithmus $E$ definiert, der für jede Transformation der

Familie gleich ist. Entsprechendes gilt für jede Dechiffriertransformation $D_K$. Für ein gegebenes $K$ und für alle $M \in \mathcal{M}$ gilt

$$D_K(E_K(M)) = M.$$

Kryptosysteme müssen auf jeden Fall drei Eigenschaften erfüllen:

(1) Chiffrier- und Dechiffriertransformationen müssen für alle Schlüssel effizient berechnet werden können.
(2) Die Systeme müssen leicht zu benutzen sein. Es muß also leicht sein, einen Schlüssel sowie die Abbildungen $E_K$ und $D_K$ zu finden.
(3) Die Sicherheit des Systems sollte auf der Geheimhaltung der Schlüssel und nicht auf der Geheimhaltung der Algorithmen beruhen. Das bedeutet, daß aufgrund der Kenntnis der Methode des Chiffrierens und Dechiffrierens noch nicht der Klartext zu gewinnen ist.

Für Geheimhaltung und Authentizität gibt es spezielle Forderungen. Geheimhaltung verlangt, daß ein Kryptoanalytiker nicht in der Lage ist, Klartext aus abgefangenem Chiffretext zu bestimmen. Authentizität erfordert, daß der Kryptoanalytiker ohne Entdeckung keinen falschen Chiffretext $C'$ für einen Chiffretext $C$ einsetzen kann. Wir formulieren zunächst die

*Geheimhaltungsanforderungen:*
Es sollte einem Kryptoanalytiker berechnungsmäßig praktisch unmöglich sein,

(1) systematisch $D_K$ aus abgefangenem Chiffretext $C$ zu bestimmen, selbst dann, wenn der Klartext $M$ mit $E_K(M) = C$ bekannt ist, und
(2) den Klartext $M$ aus $C$ zu bestimmen.

Für die Geheimhaltung muß nur $D_K$ geschützt werden, sofern $E_K$ nicht $D_K$ verrät. $E_K$ kann also publik gemacht werden. Weiter erhalten wir die

*Authentizitätsanforderungen:*
Es sollte einem Kryptoanalytiker berechnungsmäßig praktisch unmöglich sein,

(1) systematisch $E_K$ aus $C$ zu bestimmen, selbst dann, wenn der Klartext $M$ mit $E_K(M) = C$ bekannt ist, oder
(2) einen Chiffretext $C'$ zu finden, so daß $D_K(C')$ ein gültiger Klartext aus $\mathcal{M}$ ist. Dies ist bei numerischen Daten denkbar.

Durch (1) wird gesichert, daß der Kryptoanalytiker keinen falschen Klartext $M'$ durch $E_K(M') = C'$ einschieben kann. Für die Authentizität muß nur $E_K$ geschützt werden, wenn $D_K$ nicht $E_K$ verrät. $D_K$ kann also publik gemacht werden.

Wir haben gesehen, daß der Schlüssel $K$ aufgeteilt werden kann in einen Chiffrier- und einen Dechiffrierschlüssel. Speziell erhalten wir *Ein-Schlüssel-Kryptosysteme*, wenn Chiffrier- und Dechiffrierschlüssel gleich sind bzw. leicht auseinander bestimmt werden können. Daher sind $E_K$ und $D_K$ auch leicht auseinander abzuleiten. Die Geheimhaltung und Authentizität können nicht voneinander getrennt werden. Damit ist dieses System gut geeignet für private Files oder Informationen, die über ein Rechnernetz übertragen werden. Bei einem Ein-Schlüssel-Kryptosystem spricht man auch von einer *symmetrischen Chiffre*. Bei *Zwei-Schlüssel-Kryptosystemen* unterscheiden sich Chiffrier- und Dechiffrierschlüssel derart, daß ein Schlüssel nicht aus dem anderen bestimmt werden kann. Eine der Transformationen $E_K$

oder $D_K$ kann veröffentlicht werden, ohne die zweite Transformation zu gefährden. Ein solches System ist gut geeignet zur Sicherung von Datenbanken. Bei der „read-only authority“ besitzt der Benutzer $D_K$, bei der „write-only authority“ besitzt er $E_K$. Bei Zwei-Schlüssel-Kryptosystemen wird auch der Name *asymmetrische Chiffre* verwendet.

## 1.3 Informationstheoretische Grundlagen

Die Shannonsche Informationstheorie (siehe [95], [96]) behandelt zwei verwandte Probleme, nämlich das des gestörten Kanals und das der Geheimhaltung. Eine Störung im gestörten Kanal entspricht einer Chiffrierfunktion. Die Informationstheorie mißt die Menge von Informationen in einer Nachricht durch die durchschnittliche Anzahl von Bits, die notwendig sind, um alle möglichen Nachrichten (einer gewissen Art) in einer optimalen Codierung darzustellen. Diese durchschnittliche Anzahl von Bits in einer optimalen Codierung wird als *Entropie* bezeichnet.

**Definition 1.3.1** Es seien $X_1, X_2, \ldots, X_n$ die Nachrichten einer Nachrichtenquelle $X$, die mit den Wahrscheinlichkeiten

$$p(X_1), p(X_2), \ldots, p(X_n), \sum_{i=1}^{n} p(X_i) = 1,$$

auftreten. Die *Entropie* der Nachrichtenquelle wird definiert als

$$H(X) = -\sum_{i=1}^{n} p(X_i) \log_2 p(X_i) \ . \quad \square$$

Die Entropie kann anschaulich als mittlerer Informationsgehalt pro Nachricht aufgefaßt werden. $H(X)$ wird auch geschrieben als

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)} \ .$$

Jeder der Terme $\log_2 \frac{1}{p(x)}$ gibt die Anzahl der Bits an, die zur Darstellung von $x$ in einer optimalen Codierung notwendig sind. In einer optimal codierten Nachricht ist $H(X)$ also die zu erwartende Anzahl von Bits.

**Beispiel 1.3.1** Wir wollen das Geschlecht von Personen in einer Datenbank darstellen. Es bestehen die zwei Möglichkeiten „weiblich“ und „männlich“, die mit der Wahrscheinlichkeit $p(\text{weiblich}) = p(\text{männlich}) = \frac{1}{2}$ angenommen werden. Dann ist

$$H(X) = \frac{1}{2} \log_2 2 + \frac{1}{2} \log_2 2 = 1.$$

Daraus folgt, daß mit nur einem Bit von Information das Geschlecht dargestellt werden kann. $\square$

**Beispiel 1.3.2** Es sei $n = 3$, und die Buchstaben A, B, C werden mit der Wahrscheinlichkeit

$$p(A) = \frac{1}{2} \text{ und } p(B) = p(C) = \frac{1}{4}$$

angenommen. Dann folgt

$$H(X) = \frac{1}{2}\log_2 2 + 2 \cdot \frac{1}{4}\log_2 4 = \frac{1}{2} + 1 = \frac{3}{2}.$$

Eine optimale Codierung c ordnet $A$ einen 1-Bit-Code $(\log_2 2 = 1)$, $B$ und $C$ einen 2-Bit-Code $(\log_2 4 = 2)$ zu, z. B.

$$\text{c}(A)=0,\ \text{c}(B)=10 \text{ und } \text{c}(C)=11.$$

Damit wird etwa c($ABAACABC$)=010001101011. Die durchschnittliche Anzahl von Bits pro Buchstabe für dieses Wort beträgt gerade $\frac{12}{8} = \frac{3}{2}$. Wir sehen, daß $B$ nicht durch 1 codiert werden kann, da dann keine Decodierung möglich ist. Man kann feststellen, daß ein optimaler Code kurze Codewörter für häufig auftretende Nachrichten und lange für seltene Nachrichten verwendet. Dieses Prinzip wurde z. B. beim Morsealphabet realisiert. □

**Beispiel 1.3.3** Für Nachrichten $X_i, i = 1, 2, \ldots, n$, gelte $p(X_i) = \frac{1}{n}$. Dann folgt

$$H(X) = n \cdot \left(\frac{1}{n}\log_2 n\right) = \log_2 n.$$

Speziell für $n = 2^k$ ergibt sich $H(X) = k$. □

**Beispiel 1.3.4** Aus $n = 1$ und $p(X) = 1$ folgt

$$H(X) = \log_2 1 = 0.$$

Der Informationsgehalt dieser Nachricht ist gleich Null, da es keine andere Wahlmöglichkeit gibt. □

Es läßt sich zeigen, daß $H(X)$ maximal ist für $p(X_i) = \frac{1}{n}$, $i = 1, 2, \ldots, n$. Die Entropie einer Nachricht mißt deren Unsicherheit und gibt die Anzahl der Bits an, die man kennen muß, um die Nachricht zu rekonstruieren, wenn sie durch einen gestörten Kanal oder durch Chiffrierung verändert wurde. In Beispiel 1.3.1 sei dem Kryptoanalytiker bekannt, daß der Chiffretext Z$JP7K dem Klartext weiblich oder männlich entspricht. Die Unsicherheit beträgt ein Bit. Es muß nur ein Buchstabe (z. B. der erste, also „w“ oder „m“) und davon nur das unterscheidende Bit bestimmt werden.

**Definition 1.3.2** Es sei $L$ eine Sprache mit $l$ Buchstaben. $X$ sei die Menge aller Nachrichten, die $n$ Buchstaben lang sind. Dann bezeichnet

$$\begin{aligned} r &= \frac{H(X)}{n} \text{ die } \textit{Rate} \text{ oder } \textit{Dichte} \text{ von } L \text{ für Nachrichten der Länge } n, \\ R &= \log_2 l \text{ die } \textit{absolute Rate} \text{ oder } \textit{absolute Dichte} \text{ von } L \text{ und} \\ D &= R - r \text{ die } \textit{Redundanz} \text{ von } L. \quad \square \end{aligned}$$

$r$ mißt die durchschnittliche Anzahl von Bits von Informationen in jedem Buchstaben. Nimmt $n$ zu, dann nimmt die Entropie pro Buchstabe ab, da die Anzahl sinnvoller Nachrichten, relativ zur Länge gesehen, abnimmt. Die Schätzungen für die Sprache Englisch und für große $n$ sind 1,0 bis 1,5 Bits pro Buchstaben. Die Schätzung für die deutsche Sprache liegt auch in diesem Bereich. $R$ ist die maximale Anzahl von Bits von Informationen in einem Buchstaben, wenn alle Buchstabenfolgen gleichwahrscheinlich sind. Dann ist $R$ gleich der maximalen Entropie der jeweiligen Buchstaben (siehe Beispiel 1.3.3). Bei einem 26 Buchstaben umfassenden Alphabet ist die absolute Rate $R = 4,7$ Bits pro Buchstaben. Aus der Abschätzung $r = 1,5$ folgt, daß die Redundanz $D = 3,2$ ist; d. h., bei dieser Abschätzung sind Englisch und Deutsch zu 68% redundant. Dies ist auch im Deutschen so:

Vrlsngn snd ncht lngwlg.

Die Redundanz einer Sprache ergibt sich aus deren Struktur und spiegelt sich in ihren statistischen Eigenschaften wider. Die Häufigkeitsverteilungen für einzelne Buchstaben, Digramme, usw. in der deutschen (englischen) Sprache sind so gegeben, daß

(1) als einzelne Buchstaben E, N und I (E, T und A) häufiger als andere auftreten,
(2) bei Digrammen (Paaren) ER, EN und CH (TH und EN) häufiger als andere erscheinen, während QZ nie in sinnvollen Nachrichten vorkommt,
(3) bei Trigrammen EIN, ICH, DER (THE und ING) häufiger als andere vorhanden sind, aber der Anteil sinnvoller Nachrichten geringer als bei Digrammen ist, und
(4) bei N-Grammen der Anteil sinnvoller Nachrichten mit wachsendem N zunehmend geringer wird.

Solche Informationen sind für Kryptoanalytiker äußerst nützlich. Entsprechende Häufigkeitstabellen befinden sich im Anhang auf den Seiten 147 bis 158.

Die Unsicherheit einer Nachricht (Entropie) kann durch zusätzliche Informationen verringert werden. Es sei $X$ die Menge von 32-Bit-Integer mit $H(X) = 32$. Falls alle Elemente von $X$ gerade sind, kann die Entropie um ein Bit verringert werden. Das niedrigste Bit ist dann immer gleich Null.

**Definition 1.3.3** Es seien $\{X_1, X_2, \ldots, X_m\}$ und $\{Y_1, Y_2, \ldots, Y_n\}$ Nachrichten von Nachrichtenquellen $X$ bzw. $Y$ mit

$$\sum_{i=1}^{m} p(X_i) = \sum_{i=1}^{n} p(Y_i) = 1.$$

$p(x|y)$ sei die bedingte Wahrscheinlichkeit des Auftretens der Nachricht $x$ bei einer gegebenen Nachricht $y$ und $p(x \cap y)$ die Wahrscheinlichkeit des gemeinsamen Auftretens von $x$ und $y$. Dann heißt

$$H(X|Y) = -\sum_{\substack{x \in X \\ y \in Y}} p(x \cap y) \log_2 p(x|y)$$

*Äquivokation* (*equivocation*, *bedingte Entropie* von $X$ unter $Y$). □

Wegen $p(x \cap y) = p(y)\, p(x|y)$ ist

$$H(X|Y) = \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log_2 \frac{1}{p(x|y)}.$$

Wenn $y$ die bekannte Information ist, dann verbleiben als Unsicherheit über $x$ $H(X|Y)$ Bit Information. Dieses Wissensdefizit muß ein Kryptoanalytiker überbrücken, wenn er aus einem Chiffretext $y$ den zugehörigen Klartext x gewinnen will.

**Beispiel 1.3.5** Es sei $m = 4$, und es gelte

$$p(x) = \frac{1}{4} \text{ für alle } x \in X,$$

woraus $H(X) = \log_2 4 = 2$ folgt. Weiter sei $n = 4$ und

$$p(y) = \frac{1}{4} \text{ für alle } y \in Y.$$

Falls $Y_1$ gilt, dann gelte auch $X_1 \vee X_2$, beide mit gleicher Wahrscheinlichkeit. Entsprechendes gelte für $Y_2$, $Y_3$, $Y_4$. Insgesamt schreiben wir dies als

$$Y_1 : X_1 \vee X_2, Y_2 : X_2 \vee X_3, Y_3 : X_3 \vee X_4, Y_4 : X_4 \vee X_1.$$

Es ergibt sich

$$\begin{aligned} p(X_1|Y_1) &= p(X_2|Y_1) = \frac{1}{2} \text{ und} \\ p(X_3|Y_1) &= p(X_4|Y_1) = 0. \end{aligned}$$

Entsprechend erhalten wir die anderen bedingten Wahrscheinlichkeiten. Es folgt

$$H(X|Y) = 4\left(\frac{1}{4} \cdot 2\left(\frac{1}{2}\log_2 2\right)\right) = 1.$$

Die Kenntnis von $Y$ reduziert also die Unsicherheit von $X$ auf ein Bit, d. h. auf zwei Möglichkeiten. □

Für Nachrichten und Schlüssel aus den verwendeten Kryptosystemen benutzen wir folgende

*Vereinbarungen:*
(1) Klartextnachrichten $M$ kommen mit der Wahrscheinlichkeit $p(M)$ vor.
(2) Chiffretextnachrichten $C$ kommen mit der Wahrscheinlichkeit $p(C)$ vor.
(3) Schlüssel $K$ werden mit der Wahrscheinlichkeit $p(K)$ gewählt.

**Definition 1.3.4** Es sei $p(M|C)$ die Wahrscheinlichkeit dafür, daß eine Nachricht $M$ gesendet worden ist, falls $C$ empfangen wurde. Ein Kryptosystem unterliegt *perfekter Geheimhaltung* (ist *absolut sicher*), wenn

$$p(M|C) = p(M)$$

für alle $C$ und $M$ gilt. □

Das bedeutet, daß der Chiffretext dem Kryptoanalytiker keine neue Information liefert. $M$ und $C$ sind unabhängig voneinander. Es sei weiter $p(C|M)$ die Wahrscheinlichkeit, daß $C$ empfangen wurde, falls $M$ gesendet worden ist. Dann gilt

$$p(C|M) = \sum_{\substack{K \\ E_K(M)=C}} p(K).$$

In den meisten Fällen existiert höchstens ein Schlüssel mit dieser Eigenschaft. Aus der Unabhängigkeit von $C$ und $M$ folgt, daß ein Kryptosystem auch perfekter Geheimhaltung unterliegt, wenn

$$p(C|M) = p(C)$$

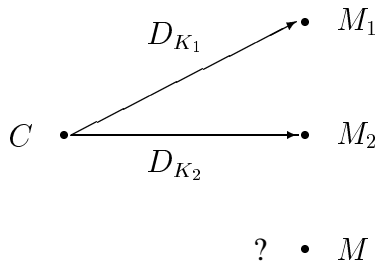
für alle $C$ und $M$ gilt. Dies bedeutet, daß die Wahrscheinlichkeit, einen speziellen Chiffretext zu empfangen, nicht vom Klartext abhängt.

Wir stellen fest, daß eine perfekte Geheimhaltung dann möglich ist, wenn zufällige Schlüssel benutzt werden, die mindestens ebenso lang sind wie die Nachrichten, die sie verschlüsseln.

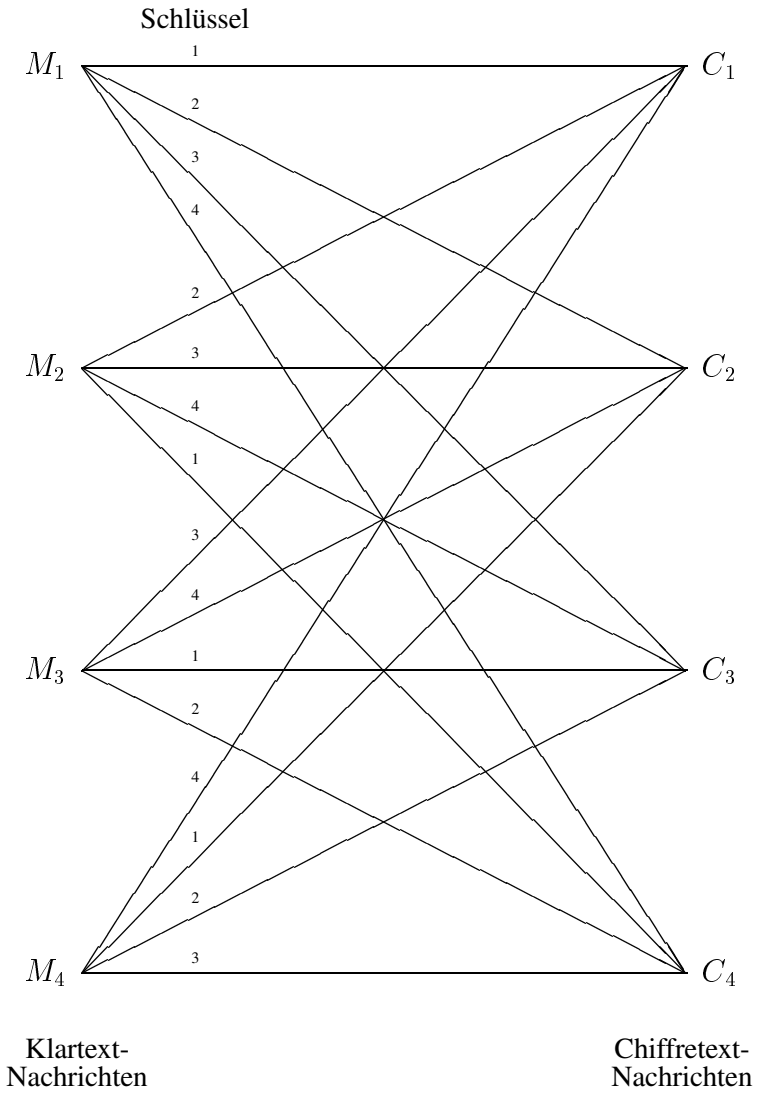
**Beispiel 1.3.6** Das Bild von Seite 9 illustriert ein Kryptosystem, das perfekter Geheimhaltung unterliegt. Für alle $M$ und $C$ gilt

$$\begin{aligned} p(M|C) &= p(M) = \frac{1}{4} \text{ bzw.} \\ p(C|M) &= p(C) = \frac{1}{4}. \end{aligned}$$

Wird z. B. der Chiffretext $C_2$ empfangen, so ist es völlig unklar, welche der Nachrichten $M_1, M_2, M_3$ oder $M_4$ chiffriert worden ist. Eine perfekte Geheimhaltung verlangt, daß die Anzahl der Schlüssel mindestens ebenso groß ist wie die Anzahl der möglichen Nachrichten. Ist dagegen die Anzahl der Schlüssel kleiner als die Anzahl der möglichen Nachrichten, so existiert eine Nachricht $M$, bei der es für einen gegebenen Chiffretext $C$ keinen Schlüssel $K$ mit $D_K(C) = M$ gibt. Dies wird durch das folgende Bild veranschaulicht:




Dann folgt $p(M|C) = 0$. Dadurch ist es möglich, denkbare Klartexte für $C$ auszuschließen. □




Perfekte Geheimhaltung

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

Lösung der Substitutionschiffre

**Beispiel 1.3.7** Der Chiffretext $C$=LZWJWAKFGGLZWJDSFYMSYWTMLXJWFUZ, der aus 31 Buchstaben besteht, wurde mit $0 \leq k \leq 25$ durch eine Cäsar-Chiffre erzeugt. Da die Anzahl der möglichen Schlüssel (26) kleiner ist als die Anzahl der möglichen Sätze der Länge 31, gibt es keine perfekte Geheimhaltung. Die Chiffre wird durch Probieren aller 26 Schlüssel gebrochen, was wir mit Hilfe der Tabelle auf Seite 10 durchführen. Wir erkennen, daß die Lösung durch den Schlüssel 18 erfolgt:

$M$=THEREISNOOTHERLANGUAGEBUTFRENCH.

Da sich nur für den Schlüssel 18 ein sinnvoller Text ergibt, gilt

$$\begin{aligned} p(M|C) &= 1 \text{ und } p(M'|C) = 0 \text{ für } M' \neq M \text{ sowie} \\ p(C|M) &= p(18) = \frac{1}{26} \text{ und } p(C|M') = 0 \text{ für } M' \neq M. \end{aligned}$$ □

**Beispiel 1.3.8** Durch eine kleine Änderung wird perfekte Geheimhaltung erreicht: Jeder Buchstabe wird um ein zufälliges Stück verschoben. $K$ ist durch einen Strom $k_1, k_2, \ldots$ gegeben, wobei jedes der $k_i$ eine zufällige Zahl mit $0 \leq k_i \leq 25$ ist. Ein Chiffretext $C$ kann damit jeden möglichen Klartext liefern. Mit dem Schlüsselstrom

10, 8, 22, 15, 9, 8, 8, 24, 10, 2, 3, 19, 18, 18, 11, 25, 11, 21, 8, 5, 5, 18, 6, 20, 3, 10, 6, 12, 20, 0, 19

wird der Chiffretext aus Beispiel 1.3.7 zu

BRAUNSCHWEIGERSTUDENTENSINDKLUG

dechiffriert. Der Schlüsselstrom darf nicht wiederholt oder für eine andere Nachricht verwendet werden, da sonst durch Korrelation der beiden Chiffretexte eventuell die Chiffre gebrochen werden kann. Diese Schlüsselart wird als *one-time pad* bezeichnet. □

**Definition 1.3.5** Es sei $K$ eine Menge von Schlüsseln und $C$ eine Menge von Chiffretexten. Dann wird

$$H(K|C) = \sum_{c \in C} p(c) \sum_{k \in K} p(k|c) \log_2 \frac{1}{p(k|c)}$$

als *Schlüsseläquivokation* (*key equivocation*) von $K$ unter der Bedingung $C$ bezeichnet. □

$p(k|c)$ gibt die Wahrscheinlichkeit dafür an, daß bei einem empfangenen Chiffretext $c$ der Schlüssel $k$ zur Verschlüsselung verwendet wurde. Die Schlüsseläquivokation gibt die erwartete Anzahl von Bits an, die zur Bestimmung eines Schlüssels $k \in K$ bekannt sein müssen, sofern ein Chiffretext $c \in C$ gegeben ist. $H(K|C)$ ist also ein Maß für die Unsicherheit des Schlüssels. Sei $C_n$ eine Menge von Chiffretexten der Länge $n$. Mit wachsendem $n$ wird $H(K|C_n)$ meistens kleiner. Ist $H(K|C_n) = 0$, dann besteht keine Unsicherheit mehr und die Chiffre ist unter der Voraussetzung, daß ausreichende Rechnerleistung zur Verfügung steht, theoretisch zu brechen.

Im folgenden benötigen wir einige „Definitionen“. Die *Eindeutigkeitsdistanz* (*unicity distance*) eines Kryptosystems bezeichnet die Menge des Chiffretextes, die notwendig ist,

um den Schlüssel eindeutig bestimmen zu können. Wenn $n$ die Anzahl der empfangenen Chiffretextzeichen ist, so ist die Eindeutigkeitsdistanz durch das kleinste $n$ gegeben, für das $H(K|C_n)$ „nahe 0“ ist.

Eine Chiffre gilt als *uneingeschränkt sicher*, wenn $H(K|C_n)$ sich auch für große $n$ nicht 0 nähert. Die meisten Chiffren sind zu komplex, um die Wahrscheinlichkeiten zu berechnen, mit deren Hilfe dann die Eindeutigkeitsdistanz bestimmt werden könnte. Shannon [96] zeigte jedoch, daß es möglich ist, die Eindeutigkeitsdistanz für gewisse Kryptosysteme unter Benutzung eines „random-cipher“-Modells abzuschätzen. Wir folgen den etwas anderen Betrachtungen von Hellman [38]. Er nimmt an, daß jeder Klartext und jeder Chiffretext aus einem endlichen Alphabet von $l$ Symbolen stammt. Dann existieren $2^{Rn}$ Nachrichten der Länge $n$, die in zwei Teilmengen aufgeteilt werden, und zwar in die

(1) Menge von $2^{rn}$ sinnvollen Nachrichten mit der Wahrscheinlichkeit $\frac{1}{2^{rn}}$ und in die
(2) Menge von $2^{Rn} - 2^{rn}$ sinnlosen Nachrichten mit der Wahrscheinlichkeit 0.

Dabei ist $R = \log_2 l$ die absolute Rate und $r$ die Rate der Sprache. Außerdem werde angenommen, daß $2^{H(K)}$ gleichwahrscheinliche Schlüssel existieren, wobei $H(K)$ die Schlüssel-Entropie (erwartete Anzahl von Bits im Schlüssel bei einer optimalen Codierung) ist.

Wir kommen nun zum Begriff der *zufälligen Chiffre*. Dabei sei für jeden Schlüssel $K$ und jeden Chiffretext $C$ die Dechiffrierung $D_K(C)$ eine unabhängige Zufallsvariable, die gleichmäßig über alle $2^{Rn}$ Nachrichten (sinnvoll oder sinnlos) verteilt ist. Intuitiv bedeutet das, daß für gegebenes $K$ und $C$ die Dechiffrierung $D_K(C)$ mit derselben Wahrscheinlichkeit den einen Klartext wie jeden anderen Klartext produziert. Tatsächlich ist sie aber nicht ganz unabhängig, da ein Schlüssel eindeutig eine gegebene Nachricht chiffrieren muß, d. h., aus $C \neq C'$ muß $D_K(C) \neq D_K(C')$ folgen.

Gegeben sei ein Schlüssel $K$ und Klartext $M$ mit zugehörigem Chiffretext $C = E_K(M)$. Gibt es zu diesem Chiffretext $C$ einen Klartext $M'$ $(M' \neq M)$ und einen Schlüssel $K'$ $(K' \neq K)$ mit $C = E_{K'}(M)$ oder $C = E_{K'}(M')$, so liegt eine *falsche Schlüssel-Dechiffrierung* (spurious key decipherment) vor. Das Bild von Seite 14 zeigt zwei falsche Schlüssel-Dechiffrierungen. Eine falsche Schlüsseldechiffrierung resultiert aus dem Chiffretext $C_3$, eine andere aus $C_6$. Ein Kryptoanalytiker, der $C_3$ erhält, kennt nicht den zugehörigen Klartext (war es $M_1$ mit Schlüssel 2 oder $M_3$ mit Schlüssel 1?). Bei $C_6$ kennt er nicht den Schlüssel (war es $M_4$ mit Schlüssel 1 oder mit Schlüssel 2?). Für jede richtige Lösung (Schlüssel) eines speziellen Chiffretextes $C$ gibt es $2^{H(K)} - 1$ weitere Schlüssel. Jeder dieser Schlüssel $K'$ hat dieselbe Wahrscheinlichkeit $q$, eine falsche Schlüssel-Dechiffrierung von $C$ zu liefern. Eine falsche Schlüssel-Dechiffrierung liegt aber nur dann vor, wenn sich dabei ein sinnvoller Klartext ergibt. Nach unserem Modell der zufälligen Chiffre ist diese Wahrscheinlichkeit

$$q = \frac{2^{rn}}{2^{Rn}} = 2^{(r-R)n} = 2^{-Dn}.$$

Da $\frac{1}{2^{Dn}}$ mit wachsendem $n$ schnell gegen 0 strebt und $2^{H(K)} \gg 1$ gilt, folgt für die erwartete Anzahl der Fehlentscheidungen, also der falschen Schlüssel-Dechiffrierungen,

$$F = \left(2^{H(K)} - 1\right) \frac{1}{2^{Dn}} \cong 2^{H(K)-Dn}.$$

Aufgrund der starken Abnahme von $F$ mit wachsendem $n$ wird durch $F = 1$ der Punkt festgelegt, ab dem die Anzahl der falschen Schlüssel-Dechiffrierungen klein genug ist, um die Chiffre zu brechen. Für $F = 1$ gilt

$$\log_2 F = 0 = H(K) - Dn.$$

Damit ist

$$n = \frac{H(K)}{D}$$

der approximative Wert für die Eindeutigkeitsdistanz. $n$ ist die Anzahl der Chiffretextzeichen, die wenigstens erforderlich sind, um die Chiffre zu brechen. Ein System wird also unsicher, falls die Anzahl der Chiffretextzeichen diesen Wert $n$ überschreitet.

Falls bei einem gegebenen $n$ die Anzahl der möglichen Schlüssel mindestens so groß wie die Anzahl der möglichen Nachrichten ist, also

$$H(K) \geq \log_2 \left(2^{Rn}\right) = Rn$$

gilt, dann folgt

$$H(K) - Dn \geq (R - D)n = rn \neq 0.$$

Das bedeutet, daß bei nur einmaliger Anwendung der Chiffre diese theoretisch nicht gebrochen werden kann. Dies ist das Prinzip, das hinter dem „one-time pad“ steckt (siehe Beispiel 1.3.8).

**Beispiel 1.3.9** Im DES (Data Encryption Standard, siehe Abschnitt 2.6) werden 64-Bit-Blöcke (8 Zeichen) durch einen 56-Bit Schlüssel chiffriert. Der DES entspricht einer vernünftigen Approximation des Modells der zufälligen Chiffre mit $H(K) = 56$ und $D = 3,2$ (für Englisch). Daraus folgt $n = \frac{56}{3,2} = 17,5$ Zeichen. □

**Beispiel 1.3.10** Wir betrachten die Cäsar-Chiffre mit $k$, $0 \leq k \leq 25$. Dann folgt

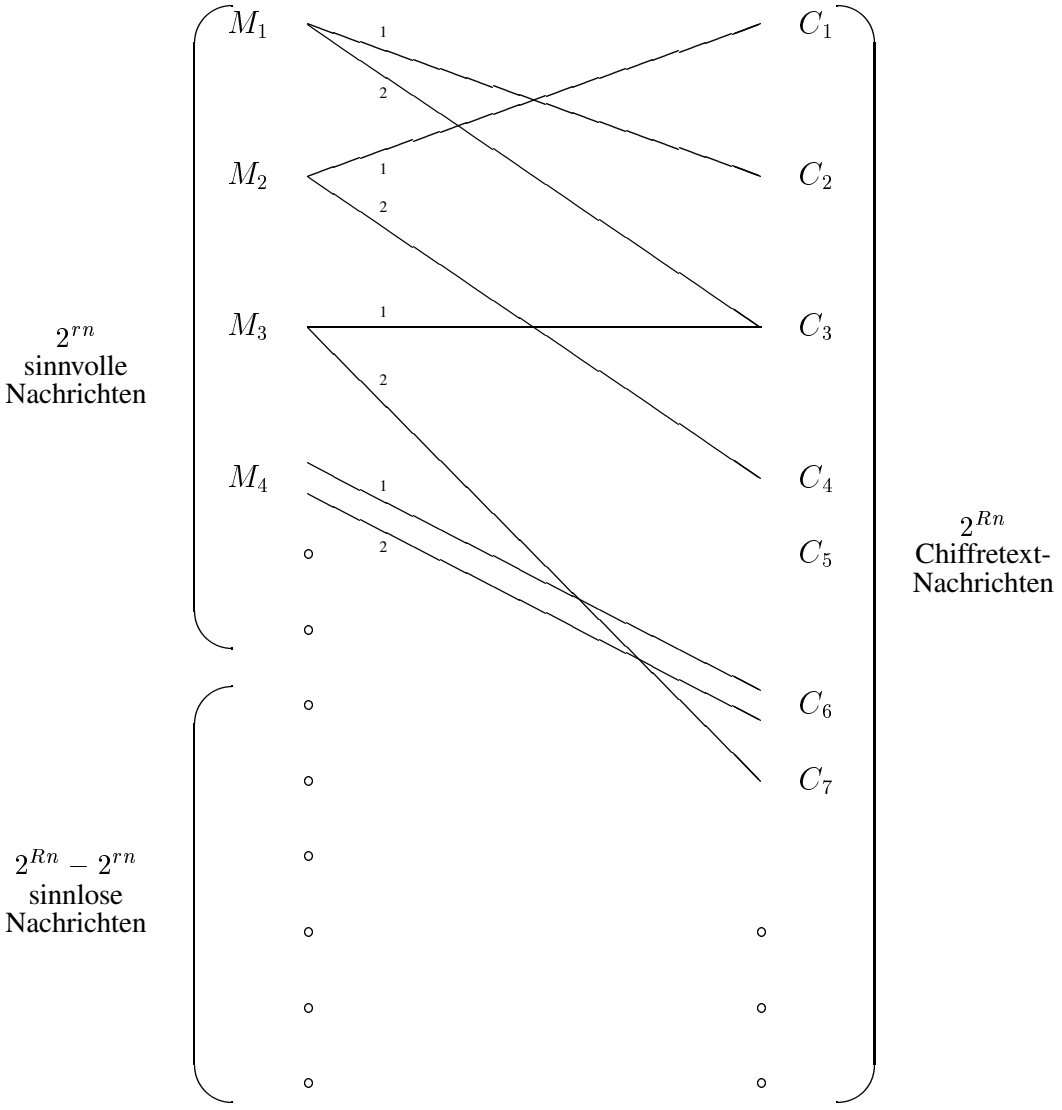
$$H(K) = \log_2 26 = 4,7$$

und damit

$$n = \frac{4,7}{3,2} = 1,5 \text{ Zeichen.}$$

Wir wissen jedoch, daß keine Cäsar-Chiffre mit nur ein oder zwei Zeichen gelöst werden kann. Der Grund dafür ist hier die schlechte Approximation des Modells der zufälligen Chiffre. Die meisten Chiffretexte können nicht durch einen sinnvollen Klartext erzeugt werden. Zum Beispiel kann QQQQ nur durch AAAA, BBBB, CCCC, ... erzeugt werden. Das ist gleichbedeutend damit, daß die Dechiffrierung nicht gleichmäßig über alle Nachrichten verteilt ist. □

$\frac{H(K)}{D}$ ist nur eine Abschätzung für die Anzahl der Chiffretextzeichen, die benötigt werden, um eine Chiffre zu brechen. Allgemein haben aber Chiffren eine Eindeutigkeitsdistanz, die wenigstens $\frac{H(K)}{D}$ ist. Trotzdem ist dieser Wert eine gute Approximation. Für $D \to 0$ wird selbst eine Cäsar-Chiffre unbrechbar. Dies ist u. a. der Fall, wenn Zahlen chiffriert werden, bei denen jede Ziffer zählt. Dann ist keine Redundanz vorhanden.


$M_1$
$M_2$
$M_3$
$M_4$
$C_1$
$C_2$
$C_3$
$C_4$
$C_5$
$C_6$
$C_7$
1
2
1
2
1
2
1
2
$2^{rn}$
sinnvolle
Nachrichten
$2^{Rn}$
Chiffretext-
Nachrichten
$2^{Rn} - 2^{rn}$
sinnlose
Nachrichten
Zufällige Chiffre

# 2 Klassische kryptographische Verfahren

## 2.1 Transpositionschiffren

Transpositionschiffren ordnen die Buchstaben oder Bits des Klartextes nach irgendeinem Schema oder einer geometrischen Figur um. Häufig ist dieses Schema eine zweidimensionale Matrix.

Bei einer *Spaltentransposition* wird der Klartext zeilenweise in die Matrix eingeschrieben. Der Chiffretext ergibt sich aus den Spalten, die in irgendeiner Weise angeordnet sind.

**Beispiel 2.1.1** Der Klartext VORLESUNGEN werde in eine $3 \times 4$ Matrix zeilenweise eingetragen:

$$\begin{pmatrix} \text{V} & \text{O} & \text{R} & \text{L} \\ \text{E} & \text{S} & \text{U} & \text{N} \\ \text{G} & \text{E} & \text{N} & \end{pmatrix}$$

Die Spalten in der Ordnung 2-4-1-3 ergeben den Chiffretext

OSELNVEGRUN. □

Viele Transpositionschiffren permutieren die Zeichen des Klartextes mit einer *festen Periode* $d$. Es sei $D = \{1, \ldots, d\}$. $f : D \longrightarrow D$ sei eine Permutation. Der Schlüssel einer Transpositionschiffre ist dann durch ein Paar $K = (d, f)$ gegeben. Aus dem Klartext

$$M = m_1 \ldots m_d \, m_{d+1} \ldots m_{2d} \ldots$$

ergibt sich der Chiffretext

$$C = E_K(M) = m_{f(1)} \ldots m_{f(d)} \, m_{d+f(1)} \ldots m_{d+f(d)} \ldots \ .$$

Die Dechiffrierung erfolgt durch die inverse Permutation.

**Beispiel 2.1.2** Es sei $d = 4$ und $f$ durch

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $f(i)$ | 2 | 4 | 1 | 3 |

gegeben. Der Klartext $M$ =VORLESUNGEN liefert den Chiffretext

$C = E_K(M)$ =OLVRSNEUEGN. □

Beispiel 2.1.2 ergibt sich auch ähnlich wie Beispiel 2.1.1. Der Klartext wird zeilenweise in eine $3 \times 4$-Matrix eingetragen:

$$\begin{pmatrix} \text{V} & \text{O} & \text{R} & \text{L} \\ \text{E} & \text{S} & \text{U} & \text{N} \\ \text{G} & \text{E} & \text{N} & \end{pmatrix}$$

Deren Spalten werden gemäß $f$ permutiert, also in der Reihenfolge 2-4-1-3:

$$\begin{pmatrix} \text{O} & \text{L} & \text{V} & \text{R} \\ \text{S} & \text{N} & \text{E} & \text{U} \\ \text{E} & & \text{G} & \text{N} \end{pmatrix}$$

Zeilenweises Auslesen ergibt den Chiffretext $C = E_K(M)$ aus Beispiel 2.1.2. Für dieses Vorgehen muß eine Matrix aufgestellt werden. Das Beispiel 2.1.2 kann dagegen direkt berechnet werden und ist daher effizienter.

Durch Überprüfung der relativen Häufigkeiten der Buchstaben kann festgestellt werden, ob es sich bei der Chiffre um eine Transpositionschiffre handelt. Der Wert muß dem erwarteten Wert der jeweiligen Buchstaben im Klartext entsprechen. Diese Chiffren werden häufig durch „anagramming“ gebrochen (siehe [98], [28]). Dies ist ein Prozeß, der die Buchstaben wieder an die richtige Stelle bringt. Er wird durch den Einsatz von Tafeln für die Häufigkeit von Di- und Trigrammen erleichtert (Tafeln auf den Seiten 147 bis 158).

Wir wollen nun die erwartete Anzahl von Buchstaben bestimmen, die zum Brechen einer Permutationschiffre der Periode $d$ nötig sind. Es sind $d!$ Schlüssel vorhanden. Aus der Annahme, daß alle Schlüssel gleichwahrscheinlich sind, folgt $H(K) = \log_2 d!$ und damit

$$n = \frac{H(K)}{D} = \frac{\log_2 d!}{D}.$$

Mit der Stirlingschen Approximation für $d!$ ergibt sich die Eindeutigkeitsdistanz

$$n = d \cdot \frac{\log_2 \left(\frac{d}{e}\right)}{3,2} = 0,3d \, \log_2 \left(\frac{d}{e}\right).$$

**Beispiel 2.1.3** Es sei $d = 27$. Dann folgt $\frac{d}{e} \approx 10$, $\log_2 \frac{d}{e} \cong 3,2$ und damit die Eindeutigkeitsdistanz $n \cong 27$. □

## 2.2 Chiffren mit einfacher Substitution

Es gibt vier Typen von Substitutionschiffren:

(1) *Einfache Substitution*: Jeder Buchstabe des Klartextes wird durch einen Buchstaben des Chiffretextes ersetzt. Dabei wird eine bijektive Abbildung zwischen Klartext- und Chiffretextalphabet benutzt.

(2) *Homophone Substitution*: Jeder Buchstabe des Klartextes kann durch verschiedene Buchstaben des Chiffretextes ersetzt werden.

(3) *Polyalphabetische Substitution*: Die Buchstaben des Klartextes werden in irgendeiner Reihenfolge (z. B. periodisch) durch verschiedene Abbildungen chiffriert.

(4) *Polygramm-Substitution*: Ganze Blöcke von Buchstaben des Klartextes werden gemeinsam ersetzt.

**Definition 2.2.1** Es seien $\mathcal{A}$ und $\mathcal{C}$ Alphabete gleicher Mächtigkeit (Klartext- bzw. Chiffretextalphabet). Eine *Chiffre mit einfacher Substitution* wird durch eine bijektive Abbildung $f : \mathcal{A} \longrightarrow \mathcal{C}$ gegeben. Dabei ist $f$ der Schlüssel der Chiffre. □

Die Chiffrierung erfolgt durch den durch $f$ bestimmten Homomorphismus, den wir mit demselben Symbol $f$ als

$$f : \mathcal{A}^* \longrightarrow \mathcal{C}^*$$

schreiben.

**Beispiel 2.2.1** Es sei $f$ gegeben durch:

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

Der Klartext $M$=VORLESUNGEN wird zum Chiffretext

$$E_K(M)\text{=VIOCEPRHTEH.}$$

Das Beispiel benutzt das Schlüsselwort KLUGESTUDENTEN, das zunächst im Chiffretextalphabet $\mathcal{C}$ aufgelistet wird, wobei Duplikate (U, E, T, N) natürlich nur einmal eingetragen werden. Die nicht im Schlüssel vorkommenden Buchstaben werden dahinter aufgelistet. □

**Beispiel 2.2.2** Wir betrachten Verschiebechiffren. Gegeben sei $\mathcal{A} = \{a_0, a_1, \ldots, a_{n-1}\}$ und $f(a) = (a + k) \bmod n$. Dabei ist $a$ sowohl Buchstabe als auch Position in $\mathcal{A}$. $k$ gibt die Größe der Verschiebung an. Die Position der Buchstaben liefert die folgende Tabelle:

| 0 | 1 | $\cdots$ | 25 |
|---|---|---|---|
| A | B | $\cdots$ | Z |

Damit entspricht das Beispiel der Cäsar-Chiffre aus Beispiel 1.1.2. □

**Beispiel 2.2.3** Gegeben sei $f(a) = (ak) \bmod n$ mit ggt$(k, n) = 1$. Die Buchstaben unter $f$ bilden eine vollständige Menge von Resten. Wir wählen $n = 26$ und $k = 9$ und erhalten

$\mathcal{A}$ : ABC DEF GHI JKL MNO PQR STU VWX YZ
$\mathcal{C}$ : AJS BKT CLU DMV ENW FOX GPY HQZ IR .

Falls ggt$(k, n) \neq 1$ gilt, werden verschiedene Buchstaben von $\mathcal{A}$ durch denselben Buchstaben aus $\mathcal{C}$ chiffriert. Für $n = 26$ und $k = 13$ ergibt sich

$$f(A)\text{=}f(C)\text{=}f(E)\text{=}\ldots\text{=}f(Y)\text{=}A \quad (=0),$$
$$f(B)\text{=}f(D)\text{=}f(F)\text{=}\ldots\text{=}f(Z)\text{=}N \quad (=13). \quad \square$$

**Beispiel 2.2.4** (a) *Chiffren durch affine Transformation* sind durch

$$f(a) = (ak_1 + k_0) \bmod n \text{ mit ggt}(k_1, n) = 1$$

definiert. Dies entspricht einer Kombination von Beispiel 2.2.2 mit Beispiel 2.2.3.

(b) *Chiffren durch polynomiale Transformation vom Grad $t$* sind durch

$$f(a) = (a^t k_t + a^{t-1} k_{t-1} + \cdots + a^1 k_1 + k_0) \bmod n$$

definiert. Eine Bedingung, die die $k_i$, $i = 0, \ldots, t$, erfüllen müssen, so daß hierdurch tatsächlich eine Bijektion gegeben wird, ist allgemein nicht bekannt. Durch Überprüfung stellt man jedoch fest, daß etwa

$$f(a) = (a^7 + k_0) \bmod 31$$

für alle $k_0 \in \{0, \ldots, 30\}$ eine Bijektion ist. Ebenso sind zum Beispiel

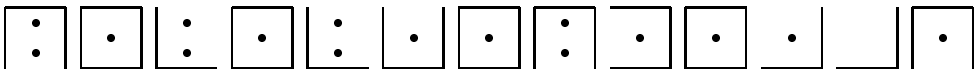
$$\begin{aligned} f(a) &= (25a^5 + 25a^4 + 25a^3 + 24a^2 + 22a + k_0) \bmod 26, \\ f(a) &= (13a^6 + a^5 + a^4 + a^3 + 2a^2 + 17a + k_0) \bmod 26 \text{ und} \\ f(a) &= (13a^2 + 2a + k_0) \bmod 26 \end{aligned}$$

für alle $k_0 \in \{0, \ldots, 25\}$ Bijektionen. Man kann zeigen (siehe Satz 3.10), daß für alle Primzahlen $p$ und Zahlen $t \in \mathbb{N}$ mit ggt$(t, p-1) = 1$ durch

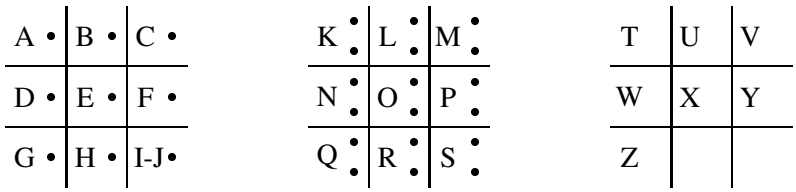
$$f(a) = (a^t + k_0) \bmod p$$

für alle $k_0 \in \{0, \ldots, p-1\}$ eine Bijektion gegeben wird. □

**Beispiel 2.2.5** Einige Substitutionschiffren benutzen unübliche Chiffretext-Alphabete. Ein Beispiel ist die Friedhofchiffre, die sich auf einem Grabstein des Trinity-Friedhofes in New York befindet. Der Grabstein wurde bereits 1794 aufgestellt, die Lösung jedoch erst 1896 im *New York Herald* veröffentlicht:



Chiffretext




Friedhofchiffre □

Es wurden zum Teil auch musikalische Symbole als Chiffre verwendet, zum Teil ernsthaft, aber auch als Spaß. Bach hat z. B. seinen Namen in der Kunst der Fuge kodiert.

Wir wollen nun die Anzahl der Buchstaben bestimmen, die nötig sind, eine allgemeine Substitutionschiffre zu brechen. Es sei ein Alphabet der Größe $l$ gegeben. Folglich sind $l!$ Schlüssel vorhanden. Falls alle Schlüssel gleichwahrscheinlich sind, gilt

$$n \cong \frac{H(K)}{D} = \frac{\log_2 l!}{D}.$$

Für Englisch ist die Eindeutigkeitsdistanz

$$n \cong \frac{\log_2 26!}{3,2} \cong \frac{88,4}{3,2} = 27,6.$$

Dies erkärt die Schwierigkeit, die Friedhofchiffre zu brechen. Chiffren mit polynomialer Transformation haben kleinere Eindeutigkeitsdistanzen. Bei Verschiebechiffren ist $n \cong 1,5$ (siehe Beispiel 1.3.10), obwohl zum Brechen der Chiffre mehr Chiffretext erforderlich ist.

Chiffren mit einfacher Substitution sind unter Benutzung der Häufigkeitsverteilung der einzelnen Buchstaben i. allg. leicht mit einem nur-Chiffretext-Angriff zu brechen. Die zweite Spalte der ersten Tabelle von Seite 20 zeigt die erwartete Häufigkeitsverteilung der einzelnen Buchstaben der englischen Sprache. Die dritte Spalte enthält die Häufigkeitsverteilung für einen speziellen englischen Text mit 67375 Buchstaben, die auch durch die Anzahl der Sterne angedeutet wird. Aus dem aktuellen Text sind E und T sicher herauszufinden. A, O, N, I, R und S weisen dagegen gewisse Abweichungen auf. Mit Di- und Trigrammverteilungen kommt man dann weiter. Computerfiles dagegen haben sicher andere Häufigkeitsverteilungen. Das Leerzeichen ist darin am häufigsten vertreten! Verschiebechiffren sind leichter zu brechen (siehe Beispiel 1.3.7).

Die zweite Tabelle von Seite 20 gibt die erwartete Häufigkeitsverteilung der einzelnen Buchstaben der deutschen Sprache an. Die verschiedenen Tabellen befinden sich auch im Anhang (siehe Seite 147 bis Seite 158). Man beachte, daß je nach Art der benutzten Texte in der Literatur sehr unterschiedliche Häufigkeitstabellen erscheinen.

**Beispiel 2.2.6** Es sei eine Chiffre durch eine affine Transformation definiert. Wir wollen ein Verfahren zum Brechen solcher Chiffren darstellen. Gegeben seien $t$ vermutete Zuordnungen von Klartext- und Chiffretextbuchstaben ($m_i \longleftrightarrow c_i,\ 1 \leq i \leq t$). Ggf. ist das Gleichungssystem

$$\begin{aligned}(m_1 k_1 + k_0) \bmod n &= c_1 \\ &\cdots \\ (m_t k_1 + k_0) \bmod n &= c_t\end{aligned}$$

zur Bestimmung von $k_0$ und $k_1$ lösbar. Speziell betrachen wir die Zuordnung

| Klartext | | Chiffretext | |
|---|---|---|---|
| E | (4) | K | (10) |
| J | (9) | T | (19) |
| N | (13) | U | (20) . |

Daraus folgt das Gleichungssystem

$$\begin{aligned}(4k_1 + k_0) \bmod 26 &= 10 \quad (1) \\ (9k_1 + k_0) \bmod 26 &= 19 \quad (2) \\ (13k_1 + k_0) \bmod 26 &= 20 \quad (3)\ .\end{aligned}$$

Gleichung (1) von (2) subtrahiert ergibt

$$(5k_1) \bmod 26 = 9,$$

und daraus folgt $k_1 = 7$. Da $\text{ggt}(5, 26) = 1$ gilt, ist $k_1$ eindeutig bestimmt. Andernfalls wären mehr Gleichungen zur Berechnung von $k_1$ nötig. Einsetzen in (1) liefert

$$(28 + k_0) \bmod 26 = 10$$

| Zeichen | erwartet | aktuell % | |
|---|---|---|---|
| A | 8,17 | 7,5 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| B | 1,49 | 1,4 | ∗∗∗ |
| C | 2,78 | 4,1 | ∗∗∗∗∗∗∗∗ |
| D | 4,25 | 3,2 | ∗∗∗∗∗∗ |
| E | 12,70 | 12,7 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| F | 2,23 | 2,3 | ∗∗∗∗∗ |
| G | 2,02 | 1,9 | ∗∗∗∗ |
| H | 6,09 | 3,8 | ∗∗∗∗∗∗∗∗ |
| I | 6,97 | 7,7 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| J | 0,15 | 0,2 | |
| K | 0,77 | 0,4 | ∗ |
| L | 4,03 | 3,8 | ∗∗∗∗∗∗∗∗ |
| M | 2,41 | 3,0 | ∗∗∗∗∗∗ |
| N | 6,75 | 7,0 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| O | 7,51 | 7,5 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| P | 1,93 | 3,0 | ∗∗∗∗∗∗ |
| Q | 0,08 | 0,2 | |
| R | 5,99 | 6,7 | ∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| S | 6,33 | 7,3 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| T | 9,06 | 9,2 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| U | 2,76 | 2,8 | ∗∗∗∗∗∗ |
| V | 0,98 | 1,0 | ∗∗ |
| W | 2,36 | 1,4 | ∗∗∗ |
| X | 0,15 | 0,3 | ∗ |
| Y | 1,97 | 1,6 | ∗∗∗ |
| Z | 0,07 | 0,1 | |

Jeder ∗ repräsentiert 0,5% des aktuellen Textes
Anzahl der Buchstaben des aktuellen Textes= 67375

Häufigkeitsverteilung der Buchstaben für Englisch

| Zeichen | erwartet in % | |
|---|---|---|
| A | 6,43 | ∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| B | 1,85 | ∗∗∗∗ |
| C | 3,26 | ∗∗∗∗∗∗∗ |
| D | 5,12 | ∗∗∗∗∗∗∗∗∗∗ |
| E | 17,74 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| F | 1,56 | ∗∗∗ |
| G | 2,69 | ∗∗∗∗∗ |
| H | 5,22 | ∗∗∗∗∗∗∗∗∗∗ |
| I | 7,60 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| J | 0,23 | |
| K | 1,40 | ∗∗∗ |
| L | 3,49 | ∗∗∗∗∗∗∗ |
| M | 2,75 | ∗∗∗∗∗∗ |
| N | 10,01 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| O | 2,39 | ∗∗∗∗∗ |
| P | 0,64 | ∗∗ |
| Q | 0,01 | |
| R | 6,98 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| S | 6,88 | ∗∗∗∗∗∗∗∗∗∗∗∗∗∗ |
| T | 5,94 | ∗∗∗∗∗∗∗∗∗∗∗∗ |
| U | 4,27 | ∗∗∗∗∗∗∗∗∗ |
| V | 0,64 | ∗∗ |
| W | 1,73 | ∗∗∗ |
| X | 0,02 | |
| Y | 0,04 | |
| Z | 1,10 | ∗∗ |

Jeder ∗ repräsentiert 0,5%
Anzahl der Buchstaben = 224826

Häufigkeitsverteilung der Buchstaben für Deutsch

und damit $k_0 = 8$. Jetzt muß noch überprüft werden, ob (3) erfüllt ist. Das ist hier nicht der Fall. Falls (3) nicht erfüllt ist, ist eine der drei Korrespondenzen falsch oder die Chiffre ist nicht durch eine affine Transformation gegeben. □

Allgemeine Verfahren zum Lösen von Substitutionschiffren finden sich in [68].

## 2.3 Chiffren mit homophoner Substitution

**Definition 2.3.1** Es seien $\mathcal{A}$ und $\mathcal{C}$ Alphabete. Eine Chiffre mit *homophoner Substitution* ist durch eine Abbildung $f : \mathcal{A} \longrightarrow 2^{\mathcal{C}}$ gegeben, für die für alle $a_1, a_2 \in \mathcal{A}, a_1 \neq a_2$, $f(a_1) \neq \emptyset$ sowie die Beziehung $f(a_1) \cap f(a_2) = \emptyset$ gilt. □

Ein Klartext $M = m_1\, m_2$ wird also chiffriert als $C = c_1\, c_2$ mit zufälligem $c_i \in f(m_i)$ für $i = 1, 2$.

**Beispiel 2.3.1** Die Buchstaben werden chiffriert als Zahlen zwischen 0 und 99. Die Anzahl der Zahlen, die einem Buchstaben zugeordnet werden, ist proportional zur relativen Häufigkeit des Auftretens dieses Buchstabens (gemäß der deutschen Sprache, siehe Tabelle auf Seite 20). Wir wählen:

| Buchstaben | Homophone | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | 15 | 22 | 43 | 87 | 98 | | | | | | | | | | | | | |
| E | 02 | 05 | 14 | 17 | 19 | 24 | 26 | 34 | 46 | 50 | 57 | 63 | 68 | 75 | 80 | 81 | 91 | 99 |
| G | 01 | 86 | 88 | | | | | | | | | | | | | | | |
| K | 44 | | | | | | | | | | | | | | | | | |
| L | 04 | 28 | 82 | | | | | | | | | | | | | | | |
| N | 08 | 21 | 42 | 45 | 53 | 66 | 72 | 77 | 92 | 95 | | | | | | | | |
| S | 10 | 18 | 36 | 39 | 61 | 83 | 93 | | | | | | | | | | | |
| T | 03 | 11 | 12 | 23 | 30 | 85 | | | | | | | | | | | | |
| U | 07 | 25 | 60 | 70 | | | | | | | | | | | | | | |

Als Beispiel einer Chiffrierung erhalten wir:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ = | K | L | U | G | E | S | T | U | D | E | N | T | E | N |
| $C$ = | 44 | 82 | 25 | 01 | 57 | 61 | 23 | 70 | 43 | 17 | 92 | 85 | 34 | 21 |

□

Man erkennt, daß die Häufigkeitsverteilung von Einzelbuchstaben hierbei nicht nützt, um die Chiffre zu brechen. Unter Umständen ist dies jedoch durch Digrammverteilungen möglich. Falls dagegen jedem Buchstaben des Klartextes ein jeweils anderes Chiffresymbol zugeordnet wird, kann eine Chiffre unbrechbar sein.

Unter der Voraussetzung, daß genügend viel Chiffretext $C$ empfangen worden ist und zusätzlich beliebig viel Zeit zur Verfügung steht, sind fast alle Chiffren brechbar. Der Grund dafür ist, daß es in der Regel nur einen Schlüssel gibt, der $C$ in einen sinnvollen Text dechiffriert. Daraus ergibt sich, daß eine Chiffre so zu konstruieren ist, daß ein Chiffretext unter verschiedenen Schlüsseln zu verschiedenen sinnvollen Nachrichten dechiffriert werden kann (siehe [37]). Wir gehen nun von einem Klartextalphabet

$$A = \{a_1, \ldots, a_n\}$$

aus und konstruieren eine Matrix K

| | $a_1 \ \cdots \ a_n$ |
|---|---|
| $a_1$ | Trage alle Zahlen |
| $\vdots$ | von 1 bis $n^2$ |
| $\vdots$ | zufällig in das |
| $a_n$ | Schema ein |

Zu jedem $a_i \in A, 1 \leq i \leq n$, gibt es jetzt zwei Sätze von Homophonen (zwei Schlüssel), nämlich

$f_1(a_i)$ als Menge der Elemente der $i$-ten Zeile und
$f_2(a_j)$ als Menge der Elemente der $j$-ten Spalte.

Gegeben seien der Klartext $M = m_1 m_2 \ldots$ und eine Dummy-Nachricht $X = x_1 x_2 \ldots$, die ebenfalls einen sinnvollen Klartext darstellt. Dann setze man

$$C = c_1 c_2 \ldots \text{ mit } c_i = K[m_i, x_i], i = 1, 2, \ldots.$$

Es gilt $c_i = f_1(m_i) \cap f_2(x_i)$. $C$ wird mit dem Schlüssel $f_1$ zu $M$ und mit $f_2$ zu $X$ dechiffriert. Nur der legale Empfänger weiß, daß $f_1$ der richtige Schlüssel ist.

**Beispiel 2.3.2** Es sei $n = 5$. Im folgenden wird eine $5 \times 5$ Matrix für das 5-elementige Klartextalphabet {E, I, L, M, S} beschrieben:

| | E | I | L | M | S |
|---|---|---|---|---|---|
| E | 10 | 22 | 18 | 02 | 11 |
| I | 12 | 01 | 25 | 05 | 20 |
| L | 19 | 06 | 23 | 13 | 07 |
| M | 03 | 16 | 08 | 24 | 15 |
| S | 17 | 09 | 21 | 14 | 04 |

Die Nachricht SMILE wird durch die Dummy-Nachricht LIMES wie folgt chiffriert:

| | | | | | |
|---|---|---|---|---|---|
| $M$ = | S | M | I | L | E |
| $X$ = | L | I | M | E | S |
| $C$ = | 21 | 16 | 05 | 19 | 11 |

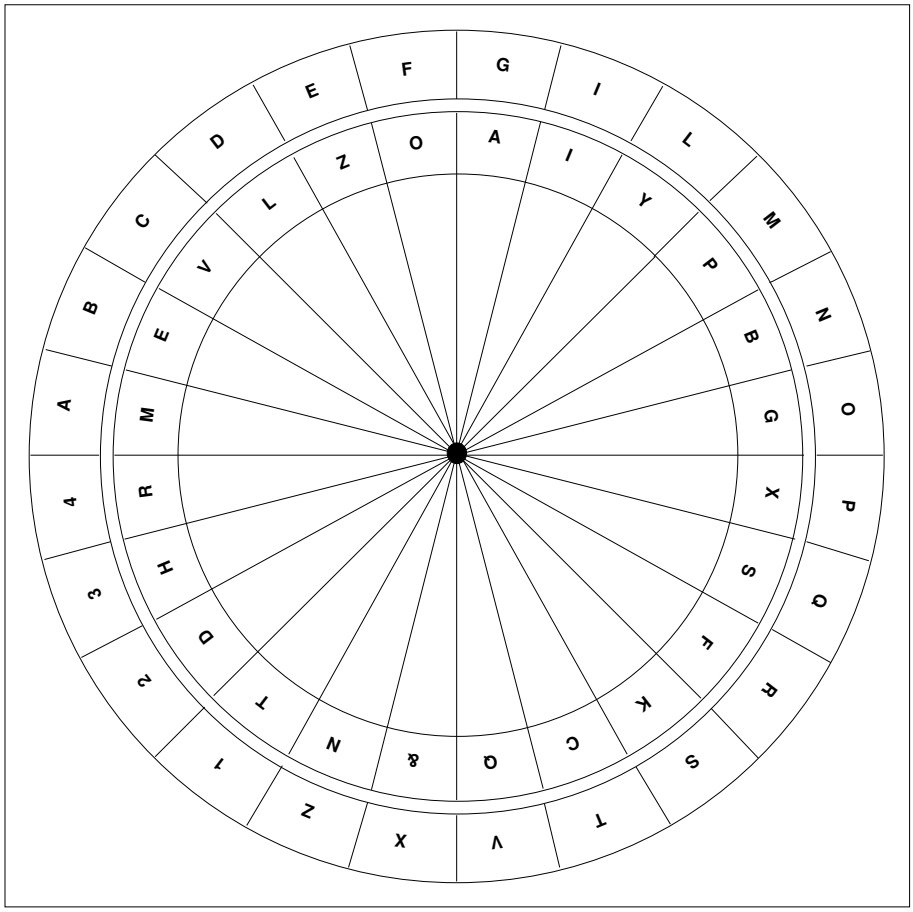
□

## 2.4 Chiffren mit polyalphabetischer Substitution

Chiffren mit polyalphabetischer Substitution verbergen die Verteilung der Buchstaben, indem sie verschiedene Substitutionen benutzen. Leon Battista Alberti beschrieb bereits 1568 eine Chiffrierscheibe (siehe Abbildung auf Seite 23). Sie besitzt 24 mögliche Substitutionen von Klartextbuchstaben des äußeren Ringes in Chiffretextbuchstaben des inneren Ringes. Die Chiffrierung ist dabei abhängig von der Position des beweglichen inneren Ringes.

Die meisten Chiffren mit polyalphabetischer Substitution sind Chiffren mit periodischer Substitution.

**Definition 2.4.1** Es sei $\mathcal{A}$ ein Klartextalphabet und $\mathcal{C}_1, \ldots, \mathcal{C}_d$ seien Chiffretextalphabete. Eine Chiffre mit *periodischer Substitution* ist durch bijektive Abbildungen $f_i : \mathcal{A} \longrightarrow \mathcal{C}_i$, $1 \leq i \leq d$, gegeben. Ein Klartext $M = m_1 \ldots m_d \, m_{d+1} \ldots m_{2d} \ldots$ wird chiffriert durch $E_K(M) = f_1(m_1) \ldots f_d(m_d) \, f_1(m_{d+1}) \ldots f_d(m_{2d}) \ldots$ □



Chiffrierscheibe

Für $d = 1$ ist die Chiffre *monoalphabetisch* und ist somit eine Chiffre mit einfacher Substitution.

**Beispiel 2.4.1** Die *Vigenère-Chiffre* beruht auf verschobenen Alphabeten. Eine Folge von Buchstaben $K = k_1, k_2, \ldots, k_d$ ist der Schlüssel, wobei $k_i$, $1 \leq i \leq d$, die Größe der Verschiebung im $i$-ten Alphabet der Größe $n$ angibt, d. h.

$$f_i(a) = (a + k_i) \bmod n.$$

Bei gleichem Klartext- und Chiffretextalphabet erhält man z. B.

$$\begin{array}{rlll} M = & \text{VORL} & \text{ESUN} & \text{GEN} \\ K = & \text{BAND} & \text{BAND} & \text{BAN} \\ E_K(M) = & \text{WOEO} & \text{FSHQ} & \text{HEA}. \end{array}$$

In jeder Gruppe von jeweils vier Buchstaben wird der erste Buchstabe um $1 \bmod 26$, der zweite um 0, der dritte um $13 \bmod 26$ und der vierte um $3 \bmod 26$ Positionen verschoben. Die Dechiffrierung erfolgt mit Hilfe des Vigenère-Tableaus (siehe Seite 25), das 1868 von Lewis Caroll publiziert wurde (Mathematikprofessor in Oxford und Autor von „Alice im Wunderland“).

Für die Chiffrierung des Klartextbuchstabens $a$ mit dem Schlüsselbuchstaben $k$ findet man den Chiffretextbuchstaben $c$ in Spalte $a$ der Zeile $k$. Die Dechiffrierung des Chiffretextbuchstabens $c$ mit dem Schlüsselbuchstaben $k$ liefert den Klartextbuchstaben $a$ der Spalte $a$, die $c$ in Zeile $k$ enthält. □

**Beispiel 2.4.2** Wir betrachten die *Beaufort-Chiffre*. Die Chiffrierung erfolgt durch

$$f_i(a) = (k_i - a) \bmod n,$$

die Dechiffrierung durch

$$f_i^{-1}(c) = (k_i - c) \bmod n.$$

$f_i$ läßt sich auch als

$$f_i(a) = ((n-1) - a + (k_i + 1)) \bmod n$$

schreiben. Das bedeutet, daß die Beaufort-Chiffre die Reihenfolge der Buchstaben umdreht (A $\rightarrow$ Z, B $\rightarrow$ Y, ...) und sie dann um $k_i + 1$ Positionen nach rechts verschiebt. Wir betrachten speziell $k_i = D$, also

$$f_i(a) = (D - a) \bmod 26.$$

Dies wird auch durch die Tabelle

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{A}$: | 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 |
| $\mathcal{C}$: | D | C | B | A | Z | Y | X | W | V | U | T | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | E |

ausgedrückt. Um eine Beaufort-Chiffre zu chiffrieren bzw. zu dechiffrieren, kann das Vigenère-Tableau benutzt werden. Für die Chiffrierung des Klartextbuchstabens $a$ mit dem Schlüsselbuchstaben $k$ ist der Chiffretextbuchstabe $c$ durch die Zeile gegeben, die den Schlüsselbuchstaben $k$ in der Spalte $a$ enthält. Dies ergibt sich aus der Gleichung $(a + f_i(a)) \bmod n = k_i$, wenn man sie mit $(a + k_i) \bmod n = f_i(a)$ bei der Vigenère-Chiffre vergleicht. Für die Dechiffrierung des Chiffrebuchstabens $c$ mit dem Schlüsselbuchstaben $k$ ist der Klartext $a$ durch die Spalte $a$ gegeben, die den Schlüsselbuchstaben $k$ in der Zeile $c$ enthält. □

**Beispiel 2.4.3** Wir betrachten die *Variante der Beaufort-Chiffre*. Sie ist durch

$$f_i(a) = (a - k_i) \bmod n$$

gegeben. Wegen $(a - k_i) \bmod n = (a + (n - k_i)) \bmod n$ ist die Variante der Beaufort-Chiffre äquivalent zu einer Vigenère-Chiffre mit dem Schlüssel $n - k_i$. Sie ist damit das Inverse der Vigenère-Chiffre. Die beiden Chiffren können paarweise zum Chiffrieren und Dechiffrieren benutzt werden. □

Wir wollen nun die Eindeutigkeitsdistanz für Chiffren mit periodischer Substitution bestimmen. Es sei $s$ die Anzahl der möglichen Schlüssel für jede einfache Substitution und damit $s^d$ die Anzahl der möglichen Schlüssel bei $d$ benutzten Substitutionen. Daraus ergibt sich

$$n \cong \frac{H(K)}{D} = \log_2 s \cdot \frac{d}{D}.$$

Im Vergleich zu einer individuellen Substitutionschiffre werden hier also $d$-mal so viele Buchstaben zum Brechen benötigt. Für Vigenère-Chiffren mit der Periode $d$ und $s = 26$ folgt

$$n \cong \frac{4,7d}{3,2} \cong 1,5d.$$

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

Vigenère-Tableau

| Zeichen | Prozent % | |
|---|---|---|
| A | 4,0 | ******** |
| B | 0,9 | ** |
| C | 6,1 | ************ |
| D | 2,0 | **** |
| E | 4,9 | ********** |
| F | 3,5 | ******* |
| G | 4,0 | ******** |
| H | 3,2 | ****** |
| I | 3,5 | ******* |
| J | 4,6 | ********* |
| K | 5,2 | ********** |
| L | 5,8 | ************ |
| M | 3,2 | ****** |
| N | 4,6 | ********* |
| O | 4,0 | ******** |
| P | 2,0 | **** |
| Q | 3,8 | ******** |
| R | 8,7 | ***************** |
| S | 4,3 | ********* |
| T | 2,0 | **** |
| U | 3,5 | ******* |
| V | 4,0 | ******** |
| W | 1,7 | *** |
| X | 0,6 | * |
| Y | 6,1 | ************ |
| Z | 3,8 | ******** |

Anzahl Zeichen = 346
Koinzidenzindex = 0,0434

Häufigkeitsverteilung der Buchstaben aus Beispiel 2.4.4

Um Chiffren mit periodischer Substitution zu brechen, ist es nötig, zuerst die Periode zu ermitteln. Danach kann mit Hilfe der üblichen Häufigkeitsverteilungen der Buchstaben die Lösung bestimmt werden. Dazu sind zwei Mittel hilfreich, und zwar der Koinzidenzindex und die Methode von Kasiski. Wir gehen zunächst auf den Koinzidenzindex ein, der von *Friedman* [26] eingeführt wurde.

Der *Koinzidenzindex* mißt die Variation der Häufigkeiten der Buchstaben im Chiffretext. Mit wachsender Periode wird der Koinzidenzindex kleiner. Bevor wir ihn genauer definieren, geben wir zunächst, *Sinkov* [98] folgend, das *Maß der Rauheit* $(MR)$, welches die Varianz der Häufigkeiten der einzelnen Buchstaben relativ zu einer gleichmäßigen Verteilung mißt, durch

$$MR = \sum_{i=0}^{n-1}(p_i - \frac{1}{n})^2$$

an. Dabei ist $p_i$ die Wahrscheinlichkeit, daß ein beliebig gewählter Buchstabe aus einem zufälligen Chiffretext gleich dem $i$-ten Buchstaben $a_i$, $i = 0, 1, \ldots, n-1$, des Chiffretextalphabetes ist. Es gilt also $\sum_{i=0}^{n-1} p_i = 1$. Für das übliche Alphabet erhalten wir

$$\begin{aligned} MR &= \sum_{i=0}^{25}(p_i - \frac{1}{26})^2 \\ &= \sum_{i=0}^{25} p_i^2 - \frac{2}{26}\sum_{i=0}^{25} p_i + 26 \cdot \left(\frac{1}{26}\right)^2 \\ &= \sum_{i=0}^{25} p_i^2 - \frac{2}{26} + \frac{1}{26} \\ &= \sum_{i=0}^{25} p_i^2 - 0,03846. \end{aligned}$$

Wäre $d = 1$, so kämen alle $p_i$ aus der Tabelle für die relativen Häufigkeiten auf Seite 20 vor, ggf. in einer anderen Reihenfolge. Da die Periode $d$ unbekannt ist, sind jedoch auch die $p_i$ unbekannt und $MR$ kann nicht berechnet werden. Aber unter Benutzung der Häufigkeitsverteilung der Buchstaben im Chiffretext kann $MR$ mit Hilfe von

$$MR + 0,03846 = \sum_{i=0}^{25} p_i^2$$

abgeschätzt werden. Aus der Summendarstellung ergibt sich, daß dies die Wahrscheinlichkeit ist, daß zwei beliebig gewählte Buchstaben aus einem beliebigen Chiffretext gleich sind. Es sei ein Chiffretext $C$ mit der Länge $N$ gegeben. Dann ist

$$\binom{N}{2} = \frac{N(N-1)}{2}$$

die Anzahl der Buchstabenpaare, die aus $C$ gewählt werden können. Weiterhin sei $F_i$ die Häufigkeit des $i$-ten Buchstabens im Chiffretext, so daß

$$\sum_{i=0}^{25} F_i = N$$

gilt. Dann folgt, daß

$$\frac{F_i \cdot (F_i - 1)}{2}$$

die Anzahl der Paare ist, bei denen beide Buchstaben genau dem $i$-ten Chiffretextbuchstaben gleichen. Der *Koinzidenzindex*

$$IC = \frac{\sum_{i=0}^{25} F_i(F_i - 1)}{N(N-1)}$$

gibt also die relative Häufigkeit dafür an, daß zwei zufällig gewählte Buchstaben aus $C$ gleich sind. Nach der obigen Überlegung ist dies eine Approximation für $\sum_{i=0}^{25} p_i^2$. Damit ist $IC$ auch eine Schätzung für $MR + 0,03846$. Der Koinzidenzindex ist aber im Gegensatz zum Maß der Rauheit aus dem Schlüsseltext zu berechnen.

Für $d = \infty$ hat das Maß der Rauheit $MR$ den Wert 0, da alle Buchstaben gleichwahrscheinlich vorkommen (alle $p_i = \frac{1}{26}$). Es steigt bis auf 0,03887 (bzw. 0,02706) an bei $d = 1$ und der deutschen (bzw. englischen) Sprache ($p_i$ siehe Tabelle auf Seite 20, $\sum_{i=0}^{25} p_i^2 = 0,07732583$ für Deutsch bzw. $\sum_{i=0}^{25} p_i^2 = 0,0655219$ für Englisch). Entsprechend nimmt der Koinzidenzindex $IC$ die Werte 0,03846 für $d = \infty$ bzw. 0,07733 (deutsch) oder 0,06552 (englisch) für $d = 1$ an. Für Chiffren mit der Periode $d$ kann der Erwartungswert des Koinzidenzindex durch

$$\frac{1}{d} \cdot \frac{N-d}{N-1} \cdot 0,07733 + \frac{d-1}{d} \cdot \frac{N}{N-1} \cdot 0,03846$$

approximiert werden [40] (für Deutsch, für Englisch muß 0,07733 durch 0,06552 ersetzt werden). Nachfolgend sind einige Werte für den Erwartungswert ($N = 100$) angegeben:

| $d$ | $IC$ (deutsch) | $IC$ (englisch) |
|---|---|---|
| 1 | 0,07733 | 0,06552 |
| 2 | 0,05770 | 0,05185 |
| 3 | 0,05115 | 0,04730 |
| 4 | 0,04788 | 0,04502 |
| 5 | 0,04592 | 0,04365 |
| 6 | 0,04461 | 0,04274 |
| 7 | 0,04368 | 0,04209 |
| 8 | 0,04297 | 0,04160 |
| 9 | 0,04243 | 0,04122 |
| 10 | 0,04199 | 0,04092 |
| $\infty$ | 0,03846 | 0,03846 |

Die Schätzung der Periode einer gegebenen Chiffre erfolgt nun durch

(1) Messung der Häufigkeiten $F_i$ der Buchstaben des Chiffretextes,
(2) Berechnung des Koinzidenzindex $IC$ und danach
(3) Vergleich mit dem oben ermittelten Erwartungswert des Koinzidenzindex.

Da der Koinzidenzindex nur eine statistische Größe ist, ergibt sich nicht notwendigerweise die exakte Periode. Wenn außerdem eine einfache Substitution innerhalb der Periode $d$ mehrfach auftritt, sind die obigen Überlegungen nicht mehr richtig. Im allgemeinen wird

dann der Koinzidenzindex größer ausfallen als es der Periode entspricht. Die Schätzung der Periode wird in diesem Fall also zu klein sein. Der Koinzidenzindex liefert jedoch zumindestens einen Hinweis auf die Größe der Periode.

Nun betrachten wir die *Kasiski-Methode* (1863 von *Friedrich W. Kasiski* [46], einem preußischen Offizier, eingeführt). Zur Bestimmung der exakten Periode werden alle Wiederholungen des Chiffretextes analysiert. Es sei z. B. ein Text mit der Vigenère-Chiffre und dem Schlüssel MEPH chiffriert.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M =$ | I | C | H | B | I | N | D | E | R | G | E | I | S | T | D | E | R | S | T | ... |
| $K =$ | M | E | P | H | M | E | P | H | M | E | P | H | M | E | P | H | M | E | P | ... |
| $E_K(M) =$ | U | G | W | I | U | R | S | L | D | K | T | P | E | X | S | L | D | W | I | ... |

Dieser Chiffretext enthält zwei Wiederholungen der Zeichenfolge SLD mit einem Abstand von acht Buchstaben. Wiederholungen kommen im Chiffretext vor, wenn das Klartextmuster sich in einem Abstand wiederholt, der ein Vielfaches der Schlüssellänge ist. Zufällige Wiederholungen von mehr als zwei Chiffresymbolen sind unwahrscheinlich. Bei $n$ Chiffretextwiederholungen mit den Intervallängen $I_j$, $1 \leq j \leq n$, ist die Periode wahrscheinlich die Zahl, die die meisten $I_j$ teilt. Im obigen Beispiel beträgt die Intervallänge $I_1 = 8$, und daraus ergibt sich eine mutmaßliche Periode von 1, 4, oder 8 (hier 4). Zur Bestätigung der so bestimmten Periode ist der oben beschriebene Koinzidenzindex nützlich. Es sei $c_1\, c_2 \ldots$ der Chiffretext. Man berechnet für jeden der Texte

$$\begin{array}{llll} 1: c_1 & c_{d+1} & c_{2d+1} & \ldots \\ 2: c_2 & c_{d+2} & c_{2d+2} & \ldots \\ & \ldots & & \\ d: c_d & c_{2d} & c_{3d} & \ldots \end{array}$$

den Koinzidenzindex. Wenn jede Folge mit demselben Schlüssel chiffriert worden ist, haben alle einen Koinzidenzindex $IC \cong 0,07733$. In diesem Fall ist die Periode richtig bestimmt worden, und jeder Teil muß als einfache Substitution einzeln analysiert werden.

**Beispiel 2.4.4** Wir betrachten den Chiffretext:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ZHYME | ZVELK | OJUBW | CEYIN | CUSML | RAVSR | YARNH | CEARI | UJPGP | VARDU |
| QZCGR | NNCAW | JALUH | GJPJR | YGEGQ | FULUS | QFFPV | EYEDQ | GOLKA | LVOSJ |
| TFRTR | YEJZS | RVNCI | HYJNM | ZDCRO | DKHCR | MMLNR | FFLFN | QGOLK | ALVOS |
| JWMIK | QKUBP | SAYOJ | RRQYI | NRNYC | YQZSY | EDNCA | LEILX | RCHUG | IEBKO |
| YTHGV | VCKHC | JEQGO | LKALV | OSJED | WEAKS | GJHYC | LLFTY | IGSVT | FVPMZ |
| NRZOL | CYUZS | FKOQR | YRYAR | ZFGKI | QKRSV | IRCEY | USKVT | MKHCR | MYQIL |
| XRCRL | GQARZ | OLKHY | KSNFN | RRNCZ | TWUOC | JNMKC | MDEZP | IRJEJ | W |

Daraus entnehmen wir die Häufigkeitsverteilung der Buchstaben zur Bestimmung des Koinzidenzindex. Die entsprechende Tabelle ist auf Seite 25 dargestellt. Die Berechnung des Koinzidenzindexes ergibt $0,0434$.

| Zeichen | Prozent % | |
|---|---|---|
| A | 0,0 | |
| B | 0,0 | |
| C | 6,0 | ********** |
| D | 1,7 | *** |
| E | 4,3 | ********* |
| F | 5,2 | ********** |
| G | 2,6 | ***** |
| H | 0,9 | ** |
| I | 5,2 | ********** |
| J | 8,6 | ****************** |
| K | 13,8 | ***************************** |
| L | 1,7 | *** |
| M | 0,9 | ** |
| N | 0,9 | ** |
| O | 0,0 | |
| P | 2,6 | ***** |
| Q | 1,7 | *** |
| R | 9,5 | ******************** |
| S | 0,0 | |
| T | 0,9 | ** |
| U | 5,2 | ********** |
| V | 11,2 | ********************** |
| W | 1,7 | *** |
| X | 1,7 | *** |
| Y | 4,3 | ********* |
| Z | 9,5 | ******************** |

Anzahl Zeichen der Folge = 116
Koinzidenzindex = 0,06747

Folge (1)

| Zeichen | Prozent % | |
|---|---|---|
| A | 9,6 | ******************** |
| B | 0,9 | ** |
| C | 1,7 | *** |
| D | 2,6 | ***** |
| E | 10,4 | ********************** |
| F | 2,6 | ***** |
| G | 2,6 | ***** |
| H | 8,7 | ****************** |
| I | 5,2 | ********** |
| J | 0,0 | |
| K | 0,9 | ** |
| L | 1,7 | *** |
| M | 1,7 | *** |
| N | 9,6 | ******************** |
| O | 12,2 | ************************* |
| P | 1,7 | *** |
| Q | 0,0 | |
| R | 4,3 | ********* |
| S | 8,7 | ****************** |
| T | 5,2 | ********** |
| U | 4,3 | ********* |
| V | 0,9 | ** |
| W | 0,9 | ** |
| X | 0,0 | |
| Y | 3,5 | ******* |
| Z | 0,0 | |

Anzahl Zeichen der Folge = 115
Koinzidenzindex = 0,06499

Folge (2)

| Zeichen | Prozent % | |
|---|---|---|
| A | 2,6 | ***** |
| B | 1,7 | *** |
| C | 10,4 | ********************** |
| D | 1,7 | *** |
| E | 0,0 | |
| F | 2,6 | ***** |
| G | 7,0 | ************** |
| H | 0,0 | |
| I | 0,0 | |
| J | 5,2 | ********** |
| K | 0,9 | ** |
| L | 13,9 | ***************************** |
| M | 7,0 | ************** |
| N | 3,5 | ******* |
| O | 0,0 | |
| P | 1,7 | *** |
| Q | 9,6 | ******************** |
| R | 12,2 | ************************* |
| S | 4,3 | ********* |
| T | 0,0 | |
| U | 0,9 | |
| V | 0,0 | |
| W | 2,6 | ***** |
| X | 0,0 | |
| Y | 10,4 | ********************** |
| Z | 1,7 | *** |

Anzahl Zeichen der Folge = 115
Koinzidenzindex = 0,07597

Folge (3)

Aus $IC = 0,0434$ ergibt sich die Schätzung $d = 5$. Wir bestimmen nun die Periode nach der Kasiski-Methode. In diesem Chiffretext treten drei Wiederholungen des Chiffretextes QGOLKALVOSJ auf. Der Abstand zwischen dem ersten und zweiten Auftreten beträgt 51, zwischen dem zweiten und dritten Auftreten 72 Buchstaben. Da drei der einzige gemeinsame Teiler von 51 und 72 ist, ist die Größe der Periode höchstwahrscheinlich drei.

Für die Chiffretextfolgen

$$\begin{array}{llll} (1) & c_1 & c_4 & c_7 \ldots \\ (2) & c_2 & c_5 & c_8 \ldots \\ (3) & c_3 & c_6 & c_9 \ldots \end{array}$$

muß jetzt jeweils der $IC$ berechnet werden. Für die Folge (1) bis (3) sind die Häufigkeitsverteilung auf Seite 29 dargestellt. Jede der drei Folgen hat einen Koinzidenzindex nahe oder größer als $0,0659$. Dies bestätigt die Annahme, daß $d = 3$ ist.

Als nächstes muß der Typ der Chiffre bestimmt werden. Zunächst muß festgestellt werden, ob es sich um eine Vigenère- oder Beaufort-Chiffre handelt. Die Folge (2) zeigt eine Häufigkeitsverteilung, die der Verteilung der Buchstaben in der englischen Sprache (siehe Tabelle auf Seite 20) sehr ähnlich ist. Daraus folgt, daß es sich wohl um eine Vigenère-Chiffre mit A als zweitem Buchstaben des Schlüssels handelt. In den Folgen (1) und (3) sind E, T und A die häufigsten Buchstaben. Die Verteilung der Spitzen läßt darauf schließen, daß in der Folge (1) A auf R bzw. in der Folge (3) A auf Y abgebildet wird. Damit lautet der Schlüssel RAY. Durch die Anwendung dieses Schlüssels auf das Vigenère-Tableau (Tabelle auf Seite 25) ergibt sich der folgende Klartext:

> I have been told by learned sources that when a truely great musician plays and his playing sounds so free and spontaneous that the listener has no idea of the amount of nonspontaneous work, study, scholarship analysis and planning required to achieve these spontaneous effects. I shall not argue the point, I only wish to say that if it is true it leads me to the amazing realization that spontaneity does not come by itself.

Dieser Text konnte nur deshalb so einfach gebrochen werden, weil der zugehörige Schlüssel sehr kurz ist. □

Die Sicherheit einer Substitutionschiffre wächst mit zunehmender Schlüssellänge. In einer *Chiffre mit laufendem Schlüssel (running-key cipher)* ist der Schlüssel mindestens genauso lang wie der Klartext. Die Kasiski-Methode hat dann keinen Erfolg, sofern der Schlüssel nicht wiederholt wird. Gern wird der Text eines Buches als Schlüssel einer Substitutionschiffre benutzt, die auf verschobenem Alphabet beruht, die also eine nichtperiodische Vigenère-Chiffre ist. Der Schlüssel wird durch den Titel des Buches und die Startposition spezifiziert. Falls der Schlüssel jedoch noch einmal eingesetzt wird und dadurch der Chiffretext länger wird als der Schlüssel, kann die Kasiski-Methode eventuell zum Erfolg führen. Aber auch bei einmaliger Anwendung des Schlüssels ist eine perfekte Geheimhaltung nicht gewährleistet, da der Text als deutscher oder englischer Text eine sprachabhängige Redundanz besitzt.

Mit dem *Ansatz von Friedman* [25] kann gegebenenfalls eine solche Chiffre gebrochen werden. Große Teile der Buchstaben des Chiffretextes entsprechen hierbei Chiffrierungen, bei denen sowohl Klartext- als auch Schlüsselbuchstaben in die Kategorie häufigsten Buchstaben fallen. Im Englischen sind dies die Buchstaben E, T, A, O, N, I, R, S und H. Ausgegangen wird von der Annahme, daß alle Chiffretextbuchstaben Paaren von Klartext- und

Schlüsselbuchstaben der häufigsten Kategorie entsprechen. Diese ursprünglichen Annahmen werden mit Di- bzw. Trigrammverteilungen und wahrscheinlichen Worten in Beziehung gesetzt, um daraus die tatsächlichen Paare zu bestimmen.

**Beispiel 2.4.5** Mit dem Klartext „The treasure ist buried . . . “ und dem Schlüsseltext „The second cipher is an . . . “ erhalten wir

$$\begin{aligned} M &= \underline{\mathrm{T}}\,\mathrm{H}\underline{\mathrm{E}}\,\underline{\mathrm{T}}\,\underline{\mathrm{R}}\,\mathrm{E}\,\underline{\mathrm{A}}\,\underline{\mathrm{S}}\,\mathrm{U}\,\mathrm{R}\,\underline{\mathrm{E}}\;\mathrm{I}\;\underline{\mathrm{S}}\,\mathrm{B}\,\mathrm{U}\,\underline{\mathrm{R}}\,\underline{\mathrm{I}}\,\underline{\mathrm{E}}\,\mathrm{D}\ldots \\ K &= \underline{\mathrm{T}}\,\mathrm{H}\underline{\mathrm{E}}\,\underline{\mathrm{S}}\,\underline{\mathrm{E}}\,\mathrm{C}\,\underline{\mathrm{O}}\,\underline{\mathrm{N}}\,\mathrm{D}\,\mathrm{C}\;\underline{\mathrm{I}}\;\mathrm{P}\,\underline{\mathrm{H}}\,\mathrm{E}\,\mathrm{R}\,\underline{\mathrm{I}}\,\underline{\mathrm{S}}\,\underline{\mathrm{A}}\,\mathrm{N}\ldots \\ E_K(M) &= \mathrm{M}\,\mathrm{O}\,\mathrm{I}\,\mathrm{L}\,\mathrm{V}\,\mathrm{G}\,\mathrm{O}\,\mathrm{F}\,\mathrm{X}\,\mathrm{T}\,\mathrm{M}\,\mathrm{X}\,\mathrm{Z}\,\mathrm{F}\,\mathrm{L}\,\mathrm{Z}\,\mathrm{A}\,\mathrm{E}\,\mathrm{Q}\ldots \end{aligned}$$

Die Paare der häufigsten Verteilungen sind hier unterstrichen. Wir betrachten die ersten drei Chiffretextbuchstaben MOI. Bei einer (nichtperiodischen) Vigenère-Chiffre gibt es für jeden Chiffretextbuchstaben 26 mögliche Paare von Klartext- und Chiffretextbuchstaben. In der folgenden Tabelle stehen in der ersten Zeile die Klartextbuchstaben, in der zweiten die entsprechenden Schlüsselbuchstaben für den Chiffretextbuchstaben M, in der dritten und vierten die für O bzw. I.

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

Auch hier sind die Paare der häufigsten Verteilungen durch Unterstreichung der entsprechenden Schlüsselbuchstaben markiert.

Für die Chifftretextzeichen MOI ergeben sich also folgende Paare von Klartext- und Schlüsselbuchstaben der häufigsten Kategorie:

| M | O | I |
|---|---|---|
| E-I | A-O | A-I |
| I-E | O-A | I-A |
| T-T | H-H | E-E |
| | | R-R . |

Es existieren somit $3 \cdot 3 \cdot 4 = 36$ mögliche Kombinationen der Paare:

| | | | | |
|---|---|---|---|---|
| Klartext: | EAA | EAI | EAE | EAR . . . THE . . . THR |
| Schlüssel: | IOI | IOA | IOE | IOR . . . THE . . . THR |
| Chiffre: | MOI | MOI | MOI | MOI . . . MOI . . . MOI |

Einerseits sind viele der Trigramme des Klartextes und des Schlüssels sehr unwahrscheinlich. Andererseits ist die Zuordnung des Klartextes THE zu dem Schlüssel THE sehr wahrscheinlich. Daraus läßt sich schließen, daß das erste Wort des Schlüssels sowie des Klartextes richtig geraten worden ist. □

Als *one-time pad* werden Schlüssel bezeichnet, die eine zufällige Folge von Buchstaben sind und nicht wiederholt werden. Wir betrachen dazu

**Beispiel 2.4.6** Es handelt sich hier um die *Vernam-Chiffre*. Es sei $M = m_1\, m_2\, \ldots$ ein Klartext und $K = k_1\, k_2\, \ldots$ ein Schlüssel aus Bits. Dann setze man

$$C = E_K(M) = c_1\, c_2\, \ldots \text{ mit } c_i = (m_i + k_i) \bmod 2,\ i = 1, 2, \ldots\ .$$

Damit entspricht die Vernam-Chiffre einer Vigenère-Chiffre mit einem Chiffretextalphabet $C = \{0, 1\}$. Die Chiffre kann durch das exklusive Oder ($\oplus$) dargestellt werden, da $c_i = m_i \oplus k_i$ gilt. Das läßt sich direkt durch eine Hardwareschaltung realisieren und ist daher sehr schnell zu berechnen. Die Dechiffrierung erfolgt durch dieselbe Operation, und zwar

$$c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i\ . \quad \square$$

Falls der Schlüssel der Vernam-Chiffre mehrmals benutzt wird, ist die Chiffre äquivalent zu einer Chiffre mit laufendem Schlüssel $K$. Seien nämlich $M$ und $M'$ Klartexte, die mit demselben Schlüssel $K$ die Chiffretexte $C$ und $C'$ liefern. Dann gilt

$$c_i = m_i \oplus k_i \text{ bzw. } c'_i = m'_i \oplus k_i,\ i = 1, 2, \ldots\ .$$

Wird nun $C''$ durch

$$\begin{aligned} c''_i &= c_i \oplus c'_i \\ &= m_i \oplus k_i \oplus m'_i \oplus k_i \\ &= m_i \oplus m'_i \end{aligned}$$

gebildet, so ist das gleichbedeutend mit einer Chiffrierung des Klartextes $M$ durch den Schlüssel $M'$. $C''$ kann eventuell durch den Ansatz von Friedman gebrochen werden. Die Berechnung des Schlüssels $K$ erfolgt dann durch $k_i = c_i \oplus m_i$.

## 2.5 Chiffren mit Polygramm-Substitution

Die Chiffren der vorhergehenden Abschnitte haben jeweils einen einzelnen Klartextbuchstaben chiffriert. Chiffren mit Polygramm-Substitution ersetzen dagegen zusammenhängende Folgen von Buchstaben, also Blöcke von Buchstaben. Dadurch wird die Bedeutung der Häufigkeitsverteilung der einzelnen Buchstaben verringert.

Wir betrachten zunächst die *Playfair-Chiffre* (1854). Sie wurde von den Briten noch während des ersten Weltkriegs benutzt. Sie ist eine Digramm-Substitutionschiffre. Der Schlüssel ist eine $5 \times 5$ Matrix, in die die Buchstaben des Alphabets (ohne J) eingetragen werden. Jedes Klartextbuchstabenpaar $m_1\ m_2$wird nach folgenden Regeln durch ein Chiffretextpaar $c_1\ c_2$ chiffriert.

(1) Wenn sich $m_1$ und $m_2$ in derselben Zeile befinden, dann steht $c_1$ rechts von $m_1$ und $c_2$ rechts von $m_2$ (wobei die erste Spalte auch rechts von der letzten Spalte steht).

(2) Wenn sich $m_1$ und $m_2$ in derselben Spalte befinden, dann steht $c_1$ unter $m_1$ und $c_2$ unter $m_2$ (wobei die erste Zeile auch unter der letzten Zeile steht).

(3) Wenn sich $m_1$ und $m_2$ in verschiedenen Zeilen bzw. Spalten befinden, dann bilden $m_1$ und $m_2$ zwei Ecken eines Rechtecks. Die beiden anderen Eckpunkte werden durch $c_1$ und $c_2$ bestimmt. Hierbei steht $c_1$ in derselben Zeile wie $m_1$ bzw. $c_2$ in derselben Zeile wie $m_2$.

(4) Wenn $m_1 = m_2$ gilt, dann muß ein vorher festgelegter Dummy-Buchstabe in den Klartext eingefügt werden.
(5) Wenn der Klartext eine ungerade Anzahl von Buchstaben enthält, dann muß am Ende des Klartextes ebenfalls der Dummy-Buchstabe angehängt werden.

**Beispiel 2.5.1** Die Matrix werde zunächst durch das Schlüsselwort

BRAUNSCHWEIGERSTUDENTENSINDKLUG

zeilenweise fortlaufend von links nach rechts gefüllt. Duplikate werden nur beim ersten Auftreten eingesetzt. Der verbleibende Rest der Buchstaben wird in beliebiger Reihenfolge angehängt:

$$\begin{pmatrix} \text{B} & \underline{\text{R}} & \text{A} & \text{U} & \text{N} \\ \text{S} & \text{C} & \text{H} & \text{W} & \text{E} \\ \text{I} & \text{G} & \text{T} & \text{D} & \text{K} \\ \underline{\text{L}} & \text{F} & \text{Z} & \text{Y} & \text{M} \\ \text{X} & \text{Q} & \text{O} & \text{P} & \text{V} \end{pmatrix}$$

Es sei $M$=VORLESUNGENX (X Dummy-Buchstabe) der Klartext. Dann ergibt sich der Chiffretext $C$=XPBFSCNBKCBV. Für das zweite Buchstabenpaar (RL) muß die Regel (3) benutzt werden. Die entsprechenden Buchstaben sind in der obigen Matrix markiert und liefern den Chiffretext BF. □

Die *Hill-Chiffre* (1929) [39] führt eine lineare Abbildung auf $d$ Klartextbuchstaben durch, um $d$ Chifftretextbuchstaben zu erhalten. Als Schlüssel wird eine $d$-dimensionale quadratische Matrix $A$ benutzt, zu der die inverse Matrix $A^{-1}$ existiert. $A$ und $A^{-1}$ dürfen nur Zahlen modulo $n$ beinhalten, wobei $n$ die Größe des Alphabets ist. Die Chiffrierung erfolgt durch

$$C = E_K(M) = MA,$$

die Dechiffrierung durch

$$D_K(C) = CA^{-1} = MAA^{-1} = M.$$

**Beispiel 2.5.2** Es sei $d = 2$ und $n = 26$ (übliches Alphabet). Wir wählen

$$A = \begin{pmatrix} 3 & 2 \\ 3 & 5 \end{pmatrix}$$

Dann ist

$$A^{-1} = \begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix} \text{ (modulo 26).}$$

Wir betrachten den Klartext KARL, der als $M = (11, 1, 18, 12)$ kodiert wird. Wir erhalten

$$(11, 1)\begin{pmatrix} 3 & 2 \\ 3 & 5 \end{pmatrix} = (10, 1) \text{ und } (18, 12)\begin{pmatrix} 3 & 2 \\ 3 & 5 \end{pmatrix} = (12, 18)$$

Es folgt $C = (10, 1, 12, 18)$ und damit der Chiffretext JALR. □

Die Kryptoanalyse dieser Chiffre ist nicht leicht, wenn nur der Chiffretext bekannt ist. Die Chiffre ist jedoch oft zu brechen, falls einige Buchstaben des Klartextes bekannt sind. Diese kann der Kryptoanalytiker erhalten, wenn er z. B. die Chiffrierung eines von ihm gewählten Klartextes veranlassen kann. Hat der Kryptoanalytiker die Klartexte $M_1 = (m_1, m_2)$ und $M_2 = (m_3, m_4)$ mit den zugehörigen Chiffretexten $C_1 = (c_1, c_2)$ und $C_2 = (c_3, c_4)$, dann kann er

$$M = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \text{ und } C = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$$

bilden. Ist die Matrix $M$ (modulo 26) regulär, so kann die Schlüsselmatrix $A$ durch $A = M^{-1}C$ berechnet werden. Die Berechnung ist durch einen Algorithmus der Ordnung $O(d^3)$ möglich.

## 2.6 Produktchiffren und der DES

Eine Produktchiffre ist die Komposition von $t$ Funktionen (Chiffren) $F_1, F_2, \ldots, F_t$, wobei jedes der $F_i,\ 1 \leq i \leq t,$ eine Substitutionschiffre oder Transpositionschiffre ist. Als Beispiel wollen wir hier den Data Encryption Standard (DES) betrachten. Er wurde vom National Bureau of Standards 1977 angekündigt [59] und von IBM entwickelt. Der DES chiffriert je 64-Bit-Blöcke von Daten mit einem 56-Bit-Schlüssel. Es ist umstritten, ob 56 Bit als Schlüssel genügen, um diese Chiffre berechnungs-sicher zu machen. Der Algorithmus arbeitet nach dem in Bild 2.6.2 auf Seite 37 beschriebenen Schema. Im wesentlichen erfolgt auch die Dechiffrierung nach diesem Schema. Dabei wird zunächst ein Eingabeblock $T$ einer initialen Permutation $IP$ unterworfen, also $T_0 = IP(T)$. $IP$ und die später benutzte inverse Permutation $IP^{-1}$ sind durch die folgenden Tafeln gegeben:

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

**Tafel 2.6.1:** Initiale Permutation $IP$

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

**Tafel 2.6.2:** Finale Permutation $IP^{-1}$

Zur Verdeutlichung merken wir an, daß der Eingabeblock $T = t_1\, t_2\, \ldots\, t_{64}$ durch die initiale Permutation in

$$IP(T) = t_{58}\, t_{50}\, t_{42}\, \ldots\, t_{15}\, t_7$$

abgebildet wird. Die Tafeln müssen also zeilenweise von links nach rechts gelesen werden.

$T_0$ durchläuft dann, wie in Bild 2.6.2 von Seite 37 dargestellt, 16 Iterationen einer Funktion $f$. Um das Endergebnis zu erhalten, wird das letzte Zwischenresultat der inversen Permutation $IP^{-1}$ übergeben.

Die Funktion $f$ kombiniert Substitution und Transposition. Es sei $T_i = L_i\, R_i$ mit $L_i = t_1\, t_2\, \ldots\, t_{32}$ und $R_i = t_{33}\, t_{34}\, \ldots\, t_{64}$ das Ergebnis der $i$-ten Iteration, $1 \leq i \leq 16$. Dann gilt

$$L_i = R_{i-1} \text{ und } R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

wobei $\oplus$ die Bildung des exklusiven Oder bedeutet und $K_i$ ein später beschriebener 48-Bit-Schlüssel ist. Nach der letzten Iteration wird $R_{16}$ nicht mehr mit $L_{16}$ vertauscht, d. h. $T_{16} = R_{16}\, L_{16}$.

Als nächstes betrachten wir die Berechnung der Funktion $f$. Dies geschieht mit Hilfe der sogenannten $S$-Boxen. Die Berechnung wird durch Bild 2.6.1 skizziert (siehe Seite 36). Es wird zunächst $R_{i-1}$ durch die Tafel $E$ zu einem 48-Bit-Block $E(R_{i-1})$ erweitert. Dies geschieht mit der folgenden Bit-Auswahl-Tafel.

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

**Tafel 2.6.3:** Bit-Auswahl Tafel $E$

Aus $r_1 r_2 r_3 r_4 r_5 \ldots r_{32}$ wird also $r_{32} r_1 r_2 r_3 r_4 r_5 r_4 \ldots r_{32} r_1$. Anschließend wird $E(R_{i-1}) \oplus K_i$ gebildet (wie $K_i$ berechnet wird, wird weiter unter beschrieben) und das Ergebnis in acht 6-Bit-Blöcke $B_1, B_2, \ldots, B_8$ aufgeteilt, also

$$E(R_{i-1}) \oplus K_i = B_1\, B_2 \ldots B_8.$$

Jeder dieser 6-Bit-Blöcke $B_j = b_1\, b_2\, b_3\, b_4\, b_5\, b_6$, $1 \leq j \leq 8$, wird an die entsprechende $S$-Box $S_j$ übergeben. Eine $S$-Box ist eine Substitution, die einen 4-Bit-Block $S_j(B_j)$ als Ergebnis liefert. Dabei ist $S_j$ durch eine $4 \times 16$-Matrix dargestellt (siehe Tafel 2.6.4 auf Seite 38), und $S_j(B_j)$ ist die Zahl, die der 4-Bit-Darstellung desjenigen Elements in dieser Matrix entspricht, dessen Zeile aus den Bits $b_1\ b_6$ (dezimal zwischen 0 und 3) und dessen Spalte aus den Bits $b_2\ b_3\ b_4\ b_5$ (dezimal zwischen 0 und 15) berechnet wird (Es sei $B_1 = 010011$. Daraus ergibt sich die Zeile 1 (binär 01) und die Spalte 9 (binär 1001) und damit $S_1(B_1) = 6_{10} = 0110_2$ ). Danach wird die Permutation $P$ auf die Konkatenation aller 4-Bit-Blöcke angewandt. $P$ wird durch die Tafel 2.6.5 auf Seite 38 definiert. Als Ergebnis der Funktion $f$ ergibt sich also

$$f(R_{i-1}, K_i) = P(S_1(B_1)\, S_2(B_2) \ldots S_8(B_8)).$$

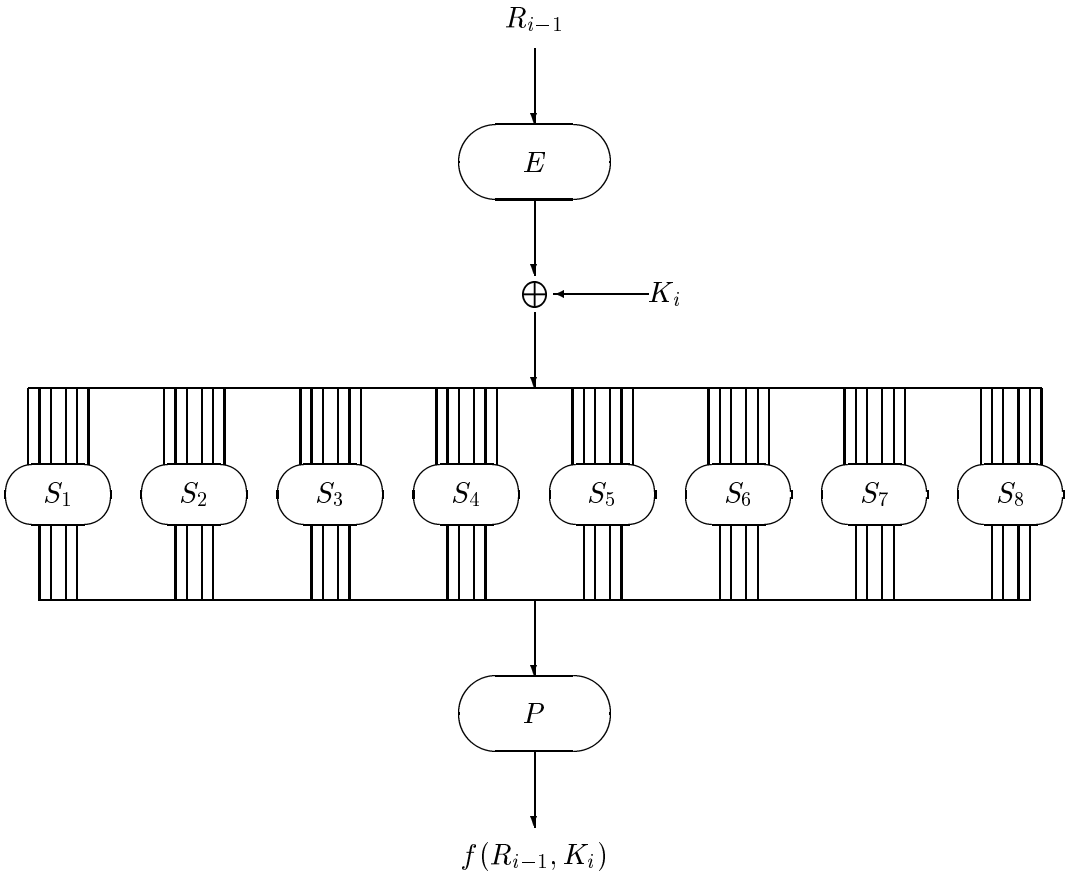



**Bild 2.6.1:** Berechnung von $f(R_{i-1}, K_i)$

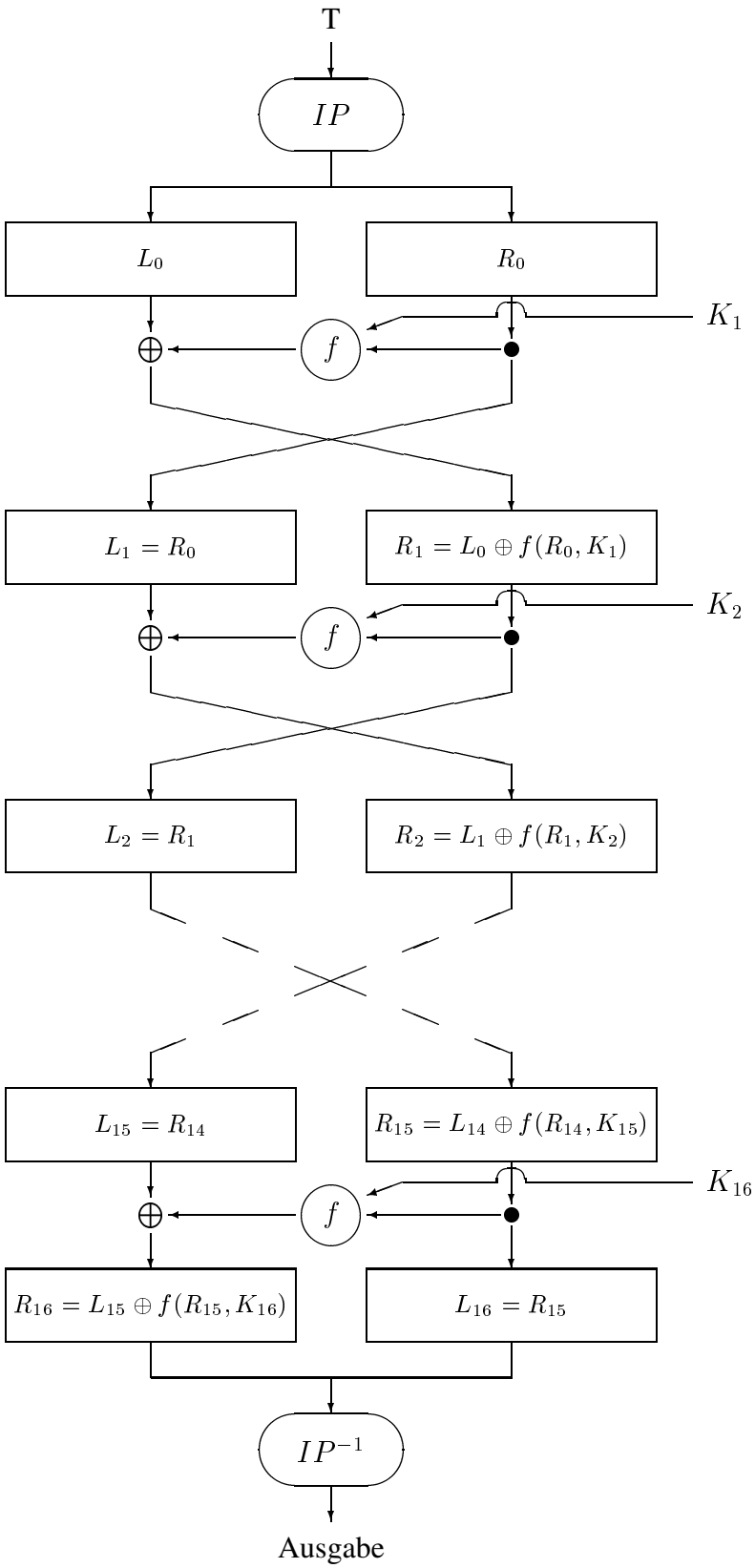



**Bild 2.6.2:** DES Verschlüsselungsalgorithmus

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

**Tafel 2.6.4:** Auswahlfunktion ($S$-Boxes)

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

**Tafel 2.6.5:** Permutation $P$

Es fehlt noch die Berechnung der Schlüssel $K_i$, $1 \leq i \leq 16$, aus dem initialen Schlüssel $K$. $K$ besteht aus 56 Bits. Jede Iteration benutzt einen anderen 48-Bit-Schlüssel $K_i$. Die Berechnung ist in Bild 2.6.3 von Seite 40 graphisch dargestellt. Zunächst wird $K$ durch acht Parity-Bits (an den Positionen 8, 16, ..., 64) auf 64 Bits erweitert und in dieser Form an $PC-1$ übergeben. Die Permutation $PC-1$ ($PC =$ permuted choice) streicht diese Parity-Bits wieder und vertauscht die übrigen 56 Bits, wie es durch die Tafel 2.6.6 angegeben wird.

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

**Tafel 2.6.6:** Schlüsselpermutation $PC-1$

Das Ergebnis $(PC-1)(K)$ wird in $C_0$ und $D_0$ zu je 28 Bits aufgeteilt. $C_i$ und $D_i$, $i = 1, 2, \ldots, 16$, werden dann durch

$$C_i = LS_i(C_{i-1}) \text{ bzw. } D_i = LS_i(D_{i-1})$$

berechnet. $LS_i$ ist eine zirkuläre Linksverschiebung um eine Anzahl von Positionen, die durch die folgende Tafel angegeben ist:

| Iteration $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Anzahl von zirkulären Linksverschiebungen | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

**Tafel 2.6.7:** Anzahl der Linksverschiebungen für verschiedene Schlüssel

Der Schlüssel $K_i$, $i = 1, 2, \ldots, 16$, ergibt sich nun durch

$$K_i = (PC-2)(C_i, D_i).$$

Dabei ist die Permutation $PC-2$ durch die folgende Tafel dargestellt:

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

**Tafel 2.6.8:** Schlüsselpermutation $PC-2$

Durch $PC-2$ werden die Bits auf den Positionen 9, 18, 22, 25, 35, 38, 43 und 54 entfernt. Insgesamt ist damit die Chiffrierung dargestellt.

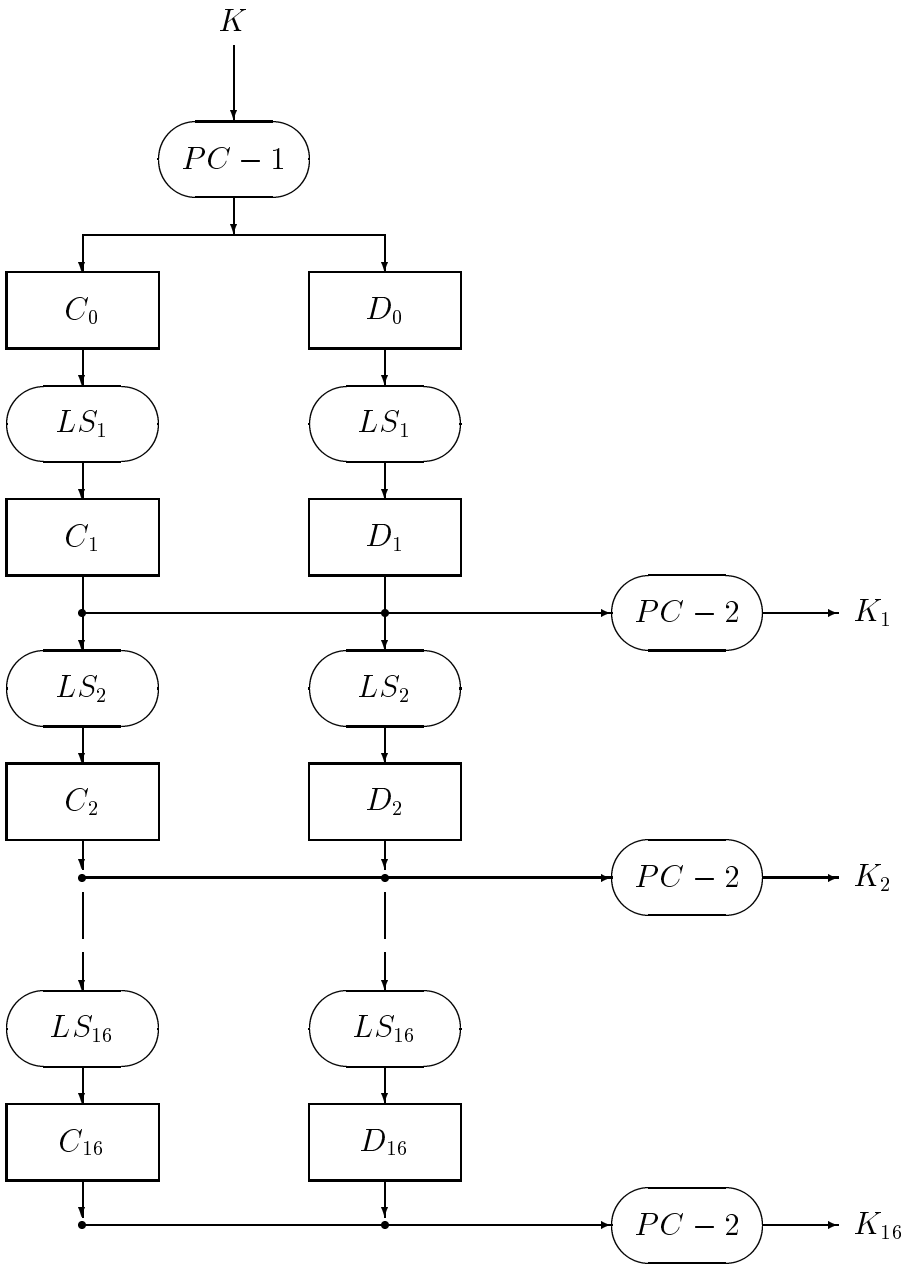
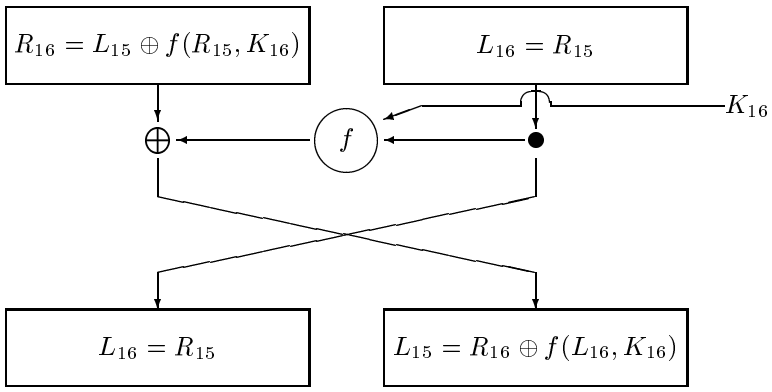



**Bild 2.6.3:** Schlüsselberechnung

Die *Dechiffrierung* erfolgt nach demselben Schema (siehe Bild 2.6.2). Dabei werden die Schlüssel allerdings in der umgekehrten Reihenfolge benutzt: $K_{16}$ in der ersten, $K_{15}$ in der zweiten, ..., $K_1$ in der sechzehnten Iteration. Wir wollen uns klarmachen, daß hierdurch tatsächlich die Dechiffrierung bestimmt ist. Bei der Berechnung von $D_K\, E_K$ kommt in der Mitte $IP \circ IP^{-1} (= id)$ vor. Nach der Ausführung von 16 Iterationen von $E_K$ kann also die erste Iteration von $D_K$ angewendet werden, was durch das folgende Bild dargestellt wird.




Allgemein gilt nach dem Bild 2.6.2 von Seite 37 für $i = 1, 2, \ldots, 16$

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

und

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

Nach der sechzehnten Iteration, die mit dem Schlüssel $K_1$ erfolgt, werden $L_0$ und $L_1$ nicht mehr vertauscht. Daraus folgt

$$T_0 = L_0\, L_1 = L_0\, R_0.$$

Die Anwendung von $IP^{-1}$ auf $T_0$ liefert dann $T$.

Der DES ist als injektive Abbildung von 64-Bit-Blöcken auf 64-Bit-Blöcke eine Permutation. Folglich ergibt sich bei wiederholter Anwendung irgendwann wieder der ursprüngliche Klartext. Die kleinste Anzahl von Anwendungen, bei der wir wieder den Klartext erhalten, nennen wir die Zyklenlänge. Es gibt Schlüssel mit kleiner Zyklenlänge. So existieren zum Beispiel 4 Schlüssel mit Zyklenlänge 2. Die Wahl eines solchen Schlüssels ist jedoch sehr unwahrscheinlich.

Der DES ist sowohl in Soft- als auch in Hardware implementiert. Bei der Hardware-Implementierung liegt die Chiffrierungsrate bei einigen Millionen Bits pro Sekunde (bps).

Mögliche *Schwächen* des DES bestehen darin, daß

(1) die Schlüsselgröße von 56 Bits eventuell nicht ausreichend ist, um den DES berechnungs-sicher zu machen und
(2) die $S$-Boxen vielleicht versteckte Falltüren (trapdoors) enthalten, mit deren Hilfe die Chiffre auch ohne Kenntnis des Schlüssels zu brechen ist.

Durch systematisches Probieren aller möglichen $2^{56}$ Schlüssel und Überprüfung, ob sich dadurch ein vernünftiger Klartext ergibt, könnte die Chiffre gebrochen werden. In der Tat ist die Schlüsselgröße von 56 Bits eine Schwäche. *Diffie* und *Hellman* [19] haben gezeigt, daß ein Spezialrechner mit einer Million LSI-Chips alle $2^{56} \cong 7 \cdot 10^{16}$ Schlüssel an einem Tag probieren kann. Ein Chip probiert dabei einen Schlüssel pro Mikrosekunde, was $8,64 \cdot 10^{10}$ Schlüsseln pro Tag entspricht. Die Kosten einer solchen Maschine betragen nach ihrer Abschätzung 20 Millionen US-Dollar. Amortisiert über 5 Jahre sind dies 10000 US-Dollar pro Tag, also auch 10000 US-Dollar für eine Lösung. Dieser Wert reduziert sich auf etwa 5000 US-Dollar, da im Mittel nur die Hälfte aller Schlüssel durchsucht zu werden braucht. Die Abschätzung der durchschnittlichen Suchzeit wurde später [17] auf zwei Tage erhöht. Dabei wurde der Einsatz einer Maschine, die dem technologischen Stand von 1980 entsprach, angenommen. Zum damaligen Zeitpunkt hätte sie etwa 50 Millionen US-Dollar gekostet. Mit der heutigen VLSI-Technologie, den schnelleren und billigeren Rechnern und größeren Speicherkapazitäten sind die Kosten deutlich geringer anzusetzen. 1993 schätzte man den Preis einer Maschine, die innerhalb von $3\frac{1}{2}$ Stunden mit diesem Brute-Force-Angriff aus dem Chiffretext den Klartext liefert, auf 1 Million Dollar. 1998 wurde für die US-Bürgerrechtsorganisation Electronic Frontier Foundation (EFF) für 250000 Dollar ein Spezialrechner mit fast 2000 Prozessoren gebaut, der in 56 Stunden den DES-Code knacken kann. Bei tatsächlich wichtigen Texten würde sich der Aufwand der Benutzung dieser Maschine lohnen, auch im Zusammenhang mit dem elektronischen Zahlungsverkehr, sicherlich jedoch nicht für harmlose E-Mails. Der DES kann also eigentlich nicht mehr als sicher angesehen werden. Darüber hinaus sind inzwischen weitere Ansätze zur Kryptoanalyse des DES vorgeschlagen worden (siehe z.B. [6], [7]), die eine wesentliche Verbesserung gegenüber der oben beschriebenen ausführlichen Suche darstellen. Diesen Suchstrategien kann allerdings zum Teil mit dem Chiffre-Block-Chaining (siehe Abschnitt 9.2) entgegengewirkt werden.

Zur größeren Sicherheit wurde eine Verdoppelung der Schlüsselgröße auf 112 Bits empfohlen. Da die Suchzeit exponentiell mit der Anzahl der Bits steigt, kann die Chiffre berechnungs-sicher werden. Eine andere Empfehlung [101] besteht darin, das Chiffreschema mit zwei unterschiedlichen Schlüsseln wie in dem Bild von Seite 42 zu benutzen. Dies Verfahren wird auch Tripel-DES genannt. Ein Brute-Force-Angriff hat hier keine Chance mehr, denn der Aufwand ist $2^{56}$-mal so groß wie beim normalen DES.

Beliebt ist heute auch ein anderes Chiffrierverfahren, und zwar der IDEA (International Data Encryption Algorithm) aus dem Jahre 1992 (siehe z.B. [103], [86]), der auch in der Software PGP (Pretty Good Privacy) verwendet wird. Der IDEA verschlüsselt Klartextblöcke der Länge 64 Bit mit einem Schlüssel von 128 Bit. Auch hier findet eine Vermischung unterschiedlicher Operationen statt. Chiffrierung und Dechiffrierung erfolgen wie bei dem DES mit demselben Schlüssel. Es handelt sich also sowohl bei dem IDEA als auch bei dem DES um symmetrische Chiffrierverfahren.

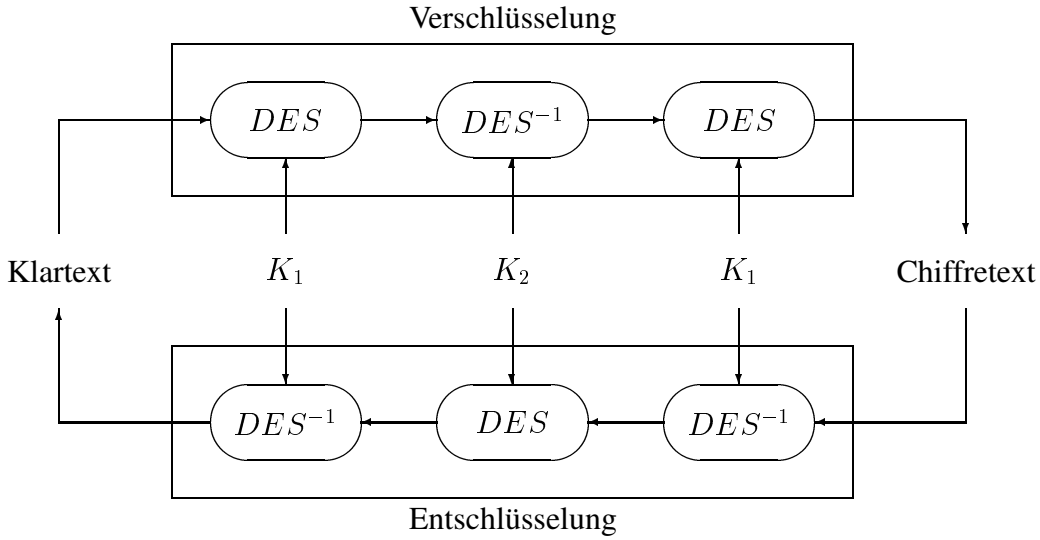



**Bild 2.6.4:** Mehrfachchiffrierung mit DES

# 3 Zahlentheoretische Grundlagen

Die Konzepte der Zahlentheorie, die in den weiteren Abschnitten benötigt werden, sollen hier zusammengefaßt werden. Mit $\mathbb{N}$ bezeichnen wir die Menge der natürlichen Zahlen und mit $\mathbb{Z}$ die Menge der ganzen Zahlen. Weiter ist $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

**Definition 3.1** Es seien $a, b \in \mathbb{Z}$ und $n \in \mathbb{N}$. $a$ heißt *kongruent* $b$ modulo $n$ (in Zeichen: $a \equiv_n b$), wenn es ein $k \in \mathbb{Z}$ gibt mit $a - b = k \cdot n$. □

Die Bedingung der Definition ist auch gleichwertig damit, daß $n$ die Zahl $a - b$ teilt (in Zeichen: $n|(a - b)$). Dabei heißt $b$ *Rest* von $a$ modulo $n$ und $a$ *Rest* von $b$ modulo $n$.

**Definition 3.2** Es seien $n \in \mathbb{N}$ und $r_1, r_2, \ldots, r_n \in \mathbb{Z}$. Die Menge $\{r_1, r_2, \ldots, r_n\}$ heißt *vollständige Menge von Resten* modulo $n$, wenn es für alle $a \in \mathbb{Z}$ genau ein $r_i$ mit $a \equiv_n r_i$ gibt. □

Für jedes $n$ bildet $\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}$ eine vollständige Menge von Resten modulo $n$. $\mathbb{Z}_n$ kann auch als Menge der Kongruenzklassen von $\equiv_n$ aufgefaßt werden. Sie werden als *Restklassen* modulo $n$ bezeichnet.

**Definition 3.3** Mit $a \bmod n$ bezeichnen wir den *Rest von* $a$ modulo $n$ im Intervall $[0, n - 1]$. □

Aus $a \bmod n = r$ folgt offenbar $a \equiv_n r$. Die Umkehrung gilt jedoch nicht. Weiter erhalten wir

$$a \equiv_n b \Longleftrightarrow a \bmod n = b \bmod n,$$

d.h., kongruente Zahlen haben denselben Rest im Intervall $[0, n - 1]$. Auf $\mathbb{Z}_n$ können wir eindeutig die folgenden Verknüpfungen definieren:

$$a \oplus b = (a + b) \bmod n,\ a \odot b = (a \cdot b) \bmod n \text{ für alle } a, b \in \mathbb{Z}_n.$$

Damit ist $(\mathbb{Z}_n, \oplus, \odot)$ eine algebraische Struktur.

**Satz 3.1** Es sei $n \in \mathbb{N}$ und $(\mathbb{Z}, +, \cdot)$ der Ring der ganzen Zahlen. Dann wird durch $f_n(a) = a \bmod n$ ein Homomorphismus $f_n : (\mathbb{Z}, +, \cdot) \to (\mathbb{Z}_n, \oplus, \odot)$ definiert.

*Beweis.* Gezeigt werden muß $f_n(a + b) = f_n(a) \oplus f_n(b)$ und $f_n(a \cdot b) = f_n(a) \odot f_n(b)$. Wir beweisen die erste Gleichung, also

$$(a + b) \bmod n = (a \bmod n) \oplus (b \bmod n).$$

Es gelte $a' = a \bmod n$ und $b' = b \bmod n$. Nach der Bemerkung im Anschluß an Definition 3.3 und nach Definition 3.1 folgt

$$n|(a - a') \text{ und } n|(b - b').$$

Wir erhalten

$$n|((a-a')+(b-b')) = ((a+b)-(a'+b'))$$

und damit

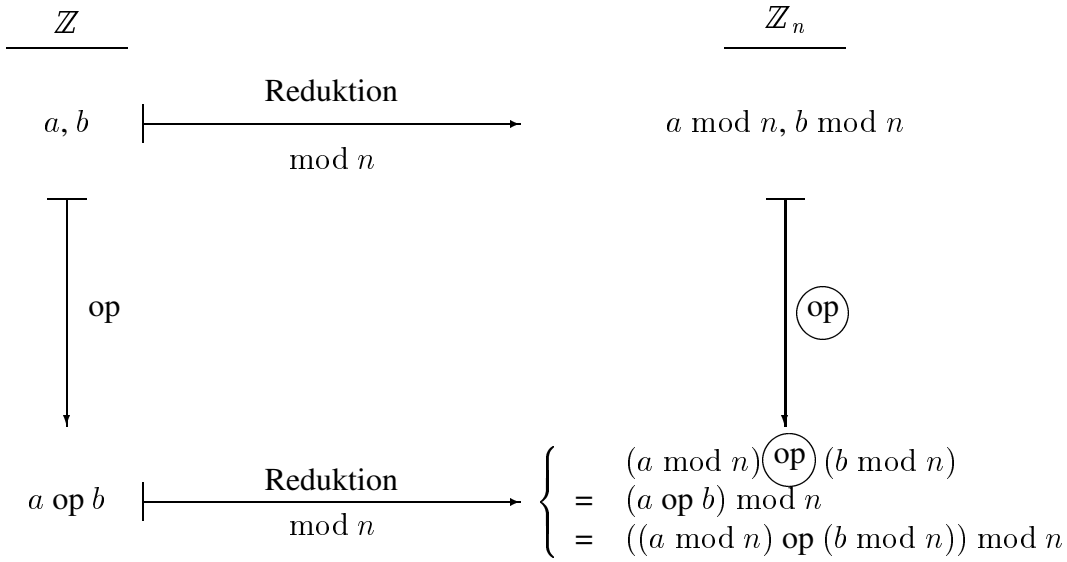
$$(a+b) \bmod n = (a'+b') \bmod n.$$

Wegen $a', b' \in \mathbb{Z}_n$ ergibt sich nach der Definition von $\oplus$ weiter

$$(a'+b') \bmod n = a' \oplus b' = (a \bmod n) \oplus (b \bmod n),$$

womit insgesamt die gewünschte Gleichung bewiesen ist. Ähnlich erhalten wir die zweite Gleichung. □

Als homomorphes Bild eines Ringes ist auch $(\mathbb{Z}_n, \oplus, \odot)$ ein Ring (siehe z.B. §21, Satz 1 in [41]). Aus Satz 3.1 ergibt sich das folgende Prinzip der modularen Arithmetik, wobei op : $\mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z}$ den jeweiligen Operator darstellt:




Modulare Arithmetik kann auch auf die Exponentiation angewendet werden. Wir erhalten dann

$$e^t \bmod n = \left( \prod_{i=1}^{t} (e \bmod n) \right) \bmod n.$$

Man beachte, daß für $t > n$ i. allg.

$$e^{t \bmod n} \bmod n \neq e^t \bmod n$$

ist, so z. B. $(2^{5 \bmod 3}) \bmod 3 = 1 \neq 2 = 2^5 \bmod 3$.

**Beispiel 3.1** Die Berechnung des Ausdrucks $3^5 \bmod 7$ kann durch wiederholtes Quadrieren und Multiplizieren auf zwei unterschiedliche Arten erfolgen:

(a) 1. Quadriere 3: $3 \cdot 3 = 9$

2. Quadriere das Ergebnis: $9 \cdot 9 = 81$

3. Multipliziere mit 3: $81 \cdot 3 = 243$
4. Reduziere modulo 7: $243 \bmod 7 = 5$

(b) 1. Quadriere 3: $(3 \cdot 3) \bmod 7 = 2$
2. Quadriere das Ergebnis: $(2 \cdot 2) \bmod 7 = 4$
3. Multipliziere mit 3 : $(4 \cdot 3) \bmod 7 = 5$ □

Falls $n$ aus $k$ Bits besteht, $2^{k-1} \leq n < 2^k$, bewirkt die modulare Arithmetik, daß das Ergebnis nach der Addition, Subtraktion oder Multiplikation zweier Werte höchstens $2k$ Bits umfaßt. Die Rechnungen werden damit einfacher. Wir geben einen entsprechenden Algorithmus für die Exponentiation an.

**Algorithmus 3.1**
fastexp($a, z, n$); {schnelle Exponentiation}
**begin** {liefert $x = a^z \bmod n$}
$a_1 := a; z_1 := z;$
$x := 1$
**while** $z_1 \neq 0$ **do** {Schleifeninvariante: $(x \cdot a_1^{z_1}) \bmod n = a^z \bmod n$}
  **while** $(z_1 \bmod 2) = 0$ **do**
    $z_1 := z_1/2;$
    $a_1 := (a_1 * a_1) \bmod n$
    **end**;
  $z_1 := z_1 - 1;$
  $x := (x * a_1) \bmod n$
  **end**;
fastexp:= $x$
**end** □

**Satz 3.2** Algorithmus 3.1 terminiert und liefert $a^z \bmod n$ als Ergebnis.

*Beweis.* Die äußere Schleife wird verlassen, wenn $z_1 = 0$ wird. Diese Bedingung tritt, da $z_1$ fortlaufend verkleinert wird, immer ein. Damit ist die Terminierung gesichert. Es muß noch die Richtigkeit der Schleifeninvariante nachgewiesen werden. Dies geschieht durch Induktion. Beim erstmaligen Eintritt in die Schleife ist die Schleifeninvariante offenbar erfüllt. Vor einem Durchlauf der äußeren Schleife habe $x$ den Wert $x'$, $a_1$ den Wert $a_1'$ und $z_1$ den Wert $z_1'$. Die innere Schleife werde nun $m$-mal durchlaufen. Dann ist $z_1 = \frac{z_1'}{2^m}$ und $a_1 = (a_1')^{2^m} \bmod n$. Anschließend wird

$$z_1 := \frac{z_1'}{2^m} - 1 \text{ und } x := x' \cdot (a_1')^{2^m} \bmod n$$

gebildet. Damit gilt nach diesem einmaligen Durchlauf der äußeren Schleife

$$\begin{aligned}
(x \cdot a_1^{z_1}) \bmod n &= \left( x' (a_1')^{2^m} (a_1')^{2^m \cdot \left(\frac{z_1'}{2^m} - 1\right)} \right) \bmod n \\
&= \left( x' \, (a_1')^{2^m + z_1' - 2^m} \right) \bmod n \\
&= \left( x' \, (a_1')^{z_1'} \right) \bmod n \\
&= a^z \bmod n \text{ (laut Induktionsannahme).}
\end{aligned}$$

Der Abbruch der äußeren Schleife erfolgt mit $z_1 = 0$. Da $x < n$ gilt, folgt aus der Schleifeninvariante, daß dann

$$x = (x \cdot a_1^0) \bmod n = a^z \bmod n$$

als Ergebnis geliefert wird. □

Die binäre Darstellung von $z$ sei $z_{k-1}\, z_{k-2} \ldots z_0$. Der Algorithmus verarbeitet die Bits in der Reihenfolge $z_0, z_1, \ldots, z_{k-1}$. Dabei wird quadriert, wenn ein Bit 0 ist, und multipliziert und quadriert, wenn ein Bit 1 ist. In der Hardware-Implementierung können diese Bits direkt angesprochen werden, so daß die Rechnungen

$$z_1 \bmod 2,\ z_1/2 \text{ und } z_1 - 1$$

entfallen können.

**Satz 3.3** Es sei $T$ die Anzahl der Multiplikationen, $z$ der Exponent von Algorithmus 3.1 und $k = [\log_2 z]$. Es folgt

$$k + 1 \leq T \leq 2k + 1.$$

*Beweis.* Nach den Satz 3.3 vorangehenden Ausführungen veranlaßt jedes 0-Bit eine Multiplikation, jedes 1-Bit zwei Multiplikationen. Eine Ausnahme stellt das am weitesten links gelegene 1-Bit dar, welches nur eine Multiplikation bewirkt. Es folgt $k + 1 \leq T \leq 2k + 1$. □

Die erwartete Anzahl von Multiplikationen ist $1.5 \cdot k + 1$. Ein naiver Algorithmus für die Exponentiation benötigt dagegen $z - 1$ Multiplikationen, ist also exponentiell in der Länge von $z$. Im folgenden interessieren wir uns für das multiplikative Inverse in $\mathbb{Z}_n$.

**Satz 3.4** Es sei $a \in \mathbb{Z}$, $n \in \mathbb{N}$, und es gelte ggt$(a, n) = 1$. Dann folgt, daß für alle $i, j \in \mathbb{N}_0$, $0 \leq i < j < n$,

$$(a \cdot i) \bmod n \neq (a \cdot j) \bmod n$$

gilt.

*Beweis.* Wir nehmen das Gegenteil an. Dann folgt $n|(ai - aj) = a(i - j)$. Wegen ggt$(a, n) = 1$ erhalten wir $n|(i - j)$. Dies ist ein Widerspruch zu $0 \leq i < j < n$. □

**Folgerung** Es sei $a \in \mathbb{Z}$, $n \in \mathbb{N}$, und es gelte ggt$(a, n) = 1$. Die $(a \cdot i) \bmod n$, $i = 0, 1, \ldots, n - 1$, liefern jeweils verschiedene Reste modulo $n$ und bilden die Menge

$$\mathbb{Z}_n = \{(a \cdot i) \bmod n | i = 0, 1, \ldots, n - 1\}. \quad \square$$

Unter der Annahme von Satz 3.4 hat $a$ ein multiplikatives Inverses. Dies zeigt

**Satz 3.5** Es sei $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$, und es gelte ggt$(a, n) = 1$. Dann existiert genau ein $x \in \mathbb{N}$, $0 < x < n$, mit

$$(a \cdot x) \bmod n = 1.$$

*Beweis.* Nach der Folgerung gibt es genau ein $x = i$ mit $(a \cdot i) \bmod n = 1$. □

Die folgenden Überlegungen führen zu einem Algorithmus zur Berechnung des Inversen.

**Definition 3.4** $E_n = \{a \in \mathbb{Z}_n \mid \text{ggt}(a,n) = 1\}$ heißt die *reduzierte Menge der Reste* modulo $n$. □

**Beispiel 3.2** Die reduzierte Menge der Reste modulo 10 ist $E_{10} = \{1,3,7,9\}$, die reduzierte Menge der Reste modulo $n$, wobei $n$ eine Primzahl ist, ist $E_n = \{1,2,\ldots,n-1\}$. □

**Definition 3.5** $\varphi(n)$ ($\varphi$ *Eulersche Funktion*) bezeichnet die Anzahl der Elemente der reduzierten Menge der Reste modulo $n$. Sie ist gleich der Anzahl der Zahlen $x, 1 \leq x < n$, mit $\text{ggt}(x,n) = 1$. □

**Satz 3.6** Es sei $n \in \mathbb{N}$, $n \geq 2$. Dann ist die reduzierte Menge $E_n$ der Reste modulo $n$ eine multiplikative Gruppe.

*Beweis.* Es sei $E_n = \{r_1, r_2, \ldots, r_{\varphi(n)}\}$ und $a \in E_n$. Dann gilt $\text{ggt}(a,n) = 1$. Wir schließen, daß für alle $i \in \mathbb{N}$, $1 \leq i \leq \varphi(n)$, $\text{ggt}(ar_i, n) = 1$ gilt und ein $j$, $1 \leq j \leq \varphi(n)$, existiert mit $(ar_i) \bmod n = r_j$. Das Produkt zweier Zahlen aus $E_n$ gehört also wieder zu $E_n$. Diese $(ar_i) \bmod n$ sind (siehe Folgerung zu Satz 3.4) für verschiedene $i$ paarweise verschieden. Somit erhalten wir

$$E_n = \{(ar_i) \bmod n | i = 1,2,\ldots,\varphi(n)\} = \{r_1, r_2, \ldots, r_{\varphi(n)}\}.$$

Weil $1 \in E_n$ gilt, existiert ein $r_i$ mit $(ar_i) \bmod n = 1$. $r_i$ ist somit das Inverse von $a$. $E_n$ ist also eine multiplikative Gruppe. □

**Satz 3.7** Es sei $p$ eine Primzahl. Dann gilt:

(a) $\varphi(p) = p - 1$.

(b) Ist $k \in \mathbb{N}$, dann folgt $\varphi(p^k) = p^{k-1}(p-1)$.

(c) Ist $q$ eine Primzahl mit $p \neq q$, dann folgt $\varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$.

*Beweis.*

(a) trivial.

(b) Von den $p^k - 1$ Zahlen von 1 bis $p^k - 1$ sind genau die Zahlen $p, 2p, \ldots, p^k - p = p\left(p^{k-1} - 1\right)$ Vielfache von $p$. Die übrigen sind zu $p$ teilerfremd. Daraus folgt

$$\varphi(p^k) = p^k - 1 - \left(p^{k-1} - 1\right) = p^{k-1}(p-1).$$

(c) Von den $pq - 1$ Zahlen von 1 bis $pq - 1$ sind genau die Zahlen $p, 2p, \ldots, (q-1)p$ und $q, 2q, \ldots, (p-1)q$ Vielfache von $p$ bzw. $q$. Da der Primteiler $p$ nicht in $q$ und $1, \ldots, p-1$ vorkommt und entsprechend $q$ nicht in $p$ und $1, \ldots, q-1$, sind diese Zahlen paarweise verschieden. Somit ist

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

Aus der Primfaktorzerlegung $n = p_1^{l_1}\, p_2^{l_2} \ldots p_t^{l_t}$ einer Zahl $n$ ergibt sich, wie man z.B. in [16] nachlesen kann,

$$\varphi(n) = \prod_{i=1}^{t} p_i^{l_i - 1}(p_i - 1).$$

**Satz 3.8** *(Fermat)* Es sei $p$ eine Primzahl und $a \in \mathbb{Z}$ mit ggt$(a, p) = 1$. Dann folgt

$$a^{p-1} \bmod p = 1. \quad \square$$

Der Beweis folgt aus

**Satz 3.9** Es seien $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit ggt$(a, n) = 1$. Dann folgt

$$a^{\varphi(n)} \bmod n = 1.$$

*Beweis.* Nach dem Beweis von Satz 3.6 betrachten wir die Menge

$$E_n = \{r_1, r_2, \ldots, r_{\varphi(n)}\} = \{(ar_i) \bmod n \mid i = 1, 2, \ldots, \varphi(n)\}.$$

Dann ist

$$a^{\varphi(n)} \bmod n \bullet \prod_{i=1}^{\varphi(n)} r_i = \prod_{i=1}^{\varphi(n)} ((ar_i) \bmod n) = \prod_{i=1}^{\varphi(n)} r_i.$$

Dabei sind $\prod$ und $\bullet$ Produkte in $E_n$. Kürzen in der multiplikativen Gruppe $E_n$ liefert die Gleichung $a^{\varphi(n)} \bmod n = 1$. $\square$

**Folgerung** Es seien $n \in \mathbb{N}, n \geq 2, a \in \mathbb{Z}$ mit ggt$(a, n) = 1$ und $(ax) \bmod n = 1$. Dann gilt

$$x = a^{\varphi(n)-1} \bmod n.$$

Falls $n$ prim ist, folgt $x = a^{n-2} \bmod n$. $\square$

Offenbar ist $x$ das Inverse von $a \bmod n$. Speziell betrachten wir $e, n \in \mathbb{N}$ mit ggt$(e, \varphi(n)) = 1$. Dann wird nach der Folgerung das Inverse von $e$ modulo $\varphi(n)$ durch $d = e^{\varphi(\varphi(n))-1} \bmod \varphi(n)$ bestimmt. Dies wird in dem folgenden Satz benutzt.

**Satz 3.10** Es seien $e, n \in \mathbb{N}$, und es gelte ggt$(e, \varphi(n)) = 1$. Dann ist die Abbildung $f : E_n \to E_n$ mit $f(a) = a^e \bmod n$ für alle $a \in E_n$ eine Bijektion.

*Beweis.* Es gelte $ed \bmod \varphi(n) = 1$. Damit definieren wir $g : E_n \to E_n$ mit $g(a) = a^d \bmod n$ für alle $a \in E_n$. Für alle $a \in E_n$ gilt

$$g(f(a)) = f(g(a)) = a^{ed} \bmod n = a^{1+k\varphi(n)} \bmod n = (a(a^{\varphi(n)})^k) \bmod n = a \bmod n$$

mit einem geeigneten $k \in \mathbb{N}_0$, da für $a \in E_n$ nach Satz 3.9 die Beziehung $a^{\varphi(n)} \bmod n = 1$ gilt. Folglich ist $g$ das Inverse von $f$. $\square$

Falls $n$ eine Primzahl ist, gilt $\mathbb{Z}_n = E_n \cup \{0\}$. Dann kann $f$ zu einer bijektiven Abbildung auf $\mathbb{Z}_n$ erweitert werden.

Wir wollen nun das Inverse $x$ von $a \bmod n$ praktisch berechnen. Die erste Möglichkeit kann angewendet werden, wenn $\text{ggt}(a, n) = 1$ gilt und $\varphi(n)$ bekannt ist. Dann erhalten wir

$$x = \text{fastexp}(a, \varphi(n) - 1, n)$$

mit Hilfe von Algorithmus 3.1. Die zweite Möglichkeit benutzt die iterative Version des Euklidischen Algorithmus zur Bestimmung des ggt:

**Algorithmus 3.2**
ggt($a$, $n$);
**begin**
$g_0 := n$;
$g_1 := a$;
$i := 1$;
**while** $g_i \neq 0$ **do**
  $g_{i+1} := g_{i-1} \bmod g_i$;
  $i := i + 1$
  **end**;
ggt:=$g_{i-1}$
**end** □

Dieser Algorithmus wird so erweitert, wie es in [47], S. 325, beschrieben ist. Dann kann auch $x$ berechnet werden.

**Algorithmus 3.3**
inv($a$, $n$);
**begin** {liefert $x$ mit $(ax) \bmod n = 1,\ 0 < a < n$}
$g_0 := n; g_1 := a$;
$u_0 := 1; v_0 := 0$;
$u_1 := 0; v_1 := 1$;
$i := 1$;
**while** $g_i \neq 0$ **do** {Schleifeninvariante: $g_i = u_i n + v_i a$}
  $y := g_{i-1}$ div $g_i$; {ganzahlige Division, Rest vergessen}
  $g_{i+1} := g_{i-1} - y * g_i$;
  $u_{i+1} := u_{i-1} - y * u_i$;
  $v_{i+1} := v_{i-1} - y * v_i$;
  $i := i + 1$
  **end**;
$x := v_{i-1}$;
**if** $x > 0$ **then** inv:=$x$ **else** inv:=$x + n$ **end**
**end** □

**Satz 3.11** Es sei $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$, und $\text{ggt}(a, n) = 1$. Dann berechnet Algorithmus 3.3 das Inverse $x$ von $a \bmod n$.

*Beweis.* Offensichtlich terminiert der Algorithmus, und es ist bekannt, daß für $g_i = 0$ die Gleichung $g_{i-1} = \text{ggt}(a, n)$ gilt. Falls $\text{ggt}(a, n) = 1$ ist, folgt unter Beachtung der Schleifeninvariante für $i - 1$

$$g_{i-1} = 1,\ v_{i-1}a - 1 = -u_{i-1}n.$$

Nach Definition 3.1 ist dann

$$v_{i-1}a \equiv_n 1.$$

$v_{i-1}$ ist ein Inverses von $a \bmod n$. Es läßt sich zeigen, daß $-n < v_{i-1} < n$ gilt, so daß die letzte Zeile des Algorithmus das Ergebnis liefert. Somit muß nur noch die Schleifeninvariante überprüft werden. Dies erfolgt durch Induktion. Für $i = 0$ und $i = 1$ ist sie erfüllt. Bei Eintritt in die **while**-Schleife gilt $i = 1$. Es ist zu zeigen, daß $g_{i+1} = u_{i+1}n + v_{i+1}a$ gilt. Speziell für $y = g_{i-1}$ div $g_i$ gilt aufgrund der Zuweisungen in der Schleife und der Gültigkeit der Schleifeninvariante für $i$ und $i - 1$

$$\begin{aligned} g_{i+1} = g_{i-1} - yg_i &= u_{i-1}n + v_{i-1}a - y(u_in + v_ia) \\ &= (u_{i-1} - yu_i)n + (v_{i-1} - yv_i)a \\ &= u_{i+1}n + v_{i+1}a. \quad \square \end{aligned}$$

Knuth hat gezeigt, daß die durchschnittliche Anzahl von Divisionen im Algorithmus 3.3 ungefähr gleich $0,843 \log_e(n) + 1,47$ ist. Man erkennt, daß statt $g_{i-1}, g_i$ und $g_{i+1}$ auch drei lokale Variablen $g_{last}, g$ und $g_{next}$ verwendet werden können. Für $v$ gilt entsprechendes. Die Variablen $u$ können entfallen.

Der Algorithmus 3.3 kann zur Lösung des folgenden Problems erweitert werden: Gegeben seien $a \in \mathbb{Z}$ und $n \in \mathbb{N}$ mit ggt$(a, n) = 1$. Gesucht ist ein $x$ mit

$$(ax) \bmod n = b \bmod n$$

für ein $b \in \mathbb{Z}$. Dazu muß eine Lösung $x_0$ von $(ax) \bmod n = 1$ gefunden werden. Aus $(ax_0) \bmod n = 1$ folgt nämlich $(abx_0) \bmod n = b \bmod n$, also ist $x = (bx_0) \bmod n$ eine Lösung in $[1, n-1]$. Sie ist auch die einzige Lösung. Es sei nämlich $x_1$ eine weitere Lösung. Dann gilt $(abx_0 - ax_1) \bmod n = 0 = a(bx_0 - x_1) \bmod n$. Wegen ggt$(a, n) = 1$ folgt $(bx_0 - x_1) \bmod n = 0$ und damit $x_1 = bx_0 \bmod n$.

**Beispiel 3.3** Bei der Berechnung von $(3x) \bmod 7 = 1$ mit Algorithmus 3.3 werden folgende Werte durchlaufen:

| $i$ | $g_i$ | $u_i$ | $v_i$ | $y$ |
|---|---|---|---|---|
| 0 | 7 | 1 | 0 | - |
| 1 | 3 | 0 | 1 | 2 |
| 2 | 1 | 1 | $-2$ | 3 |
| 3 | 0 | | | |

Die Lösung lautet $x = -2 + 7 = 5$. $\square$

Falls ggt$(a, n) \neq 1$ ist, dann hat die Gleichung $(ax) \bmod n = b$ entgegen dem Fall ggt$(a, n) = 1$ mehr als eine bzw. keine Lösung. Dieser Fall wird im folgenden Satz betrachtet.

**Satz 3.12** Es seien $a \in \mathbb{Z}$ und $n \in \mathbb{N}$, $n \geq 2$. Es gelte $g =$ggt$(a, n)$. Aus $g|b$ folgt, daß die Gleichung

$$(ax) \bmod n = b$$

genau $g$ Lösungen der Form

$$x = \left(\frac{b}{g} \cdot x_0 + t \cdot \frac{n}{g}\right) \bmod n, \;\; t = 0, 1, 2, \ldots, g-1,$$

hat. Dabei ist $x_0$ die Lösung von

$$\left(\frac{a}{g} \cdot x\right) \bmod \left(\frac{n}{g}\right) = 1.$$

Aus $g \not| b$ folgt, daß $(ax) \bmod n = b$ keine Lösung hat.

*Beweis.* Falls $(ax) \bmod n = b$ eine Lösung in $[1, n-1]$ besitzt, dann ergibt sich nach Definition 3.1 $n|(ax - b)$. Aus $g|n$ und $g|ax$ folgt $g|b$. Wir erkennen daraus, daß für $g \not| b$ keine Lösung existiert. Weiter gilt

$$\text{ggt}(\tfrac{a}{g}, \tfrac{n}{g}) = 1.$$

Dann besitzt $\left(\frac{a}{g}\right) \cdot x \bmod \left(\frac{n}{g}\right) = 1$ eine eindeutige Lösung $x_0$ im Intervall $[1, \frac{n}{g} - 1]$. Nach den Überlegungen vor Beispiel 3.3 ist damit $x_1 = \frac{b}{g} \cdot x_0 \bmod \left(\frac{n}{g}\right)$ eine eindeutige Lösung von

$$\left(\frac{a}{g} \cdot x\right) \bmod \left(\frac{n}{g}\right) = \frac{b}{g}$$

im Intervall $[1, \frac{n}{g} - 1]$. Es existiert also ein $k \in \mathbb{Z}$ mit

$$\frac{a}{g} \cdot x_1 - \frac{b}{g} = k \cdot \frac{n}{g}.$$

Durch Multiplikation mit $g$ erhalten wir

$$ax_1 - b = k \cdot n,$$

d. h., $x_1$ ist eine Lösung von $(ax) \bmod n = b$. Jedes $x \in \{0, 1, \ldots, n-1\}$ mit $x \equiv_{\frac{n}{g}} x_1$ ist ebenfalls eine Lösung von $(ax) \bmod n = b$. Alle anderen $x$ sind keine Lösungen, da sonst $x_1' = x \bmod \left(\frac{n}{g}\right) \neq x_1,\ x_1' \in [1, \frac{n}{g} - 1]$, ebenfalls eine Lösung der Gleichung

$$\left(\frac{a}{g} \cdot x\right) \bmod \left(\frac{n}{g}\right) = \frac{b}{g}$$

wäre. Alle Lösungen von $(ax) \bmod n = b$ lauten also

$$x = x_1 + t \cdot \frac{n}{g},\ t = 0, 1, \ldots, g-1. \quad \square$$

**Beispiel 3.4** Gegeben sei die Gleichung $(6x) \bmod 10 = 4$. Dann gilt $g = \text{ggt}(6, 10) = 2$. Wegen $2|4$ existieren zwei Lösungen. Wir lösen zunächst

$$\left(\frac{6}{2} \cdot x\right) \bmod \left(\frac{10}{2}\right) = 1 \Longleftrightarrow (3x) \bmod 5 = 1.$$

Es folgt $x_0 = 2$ und damit

$$x_1 = \left(\frac{4}{2} \cdot 2\right) \bmod \left(\frac{10}{2}\right) = 4 \bmod 5 = 4.$$

Damit erhalten wir für $t = 0$ die Lösung $x = 4$ und für $t = 1$ die Lösung

$$x = (4 + \frac{10}{2}) \bmod 10 = 9. \quad \square$$

Die Lösung von $(ax) \bmod n = b$ kann auch durch eine Primfaktorzerlegung von $n$ erfolgen. Es sei

$$n = d_1\, d_2 \ldots d_t \text{ mit } d_i = p_i^{l_i},\ 1 \leq i \leq t,$$

mit jeweils verschiedenen Primzahlen $p_i$. Damit gilt ggt$(d_i, d_j) = 1$ für $i \neq j$. Die Lösung ergibt sich aus dem allgemeineren

**Satz 3.13** Gegeben seien $d_i \in \mathbb{N},\ 1 \leq i \leq t$, mit ggt$(d_i, d_j) = 1$ für $i \neq j$. Es sei $n = d_1 \ldots d_t,\ f$ ein Polynom und $x \in \mathbb{Z}$. Dann gilt

$$f(x) \bmod n = 0 \Longleftrightarrow f(x) \bmod d_i = 0,\ 1 \leq i \leq t.$$

*Beweis.* Aufgrund der Voraussetzungen gilt

$$n | f(x) \Longleftrightarrow d_i | f(x),\ 1 \leq i \leq t. \quad \square$$

Damit erhalten wir eine weitere Möglichkeit zur Lösung von Gleichungen der Form $(ax) \bmod n = b$. Wir betrachten $(ax - b) \bmod n = 0$. Die Lösung ist eine gemeinsame Lösung der Gleichungen $(ax - b) \bmod d_i = 0, 1 \leq i \leq t$, die äquivalent zu den Gleichungen

$$(ax) \bmod d_i = b \bmod d_i,\ 1 \leq i \leq t,$$

sind.

Wir wollen nun eine gemeinsame Lösung zu den Gleichungen $f(x) \bmod d_i = 0$ konstruieren, und zwar aus unabhängigen Lösungen $x_1, x_2, \ldots, x_t$ von

$$f(x) \bmod d_i = 0,\ 1 \leq i \leq t.$$

Jedes $x$ mit $x \bmod d_i = x_i$ ist eine Lösung von $f(x) \bmod d_i = 0$. Nach Satz 3.13 folgt, daß $x$ genau dann eine Lösung von $f(x) \bmod n = 0$ ist, wenn $x \bmod d_i = x_i,\ 1 \leq i \leq t$, gilt. Die Berechnung einer gemeinsamen Lösung $x$ erfolgt nun nach dem Chinesischen Restesatz.

**Satz 3.14** (*Chhin Chiu Shao*, 1247) Es seien $d_i \in \mathbb{N}, x_i \in \mathbb{Z}, 1 \leq i \leq t$, mit ggt$(d_i, d_j) = 1$ für $i \neq j$. Es sei $n = d_1\, d_2\, \ldots\, d_t$. Dann besitzt das System der Gleichungen

$$x \bmod d_i = x_i \bmod d_i,\ i = 1, 2, \ldots, t,$$

genau eine Lösung $x \in [0, n - 1]$.

*Beweis.* Für alle $i$, $1 \leq i \leq t$, gilt ggt$(d_i, \frac{n}{d_i}) = 1$. Dann folgt (siehe Schleifeninvariante des Algorithmus 3.3), daß $u_i, y_i \in \mathbb{Z}$ existieren mit

$$u_i d_i + y_i \frac{n}{d_i} = 1.$$

Somit ist

$$\frac{n}{d_i} \cdot y_i \bmod d_i = 1.$$

Da $d_j | \frac{n}{d_i}$ gilt, folgt $\frac{n}{d_i} \cdot y_i \bmod d_j = 0$ für $i \neq j$. Wir setzen

$$x = \sum_{j=1}^{t} \frac{n}{d_j} \cdot y_j \cdot x_j \bmod n.$$

Dann ergibt sich $x \bmod d_i = \frac{n}{d_i} \cdot y_i \cdot x_i \bmod d_i = x_i \bmod d_i$, also ist $x \in [0, n-1]$ eine Lösung.

Aus der Annahme, daß $x, x' \in [0, n-1], x \neq x'$, zwei verschiedene Lösungen sind, folgt $(x - x') \bmod d_i = 0$. Wir schließen $d_i | (x - x'),\ i = 1, 2, \ldots, t$. Da die $d_i$ paarweise relativ prim sind, erhalten wir

$$n = d_1\, d_2 \ldots d_t | (x - x').$$

Dies ist ein Widerspruch zu $|x - x'| < n = d_1\, d_2 \ldots d_t$. Somit existiert genau eine Lösung. □

Aus dem Beweis ergibt sich der folgende

**Algorithmus 3.4**
Lösung des Kongruenzsystems nach dem Chinesischen Restesatz.
crt$(n, d_1, d_2, \ldots, d_t, x_1, x_2, \ldots, x_t)$;
{liefert $x \in [0, n-1]$ mit $x \bmod d_i = x_i \bmod d_i,\ 1 \leq i \leq t$}
**begin**
**for** $i := 1$ **to** $t$ **do**
  $y_i :=$inv$\left(\left(\frac{n}{d_i}\right) \bmod d_i, d_i\right)$ {Algorithmus 3.3}
**end**;
$x := 0$;
**for** $i := 1$ **to** $t$ **do**
  $x := \left(x + \frac{n}{d_i} * y_i * x_i\right) \bmod n$
**end**;
crt:=$x$
**end** □

**Beispiel 3.5** Wir betrachten die Gleichung

$$(3x) \bmod 10 = 1.$$

Wir werden sie mit Hilfe des Chinesischen Restesatzes lösen. Wegen $10 = 2 \cdot 5$ folgt $d_1 = 2$, $d_2 = 5$. Die Lösung ist nach Satz 3.13 eine gemeinsame Lösung von

$$(3x) \bmod 2 = 1 \text{ und } (3x) \bmod 5 = 1.$$

Die erste Gleichung hat die Lösung $x_1 = 1$, die zweite $x_2 = 2$. Die gemeinsame Lösung wird mit dem Chinesischen Restesatz konstruiert. Wir betrachten

$$x \bmod 2 = x_1 = 1$$
$$x \bmod 5 = x_2 = 2.$$

Nach Algorithmus 3.4 berechnen wir $y_1$ und $y_2$ aus

$$\frac{10}{2} y_1 \bmod 2 = 1 \text{ und } \frac{10}{5} y_2 \bmod 5 = 1.$$

Es folgt $y_1 = 1$ und $y_2 = 3$. Damit erhalten wir

$$x = (\frac{10}{2} \cdot y_1 \cdot x_1 + \frac{10}{5} \cdot y_2 \cdot x_2) \bmod 10 = (5 \cdot 1 \cdot 1 + 2 \cdot 3 \cdot 2) \bmod 10 = 7. \quad \square$$

**Beispiel 3.6** Die Gleichung

$$(6x) \bmod 10 = 4$$

aus Beispiel 3.4 soll auf andere Weise gelöst werden. Lösungen sind nach Satz 3.13 gemeinsame Lösungen von

$$6x \bmod 5 = 4$$
$$6x \bmod 2 = 0.$$

Die erste Gleichung hat die Lösung $x_1 = 4$. Die zweite besitzt wegen $\text{ggt}(6, 2) = 2$ nach Satz 3.12 genau zwei Lösungen, und zwar sind dies $y_1 = 0$ und $y_2 = 1$. Die zwei Lösungen der gegeben Gleichung ergeben sich aus den beiden Gleichungssystemen

$$x \bmod 5 = x_1 = 4$$
$$x \bmod 2 = y_1 = 0$$

und

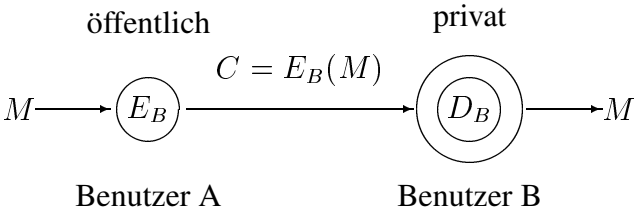
$$x \bmod 5 = x_1 = 4$$
$$x \bmod 2 = y_2 = 1.$$

Das erste System liefert die Lösung $x = 4$, das zweite $x = 9$. $\square$

# 4 Das RSA-Verfahren

## 4.1 Public-Key-Kryptosysteme

Public-Key-Kryptosysteme wurden 1976 von *Diffie* und *Hellman* [18] eingeführt. Jeder Benutzer eines solchen Systems hat einen öffentlichen und einen privaten Schlüssel. Die Kommunikation zweier Benutzer erfolgt unter der Kenntnis der gegenseitigen öffentlichen Schlüssel. Jeder Benutzer A besitzt eine *öffentliche* Chiffriertransformation $E_A$ und eine *private* Dechiffriertransformation $D_A$. $D_A$ wird durch den zugehörigen privaten Schlüssel, $E_A$ durch den zugehörigen öffentlichen Schlüssel beschrieben. Das System stellt $E_A$ allen Benutzern zur Verfügung (z. B. durch Registrierung in einem öffentlichen Directory), $D_A$ ist dagegen nur A bekannt. Die Chiffriertransformation $E_A$ muß aus $D_A$ leicht berechenbar sein, die Umkehrung sollte jedoch schwer sein, d. h. praktisch nicht berechenbar. Die *Geheimhaltung* (siehe Abschnitt 1.2) in einem Public-Key-Kryptosystem wird durch das folgende Bild veranschaulicht:

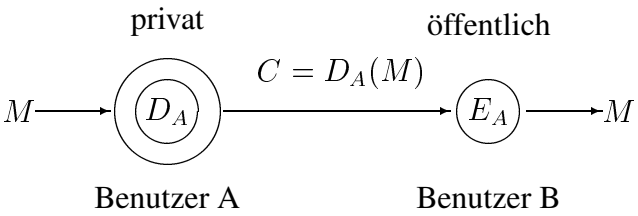



Alice (A) möchte an Bob (B) eine private Nachricht $M$ schicken. A kennt B's öffentlichen Schlüssel und damit $E_B$. A bildet den Chiffretext $C = E_B(M)$ und sendet ihn B. Nur B kennt die Dechiffrierfunktion $D_B$ und kann damit den Text entschlüsseln:

$$D_B(C) = D_B(E_B(M)) = M.$$

Allerdings kann B nicht sicher sein, daß der Text von A stammt.

Die *Authentizität* (ohne Geheimhaltung) ergibt sich aus dem folgenden Bild:



Dabei sind gegenüber den Ausführungen über Authentizität in Abschnitt 1.2 die Abbildungen $E_A$ und $D_A$ vertauscht. A sendet $C = D_A(M)$ zu B. Durch die öffentliche Transformation $E_A$ kann B den Klartext $M$ zurückgewinnen, nämlich

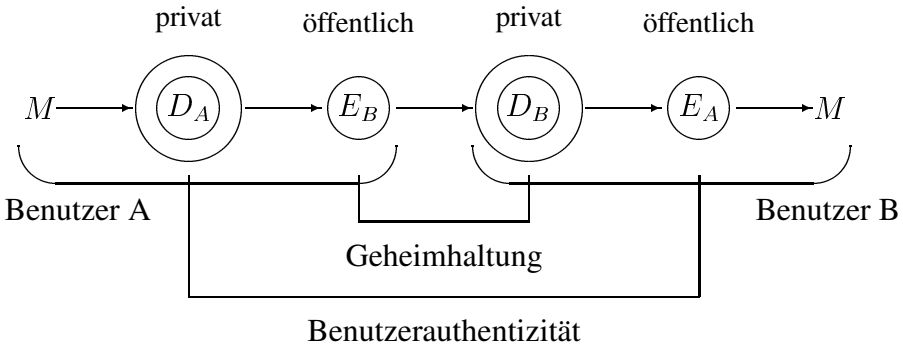
$$E_A(C) = E_A(D_A(M)) = M.$$

Jeder Benutzer, der $E_A$ kennt, kann diese Nachricht $M$ lesen.

Sollen nun Geheimhaltung und Authentizität gleichzeitig gewährleistet werden, so werden folgende Forderungen an die Public-Key-Transformationen gestellt:

| Geheimhaltung | Authentizität | Geheimhaltung & Authentizität |
|---|---|---|
| $E_A : \mathcal{M} \longrightarrow \mathcal{C}$ | $D_A : \mathcal{M} \longrightarrow \mathcal{C}$ | $E_A : \mathcal{M} \longrightarrow \mathcal{M}$ |
| $D_A : \mathcal{C} \longrightarrow \mathcal{M}$ | $E_A : \mathcal{C} \longrightarrow \mathcal{M}$ | $D_A : \mathcal{M} \longrightarrow \mathcal{M}$ |
| $D_A(E_A(M)) = M$ | $E_A(D_A(M)) = M$ | $D_A(E_A(M)) = M = E_A(D_A(M))$ |

Die Voraussetzung für gleichzeitige Geheimhaltung und Authentizität ist, daß der Chiffreraum $\mathcal{C}$ und der Nachrichtenraum $\mathcal{M}$ übereinstimmen. Die Vorgehensweise erläutert das folgende Bild:




Einige Systeme, die diesen Forderungen genügen, werden in den nächsten Abschnitten diskutiert.

Wir kommen nun auf Anwendungen von Public-Key-Kryptosystemen zu sprechen. Wir betrachten zunächst *digitale Signaturen*. Eine elektronische Unterschrift muß sowohl von der Nachricht als auch vom Unterzeichner abhängen. Anderenfalls könnte der Empfänger ja die Nachricht abändern oder die Signatur an eine beliebige Nachricht anhängen, bevor er das Paar aus Nachricht und Signatur im Streitfall einem Richter vorweist.

Wir geben jetzt ein Protokoll an, wie Alice (A) an Bob (B) eine unterzeichnete Nachricht schicken kann.

**Protokoll 4.1.1**

(1) Alice unterzeichnet $M$ durch die Berechnung von $C = D_A(M)$.

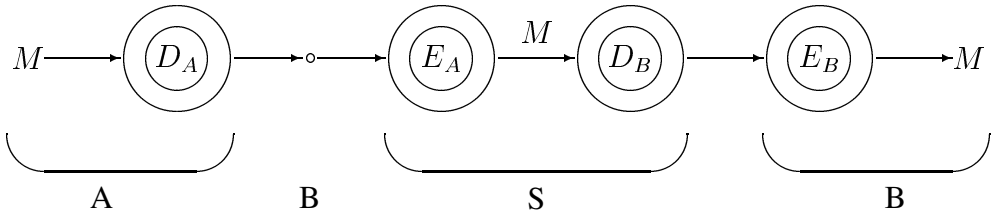
(2) Bob prüft die Unterschrift von Alice durch die Berechnung von $E_A(C) = M$. Ist $M$ ein vernünftiger Klartext, so weiß Bob, daß Alice ihm $C$ geschickt hat, denn niemand anders als Alice kennt $D_A$ und kann $C = D_A(M)$ erzeugen.

(3) In einem Streitfall zwischen Alice und Bob kann ein neutraler Richter überprüfen, ob $E_A(C) = M$ gilt. □

Damit ist die Authentizität von Nachricht und Signatur gewährleistet. Durch zusätzliche Verwendung der Schlüssel von Bob kann auch wie oben die Geheimhaltung erreicht werden.

Konventionelle Systeme wie der DES gewährleisten die Datenauthentizität, nicht jedoch die Authentizität des Senders. Empfänger und Sender teilen jeweils denselben Schlüssel, und der Empfänger könnte die Signatur des Senders fälschen, da beide $D_A$ und $E_A$ kennen.

Um Abhilfe zu schaffen, muß eine vertrauenswürdige dritte Partei S zur Verfügung stehen. Jeder Benutzer A läßt seine privaten Transformationen $E_A$ und $D_A$ bei S registrieren. Die Vorgehensweise wird durch das folgende Bild erläutert:




A sendet $C = D_A(M)$ an B, der $C$ an S zur Entschlüsselung vermöge $E_A$ weitergibt. Dadurch überprüft S die Authentizität des Senders. B erhält von S die verschlüsselte Nachricht $D_B(M)$, aus der er den Klartext gewinnen kann.

Wir wollen noch Anwendungen von digitalen Signaturen angeben.

**Beispiel 4.1.1** Alice Bank erhält auf elektronischem Weg den Auftrag, 100000 DM abzubuchen. Nach Anwendung des oben beschriebenen Authentifizierungsverfahrens kann die Bank sicher sein, daß Alice die Auftraggeberin ist. In einem Streitfall kann die Bank den Beweis für den Auftrag liefern. Hierbei müssen zusätzlich auch die Geheimhaltungsanforderungen erfüllt sein. □

**Beispiel 4.1.2** Zwei Länder schließen ein Nukleareteststop-Abkommen. Es wird ein System zur Kontrolle dieses Abkommens angegeben. Jede Nation installiert im anderen Land eine seismische Überwachungsstation. Sie übermittelt die Daten an eine Kontrollinstanz im eigenen Land. Es werden drei Forderungen gestellt:

(1) Die Kontrollinstanz muß sicher sein, daß die empfangene Information aus der eigenen Station stammt und nicht verändert wurde. Dies ist die Forderung der Sender- und Datenauthentizität.
(2) Das Land, in dem die Station steht, muß sicher sein, daß diese nicht für andere Zwecke benutzt wird. Es muß die übertragenen Nachrichten lesen können.
(3) Keine der Nationen darf Nachrichten produzieren können, die aus irgendeiner dieser Stationen stammen könnten. Die Vereinten Nationen müssen einen Streit schlichten können.

Die Forderungen sind erfüllt, falls ein Public-Key-Authentifizierungssystem benutzt wird, bei dem der private Schlüssel $D$ keiner Nation bekannt ist und beide den öffentlichen Schlüssel $E$ kennen. □

**Beispiel 4.1.3** Ein Rechensystem könnte es ablehnen, ein Programm auszuführen, das nicht durch eine bestimmte Autorität signiert ist. □

Detaillierte Untersuchungen über digitale Signaturen wurden z.B. in [2], [33], [35] und [58] durchgeführt. Da der Rechenaufwand bei der direkten Durchführung von Protokoll 4.1.1 in den meisten Fällen zu aufwendig ist, wird in der Praxis ein abgewandeltes Authentifizierungsverfahren verwendet, das wir in Kapitel 6 in Protokoll 6.1.1 darstellen werden. Außerdem werden wir in Abschnitt 5.2 Protokolle betrachten, die es ermöglichen, elektronisch Verträge auszutauschen. Da dieser Austausch „gleichzeitig“ erfolgen muß, reicht Protokoll 4.1.1 (oder Protokoll 6.1.1) für diese Aufgabe nicht aus.

## 4.2 Exponentiationschiffren

Eine *Exponentiationschiffre* verschlüsselt einen Nachrichtenblock $M \in [0, n-1]$ durch

$$E(M) = C = M^e \bmod n,$$

wobei $e$, $n \in \mathbb{N}$ die Schlüssel der Chiffre sind. Die Dechiffrierung erfolgt mit einem weiteren Exponenten $d \in \mathbb{N}$ durch

$$D(C) = M = C^d \bmod n.$$

$e$, $d$ und $n$ müssen so gewählt werden, daß die Dechiffrierung das Inverse der Chiffrierung ist. Chiffrierung und Dechiffrierung können durch den schnellen Exponentiationsalgorithmus (Algorithmus 3.1) implementiert werden, also

$$C = \text{fastexp}(M, e, n) \text{ und } M = \text{fastexp}(C, d, n).$$

Eine weiter unten beschriebene Wahl von $e$, $d$ und $n$ beruht auf

**Satz 4.2.1** Gegeben seien $e$, $d$, $n \in \mathbb{N}$ mit $(ed) \bmod \varphi(n) = 1$. Es sei $M \in [0, n-1]$ eine Nachricht mit $\text{ggt}(M, n) = 1$. Dann folgt

$$(M^e \bmod n)^d \bmod n = M.$$

*Beweis.* Aus

$$(ed) \bmod \varphi(n) = 1$$

folgt, daß ein $t \in \mathbb{N}$ existiert mit

$$ed = t\,\varphi(n) + 1.$$

Damit schließen wir

$$\begin{aligned}
(M^e \bmod n)^d \bmod n &= M^{ed} \bmod n \\
&= M^{t\varphi(n)+1} \bmod n \\
&= M\left(M^{t\varphi(n)}\right) \bmod n \\
&= M\left(M^{\varphi(n)} \bmod n\right)^t \bmod n \\
&= M \cdot 1^t \bmod n \text{ (Satz 3.9)} \\
&= M. \quad \square
\end{aligned}$$

Aufgrund der Symmetrie von $e$ und $d$ sind die Chiffrierung und die Dechiffrierung kommutativ. Es gilt also auch

$$\left(M^d \bmod n\right)^e \bmod n = M^{de} \bmod n = M.$$

Diese Symmetrie ermöglicht es, die unten beschriebenen Verfahren sowohl für Geheimhaltung als auch für Authentizität in einem Public-Key-Kryptosystem zu benutzen.

Es soll nun dargelegt werden, wie man ein Paar $(e, d)$ mit $(ed) \bmod \varphi(n) = 1$ erzeugt. Falls $\varphi(n)$ bekannt ist, wählen wir ein $d$ mit ggt$(d, \varphi(n)) = 1$. Algorithmus 3.3 liefert dann $e =$inv$(d, \varphi(n))$. Auch hier können $e$ und $d$ vertauscht werden, statt $d$ könnte also $e$ gewählt werden.

Falls $\varphi(n)$ bekannt und $e$ gegeben ist, dann ist also die Berechnung von $d$ einfach. Wird dagegen nur $n$ und $e$ bekanntgemacht, ohne $\varphi(n)$ und $d$ zu verraten, dann kann die Dechiffriertransformation geheimgehalten werden, während die Chiffriertransformation öffentlich ist. Der Grund ist darin zu sehen, daß die Primfaktorzerlegung einer sehr großen Zahl $n$ ein „schwieriges“ Problem ist und $\varphi(n)$ nur aus einer solchen Faktorisierung leicht zu berechnen ist. Im RSA-Schema kann $\varphi(n)$ verborgen werden, was im Pohlig-Hellman-Schema nicht der Fall ist.

Im *Pohlig-Hellman-Schema* [69] wird für $n$ eine große Primzahl $p$ gewählt. Die Chiffrierung und die Dechiffrierung wird durch

$$\begin{aligned} E(M) &= C = M^e \bmod p \text{ und} \\ D(C) &= M = C^d \bmod p \end{aligned}$$

gegeben. Für alle Nachrichten $M \in [0, p-1]$ gilt ggt$(M, p) = 1$. Damit ist die entsprechende Voraussetzung von Satz 4.2.1 erfüllt. Alle Rechnungen erfolgen im Körper $\mathbb{Z}_p = GF(p)$. Da $p$ eine Primzahl ist, folgt nach Satz 3.7 (a) $\varphi(p) = p - 1$. Das Pohlig-Hellman-Schema ist damit nach der obigen Überlegung nur für die konventionelle Chiffrierung nutzbar. $e$ und $d$ müssen also beide geheim gehalten werden.

**Beispiel 4.2.1** Es sei $p = 11$ und $d = 7$. Dann gilt $\varphi(p) = 10$ und ggt$(7, 10) = 1$. Dann folgt

$$e = \text{inv}(d, \varphi(p)) = \text{inv}(7, 10) = 3.$$

Für $M = 5$ liefert die Chiffrierung

$$C = 5^3 \bmod 11 = 4.$$

Die Dechiffrierung

$$4^7 \bmod 11 = 5 = M$$

ergibt wieder den Klartext. □

Die Sicherheit des Schemas basiert auf der Komplexität der Berechnung diskreter Logarithmen in $GF(p)$. Dies ergibt sich daraus, daß bei einem Angriff mit bekanntem Klartext das Paar $(M, C)$ bekannt ist und aus $C = M^e$ in $GF(p)$ sich $e = \log_M C$ in $GF(p)$ ergibt.

Der Kryptoanalytiker kann dabei $p$ ggf. aus der Beobachtung der Größen der Klartext- und Chiffretextblöcke schließen.

*Pohlig* und *Hellman* [69] zeigen, daß, falls $p-1$ nur kleine Primfaktoren enthält, der diskrete Logarithmus mit einem Zeitbedarf der Ordnung $O(\log^2 p)$ zu berechnen ist. Auch für große $p$ ist dieser Zeitbedarf ziemlich unbefriedigend. Es wird daher empfohlen, $p$ als eine Primzahl mit $p = 2p' + 1$ zu wählen, wobei $p'$ eine große Primzahl ist. Der schnellste bekannte Algorithmus zur Berechnung des diskreten Logarithmus in $GF(p)$ (Adleman, [1]) benötigt etwa

$$T = e^{\sqrt{\ln(p)\cdot\ln(\ln(p))}}$$

Berechnungsschritte.

**Beispiel 4.2.2** Es sei $p$ 200 Bits lang. Dann benötigt der Algorithmus von *Adleman* etwa $T = 2,7 \cdot 10^{11}$ Schritte. Bei $10^{11}$ Schritten pro Tag, das entspricht ungefähr 1 Schritt pro $\mu$sec, sind dies 2,7 Tage. Wird $p$ jedoch auf 664 Bits (200 Dezimalstellen) erweitert, so werden $T = 1,2 \cdot 10^{23}$ Schritte bzw. $10^{12}$ Tage benötigt. □

Wir kommen nun zum *Rivest-Shamir-Adleman-Schema (RSA-Schema)* [78]. Es wird $n = pq$ gewählt, wobei $p$ und $q$ große Primzahlen sind. Aus Satz 3.7 (c) folgt

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

*Rivest*, *Shamir* und *Adleman* empfehlen, für die Erzeugung eines Paares $(e, d)$ mit $(ed) \bmod \varphi(n) = 1$ ein $d \in [\max(p,q)+1, n-1]$ mit ggt$(d, \varphi(n)) = 1$ zu wählen. Diese Bedingung wird von jeder Primzahl innerhalb des Intervalls erfüllt. Man berechnet $e$ =inv$(d, \varphi(n))$. Falls $e < \log_2 n$ ist, muß das Verfahren mit einem neuen Wert für $d$ wiederholt werden. Für $e = \log_2 n$ und $M = 2$ gilt nämlich

$$E(2) = 2^{\log_2 n} \bmod n = n \bmod n = 0$$

Das bedeutet, daß für $e < \log_2 n$ nicht jede von 1 verschiedene Nachricht, z. B. auch nicht 2, einer Reduktion modulo $n$ unterzogen wird.

Es muß gezeigt werden, daß durch $e$ und $d$ tatsächlich Chiffrierung und Dechiffrierung gegeben sind. Die Voraussetzungen von Satz 4.2.1 sind hier nicht für alle $M \in [0, n-1]$ erfüllt. Es gilt jedoch

**Satz 4.2.2** Es sei $n = pq \in \mathbb{N}$ mit Primzahlen $p$ und $q$, $p \neq q$. Es gelte weiter $e, d \in \mathbb{N}$ mit

$$ed \bmod \varphi(n) = 1,$$

und es sei $M \in [0, n-1]$ eine Nachricht. Dann folgt

$$(M^e \bmod n)^d \bmod n = M.$$

*Beweis.* Zu zeigen ist $M^{ed} \bmod n = M$. Gilt $M = 0$, so ist nichts zu beweisen. Im folgenden sei $M \in [1, n-1]$. Aus $(e \cdot d) \bmod \varphi(n) = 1$ folgt, daß ein $k \in \mathbb{N}$ existiert mit

$$ed = k \cdot \varphi(n) + 1 = k \cdot (p-1)(q-1) + 1.$$

Da $1 \leq M < p \cdot q$ ist und $p$ und $q$ prim sind, gilt

$$(p \not| M \wedge q \not| M) \vee (p \not| M \wedge q | M) \vee (p | M \wedge q \not| M).$$

Wir nehmen zunächst $p \not| M$ an. Dann ist ggt$(p, M) = 1$, und wir schließen nach Satz 3.9

$$M^{\varphi(p)} \bmod p = 1.$$

Wegen $\varphi(p) = p - 1$ folgt dann

$$\begin{aligned} M^{ed} \bmod p &= M^{k(p-1)(q-1)+1} \bmod p \\ &= \left((M^{p-1} \bmod p)^{k(q-1)} \cdot M\right) \bmod p \\ &= M \bmod p. \end{aligned}$$

Diese Gleichung gilt auch für $M \bmod p = 0$, also für $p|M$. Wir erhalten also in jedem Fall, daß ein $j \in \mathbb{N}$ existiert mit

$$M^{ed} = jp + M.$$

Bei entsprechender Argumentation für $q$ schließen wird die Existenz eines $j' \in \mathbb{N}$ mit

$$M^{ed} = j'q + M.$$

Aus diesen beiden Gleichungen für $M^{ed}$ folgt $jp = j'q$. Da $p$ und $q$ prim sind, existiert ein $r \in \mathbb{N}$ mit

$$j = rq \text{ und } j' = rp.$$

Somit ist

$$M^{ed} = rpq + M = rn + M$$

und damit

$$M^{ed} \bmod n = M. \quad \square$$

**Beispiel 4.2.3** Es sei $p = 5$ und $q = 7$ mit $n = pq = 5 \cdot 7 = 35$. Dann ergibt sich

$$\varphi(n) = (p-1)(q-1) = (5-1)(7-1) = 4 \cdot 6 = 24.$$

Wir wählen $d = 11$ und berechnen $e =$inv$(11, 24) = 11$ (speziell gilt hier $e = d$). Die Chiffrierung von $M = 2$ liefert

$$C = 2^{11} \bmod 35 = 2048 \bmod 35 = 18.$$

Die Dechiffrierung ergibt wieder

$$18^{11} \bmod 35 = 2 = M. \quad \square$$

**Beispiel 4.2.4** Es sei $p = 53$ und $q = 61$ und damit $n = pq = 53 \cdot 61 = 3233$. Daraus folgt

$$\varphi(n) = (p-1)(q-1) = (53-1)(61-1) = 52 \cdot 60 = 3120.$$

Wir wählen $d = 791$ und berechnen $e =$inv$(791, 3120) = 71$. Der gewählte Klartext

$$M = \text{RENAISSANCE_}$$

wird so eingeteilt, daß je zwei Buchstaben einen Block von vier Ziffern liefern: A=00, B=01, ..., Z=25, _=26. In der Praxis wird jeder Buchstabe durch einen 8-Bit-ASCII-Code dargestellt. Damit erhalten wir für unseren Klartext die Codierung

$$\begin{array}{rcccccccc} M & = & \text{RE} & \text{NA} & \text{IS} & \text{SA} & \text{NC} & \text{E_} \\ & & 1704 & 1300 & 0818 & 1800 & 1302 & 0426\,. \end{array}$$

Die Chiffrierung erfolgt blockweise. Für RE erhalten wir

$$1704^{71} \bmod 3233 = 3106,$$

für NA

$$1300^{71} \bmod 3233 = 100\ .$$

Insgesamt ergibt sich

$$C = \ 3106\ 0100\ 0931\ 2691\ 1984\ 2927. \quad \square$$

$\varphi(n)$ kann ohne Kenntnis der Primfaktoren $p$ und $q$ praktisch nicht bestimmt werden. Dadurch kann $d$ auch dann geheimgehalten werden, wenn $e$ und $n$ öffentlich sind. Die Sicherheit des Systems basiert auf der Schwiergkeit, $n$ in $p$ und $q$ zu faktorisieren. Der schnellste 1979 bekannte Faktorisierungsalgorithmus (Schroeppel) benötigt

$$T \sim e^{\sqrt{\ln(n)\cdot\ln(\ln(n))}}$$

Schritte. Dies ist die gleiche Anzahl, die zum Berechnen des diskreten Logarithmus erforderlich ist. Wie das Beispiel 4.2.2 zeigt, würde die Faktorisierung bei 1 Schritt pro $\mu$sec etwa $10^{12}$ Tage dauern, wenn für $p$ und $q$ je 100-stellige Zahlen gewählt werden, also $n$ 200-stellig ist. Aufgrund dieser langen Rechenzeiten hat es Rivest 1977 gewagt, einen kurzen Text mit einer Zahl $n$ von 428 Bits (129 Dezimalstellen) zu verschlüsseln und als Rätsel zu veröffentlichen, mit einem Lösungsanreiz von einigen Dollars. Er hielt eine Zeit von 40 Quadrillionen ($40 \cdot 10^{24}$ )Jahre zur Dechiffrierung dieser Zahl für notwendig. Nun kann man aber die Entwicklung der Mathematik nicht außer acht lassen. 1994 wurde die Zahl, unter Verwendung neuerer mathematischer Ergebnisse, zerlegt und damit der Text geknackt. Mit 1600 Rechnern im Internet wurde innerhalb von 8 Monaten der Klartext ermittelt. Die Faktorisierung einer 512-Bit-Zahl hätte noch 100-mal so lange gedauert. Inzwischen gibt es noch etwas bessere mathematische Theorien zur Primzahlzerlegung, aber bei 1024 Bits für die Zahl $n$ ist man hoffentlich auch in Zukunft auf der sicheren Seite. Schnelleren Rechnern kann man durch größere Schlüssellänge begegnen. Problematisch wäre nur ein großer mathematischer Durchbruch, der zu einem neuen schnellen Algorithmus zur Primzahlzerlegung führen würde. Wenn ein solcher Algorithmus gefunden wird, ist das RSA-Verfahren hinfällig und große Teile des Geschäfts- und Bankwesens brechen zusammen. Wenn wir jedoch mit einem solchen Durchbruch nicht rechnen, ist das RSA-Verfahren gut für Public-Key-Kryptosysteme geeignet.

Die Frage ist natürlich, wie man so große Primzahlen erhält. Die *Suche großer Primzahlen* wird im folgenden beschrieben. Aufgrund des Primzahlsatzes, der etwas über die Dichte

der Primzahlen aussagt, kann der folgende heuristische Algorithmus zum Finden großer Primzahlen mit z. B. $250$ binären Stellen benutzt werden. Man erzeuge zufällig eine Folge

$$m = \alpha_1, \alpha_2, \ldots, \alpha_{250}, \alpha_i \in \{0, 1\} \text{ und } \alpha_1 = 1,$$

und betrachte $m$ als eine Binärzahl. Wir nehmen an, daß $m$ gerade ist. Dann müssen fortlaufend die Summen $m + 1, m + 3, \ldots$ gebildet werden. Für jedes der $m + i$ wird ein Primzahltest durchgeführt, wie er z. B. unten in Algorithmus 4.2.1 beschrieben ist. Die Dichte der Primzahlen um $2^{250}$ liegt bei

$$\frac{1}{\ln n} > \frac{1}{250}.$$

Dadurch ist eine Abbruchregel motiviert. Wenn innerhalb von 500 Versuchen keine Primzahl gefunden wurde, dann wird das Verfahren mit einer neuen zufälligen Folge $m$ begonnen.

Ein Primzahltest kann mit dem folgenden probabilistischem Algorithmus ([70], [72]) durchgeführt werden.

**Algorithmus 4.2.1**
Es sei $n > 1$ ungerade mit $n - 1 = 2^l m$ ($m$ ungerade).
Wähle zufällig ein $x$ mit $1 \leq x < n$;
$x_0 := x^m \bmod n$;
**if** $x_0 = 1$ **or** $x_0 = n - 1$
**then** write(„$n$ ist eine Primzahl“);
**goto** ENDE
**end**;
**for** $i := 1$ **to** $l - 1$ **do**
$x_i := x_{i-1}^2 \bmod n$;
**if** $x_i = n - 1$
**then** write(„$n$ ist eine Primzahl“);
**goto** ENDE
**else if** $x_i = 1$
**then** write(„$n$ ist zusammengesetzt“); {sicher}
**goto** ENDE
**end**
**end**
**end**;
write(„$n$ ist zusammengesetzt“); {sicher}
ENDE: □

Dieser Algorithmus ist eine Verfeinerung von

**Algorithmus 4.2.2**
Es sei $n \in \mathbb{N}, n > 1$, gegeben.
Wähle zufällig $b$, $1 \leq b < n$;
**if** $W_n(b)$
**then** write(„$n$ ist zusammengesetzt“)
**else** write(„$n$ ist eine Primzahl“)
**end**

Dabei ist $W_n(b) :=$
(a) $1 \leq b < n$
(b) ($\alpha$) $b^{n-1} \not\equiv 1 \bmod n$ oder
($\beta$) es existiert ein $i$ mit $2^i | (n-1)$ und $1 < \text{ggt}(b^{\frac{n-1}{2^i}} - 1, n) < n$. $\square$

Man sagt, daß $b$ ein Zeuge der Zusammengesetztheit von $n$ ist. Wenn die Bedingung $(\alpha)$ gilt, dann kann nach der Fermatschen Relation (Satz 3.8) $n$ keine Primzahl sein. Weiter ist klar ersichtlich, daß eine Zahl $n$, die die Bedingung $(\beta)$ erfüllt, ebenfalls keine Primzahl sein kann. Falls $W_n(b)$ erfüllt ist, ist also die Antwort „$n$ ist zusammengesetzt“ in jedem Fall richtig. Andernfalls wird „$n$ ist eine Primzahl“ ausgegeben. Das kann aber auch dann eintreten, wenn $n$ zusammengesetzt ist.

Wir wollen uns überlegen, warum Algorithmus 4.2.1 ein entsprechendes Verhalten wie Algorithmus 4.2.2 hat. Gilt $x_0 = x^m \bmod n = 1$, so folgt $x^{n-1} \bmod n = 1$. Wegen $m = \dfrac{n-1}{2^l}$ ist weiter $\text{ggt}(x^{\frac{n-1}{2^l}} - 1, n) = n$. Insgesamt ist also die Bedingung (b) nicht erfüllt, und der Schluß von Algorithmus 4.2.1 in diesem Fall, daß $n$ prim ist, ist gleich dem Schluß von Algorithmus 4.2.2.

Es sei nun $n = 2^l m + 1$ eine Primzahl mit $x^m \bmod n \neq 1$. Dann endet die Folge

$$x^m, x^{2m}, x^{4m}, \ldots, x^{2^l m} \pmod n$$

mit 1, da nach Satz 3.8 $x^{2^l m} \bmod n = x^{n-1} \bmod n = 1$ gilt. Die einzige Zahl, die quadriert 1 liefert, ist außer 1 die Zahl $n-1$. Da nämlich $n$ eine Primzahl ist, hat die Gleichung $y^2 \bmod n = 1$ genau zwei Lösungen, und zwar $y = 1$ und $y = -1 \bmod n = n-1$. Der ersten 1 in dieser Folge geht also die Zahl $n-1$ unmittelbar voran. Deswegen wird im Algorithmus, falls $n-1$ auftaucht (auch, falls $x_0 = n-1$), „$n$ ist eine Primzahl“ ausgegeben. Diese Aussage kann jedoch falsch sein.

Geht dagegen der 1 nicht unmittelbar $n-1$ voran, so ist man sicher, daß $n$ zusammengesetzt ist. Das ergibt sich im Algorithmus, falls man für $i \geq 1$ auf $x_i = 1$ stößt. Das ist nur möglich für $x_{i-1} \neq n-1$, da andernfalls für $i-1$ die Schleife abgebrochen worden wäre. Wird die Schleife bis zum Ende durchlaufen und gilt $x_{l-1} \neq n-1$, $x_{l-1} \neq 1$, dann folgt, und zwar unabhängig davon, ob $x_l = 1$ oder $x_l \neq 1$ gilt, daß $n$ zusammengesetzt ist.

**Satz 4.2.3** Der Algorithmus 4.2.1 benötigt höchstens $4 \log_2 n$ Berechnungsschritte. Wenn $n$ eine Primzahl ist, dann ist die Aussage des Algorithmus immer richtig. Ist $n$ dagegen zusammengesetzt, so ist die Wahrscheinlichkeit eines Irrtums $< \frac{1}{4}$.

*Beweis.* Wenn bei einer Multiplikation eine Zahl $d \geq n$ erhalten wird, so muß $d_1 = d \bmod n$ bestimmt und damit fortgefahren werden. Somit kommen nur Multiplikationen von Zahlen $a, b < n$ und Divisionen von $d < n^2$ durch $n$ mit Rest $d_1$ vor. Für jede Multiplikation werden also zwei Schritte benötigt. Von $x_0$ ausgehend sind noch $l-1$ Multiplikationen erforderlich. Die Berechnung von $x_0 = x^m \bmod n$ erfolgt nach Satz 3.3 mit höchstens $2 \log_2 m + 1$ Multiplikationen. Daraus ergibt sich, daß die Anzahl der Schritte

$$\begin{aligned} &\leq 2(l - 1 + 2\log_2 m + 1) = 2(l + 2\log_2 m) \\ &\leq 2(2l + 2\log_2 m) = 4(l + \log_2 m) \\ &= 4\log_2(n-1) < 4\log_2 n \end{aligned}$$

ist. Die weiteren Aussagen werden in [72] bewiesen. $\square$

Es ist also möglich, daß der Algorithmus 4.2.1 eine Zahl als Primzahl erkennt, obwohl die Zahl zusammengesetzt ist. Wird der Test 100-mal wiederholt, dann sinkt die Irrtumswahrscheinlichkeit auf $< \frac{1}{4^{100}}$ ab. Die Antwort „$n$ ist zusammengesetzt“ ist immer richtig. Wenn ein Primzahlkandidat nach diesem Verfahren gefunden ist, kann man zur Bestätigung den deterministischen Primzahltest von *Rumeley-Adleman* anwenden, der, falls er hält, die Antwort in $O((\log n)^{c \log(\log(\log n))})$ Schritten liefert (siehe z. B. [50], S. 76).

Besserer Schutz gegen eine Faktorisierung kann durch zusätzliche Vorsichtsmaßnahmen erreicht werden, z. B.:

(1) $p$ und $q$ sollten sich in der Länge um einige Stellen unterscheiden.
(2) $(p-1)$ und $(q-1)$ sollten große Primfaktoren enthalten.
(3) ggt$(p-1, q-1)$ sollte klein sein.

Wie findet man nun Primzahlen, die (2) erfüllen? Zunächst wird nach dem obigen Verfahren eine große Primzahl $p'$ erzeugt. Danach wird solange $p = ip' + 1, i = 2, 4, 6, \ldots$, gebildet, bis $p$ eine Primzahl ist. Die Sicherheit wird zusätzlich dadurch vergrößert, daß auch $p' - 1$ einen großen Primfaktor besitzt. Man hat gezeigt, daß das RSA-Schema gebrochen werden kann, wenn $p$ und $q$ nicht sorgfältig gewählt werden. Wir betrachten zwei solcher Möglichkeiten:

(1) Für gewisse Schlüssel ergibt eine mehrfache Anwendung der Chiffrierung den Klartext. Es sei etwa $C_0 = M^e \bmod n$ der Chiffretext und $(e, n)$ der öffentliche Schlüssel. Ein Kryptoanalytiker erhält vielleicht durch

$$C_i = C_{i-1}^e \bmod n,\ i = 1, 2, \ldots$$

für irgendein $i$ einen sinnvollen Klartext. Das ist aber nur dann nützlich, wenn $i$ „vernünftig“ klein ist, etwa $i < 10^6$. *Rivest* [76] hat gezeigt, daß die Wahrscheinlichkeit eines entsprechenden erfolgreichen Angriffs sehr klein ist, falls $n$ so gewählt wird, daß $p - 1$ einen großen Primfaktor $p'$ enthält und $p' - 1$ wiederum einen großen Primfaktor $p''$. Für $p > 10^{90}$ ist die Wahrscheinlichkeit des Brechens der Chiffre $< 10^{-90}$.

(2) Nach *Blakley, Blakley* und *Borosh* ([8], [9]) gilt für alle $e$ und $n$, daß $M^e \bmod n = M$ für wenigstens 9 Nachrichten $M$ ist. Die Wahrscheinlichkeit, gerade eine derartige Nachricht zu senden, ist sehr klein. Bei ungünstiger Wahl von $e$ und $n$ können es jedoch wesentlich mehr als 9 Klartexte sein. Zur Abhilfe wird die Wahl einer sicheren Primzahl empfohlen. Eine Primzahl $p$ gilt als sicher, wenn $p = 2p' + 1$ ist, wobei $p'$ eine ungerade Primzahl ist.

Das Brechen des RSA-Schemas kann nicht schwieriger als die Faktorisierung sein, da ein schneller Faktorisierungsalgorithmus auch eine effiziente kryptoanalytische Prozedur liefert. *Rabin* hat eine Variante des RSA-Verfahrens geliefert [71], bei der die kryptoanalytische Prozedur äquivalent zur Faktorisierung ist.

Wir gehen jetzt auf den *Speicher- und Zeitbedarf* des RSA-Verfahrens ein. Es seien $n, d$ und $e$ je 200 dezimale Stellen (664 Bits) groß. Dann müssen ungefähr $2000$ Bits pro Benutzer gespeichert werden. Da $e$ und $n$ öffentlich sind, ist es nötig, ungefähr 1300 Bits pro Benutzer im öffentlichen Directory abzulegen. Der DES benötigt dagegen nur 56 (bzw. 112) Bits pro Benutzer.

Zum Chiffrieren oder Dechiffrieren einer 664-Bit-Zahl werden etwa 1000 Multiplikationen $(< \log_2 n)$ durchgeführt. *Rivest* [77] hat 1980 einen speziellen Chip entwickelt, der einige 1000 Bits/Sekunde verarbeiten kann. *Sedlak* (TU Braunschweig, Diplomarbeit 1985, [89], siehe auch [90], [91]) entwickelte einen Chip, der im ungünstigsten Fall 64000 Bits/Sekunde verarbeitet. Dabei handelt es sich um einen aus 80000 Transitoren bestehenden VLSI-Chip in 2-$\mu$m-CMOS-Technologie mit einer Chipfläche von $5,2$ mm $\times 5,2$ mm.

Darauf aufbauend wurden von der Firma Siemens verschiedene Chips entwickelt (z.B. SLE44C200), die in verschiedene Smart Cards integriert worden sind.

Die Chiffrier- und Dechiffrierfunktionen sind gegenseitig invers. Folglich kann das RSA-Verfahren nach Abschnitt 4.1 für Geheimhaltung und Authentizität benutzt werden. Dies wollen wir hier noch einmal zusammenstellen. Jeder Benutzer A besitzt $n_A$, $e_A$ und $d_A$. Davon hält er $d_A$ geheim, wohingegen $n_A$ und $e_A$ im öffentlichen Directory registriert sind. Für die *Geheimhaltung* sendet B an A eine geheime Nachricht $M$ durch

$$E_A(M) = M^{e_A} \bmod n_A.$$

A dechiffriert diese Nachricht durch

$$D_A(E_A(M)) = M^{e_A d_A} \bmod n_A = M.$$

Für die *Authentizität* sendet A an B eine signierte Nachricht $M$ durch

$$D_A(M) = M^{d_A} \bmod n_A\ .$$

B prüft die Authentizität durch

$$E_A(D_A(M)) = M^{d_A e_A} \bmod n_A = M.$$

Wenn sich ein vernünftiger Klartext ergibt, kann nur A diese Nachricht geschickt haben.

Für gleichzeitige *Geheimhaltung und Authentizität* sendet A an B eine signierte und geheime Nachricht $M$ durch $C = E_B(D_A(M))$. Ein Problem stellen jedoch die i. allg. verschiedenen Moduli $n_A$ und $n_B$ dar. Unter der Annahme, daß $n_A > n_B$ gilt, kann sich auch $D_A(M) \notin [0, n_B - 1]$ ergeben. $D_A(M) \bmod n_B$ liefert eine falsche Dechiffrierung. Eine Lösungsmöglichkeit bildet die Entblockung von $D_A(M)$. Das bedeutet, daß $D_A(M)$ im normalen Alphabet wiederhergestellt wird und dann ein neuer Block unter Benutzung von $n_B$ erzeugt wird. Ohne Entblockung (*Rivest, Shamir* und *Adleman*) ist aber Geheimhaltung und Authentizität auch zu erreichen. Es sei $h$ ein Schwellwert, etwa $h = 10^{99}$. Jeder Benutzer hat zwei Transformationspaare $(E_{A_1}, D_{A_1})$ für die Signatur und $(E_{A_2}, D_{A_2})$ für die Geheimhaltung mit $n_{A_1} < h < n_{A_2}$. A sendet an B eine signierte und geheime Nachricht durch

$$C = E_{B_2}(D_{A_1}(M)).$$

Aufgrund der oben angegebenen Bedingung ist $C$ berechenbar. B dechiffriert diese Nachricht und überprüft die Authentizität durch

$$E_{A_1}(D_{B_2}(C)) = E_{A_1}(D_{B_2}(E_{B_2}(D_{A_1}(M)))) = M.$$

Zwischenwerte, die bei der Dechiffrierung auftreten, liegen wegen der Art der Verschlüsselung in den richtigen Intervallen.

Ein anderer Ansatz stammt von *Konfelder* [49]. A sendet an B eine signierte und geheime Nachricht $M$ durch

$$\begin{aligned} C &= E_B(D_A(M)) \text{ für } n_A < n_B \text{ bzw.} \\ C' &= D_A(E_B(M)) \text{ für } n_A > n_B. \end{aligned}$$

Die Voraussetzung für die Kommunikation ist die gegenseitige Kenntnis von $n_A$ und $n_B$. B dechiffriert die Nachricht und überprüft deren Authentizität durch

$$\begin{aligned} E_A(D_B(C)) &= E_A(D_B(E_B(D_A(M))) = M \text{ für } n_A < n_B \text{ bzw.} \\ D_B(E_A(C')) &= D_B(E_A(D_A(E_B(M))) = M \text{ für } n_A > n_B. \end{aligned}$$

Bei einem Konflikt zwischen A und B, ob die Signatur von A echt ist, die Nachricht also von A stammt, ist eine Überprüfung möglich. Für $n_A < n_B$ müssen $X = D_B(C)$ und $M$ einem Richter übergeben werden. Dieser berechnet $M' = E_A(X)$ und überprüft, ob $M' = M$ gilt. Für $n_A > n_B$ müssen $C'$ und $M$ überprüft werden, da B die geheime Transformation $D_B$ nicht dem Richter geben möchte. Der Richter berechnet

$$X = E_B(M) \text{ und } X' = E_A(C') = E_A(D_A(E_B(M)))$$

und überprüft, ob $X = X'$ ist.

Das RSA-Verfahren kann auch in einer Umgebung angewendet werden, in der die Moduli $n$ und die öffentlichen und die geheimen Transformationen durch ein Vertrauenszentrum verteilt werden. Dabei ist jedoch darauf zu achten, daß die verschiedenen Benutzer paarweise verschiedene Moduli $n$ erhalten. Anderenfalls können die geheimen Transformationen leicht berechnet werden. Dies zeigt

**Satz 4.2.4** Es werde das RSA-Verfahren mit dem gemeinsamen Modulus $n$ benutzt. Alice kenne $n$, $e_A$, $e_B$ und $d_A$, Bob kenne $n$, $e_A$, $e_B$ und $d_B$. Mit einem Algorithmus der Komplexität $O(\log^2 n)$ kann Alice $d_B$ und Bob $d_A$ berechnen (ohne $n$ zu faktorisieren).

*Beweis.* Wir zeigen, wie Bob $d_A$ berechnen kann. Wegen $(e_B d_B) \bmod \varphi(n) = 1$ existiert ein $k \in \mathbb{N}$ mit

$$e_B d_B - 1 = k \cdot \varphi(n).$$

Bob kennt zwar nicht $k$, jedoch $n$, $e_B$, $d_B$ und $e_A$. Wir zeigen zunächst, wie Bob einen Teiler $t$ von $e_B d_B - 1$ mit der Eigenschaft

$$\alpha = \frac{e_B d_B - 1}{t} \text{ und } \operatorname{ggt}(\alpha, e_A) = 1$$

berechnen kann. Es sei

$$g_0 = e_B d_B - 1, \quad h_0 = \operatorname{ggt}(g_0, e_A).$$

Für $i \geq 1$ definieren wir induktiv

$$g_i = \frac{g_{i-1}}{h_{i-1}}, \quad h_i = \operatorname{ggt}(g_i, e_A).$$

Für $h_i = 1$ setzen wir $t = h_0 \cdots h_i$ und erhalten damit $\alpha = g_i$. Für $h_i \geq 2$ gilt $g_{i+1} \leq \frac{g_i}{2}$. Folglich findet man $h_i = 1$ in $O(\log(e_B d_B - 1)) = O(\log n)$ Schritten. In jedem Schritt wird der Euklidische Algorithmus 3.2 aufgerufen, der, wie im Anschluß an den Beweis von Satz 3.11 angemerkt wurde, den Zeitbedarf $O(\log n)$ hat. Für die Berechnung von $t$ benötigt Bob also $O(\log^2 n)$ Schritte.

Anschließend berechnet Bob mit dem Euklidischen Algorithmus 3.3 Zahlen $a$, $b \in \mathbb{Z}$ mit

$$a\alpha + be_A = 1.$$

Diese Rechnungen haben wieder einen Zeitbedarf der angegebenen Komplexität.

Da jeder Primteiler von $t = h_0 \cdots h_i$ nach der obigen Konstruktion zu $e_A$ gehört, folgt aus $\operatorname{ggt}(e_A, \varphi(n)) = 1$ auch die Gültigkeit von $\operatorname{ggt}(t, \varphi(n)) = 1$. Das bedeutet, daß in der Gleichung

$$\alpha = \frac{(e_B d_B - 1)}{t} = \frac{k \cdot \varphi(n)}{t}$$

$\frac{k}{t} \in \mathbb{Z}$ ist und somit $\varphi(n)$ die Zahl $\alpha$ teilt. Wir erhalten damit

$$be_A \bmod \varphi(n) = 1.$$

Folglich kann $b \bmod n$ als $d_A$ benutzt werden. □

Man kann zeigen (siehe [83], Theorem 4.1, p. 143), daß sich dieser Algorithmus in einen probabilistischen Algorithmus zur Faktorisierung von $n$ verwandeln läßt, der polynomialen Zeitbedarf hat.

Der Zeitbedarf für die Ver- und Entschlüsselung des RSA-Verfahrens ist für eine sehr schnelle Übertragung von Daten, insbesondere von großen Datenmengen, zur Zeit nicht geeignet. Die schnellsten DES-Implementierungen schaffen dagegen viele Millionen von Bits pro Sekunde. Deswegen wird das RSA-Verfahren überwiegend nur zur Signatur von Nachrichten verwendet. Auf diesem Feld wird auch der Chip von *Sedlak* (siehe Seite 67) eingesetzt. Außerdem werden häufig hybride Verfahren verwendet, bei denen z.B. das RSA-Verfahren nur am Anfang zur Übermittlung des Schlüssels für ein konventionelles Verfahren, z.B. dem DES, benutzt wird. Anschließend wird dann mit dem DES die Kommunikation fortgesetzt.

# 5 Kryptographische Protokolle

## 5.1 Mental Poker (Kopf-Poker)

Der vielleicht erste Versuch, im Kopf Poker zu spielen, wurde 1933 von Niels Bohr während eines Ski-Urlaubs in Oberaudorf gemacht. Bohr, sein Sohn Christian sowie Felix Bloch, Carl Friedrich von Weizsäcker und Werner Heisenberg versuchten, ohne Karten zu pokern. Der Erfolg soll allerdings nicht allzu groß gewesen sein. Da die Beteiligten sich später anderen wichtigen Aufgaben widmeten, wurde das Kopf-Poker-Spiel von ihnen nicht weiterentwickelt.

Das Poker stellt ein Modell für ein System dar, in dem Benutzer nur partielle Informationen über die dynamische Zuordnung der Resourcen haben.

Die *Regeln* sind wie beim üblichen Poker gegeben, jedoch existieren keine Karten, und die Kommunikation zwischen den Spielern findet ausschließlich durch die Übertragung von Nachrichten statt. Jegliche andere verbale Kommunikation ist untersagt. Den Spielern ist es nicht verboten, die anderen Spieler zu betrügen. Jedoch müssen gewisse *Fairplay-Regeln* eingehalten werden.

(1) Das Spiel beginnt mit dem fairen Austeilen der „Karten“. Wenn dies durch den Austausch einer Folge von Nachrichten erreicht ist, muß gelten:
   (a) Die Spieler sollten ihre eigenen, aber nicht die Karten der anderen kennen.
   (b) Jede Hand muß zu den anderen Händen disjunkt sein. Das ist gleichbedeutend damit, daß jede Karte nur einmal vergeben wird.
   (c) Alle möglichen Hände müssen für jeden Spieler gleichwahrscheinlich sein.

(2) Die Verteilung neuer Karten während des Spiels muß ebenso fair erfolgen wie unter (1) beschrieben. Die Spieler müssen ihre Karten aufdecken können, ohne die Sicherheit der übrigen Karten zu gefährden.

(3) Am Ende jedes Spiels müssen die Spieler prüfen können, ob das Spiel fair verlaufen ist und ihre Gegner nicht betrogen haben.

*Rivest, Shamir* und *Adleman* ([80], [94]) haben 1978 ein Modell für Kopf-Poker angegeben, das eine kommutative Chiffre benutzt, wie z. B. die Exponentiationschiffre, bei der die Spieler denselben Modulus $n$ verwenden. Folgende vereinfachende Annahmen wurden getroffen: Am Spiel nehmen nur zwei Spieler teil, und zwar Alice und Bob. Jeder der Spieler hat einen geheimen Schlüssel, der erst am Ende des Spiels aufgedeckt wird. Die geheimen Transformationen $E_A$ bzw. $D_A$ von Alice und $E_B$ bzw. $D_B$ von Bob liefern kommutative Chiffriertransformationen, also

$$E_A(E_B(M)) = E_B(E_A(M)).$$

Die 52 Karten werden durch Nachrichten dargestellt, nämlich

$$\begin{aligned} M_1&: \text{„Karo 2“},\\ M_2&: \text{„Karo 3“},\\ &\dots\\ M_{52}&: \text{„Kreuz As“}\ . \end{aligned}$$

Bob beginnt das Spiel mit der Verteilung der Karten. Ein *Protokoll zum fairen Austeilen* der Karten sieht nun so aus:

**Protokoll 5.1.1**

(1) Bob chiffriert die 52 Nachrichten (Karten) durch

$$E_B(M_i),\ i = 1, 2, \ldots, 52.$$

Anschließend mischt er zufällig den Stapel der chiffrierten Nachrichten und sendet ihn an Alice.

(2) Alice wählt zufällig 5 der chiffrierten Nachrichten aus und sendet sie an Bob zurück. Bob dechiffriert sie und erhält dadurch seine eigene Hand.

(3) Danach wählt Alice zufällig fünf weitere chiffrierte Nachrichten $C_1, C_2, \ldots, C_5$. Sie bildet

$$C_i' = E_A(C_i),\ i = 1, 2, \ldots, 5,$$

und sendet die $C_i'$ an Bob.

(4) Bob dechiffriert jede der Nachrichten $C_i'$ durch

$$D_B(C_i') = D_B(E_A(E_B(M_j))) = D_B(E_B(E_A(M_j))) = E_A(M_j)$$

für 5 durch Alice Wahl bestimmte $M_j$ und sendet das Ergebnis zurück an Alice.

(5) Alice dechiffriert sie mit $D_A$ und erhält ihre eigene Hand. □

Während des Spiels können zusätzliche Karten durch Wiederholung derselben Prozedur ausgeteilt werden. Am Ende des Spiels werden die Schlüssel aufgedeckt, und durch Nachrechnen kann bewiesen werden, ob jemand betrogen hat.

Wie schon oben gesagt wurde, kann Kopf-Poker mit einer Exponentiationschiffre gespielt werden. Bob und Alice einigen sich auf einen großen Modulus $n$, z.B. auf eine große Primzahl, mit entsprechendem $\varphi(n)$. Jeder bestimmt ein privates Paar $(e, d)$ mit $(ed) \bmod \varphi(n) = 1$. Mit dem Paar $(e, d)$ sind wie üblich auch $E_A$ und $D_A$ bzw. $E_B$ und $D_B$ bestimmt.

*Lipton* [53] zeigt jedoch, daß bei dieser Chiffriermethode ein Betrug möglich ist. Eine der Möglichkeiten benutzt quadratische Reste.

**Definition 5.1.1** Ein $a \in \mathbb{N}$ heißt *quadratischer Rest* modulo $n$, wenn $\text{ggt}(a, n) = 1$ gilt und ein $x \in \mathbb{N}$ mit $x^2 \bmod n = a$ existiert. Ist die Bedingung nicht erfüllt, heißt $a$ ein *quadratischer Nichtrest* modulo $n$. Man setze

$$R_2 := \{a \in \mathbb{N} \mid a \text{ ist ein quadratischer Rest modulo } n\}. \quad \square$$

Ist $a$ ein quadratischer Rest modulo $n$, so ist $x$ die Quadratwurzel von $a$ modulo $n$.

**Satz 5.1.1** Es sei $a \in \mathbb{N}$ mit $0 < a < n$ gegeben. Für $n$ seien $e$ und $d$ mit $(ed) \bmod \varphi(n) = 1$ bestimmt, und es gelte $E_K(a) = a^e \bmod n$. Dann gilt

$$a \in R_2 \Longleftrightarrow E_K(a) \in R_2.$$

*Beweis.* Es gelte $a \in R_2$. Dann existiert ein $x \in \mathbb{N}$ mit $x^2 \bmod n = a$. Wegen

$$E_K(a) = a^e \bmod n = (x^2)^e \bmod n = (x^e)^2 \bmod n$$

folgt $E_K(a) \in R_2$.

Es sei umgekehrt $E_K(a) \in R_2$. Dann existiert ein $y \in \mathbb{N}$ mit $y^2 \bmod n = E_K(a)$. Wegen

$$a = (E_K(a))^d \bmod n = (y^2)^d \bmod n = (y^d)^2 \bmod n$$

folgt $a \in R_2$. □

**Beispiel 5.1.1** Es sei $n = 11$. Wegen $5^2 \bmod 11 = 3$ ist 3 ein quadratischer Rest modulo 11. Für $e = 3$ ist nach Satz 5.1.1 $3^3 \bmod 11 = 5$ ebenfalls ein quadratischer Rest. In der Tat gilt $4^2 \bmod 11 = 5$. □

Wenn Alice feststellen kann (siehe Satz 5.1.4), welche Karten Nachrichten aus $R_2$ beinhalten, kann sie das Ergebnis aus Satz 5.1.1 ausnutzen. Nachdem Bob die Nachrichten chiffriert und gemischt hat, kann Alice sie zwar nicht dechiffrieren, sie weiß jedoch, welche der Nachrichten bzw. der chiffrierten Nachrichten aus $R_2$ stammen. Falls der Modulus $n$ prim ist (Pohlig-Hellman-Schema), ist die Wahrscheinlichkeit, daß eine Nachricht in $R_2$ liegt, gleich $\frac{1}{2}$ (siehe Satz 5.1.3). Dieses Ergebnis liefert Alice ein zusätzliches Bit an Information pro Karte, das dazu beitragen kann, das Spiel zu gewinnen. Falls sie z. B. feststellt, daß die Klartextnachrichten für alle vier Asse ein quadratischer Rest sind, wählt sie natürlich quadratische Reste für sich und quadratische Nichtreste für Bob. Wir beweisen die für diese Überlegungen notwendigen Sätze.

**Satz 5.1.2** Es sei $p > 2$ eine Primzahl und $a \in \mathbb{N}$ mit $0 < a < p$. Dann besitzt $x^2 \bmod p = a$ für $a \in R_2$ zwei, für $a \notin R_2$ keine Lösungen.

*Beweis.* Es sei $a \in R_2$. Nach Definition 5.1.1 existiert eine Lösung $x_1 \in \mathbb{N}$. Dann folgt

$$\begin{aligned}(p - x_1)^2 \bmod p &= (p^2 - 2x_1p + x_1^2) \bmod p \\ &= x_1^2 \bmod p \\ &= a.\end{aligned}$$

Damit ist $p - x_1$ ebenfalls eine Lösung. Die Lösungen sind jedoch verschieden, weil $p - x_1 = x_1$ höchstens für $2|p$ möglich wäre. Weitere Lösungen gibt es nicht, da $f(x) = x^2 - a$ im Körper $\mathbb{Z}_p$ höchstens zwei Nullstellen hat. Für $a \notin R_2$ existiert aufgrund von Definition 5.1.1 keine Lösung. □

**Satz 5.1.3** Es sei $p > 2$ eine Primzahl. Dann existieren $\frac{p-1}{2}$ quadratische Reste modulo $p$ und $\frac{p-1}{2}$ quadratische Nichtreste modulo $p$.

*Beweis.* Die Reste

$$1^2, 2^2, \ldots, \left(\frac{p-1}{2}\right)^2 \bmod p$$

sind offenbar quadratische Reste. Sie sind alle voneinander verschieden, da jedes $n^2 \bmod p$ genau die verschiedenen Lösungen $n$ und $p - n$ hat. Da für jedes $a \in R_2$ wenigstens eine der Wurzeln $x_1$ und $p - x_1$ in $[1, \frac{p-1}{2}]$ liegt, existieren keine weiteren quadratischen Reste. Alle $a \in R_2$ kommen also in der obigen Folge vor. □

**Satz 5.1.4** Es sei $p > 2$ eine Primzahl und $a \in \mathbb{N}$, $0 < a < p$. Dann folgt

$$a^{\frac{p-1}{2}} \bmod p = \begin{cases} 1 & \text{für } a \in R_2 \\ p-1 & \text{sonst.} \end{cases}$$

*Beweis.* Nach dem Satz 3.8 (Fermat) gilt

$$\left(a^{p-1} - 1\right) \bmod p = 0.$$

Da $p$ ungerade ist, ergibt sich

$$\left(a^{\frac{p-1}{2}} + 1\right)\left(a^{\frac{p-1}{2}} - 1\right) \bmod p = \left(a^{p-1} - 1\right) \bmod p = 0.$$

Somit folgt

$$p \mid \left(a^{\frac{p-1}{2}} + 1\right) \vee p \mid \left(a^{\frac{p-1}{2}} - 1\right).$$

Da die Differenz der beiden Terme 2 ist, kann $p$ nicht beide teilen, und es folgt

$$a^{\frac{p-1}{2}} \equiv_p \pm 1.$$

Weiterhin sei $a \in R_2$. Dann existiert ein $x \in \mathbb{N}$ mit $a = x^2 \bmod p$. Wir erhalten

$$a^{\frac{p-1}{2}} \bmod p = (x^2)^{\frac{p-1}{2}} \bmod p = x^{p-1} \bmod p = 1,$$

da $p - 1$ die Ordnung der multiplikativen Gruppe $\mathbb{Z}_p$ ist. Daraus folgt, daß die $\frac{p-1}{2}$ quadratischen Reste Lösungen von $x^{\frac{p-1}{2}} \bmod p = 1$ sind. Da diese Gleichung im Körper $\mathbb{Z}_p$ höchstens $\frac{p-1}{2}$ Lösungen besitzt, sind die $\frac{p-1}{2}$ Nichtreste wegen $-1 \bmod p = p-1$ Lösungen von

$$x^{\frac{p-1}{2}} \bmod p = p - 1. \quad \square$$

Mit Hilfe dieses Satzes kann Alice, wie oben schon gesagt wurde, sehr einfach feststellen, welche der Karten aus $R_2$ stammt.

Wir geben an dieser Stelle noch einen weiteren Satz über quadratische Reste an, den wir später benötigen werden.

**Satz 5.1.5** Es seien $r, s \in \mathbb{N}$, $r, s > 1$, Zahlen mit ggt$(r, s) = 1$. Ein $a \in \mathbb{N}$ ist quadratischer Rest modulo $rs$ genau dann, wenn $a$ ein quadratischer Rest modulo $r$ und modulo $s$ ist.

*Beweis.* Es sei $a$ ein quadratischer Rest modulo $rs$. Dann existiert ein $x \in \mathbb{N}$ mit $a = x^2 \bmod (rs)$. Somit gilt $rs \mid (a - x^2)$. Wegen ggt$(r, s) = 1$ folgt daraus $r \mid (a - x^2)$ und $s \mid (a - x^2)$, d.h. $a = x^2 \bmod r$ und $a = x^2 \bmod s$. Folglich ist $a$ sowohl ein quadratischer Rest modulo $r$ als auch modulo $s$.

Umgekehrt seien $x_1, x_2 \in \mathbb{N}$ Zahlen mit $a = x_1^2 \bmod r$ und $a = x_2^2 \bmod s$. Wegen ggt$(r, s) = 1$ existiert nach Satz 3.14 ein $x \in \mathbb{N}$ mit $x \bmod r = x_1 \bmod r$ und $x \bmod s = x_2 \bmod s$. Es folgt $a = x^2 \bmod r = x^2 \bmod s$. Das bedeutet, daß sowohl $r$ als auch $s$ die Zahl $a - x^2$ teilen. Wegen ggt$(r, s) = 1$ erhalten wir daraus $rs \mid (a - x^2)$ und damit $a = x^2 \bmod (rs)$. Somit ist $a$ ein quadratischer Rest modulo $rs$. $\square$

Wir wollen nun noch weitere Bemerkungen zum Mental-Poker machen. Eine Chiffriertransformation kann auch andere Informationen über eine Nachricht $M = a$ bewahren, z. B. sei $R_t = \{a | x^t \bmod n = a\}$. Dann gilt

$$a \in R_t \Longleftrightarrow a^e \bmod n \in R_t.$$

*Lipton* [54] schlägt daher Modifikationen vor, die erzwingen, daß alle Nachrichten quadratische Reste sind.

Eine erste Methode besteht darin, weitere Bits an jede Nachricht $M$ anzuhängen, die sie zu einem quadratischen Rest machen. Die ursprüngliche Nachricht wird durch Streichen dieser Bits zurückgewonnen. Bei der zweiten Methode wird jede Nichtrestnachricht mit einem festen Nichtrest $w$ multipliziert. Das Produkt zweier Nichtreste ist natürlich ein quadratischer Rest. Die ursprüngliche Nachricht wird durch Multiplizieren mit $w^{-1}$ zurückgewonnen.

Das Modell des Kopf-Pokers zeigt, daß es Anwendungen gibt, für die es nicht ausreicht, daß der Chiffrieralgorithmus schwer berechenbar ist. Eine solche Anwendung benötigt einen Algorithmus, der nicht nur die Nachrichten, sondern auch deren mathematische Eigenschaften verbirgt. Aber auch *Liptons* Methoden bieten keine Garantie dafür, daß das Verfahren sicher ist.

*Goldwasser* und *Micali* [31] haben 1982 ein neues Protokoll für Kopf-Poker veröffentlicht, das beweisbar sicher ist. Allerdings funktioniert ihr Protokoll nur für zwei Spieler. *Barany* und *Füredi* [3] lieferten 1983 ein einfaches Protokoll für drei und mehr Spieler. Die Voraussetzungen für ein faires Spiel sind jedoch, daß kein Spieler mit einem anderen eine geheime Absprache getroffen hat. *Fortune* und *Merritt* [23] stellten 1984 ein Protokoll vor, bei dem die Sicherheit auch in diesem Fall gewährleistet ist. Allerdings wird jetzt ein vertrauenswürdiger Kartenverteiler benötigt. Solange dieser und mindestens ein weiterer Spieler fair spielen, können keine weiteren Spieler oder Gruppen von zusammenarbeitenden Spielern Informationen über die Karten erhalten, die nicht in ihren Händen sind.

## 5.2 Vergeßliche Übertragung (Oblivious Transfer)

1981 wurde von *Rabin* [73] ein Protokoll vorgeschlagen, mit dem Alice an Bob, falls beide sich an das Protokoll halten, ein beliebiges Geheimnis mit der Wahrscheinlichkeit $\frac{1}{2}$ übermitteln kann. Das bedeutet, daß Bob mit der Wahrscheinlichkeit $\frac{1}{2}$ das Geheimnis empfängt bzw. mit derselben Wahrscheinlichkeit nicht empfängt. Die Schwierigkeit liegt darin, daß Alice nicht weiß, ob Bob das Geheimnis empfangen hat oder nicht. Diese Unsicherheit muß von beiden akzeptiert werden. Nach *Blum* [10] wird dies beschrieben durch das *Oblivious-Transfer-Protokoll*.

**Protokoll 5.2.1**

(1) Alice sendet an Bob $n = pq$. $p$ und $q$ sind ungerade Primzahlen mit $p \neq q$, die Alice Geheimnis darstellen. Das könnten z. B. die geheimen Parameter für das RSA-Schema sein.

(2) Bob wählt zufällig ein $x \in \mathbb{N},\ 0 < x < n$, mit $\text{ggt}(x, n) = 1$ und sendet Alice

$(*)$ $$a = x^2 \bmod n\ .$$

(3) Alice, die $p$ und $q$ kennt, berechnet die vier Wurzeln von $a$, nämlich

$$x, n-x, y, n-y.$$

Danach wählt sie zufällig eine der Wurzeln aus und sendet sie an Bob.

(4) Falls Bob $y$ oder $n-y$ empfängt, kann er $p$ und $q$ aus $x$ und $y$ ($x$ hat er selber in (2) gewählt) durch

$$\text{ggt}(x+y, n) = p \text{ oder } q$$

berechnen. Falls er $x$ oder $n-x$ empfängt, lernt er nichts. □

*M. Fisher* hat bezüglich der Wahl von $x$ in Schritt (2) den Einwand erhoben, daß es eventuell eine Methode geben kann, mit deren Hilfe Bob für ein $a$ die Quadrateigenschaft feststellen kann, ohne $x$ zu kennen. Wenn also jemand eine solche Methode finden würde, dann wäre Protokoll 5.2.1 nicht mehr fair. Zur Vermeidung dieser Unsicherheit könnte man in Schritt (2) verlangen, daß Bob seine Kenntnis von $x$ mit der Eigenschaft $(*)$ beweist (siehe Protokoll 7.2.1).

Dies Protokoll stellt keine Einschränkung des allgemeinen Problems dar. Mit Hilfe der geheimen Primzahlen $p$ und $q$ kann ja jede beliebige Nachricht verschlüsselt werden, die dann das eigentlich zu übermittelnde Geheimnis darstellt.

Wir zeigen, wie Alice in (3) die Wurzeln berechnen und daß außerdem Bob, so wie es in (4) angegeben ist, $p$ oder $q$ ermitteln kann. Die Gleichung $(*)$ hat vier Wurzeln, denn nach Satz 3.13 ist jede Wurzel eine gemeinsame Lösung von

(a) $$x^2 \bmod p = a \bmod p \text{ und}$$
(b) $$x^2 \bmod q = a \bmod q.$$

Nach Satz 5.1.2 besitzt (a) die zwei Lösungen $x_1$ und $p-x_1$ im Intervall $[0, p-1]$ und (b) die zwei Lösungen $x_2$ und $q-x_2$ im Intervall $[0, q-1]$. Ihre Berechnung kann nach dem unten angegebenen Algorithmus 5.2.1 durchgeführt werden. Gemeinsame Lösungen in $[0, n-1]$ ergeben sich nach dem Chinesischen Restesatz (Satz 3.14) durch

$$x \bmod p = x_p \text{ bzw. } x \bmod q = x_q,$$

wobei für $x_p$ die Lösungen $x_1$ und $p-x_1$ und $x_q$ die Lösungen $x_2$ und $q-x_2$ eingesetzt werden können. Die Kombination der jeweils zwei Lösungen ergibt vier Gleichungssysteme, die jeweils genau eine Lösung in $[0, n-1]$ besitzen:

| | | | |
|---|---|---|---|
| $x \bmod p = x_1$<br>$x \bmod q = x_2$ | $x \bmod p = x_1$<br>$x \bmod q = q - x_2$ | $x \bmod p = p - x_1$<br>$x \bmod q = x_2$ | $x \bmod p = p - x_1$<br>$x \bmod q = q - x_2$ |
| ↓ | ↓ | ↓ | ↓ |
| Lösung $x$ | Lösung $y$ | Lösung $n-y$ | Lösung $n-x$ |

Da die Bezeichnung von $x_1$ mit $p-x_1$ sowie die von $x_2$ mit $q-x_2$ vertauscht werden kann, kann o.B.d.A. $x$ als Lösung des linken Systems angenommen werden. Dann ist offenbar $n-x$ wegen

$$\begin{aligned} (n-x) \bmod p &= (p-x) \bmod p = p - x_1, \\ (n-x) \bmod q &= (q-x) \bmod q = q - x_2 \end{aligned}$$

die Lösung des Systems ganz rechts. Entsprechende Überlegungen gelten für die beiden mittleren Gleichungssysteme. Alle Lösungen sind voneinander verschieden, da die rechten Seiten der obigen Gleichungssysteme paarweise verschieden sind, insbesondere gilt

$(**)$ $$x \neq y \text{ und } y \neq n - x.$$

Es folgt

$$\left(x^2 - y^2\right) \bmod n = 0$$

und daraus

$$n = pq | (x + y)(x - y).$$

Würde $n|(x + y)$ gelten, so erhielten wir wegen $x, y \in [0, n - 1]$ im Widerspruch zu $(**)$ die Gleichungen $x = y = 0$ oder $x + y = n$. Für $n|(x - y)$ ergäbe sich $x - y = 0$. Wir schließen, daß

$$p|(x + y), q|(x - y) \text{ oder } q|(x + y), p|(x - y)$$

gilt. Somit folgt

$$\text{ggt}(x + y, n) = p \text{ oder } q.$$

Falls Bob in (4) $x$ oder $n - x$ empfängt, dann kann er ohne die Kenntnis von $p$ und $q$ die andere Lösung $y$ von $a = x^2 \bmod n$ nicht berechnen. Es besteht jedoch die Möglichkeit, daß er zufällig $x$ so gewählt hat, daß $x$ ein Vielfaches von $p$ (oder $q$) ist. Dann kann $\text{ggt}(x, n) = p$ (oder $q$) berechnet werden und damit auch die Faktorisierung von $n$. Die Wahrscheinlichkeit einer solchen Wahl ist allerdings sehr gering.

Das Bestimmen der Lösungen der Gleichungen (a) und (b) ist besonders einfach, wenn $4|(p+1)$ und $4|(q+1)$ gilt. Nach (a) ist $a \bmod p$ ein quadratischer Rest modulo $p$. Aus Satz 5.1.4 folgt zunächst

$$a^{\frac{p-1}{2}} \bmod p = 1,$$

und damit erhalten wir

$$\left(a^{\frac{p+1}{4}}\right)^2 \equiv_p a^{\frac{p+1}{2}} \equiv_p a \cdot a^{\frac{p-1}{2}} \equiv_p a.$$

Somit ist $x_1 = a^{\frac{p+1}{4}} \bmod p$ eine Lösung von (a) und analog

$$x_2 = a^{\frac{q+1}{4}} \bmod q$$

eine Lösung von (b). Allgemein kann der folgende probabilistische Algorithmus (siehe [50]) verwendet werden, der im Mißerfolgsfall (Wahrscheinlichkeit $\frac{1}{2}$) wiederholt bis zur Berechnung des dann immer richtigen $x$ angewendet wird.

**Algorithmus 5.2.1**
{Es sei $p > 2$ eine Primzahl und $a \in R_2$, mit Wahrscheinlichkeit $\frac{1}{2}$
wird ein $x \in [1, p - 1]$ mit $x^2 \bmod p = a \bmod p$ geliefert}
**begin** wähle zufällig $b \in \mathbb{N}$ mit $1 \leq b < p$.
**if** $b^{\frac{p-1}{2}} \bmod p = p - 1$
**then** bestimme $l, m \in \mathbb{N}_0$, $m$ ungerade, mit $p - 1 = 2^l m$;
$y := a$;
$n := 1$;
suche das kleinste $k[n] \geq 0$ mit $y^{2^{k[n]}m} \bmod p = 1$;
**while** $k[n] \neq 0$

**do** $y := yb^{2^{l-k[n]}} \bmod p$;
$n := n + 1$;
suche das kleinste $k[n] \geq 0$ mit $y^{2^{k[n]}m} \bmod p = 1$;
**end**;
$r := y^{\frac{m+1}{2}} \bmod p$;
**for** $i := n$ **downto** 2
**do** $r := r(b^{2^{l-k[i-1]-1}})^{-1} \bmod p$;
**end**;
$x := r$;
**else** write(„kein Erfolg“)
**end** □

**Satz 5.2.1** Es sei $p > 2$ eine Primzahl und $a \in R_2$. Algorithmus 5.2.1 liefert mit Wahrscheinlichkeit $\frac{1}{2}$ ein $x \in [1, p-1]$ mit $x^2 \bmod p = a \bmod p$. Der Zeit- und Platzbedarf des Algorithmus ist polynomial in $\log_2 p$.

*Beweis.* Nach Satz 5.1.4 wird durch den ersten Test geprüft, ob $b$ ein quadratischer Nichtrest ist. Nach Satz 5.1.3 ist dies mit Wahrscheinlichkeit $\frac{1}{2}$ der Fall. Dann durchläuft die Variable $y$ eine Folge $a_1, a_2, \ldots$ quadratischer Reste modulo $p$, und die Werte der Variablen $k[n]$, $n = 1, 2 \ldots$, sind Zahlen mit den Bezeichnungen $k_1, k_2, \ldots$. Nach Algorithmus 5.2.1 gilt dabei

$$a_1 = a,$$

und für $n \geq 2$ erhalten wir

$$\begin{aligned} k_{n-1} &= \text{kleinstes } k \geq 0 \text{ mit } a_{n-1}^{2^k m} \bmod p = 1, \\ a_n &= a_{n-1} b^{2^{l-k_{n-1}}} \bmod p. \end{aligned}$$

Die Existenz von $k_1$ mit $k_1 \leq l$ ist wegen

$$a^{\varphi(p)} \bmod p = a^{p-1} \bmod p = a^{2^l m} \bmod p = 1$$

gesichert, bei bereits definiertem $a_n$ die von $k_n$ mit $k_n \leq k_{n-1} - 1 < k_{n-1}$, $n > 1$, durch

$$\begin{aligned} a_n^{2^{k_{n-1}-1}m} \bmod p &= (a_{n-1} b^{2^{l-k_{n-1}}})^{2^{k_{n-1}-1}m} \bmod p \\ &= a_{n-1}^{2^{k_{n-1}-1}m} b^{2^{l-1}m} \bmod p \\ &= a_{n-1}^{2^{k_{n-1}-1}m} b^{\frac{p-1}{2}} \bmod p \\ &= a_{n-1}^{\frac{2^{k_{n-1}}m}{2}} \cdot (-1) \bmod p \\ &= (-1) \cdot (-1) \bmod p \\ &= 1. \end{aligned}$$

Dabei gilt die vorletzte Gleichheit, weil $k_{n-1}$ die kleinste Zahl mit $a_{n-1}^{2^{k_{n-1}}m} \bmod p = 1$ ist und bei Rechnungen modulo $p$ außer 1 nur $-1$ quadriert 1 liefert.

Aufgrund der vorstehenden Überlegungen ist

$$l \geq k_1 > k_2 > \ldots \geq 0.$$

Folglich existiert ein $n' \in \mathbb{N}$, $n' \leq l$, mit $k_{n'} = 0$. Dann gilt nach der obigen Gleichung für $k_{n-1}$ mit $n - 1 = n'$

$$a_{n'}^{m \cdot 2^{k_{n'}}} \bmod p = a_{n'}^{m} \bmod p = 1.$$

Multiplikation mit $a_{n'}$ liefert

$$a_{n'}^{m+1} \bmod p = a_{n'} \bmod p, \text{ also } (a_{n'}^{\frac{m+1}{2}})^2 \bmod p = a_{n'} \bmod p.$$

Es ist somit $a_{n'}^{\frac{m+1}{2}}$ eine Quadratwurzel von $a_{n'}$.

Anschließend durchläuft die Variable $r$ eine Folge $r_{n'}, \ldots, r_1$ von quadratischen Wurzeln modulo $p$. Dabei gilt

$$r_{n'} = a_{n'}^{\frac{m+1}{2}} \bmod p \text{ und } r_i = r_{i+1}(b^{2^{l-k_i-1}})^{-1} \bmod p$$

für $i = n' - 1, \ldots, 1$. Zwischen den $r_i$ und $a_i$ bestehen die Beziehungen

$$r_i^2 \bmod p = a_i \bmod p \text{ für } i = n', \ldots, 1.$$

Dies beweisen wir durch Induktion. Für $i = n'$ ist die Aussage erfüllt. Es sei nun die Aussage für $i + 1$ mit $i < n'$ richtig. Dann folgt

$$\begin{aligned} r_i^2 \bmod p &= r_{i+1}^2((b^{2^{l-k_i-1}})^{-1})^2 \bmod p \\ &= a_{i+1}(b^{2^{l-k_i}})^{-1} \bmod p \\ &= a_i(b^{2^{l-k_i}})(b^{2^{l-k_i}})^{-1} \bmod p \\ &= a_i \bmod p. \end{aligned}$$

Insbesondere gilt $x^2 \bmod p = r_1^2 \bmod p = a_1 \bmod p = a \bmod p$.

Während des Algorithmus müssen sich $n' \leq l \leq \log_2 p$ Variablen $k_i$ gemerkt werden, so daß der Platzbedarf $O(\log_2 p)$ ist. Die beiden Schleifen werden $n'$-mal durchlaufen. Die Exponentiationen benötigen jeweils $O(\log_2 p)$ Multiplikationen von Zahlen der Länge $O(\log_2 p)$. Auch die Bildung des Inversen, die mit Hilfe des Algorithmus 3.1 (siehe Bemerkung auf Seite 51) durchgeführt werden kann, hat dieselbe Komplexität. Das Suchen eines kleinsten $k[n]$ erfordert höchstens $\log_2 p$ Überprüfungen der jeweiligen Komplexität $O(\log_2^2 p)$. Insgesamt ist der Zeitbedarf also durch $O(\log_2^4 p)$ gegeben. □

Endet Algorithmus 5.2.1 mit „kein Erfolg“, so wird er wiederholt. Mit großer Wahrscheinlichkeit hat man nach wenigen Versuchen ein $x$ mit $x^2 \bmod p = a \bmod p$ bestimmt.

**Beispiel 5.2.1** Es seien $p = 3$ und $q = 7$. Dann ergibt sich $n = pq = 21$. Bob wählt $x = 5$ und sendet $a = 5^2 \bmod 21 = 4$ an Alice. Wegen $4|p + 1$ und $4|q + 1$ berechnet Alice die Wurzeln

$$\begin{aligned} x_1 &= 4^{\frac{3+1}{4}} \bmod 3 = 1 \text{ bzw.} \\ x_2 &= 4^{\frac{7+1}{4}} \bmod 7 = 2. \end{aligned}$$

Mit dem Chinesischen Restesatz (Algorithmus 3.4) werden die vier gemeinsamen Lösungen berechnet:

$$z_1 = \text{crt}(n, p, q, x_1, x_2) = 16$$
$$z_2 = \text{crt}(n, p, q, x_1, q - x_2) = 19$$
$$z_3 = \text{crt}(n, p, q, p - x_1, x_2) = 2$$
$$z_4 = \text{crt}(n, p, q, p - x_1, q - x_2) = 5$$

Wegen der obigen Überlegungen gilt $z_4 = n - z_1$ bzw. $z_3 = n - z_2$, d. h. $z_4$ und $z_3$ können direkt ohne Benutzung des Chinesischen Restesatzes bestimmt werden. Falls Alice nun $y = 19$ an Bob sendet, kann er $p$ und $q$ durch ggt$(x + y, n)$ ermitteln, nämlich

$$\text{ggt}(x + y, n) = \text{ggt}(5 + 19, 21) = \text{ggt}(24, 21) = 3.$$

Damit berechnet er

$$q = \frac{n}{p} = \frac{21}{3} = 7. \quad \square$$

Das Oblivious-Transfer-Protokoll kann z.B. benutzt werden, um durch das Telephon Münzen zu „werfen“. Jeder der beiden Teilnehmer soll 50% Gewinnchancen haben. Die Münzen können selbstverständlich nicht unverschlüsselt geworfen werden, da sonst die Möglichkeit eines Betruges besteht. Wir erhalten folgendes *Protokoll zum Werfen von Münzen* [11].

**Protokoll 5.2.2**

(1) Alice wählt zwei große Primzahlen $p$ und $q$ und sendet $n = pq$ an Bob.
(2) Bob prüft, ob $n$ eine Primzahl oder gerade ist. Ist dies der Fall, so hat Alice betrogen. Bob wählt ein $x, 0 < x < n$, mit ggt$(x, n) = 1$ und sendet $a = x^2 \bmod n$ an Alice.
(3) Alice berechnet die vier Wurzeln von $a$, wählt zufällig eine aus und schickt sie an Bob.
(4) Bob gewinnt, wenn er $n$ faktorisieren kann. Dies ist möglich, wenn er $y$ oder $n - y$ empfängt.

Nach dem Spiel können $p$ und $q$ offengelegt werden. Damit kann dann überprüft werden, ob Alice betrogen hat. $\square$

Ein Protokoll, das ein allgemeineres Problem als das oben angegebene Oblivious-Transfer-Protokoll 5.2.1 behandelt, beruht auf einem Vorschlag von *Micali* und wurde in [21] vorgestellt. Die Eigenschaften sind wie folgt gegeben:

Alice sendet zwei Nachrichten an Bob, wobei die folgenden Bedingungen erfüllt sein sollen:

(1) Falls Alice das Protokoll richtig ausführt, dann kann Bob *genau eine* Nachricht lesen. Die Wahrscheinlichkeit für den Erhalt jeder dieser Nachrichten beträgt $\frac{1}{2}$. Falls Bob keine Nachricht lesen will, dann kann er bei der Ausführung des Protokolls auch keine Informationen über die Nachrichten herausfinden.
(2) Alice weiß jeweils mit Wahrscheinlichkeit $\frac{1}{2}$, daß Bob die erste (bzw. die zweite) Nachricht erhalten hat.
(3) Falls Alice versucht, diese Voraussagewahrscheinlichkeit über die von Bob erhaltene Nachricht zu erhöhen, dann kann Bob dieses mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2}$ feststellen.

Da eine von zwei Nachrichten empfangen wurde, spricht man auch von einem „1-von-2-Oblivious-Transfer-Protokoll“. Mit Hilfe eines solchen Protokolls kann auch das ursprüngliche Oblivious-Transfer-Protokoll simuliert werden. Dies ist sofort einzusehen. Eine der beiden von Alice gesendeten Nachrichten sei öffentlich, während die andere von Alice geheim gewählt wird. Diese geheime Nachricht empfängt Bob dann mit einer Wahrscheinlichkeit $\frac{1}{2}$. Wenn er stattdessen die öffentliche Nachricht empfängt, erfährt er nichts Neues.

Wir beschreiben das *1-von-2-Oblivious-Transfer-Protokoll*. Wir gehen davon aus, daß es auf dem RSA-Verfahren beruht (man kann auch ein anderes Public-Key-Kryptosystem wählen), wobei $n$ das Produkt $n = p \cdot q$ ist. Wie üblich sei $E_A$ die öffentliche und $D_A$ die geheime Transformation von Alice.

**Protokoll 5.2.3**
Eine von 2 Nachrichten $M_0$ und $M_1$ soll Bob von Alice durch Anwendung des Protokolls erhalten.

(1) Alice wählt zufällig zwei weitere Nachrichten $m_0$ und $m_1$ aus $[0, n-1]$ und sendet sie an Bob.

(2) Bob wählt zufällig ein $r \in \{0,1\}$ sowie eine zufällige Nachricht $k \in [0, n-1]$ und sendet
$$q = (E_A(k) + m_r) \bmod n$$
an Alice.

(3) Alice berechnet
$$k'_i = D_A((q - m_i) \bmod n) \text{ für } i = 0, 1.$$
Anschließend wählt sie zufällig ein $s \in \{0,1\}$ und übermittelt Bob
$$((M_0 + k'_s) \bmod n, (M_1 + k'_{s\oplus 1}) \bmod n, s).$$

(4) Bob berechnet $M_{s\oplus r} = ((M_{s\oplus r} + k'_{s\oplus r\oplus s}) \bmod n - k) \bmod n$. □

Wir zeigen, daß dieses Protokoll den obigen Eigenschaften genügt. Bob hat zufällig $r$ und $k$ gewählt und weiß, daß
$$k'_r = k'_{s\oplus r\oplus s} = D_A(((E_A(k) + m_r) \bmod n - m_r) \bmod n) = k$$
gelten muß. $k'_{r\oplus 1}$ kennt er nicht. Bob kann durch Abziehen von $k$ in (4) genau eine der beiden Nachrichten lesen, und zwar $M_{s\oplus r}$. Die andere Nachricht bleibt ihm aufgrund seiner Unkenntnis von $k'_{r\oplus 1}$ verborgen. Wegen der zufälligen Wahlen von $r$ und $s$ erhält er jede der beiden Nachrichten mit Wahrscheinlichkeit $\frac{1}{2}$. Damit ist die erste geforderte Eigenschaft des Protokolls erfüllt.

Wenn beide das Protokoll richtig ausführen, ist die einzige Information, die Alice von Bob erhält, $q = (E_A(k) + m_r) \bmod n$. Alice kennt zwar $k'_0$ und $k'_1$, sie weiß jedoch nicht, welcher dieser Werte gleich $k$ ist. Somit hat sie auch keine Information über $r$, was sonst in der ersten Berechnung von (3) feststellbar wäre. Ist nämlich $k'_i = k$, so muß $r = i$ sein. Also weiß sie nicht, welche der Nachrichten $M_{s\oplus r}$ ist. Außerdem muß sie auch $k'_0$ und $k'_1$ in (3) benutzen, da Bob sonst ggf. gar keine sinnvolle Nachricht erhält. Folglich ist auch die zweite geforderte Eigenschaft des Protokolls erfüllt.

Einen Betrug gemäß der dritten Forderung kann Alice nur erreichen, wenn sie für $M_0$ und $M_1$ dieselbe Nachricht wählt. Ein solcher Betrug ist jedoch mit Wahrscheinlichkeit 1

aufzudecken. Da Bob die Größen $r$, $q$, $M_{s\oplus r}$ sowie $(M_{s\oplus r\oplus 1} + k'_{s\oplus r\oplus 1\oplus s}) \bmod n$ bekannt sind, kann er immer

$$\bar{k}_{r\oplus 1} = ((M_{s\oplus r\oplus 1} + k'_{s\oplus r\oplus 1\oplus s}) \bmod n - M_{s\oplus r}) \bmod n$$

berechnen. Damit gilt

$$M_0 = M_1 \Longleftrightarrow M_{s\oplus r} = M_{s\oplus r\oplus 1} \Longleftrightarrow \bar{k}_{r\oplus 1} = k'_{r\oplus 1}.$$

Wegen der Äquivalenz von $k'_{r\oplus 1} = D_A((q-m_{r\oplus 1}) \bmod n)$ mit $E_A(k'_{r\oplus 1}) = (q-m_{r\oplus 1}) \bmod n$ folgt dann

$$M_0 = M_1 \Longleftrightarrow E_A(\bar{k}_{r\oplus 1}) = (q - m_{r\oplus 1}) \bmod n,$$

so daß Bob durch Nachrechnen der rechten Gleichung sofort einen Betrugsversuch feststellen kann.

Als Beispiel für die Anwendung von Oblivious-Transfer-Protokollen wollen wir nun den Austausch von Geheimnissen behandeln. Hierdurch kann beispielsweise der Austausch von geheimen Schlüsseln realisiert werden, die bei der Benutzung von Kryptosystemen benötigt werden.

Es werde vorausgesetzt, daß alle Geheimnisse in binär kodierter Form vorliegen. Die beteiligten Parteien seien, wie üblich, mit Alice und Bob benannt. Alice besitze $n$ Paare von Geheimnissen der Länge $m$ Bits, die mit

$$(a_{1,1}, a_{1,2}), (a_{2,1}, a_{2,2}), \ldots, (a_{n,1}, a_{n,2})$$

bezeichnet sind. Bob kennt genau ein Geheimnis von jedem dieser Paare, wobei Alice nicht weiß, um welches Geheimnis es sich handelt. Diese Bedingung kann am besten durch ein 1-von-2-Oblivious-Transfer-Protokoll erreicht werden. Dieses garantiert, daß Bob genau eines von zwei Geheimnissen erfährt. Analog besitze Bob $n$ Paare von Geheimnissen der Länge $m$ Bits

$$(b_{1,1}, b_{1,2}), (b_{2,1}, b_{2,2}), \ldots, (b_{n,1}, b_{n,2}),$$

von denen Alice auch jeweils nur eines kennt. Beide wollen nun das ihnen jeweils unbekannte Geheimnis eines jeden Paares herausfinden. Dabei wird vorausgesetzt, daß das „Berechnen“ eines Geheimnisses nur durch eine vollständige Suche in der Menge aller Wörter von $m$ Bits möglich ist.

In [21] findet sich das folgende Protokoll, mit dessen Hilfe die gleichzeitige Kenntnis eines Paares von Geheimnissen erreichbar ist.

**Protokoll 5.2.4**

**for** $k := 1$ **to** $m$ **do**
Alice sendet das $k$-te Bit von jedem $a_{i,j}, i = 1, \ldots, n, j = 1, 2$;
Bob sendet das $k$-te Bit von jedem $b_{i,j}, i = 1, \ldots, n, j = 1, 2$
**end**. □

Eine unehrliche Partei, die verhindern möchte, daß die Gegenseite irgendein Geheimnis erhält, muß unkorrekte Bits für mindestens ein Element eines jeden Paares senden. Die Wahrscheinlichkeit, dies mit Erfolg durchzuführen, ist allerdings nur $\frac{1}{2^n}$, da die Gegenseite

ja ein Element eines jeden Paares kennt. Um sich vor Betrug zu schützen, sollte also jede Partei überprüfen, ob die Bits der Geheimnisse, die sie bereits durch Anwendung des Oblivious-Transfer-Protokolls kennt, mit den jetzt gesendeten Werten übereinstimmen. Sie sollte sofort die weitere Ausführung des Protokolls beenden, sobald sie einen versuchten Betrug der anderen Partei entdeckt. Falls beide Parteien den Regeln folgen und keine die andere durch Senden von falschen Werten betrügt, kennen beide nach Ablauf des Protokolls alle Geheimnis-Paare der anderen Partei.

Protokoll 5.2.4 hat allerdings den Nachteil, daß es nicht gerecht ist. Falls nämlich Bob die Durchführung des Protokolls abbricht, nachdem Alice ihm die k-ten Bits gesendet hat, besitzt er einen 2-zu-1-Berechnungsvorteil. Um ein Paar „berechnen“ zu können, muß er noch eine Teilmenge von $2^{m-k}$ möglichen Geheimnissen untersuchen, während Alice dagegen eine Teilmenge der Größe $2^{m+1-k}$ zu betrachten hat. Diese Menge ist aber gerade doppelt so groß wie Bobs Menge. Im folgenden wird eine Methode von *Tedrick* [100] vorgestellt, bei der Bobs Berechnungsvorteil zu jedem Zeitpunkt durch einen vorgegebenen Wert beschränkt ist. Dieses Verfahren wird zunächst anhand eines Beispiels erläutert.

**Beispiel 5.2.2** Bob und Alice einigen sich zuerst darauf, daß Bobs Berechnungsvorteil nur noch 5 zu 4 beträgt und nicht mehr 2 zu 1 wie im obigen Protokoll 5.2.4. Dann speichert Alice für jedes $a_{i,j}$ die Folge der binären Wörter (in aufsteigender Reihenfolge bezüglich des Zahlenwertes)

$$000 \quad 001 \quad \underline{010} \quad 011 \quad 100 \quad 101 \quad 110 \quad 111.$$

Genau eines dieser Wörter stimmt exakt mit den ersten drei Bits von $a_{i,j}$ überein. Dieses Wort sei beispielsweise 010, das bereits durch Unterstreichung gekennzeichnet ist. Bob verfährt ebenso. Anschließend wird eine Reihe von Nachrichten ausgetauscht. Für jedes $a_{i,j}$ sendet Alice die Nachricht

> „Die ersten drei Bits von $a_{i,j}$ werden nicht durch die $y$-te Bitfolge $(1 \leq y \leq 8)$ dargestellt.“

Bob antwortet analog. Wir stellen fest, daß Bob, nachdem Alice ihre Mitteilungen an Bob gesendet hat, zunächst einen 8-zu-7-Vorteil besitzt. Das bedeutet, daß Alice, falls sie betrügen will, das Geheimnis unter 8 Wörtern suchen muß, Bob dagegen nur unter 7 Wörtern. Nach dem nächsten Durchgang hat Bob einen 7-zu-6-, dann einen 6-zu-5- und schließlich einen 5-zu-4-Vorteil. Wenn Bob dann auch seine Nachricht geschickt hat, stehen nur noch die Hälfte der Wörter zur Verfügung, und der Nachrichtenaustausch wird zunächst unterbrochen. Wir nehmen zum Beispiel an, daß nur noch die folgenden vier Wörter zur Auswahl stehen:

$$001 \quad \underline{010} \quad 110 \quad 111.$$

Aus diesen werden acht neue Wörter erzeugt, indem an jedes von ihnen 0 und 1 angehängt wird:

$$0010 \quad 0011 \quad 0100 \quad \underline{0101} \quad 1100 \quad 1101 \quad 1110 \quad 1111.$$

Genau eines von ihnen stimmt mit den ersten vier Bits von $a_{i,j}$ überein. Der obige Nachrichtenaustausch wird wiederholt (wobei „Die ersten drei . . .“ durch „Die ersten vier . . .“ ersetzt wird), bis für jedes $a_{i,j}$ wieder nur noch vier Wörter zur Auswahl stehen. Aus diesen werden wiederum acht neue Wörter gebildet, und das Verfahren wird wiederholt.

Bis einschließlich zur Wiederholung des Verfahrens mit Wörtern der Länge $m$ Bits hat Bob immer nur einen maximalen Vorteil von 5 zu 4, so wie es vorher vereinbart wurde.

Dann stehen noch vier Wörter zur Auswahl. Eine Berechnung des jeweiligen Geheimnisses ist für beide jetzt nicht mehr schwer. So können zum Abschluß noch ohne Bedenken wechselseitig die jeweils drei falschen Wörter ausgetauscht werden, obwohl sich dabei der Berechnungsvorteil zugunsten von Bob erhöht. □

In dem nachstehenden Protokoll wird das in dem Beispiel beschriebene Verfahren systematisiert.

**Protokoll 5.2.5**

(1) Zunächst wird ein für beide Teilnehmer akzeptabler Wert $k$ so festgelegt, daß der maximale Berechnungsvorteil $2^k + 1$ zu $2^k$ beträgt.

(2) Für jedes $a_{i,j}$ und $b_{i,j}$, $i = 1, \ldots, n$, $j = 1, 2$, werden die der Größe nach geordneten Folgen aller $2^{k+1}$ Binärwörter der Länge $k + 1$ abgespeichert.

(3) **while** noch nicht alle $m$ Bits gesendet wurden
**do** $M_{A,i,j} := M_{B,i,j} := \emptyset$;
**for** $x := 1$ **to** $2^k$
**do** Alice sendet für jedes $a_{i,j}$ ein $y \in \mathbb{N}$, $y \notin M_{A,i,j}$, $1 \leq y \leq 2^{k+1}$,
so daß das $y$-te Wort der Folge kein Präfix von $a_{i,j}$ ist,
und bildet $M_{A,i,j} := M_{A,i,j} \cup \{y\}$;
Bob sendet für jedes $b_{i,j}$ ein $y' \in \mathbb{N}$, $y' \notin M_{B,i,j}$, $1 \leq y' \leq 2^{k+1}$,
so daß das $y'$-te Wort der Folge kein Präfix von $b_{i,j}$ ist,
und bildet $M_{B,i,j} := M_{B,i,j} \cup \{y'\}$;
**end**;
Alice und Bob bilden $2^{k+1}$ Wörter für jedes $a_{i,j}$ und $b_{i,j}$,
indem sie 0 und 1 an die nicht gesendeten Wörter anhängen
**end**. □

Jede Partei muß in jedem Schritt des Protokolls 5.2.5 $n \cdot 2 \cdot 2^{k+1}$ Wörter von höchstens $m$ Bits Länge speichern. Folglich ist der Platzbedarf des Protokolls $O(n \cdot m \cdot 2^k)$. Da nicht mehr als $m$ Durchgänge durchgeführt werden, ist der Zeitbedarf $O(n \cdot m^2 \cdot 2^k)$.

Im folgenden wollen wir die Anwendung der Protokolle 5.2.3 und 5.2.4 bzw. 5.2.5 für die Signierung von Verträgen sowie für die Simulierung des Sendens von „Einschreiben“ betrachten.

Für Geschäftsbeziehungen ist es von großer Bedeutung, daß Verträge und Abkommen durch eine Unterschrift bestätigt werden. Mit der zunehmenden Verbreitung von Computernetzen ist man daran interessiert, daß solche Aufgaben auch auf elektronischem Weg erledigt werden können. Für dieses Problem werden wir ein sicheres Protokoll angeben, das in [21] beschrieben ist.

Ein solches Protokoll für die Unterzeichnung eines Vertrages sollte die folgenden Bedingungen erfüllen:

(1) Nach Beendigung der richtigen Ausführung des Protokolls muß jede Partei von dem Geschäftspartner eine Unterschrift unter dem Vertrag haben.

(2) Falls die Partei $X$ das Protokoll richtig ausführt, dann kann die Gegenpartei $Y$ die Unterschrift von $X$ nicht erhalten, wenn sie selbst nicht den Vertrag unterzeichnet.

Einfache Protokolle für die Unterzeichnung von Verträgen, die diese Bedingungen erfüllen, lassen sich schnell konzipieren, wenn eine dritte, vertrauenswürdige Partei, z.B. ein Notar, vorhanden ist. Die Geschäftspartner brauchen dann nur ihre unterschriebenen Verträge für die jeweils andere Partei an diese dritte Person zu senden, die die Verträge erst dann entsprechend weiterleiten wird, wenn sie beide Verträge erhalten hat. Erhält sie dagegen von einem Teilnehmer keinen unterschriebenen Vertrag in einem gewissen Zeitraum, bekommt der andere seinen bereits unterzeichneten Vertrag zurück.

Im folgenden soll nun aber ein Protokoll vorgestellt werden, das keine dritte Partei benötigt und trotzdem die beiden Bedingungen gewährleistet. Es sei $C$ ein Vertrag, den Alice und Bob abschließen wollen. Sie haben den Vertrag bereits ausgehandelt und stehen vor seiner Unterzeichnung. Das unten angegebene Protokoll 5.2.6 garantiert ihnen den Austausch der Unterschriften zu diesem Vertrag. Das Wesentliche des Protokolls besteht darin, daß Alice zufällig eine Menge von Geheimnis-Paaren wählt und erklärt, daß sie selbst den Vertrag als von ihr unterzeichnet anerkennt, falls Bob eines ihrer Geheimnispaare kennt. Eine entsprechende Aussage gilt auch umgekehrt für Bob. Die Menge der Geheimnis-Paare wird dann beispielsweise durch Anwendung eines der vorstehenden Protokolle 5.2.4 oder 5.2.5 ausgetauscht, je nach Wahl des Berechnungsvorteils.

Vorausgesetzt wird in diesem Protokoll ein Public-Key-Signatur-Schema, wie es beispielsweise das RSA-Schema ist. Außerdem wird ein weiteres, *sicheres* Kryptosystem $F$ benötigt. $F$ ist sicher, wenn es in $F$ unmöglich ist, eine Nachricht $M$ sowie Schlüssel $K_1$ und $K_2$ so zu finden, daß $E_{K_1}(M) = E_{K_2}(M)$ gilt. Dies Kryptosystem $F$ kann z.B. das Pohlig-Hellman-Schema sein, wenn die Nachrichten 0 und 1 ausgeschlossen werden. Ist die Primzahl $p$ veröffentlicht, so können wir den Schlüssel durch $e$ angeben, $d$ kann dann leicht berechnet werden (siehe Seite 61). Allgemein sei mit einem Schlüssel $K$ auch $E_K$ bekannt. Wir sagen, der Schlüssel $K$ ist die Lösung des $M$-Puzzles $C$, falls $C = E_K(M)$ gilt. Weiterhin seien die Rechnerkapazitäten beider Parteien ungefähr gleich.

Wir erhalten nun das folgende *Vertragsunterzeichnungsprotokoll*:

**Protokoll 5.2.6**

Es sei $M$ eine beliebige zuvor verabredete Nachricht. Außerdem werde eine beliebige Zahl $n \in \mathbb{N}$ gemeinsam festgelegt.

(1) Alice wählt zufällig $n$ Paare von Schlüsseln $(A_{1,1}, A_{1,2}) \ldots, (A_{n,1}, A_{n,2})$ und berechnet

$$C_{A_{i,j}} = E_{A_{i,j}}(M), 1 \leq i \leq n, j = 1, 2.$$

Anschließend gibt sie die folgende Erklärung ab:
„Die (formalen) Symbole $K^A_{i,j}$ sind Lösungen der zugehörigen $M$-Puzzles $C_{A_{i,j}}$, $1 \leq i \leq n$, $j = 1, 2$. Der Vertrag $C$ ist von mir unterzeichnet,wenn Bob für ein $i, 1 \leq i \leq n$, beide Symbole $K^A_{i,1}$ und $K^A_{i,2}$ nennen kann, d. h., wenn er die Lösung des $(i, 1)$-ten und $(i, 2)$-ten Puzzles kennt.“
Alice unterzeichnet diese Erklärung und schickt sie an Bob.

Bob verfährt analog. Er erzeugt Paare $(B_{1,1}, B_{1,2}), \ldots, (B_{n,1}, B_{n,2})$ von Schlüsseln und bezeichnet die Lösung des $M$-Puzzles $E_{B_{i,j}}(M)$ als $K^B_{i,j}$, $1 \leq i \leq n$, $j = 1, 2$.

(Zu diesem Zeitpunkt besitzen also beide Parteien eine unterzeichnete Erklärung der Gegenpartei, in der genau angegeben wird, wie deren Unterschrift zu dem Vertrag $C$ zustande kommt. Wir wissen, daß kein Gegner eine Unterschrift für diese Erklärungen fälschen kann, wenn ein Public-Key-Kryptosystem verwendet wird.)

(2) Durch Anwendung des 1-von-2-Oblivious-Transfer-Protokolls 5.2.3 erhält jede Partei zunächst von jedem Geheimnis-Paar

$$(A_{i,1}, A_{i,2}) \text{ bzw. } (B_{i,1}, B_{i,2}) \text{ für } i = 1, \ldots, n$$

genau ein Geheimnis, wobei der Sender nicht weiß, welches bekannt wird.

(3) Anschließend wird eins der Protokolle 5.2.4 oder 5.2.5 für den geheimen Nachrichtenaustausch angewendet, um auch die jeweils noch fehlenden Geheimnisse zu übermitteln. □

Wir wollen keinen exakten Beweis führen, daß dieses Protokoll auch bei versuchtem Betrug das angegebene Problem zufriedenstellend lösen kann. Wir wollen jedoch einige Plausibilitätsbetrachtungen dafür anstellen. Wir haben angenommen, daß Alice und Bob nur gleichartige Rechnerkapazitäten zur Verfügung haben. Auch sei die Zeit beschränkt, um die Lösung des $(i, 1)$-ten und $(i, 2)$-ten Puzzles zu nennen.

Nach Anwendung des 1-von-2-Oblivious-Transfer-Protokolls kann zum Beispiel Bob $E_{A_{i,j}}(M) = C^A_{i,j}$ mit den ihm übersandten Schlüsseln überprüfen. Mit Wahrscheinlichkeit $1 - \frac{1}{2^n}$ (man wähle $n$ genügend groß) entdeckt er einen Betrugsversuch. In diesem Fall bricht er das Protokoll ab. Die Berechnung der fehlenden Schlüssel ist dann für Alice, aber auch für Bob, praktisch undurchführbar. Beim anschließenden Geheimnisaustausch wird ein Betrug immer entdeckt. Auch dann wird das Protokoll abgebrochen, und beide Teilnehmer können nun versuchen, durch Testen der fehlenden Bits das Geheimnis zu ergründen. Weiter untersucht werden müssen nur die Paare, bei denen vor Abbruch kein Betrug entdeckt wurde. Natürlich können darunter auch gefälschte Werte sein. Ein Betrüger darf aber nicht zu früh falsche Werte schicken, da sonst mit großer Wahrscheinlichkeit zu früh abgebrochen würde und dann die Schlüssel nicht berechnet werden könnten. Somit hat auch die andere Partei später gute Chancen, mindestens einen fehlenden Schlüssel zu berechnen, d.h., für mindestens ein $i$ die Lösung des $(i, 1)$-ten und $(i, 2)$-ten Puzzles anzugeben.

Nun betrachten wir noch die elektronische Versendung von Einschreiben. Alice möchte Bob die Nachricht $M$ als Einschreiben schicken. Sie möchte eine Bestätigung haben, daß die Post bei Bob angekommen ist. Außerdem soll Bob den Inhalt von $M$ nur genau dann lesen können, wenn Alice diese Empfangsbescheinigung erhalten hat. Ein Protokoll aus [21] erfüllt diese Eigenschaften. Dabei sendet Alice zunächst eine Chiffrierung der Post an Bob, und Bob bestätigt anschließend diese chiffrierte Mitteilung. Zeitgleich mit der Bestätigung erhält er den Schlüssel $K$, der für die Chiffrierung verwendet wurde. Auch hier wird deshalb jede Partei zufällig eine Menge von Geheimnissen erzeugen. Die Kenntnis eines Geheimnis-Paares von Alice liefert den Schlüssel $K$, während das Wissen eines von Bobs Geheimnis-Paaren einen Teil der Bestätigung darstellt. Der andere Teil für die Bestätigung besteht darin, daß Bob nachprüfen kann, ob Alice ihre Geheimnis-Paare so gewählt hat, daß diese alle den Schlüssel spezifizieren.

Auch hier wird wieder ein sicheres Kryptosystem $F$ vorausgesetzt. Es ist also in $F$ unmöglich, eine Nachricht $M$ sowie Schlüssel $K_1$ und $K_2$ so zu finden, daß $E_{K_1}(M) = E_{K_2}(M)$ gilt.

**Protokoll 5.2.7**
Es sei $S$ eine beliebige zuvor verabredete Nachricht. Außerdem werde $n \in \mathbb{N}$ entsprechend der gewünschten Sicherheit gewählt.

(1) Alice erzeugt zufällig $n+1$ Schlüssel

$$(A_0, A_{1,1}, A_{2,1}, \ldots, A_{n,1})$$

für $F$ und berechnet $A_{i,2} = A_0 \oplus A_{i,1}$, $1 \leq i \leq n$, wobei $\oplus$ die bitweise Addition modulo 2 bezeichnet. Anschließend berechnet sie

$$C_{A_{i,j}} = E_{A_{i,j}}(S) \text{ und } C = E_{A_0}(M),\ 1 \leq i \leq n, j = 1, 2.$$

Alice übermittelt Bob durch ein sicheres Signaturschema die Nachrichten $C_{A_{i,j}}$, $1 \leq i \leq n$, $j = 1, 2$, sowie $C$.

(Zu diesem Zeitpunkt besitzt Bob also mit $C$ eine Verschlüsselung der Post.)

(2) Bob wählt zufällig $n$ Paare von Schlüsseln

$$(B_{1,1}, B_{1,2}), \ldots, (B_{n,1}, B_{n,2})$$

für $F$ und berechnet $C_{B_{i,j}} = E_{B_{i,j}}(S)$, $1 \leq i \leq n$, $j = 1, 2$. Bob übermittelt Alice durch ein sicheres Signaturschema die Nachrichten $C_{B_{i,j}}$, $1 \leq i \leq n$, $j = 1, 2$.
Bob erklärt:
„Die (formalen) Symbole $K_{A_{i,j}}$ und $K_{B_{i,j}}$ sind Lösungen der zugehörigen S-Puzzles $C_{A_{i,j}}$ bzw. $C_{B_{i,j}}$, $1 \leq i \leq n$, $j = 1, 2$. Das Symbol $K_{A_0}$ bezeichnet den Schlüssel zur Chiffrierung und Dechiffrierung von $M$ und muß die folgende Bedingung (b) erfüllen. Ich bestätige den Empfang der Nachricht dadurch, daß $C$ durch die Anwendung von $K_{A_0}$ entschlüsselt wird und daß Alice zum Nachweis meiner Bestätigung folgende Bedingungen erfüllt:

(a) Alice kann $(B_{i,1}, B_{i,2})$ vorweisen für ein $i$, $1 \leq i \leq n$.
(b) Alice kann $K_{A_{i,j}}$, $1 \leq i \leq n$, $j = 1, 2$, vorweisen, so daß $K_{A_0} = K_{A_{i,1}} \oplus K_{A_{i,2}}$ für jedes $i$, $1 \leq i \leq n$, gilt."

Bob unterzeichnet diese Erklärung und schickt sie Alice.

(Zu diesem Zeitpunkt besitzt Alice also eine von Bob unterschriebene Erklärung, die eindeutig festlegt, wie Bob den Empfang der Post bestätigt. Diese Erklärung kann nicht gefälscht werden, da ein sicheres Signatur-Schema vorausgesetzt worden ist.)

(3) Durch Anwendung des 1-von-2-Oblivious-Transfer-Protokolls 5.2.3 erhält jede Partei zunächst von jedem Geheimnis-Paar

$$(A_{i,1}, A_{i,2}) \text{ bzw. } (B_{i,1}, B_{i,2}) \text{ für } i = 1, \ldots, n$$

genau ein Geheimnis, wobei der Sender nicht weiß, welches bekannt wird.

(4) Anschließend wird Protokoll 5.2.4 bzw. Protokoll 5.2.5 für den (fairen) Nachrichtenaustausch angewendet, um die noch jeweils fehlenden Geheimnisse eines jeden Paares auszutauschen. □

Nach Anwendung des 1-von-2-Oblivious-Transfer-Protokolls in Schritt (3) können beide Parteien durch Test von $E_{A_{i,j}}(S) = C_{A_{i,j}}$ bzw. $E_{B_{i,j}}(S) = C_{B_{i,j}}$ für alle $i$, $1 \leq i \leq n$, und jeweils ein $j$, $j = 1, 2$, mit Wahrscheinlichkeit $1 - \frac{1}{2^n}$ entdecken, ob schon in diesem Schritt ein Betrug versucht wurde. Bob ist nach Ausführung von Schritt (3) noch nicht in der Lage, $A_0$ zu bestimmen. Erst wenn er beide Teile eines Geheimnis-Paares $(A_{i,1}, A_{i,2})$ kennt, kann er $A_0$ einfach bestimmen. Über die Betrugsmöglichkeiten bzw. deren Entdeckung in Schritt

(4) haben wir oben unter Protokoll 5.2.6 gesprochen. Wir können davon ausgehen, daß nach Ausführung von (4) sowohl Alice als auch Bob jeweils mindestens ein Paar $(B_{i,1}, B_{i,2})$ bzw. $(A_{i',1}, A_{i',2})$ kennen. Dadurch ist Bob in der Lage, $A_0$ durch $A_0 = A_{i',1} \oplus A_{i',2}$ zu berechnen und damit $C$ zu entschlüsseln. Da Alice $(B_{i,1}, B_{i,2})$ für mindestens ein $i$ empfangen hat, ist sie sicher, daß auch Bob ein entsprechendes Paar erhalten hat und damit die Nachricht $M$ lesen konnte. Hätte Alice jedoch $A_0$ unabhängig von den $(A_{i,1}, A_{i,2})$ bestimmt, so hätte Bob $A_0$ nicht berechnen können. Zu einer gültigen Bestätigung gehört deshalb also auch, daß Alice die Bedingung (b) erfüllt. Dann kann $C_{A_{i,j}} = E_{A_{i,j}}(S)$ mit den authentisch in Schritt (1) übermittelten $C_{A_{i,j}}$ verglichen werden. Sind diese Werte in Ordnung, dann hätte Bob einen Betrugsversuch von Alice mit großer Wahrscheinlichkeit (s.o.) schon vorher entdeckt.

## 5.3 Schlüsselverwaltung

Bei der Verwendung eines Kryptosystems, auch bei einem Public-Key-Kryptosystem, stellt sich prinzipiell die Frage nach der Authenzität der Personen, die gewisse Schlüssel oder Schlüsselpaare verwenden. Wieso kann ein Benutzer sicher sein, daß z.B. das Paar $(E_A, D_A)$ im RSA-Verfahren wirklich zum Benutzer $A$ gehört? Deswegen müssen die Schlüssel im Netzwerk durch entsprechende vertrauenswürdige Instanzen so verwaltet werden, daß diese Authentizität gewährleistet ist.

Bevor wir auf entsprechende Protokolle zur Schlüsselverteilung zu sprechen kommen, wollen wir zunächst das öffentliche Schlüsselaustauschsystem von *Diffie und Hellman* angeben (siehe [18]), bei dem der Schlüsselaustausch durch Exponentiation erfolgt. Die Authentizität der Teilnehmer, d.h., ob sie die Personen sind, die sie behaupten, ist dabei nicht gewährleistet.

Um dieses System verstehen zu können, müssen wir den Begriff einer primitiven Wurzel einführen. Es sei $p$ eine Primzahl und

$$\mathbb{Z}_p^* = \mathbb{Z}_p - \{0\} = \{1, \ldots, p-1\}.$$

Für $a \in \mathbb{Z}_p^*$ sei $m \in \mathbb{N}$ eine Zahl mit $a^m \bmod p = 1$ sowie $a^r \bmod p \neq 1$ für jedes $r \in \mathbb{N}$, $0 < r < m$. Dann sagen wir, daß $a$ *modulo* $p$ *zum Exponenten* $m$ *gehört*. Die Existenz von $m$ ist durch Satz 3.8 (Satz von *Fermat*) gesichert, der $m < p$ liefert. Wir definieren

$$Z_p(a) = \{a^x \bmod p \mid 0 \leq x < m\},$$

wobei $a^0 \bmod p = 1$ gesetzt wird. Wir erhalten

**Satz 5.3.1** Es sei $p$ eine Primzahl, $a \in \mathbb{Z}_p^*$, und $a$ gehöre modulo $p$ zum Exponenten $m$. Dann gilt:
(1) $|Z_p(a)| = m$.
(2) $Z_p(a)$ ist eine zyklische Untergruppe von $\mathbb{Z}_p^*$.
(3) $m \mid (p-1)$.

*Beweis.* Zum Beweis von (1) nehmen wir an, daß Zahlen $k, j \in \mathbb{N}_0$, $0 \leq k < j < m$, existieren mit $a^k \bmod p = a^j \bmod p$. Es folgt $a^{j-k} \bmod p = 1$. Dies ist ein Widerspruch zu der Voraussetzung, daß $a$ modulo $p$ zum Exponenten $m$ gehört. Also gilt $|Z_p(a)| = m$. Wir betrachten weiter zwei beliebige Elemente $a^x \bmod p, a^y \bmod p \in Z_p(a)$. Dann kann

$x, y \in \mathbb{N}_0$ mit $0 \leq x, y < m$ angenommen werden. Ihr Produkt ist $a^{x+y} \bmod p$. Für $x+y < m$ liegt es offensichtlich in $Z_p(a)$, anderenfalls gilt mit $r = (x+y) - m < m$ wegen $a^m \bmod p = 1$ die Beziehung $a^{x+y} \bmod p = a^r \bmod p \in Z_p(a)$. $Z_p(a)$ ist also unter Multiplikation abgeschlossen und ist somit eine Untergruppe von $\mathbb{Z}_p^*$ (die von $a$ erzeugte zyklische Untergruppe von $\mathbb{Z}_p^*$). Dies beweist (2). Schließlich ist aus der Gruppentheorie bekannt (siehe z.B. [41], S. 36, Satz 3), daß die Ordnung (die Anzahl der Elemente) einer Untergruppe die Ordnung der ganzen Gruppe teilt. Die Ordnung von $Z_p(a)$ ist $m$, die von $\mathbb{Z}_p^*$ ist $p-1$. □

**Definition 5.3.1** Es sei $p$ eine Primzahl. $g \in \mathbb{Z}_p^*$ gehöre modulo $p$ zum Exponenten $p-1$. Dann heißt $g$ primitive Wurzel modulo $p$ . □

**Satz 5.3.2** Es sei $p$ eine Primzahl und $g$ eine primitive Wurzel modulo $p$. Dann gilt $Z_p(g) = \mathbb{Z}_p^*$ (d.h., jedes $y \in \mathbb{N}$, $y \leq p-1$, besitzt eine eindeutige Darstellung $g^x \bmod p$ für ein $x \in \mathbb{N}_0$, $x \leq p-2$.).

*Beweis.* Nach Satz 5.3.1 gilt $|Z_p(g)| = p-1 = |\mathbb{Z}_p^*|$, und $Z_p(g)$ ist eine Untergruppe von $\mathbb{Z}_p^*$. Es folgt $Z_p(g) = \mathbb{Z}_p^*$. □

Eine primitive Wurzel $g$ modulo $p$ erzeugt die ganze Gruppe $\mathbb{Z}_p^*$.

**Satz 5.3.3** Es sei $p > 2$ eine Primzahl, und $p-1$ besitze $k$ verschiedene Primfaktoren, die bekannt sind. Dann gilt:

(1) Die Anzahl der primitiven Wurzeln modulo $p$ ist $\varphi(p-1)$ ($\varphi$ Eulersche Funktion, siehe Definition 3.5).

(2) Der Test, ob ein gegebenes $a \in \mathbb{N}$, $a \leq p-1$, eine primitive Wurzel modulo $p$ ist, kann in $O(k \cdot \log p) = O((\log p)^2)$ Schritten durchgeführt werden.

*Beweis.* Zum Beweis von (1) benutzen wir, daß wegen $p > 2$ nach Satz 2, S. 221, aus [41] $\mathbb{Z}_p^*$ zyklisch ist. Das bedeutet, daß ein $a \in \mathbb{Z}_p^*$ mit $\mathbb{Z}_p^* = \{a^0, a^1, \ldots, a^{p-2}\}$ existiert. Es ist also $a$ eine primitive Wurzel modulo $p$. Folglich gilt $a^x \bmod p = 1$ für genau die $x \in \mathbb{N}_0$, die sich als $x = (p-1) \cdot y$ mit einem $y \in \mathbb{N}_0$ darstellen lassen. Für jedes $n$, $n = 0, 1, \ldots, p-2$, ist $b = a^n \bmod p$ nach Definition 5.3.1 genau dann keine primitive Wurzel modulo $p$, wenn ein $m$, $m = 1, \ldots, p-2$, mit $b^m \bmod p = a^{nm} \bmod p = 1$ existiert. Da $a$ eine primitive Wurzel ist, ist dies nach der oben angegebenen Eigenschaft von $a$ äquivalent zu $(p-1) \mid n \cdot m$. Wegen $m < p-1$ muß dabei ein Primfaktor von $(p-1)$ auch in $n$ vorkommen. Aus $(p-1) \mid n \cdot m$ folgt somit ggt$(p-1, n) > 1$. Gilt umgekehrt $t =$ggt$(p-1, n) > 1$, so können wir $m = \frac{p-1}{t}$ wählen. Dann gilt $1 \leq m < p-1$ und $nm = \frac{n}{t} \cdot (p-1)$, also $(p-1) \mid nm$. Durch Negation dieser Äquivalenzen erhalten wir, daß für jedes $n$, $n = 0, 1, \ldots, p-2$, genau dann $b = a^n \bmod p$ eine primitive Wurzel modulo $p$ ist, wenn ggt$(p-1, n) = 1$ erfüllt ist. Es gibt genau $\varphi(p-1)$ Zahlen mit dieser Eigenschaft, was die Aussage (1) des Satzes beweist.

Wir betrachten (2). Wenn $a$ modulo $p$ zum Exponenten $m$ gehört, erhalten wir nach Satz 5.3.1 $m|(p-1)$. Dabei ist $a$ genau dann keine primitive Wurzel modulo $p$, wenn $m < p-1$ gilt. Das ist jedoch genau dann der Fall, wenn $m|\frac{p-1}{q}$ für einen Primfaktor $q$ von $p-1$ gilt,

also genau für $a^{\frac{p-1}{q}} \bmod p = 1$. Somit ist $a$ genau dann eine primitive Wurzel modulo $p$, wenn $a^{\frac{p-1}{q}} \bmod p \neq 1$ für jeden Primfaktor von $p-1$ gilt. Diese Bedingung kann nach Satz 3.3 für jeden der $k$ Faktoren mit $O(\log p)$ Multiplikationen überprüft werden. Offenbar ist $k = O(\log p)$, so daß sich insgesamt $O((\log p)^2)$ Schritte ergeben. □

Wenn speziell $p = 2q + 1$ gilt mit einer Primzahl $q$, dann ist es nicht schwer, für ein gegebenes $a$ den zugehörigen Exponenten $m$ modulo $p$ zu bestimmen. Es muß $m|2q$ gelten, also kommen nur $m = 2, q$ oder $2q$ in Frage. Außerdem ist es hier leicht, primitive Wurzeln modulo $p$ zu finden. Es gilt $\varphi(p-1) = \varphi(2) \cdot \varphi(q) = q - 1 = \frac{p-3}{2}$. Für große $p$ hat also ein zufällig gewähltes Element aus $\mathbb{Z}_p^*$ ungefähr die Wahrscheinlichkeit $\frac{1}{2}$, eine primitive Wurzel modulo $p$ zu sein.

**Satz 5.3.4** Es sei $p > 2$ eine Primzahl, $g$ eine primitive Wurzel modulo $p$ und $s = \frac{p-1}{2}$. Dann gilt:
(1) $g^s \bmod p = -1 \bmod p$.
(2) $g$ ist ein quadratischer Nichtrest modulo $p$.

*Beweis.* Wegen $s \in \mathbb{N}$ und $s < p - 1$ gilt $g^s \bmod p \neq 1$. Nach Satz 5.1.4 kommt dann nur der Wert $-1$ in Frage, womit nach Satz 5.1.4 auch die Aussage (2) erfüllt ist. □

Wir kommen nun zu dem öffentlichen Schlüsselaustauschsystem. Es wird angenommen, daß sich Alice und Bob auf eine gemeinsame große Primzahl $p$ und eine gemeinsame primitive Wurzel $g$ modulo $p$ verständigt haben. Sowohl $p$ und $g$ können öffentlich bekannt sein.

**Protokoll 5.3.1**
(1) Alice wählt zufällig eine Zahl $x_A \in \mathbb{Z}_p^*$ und berechnet

$$y_A = g^{x_A} \bmod p.$$

Sie hält $x_A$ geheim und sendet $y_A$ an Bob.
(2) Bob wählt zufällig eine Zahl $x_B \in \mathbb{Z}_p^*$ und berechnet

$$y_B = g^{x_B} \bmod p.$$

Er hält $x_B$ geheim und sendet $y_B$ an Alice.
(3) Alice berechnet

$$k_{AB} = y_B^{x_A} \bmod p = (g^{x_B} \bmod p)^{x_A} \bmod p = g^{x_B x_A} \bmod p.$$

Bob erhält denselben Wert durch

$$k_{AB} = y_A^{x_B} \bmod p = (g^{x_A} \bmod p)^{x_B} \bmod p = g^{x_A x_B} \bmod p.$$

(4) Alice und Bob benutzen $k_{AB}$ als einen geheimen Schlüssel für den weiteren Nachrichtenaustausch. □

Durch ein entsprechendes Herausgreifen von Bits können Alice und Bob z.B. $k_{AB}$ als Schlüssel im DES verwenden. Wie wir wissen, sind die Rechnungen aus Protokoll 5.3.1

schnell durchzuführen. Andererseits sind für die Berechnung von $k_{AB}$ die geheimen Werte von $x_A$ bzw. $x_B$ notwendig. Dazu muß man wohl, eine andere Möglichkeit ist jedenfalls nicht bekannt, die Logarithmen von $y_A$ bzw. $y_B$ modulo $p$ berechnen. Wir wissen, daß dieses Problem praktisch nicht zu lösen ist.

Um eine sichere Schlüsselverteilung in einem größeren Netzwerk zu erreichen, ist ein Schlüsselverteilungszentrum (key distribution center, KDC) erforderlich, das eine vertrauenswürdige Netzwerkresource sein muß. In Protokoll 5.3.2 geben wir zunächst ein Verfahren an, das kein Public-Key-Kryptosystem benutzt.

Das KDC teilt einen gemeinsamen Schlüssel $K_A = (E_A, D_A)$ mit jedem Benutzer $A$. Es benutzt $K_A$, um $A$ zusätzliche Schlüssel bei Bedarf zuzuteilen. Wenn ein Teilnehmer sicher mit einem anderen kommunizieren möchte, wendet er sich an das KDC, um einen Schlüssel für diese spezielle „Unterhaltung“ zu erhalten. Dies kann mit vielen unterschiedlichen Schlüsselverteilungsprotokollen erreicht werden. Die Unterschiede sind abhängig von den erlaubten Kosten der Nachrichten oder davon, ob mehrere gleichzeitige Verbindungen verfügbar sein sollen, ob die Teilnehmer synchronisierte Uhren haben oder ob das KDC auch berechtigt sein soll, Kommunikationen zu erlauben oder zu verbieten.

Unser Protokoll 5.3.2 ist typisch und benutzt eine wichtige Eigenschaft der kryptographischen Authentifizierung. Es benutzt das Konzept eines „Zertifikats“. Es handelt sich dabei um eine kryptographisch authentische Nachricht, die den Schlüssel für die Kommunikation enthält. Nur das vertrauenswürdige KDC konnte das Zertifikat erzeugen.

**Protokoll 5.3.2**

(1) Alice möchte einen sicheren Nachrichtenaustausch mit Bob eröffnen. Sie wendet sich an das KDC und bittet um einen Schlüssel $K$ zur Kommunikation mit Bob.

(2) Das KDC wählt einen Schlüssel $K = (E, D)$ aus und schickt Alice ein Paar von Zertifikaten, nämlich $(E_A(K), E_B(K))$. Die erste Komponente kann nur von Alice, die zweite nur von Bob entschlüsselt werden.

(3) Alice berechnet $K = D_A E_A(K)$ und sendet die andere Komponente $E_B(K)$ an Bob. Damit führt sie sich bei Bob (und nur bei Bob) ein. Bob berechnet $K = D_B E_B(K)$.

(4) Alice und Bob beginnen die sichere Kommunikation mit diesem Sitzungsschlüssel. □

Alice und Bob müssen nicht bei jeder Kommunikation das ganze Protokoll wiederholen, sondern können für späteren Gebrauch die Zertifikate aufheben. So muß nicht jedes Mal das KDC angerufen werden. Trotzdem ist die Anzahl der KDC-Anrufe proportional der Anzahl der verschiedenen Paare von Benutzern, die miteinander kommunizieren wollen. Außerdem sollten die Benutzer immer mal wieder die Schlüssel wechseln. Ein ernsterer Nachteil des Protokolls ist jedoch, daß die Benutzer das Geheimnis ihrer verschiedenen Schlüssel mit dem KDC teilen müssen. Gelingt ein Einbruch in das KDC, sind auch die Schlüssel der Benutzer aufgedeckt.

Das folgende Protokoll 5.3.3 beruht auf der Kommunikation der Teilnehmer durch ein Public-Key-Kryptosystem. Jeder Teilnehmer $A$ besitzt einen öffentlichen Schlüssel $E_A$ und einen geheimen Schlüssel $D_A$, die entsprechenden Schlüssel für das KDC werden mit $E_K$ und $D_K$ bezeichnet.

**Protokoll 5.3.3**

(1) Jeder Teilnehmer $A$ teilt dem KDC seinen öffentlichen Schlüssel $E_A$ mit (diese Mitteilung kann mit Hilfe des öffentlichen Schlüssels $E_K$ geheim geschehen). $D_K(E_A)$

stellt das Zertifikat für den öffentlichen Schlüssel von $A$ dar. Dabei kann $E_A$ durch zusätzliche Informationen ergänzt werden.

(2) Das KDC sendet jedem Teilnehmer $A$ sein zugehöriges Zertifikat. Durch Anwendung von $E_K$ kann $A$ überprüfen, ob es tatsächlich $E_A$ enthält.

(3) Alice und Bob möchten miteinander einen sicheren Nachrichtenaustausch eröffnen. Alice sendet ihr Zertifikat an Bob und Bob sendet sein Zertifikat an Alice.

(4) Alice und Bob prüfen die Authentizität der Zertifikate durch Anwendung von $E_K$.

(5) Alice und Bob beginnen ihre Kommunikation mit den in den Zertifikaten enthaltenen Schlüsseln. □

Protokoll 5.3.3 hat viele Vorteile gegenüber Protokoll 5.3.2. Für jeden Benutzer reicht ein Zertifikat, mit dem er sich bei allen anderen Benutzern (durch Schritt 3 des Protokolls) einführen kann. Vor einer speziellen Kommunikation muß also das KDC nicht angerufen werden. Da das Zertifikat den öffentlichen, aber nicht den geheimen Schlüssel enthält, muß man es nicht unbedingt sicher verwahren. Auch das KDC kann, im Gegensatz zu Protokoll 5.3.2, die zwischen Alice und Bob ausgetauschten Nachrichten nicht lesen. Das hat den Vorteil, daß auch ein Eindringling in das KDC keinen direkten Zugang zu den geheimen Nachrichten hat.

Besitzt ein Eindringling $S$ den geheimen Schlüssel $D_K$, so muß er, um eine Nachricht von Alice an Bob zu erhalten, zunächst die Nachricht von Bob an Alice abfangen, in der Bob Alice sein Zertifikat mitteilt. $S$ erzeugt ein Paar $(E_S, D_S)$ und damit ein Zertifikat $D_K(E_S)$ und sendet dieses an Stelle von Bobs Zertifikat an Alice. Damit kann $S$ jede Nachricht von Alice an Bob lesen. Wenn jedoch eine so mit $E_S$ falsch chiffrierte Nachricht Bob erreicht, so kann er Alice darauf aufmerksam machen. Deswegen muß $S$ die Nachrichten von Alice an Bob, die jetzt mit $E_S$ verschlüsselt sind, auffangen, sie entschlüsseln und dann mit Bobs öffentlichem Schlüssel chiffrieren und an Bob weitersenden, um den Betrug aufrechtzuerhalten. Wenn $S$ Bobs Antworten verstehen will, muß er dieselbe Prozedur mit Bob durchführen. Die angegebene Prozedur ist auf jeden Fall sehr schwerfällig. Damit $S$ die Kontrolle behält, insbesondere, wenn er mehrere Teilnehmer ausspionieren will, muß er an vielen Stellen im Netz eingreifen. Wenn die Nachrichten über Radiowellen übertragen werden, ist eine Ersetzung der Nachrichten äußerst schwierig.

Man kann Protokoll 5.3.3 in ein hybrides System umwandeln, indem in Schritt 5 Alice und Bob mit Hilfe der gegenseitigen öffentlichen Schlüssel einen geheimen Schlüssel für z.B. den DES austauschen. In diesem Fall hat es der Eindringling jedoch etwas leichter. Nach Wahl des DES-Schlüssels, den er nach der obigen Methode abfangen kann, muß er den Kanal zwischen Alice und Bob nicht mehr kontrollieren, er muß nur noch ihre Nachrichten lesen. Dieser Gefahr können jedoch Alice und Bob entgegenwirken, indem sie immer wieder unter Benutzung der öffentlichen Schlüssel neue Sitzungsschlüssel vereinbaren.

Aufgedeckte Schlüssel sind eine große Gefahr, da damit auch die Nachrichten gelesen werden können, die damit in der Vergangenheit verschlüsselt wurden. Deswegen müssen nicht mehr gebrauchte Schlüssel zerstört werden. Dieses Problem wird gelöst in einem sicheren Telefon, das bei Bell-Northern Research entwickelt wurde (siehe [20], [66]) und im ISDN-Telefonnetz verwendet werden kann. Jedes sichere ISDN-Telefon besitzt ein Paar aus einem öffentlichen und einem geheimen Schlüssel, wobei der öffentliche Schlüssel wieder in einem Zertifikat enthalten ist, das, zusammen mit Telefonnummer und Name des Benutzers, von der Schlüsselverwaltungsstelle mit Hilfe eines Public-Key-Systems signiert wurde.

**Protokoll 5.3.4**

(1) Die Telefone führen Protokoll 5.3.1 aus, um den Sitzungsschlüssel (eines konventionellen Kryptosystems) für die aktuelle Verbindung zu bestimmen. Der weitere Nachrichtenaustausch erfolgt ab sofort mit diesem Sitzungsschlüssel.
(2) Jedes Telefon sendet dem anderen sein Zertifikat.
(3) Jedes Telefon überprüft die Signatur des Zertifikats und entnimmt ihm den öffentlichen Schlüssel des anderen Telefons.
(4) Die Telefone fordern sich jeweils auf, Testnachrichten mit den jeweiligen geheimen Schlüsseln zu chiffrieren, und sie überprüfen die Authentizität des anderen Telefons durch Anwendung des öffentlichen Schlüssels aus dem übermittelten Zertifikat auf die chiffrierte Testnachricht.
(5) Jedes Telefon stellt für seinen Benutzer auf dem Display die Telefonnummer und den Benutzernamen des anderen Telefons dar.
(6) Der geheime Nachrichtenaustausch (Telefongespräch) der Benutzer beginnt.
(7) Nach Beendigung der Verbindung wird der Sitzungsschlüssel zerstört. □

Auch die spätere Aufdeckung des im Telefon vorhandenen Paars aus öffentlichem und geheimen Schlüssel kann nicht zur Entschlüsselung der zuvor geführten Gespräche führen.

## 5.4 Protokoll zum Altersvergleich

Das folgende Protokoll stammt von *Yao* [104]. Zwei Personen, Alice und Bob, möchten herausfinden, wer von ihnen älter ist, ohne sich gegenseitig das Alter zu verraten. Alice ist $i$ Jahre, Bob $j$ Jahre alt. Am Ende wissen Alice und Bob, ob $i \geq j$ oder $i < j$ gilt, ohne jedoch weitere Eigenschaften von $i$ bzw. $j$ zu kennen. Es wird angenommen, daß das Alter der Spieler zwischen 1 und 100 Jahren liegt. Das Protokoll beruht auf einem Public-Key-Kryptosystem. Alice veröffentlicht dabei ihren Chiffrierschlüssel $E_A$. Es wird angenommen, daß $E_A(x)$ „vernünftig zufällige“ Werte für zufällig gewählte Werte von $x$ annimmt. Wir erhalten folgendes

**Protokoll 5.4.1**

(1) Bob wählt zufällig eine große Zahl $x$, deren Anzahl von Bits zuvor festgelegt wurde, und berechnet $E_A(x) = k$.
(2) Bob sendet Alice die Zahl $k - j$.
(3) Alice berechnet

$$y_u = D_A(k - j + u),\ u = 1, 2, \ldots, 100.$$

Hierbei muß $y_u \neq 0$ für alle $u$ gelten. Anderenfalls muß das Protokoll mit einem neuen $x$ von vorne beginnen. Alice wählt eine große Primzahl $p$, deren Anzahl von Bits ebenfalls vorher festgelegt wurde, und zwar etwas kleiner als die Anzahl der Bits von $x$. Daraus berechnet Alice

$$z_u = y_u \bmod p,\ u = 1, 2, \ldots, 100.$$

Es gilt $z_u < p$ für alle $z_u$. Falls $|z_u - z_{u'}| \geq 2$ für $u \neq u'$ und $0 < z_u < p - 1$ ist, hat Alice ein richtiges $p$ gewählt, andernfalls muß die Berechnung mit einem anderen $p$ erneut durchgeführt werden.

(4) Alice sendet Bob die Folge der Zahlen

$$z_1, z_2, \ldots, z_i, z_{i+1} + 1, z_{i+2} + 1, \ldots, z_{100} + 1, p.$$

(5) Bob prüft, ob $f_j = x \bmod p$ für die $j$-te Zahl $f_j$ der Folge gilt. Falls das der Fall ist, schließt Bob, daß $i \geq j$ gilt. Andernfalls gilt $i < j$.
(6) Bob teilt Alice seinen Schluß mit. □

Wir diskutieren dieses Protokoll. Wir betrachen den Schritt (5). Die Zahl $f_j$ bezieht sich auf Bobs Alter. Für $j \leq i$ gilt

$$\begin{aligned} f_j = z_j = y_j \bmod p &= D_A(k - j + j) \bmod p = D_A(E_A(x)) \bmod p \\ &= x \bmod p. \end{aligned}$$

Für $j > i$ gilt

$$\begin{aligned} f_j &= z_j + 1 \equiv_p (y_j + 1) \bmod p = (D_A(k - j + j) + 1) \bmod p \\ &= (D_A(E_A(x)) + 1) \bmod p \\ &= (x + 1) \bmod p \neq x \bmod p. \end{aligned}$$

Damit ist gezeigt, daß Bob die richtigen Schlüsse gezogen hat.

Wir überlegen uns weiter, warum Alice in Schritt (3) $z_u = y_u \bmod p$ bildet. Wenn wir annehmen, daß Alice Bob direkt die Folge

$$y_1, y_2, \ldots, y_i, y_{i+1} + 1, y_{i+2} + 1, \ldots, y_{100} + 1$$

mitteilt, dann erhält Bob durch die Anwendung von $E_A$

$$k - j + 1, \ldots, k - j + i, \ldots.$$

$i$ ergibt sich aus dem Index des Folgegliedes $k - j + i$, dessen Nachfolger sich erstmalig von $k - j + i + 1$ unterscheidet. Die Anwendung von $E_A$ auf $z_1, z_2, \ldots$ liefert dagegen nicht die Werte $k - j + 1, \ldots$. Wegen der Modulo-Bildung gibt es i. allg. viele verschiedene Sprünge.

Warum hat Alice in (3) bei $|z_u - z_{u'}| \geq 2$ für $u \neq u'$ und $0 < z_u < p - 1$ ein richtiges $p$ gewählt? Für $i < j$ kommt dadurch $x \bmod p$ in der in (4) an Bob gesendeten Folge nicht direkt vor, denn es gilt

$$(D_A(E_A(x) - j + j) + 1) \bmod p = (x + 1) \bmod p.$$

Andere Werte der Folge unterscheiden sich davon um mindestens 2. Das verhindert, daß Bob anderenfalls das $j'$ an der Stelle, die eventuell $x \bmod p$ liefert, als sein Alter angibt, was ihm nach (5) den Schluß $i \geq j$ erlauben würde, obwohl das Gegenteil richtig ist. Darüberhinaus ist nicht sicher, daß Bob in (6) die Wahrheit sagt.

Es kann aber ein noch ausgeklügelteres Protokoll angegeben werden, bei dem die Wahrscheinlichkeit eines Betruges verschwindend klein wird.

Das oben beschriebene Problem ist ein Spezialfall des folgenden allgemeinen Problems. Es gebe $n$ Parteien $A_1, A_2, \ldots, A_n$, die alle die Definition einer Funktion $f$ von $n$ Variablen kennen, also $f(x_1, x_2, \ldots, x_n)$. Zur Vereinfachung nehmen wir an, daß alle Variablen und auch der Funktionswert aus der gleichen endlichen Menge stammen. Jeder der $A_i$, $i =$

$1, 2, \ldots, n$, kennt nur den Wert von $x_i = a_i$, nicht aber $x_j = a_j$ mit $j \neq i$. Die Teilnehmer $A_1, A_2, \ldots, A_n$ möchten den Funktionswert $f(a_1, a_2, \ldots, a_n)$ berechnen, ohne dabei den anderen $A_j, j \neq i$, Informationen mitzuteilen, die den Wert $a_i$ ihrer Variablen $x_i$ betreffen. Das obige Beispiel mit Alice und Bob wird hier durch

$$f(x_1, x_2),\ 1 \leq x_1, x_2, \leq 100 \text{ und } f(x_1, x_2) = \begin{cases} 1 \text{ für } x_1 \geq x_2 \\ 2 \text{ für } x_1 < x_2 \end{cases}$$

dargestellt. Für diesen allgemeinen Fall kann ein recht kompliziertes Protokoll entworfen werden. Die notwendigen Maßnahmen zur Gewährleistung der Sicherheit sind schwierig zu formalisieren. Insbesondere sind die Maßnahmen zur Verhinderung des kollektiven Betruges, bei dem sich mehrere Parteien verbünden, um die anderen zu betrügen, schwer zu beschreiben. Auf jeden Fall eröffnen solche Protokolle neue Ausblicke auf Möglichkeiten der vertraulichen Kommunikation. Im folgenden Abschnitt werden wir sehen, wie Protokoll 5.4.1 auf einige Probleme des Wirtschaftslebens angewendet werden kann.

## 5.5 Protokolle für Auktionen und Geschäfte

Wir beginnen mit Auktionen. Im wesentlichen sind zwei Typen von Auktionen bekannt. Bei der englischen Auktion rufen die Bieter ihre Gebote für ein angebotenes Objekt (alte Fahrräder, Schmuck, Kunstobjekte, Immobilien usw.) laut in aufsteigender Ordnung des Wertes aus, und der Auktionator schlägt das Objekt dem Bieter zu, der am Ende das höchste Gebot abgeben hat. Dies höchste Gebot stellt dann auch den Preis dar. Bei der holländischen Auktion, die oft bei Fisch- oder Tulpenversteigerungen angewendet wird, ruft der Auktionator den Preis in absteigender Ordnung aus, bis der erste Bieter den gerade genannten Preis akzeptiert. Dann erhält er das Objekt zu diesem Preis.

Über Auktionen gibt es verschiedene theoretische Untersuchungen. *Vickrey* [102] hat gezeigt, daß diese mündlich geführten Auktionen auch in versiegelter Form geführt werden können. Bei der versiegelten Form werden die Gebote innerhalb einer bestimmten Frist beim Auktionator versiegelt eingereicht.

Die englische Auktion entspricht dann der Zweiten-Preis-Auktion. Bei diesem Auktions-Typ erhält derjenige Bieter den Zuschlag, der das höchste Gebot abgegeben hat, und er muß als Preis den Betrag des zweithöchsten Gebots bezahlen. Die holländische Auktion entspricht der Ersten-Preis-Auktion, wo der Meistbietende zum Preis seines Gebots den Zuschlag erhält.

In welchem Sinn entsprechen die versiegelten Auktionen den mündlichen? Das Problem eines Bieters bei einer holländischen Auktion ist es doch, sich vorab festzulegen, zu welchem Preis er das Objekt erwerben möchte. Genau dasselbe Problem stellt sich ihm bei der Ersten-Preis-Auktion. Unberücksichtigt bleibt dabei natürlich, daß man sich in der Atmosphäre eines Versteigerungssaales dazu hinreißen lassen kann, früher das Gebot anzunehmen als eigentlich geplant.

Bei einer englischen Auktion wird sich der Bieter im vorab darüber klar sein müssen, welchen Wert das Objekt hat oder für ihn darstellt. Nur bis zu diesem Betrag wird er mitbieten, danach ist es für ihn sinnlos. Wenn nur noch zwei Bieter im Rennen sind und einer aufgibt, dann muß der Meistbietende also nur wenig über diesen Betrag gehen, um den Zuschlag zu erhalten. Im wesentlichen erhält er das Objekt zu dem Preis des zweithöchsten Gebots. Er kann also ggf. weit unter dem Preis bleiben, den er vorab gewillt wäre zu zahlen. Somit entspricht die englische Auktion der Zweiten-Preis-Auktion. Bei der öffentlichen

Version können jedoch Koalitionen von Bietern unter Umständen den Preis hochtreiben. Insofern sind die öffentliche und versiegelte Version doch nicht äquivalent.

Die versiegelten Auktionstypen werden auch in der Realität abgehalten. So werden z.B. Schürfrechte auf Ländereien, die der US-Regierung gehören, durch Erste-Preis-Auktionen vergeben. Bei der Ersten-Preis-Auktion ist es sicher nicht die Hauptstrategie, als Gebot den Preis anzugeben, den man für den wahren Wert des Objekts hält.

Gelegentlich werden Autographen oder Briefmarken durch Zweite-Preis-Auktionen verkauft. Es wurde gezeigt ([102], [79]), daß die Zweite-Preis-Auktion einige schöne Eigenschaften hat. Unter anderem ist es die beste Strategie, die ein Bieter verfolgen kann, ein ehrliches Gebot abzugeben, das heißt ein Gebot zu einem Preis, der dem Bieter angemessen für das gebotene Objekt erscheint. Dies gilt unabhängig davon, ob auch die anderen Bieter ebenso ehrlich verfahren. Die Voraussetzungen, unter denen dies gilt, sind:

1. Die Bieter verhalten sich rational.
2. Die Werte, die die Bieter den Objekten zuordnen, sind privat und statistisch unabhängig.
3. Die Bieter sind symmetrisch, d.h. voneinander ununterscheidbar.
4. Der Preis bestimmt sich nur aus den Geboten.

Abgesehen vom letzten und, schon mit Einschränkungen, vom ersten Punkt, charakterisieren diese Voraussetzungen selten Auktionen der realen Welt. Zur weiteren Information verweisen wir auf [61], wo auch noch weitere Literatur zu dieser Problematik genannt wird.

Im folgenden werden wir zeigen, wie Erste-Preis- und Zweite-Preis-Auktionen mit Hilfe von Protokoll 5.4.1 implementiert werden können. Bei den traditionellen versiegelten Auktionen werden die Gebote zur praktisch selben Zeit geöffnet. Die kryptographische Form besitzt demgegenüber den Vorteil, daß nur wenige Gebote den Bietern und dem Auktionator aufgedeckt werden müssen, die meisten Gebote werden nicht geöffnet. Dies hat viele Vorteile. Oftmals möchte man, wenn man schon nicht den Zuschlag erhalten hat, sein Gebot geheim halten, weil daraus ggf. Rückschlüsse auf die eigenen finanziellen Möglichkeiten gezogen werden könnten. Außerdem können bei der normalen versiegelten Zweiten-Preis-Auktion die Bieter befürchten, daß der Auktionator sie betrügt. Das kann er dadurch erreichen, daß er zunächst die eingegangenen Gebote öffnet und dann einen Komplizen auffordert, ein möglichst hohes Gebot abzugeben, das nur knapp unter dem höchsten Gebot liegt. Diese Möglichkeit könnte die Bieter davon abhalten, ehrliche Gebote abzugeben. Dieser Nachteil entfällt bei der kryptographischen Version. Außerdem können die kryptographischen Erste- und Zweite-Preis-Auktionen in einem Computer-Netzwerk geführt werden, man benötigt weder die Post noch Kuriere.

Wir geben nun eine kryptographische Version der Ersten-Preis-Auktion an (siehe auch [61]). Es sei $A$ der Auktionator und $B_1, \ldots, B_n$ seien für $n \in \mathbb{N}$, $n \geq 2$, die Bieter. Die entsprechenden geheimen Dechiffrierfunktionen seien $D_A$, $D_{B_1}, \ldots, D_{B_n}$, die öffentlichen Chiffrierfunktionen seien durch $E_A$, $E_{B_1}, \ldots, E_{B_n}$ gegeben. Das Gebot von $B_i$ werde mit $p_i$ bezeichnet.

**Protokoll 5.5.1**

(1) Die Bieter melden sich bei $A$ an, der sie ohne Beschränkung der Allgemeinheit in der Reihenfolge $B_1, \ldots, B_n$ anordnet. $A$ informiert $B_l$ über die folgenden Bieter $B_{l+1}, \ldots, B_n$, $l = 1, \ldots, n-1$. Die möglichen Gebote werden durch Zahlen aus $\mathbb{N}$ im Intervall $[1, 100]$ dargestellt, also $p_l \in [1, 100]$ für $l = 1, \ldots, 100$.

(2) $B_1$ bis $B_n$ geben ihr Gebot bei $A$ ab, indem sie $A$ jeweils die Nachricht

$$(E_A(E_{B_l}(p_l s_l)), E_A(E_{B_l}(s_l))), l = 1, \ldots, n,$$

schicken. Dabei ist $s_l$ eine von $B_l$ geheim gewählte Zahl. Diese Nachrichten sind in ungeordneter Form allen Bietern zugänglich.

(3) „Bob“:= $B_1$; $j := p_1$; „Alice“:= $B_2$; $i := p_2$.

**for** $l := 1$ **to** $n - 1$ **do**

(4) „Alice“ und „Bob“ führen Protokoll 5.4.1 aus. Dabei teilt „Bob“ seinen Schluß $i < j$ „Alice“ in der Form „ich gewinne“ mit, den Schluß $i \geq j$ dagegen in der Form „du gewinnst“.

(5) **if** $l \neq n - 1$
**then if** „du gewinnst“ **then** „Bob“:= „Alice“; $j := i$ **end**;
„Alice“:= $B_{l+2}$; $i := p_{l+2}$
**end**

**end**.

(6) **if** „du gewinnst“ **then** $G :=$ „Alice“ **else** $G :=$ „Bob“ **end**.

(7) Der Gewinner $G$ sei $B_m$. $B_m$ teilt sich $A$ als Gewinner mit. $A$ verlangt von $B_m$ den geheimen Schlüssel $D_{B_m}$, der mit $D_A$ auch allen Bietern mitgeteilt wird. Mit $D_{B_m}$ und $D_A$ kann $A$ die Zahlen $s_m$ und $p_m s_m$ und damit das Gebot $p_m$ von $B_m$, also den Preis des Objekts, berechnen. □

Das Protokoll läuft so ab, daß ein Bieter, der einen Vergleich als „Alice“ durchführt, aufgrund von (1) nicht die Identität von „Bob“ kennt. Somit kennt für $n \geq 3$ nur $A$ den Gewinner, eventuell auch $B_{n-1}$ (dann muß $B_{n-1}$ seinen ersten Vergleich gewonnen und den zweiten mit $B_n$ verloren haben). Durch Anwendung der Schlüssel $D_A$ und $D_{B_m}$ auf alle in (2) eingereichten Nachrichten ergibt genau eine von ihnen einen vernünftigen Klartext, aus dem dann der Preis des Objektes bestimmt werden kann. Existieren mehrere Bieter, die dasselbe Gebot abgegeben haben, so wird durch Protokoll 5.5.1 genau derjenige mit dem höchsten Index als Gewinner bestimmt. Offensichtlich ist der kritische Punkt des Protokolls die Mitteilung von „Bob“ jeweils in Schritt (4). Niemand kann den jeweiligen Bieter „Bob“ daran hindern zu lügen. Von einer Lüge hat „Bob“ jedoch keinen Vorteil. Behauptet er fälschlich, daß sein Gebot niedriger ist als das von „Alice“ ist, so verzichtet er zu seinen Ungunsten auf eine weitere Beteiligung an der Auktion. Falls er fälschlich sagt, daß sein Gebot höher ist, kann jeder unterlegene Bieter durch Veröffentlichung seines geheimen Schlüssels diesen Betrugsversuch aufdecken. Andererseits möchte der Gewinner einen möglichst niedrigen Preis zahlen. Da das Gebot vorab schon dem Auktionator als auch allen anderen Bietern in verschlüsselter Form übergeben wurde, muß der Gewinner sich an sein Gebot halten, aber auch der Auktionator kann es z.B. nicht für einen Freund erniedrigen.

Für eine kryptographische Version der Zweiten-Preis-Auktion werden dieselben Vorbedingungen wie in Protokoll 5.5.1 vorausgesetzt.

**Protokoll 5.5.2**

(1) Es wird Protokoll 5.5.1 bis auf Schritt (7) durchgeführt.

(2) Der Gewinner $G$ sei $B_m$. $B_m$ teilt sich $A$ als Gewinner mit.

(3) Protokoll 5.5.1 wird von Schritt (3) bis Schritt (6) wiederholt, wobei $B_m$ sein Gebot in ein verlierendes Gebot geändert hat (z.B. $p_m = 1$).

(4) Der Gewinner sei $B_{m'}$. $B_{m'}$ übergibt $A$ sowie allen anderen Bietern seinen Schlüssel $D_{B_{m'}}$. Damit wird das zweithöchste Gebot und damit der Preis des Objekts bestimmt, für den $B_m$ das Objekt erwirbt.

Die Überlegungen für die erste Runde (Schritt (1)) sind dieselben wie in Protokoll 5.5.1. Für die zweite Runde soll angenommen werden, daß alle Bieter sich daran beteiligen, obwohl alle bis auf den Gewinner $B_m$ wissen, daß sie nicht den Zuschlag erhalten. Sollte im Gegensatz zu dieser Annahme ein Bieter die Teilnahme sabotieren, so könnte durch die Verabredung, daß dann alle Schlüssel und Identitäten offengelegt werden, das zweithöchste Gebot herausgefunden werden. Dann wäre allerdings der Vorteil der kryptographischen Version verloren. Ist dabei jemand nicht zu der Offenlegung bereit, so kann er mit Hilfe der Anmeldungsliste herausgefunden und von künftigen Auktionen ausgeschlossen werden. Diese Verabredung kann aber zur Folge haben, daß sich einige Bieter, die ihre Anonymität gewahrt wissen wollen, sich von vornherein nicht an der Auktion beteiligen. Deshalb könnte diese Offenlegung vielleicht gegenüber einer weiteren vertrauenswürdigen Partei erfolgen. Bei einer derartigen Verabredung werden in der Regel alle Bieter an der zweiten Runde teilnehmen.

Außerdem kann ein Bieter in der zweiten Runde lügen. Dadurch könnte es zu einer Erniedrigung des Preises kommen, also zu einem Nachteil für den Auktionator. Einem Konkurrenten das Objekt günstiger zukommen zu lassen, wird jedoch nicht im Interesse eines Bieters liegen. Außerdem kann dann jeder Bieter, der ein höheres Gebot abgegeben hat, diesen Betrugsversuch aufdecken. Es kann weiter versucht werden, z.B. durch Komplizen des Auktionators, einen Preis knapp unter dem Gebot des Hauptinteressenten zu erreichen, von dem vielleicht bekannt ist, daß er das Objekt unbedingt erwerben möchte. Dann müßten die Komplizen ihre Preise aber schon ganz am Anfang als Nachrichten an $A$ senden. Das könnte dazu führen, daß sie selber aus der ersten Runde als Gewinner hervorgehen. Wir sehen also, daß es im Interesse aller Bieter liegt, sich an das Protokoll zu halten.

Als letztes wollen wir den Fall betrachten, daß ein potentieller Käufer $K$ und ein Verkäufer $V$ eines Objekts feststellen wollen, ob es zwischen ihnen überhaupt zu einem Geschäft kommen kann. Wir nehmen an, daß $K$ höchstens bereit ist, den Betrag $k$ zu zahlen, wohingegen $V$ mindestens den Betrag $v$ verlangen will. Nur für $k \geq v$ kann ein Geschäft stattfinden. Offenbar können sie das mit Hilfe von Protokoll 5.4.1 feststellen. Was sind ihre Anreize, sich an das Protokoll zu halten? Gilt $V =$ „Bob“ und ist $v > k$, so kann ein Geschäft nicht zustande kommen und $V$ hat kein Interesse daran, $K$ darüber im Unklaren zu lassen. Ist $k \geq v$, so ist ein Geschäft möglich, warum sollte das $K$ nicht wissen? Entsprechende Überlegungen gelten, wenn $K =$ „Bob“ ist. Erst wenn so festgestellt wurde, daß ein Geschäft zustande kommen könnte, gehen $V$ und $K$ in mühevollere Detailverhandlungen ein, um den Preis des Objekts zu bestimmen.

## 5.6 Ein Protokoll für Wahlen

Im folgenden soll ein Protokoll zur Durchführung von geheimen Wahlen vorgestellt werden, daß im wesentlichen von *Nurmi* und *Salomaa* [63] vorgestellt wurde. Ähnliche Protokolle, von denen einige Aspekte zusätzlich übernommen wurden , befinden sich in [62], [64] und [60]. Das System erfüllt die folgenden Bedingungen:

(1) Die Stimmabgabe ist geheim in dem Sinne, daß nur der Wähler die von ihm ausgeübte Wahl (oder allgemeiner die Wahlstrategie) kennt.
(2) Nur legitimierte Wähler können wählen.
(3) Jeder Wähler kann nur eine Stimme abgeben.
(4) Die Wahl kann in einem Rechnernetz stattfinden.
(5) Jeder Wähler kann überprüfen, ob seine Stimme richtig gezählt wurde.
(6) Jeder Wähler kann Fehler des Stimmenzählsystems bezüglich seiner eigenen Wahl korrigieren.
(7) Ein Wähler kann nach Stimmabgabe bis zu einem vorher bestimmten Zeitpunkt seine Wahl ändern.

Es ist klar, daß eventuelle Änderungen der abgegebenen Stimme erfolgen müssen, bevor Ergebnisse der Wahl veröffentlicht werden. Das Protokoll kann leicht so abgeändert werden, daß diese Änderungen nicht zulässig sind.

Die folgenden Teilprotokolle werden für jeden Wähler durchgeführt. Dabei sei $A$ ein spezieller Wähler, $B$ das System zum Zählen der Stimmen und $C$ das System zur Überwachung des Wahlrechts von $A$. $B$ und $C$ können Rechner sein, ggf. auch ein einziger Rechner. Es wird ein Public-Key-Kryptosystem benutzt, z.B. das RSA-Verfahren. Dabei ist $E_A$ der öffentliche und $D_A$ der geheime Schlüssel von $A$. Entsprechend sind $E_B$, $D_B$, $E_C$ und $D_C$ gegeben. Die Stimme, also der Name des Kandidaten oder allgemeiner die Wahlstrategie, werden als Zahlen kodiert. Die Stimmabgabe wird gemäß dem folgenden Protokoll durchgeführt.

**Protokoll 5.6.1**
(Stimmabgabe und Veröffentlichung des Wahlresultats)

(1) $B$ und $C$ wählen eine große Menge $L$ großer Zahlen, deren Kardinalität sehr viel größer ist als die Anzahl der Wähler.
(2) Jeder Wähler $A$ sendet $C$ die Nachricht

$$(E_C(A), E_C(D_A(M)),$$

wobei (in kodierter Form) $A$ der Name von $A$ ist und $M$ die Nachricht „Mein Name ist $A$, ich möchte wählen“.
(3) $C$ berechnet aus der ersten Komponente $A = D_C(E_C(A))$ und kann so

$$M = E_A(D_C(E_C(D_A(M))))$$

bestimmen.
(4) $C$ überprüft, ob $A$ wahlberechtigt ist. Falls das nicht der Fall ist, wird $A$ zurückgewiesen. Anderenfalls bietet $C$ dem Wähler $A$ die Auswahl einer der Zahlen aus $L$ an, etwa $l_A \in L$. $A$ sucht genau eine Zahl aus, wobei $C$ nicht weiß, um welches $l_A$ es sich handelt (siehe Protokoll 5.6.4, mit dem diese Bedingungen erfüllt werden können).
(5) Jeder Wähler $A$ sendet $B$ seine Stimme in der Form

$$(E_B(l_A), E_B(f_A(l_A, w))).$$

Dabei ist $w$ seine numerisch kodierte Stimme, und $f_A$ ist eine von $A$ gewählte kryptographische Hash-Funktion (d.h.: selbst wenn die Definition von $f_A$ sowie die Werte $f_A(x, y)$ und $x$ gegeben sind, ist es in vernünftiger Zeit nicht möglich, $y$ zu berechnen, wohingegen diese Aufgabe leicht wird, wenn $f_A^{-1}$ bekannt ist. Als Beispiel einer geeigneten Hash-Funktionen kann das RSA-Schema gewählt werden, s.u.).

(6) $B$ kann $l_A$ und $f_A(l_A, w)$ berechnen und schickt eine „Quittung“

$$f_A(l_A, w)$$

über das ganze Rechnernetz.

(7) $A$ sendet $B$

$$E_B(l_A, f_A^{-1}),$$

wobei $(l_A, f_A^{-1})$ in numerischer Form angegeben ist, damit $E_B$ anwendbar ist.

(8) $B$ erhält damit $(l_A, f_A^{-1})$ und kann so aus der zweiten Komponente des Paares aus (5)

$$(l_A, w) = f_A^{-1}(D_B(E_B(f_A(l_A, w))))$$

und damit (s.o. unter (5)) $w$ bestimmen.

(9) $B$ veröffentlicht die Wahlresultate, indem für jeden Kandidaten $v$ (oder allgemeiner für jede Wahlstrategie $v$) die Liste der $f_A(l_A, w)$ angegeben wird, für die $w = v$ gilt. □

Wir geben nun einige Erläuterungen zu diesem Protokoll. Wenn z.B. $10^6$ Wähler teilnehmen, dann ist eine Menge $L$ mit $10^{100}$ Elementen als groß genug aufzufassen. Es ist, trotz des Geburtstagsparadoxons, in höchstem Maße unwahrscheinlich, daß zwei Wähler dieselbe Wahllegitimation $l_A$ erhalten. Es ist nicht nötig, daß $B$ und $C$ die Menge $L$ Zahl für Zahl kennen, sondern sie müssen sich nur auf Prinzipien zur Zugehörigkeit von Zahlen zu $L$ einigen. Sollte eine Wahllegitimation zweimal auftauchen, so können die zugehörigen Stimmen nicht berücksichtigt werden, weil beide Stimmen von demselben Wähler stammen könnten. Die erste Komponente der Nachricht in (2) erlaubt $C$ in Schritt (3) die Berechnung von $M$. Bei kleinen Wählergruppen ist die erste Komponente überflüssig, da $C$ in diesem Fall alle Schlüssel aller Wähler durchprobieren kann und nur der Schlüssel von $A$ einen vernünftigen Klartext und damit die Nachricht $M$ liefert. Auf diese Weise kann $C$ in (4) die Wahlberechtigung von $A$ überprüfen. Ein anderer Wähler kann sich nicht als $A$ ausgeben, sofern er von $A$ nicht den geheimen Schlüssel $D_A$ erhalten hat. Auf die damit verbundene Problematik werden wir später eingehen.

Durch die Kenntnis von $l_A \in L$ weist sich $A$ in (5) $B$ gegenüber als legitimer Wähler aus. Aus $f_A(l_A, w)$ kann $B$ weder den Namen von $A$ noch seine Stimme entnehmen. Als Beispiel einer kryptographischen Hash-Funktion kann

$$f_A(x, y) = (x \cdot y)^e \bmod n$$

gewählt werden, wobei $x, y \in [0, n-1]$ gelten muß und $n$ und $e$ gemäß dem RSA-Schema gewählt sind. Dabei ist darauf zu achten, daß $\text{ggt}(l_A, n) = 1$ gilt. Es kann zunächst $xy \bmod n$ berechnet werden. Wenn $y = l_A$ und $f_A^{-1}$ gegeben sind, dann kann nach Satz 3.11 $y^{-1} \bmod n$ und damit $x \bmod n = x$ eindeutig bestimmt werden. Nach Ausführung von Schritt (6) weiß $A$, daß seine Stimme bei $B$ eingetroffen ist, ohne daß seine Identität durch Schicken einer persönlichen Quittung von $B$ an $A$ aufgedeckt werden müßte. Nur wenn diese Quittung erfolgt, wird $A$ in (7) die inverse Hashfunktion an $B$ schicken. Anderenfalls wird er den Schritt (5) wiederholen. In Schritt (8) erhält $B$ die Stimme des Wählers mit der Wahllegitimation $l_A \in L$, wobei $B$ jedoch nicht den Namen von $A$ kennt. Auch wenn $B$ und $C$ ihr Wissen teilen, können sie den Namen von $A$ nicht herausfinden, da ab Schritt (5) die Chiffrierfunktion $E_A$ nicht benutzt wird und somit kein Zusammenhang zu dem zu

$E_A$ gehörigen Namen aus Schritt (1) bis (4) hergestellt werden kann. Nach der Veröffentlichung des Wahlresultats in (9) kann $A$ überprüfen, ob seine Stimme für den Kandidaten $v$ korrekt zugeordnet wurde. Er muß nur überprüfen, ob $f_A(l_A, w)$ in der Liste für $v$ vorhanden ist. Außerdem kann von jedem, der in (6) die Quittungen sammelt, festgestellt werden, ob alle quittierten Stimmabgaben berücksichtigt wurden.

Wenn in dem gesamten Protokoll die Anzahl der abgegebenen Stimmen größer als die Anzahl der Wähler ist, hat ersichtlich ein Betrug stattgefunden, ohne daß man jedoch den Betrüger ermitteln könnte. Wenn jeder Wähler wählt, ist ein solcher Betrug sofort feststellbar. In jedem Fall muß ein Betrüger mit einem gültigen $l_A \in L$ eine zusätzliche Stimme wie in (5) an $B$ schicken. Das kann ein Wähler, der keinen Zugang zu $L$ hat, im allgemeinen nicht erreichen. Es bleibt noch die Möglichkeit eines Betrugs durch $C$ oder $B$, entweder bei der Wahlzulassung (Zulassung von unberechtigten Wählern) oder bei Veröffentlichung der Wahlresultate. Wenn $B$ und $C$ unter Aufsicht eines unabhängigen Wahlausschusses arbeiten, denen Sympathisanten von allen Kandidaten angehören, kann dies (vielleicht) vermieden werden.

Unabhängig von einem Wahlausschuß, der vielleicht nicht alle Fehler von $B$ bemerkt, kann $A$, falls $B$ seine Stimme beabsichtigt oder unbeabsichtigt für einen falschen Kandidaten gezählt hat, Protest einlegen und eine Verbesserung der falschen Zuordnung verlangen. Dies geschieht durch

**Protokoll 5.6.2**
(Verbesserung von falschen Zuordnungen)

(10) $A$ protestiert, indem er $B$ die Nachricht

$$E_B(l_A, f_A(l_A, w), f_A^{-1})$$

schickt.

(11) $B$ entfernt $f_A(l_A, w)$ aus der Liste der Stimmen für den falschen Kandidaten und ergänzt stattdessen die Liste für den richtigen Kandidaten $w$ durch den in (10) gesendeten Wert von $f_A(l_A, w)$.

(12) Falls $A$ nicht die korrekte Zuordnung beobachtet (oder er es so behauptet), wird das Protokoll wiederholt. Trifft dies wiederholt zu, so wird der folgende Schritt durchgeführt. Anderenfalls wird das Protokoll beendet.

(13) $A$ muß

$$(l_A, f_A(l_A, w), f_A^{-1})$$

veröffentlichen. Jeder kann überprüfen, ob $B$ die Stimme richtig zugeordnet hat. □

Die erste Komponente $l_A$ in (10) zeigt, daß $A$ ein legitimer Wähler ist, die zweite, daß er die Hash-Funktion kennt und die dritte, daß er auch deren Inverses kennt. Nur $A$ kann dies alles wissen, so daß nur $A$ eine solche Änderung veranlassen kann.

Zwei Probleme können bei diesem Protokoll auftreten. Zum einen kann $B$ absichtlich $A$'s Stimme falsch zählen oder zum anderen kann $A$ versuchen, die Wahl zu sabotieren, indem er $B$ betrügerisches Verhalten vorwirft. In diesen Fällen wird (13) durchgeführt. $A$ muß sich dabei an seine zuerst abgegebene Stimme halten, da diese bereits in Schritt (6) von Protokoll 5.6.1 quittiert wurde und er keine andere bereits quittierte gültige Stimme zusammen mit einem entsprechenden Tripel erzeugen kann. $A$ muß hier der Allgmeinheit mehr preisgeben als in Protokoll 5.6.1, jedoch nicht seinen Namen. Dafür ist $B$ aber jetzt verpflichtet, die Änderung korrekt durchzuführen.

Schließlich soll noch der Fall betrachtet werden, daß $A$ seine Stimme in einem vorher bestimmten Zeitraum ändern möchte. Dafür kann es verschiedene Gründe geben. So kann er bei der ersten Stimmabgabe einfach einen Fehler gemacht haben, oder aufgrund neuer Informationen möchte er nun doch eine andere Stimme abgeben.

**Protokoll 5.6.3**
(Änderung der Stimme)
(14) $A$ sendet $B$ die Nachricht

$$(E_B(l_A), E_B(h_A(l_A, w')), E_B(f_A(l_A, w))),$$

wobei $h_A$ eine andere kryptographische Hash-Funktion und $w'$ die neue Stimme von $A$ ist.

(15) $B$ schickt eine Quittung

$$h_A(l_A, w')$$

über das ganze Rechnernetz.

(16) $A$ sendet $B$ die Nachricht

$$E_B(l_A, h_A^{-1}, f_A^{-1}).$$

(17) $B$ berechnet aufgrund der in (14) und (16) erhaltenen Werte die Stimme $w'$. $B$ entfernt $f_A(l_A, w)$ von der Liste der Stimmen für $w$ und fügt $h_A(l_A, w')$ in die Liste der Stimmen für $w'$ ein. □

Da $B$ nach Ausführung von Schritt (14) drei Komponenten erhält, weiß er, daß es sich um eine Änderung der Stimme handelt. Nach Erhalt von $f_A^{-1}$ und $h_A^{-1}$ überzeugt $B$ sich, daß $A$ diese Änderung veranlassen möchte. Niemand, der irgendwie eine Wahllegitimation von $A$ kennt, kann eine Änderung im Namen von $A$ veranlassen. Er müßte $f_A^{-1}$ kennen, das $B$ schon zuvor in Schritt (8) von Protokoll 5.6.1 erhalten hat und mit dessen Hilfe $B$ sich überzeugen kann, daß auch in der zu ändernden Stimme $f_A(l_A, w)$ dieselbe Wahllegitimation $l_A$ vorkommt. Nur wenn ein Wähler $A$ Schritt (13) durchgeführt hat, kann ein anderer in seinem Namen Protokoll 5.6.3 durchführen. Wenn $A$ jedoch als ehrlicher Wähler Schritt (13) durchführen muß, dann ist das Vertrauen in das Zählsystem $B$ ohnehin verloren gegangen. Am Ende des Protokolls kann $A$ überprüfen, ob sein Stimmwechsel korrekt in der Liste der Wahlresultate erscheint. Anderenfalls kann er entsprechend Protokoll 5.6.2 verfahren. Kein anderer kann erkennen, welcher Wähler seine Stimme geändert hat.

Wir geben jetzt das Protokoll an, mit dessen Hilfe $A$ in Schritt (4) von Protokoll 5.6.1 seine Wahllegitimation erhält. Es ist ein ANDOS-Protokoll (all-or-nothing disclosure of secrets-protocol), das von *Renvall* [75] vorgestellt wurde. $C$ besitzt eine Menge $L$ von Geheimnissen. Die allgemeine Natur der Geheimnisse ist bekannt. So weiß vielleicht jeder, daß $l_1$ die Klausur zu einer Vorlesung $V_1$ ist, $l_2$ die Klausur zu einer Vorlesung $V_2$, $l_3$ der Kontostand einer bestimmten Person bei seiner Bank, usw. Wir nehmen an, daß $A$ an einem dieser Geheimnisse interessiert ist, aber $C$ nicht verraten möchte, welches Geheimnis er wissen möchte. Nehmen wir an, daß $A$ an dem Geheimnis $l_i$ interessiert ist, $\text{card}(L) = k$ gilt und jedes Geheimnis als eine $n$-Bit-Zahl dargestellt werden kann. Um die Übermittlung der Nachrichten im folgenden Protokoll vor Dritten zu schützen und die Authentizität zu gewährleisten, sollten die Nachrichten entsprechend mit den Schlüsseln $E_A$, $D_A$, $E_C$ und $D_C$ chiffriert werden.

**Protokoll 5.6.4**
(Geheimes Verkaufen von Geheimnissen)

(1) Nachdem $C$ festgestellt hat, daß $A$ wahlberechtigt ist und noch keine Wahllegitimation $l_A$ erhalten hat, übermittelt $C$ die Einwegfunktionen $f$ und $g$ an $A$. $C$ hält deren Inverse $f^{-1}$ und $g^{-1}$ geheim.

(2) $A$ wählt zufällig eine $n$-Bit-Zahl $a$ und eine Menge

$$\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k\}$$

von zufälligen $n$-Bit-Zahlen. Er möchte das $i$-te Geheimnis $l_i$ als Wahllegitimation $l_A$ erhalten. Er schickt an $C$ die Folge

$$v_1, \ldots, v_k \text{ mit } v_i = f(a) \oplus g(\bigoplus_{j \neq i} x_j) \text{ und } v_j = x_j \text{ für } j \neq i.$$

(3) $C$ berechnet $z_l = v_l \oplus g(\bigoplus_{j \neq l} v_j)$ $(l = 1, \ldots, k)$ und sendet $A$ die Folge

$$y_j = l_j \oplus f^{-1}(z_j),\ j = 1, \ldots, k.$$

(4) $A$ berechnet $l_A = l_i = y_i \oplus a$. □

$A$ kann die Berechnung in (4) durchführen, und es gilt in der Tat

$$y_i \oplus a = y_i \oplus f^{-1}(f(a)) = y_i \oplus f^{-1}(v_i \oplus g(\bigoplus_{j \neq i} x_j)) = y_i \oplus f^{-1}(z_i) = l_i.$$

Dabei wird das zweite Gleichheitszeichen mit der Hilfe der Gleichung aus (2) und das letzte mit der aus (3) bewiesen. Es ist klar, daß $C$ nicht $A$'s Interesse, also seinen Wert $i$, erfahren kann. Das liegt daran, daß $A$ alle Zahlen $v_1, \ldots, v_k$ an $C$ gibt. Alle diese $v_j$ bis auf $v_i$ sind zufällige Zahlen, und $v_i$ ist aus diesen $v_j$ und dem ebenfalls zufälligen $a$ vollständig bestimmt, wobei $a$ nicht aufgedeckt wird.

Wenn $C$ sicher sein will, daß $A$ ein bestimmtes $l_i$ erhält, dann muß er, da er $i$ nicht kennt, in (3) immer $l_j = l_i$ setzen. Für ein solches $j$ kann $A$ den Wert $l_i \oplus y_j = l_j \oplus y_j = f^{-1}(z_j)$ und nach Anwendung von $f$ schließlich $z_j$ berechnen, das er selbst durch die Wahl der verschiedenen Größen in (2) bestimmt hat. Geht dieser Test für mindestens zwei verschiedene $j$ positiv aus, dann hat $C$ betrogen.

Kann $A$ mehr als ein Geheimnis durch dies Protokoll erfahren? Wenn dies der Fall wäre, also etwa $l_i$ und $l_{i'}$ diese Geheimnisse sind und $v_i$ und $v_{i'}$ die entsprechenden Werte in (2), so müßte $A$ gemäß (2) diese Werte $v_i$ und $v_{i'}$, die zugehörigen $a$ und $a'$ sowie $v = \bigoplus_{j \neq i,i'} v_j$ kennen, die das folgende Gleichungssystem erfüllen:

$$v_i \oplus g(v_{i'} \oplus v) = f(a),$$
$$v_{i'} \oplus g(v_i \oplus v) = f(a').$$

Wenn $A$ die Funktionen $f$ und $g$ nicht invertieren kann, besteht der einzige Weg, das Gleichungssystem zu lösen, wohl darin, $a$, $a'$, $w_i = v_i \oplus v$ und $w_{i'} = v_{i'} \oplus v$ festzuhalten. Dann kann $v_i$ aus der ersten und $v_{i'}$ aus der zweiten Gleichung berechnet werden. Diese Werte müssen aber auch $v_i \oplus v_{i'} = w_i \oplus w_{i'}$ erfüllen. Die Wahrscheinlichkeit, daß diese Gleichung gilt, ist praktisch 0, da $v_1$ und $v_2$ zufällige Zahlen sind.

Ein ernstes Problem in Zusammenhang mit Protokoll 5.6.4 ist sofort zu erkennen. Wenn etwa $L$, wie auf Seite 100 gesagt, $10^{100}$ Elemente hat, also $k$ so riesig ist, dann ist Protokoll 5.6.4 nicht mehr durchzuführen. Es ist also unmittelbar nur für kleinere Wählergruppen anzuwenden, wo die Werte entsprechend kleiner angesetzt werden können. Zur Verringerung des Wertes $k$ im obigen Protokoll könnte man bei einer landesweiten Wahl wenigstens bei der Registrierung Stimmbezirke einführen, wobei in den Bezirken paarweise disjunkte Mengen $L$ verwendet werden. Mit diesen Maßnahmen läßt sich die gesamte Wahl praktikabel in einem Rechnernetz durchführen, und die zu Beginn dieses Abschnitts genannten Eigenschaften (1) bis (7) sind erfüllt.

Diese Eigenschaften sind jedoch nicht alle Eigenschaften, die man von einer fairen Wahl fordern sollte. Bei einer üblichen geheimen Wahl treten die Wähler in eine Wahlkabine, und niemand darf einen Wähler dabei begleiten. Dies sichert, daß Stimmen nicht gekauft werden können, da der eventuelle Käufer der Stimme am Gesamtergebnis der Wahl nicht ablesen kann, ob die gekaufte Stimme tatsächlich in seinem Sinne abgegeben wurde. Ganz anders liegen leider die Verhältnisse in dem hier vorgestellten Protokoll. Der Käufer braucht sich nur zum Verkäufer der Stimme mit an den Rechner zu setzen, um die Stimmabgabe zu beobachten. Bei Überlassung von $D_A$ kann der Käufer sogar die gesamte Wahl für $A$ durchführen. In [60] werden Protokolle vorgestellt, die diesen Stimmenkauf verhindern. Leider sind sie nicht für große Wählergruppen durchführbar.

# 6 Ein Protokoll für Kreditkartenzahlungen

Die Unternehmen MasterCard und VISA haben *Secure Electronic Transaction* (SET) entwickelt, einen offenen Standard für sichere Kreditkartenzahlungen. Ursprünglich (1995) präsentierten beide Organisationen einen eigenen Standard, doch Anfang 1996 einigten sich beide auf einen gemeinsamen, der unter anderem auch von American Express unterstützt wird. Der Sinn von SET ist es, zur größeren Akzeptanz von Kreditkarten im Internet beizutragen. Dabei wird durch den Einsatz der Kryptologie die Vertraulichkeit von Informationen (Nachrichtenverschlüsselung), die Integrität von Zahlungen sowie die Identität von Händlern und Kartenhaltern (digitale Unterschrift mit Zertifikat) gewährleistet. SET kann auf allen möglichen Hard- und Softwareplattformen implementiert werden und erhält dabei die Kompatibilität. SET erfaßt die sichere Abwicklung von Kaufvorgängen von der Bestellung bis zur Quittung. Darüber hinaus werden auch Prozeduren für die Erteilung von Zertifikaten beschrieben, die Hierarchie der Zertifizierungsstellen wird angegeben. Eine ausführliche Beschreibung von SET ist im Internet [55] abrufbar.

Die Vertraulichkeit der Versendung von Nachrichten wird in SET durch die DES-Verschlüsselung der Nachrichten durch einen zufällig erzeugten 56-Bit-DES-Schlüssel gegeben. Sehr sensible Daten wie Kreditkartennummer und Kontoinformationen werden dabei jedoch RSA-verschlüsselt, wobei Schlüssel mit 1024 Bits verwendet werden. Die Integrität der Nachrichten und die Authentizität wird durch digitale Signaturen gewährleistet, die durch den Fingerabdruck einer Nachricht (*message digest*) und die anschließende Anwendung des öffentlichen Schlüssels ausgeführt werden. In diesem Zusammenhang werden auch duale Signaturen verwendet. Im folgenden wollen wir zunächst kurz auf Hashfunktionen und digitale Signaturen eingehen, um dann einige spezielle Eigenschaften von SET zu schildern und schließlich die Protokolle für Bestellung (*purchase request*), Zahlungsauthorisierung (*payment authorization*) und die Abrechnung des Händlers mit der Kundenbank (*payment capture*) ausführlich darzustellen.

## 6.1 Hashfunktionen, digitale Signaturen und duale Signaturen

Digitale Signaturen wurden bereits im Abschnitt 4.1 betrachtet (siehe Protokoll 4.1.1). Dies kann im Prinzip mit Hilfe des RSA-Verfahrens implementiert werden. Für längere Dokumente ist das aber sehr aufwendig, so daß statt dessen aus dem Dokument ein „Fingerabdruck“ (*message digest*) gewonnen wird, der dann mit dem geheimen RSA-Schlüssel signiert wird. Der Fingerabdruck wird mit einer *Einweg-Hashfunktion* berechnet.

Eine Hashfunktion $H$ erzeugt aus einer Nachricht $M$ einen eindeutigen Hashwert

$$h = H(M) \text{ mit fester Länge } m.$$

Einweg-Hashfunktionen besitzen folgende zusätzliche Eigenschaften:

(1) Zu gegebenem $M$ kann man leicht $h$ berechnen.

(2) Zu gegebenem $h$ ist es schwer (praktisch unmöglich), eine beliebige Nachricht $M'$ zu bestimmen mit $H(M') = h$.

Damit diese Bedingungen erfüllt sind, müssen Einweg-Hashfunktionen sorgfältig konzipiert werden. Eine ganze Reihe solcher Funktionen (u.a. MD4, MD5, SHA) ist in Kapitel 18 von [86] beschrieben. Zur genaueren Information verweisen wir auf dieses Buch. SET verwendet SHA (*secure hash algorithm*). SHA erzeugt aus einer beliebigen Nachricht einen Wert der Länge 160 Bit. Bei 160 Bits ist es plausibel, daß die Eigenschaft (2) erfüllt ist.

Wir geben ein Protokoll an, das die Authentifizierung unter Verwendung einer Einweg-Hashfunktion $H$ beschreibt. Wie üblich sei $E_A$ der öffentliche und $D_A$ der geheime Schlüssel von Alice beim RSA-Verfahren.

**Protokoll 6.1.1**
(digitale Signatur)

(1) Alice bildet den Fingerabdruck (*message digest*) $H(M)$ ihrer Nachricht und die Signatur $D_A(H(M))$ und sendet Bob

$$(M, D_A(H(M))).$$

(2) Bob gewinnt durch Berechnung von $E_A(D_A(H(M)))$ den Fingerabdruck zurück und berechnet dann $H(A)$. Er überprüft, ob beide Werte gleich sind. Im positiven Fall hält er $M$ für authentisch. □

Mit Hilfe einer *dualen Signatur* ist es möglich, zwei getrennte Nachrichten mit einer gemeinsamen Unterschrift so zu verbinden, daß sie nicht aus dem Zusammenhang gerissen werden können. Jede Nachricht soll nur jeweils ein Empfänger lesen können. Beide Nachrichten müssen tatsächlich erzeugt worden sein, damit die Überprüfung der Signatur einer beliebigen der beiden Nachrichten positiv ausgeht. Wir nehmen an, daß Alice eine signierte Nachricht $M_1$ an Bob und eine signierte Nachricht $M_2$ an Charles schicken möchte. $H$ sei wieder die Einweg-Hashfunktion und $E_A$ und $D_A$ die Transformationen von Alice für das RSA-Verfahren.

**Protokoll 6.1.2**
(duale Signatur)

(1) Alice bildet die Fingerabdrücke $H(M_1)$ und $H(M_2)$ ihrer Nachrichten $M_1$ und $M_2$, konkateniert sie und bildet vom Ergebnis erneut einen Fingerabdruck. Vom Resultat erzeugt sie die Signatur

$$D_A(H(H(M_1)|H(M_2))).$$

Sie sendet Bob

$$(M_1, H(M_2), D_A(H(H(M_1)|H(M_2)))).$$

(2) Bob gewinnt durch $E_A(D_A(H((H(M_1)|H(M_2)))))$ den Fingerabdruck zurück und berechnet dann $H((H(M_1)|H(M_2)))$, da er $M_1$ und $H(M_2)$ kennt. Er überprüft, ob beide Werte gleich sind.

(3) Charles kann auf analoge Weise eine Signatur von $M_1$ erhalten. □

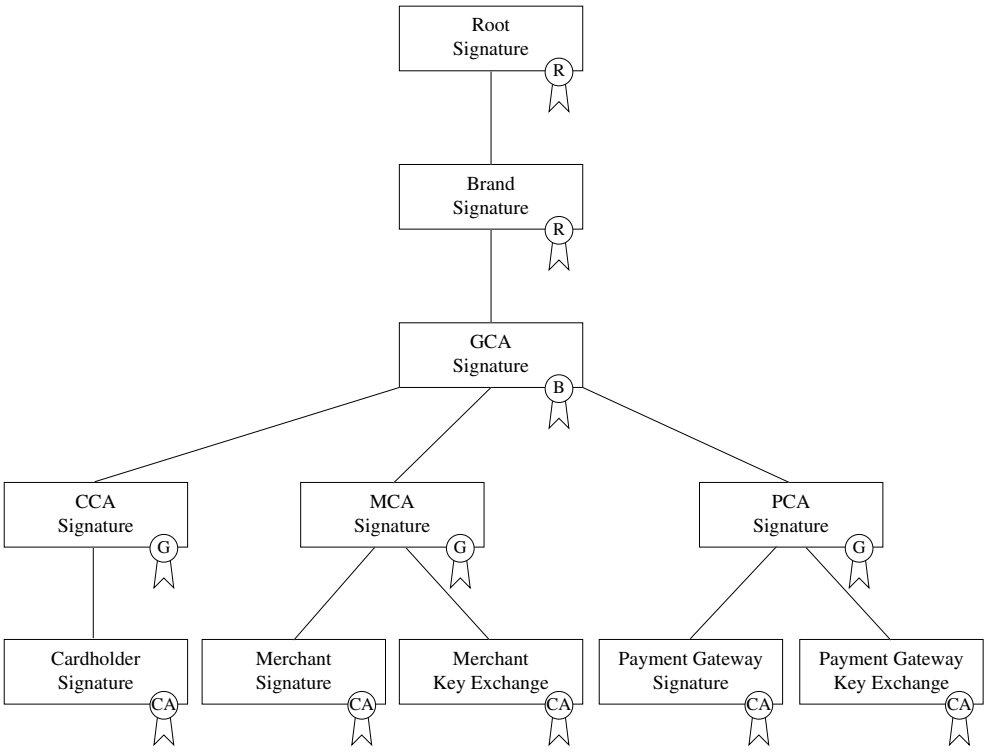
Bob weiß nichts über die Nachricht $M_2$, er besitzt jedoch eine gültige Unterschrift $H(M_2)$ dieser Nachricht, die er Charles vorlegen kann. Umgekehrt kann Charles Bob die Unterschrift $H(M_1)$ vorlegen.

## 6.2 Zertifikate

Das RSA-Verfahren verlangt die Kenntnis der öffentlichen Schlüssel des Kommunikationspartners. SET hängt die benötigten öffentlichen Schlüssel in Form von Zertifikaten an jede unterschriebene Nachricht an. Empfängt man eine Nachricht, wird zuerst das Zertifikat überprüft. Im Zertifikat befinden sich die Daten und die Signatur der ausstellenden Zertifizierungsstelle. Die Daten bestehen unter anderem aus dem Schlüssel, dem Namen der Zertifizierungsstelle und der Gültigkeitsdauer des Zertifikats. Wir wollen hier kurz und sehr vereinfacht beschreiben, wie ein Zertifikat eines Kartenhalters durch eine Kartenhalterzertifizierungsstelle (CCA, *cardholder certificate authority*) ausgegeben werden kann.

(1) Der Kartenhalter bittet das CCA um ein Zertifikat.
(2) Das CCA antwortet mit einem Zertifikat zur Verschlüsselung, das der Kartenhalter benutzen kann, um dem CCA geheim Informationen schicken zu können.
(3) Der Kartenhalter chiffriert seine Kreditkartennummer und schickt sie an das CCA.
(4) Das CCA antwortet mit dem kartenspezifischen Registrierungsformblatt.
(5) Der Kartenhalter vervollständigt dieses Formblatt, wozu auch sein öffentlicher Signaturschlüssel gehört, und sendet es zur Zertifizierung an das CCA.
(6) Das CCA überprüft die Informationen des Kartenhalters mit dem Ausgeber der Kreditkarte, erzeugt das Zertifikat, signiert es und schickt es an den Kartenhalter.

In ähnlicher Form werden auch Zertifikate für Händler und Banken durch eine Händlerzertifizierungsstelle (MCA, *merchant certificate authority*) und eine Bankenzertifizierungsstelle (PCA, *payment gateway certificate authority*) erzeugt, die jedoch jeweils neben dem öffentlichen Signaturschlüssel auch einen öffentlichen Schlüssel zum Austausch von symmetrischen DES-Schlüsseln enthalten müssen. Das Zertifikat des Kartenhalters dient zum Nachweis seiner Authentizität, es ist mit dem Präsentieren der Plastikkarte zu vergleichen. Das Zertifikat des Händlers kann man mit dem Aufkleber „Wir akzeptiern MasterCard“ an der Ladentür vergleichen. Jede Zertifizierungsstelle kann ein Zertifikat einer übergeordneten Zertifizierungsstelle vorweisen. Dadurch ergibt sich insgesamt eine Vertrauenshierarchie. Dies wird durch das folgende Bild verdeutlicht.



Die Siegel geben an, mit welchen privaten Schlüsseln die jeweiligen Zertifikate signiert wurden. Ein durch eine Zertifizierungsstelle $CA$ signierter öffentlicher Signaturschlüssel $E_C$ eines Kartenhalters $C$ können wir sehr vereinfacht als $(E_C, D_{CA}(H(E_C)))$ darstellen, wobei $D_{CA}$ der geheime Schlüssel der Zertifizierungsstelle ist und $H$ die Einweg-Hashfunktion SHA ist. Nur jemand, der den geheimen Schlüssel $D_{CA}$ von $CA$ kennt, kann dieses Zertifikat erzeugen. Zur Abkürzung werden wir es in den folgenden Protokollen mit $Z_{CA}(E_C)$ bezeichnen. Neben den bereits erwähnten Stellen ist noch die geopolitische Zertifizierungsstelle (GCA, *geopolitical certificate authority*), die Markenzertifizierungsstelle (BCA, *brand certificate authority*), wobei eine Marke z.B. MasterCard oder VISA sein kann, und die Wurzelzertifizierungsstelle (RCA, *root certificate authority*) aufgeführt. Die GCA erlauben es den Marken, in verschiedenen Regionen verschiedene Strategien für weitere Zertifizierungsstellen zu verwenden. Der Wurzelschlüssel hat eine Länge von 2048 Bits und wird mit einem selbstsignierten Zertifikat verteilt. Es steht Softwareverkäufern zur Verfügung, um es in ihre Software zu integrieren, und es kann auch weltweit in der Presse veröffentlicht werden. Ein Kartenhalter überprüft nun das Zertifikat der Kartenhalterzertifizierungsstelle, indem er alle Zertifikate mit Hilfe der Vertrauenskette zurück bis hin zur Wurzel verifiziert, d.h. alle Zertifikate auf dem Pfad zurück bis zur Wurzel.

## 6.3 SET-Protokolle

Wir geben zunächst an, wie eine Kundenbestellung abläuft. Bevor das eigentliche SET-Protokoll durchgeführt wird, hat der bestellende Kunde $C$ (*cardholder*) sich über die zu bestellende Ware informiert. Er hat ein Bestellformular vervollständigt und ist mit dessen Inhalt und Bedingungen einverstanden, z.B. mit der Anzahl der Ratenzahlungen, falls der Händler $M$ (*merchant*) solche akzeptiert. Er hat eine Kreditkarte zur Bezahlung ausgewählt. Neben dem Zertifikat $Z_{CA}(E_C)$ des Kunden $C$ für die Signatur benutzen wir auch die Zertifikate $Z_{CA}(E_M)$ und $Z_{CA}(E'_M)$ des Händles $M$ für die Signatur bzw. den Schlüsselaustausch und die Zertifikate $Z_{CA}(E_P)$ und $Z_{CA}(E'_P)$ der „Bezahlungszugangsstelle“ $P$ (*payment gateway*) des Bankinstituts des Händlers für die Signatur bzw. den Schlüsselaustausch. Wir merken an, das $CA$ nach unserer Vertrauenshierarchie für verschiedene Zertifizierungsstellen steht.

**Protokoll 6.3.1**
(purchase request, Bestellung)

(1) Der Kunde $C$ schickt dem Händler $M$ eine Initialisierungsnachricht (initiate request), in der er um eine Kopie des Zertifikats des *payment gateway* $P$ des Bankinstituts von $M$ bittet.

(2) Der Händler $M$ erzeugt eine Antwort $A$ (einschließlich einer Transaktionskennzeichnung $TI$ (transaction identifier)) und signiert sie durch

$$D_M(H(A)).$$

$M$ schickt

$$(A, D_M(H(A)), Z_{CA}(E_M), Z_{CA}(E'_P))$$

an den Kunden $C$.

(3) $C$ überprüft die Zertifikate $Z_{CA}(E_M)$ und $Z_{CA}(E'_P)$ mit Hilfe der Vertrauenskette bis zur Wurzel. $C$ überprüft die Signatur von $A$ durch Berechnung von

$$E_M(D_M(H(A))) \text{ und } H(A),$$

die beide gleich sein müssen.

(4) $C$ erzeugt eine Bestellinformation $OI$ (order information) sowie eine Instruktionen über die Zahlweise $PI$ (payment instructions), wobei in beiden Fällen die Transaktionskennzeichnung $TI$ dazugehört.

(5) $C$ erzeugt die duale Signatur

$$DS = D_C(H(H(OI)|H(PI)))$$

von $OI$ und $PI$.

(6) $C$ erzeugt zufällig einen DES-Schlüssel $K_1$ und berechnet

$$E_{K_1}(PI, DS, H(OI)).$$

Der Schlüssel $K_1$ und eine Kreditkarteninformation $KI$ werden durch den „digitalen Umschlag“ (digital envelope)

$$E'_P(K_1, KI)$$

verschlüsselt.

(7) $C$ übermittelt dem Händler die Nachricht

$$(OI, DS, H(PI), E'_P(K_1, KI), E_{K_1}(PI, DS, H(OI)), Z_{CA}(E_C)).$$

(8) Der Händler $M$ überprüft das Zertifikat $Z_{CA}(E_C)$ mit Hilfe der Vertrauenskette bis zur Wurzel. Er kennt jetzt $E_C$, $DS$, $OI$ und $H(PI)$ und kann damit

$$E_C(DS) \text{ und } H(H(OI)|H(PI))$$

berechnen, die beide gleich sein müssen. Damit ist die Signatur des Kunden $C$ überprüft.

(9) $M$ sendet neben anderen Informationen

$$(E'_P(K_1, KI), E_{K_1}(PI, DS, H(OI)))$$

an das *payment gateway* $P$ seiner Bank (siehe Protokoll 6.3.2, $M$ kann weder $KI$ noch $PI$ lesen).

(10) $M$ erzeugt eine Quittung $Q$ und sendet

$$(Q, D_M(H(Q)), Z_{CA}(E_M))$$

an den Kunden $C$.

(11) Wenn das *payment gateway* $P$ der Bank des Händlers die Zahlungsanweisung des Kunden $C$ akzeptiert (siehe Protokoll 6.3.2), wird bei übereinstimmenden Transaktionskennzeichnungen $TI$ die Bestellung von $M$ erfüllt.

(12) Der Kunde $C$ überprüft das Zertifikat $Z_{CA}(E_M)$ des Händlers $M$ mit Hilfe der Vertrauenskette bis zur Wurzel. Er überprüft die Signatur von $M$ durch Vergleich von

$$E_M(D_M(H(Q))) \text{ und } H(Q).$$

(13) Abhängig vom Inhalt von $Q$ führt $C$ einige Aktionen durch (z.B.: Änderung einer Datenbank).

In Schritt (9) von Protokoll 6.3.1 fragt der Händler bei seiner Bank an, ob sie die Zahlungsanweisung des Kunden akzeptiert. Die notwendigen Schritte beschreiben wir mit dem Protokoll zur *payment authorization*.

**Protokoll 6.3.2**
(payment authorization, Zahlungsauthorisierung)

(1) Der Händler $M$ bereitet die Anfrage $R$ auf Zahlungsfreigabe (*authorization request*) bei dem *payment gateway* $P$ seiner Bank vor und signiert $R$ durch Bildung von

$$D_M(H(R)).$$

(2) $M$ erzeugt zufällig einen DES-Schlüssel $K_2$. Der Schlüssel wird durch $E'_P(K_2)$ chiffriert und die signierte Anfrage durch $E_{K_2}(R, D_M(H(R)))$ verschlüsselt.

(3) $M$ übermittelt dem *payment gateway* $P$ die Nachricht

$$(E'_P(K_2), E_{K_2}(R, D_M(H(R))), (E'_P(K_1, KI), E_{K_1}(PI, DS, H(OI))),\\ Z_{CA}(E_C), Z_{CA}(E_M), Z_{CA}(E'_M)),$$

wobei die dritte und vierte Komponente in Schritt (6) von Protokoll 6.3.1 vom Kunden $C$ erzeugt wurden (verschlüsselte Zahlungsanweisung des Kunden) und in der letzten Komponente sein Zertifikat für den Schlüsselaustausch übermittelt wird.

(4) Das *payment gateway* $P$ überprüft die drei Zertifikate mit Hilfe der Vertrauenskette bis zur Wurzel.

(5) $P$ erhält den symmetrischen Schlüssel $K_2$ des Händlers durch Berechnung von $D'_P(E'_P(K_2))$ und kann damit

$$(R, D_M(H(R)))$$

bestimmen. $P$ überprüft die Signatur von $M$ durch Berechnung von

$$E_M(D_M(H(R))) \text{ und } H(R),$$

die beide gleich sein müssen.

(6) $P$ erhält den symmetrischen Schlüssel $K_1$ sowie die Kreditkarteninformation $KI$ des Kunden $C$ durch Berechnung von $D'_P(E'_P(K_1, KI))$ und kann damit

$$(PI, DS, H(OI))$$

bestimmen.

(7) $P$ überprüft die duale Signatur $DS$ auf $PI$ mit Hilfe von $H(OI)$ durch Berechnung von

$$E_C(DS) = E_C(D_C(H(H(OI)|H(PI)))) \text{ und } H(H(OI)|H(PI)),$$

die beide gleich sein müssen.

(8) $P$ prüft, ob die Zahlungsanweisung $PI$ des Kunden mit den Angaben des Händlers aus $R$ zusammenpassen. Insbesondere muß das Transaktionskennzeichen $TI$ aus $PI$ auch in $R$ genannt sein, wobei $M$ den Wert von $TI$ aus der Bestellinformation $OI$ entnehmen kann.

(9) $P$ sendet die Anfrage $R$ über ein Bankennetzwerk an die Bank des Kunden.

(10) Nach Erhalt einer positiven Antwort von der Bank des Kunden erzeugt $P$ eine Authorisierungs-Antwort $AR$ und signiert diese durch

$$D_P(H(AR)).$$

(11) $P$ erzeugt zufällig einen DES-Schlüssel $K_3$. Der Schlüssel und die Kontoinformation $KIM$ des Händlers wird durch $E'_M(K_3, KIM)$ chiffriert und die signierte Antwort durch $E_{K_3}(AR, D_P(H(AR)))$ verschlüsselt.

(12) $P$ erzeugt ein *capture token* $CT$, das unter anderem das Transaktionskennzeichen $TI$ aus $OI$ enthält und in Protokoll 6.3.3 zur Abrechnung benötigt wird, und signiert es durch

$$D_P(H(CT)).$$

(13) $P$ erzeugt zufällig einen DES-Schlüssel $K_4$. Der Schlüssel und die Kreditkarteninformation $KI$ des Kunden werden durch $E'_P(K_4, KI)$ chiffriert und das signierte capture token durch $E_{K_4}(CT, D_P(H(CT)))$ verschlüsselt.

(14) $P$ übermittelt dem Händler $M$ die Nachricht

$$(E'_M(K_3, KIM), E_{K_3}(AR, D_P(H(AR))), E'_P(K_4, KI), E_{K_4}(CT, D_P(H(CT))), Z_{CA}(E_P)).$$

(15) $M$ überprüft das Zertifikat $Z_{CA}(E_P)$ mit Hilfe der Vertrauenskette bis zur Wurzel.

(16) Durch Berechnung von

$$D'_M(E'_M(K_3, KIM))$$

erhält $M$ den Schlüssel $K_3$ sowie seine Kontoinformation. Mit Hilfe von $K_3$ bestimmt $M$ die Authorisierungs-Antwort

$$(AR, D_P(H(AR))).$$

(17) $M$ überprüft die Signatur von $P$ durch Berechnung und Vergleich von

$$E_P(D_P(H(AR))) \text{ und } H(AR).$$

(18) $M$ speichert die signierte Antwort sowie das verschlüsselte *capture token* $E_{K_4}(CT, D_P(H(CT)))$ und den digitalen Umschlag $E'_P(K_4, KI)$ zur späteren Verwendung (siehe auch Protokoll 6.3.3).

(19) Der Händler $M$ erfüllt den Auftrag des Kunden.

Als letztes geben wir an, wie der Händler mit seiner Bank über das entsprechende *payment gateway* zu einem späteren Zeitpunkt abrechnet (*payment capture*). Dabei verwendet er das *capture token*. Er kann dabei eine, wie unten dargestellt, oder auch mehrere Zahlungen abrechnen.

**Protokoll 6.3.3**
(payment capture, Abrechnung)

(1) Der Händler $M$ erzeugt eine Anfrage (*capture request*) $CR$, die unter anderem den Endbetrag und das Transaktionskennzeichen $TI$ aus $OI$ enthält.

(2) $M$ erzeugt zufällig einen DES-Schlüssel $K_5$. Der Schlüssel wird durch $E'_P(K_5)$ chiffriert und die signierte Anfrage durch $E_{K_5}(CR, D_M(H(CR)))$ verschlüsselt.

(3) $M$ übermittelt dem *payment gateway* $P$ seiner Bank die Nachricht

$$(E'_P(K_5), E_{K_5}(CR, D_M(H(CR))), E'_P(K_4, KI), E_{K_4}(CT, D_P(H(CT))), \\ Z_{CA}(E_M), Z_{CA}(E'_M)),$$

wobei die dritte und vierte Komponente das verschlüsselte *capture token* aus Schritt (18) von Protokoll 6.3.2 ist.

(4) $P$ überprüft die Zertifikate $Z_{CA}(E_M)$ und $Z_{CA}(E'_M)$ mit Hilfe der Vertrauenskette bis zur Wurzel.

(5) $P$ bestimmt den symmetrischen Schlüssel $K_5 = D'_P(E'_P(K_5))$ und kann dann

$$(CR, D_M(H(CR)))$$

bestimmen.

(6) $P$ überprüft die Signatur von $M$ durch Berechnung und Vergleich von

$$E_M(D_M(H(CR))) \text{ und } H(CR).$$

(7) $P$ bestimmt den symmetrischen Schlüssel $K_4$ aus $D'_P(E'_P(K_4, KI))$ und kann dann das *capture token*

$$(CT, D_P(H(CT)))$$

bestimmen.

(8) $P$ überprüft, ob $CR$ und $CT$ übereinstimmen (unter anderem dasselbe Transaktionskennzeichen $TI$ enthalten).

(9) $P$ sendet $CT$ über ein Bankennetzwerk an die Bank des Kunden (*clearing request*).

(10) $P$ erzeugt eine für den Händler bestimmte Antwort (*capture response message* $CRM$) und signiert sie durch

$$D_P(H(CRM)).$$

(11) $P$ erzeugt zufällig einen DES-Schlüssel $K_6$. Der Schlüssel wird durch $E'_M(K_6)$ chiffriert und die signierte Nachricht $CRM$ durch $E_{K_6}(CRM, D_P(H(CRM)))$ verschlüsselt.

(12) $P$ übermittelt dem Händler die Nachricht

$$(E'_M(K_6), E_{K_6}(CRM, D_P(H(CRM))), Z_{CA}(E_P)).$$

(13) Der Händler $M$ überprüft das Zertifikat $Z_{CA}(E_P)$ mit Hilfe der Vertrauenskette bis zur Wurzel.

(14) $M$ berechnet den symmetrischen Schlüssel $K_6 = D'_M(E'_M(K_6))$ und kann dann die signierte *capture response message*

$$(CRM, D_P(H(CRM)))$$

bestimmen.

(15) $M$ überprüft die Signatur von $P$ durch Berechnung und Vergleich von

$$E_P(D_P(H(CRM))) \text{ und } H(CRM).$$

(16) $M$ sichert $CRM$, die er zum Ausgleich von Zahlungen mit seiner Bank benutzen kann.

Neben den hier durch die Protokolle angegebenen Transaktionen gibt es noch weitere Transaktionen in SET. Besonders wichtig sind natürlich die Protokolle, um ein Zertifikat von einer Zertifizierungsstelle zu erhalten. Als weitere Transaktionen nennen wir *credit*, wodurch ein Händler dem Konto eines Kunden einen Kredit gewähren kann, wenn z.B. Waren zurückgegeben werden oder während des Transports beschädigt wurden. Das Protokoll kann nur durch den Händler initiiert werden. Alle Verhandlungen zwischen dem Händler und dem Kunden, die zu diesem Kredit führen, finden außerhalb von SET statt. Zur Korrektur von Anfragen aus den vorhergehenden Protokollen gibt es die Transaktionen *authorization reversal*, *capture rerversal* und auch *credit reversal*.

Die Sicherheit der kritischen Daten ist in SET durch die Verwendung des RSA-Verfahrens mit Schlüsseln von 1024 Bits Länge gewährleistet. SET wurde für den weltweiten Einsatz entwickelt und beruht auf bewährter Kreditkartenzahlung. Für kleine Zahlungen ist es ungeeignet. Auch die Anonymität der Kunden ist nicht gegeben.

Weitere Zahlungssysteme im Internet werden in [88] angesprochen.

# 7 Zero-Knowledge-Protokolle

## 7.1 Komplexitätstheorie

Dieser Abschnitt erklärt die wichtigsten Begriffe der Komplexitätstheorie. In der Komplexitätstheorie werden Probleme unter anderem bezüglich der (Rechen-)Zeit oder des (Speicher-)Platzes klassifiziert, die oder den ein Algorithmus für die ungünstigsten Fälle des Problems zur Lösung benötigt. So gibt es z. B. Probleme mit mindestens exponentiellem Zeitbedarf bzgl. der Länge der Darstellung des Problems (d. h. Zeitbedarf $\Omega(2^n)$, wobei $n$ die Länge der Darstellung des Problems ist). Wir wollen hier die Klassen $P$ und $NP$ betrachten. Es ist

$P :=$ Klasse aller Entscheidungsprobleme, die durch deterministische Turingmaschinen in polynomialer Zeit lösbar sind.

Dabei hat ein Entscheidungsproblem als Lösung für jeden speziellen Fall (jede Eingabe) des Problems die möglichen Antworten „ja“ oder „nein“. Der Zeitbedarf der Turingmaschine für die ungünstigsten Fälle des Problems, also ihre Zeitkomplexität, ist durch ein Polynom in der Länge der Eingabe nach oben beschränkt. Eine solche Turingmaschine hält immer und liefert für jeden Fall des Problems die Antwort „ja“ oder „nein“. Weiter ist

$NP :=$ Klasse aller Entscheidungsprobleme, die durch nichtdeterministische Turingmaschinen in polynomialer Zeit lösbar sind.

Eine nichtdeterministische Turingmaschine hat nach jedem Schritt in nichtdeterministischer Weise mehrere Zwischenergebnisse (als Zustände) zur Verfügung. Sie kann diejenigen Zwischenergebnisse „erraten“, die sie zu einer Lösung des Problems führen. Sie hält für jeden Fall, der die Antwort „ja“ liefert, in polynomialer Zeit. Der Zeitbedarf richtet sich nach der kürzesten möglichen Rechnung, die zu dieser Lösung führt. In anderen Fällen muß die Turingmaschine nicht halten.

Eine Turingmaschine ist ein formales Modell für den Algorithmusbegriff. Sie kann durch jeden realen Universalrechner ersetzt werden. Offenbar gilt $P \subset NP$. Weiter kann ein nichtdeterministischer Algorithmus für ein Problem aus $NP$ mit Hilfe eines deterministischen Algorithmus mit exponentiellem Zeitbedarf simuliert werden. Dieser deterministische Algorithmus ist im allgemeinen sehr ineffizient und kaum handhabbar. Wir betrachten das *Knapsack-Problem (Rucksackproblem)*:

**Beispiel 7.1.1** Gegeben sei $A = \{a_1, \ldots, a_n\}$ mit $a_i \in \mathbb{N},\ i = 1, 2, \ldots, n$ und $S \in \mathbb{N}$. Gesucht ist ein $A' \subset A$ mit

$$\sum_{a' \in A'} a' = S \ .$$

Jedes der $a_i$ repräsentiert einen Gegenstand mit dem Volumen $a_i$. Es stellt sich die Frage, ob ein Rucksack des Volumens $S$ ohne Platz zu verschenken gefüllt werden kann. Da für jedes $A' \subset A$ die Überprüfung

$$\sum_{a' \in A'} a' = S \ .$$

in polynomialer Zeit durchgeführt werden kann, gehört das Problem in die Klasse $NP$. Ein deterministischer Algorithmus für dieses Problem kann alle $2^n$ Möglichkeiten $A' \subset A$ durchprobieren. Seine Zeitkomplexität beträgt $\Omega(2^n)$. $\square$

Die Probleme aus $P$ sind für vernünftig große Eingaben in vernünftiger Zeit lösbar. Sie werden auch als *handhabbar (tractable)* bezeichnet. Probleme aus $NP - P$ werden als *unhandhabbar (untractable)* oder *schwer (hart, hard)* bezeichnet.

Die Frage, ob $P = NP$ oder $P \neq NP$ gilt, ist offen. Allgemein wird angenommen, daß $P \neq NP$ ist. Diese Annahme wird durch die Existenz von mehr als 1000 $NP$-vollständigen Problemen gestützt. Ein Problem $\Pi$ (oder formaler eine entsprechende Sprache) heißt *$NP$-vollständig*, wenn es zur Klasse $NP$ gehört und sich jedes andere Problem $\Pi'$ aus $NP$ in polynomialer Zeit auf $\Pi$ reduzieren läßt, d.h., wenn ein Polynomialzeitalgorithmus $f$ existiert, der jede Eingabe $x$ von $\Pi'$ in eine Eingabe $f(x)$ von $\Pi$ transformiert, so daß gilt:

$$x \text{ liefert die Antwort „ja“} \iff f(x) \text{ liefert die Antwort „ja“.}$$

Wenn für ein einziges dieser Probleme nachgewiesen wird (siehe [13]), daß es sich in polynomialer Zeit durch einen deterministischen Algorithmus lösen läßt, so gilt dies für alle Probleme aus $NP$. Dann würde $P = NP$ gelten. Ein $NP$-vollständiges Problem ist also mindestens genauso schwer, wie jedes andere Problem aus $NP$.

Das erste $NP$-vollständige Problem wurde 1971 von *Cook* [13] angegeben. Dabei handelt es sich um das Erfüllbarkeitsproblem für Boolesche Ausdrücke in konjunktiver Normalform.

Um zu zeigen, daß ein Problem $\Pi$ $NP$-vollständig ist, ist es ausreichend zu zeigen, daß ein bekanntes $NP$-vollständiges Problem in polynomialer Zeit auf das Problem $\Pi$ zurückgeführt werden kann. Das oben erwähnte Knapsack-Problem ist $NP$-vollständig.

Weitere Informationen über $NP$-vollständige Probleme kann man dem Buch von *Garey* und *Johnson* [29] entnehmen.

## 7.2 Zero-Knowledge-Beweise

Neue kryptographische Protokolle basieren auf sogenannten Zero-Knowledge-Beweisen, die von *Goldwasser*, *Micali* und *Rackoff* eingeführt wurden [32]. Durch einen Zero-Knowledge-Beweis kann Alice Bob davon überzeugen, daß sie über ein bestimmtes geheimes Wissen verfügt, ohne dieses Wissen preiszugeben. Man kann z.B. bei Veröffentlichung einer geigneten großen Zahl $n$ und einer weiteren Zahl $e$ als öffentlicher Schlüssel im RSA-Verfahren durch erfolgreiche Benutzung des Verfahrens nachweisen, daß man die Primfaktorzerlegung von $n$ kennt, ohne die Primfaktoren zu verraten. Viele Zero-Knowledge-Verfahren beruhen auf der Schwierigkeit, quadratische Kongruenzen modulo $n$ zu lösen, wobei an $n = pq$ dieselben Forderungen wie im RSA-Verfahren gestellt werden.

Wir zeigen zunächst, daß das Finden einer Primfaktorzerlegung $n = pq$ ebenso schwierig ist wie das Lösen der quadratischen Kongruenz

$$x^2 \equiv_n w$$

für eine beliebige Zahl $w$. Kennt man die Primfaktorzerlegung von $n$, so ist es einfach, eine Lösung der quadratischen Kongruenz anzugeben, also eine Quadratwurzel von $w$ im Restklassenring modulo $n$ zu berechnen. Dies erfolgt mit Algorithmus 5.2.1 und dem Chinesischen Restesatz (Satz 3.14), wie in Abschnitt 5.2 gezeigt wurde, und ist besonders einfach,

wenn $4|(p+1)$ und $4|(q+1)$ gilt (siehe Seite 76). Wir nehmen nun umgekehrt an, daß man zumindest eine Lösung einer quadratischen Kongruenz modulo $n$ berechnen kann. Dann wählen wir eine beliebige Zahl $y < n$, berechnen $w = y^2 \bmod n$ und lösen die quadratische Kongruenz

$$x^2 \equiv_n w.$$

Abgesehen von dem Fall, daß $\text{ggt}(y,n) = p$ oder $q$ ist, gilt $\text{ggt}(y,n) = 1$. Für diesen Fall hat die Kongruenz, wie in Abschnitt 5.2 gezeigt wurde, vier verschiedene Lösungen im Intervall $[0, n-1]$, nämlich $x$, $n-x$, $y$ und $n-y$. Da $y$ die zufällig gewählte Zahl war, sind die beiden Lösungen $y$ und $n-y$ bekannt. Mit der Wahrscheinlichkeit $\frac{1}{2}$ erhält man jedoch eine der Lösungen $x$ oder $n-x$. Durch eventuelle Wiederholung des Verfahrens erhält man sehr schnell mit großer Wahrscheinlichkeit eine dieser beiden Lösungen. Damit kann man über die Beziehung

$$\text{ggt}(x+y,n) = p \text{ oder } q \text{ (siehe Seite 76)}$$

einen Primfaktor von $n$ und damit auch den zweiten Primfaktor berechnen.

Das folgende Protokoll kann, wie schon erwähnt, zur Verbesserung von Protokoll 5.2.1 verwendet werden. Man beachte, daß $n = pq$ gilt.

**Protokoll 7.2.1**
Alice möchte Bob beweisen, daß sie ein $s$ mit $s^2 \equiv_n w$ kennt, ohne daß Bob $s$ erfährt. Beide einigen sich auf ein $k$ (je nach gewünschter Sicherheit).
**for** $i := 1$ **to** $k$ **do**

(1) Alice wählt eine Zufallszahl $r$, $1 \leq r < n$, und sendet

$$v = r^2 \bmod n$$

an Bob.

(2) Bob wählt ein zufälliges Bit $b$ und sendet es an Alice.

(3) Alice sendet

$$u = rs^b \bmod n$$

an Bob.

(4) Bob prüft, ob $u^2 \equiv_n vw^b$ gilt.

**end**. □

In Schritt (3) sendet Alice, falls $b = 0$ ist, die Quadratwurzel $r$ von $v$ an Bob und, falls $b = 1$ ist, die Quadratwurzel $rs$ von $vw$. Eine Quadratwurzel von $w \bmod n$ erfährt Bob auf diese Weise nicht. Wird der obige Dialog jedoch oft genug wiederholt, dann kann Bob sich davon überzeugen, daß Alice eine solche Quadratwurzel kennt. Dies sehen wir wie folgt ein. Wir nehmen zunächst an, daß Alice keine Quadratwurzel kennt. Wüßte sie schon in Schritt (1), welches Bit $b$ Bob in Schritt (2) wählt, so würde sie im Fall $b = 1$ bereits in Schritt (1) für Schritt (3) einfach eine beliebige Zahl $u$ wählen, dann die in $x$ lineare Kongruenz

$$u^2 \equiv_n xw$$

lösen (siehe Satz 3.12 bzw. Satz 3.13 mit Satz 3.14) und schließlich in Schritt (1) $x$ statt $v$ an Bob senden. Bob könnte in Schritt (4) keinen Betrug entdecken. Im Fall $b = 0$ würde sie in Schritt (1) $r^2$ und in Schritt (3) einfach $r$ senden. Wieder wäre ein Betrug nicht festzustellen.

Nun wird jedoch das Bit $b$ von Bob zufällig gewählt. Falls Alice keine Quadratwurzel kennt, dann kann sie nur mit Wahrscheinlichkeit $\frac{1}{2}$ ein geeignetes $u$ angeben. Vermutet sie nämlich $b = 0$, dann wählt sie in Schritt (1) $v = r^2 \bmod n$. Erhält sie anschließend das Bit $b = 1$, dann muß sie in (3) ein $u$ mit $u^2 \equiv_n r^2 w$ nennen. Dazu muß sie diese quadratische Kongruenz lösen können oder ein $s$ mit $s^2 \equiv_n w$ kennen, was jedoch nicht der Fall ist. Vermutet sie dagegen $b = 1$ und erhält $b = 0$, so hat sie bereits in Schritt (1) ein $x$ wie oben angegeben gewählt. Dann muß sie ein $u'$ mit $u'^2 \equiv_n x$ berechnen. Das Lösen einer solchen quadratischen Kongruenz ist jedoch ebenso schwierig wie die Primfaktorzerlegung von $n$, also praktisch undurchführbar. Wird der Dialog $k$-mal wiederholt, so ist die Wahrscheinlichkeit, daß Alice Bob betrügen kann, gleich $\frac{1}{2^k}$.

Anstatt den Dialog $k$-mal zu wiederholen, kann Alice auch parallel für $k$ verschiedene Zahlen $w_j$ nachweisen, daß sie eine Quadratwurzel modulo $n$ kennt. Davon wird bei dem im folgenden beschriebenen *Fiat-Shamir-Verfahren* [22] Gebrauch gemacht. Dieses Verfahren kann zur interaktiven Identifikation benutzt werden, also z.B. zur gegenseitigen Identifikation von Rechnern oder zur Benutzeridentifikation mit leistungsfähigen Chipkarten.

Wir nehmen an, daß ein Vertrauenszentrum (Trust Center), z.B. ein Kreditkartenunternehmen, eine Bank oder ein Rechenzentrum, Chipkarten herausgibt. Mit Hilfe dieses Zentrums können sich die Benutzer einem Rechnersystem gegenüber identifizieren. Dazu wählt das Vertrauenszentrum zunächst einen Modulus $n$, der das Produkt zweier geheimer, nur ihm bekannten Primzahlen $p$ und $q$ ist, und eine Pseudozufallsfunktion $f$, die eine beliebige Zeichenkette auf eine ganze Zahl im Intervall $[0, n-1]$ abbildet. Dann veröffentlicht es $n$ und $f$. Die Benutzer arbeiten alle mit diesem Modulus $n$ und dieser Pseudozufallsfunktion $f$.

Wenn ein Benutzer eine Chipkarte beantragt, die ihm Zugang zu dem System verschaffen soll, überprüft das Vertrauenszentrum die Angaben des Benutzers und erstellt einen Identifikationsstring $I$, der die relevanten Informationen über den Benutzer (z.B. Name, Anschrift, Benutzernummer, Zugriffsrechte) enthält. Dann berechnet das Zentrum Zahlen $w_j = f(I, j) \in \mathbb{N}$ (für kleine $j$) und wählt $k$ paarweise verschiedene dieser $j$ aus, für die $w_j$ ein quadratischer Rest modulo $n$ ist. Aufgrund der Kenntnis der Primfaktoren $p$ und $q$ können die $w_j$ leicht auf diese Eigenschaft hin untersucht werden (Satz 5.1.4 mit Satz 5.1.5). Zu diesen Zahlen berechnet das Zentrum die Inversen $w_j^{-1} \bmod n$ (siehe Algorithmus 3.3) sowie deren kleinsten Quadratwurzeln $s_j \bmod n$, also

$$s_j^2 \equiv_n w_j^{-1}.$$

Abschließend erstellt das Vertrauenszentrum eine Chipkarte, die den Identifikationsstring $I$, die $k$ natürlichen Zahlen $s_j$ und deren Indizes $j$ enthält. Im folgenden wird zur Vereinfachung angenommen, daß diese Indizes $j = 1, \ldots, k$ sind. Nach Aushändigung der Chipkarte an den Benutzer sind auch schon alle Tätigkeiten des Vertrauenszentrums abgeschlossen. Auf das Zentrum könnte ganz verzichtet werden, wenn wie beim RSA-Verfahren jeder Benutzer seinen eigenen Modulus $n$ benutzen würde. Aber offensichtlich ist es wesentlich effizienter, wenn z.B. ein Geldautomat nur den einen Modulus des Kreditunternehmens und nicht alle Moduli der einzelnen Kreditkartenbenutzer kennen muß.

Wir nehmen also an, daß das Identifizierungsgerät (einer elektronischen Kasse, eines Geldautomaten oder Rechnersystems) nur den Modulus $n$ und die Funktion $f$ kennt. Damit kann es mit Hilfe des folgenden Protokolls überprüfen, ob eine eingeführte Chipkarte die Zahlen $s_1, \ldots, s_k$ kennt, ohne die Zahlen selbst zu erfahren. Im Protokoll wird das Identifizierungsgerät durch Bob dargestellt.

**Protokoll 7.2.2**

(1) Alice sendet Bob den Identifikationsstring $I$.

(2) Bob berechnet $w_j = f(I, j)$ für $j = 1, 2 \ldots, k$.
**for** $i := 1$ **to** $t$ **do**

(3) Alice wählt eine Zufallszahl $r_i$, $1 \leq r_i < n$, und sendet

$$v_i = r_i^2 \bmod n$$

an Bob.

(4) Bob wählt $k$ Zufallbits $b_{i1}, \ldots, b_{ik}$ und sendet Alice den Vektor $(b_{i1}, \ldots, b_{ik})$.

(5) Alice sendet

$$u_i = r_i \cdot \prod_{\substack{j \\ b_{i,j}=1}} s_j \bmod n$$

an Bob.

(6) Bob überprüft die Gültigkeit von

$$v_i = u_i^2 \cdot \prod_{\substack{j \\ b_{i,j}=1}} w_j \bmod n.$$

**end**. □

Sind Alice und Bob dem Protokoll korrekt gefolgt, so berechnet Bob in Schritt (6)

$$u_i^2 \cdot \prod_{\substack{j \\ b_{i,j}=1}} w_j \bmod n = r_i^2 \cdot \prod_{\substack{j \\ b_{i,j}=1}} s_j^2 w_j \bmod n = r_i^2 \bmod n = v_i \bmod n.$$

Das Vertrauenszentrum wird nur zur Ausstellung der Chipkarten gebraucht, anschließend kann sich Alice jedem anderen gegenüber (z.B. Bob) ausweisen. Bob akzeptiert Alice Identität nur dann, wenn alle $t$ Protokolldurchläufe erfolgreich waren. Die Wahrscheinlichkeit, daß Alice Bob betrügen kann, ist $\frac{1}{2^{kt}}$. Ein Wert von $\frac{1}{2^{20}}$ wird bei den meisten Anwendungen als ausreichend angesehen. Dazu genügt es, $k = 5$ und $t = 4$ zu wählen. Damit brauchen sowohl Alice als auch Bob durchschnittlich $\frac{t(k+2)}{2} = 14$ Multiplikationen, um ein solches Identifikationsprotokoll $t$-mal zu durchlaufen. Zum Vergleich geben wir an, daß eine digitale Signatur mit dem RSA-Verfahren 768 Multiplikationen benötigt.

Weder ein Dritter, der die Kommunikation zwischen Alice und Bob abhört, noch Bob selbst erhalten Informationen, die es ihm erlauben würden, sich später als Alice auszugeben, da das Protokoll jedesmal mit anderen Zufallszahlen abläuft. Das Protokoll kann jedoch nicht als Nachweis dienen, daß tatsächlich eine solche Kommunikation zwischen Alice und Bob stattgefunden hat. Bob hat nämlich die Möglichkeit, ein solches Protokoll vorzutäuschen, wenn er den Identifikationsstring $I$ von Alice erfahren hat. Bereits bei dem zuerst angeführten Protokoll 7.2.1 ist angegeben, daß Alice in Schritt (1) und (3) geeignete Zahlen $v$ und $u$ angeben kann, ohne eine Quadratwurzel von $w \bmod n$ zu kennen, wenn sie vorher weiß, welches Bit $b$ Bob in Schritt (2) auswählen wird. Auch hier könnte Bob also solche Zahlen $v$ und $u$ in Abhängigkeit vom Bit $b$ berechnen und damit vortäuschen, daß ein Dialog zur Identifikation mit Alice stattgefunden hat. Man kann sagen, daß Bob sich selbst gegenüber sich als Alice ausgeben kann. Man spricht in diesem Fall von einem *Identifikationsschema*. *Fiat* und *Shamir* unterscheiden wie auch einige andere Autoren zwischen drei Sicherheitsstufen:

(1) *Authentifikationsschema:* Alice kann sich Bob gegenüber als Alice ausweisen, aber niemand sonst kann sich Bob gegenüber als Alice ausgeben (Beispiel: Paßwort-Verfahren).
(2) *Identifikationsschema:* Alice kann sich Bob gegenüber als Alice ausweisen, aber Bob kann sich keinem anderen gegenüber als Alice ausgeben (Beispiel: Protokoll 7.2.2).
(3) *Signaturschema:* Alice kann sich Bob gegenüber als Alice ausweisen, aber Bob kann sich nicht einmal sich selbst gegenüber als Alice ausweisen (Beispiel: digitale Signatur mit RSA-Verfahren, Bob kennt nicht $D_A$).

In [22] ist auch ein Signaturverfahren angegeben, das zwar auf dem obigen Protokoll 7.2.2 aufbaut, aber kein Zero-Knowledge-Verfahren ist. Es gibt inzwischen einige Modifikationen und Verbesserungen des Fiat-Shamir-Verfahrens, die den Speicherbedarf oder die Zahl der Multiplikationen reduzieren (siehe z.B. [36], [57], [67]).

Im folgenden wollen wir einige allgemeine Ideen angeben, die für Zero-Knowledge-Beweise benutzt werden können, insbesondere auch für Probleme in $NP$. Die Hauptidee der Konstruktion besteht in der von abschließbaren Schachteln (lockable boxes), in die Peter (prover), der den Beweis führt, entsprechende Informationen einschließt. Vera (verifier), die von der Richtigkeit des Beweises überzeugt werden soll, kann die Schachteln nicht öffnen, da Peter den Schlüssel besitzt. Peter ist jedoch dem Inhalt der Schachteln gegenüber verpflichtet, d.h., er kann durch Öffnen der Schachteln ihren Inhalt nicht ändern. Vera kann Peter beim Öffnen beobachten.

Die Schachteln können mit Hilfe des RSA-Verfahrens oder mit diskreten Logarithmen konstruiert werden, wobei man annimmt, daß Faktorisierung oder Berechnung diskreter Logarithmen schwierige Probleme sind, also praktisch undurchführbar sind. Wir werden darauf noch eingehen.

Wir betrachten das Problem der *3-Färbbarkeit eines Graphen G*. Eine *3-Färbung* von $G$ besteht in einer Färbung der Knoten von $G$ mit den Farben $B$ (blau), $R$ (rot) und $W$ (weiß), so daß Knoten, die durch eine Kante verbunden sind, nicht dieselbe Farbe haben. 3-Färbbarkeit ist ein $NP$-vollständiges Problem. Das folgende Protokoll stammt von [34] und ist auch in [83] dargestellt.

**Protokoll 7.2.3**
Peter möchte Vera überzeugen, daß er eine 3-Färbung eines Graphen $G$ mit $t$ Knoten $1, \ldots, t$ kennt:
**for** s:=1 **to** k **do**

(1) Peter bereitet folgende abgeschlossene Schachteln vor: $B_i$, $B_i^C$, $1 \leq i \leq 3t$, $B_{i,j}$, $1 \leq i < j \leq 3t$. Jede der Schachteln $B_i$ enthält genau einen Knoten und jede der Schachteln $B_i^C$ genau eine Farbe, und zwar so, daß für jedes Paar $(i_1, c)$ mit $1 \leq i_1 \leq t$ und $c \in \{B, R, W\}$ ein $i$ existiert mit $i_1$ aus $B_i$ und $c$ aus $B_i^C$. Dabei treten die Paare $(i_1, c)$ in den Paaren $(B_i, B_i^C)$ von Schachteln in zufälliger Ordnung auf.
Jede der Schachteln $B_{i,j}$ enthält entweder 0 oder 1. Sie enthält 1 genau dann, wenn *beide* der folgenden Bedingungen erfüllt sind:
   (a) Ist $i_1$ aus $B_i$ und $j_1$ aus $B_j$, so existiert eine Kante zwischen $i_1$ und $j_1$ in $G$.
   (b) Die Farbe, die in Peters 3-Färbung dabei dem Knoten $i_1$ bzw. $j_1$ zugeordnet ist, erscheint in $B_i^C$ bzw. $B_j^C$.

Die Schachteln $B_i$, $B_i^C$ und $B_{i,j}$ werden als Knoten-, Farb- bzw. Kantenschachteln bezeichnet.
Peter übermittelt Vera diese Schachteln.

(2) Vera wählt zufällig ein Bit $b$ und teilt es Peter mit.
(3) (a) Ist $b = 1$, so öffnet Peter alle Knoten- und Kantenschachteln.
    (b) Ist $b = 0$, so öffnet Peter alle Farbschachteln und alle diejenigen Kantenschachteln $B_{i,j}$, für die die Farben aus $B_i^C$ und $B_j^C$ übereinstimmen.
(4) Vera überprüft, ob im Fall
    (a) sie eine Kopie von $G$ sowie $2t$ isolierte Punkte erhalten hat (ist dies nicht der Fall, so bricht sie das Protokoll ab),
    (b) alle geöffneten $\frac{3t(t-1)}{2}$ Kantenschachteln 0 enthalten, ob die Farben entsprechend (3)(b) übereinstimmen und ob jede Farbe $t$-mal in den Farbschachteln vorkommt (ist dies nicht der Fall, so bricht sie das Protokoll ab).

**end**. □

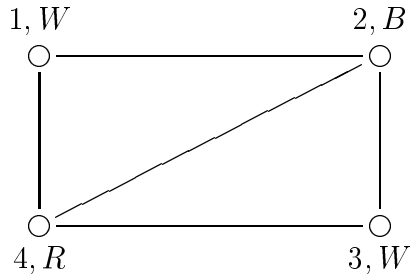
Zunächst überlegen wir uns, warum in (4)(b) genau $\frac{3t(t-1)}{2}$ Kantenschachteln geöffnet werden. Jeweils $t$ Farbschachteln $B_i^C$ müssen jeweils dieselbe Farbe enthalten. Die Indizes dieser Schachteln bestimmen die jeweils $\binom{t}{2} = \frac{t(t-1)}{2}$ zu öffnenden Kantenschachteln (jeder der $t$ Knoten einer festen Farbe muß mit jedem anderen Knoten derselben Farbe verbunden werden). Insgesamt müssen also $\frac{3t(t-1)}{2}$ Kantenschachteln geöffnet werden. Offensichtlich können die zugehörigen Knoten der nicht geöffneten Kantenschachteln jeweils nur verschieden gefärbt sein.

Die folgenden Überlegungen sind nun leicht einzusehen. Die Schachteln müssen in jeder Runde des Protokolls neu konstruiert werden, da sonst Vera in der ersten Runde $b = 1$ wählt und damit eine Kopie von $G$ erhält und ihr in der zweiten Runde bei Wahl von $b = 0$ die zugehörige Färbung bekannt wird. Werden die Schachteln immer neu konstruiert, so empfängt sie im Fall (b) nach den vorstehenden Überlegungen die Information, daß zwei benachbarte Knoten (durch einige der nicht geöffneten Kantenschachteln repräsentiert) verschieden gefärbt sind und folglich Peters 3-Färbung korrekt sein muß. Wenn Peter eine 3-Färbung von $G$ kennt, gehen die Überprüfungen des Protokolls immer positiv aus. Er kann nur auf zwei verschiedene Arten betrügen.

(1) Peter verschließt in den Schachteln nicht eine Beschreibung von $G$, sondern die eines anderen Graphen mit derselben Anzahl von Knoten $t$, dessen 3-Färbung er kennt. Falls Vera $b = 1$ gewählt hat, stellt sie diesen Betrug in Schritt (4)(a) sofort fest. Wählt sie dagegen das Bit $b = 0$, so kann sie diesen Täuschungsversuch nicht erkennen.
(2) Ein anderer Betrugsversuch ist möglich, wenn Peter zwar den richtigen Graphen wählt, aber eine falsche 3-Färbung benutzt, indem er eine falsche Anzahl von Farben wählt oder aber benachbarte Knoten gleich färbt. Die Wahl des richtigen Graphen bedeutet, daß Peter nach den Überlegungen in unmittelbarem Anschluß an das Protokoll diejenigen Knoten, die zu den Kantenschachteln $B_{i,j}$ mit $1 \in B_{i,j}$ gehören, also die Knoten von $G$, verschieden färben muß, wenn Veras Überprüfung in (4)(b) positiv ausgehen soll. Eine falsche Färbung erkennt sie also bei Wahl von $b = 0$ sofort.

Die Wahrscheinlichkeit, daß Peter ohne Entdeckung eine falsche 3-Färbung des Graphen behaupten kann, ist somit $\leq \frac{1}{2^k}$.

**Beispiel 7.2.1** Es sei $t = 4$. Der gegebene Graph $G$ ist unten dargestellt. Dabei sind die Knoten bereits mit ihren Namen und mit der Peter bekannten 3-Färbung versehen.



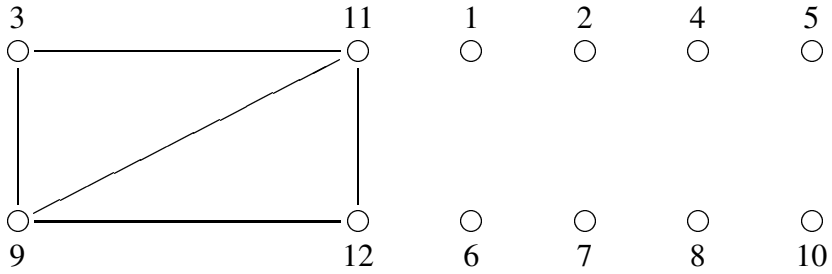

Peter bereitet die folgenden Knoten- und Farbschachteln vor:

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $B_i$ | 2 | 1 | 1 | 4 | 2 | 4 | 3 | 3 | 4 | 1 | 2 | 3 |
| $B_i^C$ | $W$ | $R$ | $W$ | $B$ | $R$ | $W$ | $B$ | $R$ | $R$ | $B$ | $B$ | $W$ |

Peter bereitet $\binom{3\cdot 4}{2} = 66$ Kantenschachteln $B_{i,j}$ vor, und zwar so, daß die 5 Schachteln

$$B_{3,11}, B_{3,9}, B_{9,11}, B_{9,12} \text{ und } B_{11,12}$$

den Wert 1 enthalten und alle anderen 61 Schachteln den Wert 0. Wird gemäß (a) vorgegangen, so erhält Vera den Graphen




Hierbei benutzen wir die Indizes der Knotenschachteln als Marken. Den geöffneten Knotenschachteln kann Vera entnehmen, daß diese Marken 3, 11, 12, 9 in dieser Reihenfolge den Knoten 1, 2, 3, 4 entsprechen. Somit erhält Vera den originalen Graphen ohne Färbung der Knoten. Außerdem erhält sie 8 isolierte Knoten.

Wird gemäß (b) vorgegangen, so werden die folgenden $\frac{3\cdot 4(4-1)}{2} = 18$ Kantenschachteln

$$B_{1,3}, B_{1,6}, B_{1,12}, B_{2,5}, B_{2,8}, B_{2,9}, B_{3,6}, B_{3,12}, B_{4,7},$$

$$B_{4,10}, B_{4,11}, B_{5,8}, B_{5,9}, B_{6,12}, B_{7,10}, B_{7,11}, B_{8,9}, B_{10,11}$$

geöffnet. Alle diese Schachteln enthalten 0. □

Jetzt wollen wir auf die Konstruktion der abschließbaren Schachteln eingehen. Wir können das RSA-Verfahren verwenden (Verschließen mit dem öffentlichen Schlüssel $E_P$), wobei jedoch das Problem besteht, z.B. bei der Schachtel $B_{i,j}$ die einzigen Inhalte 0 oder 1 durch Zusatzinformation so zu chiffrieren, daß nicht nur zwei verschiedene chiffrierte Werte der Schachtelinhalte entstehen. Eine andere Methode beruht auf der Annahme, daß die

Berechnung diskreter Logarithmen modulo $p$ praktisch undurchführbar ist (siehe auch Seite 62).

Ohne Beschränkung der Allgemeinheit kann dabei angenommen werden, daß jede Schachtel nur ein Bit enthält. Wenn sie eigentlich mehr Information enthalten soll, kann sie durch mehrere Schachteln ersetzt werden, die gleichzeitig zu öffnen sind.

Zunächst werde eine große Primzahl $p$ und eine primitive Wurzel modulo $p$ (siehe Definition 5.3.1) gewählt, also eine Zahl $g \in \mathbb{Z}_p^* = \mathbb{Z}_p - \{0\}$, die ein erzeugendes Element von $\mathbb{Z}_p^*$ ist. Das bedeutet, daß für alle $a \in \mathbb{Z}_p$, $a \neq 0$, ein $x \in \mathbb{Z}_p$ mit

$$g^x \equiv_p a$$

existiert (siehe Satz 5.3.2). $x$ ist der diskrete Logarithmus von $a$ zur Basis $g$. Es ist bekannt, daß genau $\varphi(p-1)$ erzeugende Elemente von $\mathbb{Z}_p - \{0\}$ existieren (siehe Satz 5.3.3, $\varphi$ Eulersche Funktion). Peter und Vera einigen sich auf die Zahlen $p$ und $g$. Sie werden veröffentlicht. Allgemeiner können $p$ und $g$ von allen Parteien genutzt werden, die ein solches Zero-Knowledge-Protokoll verwenden möchten.

Zu Beginn wählt Vera ein zufälliges $r$, $1 < r < p-1$, und teilt es Peter mit. Peter kann $e$ mit $g^e \equiv_p r$ nicht berechnen. Um ein Bit $b$ in einer Schachtel zu verschließen, wählt Peter zufällig und geheim eine Zahl $y$ und übermittelt Vera die „Schachtel“

$$x = r^b \cdot g^y \bmod p.$$

Offenbar ist jedes Element $a \in \mathbb{Z}_p$, $a \neq 0$, für ein geeignetes $y$ sowohl von der Form $g^y \bmod p$ als auch von der Form $r \cdot g^y \bmod p$, da $\mathbb{Z}_p - \{0\}$ eine multiplikative Gruppe ist. Folglich wird durch $x$ nichts über $b$ verraten. Wenn Peter für Vera die Schachtel öffnen will, dann teilt er ihr den Schlüssel $y$ mit. Vera kann dann überprüfen, ob $x = g^y \bmod p$ oder $x = r \cdot g^y \bmod p$ gilt.

Peter ist auch auf das Bit $b$ verpflichtet. Er kann die Schachtel nicht wunschweise als 0 oder 1 öffnen. Anderenfalls könnte Peter nämlich zwei Zahlen $y$ und $y'$ wählen mit

$$g^y \bmod p = r \cdot g^{y'} \bmod p.$$

Später könnte er, je nach gewünschtem Resultat 0 oder 1, $y$ oder $y'$ als Schlüssel nennen. Aus dieser Gleichung folgt

$$r = g^{y-y'} \bmod p.$$

Damit kann Peter den diskreten Logarithmus von $r$ berechnen. Wir haben jedoch angenommen, daß dies praktisch undurchführbar ist.

Allgemein kann man nun formulieren:

**Satz 7.2.1** Für jede „ja“-Lösung eines Problems aus $NP$ gibt es einen Zero-Knowledge-Beweis, falls sichere Kryptosysteme existieren.

*Beweis.* Es sei $x$ ein Fall des betrachteten Problems. Nach Abschnitt 7.1 existiert ein deterministischer Algorithmus $f$, der $x$ auf einen Fall $f(x)$ des $NP$-vollständigen 3-Färbbarkeitsproblems in polynomialer Zeit reduziert. $f(x)$ ist also ein Graph. Dabei gilt

$$x \text{ „ja“-Fall} \iff f(x) \text{ 3-färbbar.}$$

Wir können annehmen, daß Vera und Peter beide den Algorithmus $f$ kennen. Somit stehen beiden $x$ und $f(x)$ zur Verfügung. Wenn Peter eine Lösung für $x$ kennt, dann kennt er auch eine für den Graphen $f(x)$. Ein Zero-Knowledge-Beweis für $f(x)$ ist daher aufgrund der Äquivalenz auch einer für $x$. □

Ein Zero-Knowledge-Beweis für das folgende Problem kann gemäß Satz 7.2.1 auf einen entsprechenden Beweis für das 3-Färbbarkeit-Problem zurückgeführt werden. Wir wollen jedoch einen direkten Beweis angeben (siehe [34]), der nicht auf Zahlentheorie beruht, sondern unmittelbar auf der Graphentheorie. Es sei $G = (V, E)$ ein ungerichteter Graph mit der Knotenmenge $V$ und der Kantenmenge $E$. Zwei Graphen $G_0 = (V_0, E_0)$ und $G_1 = (V_1, E_1)$ heißen isomorph, wenn eine Bijektion $\pi : V_0 \to V_1$ existiert mit

$$(u, v) \in E_0 \iff (\pi(u), \pi(v)) \in E_1.$$

Das *Graphisomorphismus-Problem* besteht darin zu entscheiden, ob zwei vorgegebene Graphen $G_0$ und $G_1$ isomorph sind. Es ist bekannt, daß dieses Problem aus $NP$ stammt, jedoch ist ungeklärt, ob es $NP$-vollständig ist.

**Protokoll 7.2.4**
Peter möchte Vera davon überzeugen, daß er einen Isomorphismus zwischen zwei Graphen $G_0$ und $G_1$ kennt.
**for** i:=1 **to** k **do**

(1) Peter berechnet einen Zufallsgraphen $H$, der isomorph zu $G_1$ ist, und teilt ihn Vera mit.
(2) Vera wählt ein zufälliges Bit $b$ und sendet es an Peter.
(3) Peter teilt Vera einen Isomorphismus zwischen $G_b$ und $H$ mit.
(4) Vera überprüft diese Isomorphie.

**end**. □

Wenn Peter tatsächlich einen Isomorphismus zwischen den beiden Graphen $G_0$ und $G_1$ kennt, dann kennt er eine Bijektion $\pi : V_0 \to V_1$ mit

$$(u, v) \in E_0 \iff (\pi(u), \pi(v)) \in E_1.$$

Außerdem hat er in Schritt (1) eine Bijektion $\mu : V_1 \to V_H$ erzeugt und damit einen Isomorphismus zwischen den Graphen $G_1$ und $H = (V_H, E_H)$ mit

$$(u, v) \in E_1 \iff (\mu(u), \mu(v)) \in E_H$$

hergestellt. Wenn $b = 1$ in Schritt (2) gewählt wird, so teilt Peter in Schritt (3) Vera einfach $\mu : V_1 \to V_H$ mit. Ist $b = 0$, so übermittelt er ihr $\mu \circ \pi : V_0 \to V_H$, wobei $\circ$ die Verknüpfung der Abbildungen bedeutet. Kennt Peter dagegen keinen Isomorphismus zwischen $G_0$ und $G_1$, dann kann er nur für $b = 1$ den Schritt (3) richtig ausführen, also insgesamt nur mit Wahrscheinlichkeit $\frac{1}{2}$. Die Wahrscheinlichkeit, daß Peter Vera betrügen kann, ist nach $k$ Durchläufen somit $\frac{1}{2^k}$. Durch die Tatsache, daß bei jedem Durchlauf ein anderer zufälliger Graph $H$ gewählt wird, erfährt Vera von Peter keine zusätzlichen Informationen über den Isomorphismus. Dies setzt voraus, daß Peter zusätzlich darauf achtet, daß $H$ verschieden von $G_0$ und $G_1$ ist.

Die Sicherheit von Protokoll 7.2.4 beruht auf der Tatsache, daß bis jetzt kein polynomialer Algorithmus für das Graph-Isomorphismus-Problem bekannt ist.

# 8 Knapsacksysteme

## 8.1 Merkle-Hellman-Knapsacks

In den folgenden Abschnitten werden mehrere Public-Key-Kryptosysteme, die auf dem $NP$-vollständigen Knapsack-Problem beruhen, angegeben. *Merkle* und *Hellman* [56] schlugen ein Schema vor, das auf der Schwierigkeit beruht, das folgende *0-1-Knapsack-Problem* zu lösen.

> Gegeben sei $C \in \mathbb{N}$, $A = (a_1, a_2, \ldots, a_n) \in \mathbb{N}^n$. Gesucht ist ein binärer Vektor $M = (m_1, m_2, \ldots, m_n)$ mit
>
> $$C = AM = \sum_{i=1}^{n} a_i m_i$$
>
> (siehe auch Beispiel 7.1.1).

**Beispiel 8.1.1** Es sei $n = 5, C = 14$ und $A = (1, 10, 5, 22, 3)$. Dann ist $M = (1, 1, 0, 0, 1)$ eine Lösung von $C = AM$. □

Der beste bisher bekannte Algorithmus,um einen beliebigen Fall der Größe $n$ zu lösen, stammt von *Schroeppel* und *Shamir* [87]. Er hat einen Zeitbedarf von $O(2^{\frac{n}{2}})$ und einen Platzbedarf von $O(2^{\frac{n}{4}})$. Spezielle Knapsack-Probleme können jedoch in linearer Zeit gelöst werden. Wir betrachten hier *einfache Knapsacks*. Bei ihnen sind die Elemente $a_i$ superwachsend, d. h.

$$a_i > \sum_{j=1}^{i-1} a_j \text{ für } i = 2, 3, \ldots, n.$$

**Satz 8.1.1** Es sei ein einfaches 0-1-Knapsack-Problem mit super-wachsenden $a_i$, $i = 1, 2, \ldots, n$, gegeben, und das Problem sei lösbar. Dann folgt für $i = 1, 2, \ldots, n$

$$m_i = 1 \Longleftrightarrow \left( C - \sum_{j=i+1}^{n} m_j a_j \right) \geq a_i.$$

*Beweis.* Aus $m_i = 1$ folgt

$$C - \sum_{j=i+1}^{n} m_j a_j = \sum_{j=1}^{i} m_j a_j = a_i + \sum_{j=1}^{i-1} a_j m_j \geq a_i.$$

Umgekehrt gelte

$$C - \sum_{j=i+1}^{n} m_j a_j \geq a_i.$$

Dann ist

$$\sum_{j=1}^{i} m_j a_j \geq a_i.$$

Wir erhalten

$$\begin{aligned} m_i a_i &= m_i a_i + a_i - a_i > m_i a_i + \sum_{j=1}^{i-1} a_j - a_i \text{ (da die } a_i \text{ super-wachsend sind)} \\ &\geq m_i a_i + \sum_{j=1}^{i-1} m_j a_j - a_i = \sum_{j=1}^{i} m_j a_j - a_i \geq 0 \end{aligned}$$

Es folgt $m_i a_i > 0$ und damit $m_i = 1$. □

Aus diesem Satz folgt die Lösung des einfachen Knapsack-Problems durch den folgenden Linearzeitalgorithmus.

**Algorithmus 8.1.1**
snap($C, A$); {„simple knapsack solution“}
**for** $i := n$ **downto** 1 **do**
  **if** $C \geq a_i$ **then** $m_i := 1$ **else** $m_i := 0$ **end**;
  $C := C - a_i * m_i$;
  **end**;
**if** $C = 0$ **then** snap:=$M = (m_1, \ldots, m_n)$ **else** write(„keine Lösung“) **end**; □

**Beispiel 8.1.2** Der Vektor $A$ aus Beispiel 8.1.1 wird zu $A' = (1, 3, 5, 10, 22)$ umgestellt. Damit wird $A'$ ein einfacher Knapsack-Vektor. snap(14, $A'$) liefert (1, 1, 0, 1, 0). □

Ein einfacher Knapsack soll in einen *Trapdoor-Knapsack*, der ohne zusätzliche Information schwer zu lösen ist, umgewandelt werden. Dazu wird ein einfacher Knapsack-Vektor $A' = (a'_1, a'_2, \ldots, a'_n)$ gewählt. Dieser erlaubt eine einfache Lösung des Problems $C' = A'M$. Weiter wird ein $u \in \mathbb{N}$ mit

$$u \geq 2a'_n > \sum_{i=1}^{n} a'_i$$

und ein $w \in \mathbb{N}$ mit

$$\text{ggt}(u, w) = 1 \text{ und } w < u$$

gewählt. Anschließend wird $w^{-1} = \text{inv}(w, u)$ berechnet und ein schwerer Knapsack-Vektor durch

$$A = (wA') \bmod u \quad (\text{d. h. } a_i = (wa'_i) \bmod u \text{ für } i = 1, 2, \ldots, n)$$

gebildet. Die Lösung von $C = AM$ ist jetzt schwierig. Dies schwierige Problem kann jedoch unter Kenntnis der *Trapdoor-Information* $w^{-1}$ und $u$ in das folgende leichte Problem transformiert werden:

$$\begin{aligned} C' &= (w^{-1}C) \bmod u \\ &= (w^{-1}AM) \bmod u \\ &= (w^{-1}(wA')M) \bmod u \\ &= (A'M) \bmod u \\ &= A'M. \end{aligned}$$

Für die letzte Gleichung benötigen wir die Ungleichung $u \geq 2a'_n > \sum_{i=1}^{n} a'_i$. Falls eine Lösung des schwierigen Problems existiert, dann zeigt diese Rechnung, daß sie die Lösung des einfachen Knapsack-Problems ist. Wegen der Äquivalenz aus Satz 8.1.1 ergibt sich außerdem, daß höchstens eine Lösung für den einfachen Knapsack und damit auch höchstens eine für den schweren Knapsack existiert.

Bei der Anwendung des Trapdoor-Knapsacks für Public-Key-Kryptosysteme wird für den öffentlichen Schlüssel ein schwerer Knapsack-Vektor $A$ und für den geheimen Schlüssel $(A', u, w^{-1})$ ein entsprechender einfacher Knapsack-Vektor $A'$, zusammen mit den Trapdoor-Informationen $u$ und $w^{-1}$, verwendet. $A'$ kann aus $A$, $u$ und $w^{-1}$ durch

$$A' = (w^{-1}A) \bmod u$$

berechnet werden. Die Chiffrierung $E_A$ erfolgt mit dem öffentlichen Schlüssel $A$, die Dechiffrierung $D_A$ mit dem geheimen Schlüssel $(A', u, w^{-1})$.

Bei der *Chiffrierung* wird der Klartext in Blöcke $M$ von $n$ Bits Länge aufgeteilt. Die Chiffrierung erfolgt durch

$$C = E_A(M) = AM.$$

Die *Dechiffrierung* erhalten wir aus

$$D_A(C) = \text{snap}((w^{-1}C) \bmod u, A') = M.$$

**Beispiel 8.1.3** Gegeben seien $A' = (1, 3, 5, 10), u = 20$ und $w = 7$ mit ggt$(7, 20) = 1$. Daraus ergibt sich $w^{-1} = 3$, und es folgt

$$A = ((7 \cdot 1) \bmod 20, (7 \cdot 3) \bmod 20, (7 \cdot 5) \bmod 20, (7 \cdot 10) \bmod 20) = (7, 1, 15, 10).$$

Der Klartext $M = 13$, binär durch (1, 1, 0, 1) dargestellt, wird als

$$C = E_A(M) = 7 + 1 + 10 = 18$$

chiffriert. Die Dechiffrierung ergibt wieder

$$\begin{aligned} D_A(C) &= D_A(18) = \text{snap}((3 \cdot 18) \bmod 20, A') = \text{snap}(14, A') \\ &= (1, 1, 0, 1) = 13. \quad \square \end{aligned}$$

*Merkle* und *Hellman* schlugen ursprünglich vor, $n = 100$ oder größer zu wählen. Mit dem oben erwähnten Algorithmus von *Schroeppel* und *Shamir*, der einen Zeitbedarf von $O(2^{\frac{n}{2}})$ hat, ergeben sich für $n = 100$ schon $2^{50} \cong 10^{15}$ Instruktionen. Ein Prozessor mit $8,64 \cdot 10^{10}$ Instruktionen pro Tag findet die Lösung in etwa 11574 Tagen, 1000 Prozessoren benötigen dazu etwa 12 Tage. Wird $n = 200$ gewählt, so ergeben sich $2^{100} \cong 10^{30}$ Instruktionen. Somit wird der Algorithmus unhandhabbar.

Das allgemeine Knapsack-Problem ist $NP$-vollständig. Wenn auch das Trapdoor-Knapsack-Problem $NP$-vollständig wäre, wäre es ein hervorragendes Kryptosystem. Allerdings wurde dieses System bereits 1982 von *Shamir* [93] gebrochen, d. h. in polynomialer Zeit gelöst. Dies werden wir weiter unten betrachten.

Zuvor stellen wir noch einige Überlegungen zu den Größen des Trapdoor-Knapsacks an. Es werde $n = 200$ und $a'_i \in [(2^{i-1} - 1)2^{200} + 1, (2^{i-1})2^{200}]$ gewählt. Damit gibt es $2^{200}$

Wahlen für $a'_i$ ($i > 0$). Nach den obigen Forderungen soll $u \geq 2a'_{200}$ gelten. Dies ist erfüllt, wenn ein $u \in [2^{401}+1, 2^{402}-1]$ und ein $w \in [2, u-2]$ gewählt wird. Dann ist $a_i \in [1, 2^{402}-1]$, und die Klartextnachricht $M$ aus 200 Bits hat eine Chiffretextnachricht $C = AM$ mit einer 410-Bit-Darstellung. Dies liegt daran, daß bei der Addition von 200 Werten zu je 402 Bits ein Ergebnis entstehen kann, das höchstens 200-mal so groß ist wie die größte Zahl mit 402 Bits. Die Multiplikation mit 200 bedeutet, daß $\lceil \log_2 200 \rceil = 8$ zusätzliche binäre Stellen entstehen.

Dieses Verfahren könnte für die Geheimhaltung (wäre es nicht durch *Shamirs* Algorithmus gebrochen worden!), jedoch nicht für die Authentizität benutzt werden, denn offensichtlich kann nicht jede 410-Bit-Nachricht genau einer 200-Bit-Nachricht entsprechen, d.h., $D_A$ ist nicht auf jede beliebige 410-Bit-Nachricht anwendbar.

Für $n = 200$ benötigt der öffentliche Vektor $A$ ungefähr $n \cdot 400$ Bits $= 80$ kBits Speicherplatz. Dagegen benötigt das RSA-Schema für den öffentlichen Schlüssel nur 1 kBit.

Für den Zeitbedarf sieht die Situation umgekehrt aus. Für $n = 200$ werden für die Chiffrierung 200 Additionen und für die Dechiffrierung 200-mal je eine Subtraktion, eine Multiplikation und eine modulare Multiplikation benötigt. Dagegen benötigt das RSA-Schema sowohl für die Chiffrierung als auch für die Dechiffrierung etwa 1000 Multiplikationen in modularer Arithmetik in beiden Fällen.

## 8.2 Brechen des Merkle-Hellman-Knapsacks

Wir kommen nun zu *Shamirs Algorithmus zum Brechen des Merkle-Hellman-Knapsacks* [93]. Wir stehen vor der folgenden kryptoanalytischen Aufgabe. Ein Knapsack-Vektor $A = (a_1, a_2, \ldots, a_n)$ ist bekannt, der als öffentlicher Schlüssel benutzt wird. Wir wissen, daß $A$ von irgendeinem super-wachsenden Knapsack-Vektor $A'$ durch modulare Multiplikationen, wie oben beschrieben, erzeugt wurde, also

$$A = (wA') \bmod u$$

gilt. $A'$, $w$ und $u$ sind uns unbekannt. Wir möchten die Werte $u$ und $w^{-1} \bmod u$ und damit $A' = (w^{-1}A) \bmod u$ bestimmen. *Shamirs* kryptoanalytischer Angriff benutzt nur den Chiffrierschlüssel. Das System kann gebrochen werden, bevor wesentlicher Chiffretext gesendet wird. Der Algorithmus arbeitet in polynomialer Zeit bzgl. der Größe des Knapsacks $A$. Zu dieser Größe trägt die Zahl $n$ sowie die Größe der Komponenten $a_i$ bei.

Es ist nicht nötig, die ursprünglichen Werte $w^{-1}$ und $u$ zu finden. Jeder Multiplikator $w'$ und jeder Modulus $u'$ genügt, der $A' = (w'A) \bmod u'$, $A'$ super-wachsend, und $u' \geq 2a'_n$ bzgl. $A'$ erfüllt. Ein solches Paar $(w', u')$ bildet ein Trapdoor-Paar. Mit Hilfe des gesendeten Textes $C = AM$ kann dann

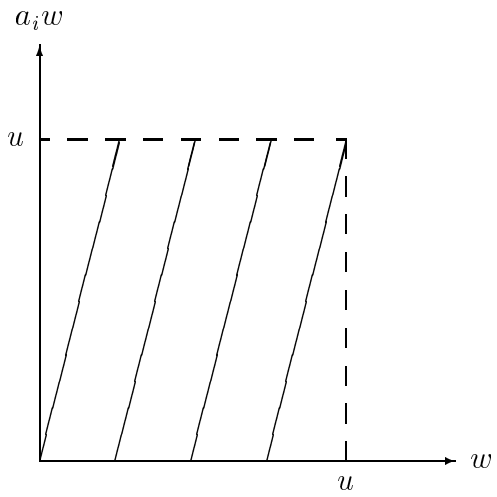
$$C' = (w'C) \bmod u' = (w'AM) \bmod u' = A'M$$

gebildet werden. Falls die Lösung $M$ des schwierigen Problems existiert, löst $M$ auch den einfachen Knapsack $A'$. Dabei existiert aber nach Satz 8.1.1 jeweils höchstens eine Lösung für den einfachen Knapsack und damit auch für den schweren Knapsack. Die Dechiffrierung

$$D_A(C) = \text{snap}((w'C) \bmod u', A')$$

erfolgt mit einem beliebigen Trapdoor-Paar $(w', u')$. Die Existenz von mindestens einem solchen Paar ist dadurch gesichert, daß der Kryptosystem-Designer ein solches benutzt

hat. Um ein Trapdoor-Paar $(w, u)$ zu finden, werden alle Graphen $(a_i w) \bmod u$ für alle $i = 1, 2, \ldots, n$ betrachtet. Die $(a_i w) \bmod u$ müssen dann die Komponenten eines super-wachsenden Knapsacks sein, und es muß $u \geq 2(a_i w \bmod u)$ gelten. Wir fassen nun $w$ als Variable über $\mathbb{R}$ (statt über $\mathbb{N}$) auf und erhalten für $i = 1, 2, \ldots, n$ Sägezahnkurven, die durch das folgende Bild gegeben werden:




Nach den Überlegungen von Seite 126 muß $w < u$ gelten. Dabei ist $u$ jedoch unbekannt. Im Vergleich zu $u$ sind die $(a_i w) \bmod u$ des gesuchten $w$ wegen $u \geq 2(a_i w \bmod u)$ klein. Insbesondere ist $a_1 w \bmod u$ als erste Komponente eines super-wachsenden Knapsacks sehr klein. Der zugehörige Wert $w$ des Trapdoor-Paares $(w, u)$ liegt in der Nähe eines $a_i$-Minimums für alle $i$. Anders ausgedrückt liegt also für alle $i$ der Wert $(a_i w) \bmod u$ in der Nähe eines Minimums der entsprechenden Sägezahnkurve. Es existiert somit ein kleines Intervall der $w$-Achse, das ein $a_i$-Minimum für alle $i$ enthält. In diesem Intervall befindet sich $w$.

Diese Ideen sollen durch Ungleichungen ausgedrückt werden. Eine erste Schwierigkeit besteht darin, daß $u$ nicht bekannt ist. Dieses Problem wird dadurch gelöst, daß das obige Bild durch Ersetzen von $u$ durch 1 reduziert wird, d. h., alle Längen werden durch $u$ geteilt. Die relative Lage der Häufungspunkte der $a_i$-Minima wird dadurch nicht beeinflußt. Wenn z. B. das $a_i$-Minimum für alle $i$ in der Nähe des 7. $a_3$-Minimums vor der Reduzierung vorkommt, so auch nach der Reduzierung.

Der Algorithmus zum Finden eines Trapdoor-Paares besteht aus zwei Teilen:

(1) Zunächst werden Kandidaten $p \in \mathbb{N}$ dafür gefunden, daß das $p$-te $a_1$-Minimum der gesuchte Häufungspunkt ist.

(2) Diese Kandidaten werden nacheinander getestet. Da für das bestehende Trapdoor--Paar des Designers ein Häufungspunkt existiert, terminiert der Algorithmus.

Da der erste Teil des Algorithmus, im Vergleich zur Größe des Problems, zu viele Kanditaten $p$ liefern könnte, werden Vorsichtsmaßnahmen getroffen. Das wird durch die Festlegung einer maximalen Anzahl $k$ von Kandidaten erreicht. Falls der Algorithmus $k + 1$ Kandidaten produziert, terminiert er mit Mißerfolg. Weiterhin müssen im ersten Teil nicht alle möglichen Komponenten $a_2, a_3, \ldots, a_n$ betrachtet werden, sondern es wird eine maximale Anzahl $l < n$ von Komponenten $a_2, a_3, \ldots, a_l$ festgelegt. Somit produziert der erste

Teil Kandidaten $p$ dafür, daß das $p$-te $a_1$-Minimum in der Nähe eines $a_i$-Minimus für alle $i = 2, 3, \ldots, l$ liegt. Die Werte $i > l$ werden nicht betrachtet, so daß der erste Teil ganz falsche Werte liefern kann. Der zweite Teil des Algorithmus überprüft jedoch alle $a_i, i = 2, 3, \ldots, n$. Ein Kandidat $p$ entfällt sofort, wenn für ein $i$ kein $a_i$-Minimum in der Nähe des $p$-ten $a_1$-Minimums liegt.

Wir beschreiben nun den ersten Teil des Algorithmus. Das erste $a_1$-Minimum ergibt sich beim kleinsten $w$ mit $a_1 w = 1$, also bei $w = \frac{1}{a_1}$. Da gleiche Abstände von Minimum zu Minimum bestehen, ist die $w$-Koordinate des $p$-ten $a_1$-Minimums gleich $\frac{p}{a_1}$. Daß ein $a_2$-Minimum in der Nähe des $p$-ten $a_1$-Minimums liegt, kann durch

$$-\varepsilon < \frac{p}{a_1} - \frac{q_2}{a_2} < \varepsilon, 1 \leq p \leq a_1 - 1, 1 \leq q_2 \leq a_2 - 1$$

ausgedrückt werden. Für $p = a_1$ bzw. $q_2 = a_2$ wird der rechte Rand der Sägezahnkurve erreicht. Durch die Multiplikation mit $a_1 a_2$ wird

$$-\delta < a_2 p - a_1 q_2 < \delta,\ 1 \leq p \leq a_1 - 1,\ 1 \leq q_2 \leq a_2 - 1$$

erreicht. Nun wird für jede Komponente $a_2, a_3, \ldots, a_l$ eine Ungleichung dieser Form aufgestellt, also

$$\begin{aligned} -\delta < a_2 p - a_1 q_2 < \delta, &\quad 1 \leq p \leq a_1 - 1, \quad 1 \leq q_2 \leq a_2 - 1 \\ &\vdots \\ -\delta < a_l p - a_1 q_l < \delta, &\quad 1 \leq p \leq a_1 - 1, \quad 1 \leq q_l \leq a_l - 1 \end{aligned}$$

Die $a_i$ sind bekannt. Gesucht sind $p$ und die $q_i$. Dies ist ein *Problem des linearen Programmierens*. 1981 wurde von *Lenstra* [52] gezeigt, daß dieses Problem in polynomialer Zeit gelöst werden kann. Der Zeitbedarf beträgt $O(n^\alpha)$, wobei der Exponent $\alpha$ noch nicht genau bekannt ist. Der dadurch bestimmte erste Teil des Algorithmus liefert also alle diejenigen $p$, für die Zahlen $q_2, q_3, \ldots, q_l$ existieren, die die Ungleichungen erfüllen.

Wir kommen nun zum zweiten Teil des Algorithmus. Die Zahlen $p$ werden getestet, bis der Algorithmus Erfolg hat. Zunächst werden dafür alle Unstetigkeitspunkte der $n$ Kurven des Intervalls $[\frac{p}{a_1}, \frac{p+1}{a_1}]$ bestimmt und in aufsteigender Ordnung sortiert. Wenn $x_t$ und $x_{t+1}$ zwei aufeinanderfolgende Punkte dieser Liste von Punkten sind, dann ist in $[x_t, x_{t+1}]$ jede $a_i$-Kurve ein Liniensegment, das die Darstellung $a_i w - c_i^t$ hat. Dabei ist $c_i^t$ eine von $i$, $t$ und $p$ abhängige Konstante. Mit Hilfe des linearen Programmierens wird nun eine Lösung des folgenden Systems linearer Ungleichungen in $w$ gesucht:

$$\begin{aligned} x_t &\leq w \leq x_{t+1} \\ \sum_{i=1}^{n} (a_i w - c_i^t) &< 1 \\ a_i w - c_i^t &> \sum_{j=1}^{i-1} (a_j w - c_j^t),\ i = 1, 2, \ldots, n. \end{aligned}$$

Dabei erhält man als Lösung ein Teilintervall von $[x_t, x_{t+1}]$, das auch leer sein kann. Eine notwendige und hinreichende Bedingung dafür, daß ein $(w, u)$ ein Trapdoor-Paar darstellt, ist, daß $\frac{w}{u}$ in einem dieser Teilintervalle von $[x_t, x_{t+1}]$ enthalten ist. Wenn nämlich $(w, u)$ ein

Trapdoor-Paar ist, dann ist $\frac{w}{u}$ wegen der Reduktion in einem $[x_t, x_{t+1}]$ enthalten. Damit ist die erste Ungleichung erfüllt. Weiter muß

$$u \geq 2(a_n w \bmod u) > \sum_{i=1}^{n} a_i \cdot w \bmod u = \sum_{i=1}^{n} (a_i w - c_i^t u),$$

für den super-wachsenden Knapsack mit den Komponenten $a_i w \bmod u$ (siehe Ausführungen vor dem Bild auf Seite 129) gelten. Division durch $u$ liefert

$$1 > \sum_{i=1}^{n} \left( a_i \frac{w}{u} - c_i^t \right).$$

Damit ist also die zweite Ungleichung für $\frac{w}{u}$ erfüllt. Die $(a_i w) \bmod u$ sind super-wachsend, woraus sich

$$a_i w - c_i^t u > \sum_{j=1}^{i-1} (a_j w - c_j^t u),\ i = 1, 2, \ldots, n,$$

ergibt. Division durch $u$ liefert die Gültigkeit der dritten Ungleichung für $\frac{w}{u}$. Existiert umgekehrt für einen Kandidaten $p \in \mathbb{N}$ und ein zugehöriges Intervall $[x_t, x_{t+1}]$ eine Lösung der Gestalt $w' = \frac{w}{u}$, so ist $(w, u)$ entsprechend den eben durchgeführten Überlegungen ein Trapdoor-Paar.

Der zweite Teil des Algorithmus läuft nun so ab, daß nacheinander die Paare $(p, t)$ untersucht werden, wobei $p$ ein Kandidat des ersten Teils und $t$ ein Index eines Punktes der sortierten Liste ist. Die Untersuchungen werden so lange durchgeführt, bis ein nichtleeres Intervall gefunden worden ist. Man wähle irgendeine rationale Zahl $\frac{w}{u}$ in diesem Intervall. Damit ist ein Trapdoor-Paar $(w, u)$ bestimmt. Falls das Trapdoor-Paar des Designers in einem nichtleeren Intervall enthalten ist oder ein anderes Trapdoor-Paar gefunden ist, terminiert der Algorithmus mit Erfolg. Es könnten aber auch, wie oben beschrieben, zu viele Kandidaten $p$ gefunden werden. Dann terminiert der Algorithmus mit Mißerfolg. Die Wahrscheinlichkeit für einen solchen Mißerfolg kann abgeschätzt werden. Es seien $k$, $l$ und $\delta$ die oben gewählten Zahlen. Es gelte $l \geq 3$ und $\delta < \sqrt{\frac{a_1}{2}}$. Dann ist die Wahrscheinlichkeit eines Mißerfolges

$$\leq \left( \frac{2}{k} \right)^{l-1}.$$

Zum Beispiel erhält man für $k = 100$ und $l = 3$, wenn also höchstens 100 Kandidaten gefunden werden und nur 2 Ungleichungen im ersten Teil des Algorithmus verwendet werden dürfen, bereits eine recht geringe Wahrscheinlichkeit eines Mißerfolges.

Die vorstehenden Überlegungen zeigen die Auseinandersetzung zwischen einem Kryptosystem-Designer und einem Kryptoanalytiker. Weitere Ausführungen zu *Shamirs* Algorithmus können in den Büchern von *Horster* [42], *OConnor* und *Seberry* [65] oder *Salomaa* [83] nachgelesen werden. In [83] wird auch ein Beispiel angegeben, wie man ein Trapdoor-Paar finden kann.

Es gibt andere Möglichkeiten, ein Knapsack-Kryptosystem sicherer zu machen.

## 8.3 Shamir-Knapsacks für Authentizität

*Shamir* [92] hat einen Trapdoor-Knapsack angegeben, der für Authentizität benutzt werden kann. *Shamirs* Schema kann jedoch nicht zur Geheimhaltung dienen. Es beruht auf dem folgendem $NP$-vollständigen Knapsack-Problem:

Es seien $n, M \in \mathbb{N}$ und $A = (a_1, a_2, \ldots, a_{2k}) \in \mathbb{N}^{2k}$. Gesucht wird ein Vektor $C = (c_1, c_2, \ldots, c_{2k})$ mit $M = (CA) \bmod n$, also mit

$$(*) \qquad M = \sum_{j=1}^{2k} c_j a_j \bmod n,$$

wobei $c_j \in \mathbb{N},\ c_j \in [0, \log_2 n],\ j = 1, 2, \ldots, 2k$, gilt.

In einem Shamir-Knapsack-System für Authentizität wird eine zufällige Primzahl $n$ von $k$ Bits gewählt, etwa $k = 100$. Mit dem Knapsack-Vektor $A$ bildet das Paar $(A, n)$ den öffentlichen Schlüssel. Es sei $M \in [0, n-1]$ die Nachricht und $C$ die Signatur von $M$. Der Empfänger des Paares $(M, C)$ kann durch die Berechung von

$$E_A(C) = CA \bmod n = M$$

diese Signatur überprüfen. Wenn das Knapsack-Problem gelöst, also ein $C$ mit $M = CA \bmod n$ gefunden werden kann, ist es möglich, die Unterschrift für eine andere Nachricht zu fälschen.

Der Absender besitzt eine geheime Trapdoor-Information, um die Signatur $C = D_A(M)$ zu erzeugen. Sie besteht aus einer zufälligen binären $k \times 2k$ Matrix $H$. Der Vektor $A$ wird nun so konstruiert, daß er das folgende System von modularen Gleichungen erfüllt. Da $n$ als Primzahl gewählt wurde, erfolgen dabei alle Rechnungen im Körper $\mathbb{Z}_n$.

$$\begin{pmatrix} h_{1,1} & \cdots & h_{1,2k} \\ \vdots & \ddots & \vdots \\ h_{k,1} & \cdots & h_{k,2k} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_{2k} \end{pmatrix} = \begin{pmatrix} 2^0 \\ \vdots \\ 2^{k-1} \end{pmatrix} \bmod n$$

Dies ist äquivalent zu

$$\sum_{j=1}^{2k} h_{ij} a_j = 2^{i-1} \bmod n,\ i = 1, 2, \ldots, k.$$

Da in diesen $k$ Gleichungen $2k$ Unbekannte $a_j$ vorkommen, können $k$ Komponenten von $A$ beliebig gewählt werden. Die übrigen ergeben sich aus der Lösung des Gleichungssystems. Die Rechnungen sind schnell durchzuführen. Falls erforderlich, können die beliebig gewählten Komponenten so abgeändert werden, daß die Rechnungen schnell werden.

Die Signatur $C$ wird für eine Nachricht $M$ und ihr binäres Spiegelbild $\overline{M} = (m_1, m_2, \ldots, m_k)$ durch

$$C = D_A(M) = \overline{M} H,$$

also

$$c_j = \sum_{i=1}^{k} m_i h_{ij}, 1 \le j \le 2k$$

erzeugt. Die Überprüfung der Signatur erfolgt durch

$$\begin{aligned} E_A(C) &= CA \bmod n \\ &= \sum_{j=1}^{2k} c_j a_j \bmod n \\ &= \sum_{j=1}^{2k} \left( \sum_{i=1}^{k} m_i h_{ij} \right) a_j \bmod n \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{k} m_i \left( \sum_{j=1}^{2k} h_{ij} a_j \right) \bmod n \\
&= \sum_{i=1}^{k} (m_i 2^{i-1}) \bmod n = M.
\end{aligned}
$$

**Beispiel 8.3.1** Es sei $k = 3$ und $n = 7$. Das bedeutet, daß Nachrichten im Bereich [0, 6] gesendet werden können. Die Matrix $H$ werde (zufällig) gewählt als

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

Wir setzen $a_1 = 1$, $a_2 = 2$ und $a_3 = 3$ in das Gleichungssystem (modulo 7)

$$
\begin{array}{llllll}
a_1+ & & a_3+ & & & a_6 = 1 \\
& a_2+ & a_3+ & a_4+ & & a_6 = 2 \\
a_1+ & & a_3+ & a_4+ & a_5 & = 4
\end{array}
$$

ein und erhalten

$$
\begin{array}{lll}
a_6 & \equiv_7 -3 & \equiv_7 4 \\
a_4 + a_6 & \equiv_7 -3 & \equiv_7 4 \\
a_4 + a_5 & \equiv_7 0 &
\end{array}.
$$

Die Lösung lautet $a_4 = a_5 = 0$, $a_6 = 4$, und damit ist der Knapsack-Vektor $A = (1, 2, 3, 0, 0, 4)$ bestimmt. Es sei $M = 3_{10} = (011)_2$. Die Chiffrierung von $\overline{M} = (1, 1, 0)$ liefert die Signatur

$$
\begin{aligned}
C &= \overline{M}H = (1, 1, 0) \cdot \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \\
&= (1, 1, 2, 1, 0, 2).
\end{aligned}
$$

Diese Signatur wird durch

$$
\begin{aligned}
(CA) \bmod 7 &= ((1, 1, 2, 1, 0, 2)(1, 2, 3, 0, 0, 4)) \bmod 7 \\
&= (1 + 2 + 6 + 0 + 0 + 8) \bmod 7 = 17 \bmod 7 = 3
\end{aligned}
$$

überprüft. □

Dieses Signatur-Verfahren ist jedoch noch unsicher. Falls genügend Paare $(M, C)$ empfangen werden, kann aus $C = \overline{M}H$ mit den Methoden der linearen Algebra $H$ bestimmt werden (siehe dazu auch Abschnitt 2.5, Hill-Chiffre). Um diesen Lösungsansatz zu verhindern, werden die Nachrichten, bevor sie signiert werden, randomisiert. Dabei wird ein zufälliger Vektor $R = (r_1, r_2, \ldots, r_{2k})$ gewählt und anschließend $M' = (M - RA) \bmod n$ berechnet. $M'$ wird wie oben signiert und liefert $C'$. Die Signatur $C$ von $M$ ist nun $C = C' + R$. $C$ ist eine gültige Signatur, denn es gilt

$$
\begin{aligned}
(CA) \bmod n &= (C' + R)A \bmod n \\
&= (C'A + RA) \bmod n \\
&= (M' + RA) \bmod n \\
&= M.
\end{aligned}
$$

**Beispiel 8.3.2** Die Werte für $k$, $n$, $H$, $A$ und $M$ werden wie in Beispiel 8.3.1 gewählt. Es sei $R = (1,0,0,0,1,1)$. Dann folgt

$$\begin{aligned} M' &= (M - RA) \bmod n \\ &= (3 - (1,0,0,0,1,1)(1,2,3,0,0,4)) \bmod 7 \\ &= (3 - (1+0+0+0+0+4)) \bmod 7 \\ &= -2 \bmod 7 = 5_{10} = 101_2. \end{aligned}$$

Damit erhalten wir

$$\begin{aligned} C' &= \overline{M'}H = (1,0,1) \cdot \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \\ &= (2,0,2,1,1,1). \end{aligned}$$

Dadurch ergibt sich die Signatur

$$C = C' + R = (2,0,2,1,1,1) + (1,0,0,0,1,1) = (3,0,2,1,2,2). \quad \square$$

Die Signatur hängt vom zufälligem Vektor $R$ ab. Daraus folgt, daß es viele Signaturen $C$ geben kann, die die Gleichung $(*)$ auf Seite 132 erfüllen. Die Beispiele 8.3.1 und 8.3.2 liefern für die Nachricht $M = 3$ zwei verschiedene Signaturen. Somit kann dieses Schema nicht für die Geheimhaltung verwendet werden. Die Dechiffrierung erzeugt viele mögliche Werte, so daß $D_A(E_A(M))$ nicht das ursprüngliche $M$ liefern muß. Positiv zu bewerten ist, daß das System schnell ist. Es stellt sich die Frage, ob es so schwierig wie das zugrundeliegende $NP$-vollständige Problem ist.

## 8.4 Ein brechbarer *NP*-vollständiger Knapsack

*Lempel* [51] hat zusammen mit *Even* und *Yacobi* eine Chiffre angegeben, die die merkwürdige Eigenschaft hat, $NP$-vollständig unter einem Angriff mit gewähltem Klartext, aber trotzdem bei genügend viel bekanntem Klartext leicht brechbar zu sein. Die Chiffre benutzt einen geheimen Schlüssel $K = (k_1, k_2, \ldots, k_n)$ von $n$ Bits und einen dem Kryptoanalytiker bekannten Knapsack-Vektor $A = (a_1, a_2, \ldots, a_n)$, wobei $a_i > 0$ für $i = 1, 2, \ldots, n$ gilt.

Die Chiffrierung einer Nachricht $M$ erfolgt dadurch, daß sie in $t$-Bit-Blöcke der Form $M = (m_1, m_2, \ldots, m_t)$ mit

$$t = \left\lceil \log_2 \left( 1 + \sum_{i=1}^{n} a_i \right) \right\rceil$$

zerlegt wird. Der Sender erzeugt einen zufälligen $n$-Bit-Vektor $R = (r_1, r_2, \ldots, r_n)$ und bildet die $t$-Bit-Summe

$$S = A(K \oplus R) = \sum_{i=1}^{n} a_i(k_i \oplus r_i),$$

wobei $\oplus$ das komponentensweise exklusive Oder ist. Der Sender bildet $L = M \oplus S$ und damit $C = (L, R)$. $C$ hat die Länge von $t + n$ Bits. Der legale Empfänger kennt $K$ und $A$ und erhält $R$ direkt aus $C$. Daraus kann $S = A(K \oplus R)$ und damit auch $M = M \oplus S \oplus S = L \oplus S$ berechnet werden. Der Kryptoanalytiker kennt $A$, und wir nehmen an, daß er

genau ein Paar $(M, C)$ kennt. Dann kann ein Angriff mit bekanntem Klartext erfolgen. Der Kryptoanalytiker berechnet $S$ durch

$$L \oplus M = (M \oplus S) \oplus M = M \oplus M \oplus S = S.$$

Zur Bestimmung von $K$ muß er das $NP$-vollständige Knapsack-Problem $S = A(K \oplus R)$ lösen. Dabei sind ihm $S$, $A$ und $R$ bekannt. Falls der Kryptoanalytiker jedoch $n$ Paare $(M_i, C_i) = (M_i, L_i, R_i)$, $i = 1, 2, \ldots, n$, kennt und außerdem die $n$ Vektoren $U_i = (\underbrace{1, \ldots, 1}_{n\text{-mal}}) - 2R_i$ linear unabhängig sind, so folgt

$$\begin{aligned} S_i &= A(K \oplus R_i) \\ &= A(R_i + K \otimes U_i) \text{ ($\otimes$ komponentenweise Multiplikation)} \\ &= AR_i + (U_i \otimes A)K,\ i = 1, 2, \ldots, n, \end{aligned}$$

da $K \oplus R_i = K + R_i - 2(K \otimes R_i) = R_i + K \otimes U_i$ ist und, wie man sich leicht überzeugt, $(U_i \otimes A)K = A(K \otimes U_i)$ gilt. Es sei $T_i = S_i - AR_i$. Dann können die obigen Gleichungen durch

$$\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{pmatrix} \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{pmatrix} \begin{pmatrix} a_1 k_1 \\ a_2 k_2 \\ \vdots \\ a_n k_n \end{pmatrix}$$

ausgedrückt werden. Die $U_1, U_2, \ldots, U_n$ sind linear unabhängige Vektoren, und das System ist nach den $x_i = a_i k_i$, $i = 1, 2, \ldots, n$, auflösbar. Da $a_i > 0$ für $i = 1, 2, \ldots, n$ gilt, folgt

$$k_i = 1 \Longleftrightarrow x_i > 0.$$

Damit kennt der Kryptoanalytiker den geheimen Schlüssel und kann wie der legale Empfänger jede Nachricht dechiffrieren. Man kann zeigen, daß die Wahrscheinlichkeit, daß $N$ Paare $(M_i, C_i)$ eine Teilmenge von $n$ linear unabhängigen $U_i$ enthalten, für $N = n$ nach unten durch etwa $\frac{1}{3}$ beschränkt ist. Mit wachsendem $N$ strebt die Wahrscheinlichkeit schnell gegen 1. Das bedeutet, daß die Chiffre, obwohl sie auf einem schweren Problem beruht, gebrochen werden kann. Die Schwäche dieses Schemas ist auf die lineare Beziehung zwischen Klar- und Chiffretext zurückzuführen.

Weitere Informationen zu Kryptosystemen, die auf Knapsacks beruhen, finden sich in den Büchern von *Horster* [42], *OConnor* und *Seberry* [65] und *Salomaa* [83].

# 9 Ein weiteres Kryptosystem und Ergänzungen

## 9.1 Ein Public-Key-Kryptosystem, das auf formalen Sprachen beruht

Für dieses von *Salomaa* ([84], siehe auch [81], [82], [85]) vorgeschlagene Kryptosystem wird als öffentlicher Chiffrierschlüssel ein T0L-System, das auf einem einfacheren DT0L-System beruht, benutzt. Die Kenntnis von T0L- und DT0L-Systemen ist zum Verständnis des folgenden nicht notwendig. Die Trapdoor-Information ist die Kenntnis, wie man das DT0L-Systems aus dem T0L-System wieder zurückgewinnt. Ohne diese Information muß der Kryptoanalytiker das $NP$-vollständige Wortproblem für T0L-Systeme lösen.

Gegeben seien ein Alphabet $\Sigma$ und zwei Homomorphismen $h_0, h_1 : \Sigma^* \longrightarrow \Sigma^*$ sowie ein $w \in \Sigma^*, w \neq \varepsilon$ (das Axiom des DT0L-Systems $(\Sigma, h_0, h_1, w)$). Es sei $i_0 i_1 \dots i_n \in \{0,1\}^*$ und $x \in \Sigma^*$. Dann definieren wir

$$(x)i_0 i_1 \dots i_n = h_{i_n}(\dots h_{i_1}(h_{i_0}(x)) \dots).$$

Außerdem seien $h_0$ und $h_1$ rückwärts deterministisch im folgenden Sinn. Es gelte

$$(w)i_0 i_1 \dots i_n = (w)j_0 j_1 \dots j_m.$$

Dann muß

$$i_0 i_1 \dots i_n = j_0 j_1 \dots j_m,$$

also auch $n = m$, folgen. Es sei $\Delta$ ein Alphabet, das wesentlich mehr Elemente als $\Sigma$ enthält. Typischerweise enthält $\Sigma$ zwei und $\Delta$ etwa 200 Symbole. Es sei $g : \Delta^* \longrightarrow \Sigma^*$ ein Homomorphismus mit $g(d) \in \Sigma \cup \{\varepsilon\}$ für $d \in \Delta$, so daß $g^{-1}(a) \neq \emptyset$ für alle $a \in \Sigma$ gilt. Es werden zwei Substitutionen $\sigma_0$ und $\sigma_1$ auf $\Delta^*$ durch endliche, nichtleere Mengen

$$(*) \qquad \sigma_i(d) \subset g^{-1}(h_i(g(d))) \text{ für alle } d \in \Delta,\ i = 0, 1,$$

definiert. Wird ein $u \in g^{-1}(w) \subset \Delta^*$ gewählt, so bildet das Quadrupel

$$(\Delta, \sigma_0, \sigma_1, u)$$

den *öffentlichen Chiffrierschlüssel*, wobei $u$ das Axiom des T0L-Systems ist. Der Klartext sei nun ein Binärwort $i_0 i_1 \dots i_n$. Der Chiffretext ist dann ein beliebiges Wort aus

$$\sigma_{i_n}(\dots(\sigma_{i_0}(u)) \dots).$$

Der Klartext kann auch in Blöcke von nicht unbedingt gleicher Länge eingeteilt werden, die jeweils nach dieser Methode chiffriert werden. Das ist insbesondere dann geeignet, wenn der Chiffretext zu lang wird.

Der *geheime Dechiffrierschlüssel* wird durch das Quadrupel $(\Sigma, h_0, h_1, w)$, das ein DT0L-System ist, und $g$ gebildet. Dabei ist $g$ die Trapdoor-Information. Falls $g$ bekannt ist,

kann das DT0L-System aus dem öffentlichen T0L-System berechnet werden. Da $g^{-1}(a) \neq \emptyset$ für alle $a \in \Sigma$ gilt, erhalten wir zunächst

$$\Sigma = g(\Delta) - \{\varepsilon\}.$$

Weiter ist $w = g(u)$. Aus $(*)$ kann man für $a \in \Sigma$ und alle $d \in g^{-1}(a)$ schließlich

$$h_i(a) = h_i(g(d)) = g(\sigma_i(d))$$

für $i = 0, 1$ berechnen.

Wir betrachten die Dechiffrierung. Es sei $x \in \sigma_{i_n}(\ldots(\sigma_{i_0}(u))\ldots)$ der Chiffretext. Dann folgt aufgrund von $(*)$

$$g(x) = h_{i_n}(\ldots(h_{i_0}(\underbrace{g(u)}_{w}))\ldots) = (w)i_0 \ldots i_n.$$

Mit Hilfe des DT0L-Systems wird hieraus der Klartext gewonnen. Notwendig dafür ist, daß $h_0$ und $h_1$ rückwärts deterministisch sind. Dann kann aus $g(x)$ der Klartext $i_0 \ldots i_n$ zurückgewonnen werden. Das wird durch das folgende Beispiel verdeutlicht. Es sei $\Sigma = \{a, b\}$. Die Homomorphismen $h_0$ und $h_1$ seien durch

$$h_0 : a \to ab, b \to b \text{ und } h_1 : a \to a, b \to ba$$

definiert, wobei die Homomorphismen durch entsprechende Produktionen beschrieben sind. Man wähle $w = ab$. Wir betrachten $\Delta = \{c_1, c_2, c_3, c_4, c_5\}$. Für praktische Zwecke ist diese Menge viel zu klein. Die Abbildung $g : \Delta^* \longrightarrow \Sigma^*$ wird durch

$$g(c_2) = g(c_4) = a,\ g(c_1) = b,\ g(c_3) = g(c_5) = \varepsilon$$

definiert. $c_2$ und $c_4$ werden als Abkömmlinge von $a$ und $c_1$ als Abkömmling von $b$ bezeichnet. $c_3$ und $c_5$ sind zusätzliche Symbole. $\sigma_0$ und $\sigma_1$ werden durch

$$\begin{aligned} \sigma_0 : \quad & c_1 \to c_5c_1, \quad c_2 \to c_2c_3c_1, \quad c_2 \to c_4c_5c_1, \quad c_3 \to c_3c_5, \\ & c_3 \to c_5c_3, \quad c_4 \to c_2c_1c_3c_5, \quad c_5 \to c_5c_5c_5, \end{aligned}$$

und

$$\begin{aligned} \sigma_1 : \quad & c_1 \to c_1c_2, \quad c_1 \to c_1c_4, \quad c_2 \to c_4, \quad c_3 \to c_3c_5c_3, \\ & c_3 \to c_5c_5, \quad c_4 \to c_2, \quad c_4 \to c_4, \quad c_5 \to c_5. \end{aligned}$$

definiert. So sind hier z. B. durch die Produktion $a \to ab$ die Produktionen von $\sigma_0$ mit $c_2$ bzw. $c_4$ auf der linken Seite nach $(*)$ als endliche nichtleere Teilmengen von

$$g^{-1}(h_0(g(c_2))) = g^{-1}(h_0(g(c_4))) = g^{-1}(h_0(a)) = g^{-1}(ab)$$

bestimmt worden. Auf der rechten Seite einer Produktion von $\sigma_0$ mit $c_2$ oder $c_4$ als linker Seite muß ein Abkömmling von $a$ von einem Abkömmling von $b$ gefolgt werden. Außerdem dürfen dazwischen beliebig viele zusätzliche Symbole $c_3$ oder $c_5$ eingefügt werden. Es gilt z.B.

$$c_2c_3c_1,\ c_4c_5c_1,\ c_2c_1c_3c_5 \in g^{-1}(ab).$$

Nun kann man z.B., wie es oben für die Tabelle geschehen ist,

$$c_2 \to c_2c_3c_1,\ c_2 \to c_4c_5c_1 \text{ und } c_4 \to c_2c_1c_3c_5$$

wählen.

Da $u \in g^{-1}(w) = g^{-1}(ab)$ ist, wird etwa $u = c_2c_3c_1$ gewählt. Der Schlüssel $(\Delta, \sigma_0, \sigma_1, u)$ kann jetzt veröffentlicht werden. Aus dem gewählten Klartext 010 wird der Chiffretext

$$x \in \sigma_0(\sigma_1(\sigma_0(c_2c_3c_1)))$$

durch die Ableitung

$$\begin{aligned} c_2c_3c_1 &\overset{\sigma_0}{\Longrightarrow} (c_2c_3c_1)(c_3c_5)(c_5c_1) \\ &\overset{\sigma_1}{\Longrightarrow} c_4(c_5c_5)(c_1c_2)(c_3c_5c_3)c_5c_5(c_1c_2) \\ &\overset{\sigma_0}{\Longrightarrow} (c_2c_1c_3c_5)c_5^3c_5^3(c_5c_1)(c_2c_3c_1)(c_3c_5)c_5^3(c_5c_3)c_5^3c_5^3(c_5c_1)(c_4c_5c_1) \\ &= c_2c_1c_3c_5^8c_1c_2c_3c_1c_3c_5^5c_3c_5^7c_1c_4c_5c_1 = x \end{aligned}$$

erzeugt. Auch andere Alternativen wären für $x$ denkbar. Ein Kryptoanalytiker wird bei einem derart kleinen öffentlichen Chiffrierschlüssel aus $x$ noch leicht den Klartext bestimmen können.

Für die Dechiffrierung wird die geheime Trapdoor-Information benutzt. Dann kann $y = g(x) \in \Sigma^* = \{a, b\}^*$ berechnet werden, wobei bekannt ist, daß $y$ von der Form $y = (w)i_0i_1 \dots i_n$ ist. Gesucht ist die Folge $i_0i_1 \dots i_n$. Wir wissen, daß $h_0$ und $h_1$ rückwärts deterministisch sind. Falls $h_i, i = 0$ oder 1, bekannt ist und außerdem $w' = h_i(w'')$ gilt, dann ist $w''$ eindeutig bestimmt. $w''$ kann sofort, bedingt durch die spezielle Gestalt von $h_0$ bzw. $h_1$ in unserem Beispiel, berechnet werden. Falls nämlich $w'$ das Teilwort $a^2$ (bzw. $b^2$) enthält, folgt $i = 1$ (bzw. $i = 0$). Aufgrund seiner Konstruktion aus $w$ hat $w'$ eine Länge von mindestens 2. Falls $w'$ kein Teilwort $a^2$ bzw. $b^2$ enthält, kann es nun aufgrund der Definition von $h_0$ und $h_1$ nur die Gestalt $w' = a(ba)^n$, $n \geq 1$, haben, und zwar mit $w'' = ab^n$, $n \geq 1$, und dem Homomorphismus $h_1$. Aus diesen Überlegungen ergibt sich, daß aus $y$ der Index $i_n$ und $y'$ mit $y' = (x)i_0i_1 \dots i_{n-1}$ zu bestimmen sind. Danach wird diese Prozedur wiederholt. Dies zeigt, daß $h_0$ und $h_1$ rückwärts deterministisch sind. Speziell wird der obige Chiffretext $x$ betrachtet. Wir erhalten

$$y = g(x) = ab^2ab^2ab.$$

Da $b^2$ ein Teilwort von $y$ ist, folgt $i_n = 0$. Für $h_0$ ergibt sich $y' = ababa$ mit $h_0(y') = y$. Da weder $a^2$ noch $b^2$ ein Teilwort von $y'$ ist, ergibt sich

$$i_{n-1} = 1 \text{ und } y'' = ab^2$$

und damit schließlich

$$i_{n-2} = 0 \text{ und } y''' = ab = w.$$

Die Dechiffrierung ist beendet $(n = 2)$. Der ermittelte Klartext lautet 010.

Falls im Klartext dezimale Notation verwendet werden soll, werden 10 Homomorphismen $h_0$ bis $h_9$ und 10 Substitutionen $\sigma_0$ bis $\sigma_9$ benötigt. Ein „nur-Chiffretext-Angriff“ ist mit großer Wahrscheinlichkeit erfolglos, denn es ist unwahrscheinlich, aus dem öffentlichen Schlüssel die geheime Abbildung $g$ zu gewinnen. Der Nachteil dieser Methode liegt darin, daß der Chiffretext sehr lang werden kann. Die Homomorphismen müssen also so gewählt werden, daß dies nicht vorkommt. Andererseits muß dabei immer die Eindeutigkeit der schnellen Dechiffrierung erfüllt sein. Nicht immer ist beides gleichzeitig möglich.

Offensichtlich kann dies Verfahren nicht für die Senderauthentizität verwendet werden.

Man kann zeigen, daß das Finden der Trapdoor-Information $g$ ein $NP$-vollständiges Problem ist (siehe [45]). Andererseits kann aus einem abgefangenen Kryptotext der Länge

$n$ der Klartext durch einen Algorithmus mit Zeitbedarf $k \cdot n^3$ berechnet werden, wobei $k$ eine ziemlich große Konstante ist (siehe [44]). Die Konstruktion des Algorithmus erfolgt mit Hilfe der Automatentheorie.

Einige weitere Ausführungen zu diesem Kryptosystem befinden sich in dem Buch von *Salomaa* [83].

## 9.2 Blockchiffren

Eine Blockchiffre bricht den Klartext $M$ in aufeinanderfolgende Blöcke $M_1$, $M_2, \ldots$ auf und chiffriert jedes $M_i$ mit demselben Schlüssel $K$, d. h.

$$E_K(M) = E_K(M_1)E_K(M_2)\ldots.$$

Jeder Block ist normalerweise mehrere Zeichen lang.

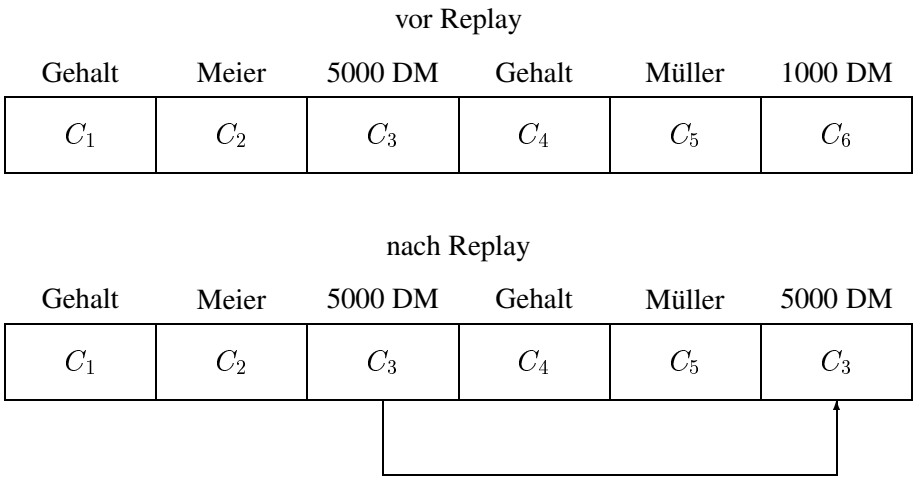
**Beispiel 9.2.1**

| Chiffre | Blockgrößen |
|---|---|
| Transposition mit Periode $d$ | $d$ Buchstaben |
| einfache Substitution | 1 Buchstabe |
| homophone Substitution | 1 Buchstabe |
| Playfair-Chiffre | 2 Buchstaben |
| Hill-Chiffre mit $d \times d$-Matrix | $d$ Buchstaben |
| DES | 64 Bits |
| Exponentiation $\bmod n$ | $\log_2 n$ Bits (Empfehlung: 664 Bits) |
| Knapsack der Länge $n$ | $n$ Bits (Empfehlung: 200 Bits) □ |

Bei Blockchiffren hat ein Übertragungsfehler in einem Block $M_i$ keine Auswirkungen auf die anderen Blöcke. Blockchiffren sind teilweise anfällig für Kryptoanalysis, da gleiche Klartextblöcke auch gleiche Chiffretextblöcke liefern. Dies kann bedeuten, daß ggf. Blöcke von Leerzeichen oder Schlüsselwörtern identifiziert und für einen Angriff mit bekanntem Klartext verwendet werden können.

Wir überlegen uns, was passieren kann, wenn eine Blockchiffre in einem Datenbanksystem verwendet wird. Die Chiffrierung eines jeden Feldes in einem Record als einzelner Block ist ungünstig. Falls die Felder nicht vollständig belegt sind, müssen sie durch Leerzeichen aufgefüllt werden. Damit sind i. allg. Platzprobleme verbunden, und es bestehen, wie oben erwähnt, unter Umständen Identifizierungsmöglichkeiten. Wird die Chiffrierung mit kleineren Blöcken durchgeführt, ist sie leichter zu brechen. Selbst wenn die Felder gefüllt sind und eine Kryptoanalyse nicht möglich ist, kann durch Betrachtung des Chiffretextes eventuell Information zurückgewonnen werden. Man betrachte in diesem Zusammenhang eine Datenbank mit persönlichen Daten. Sie beinhalte u. a. ein unverschlüsseltes Namensfeld und ein Gehaltsfeld. Die Gehaltsfelder sind mit demselben Schlüssel in einem Block chiffriert. Ein Benutzer kann jeden persönlichen Record identifizieren und so feststellen, ob zwei Mitarbeiter jeweils das gleiche Gehalt verdienen. Daraus wird deutlich, daß auch nichtvertrauliche Information verschlüsselt werden muß.

Eine Block-Chiffrierung ist anfällig gegen ein „Replay“. Falls jeder Block unabhängig mit demselben Schlüssel chiffriert wird, kann ein Block für einen anderen eingesetzt werden.




In einer Gehaltsdatenbank kann ein Benutzer sich selber eine Gehaltserhöhung verschaffen, indem er das Gehaltsfeld eines besser bezahlten Mitarbeiters auf sein eigenes Gehaltsfeld kopiert. Eine Blockchiffrierung ist somit anfällig für das Einfügen und Löschen von Blocks, da keine umgebenden Blöcke beeinflußt werden.

Es ist sicherlich schwierig, einen falschen Chiffretext für einen normalen Text zu erzeugen. Bei numerischen Daten ist das dagegen einfach. Wenn z. B. eine Zahl nur $\neq 0$ sein muß, reicht irgendeine Zahl, die verschlüsselt wird. Zur Verhinderung eines solchen Vorgehens werden z. B. Prüfsummen benutzt.

Im folgenden werden Verfahren vorgestellt, die Blockchiffren gegenüber Kryptoanalysis und Chiffretextsubstitution, einschließlich Replay, sicherer machen. Wir beginnen mit dem *Chiffre-Block-Chaining* (CBC) [30]. Wir betrachten das Bild von Seite 142. Der $i$-te Klartextblock $M_i$ wird zu

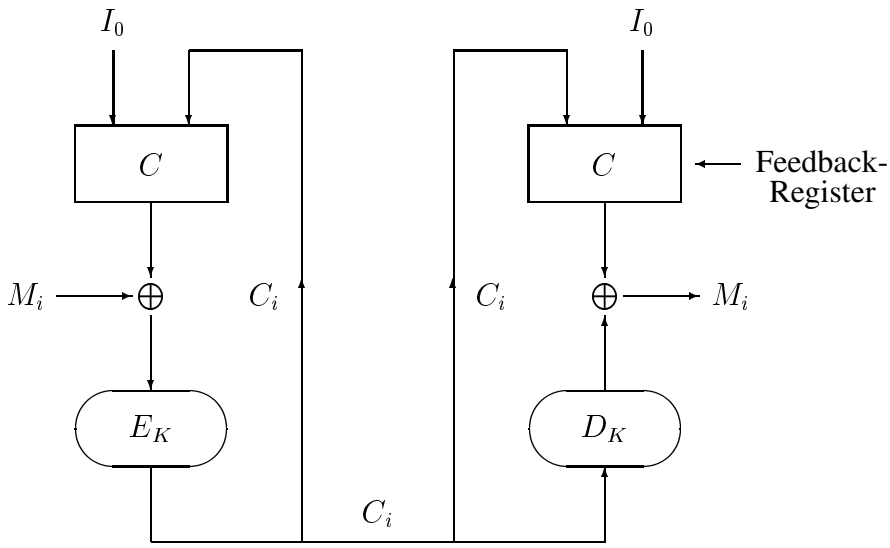
$$C_i = E_K(M_i \oplus C_{i-1})$$

chiffriert. Für $i = 1$ wird dabei $C_0 = I_0$ verwendet. Die Dechiffrierung erfolgt durch

$$\begin{aligned} D_K(C_i) \oplus C_{i-1} &= D_K(E_K(M_i \oplus C_{i-1})) \oplus C_{i-1} \\ &= M_i \oplus C_{i-1} \oplus C_{i-1} = M_i. \end{aligned}$$

Dabei wird jeder Chiffretextblock $C_i$ aus $M_i$ und dem vorhergehenden Chiffretextblock $C_{i-1}$ berechnet. Jeder Übertragungsfehler berührt daher mindestens zwei Blöcke, und jedes $C_i$ ist damit funktional abhängig von allen anderen vorhergehenden Chiffretextblöcken. Die statistischen Eigenschaften des Klartextes werden über den ganzen Chiffretext verstreut, wodurch die Kryptoanalysis schwieriger wird.

Das CBC wird im *Information Protection System* (IPS) von IBM benutzt. $E$ und $D$ sind dort durch den DES implementiert. Das Chaining kann auf einzelne Records oder auf eine Menge von Records (Record-Chaining) angewendet werden. Im ersten Fall wird ein Feedback-Register zu Beginn eines jeden Records auf $I_0$ gesetzt. Im zweiten Fall behält das Feedback-Register seinen Wert über die Recordgrenzen hinaus. Gleiche Zeilen eines Files werden verborgen. Bei einem Zugriff auf einzelne Records müssen jedoch erst alle vorhergehenden Records dechiffriert werden.




Chiffre-Block-Chaining (CBC)

Als nächstes betrachten wir *Blockchiffren mit Teilschlüsseln* [14]. Eine Datenbank wird als Menge von Records zu je $t$ Feldern modelliert. Jedes Record wird als Einheit chiffriert, dabei sind alle Felder über den Chiffretext verstreut. Die individuellen Felder können getrennt dechiffriert werden, obwohl dafür der Zugang zum ganzen Chiffretextrecord erforderlich ist. Diese Methode ist sicher gegenüber vielen denkbaren Angriffen. Für den Zugang zu den speziellen Feldern sind Teilschlüssel erforderlich, und zwar Leseteilschlüssel $d_1, d_2, \ldots, d_t$ und Schreibteilschlüssel $e_1, e_2, \ldots, e_t$.

Diese Teilschlüssel sind global für die Datenbank, d. h., alle Records werden mit denselben Teilschlüsseln chiffriert. Jeder Benutzer besitzt nur die Teilschlüssel für die Felder, die er lesen oder beschreiben darf. Zunächst folgt eine vereinfachte, aber unsichere Version des Schemas, die auf dem Chinesischen Restesatz (Satz 3.14) beruht. Jedes $d_j$ wird als zufällige Primzahl, die größer als der maximal mögliche Wert des Feldes $j$ ist, gewählt. Es sei $n := d_1 d_2 \ldots d_t$. Die $d_i$ müssen paarweise teilerfremd sein. Dann setzen wir

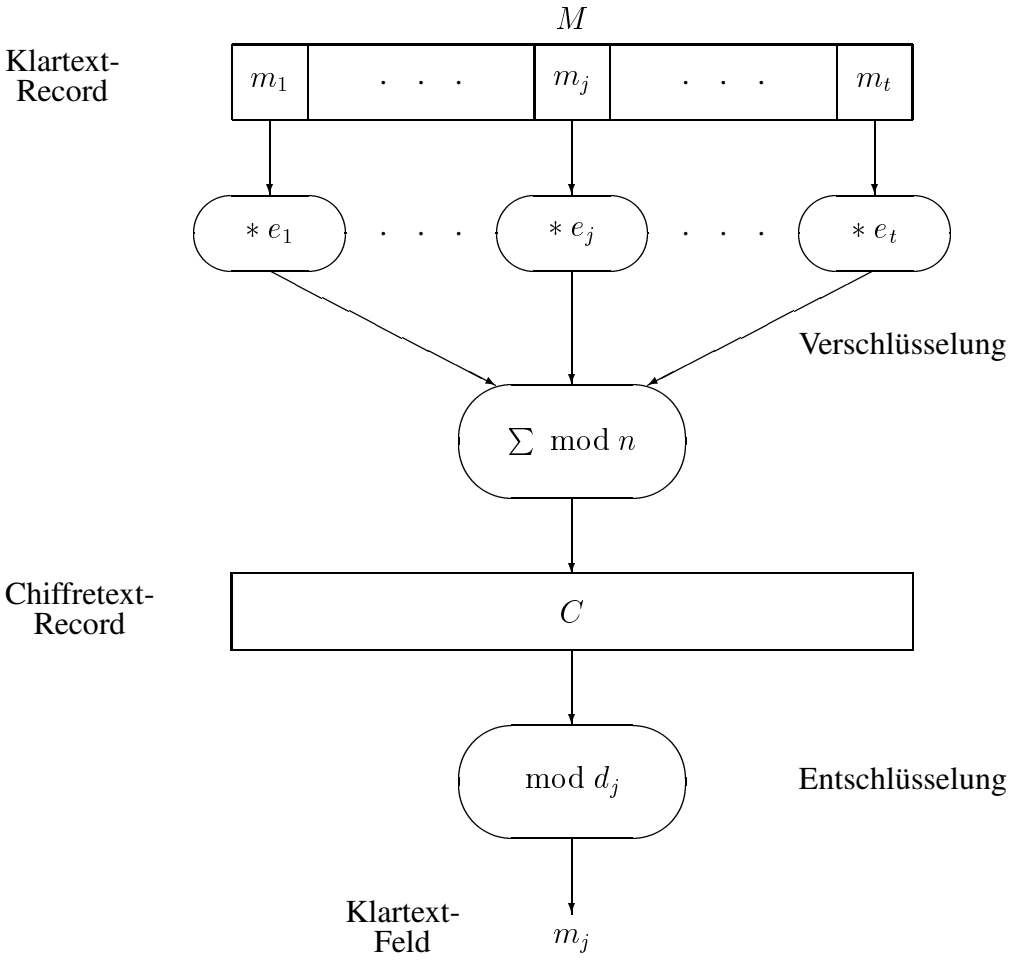
$$e_j = \frac{n}{d_j} \cdot y_j,$$

wobei $y_j$ =inv$(\frac{n}{d_j}, d_j)$ mit Hilfe von Algorithmus 3.3 bestimmt wird. Der Klartextrecord $M$ habe die Felder $m_1, \ldots, m_t$. Die Chiffrierung wird durch

$$C = \sum_{j=1}^{t} (e_j m_j) \bmod n$$

beschrieben. Nach dem Beweis von Satz 3.14 ist $C$ die Lösung des Gleichungssystems

$$(*) \qquad C \bmod d_j = m_j,\ j = 1, 2, \ldots, t.$$




Chiffrierung und Dechiffrierung mit Teilschlüsseln

Damit ist gesichert, daß das $j$-te Feld nur mit dem Teilschlüssel $d_j$ dechiffriert werden kann. Das $j$-te Feld kann geändert werden, wenn $e_j$ und $d_j$ zusammen bekannt sind, nämlich

$$C' = (C - e_j(C \bmod d_j) + e_j m_j') \bmod n.$$

Dabei ist $m_j'$ der neue Wert des $j$-ten Feldes.

**Beispiel 9.2.2** Es seien $t = 3$, $d_1 = 7$, $d_2 = 11$ und $d_3 = 5$ gegeben. Es ist also $n = 7 \cdot 11 \cdot 5 = 385$. Dann folgt

$$\begin{aligned} y_1 &= \text{inv}(\tfrac{385}{7}, 7) &&= \text{inv}(55, 7) &&= 6, \\ y_2 &= \text{inv}(\tfrac{385}{11}, 11) &&= \text{inv}(35, 11) &&= 6, \\ y_3 &= \text{inv}(\tfrac{385}{5}, 5) &&= \text{inv}(77, 5) &&= 3. \end{aligned}$$

Damit erhalten wir

$$\begin{aligned} e_1 &= 55 \cdot 6 = 330, \\ e_2 &= 35 \cdot 6 = 210, \\ e_3 &= 77 \cdot 3 = 231. \end{aligned}$$

Der Klartextrecord sei $M = (4, 10, 2)$. Dann ergibt die Chiffrierung

$$\begin{aligned} C &= (e_1 m_1 + e_2 m_2 + e_3 m_3) \bmod n \\ &= (330 \cdot 4 + 210 \cdot 10 + 231 \cdot 2) \bmod 385 \\ &= (1320 + 2100 + 462) \bmod 385 \\ &= 3882 \bmod 385 = 32. \end{aligned}$$

Die Dechiffrierung der Felder liefert

$$\begin{aligned} m_1 &= C \bmod d_1 = 32 \bmod 7 = 4, \\ m_2 &= C \bmod d_2 = 32 \bmod 11 = 10, \\ m_3 &= C \bmod d_3 = 32 \bmod 5 = 2. \end{aligned}$$

Die Änderung des Inhalts des zweiten Feldes von 10 auf 8 wird durch

$$\begin{aligned} C' &= (C - e_2(C \bmod d_2) + e_2 \cdot 8) \bmod 385 \\ &= (32 - 210 \cdot 10 + 210 \cdot 8) \bmod 385 \\ &= -388 \bmod 385 = -3 \bmod 385 = 382 \end{aligned}$$

chiffriert. Nach dieser Änderung gilt

$$\begin{aligned} m_1 &= C' \bmod d_1 = 382 \bmod 7 = 4, \\ m_2 &= C' \bmod d_2 = 382 \bmod 11 = 52 \bmod 11 = 8, \\ m_3 &= C' \bmod d_3 = 382 \bmod 5 = 2. \quad \square \end{aligned}$$

Das beschriebene Schema hat zwei Schwächen:

(1) Es sei $m_{ij}$ der Wert des $j$-ten Feldes im $i$-ten Record. Wenn ein Benutzer für zwei Records $M_1$ und $M_2$ die Werte $m_{1j}$ und $m_{2j}$ kennt, dann kann er den Leseteilschlüssel $d_j$ aus den Chiffretexten $C_1$ und $C_2$ bestimmen. Aus der Gleichung $(*)$ folgt nämlich

$$\begin{aligned} C_1 - m_{1j} &= u_1 d_j \\ C_2 - m_{2j} &= u_2 d_j \end{aligned}$$

mit geeigneten Werten für $u_1$ bzw. $u_2$. Mit großer Wahrscheinlichkeit ist nun

$$d_j = \text{ggt}(C_1 - m_{1j}, C_2 - m_{2j}).$$

Diese Schwäche kann durch das Anhängen eines zufälligen 32- oder größeren Bit-Wertes $x_j$ an $m_j$ vor der Chiffrierung behoben werden. Damit wird $m_j$ zu $\widetilde{m}_j$. Die Chiffrierung erfolgt durch

$$C = \sum_{j=1}^{t} (e_j \, \widetilde{m}_j) \bmod n.$$

(2) Durch individuelle Änderungen der einzelnen Felder können ggf. Leseteilschlüssel aufgedeckt werden. Abhilfe schafft hier die Neuchiffrierung des ganzen Records mit neuen zufälligen Werten $x_j$ für alle Felder.

Zum Beschreiben eines Feldes werden sowohl der Lese- als auch der Schreibteilschlüssel benötigt. Mit der Erlaubnis zum Beschreiben eines Feldes hat ein Benutzer A auch automatisch die Erlaubnis zum Lesen. Dies ist auch sonst üblich. Zum Beschreiben eines Feldes benötigt A zusätzlich $n$. Damit kann A mit Hilfe der Definition von $e_j$ auf Seite 142 auch die Schreibteilschlüssel für solche Felder berechnen, für die A nur Leseteilschlüssel besitzt. Das ist allerdings unüblich. Um das zu verhindern, muß $n$ in dem Programm, das den Zugang zu der Datenbank gewährt, verborgen sein. Zum Lesen eines Feldes $j$ ruft A eine Leseprozedur auf und übergibt dabei den Leseteilschlüssels $d_j$. Zum Beschreiben eines Feldes $j$ ruft A eine Schreibprozedur auf und übergibt dabei die Parameter $d_j$, $e_j$ und $m'_j$. Da $e_j$ aus $n$ und $d_j$ berechnet wird, kann jede Gruppe von Benutzern, die Zugang zu allen Leseteilschlüsseln hat, die Zahl $n = d_1 d_2 \ldots d_t$ und damit alle Schreibteilschlüssel $e_j$ berechnen. Zur Verhinderung werden in jedes Record Dummy-Felder eingefügt. Die Lese- und Schreibteilschlüssel dieser Felder sind für keinen Benutzer zugänglich.

# Anhang: Häufigkeitstabellen

| Buchstabe | Häufigkeit in % | | | | | |
|---|---|---|---|---|---|---|
| | Deutsch | | | | Englisch | |
| | mit Umlaut-ersetzung | mit Umlauten | mit Uml.-ers. u. Satz- und Leerzeichen | mit Umlauten u. Satz- und Leerzeichen | | mit Satz- und Leerzeichen |
| a | 6,43 | 6,03 | 5,14 | 4,80 | 8,17 | 6,60 |
| b | 1,85 | 1,89 | 1,48 | 1,50 | 1,49 | 1,21 |
| c | 3,26 | 3,33 | 2,61 | 2,65 | 2,78 | 2,25 |
| d | 5,12 | 5,21 | 4,09 | 4,16 | 4,25 | 3,43 |
| e | 17,74 | 16,64 | 14,19 | 13,26 | 12,70 | 10,26 |
| f | 1,56 | 1,59 | 1,25 | 1,27 | 2,23 | 1,80 |
| g | 2,69 | 2,74 | 2,15 | 2,18 | 2,02 | 1,63 |
| h | 5,22 | 5,32 | 4,17 | 4,24 | 6,09 | 4,92 |
| i | 7,60 | 7,75 | 6,08 | 6,18 | 6,97 | 5,63 |
| j | 0,23 | 0,24 | 0,19 | 0,19 | 0,15 | 0,12 |
| k | 1,40 | 1,42 | 1,12 | 1,13 | 0,77 | 0,62 |
| l | 3,49 | 3,56 | 2,79 | 2,84 | 4,03 | 3,25 |
| m | 2,75 | 2,80 | 2,20 | 2,23 | 2,41 | 1,94 |
| n | 10,01 | 10,21 | 8,01 | 8,13 | 6,75 | 5,45 |
| o | 2,39 | 2,17 | 1,92 | 1,73 | 7,51 | 6,06 |
| p | 0,64 | 0,65 | 0,51 | 0,52 | 1,93 | 1,56 |
| q | 0,01 | 0,01 | 0,01 | 0,01 | 0,08 | 0,10 |
| r | 6,98 | 7,11 | 5,58 | 5,67 | 5,99 | 4,84 |
| s | 6,88 | 6,04 | 5,51 | 4,81 | 6,33 | 5,11 |
| t | 5,94 | 6,06 | 4,75 | 4,83 | 9,06 | 7,31 |
| u | 4,27 | 3,70 | 3,41 | 2,95 | 2,76 | 2,23 |
| v | 0,64 | 0,65 | 0,51 | 0,52 | 0,98 | 0,79 |
| w | 1,73 | 1,77 | 1,39 | 1,41 | 2,36 | 1,91 |
| x | 0,02 | 0,02 | 0,01 | 0,02 | 0,15 | 0,12 |
| y | 0,04 | 0,04 | 0,03 | 0,03 | 1,97 | 1,59 |
| z | 1,10 | 1,12 | 0,88 | 0,90 | 0,07 | 0,06 |
| ä | - | 0,52 | - | 0,42 | - | - |
| ö | - | 0,27 | - | 0,22 | - | - |
| ü | - | 0,65 | - | 0,51 | - | - |
| ß | - | 0,49 | - | 0,39 | - | - |
| $*$ | - | - | 20,01 | 20,31 | - | 19,25 |

Tabelle 1:
Häufigkeit der einzelnen Buchstaben
(das Symbol $*$ steht für weitere Zeichen (s.u.),
mit Umlautersetzung bedeutet: ä = ae, ö = oe, ü = ue, ß = ss)

Die Daten für Englisch stammen aus [5]. Die deutschen Daten sind von uns hier wie auch in den weiteren Tabellen aus jeweils denselben überwiegend literarischen Texten berechnet worden. Ziffern sind in jedem Fall überlesen worden. In der dritten und vierten Spalte sind auch Leer- und Satzzeichen, Zeilenwechselzeichen usw. gezählt und unter $*$ zusammengefaßt worden. Durch die verschiedenen Arten der Zählung wurden für die erste Spalte 224826 Symbole, für die zweite 220558, für die dritte 281052 und die vierte 276784 Symbole berücksichtigt.

In den beiden folgenden Tabellen werden Umlaute als ä = ae, ö =oe, ü = ue, der Buchstabe ß wird als ss berücksichtigt. Sonstige Symbole werden nicht gezählt. Die Tabellen beruhen also auf der ersten Spalte (Deutsch) bzw. der fünften Spalte (Englisch) von Tabelle 1.

| Gruppe | Anteil der Buchstaben dieser Gruppe an einem Text |
|---|---|
| e, n | 27,75% |
| i, r, s, a, t | 33,83% |
| h, d, u, l, c, m, g, o, b, w, f, k, z | 36,84% |
| p, v, j, y, x, q | 1,58% |

Tabelle 2
Buchstabengruppen im Deutschen

| Gruppe | Anteil der Buchstaben dieser Gruppe an einem Text |
|---|---|
| e, t | 21,76% |
| a, o, i, n, s, h, r | 47,81% |
| d, l, c, u, m, w, f, g, y, p, b | 28,23% |
| v, k, j, x, q, z | 2,20% |

Tabelle 3
Buchstabengruppen im Englischen

In den Tabellen 2 und 3 sind die Buchstaben gemäß ihrem Auftreten in vier Gruppen eingeteilt. In der ersten Gruppe sind die sehr häufigen Buchstaben e und n bzw. t, in der zweiten befinden sich die Buchstaben, deren Häufigkeit noch relativ groß (d.h. etwa 7%) ist; in der dritten Gruppe sind die Buchstaben zusammengefaßt, die eine kleine, aber noch merkliche Häufigkeit haben, während in der letzten Gruppe die vernachlässigbaren Buchstaben aufgeführt sind.

## Bi-und Digrammtabellen für Deutsch ohne Berücksichtigung von Leerzeichen usw.

In den folgenden beiden Tabellen sind die Umlaute und der Buchstabe ß wie üblich ersetzt worden. Leerzeichen, Satzzeichen, Ziffern, Zeilenwechselzeichen usw. wurden überlesen. So kann z.B. ein Paar entstehen aus dem letzten Buchstaben eines Wortes mit dem ersten Buchstaben des folgenden Wortes. Insgesamt wurden 224826 Buchstaben des Textes gezählt.

| Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit |
|---|---|---|---|---|---|---|---|
| er | 3,89 | ng | 0,70 | ll | 0,41 | oe | 0,28 |
| en | 3,74 | ha | 0,67 | rt | 0,40 | ef | 0,28 |
| ch | 2,97 | eh | 0,67 | eu | 0,39 | ck | 0,28 |
| te | 2,21 | ni | 0,64 | ru | 0,39 | rg | 0,28 |
| nd | 2,11 | ed | 0,64 | ac | 0,37 | ew | 0,27 |
| ei | 2,07 | nn | 0,62 | ee | 0,36 | mm | 0,26 |
| de | 2,06 | rs | 0,62 | tt | 0,35 | tz | 0,26 |
| ie | 1,87 | em | 0,62 | td | 0,35 | hl | 0,26 |
| in | 1,87 | na | 0,60 | on | 0,35 | um | 0,26 |
| es | 1,45 | et | 0,55 | ti | 0,35 | ho | 0,25 |
| ge | 1,41 | rd | 0,55 | or | 0,35 | ea | 0,25 |
| ne | 1,26 | al | 0,55 | mi | 0,35 | ka | 0,25 |
| un | 1,24 | me | 0,54 | at | 0,34 | im | 0,24 |
| ic | 1,19 | wa | 0,54 | ab | 0,34 | nm | 0,24 |
| st | 1,11 | ar | 0,53 | rn | 0,34 | rw | 0,24 |
| an | 1,09 | it | 0,53 | hi | 0,33 | gt | 0,24 |
| se | 1,08 | ae | 0,53 | ig | 0,32 | tr | 0,24 |
| re | 1,00 | ma | 0,52 | ts | 0,32 | nh | 0,23 |
| be | 0,96 | ri | 0,51 | lt | 0,32 | nz | 0,23 |
| he | 0,95 | is | 0,51 | nk | 0,32 | sd | 0,23 |
| di | 0,90 | li | 0,50 | hn | 0,31 | rm | 0,23 |
| ns | 0,86 | we | 0,49 | vo | 0,31 | ls | 0,23 |
| au | 0,85 | eb | 0,48 | ke | 0,31 | nb | 0,22 |
| ss | 0,85 | ta | 0,47 | ag | 0,31 | ve | 0,22 |
| sc | 0,84 | nt | 0,45 | fe | 0,31 | ze | 0,22 |
| si | 0,84 | us | 0,44 | uf | 0,31 | am | 0,22 |
| el | 0,79 | hr | 0,44 | nw | 0,31 | ro | 0,22 |
| ue | 0,75 | eg | 0,44 | ih | 0,31 | oc | 0,22 |
| le | 0,75 | nu | 0,43 | tu | 0,30 | rk | 0,21 |
| da | 0,74 | wi | 0,42 | ur | 0,30 | ol | 0,21 |
| ht | 0,73 | la | 0,42 | ec | 0,30 | ut | 0,20 |
| as | 0,73 | zu | 0,42 | so | 0,29 | du | 0,20 |
| ra | 0,71 | sa | 0,41 | | | | |

Tabelle 4

Bigrammhäufigkeiten in Prozenten (Deutsch)

| Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit |
|---|---|---|---|---|---|---|---|
| ein | 1,14 | nic | 0,30 | ben | 0,21 | nsi | 0,18 |
| ich | 1,12 | ber | 0,30 | mit | 0,21 | eni | 0,18 |
| der | 0,92 | nen | 0,29 | uer | 0,21 | ung | 0,18 |
| sch | 0,84 | ebe | 0,29 | man | 0,21 | cha | 0,17 |
| und | 0,81 | ndi | 0,29 | enw | 0,21 | chs | 0,17 |
| die | 0,74 | nun | 0,28 | enu | 0,21 | wei | 0,17 |
| nde | 0,70 | ass | 0,26 | sta | 0,20 | eng | 0,17 |
| cht | 0,67 | rde | 0,26 | ren | 0,20 | ist | 0,17 |
| ine | 0,57 | auf | 0,26 | ess | 0,20 | nei | 0,17 |
| den | 0,55 | rei | 0,26 | tes | 0,20 | ndd | 0,17 |
| end | 0,54 | sic | 0,26 | wie | 0,20 | tun | 0,17 |
| che | 0,52 | ern | 0,26 | est | 0,20 | nsc | 0,17 |
| ens | 0,51 | war | 0,25 | esi | 0,20 | mme | 0,17 |
| ers | 0,51 | tte | 0,25 | eri | 0,20 | tde | 0,17 |
| ten | 0,50 | ner | 0,25 | ese | 0,20 | enk | 0,17 |
| ter | 0,44 | enn | 0,25 | lte | 0,19 | ing | 0,16 |
| gen | 0,42 | sen | 0,25 | rau | 0,19 | rge | 0,16 |
| ste | 0,38 | ene | 0,25 | eit | 0,19 | ehr | 0,16 |
| nge | 0,37 | aus | 0,24 | hre | 0,19 | erh | 0,16 |
| das | 0,37 | ert | 0,24 | och | 0,19 | ang | 0,16 |
| sie | 0,37 | ena | 0,24 | dem | 0,19 | erk | 0,16 |
| ere | 0,35 | lic | 0,23 | erw | 0,19 | nds | 0,16 |
| ach | 0,34 | abe | 0,23 | tei | 0,19 | nst | 0,16 |
| erd | 0,34 | era | 0,22 | ied | 0,18 | chw | 0,16 |
| sse | 0,34 | ege | 0,22 | ges | 0,18 | ier | 0,15 |
| hen | 0,31 | ann | 0,22 | chd | 0,18 | hat | 0,15 |
| ede | 0,31 | nte | 0,22 | uch | 0,18 | ted | 0,15 |
| and | 0,31 | hte | 0,22 | rst | 0,18 | men | 0,15 |
| ies | 0,30 | lle | 0,22 | lei | 0,18 | ger | 0,15 |
| ind | 0,30 | her | 0,22 | erg | 0,18 | ige | 0,15 |
| sei | 0,30 | ver | 0,22 | nda | 0,18 | | |

Tabelle 5

Trigrammhäufigkeiten in Prozenten (Deutsch)

## Bi-und Digrammtabellen für Deutsch mit Umlauten und ohne Berücksichtigung von Leerzeichen usw.

In den folgenden beiden Tabellen wurden Leerzeichen, Satzzeichen, Ziffern, Zeilenwechselzeichen usw. überlesen. So kann z.B. ein Paar entstehen aus dem letzten Buchstaben eines Wortes mit dem ersten Buchstaben des folgenden Wortes. Insgesamt wurden 220558 Buchstaben des Textes gezählt. Im Vergleich zu den Tabellen 4 und 5 stellen wir fest, daß sich trotz der zusätzlich berücksichtigten Umlaute die Reihenfolge der meisten Paare und Tripel kaum ändert und auch die jeweiligen Werte nur geringfügig abweichen. Eine Ausnahme stellt jedoch vor allem ss dar, der Wert in Tabelle 4 ist 0,85, in Tabelle 6 dagegen nur noch 0,34.

| Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit |
|---|---|---|---|---|---|---|---|
| er | 3,65 | ni | 0,66 | ac | 0,37 | ck | 0,28 |
| en | 3,59 | nn | 0,64 | eb | 0,36 | rg | 0,28 |
| ch | 3,03 | rs | 0,63 | ee | 0,36 | ew | 0,27 |
| te | 2,26 | ed | 0,62 | td | 0,36 | ru | 0,27 |
| nd | 2,15 | ha | 0,61 | la | 0,36 | mm | 0,26 |
| ei | 2,11 | em | 0,60 | tt | 0,36 | tz | 0,26 |
| de | 2,10 | na | 0,58 | on | 0,36 | hl | 0,26 |
| ie | 1,91 | as | 0,57 | ti | 0,36 | um | 0,26 |
| in | 1,91 | rd | 0,56 | or | 0,36 | ea | 0,25 |
| ge | 1,43 | al | 0,56 | mi | 0,36 | ef | 0,25 |
| es | 1,32 | me | 0,55 | us | 0,35 | im | 0,25 |
| ne | 1,29 | eh | 0,55 | at | 0,35 | nm | 0,24 |
| un | 1,27 | ar | 0,54 | ab | 0,35 | rw | 0,24 |
| ic | 1,21 | it | 0,54 | rn | 0,35 | gt | 0,24 |
| an | 1,11 | ri | 0,52 | eu | 0,34 | tr | 0,24 |
| st | 1,05 | li | 0,51 | ss | 0,34 | ka | 0,24 |
| re | 1,02 | we | 0,50 | hi | 0,34 | tu | 0,24 |
| be | 0,98 | wa | 0,49 | ig | 0,33 | nh | 0,23 |
| he | 0,97 | ma | 0,48 | ts | 0,33 | nz | 0,23 |
| se | 0,95 | nt | 0,46 | lt | 0,33 | rm | 0,23 |
| di | 0,92 | is | 0,45 | nk | 0,33 | ls | 0,23 |
| ns | 0,88 | hr | 0,45 | hn | 0,32 | nb | 0,23 |
| au | 0,87 | et | 0,43 | ke | 0,32 | ve | 0,23 |
| sc | 0,85 | wi | 0,43 | ag | 0,32 | ze | 0,22 |
| si | 0,82 | ta | 0,43 | fe | 0,32 | am | 0,22 |
| le | 0,76 | ll | 0,42 | vo | 0,32 | oc | 0,22 |
| da | 0,75 | zu | 0,42 | uf | 0,31 | rk | 0,21 |
| ht | 0,75 | rt | 0,40 | nw | 0,31 | ol | 0,21 |
| el | 0,74 | nu | 0,40 | ih | 0,31 | ut | 0,21 |
| ng | 0,71 | sa | 0,39 | ur | 0,30 | uc | 0,20 |
| ra | 0,66 | eg | 0,39 | so | 0,29 | rb | 0,20 |

Tabelle 6

Bigrammhäufigkeiten in Prozenten (Deutsch mit Umlauten)

| Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit |
|---|---|---|---|---|---|---|---|
| ein | 1,16 | sei | 0,30 | tes | 0,21 | ung | 0,18 |
| ich | 1,14 | ndi | 0,29 | wie | 0,20 | chs | 0,18 |
| der | 0,94 | nun | 0,29 | enn | 0,20 | wei | 0,17 |
| sch | 0,85 | das | 0,29 | esi | 0,20 | ist | 0,17 |
| und | 0,82 | rde | 0,27 | sen | 0,20 | nei | 0,17 |
| die | 0,75 | auf | 0,27 | lte | 0,20 | ndd | 0,17 |
| nde | 0,71 | rei | 0,26 | rau | 0,20 | tun | 0,17 |
| cht | 0,68 | war | 0,26 | hre | 0,20 | nsc | 0,17 |
| ine | 0,58 | ies | 0,26 | eit | 0,20 | mme | 0,17 |
| den | 0,56 | tte | 0,26 | enu | 0,20 | tde | 0,17 |
| che | 0,53 | ner | 0,26 | och | 0,20 | ing | 0,17 |
| ten | 0,51 | ern | 0,26 | dem | 0,19 | eni | 0,17 |
| ens | 0,51 | sic | 0,25 | ege | 0,19 | rge | 0,17 |
| end | 0,49 | lic | 0,24 | ebe | 0,19 | erg | 0,17 |
| ers | 0,48 | ene | 0,23 | tei | 0,19 | enk | 0,16 |
| ter | 0,45 | abe | 0,23 | eri | 0,19 | ang | 0,16 |
| gen | 0,43 | ena | 0,23 | erw | 0,19 | nds | 0,16 |
| nge | 0,38 | aus | 0,23 | ied | 0,19 | cha | 0,16 |
| sie | 0,37 | ann | 0,22 | ese | 0,19 | nst | 0,16 |
| ach | 0,35 | nte | 0,22 | ert | 0,19 | chw | 0,16 |
| ste | 0,34 | hte | 0,22 | ges | 0,19 | ier | 0,16 |
| hen | 0,32 | lle | 0,22 | chd | 0,19 | hat | 0,16 |
| and | 0,31 | her | 0,22 | est | 0,18 | ted | 0,15 |
| ind | 0,31 | ver | 0,22 | sse | 0,18 | erk | 0,15 |
| erd | 0,31 | ben | 0,22 | sta | 0,18 | men | 0,15 |
| nic | 0,30 | mit | 0,22 | uch | 0,18 | ige | 0,15 |
| ber | 0,30 | man | 0,21 | rst | 0,18 | ger | 0,15 |
| ere | 0,30 | enw | 0,21 | lei | 0,18 | gte | 0,15 |
| ede | 0,30 | era | 0,21 | nda | 0,18 | nne | 0,15 |
| nen | 0,30 | ren | 0,21 | nsi | 0,18 | run | 0,15 |

Tabelle 7

Trigrammhäufigkeiten in Prozenten (Deutsch mit Umlauten)

## Bi-und Digrammtabellen für Deutsch mit Berücksichtigung von Leerzeichen usw.

In den folgenden beiden Tabellen sind die Umlaute und der Buchstabe ß wie üblich ersetzt worden. Leerzeichen, Satzzeichen, Zeilenwechselzeichen usw. wurden berücksichtigt und unter ∗ zusammengefaßt. Es kann so z.B kein Paar normaler Buchstaben entstehen, das aus dem letzten Buchstaben eines Wortes und dem ersten Buchstaben des folgenden Wortes besteht. Ziffern wurden überlesen. Insgesamt wurden 281052 Symbole des Textes gezählt.

| Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit |
|---|---|---|---|---|---|---|---|
| ∗∗ | 4,19 | ne | 0,78 | eh | 0,42 | tt | 0,27 |
| n∗ | 3,50 | ∗m | 0,76 | it | 0,42 | ab | 0,27 |
| er | 3,06 | se | 0,75 | ae | 0,41 | ∗t | 0,27 |
| en | 2,89 | be | 0,75 | ra | 0,41 | rt | 0,27 |
| e∗ | 2,71 | ∗h | 0,72 | ng | 0,40 | mi | 0,26 |
| ch | 2,37 | m∗ | 0,69 | et | 0,40 | ta | 0,26 |
| r∗ | 2,28 | he | 0,69 | we | 0,39 | ig | 0,26 |
| ∗d | 2,28 | di | 0,68 | is | 0,39 | lt | 0,26 |
| ∗s | 1,87 | au | 0,68 | nn | 0,39 | vo | 0,25 |
| t∗ | 1,61 | sc | 0,67 | me | 0,39 | ag | 0,25 |
| te | 1,59 | ∗g | 0,66 | li | 0,38 | ke | 0,24 |
| de | 1,58 | re | 0,63 | ∗l | 0,38 | ih | 0,24 |
| ei | 1,50 | ∗n | 0,61 | ma | 0,37 | uf | 0,24 |
| in | 1,49 | ∗b | 0,59 | em | 0,36 | f∗ | 0,24 |
| ie | 1,49 | si | 0,58 | ni | 0,35 | ns | 0,24 |
| s∗ | 1,33 | ue | 0,58 | hr | 0,34 | na | 0,24 |
| ∗e | 1,28 | ht | 0,58 | g∗ | 0,34 | u∗ | 0,24 |
| nd | 1,14 | le | 0,57 | wi | 0,34 | sa | 0,24 |
| ge | 1,09 | as | 0,56 | ll | 0,32 | eg | 0,24 |
| ∗a | 1,08 | ss | 0,55 | zu | 0,32 | ur | 0,23 |
| ∗w | 1,04 | da | 0,54 | us | 0,32 | ec | 0,23 |
| un | 0,99 | el | 0,54 | la | 0,31 | fe | 0,23 |
| ic | 0,95 | ∗k | 0,53 | l∗ | 0,31 | ck | 0,22 |
| h∗ | 0,92 | ha | 0,47 | nt | 0,30 | so | 0,22 |
| d∗ | 0,88 | ∗v | 0,46 | ri | 0,29 | oe | 0,22 |
| st | 0,87 | ∗z | 0,44 | ac | 0,29 | ru | 0,21 |
| an | 0,86 | ∗f | 0,44 | on | 0,28 | ∗r | 0,21 |
| ∗i | 0,86 | al | 0,43 | or | 0,28 | hi | 0,21 |
| ∗u | 0,85 | wa | 0,43 | eb | 0,28 | rs | 0,21 |
| es | 0,81 | ar | 0,42 | at | 0,27 | eu | 0,20 |

Tabelle 8

Bigrammhäufigkeiten in Prozenten (Deutsch, mit zusätzlichen Zeichen)

| Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit |
|---|---|---|---|---|---|---|---|
| en∗ | 2,09 | es∗ | 0,36 | st∗ | 0,24 | lic | 0,18 |
| er∗ | 1,74 | n∗d | 0,36 | ∗∗a | 0,23 | abe | 0,18 |
| ∗∗∗ | 1,18 | n∗s | 0,36 | ∗ih | 0,23 | ers | 0,18 |
| ∗de | 0,92 | ∗sc | 0,35 | ∗ni | 0,23 | ∗ic | 0,18 |
| ie∗ | 0,90 | che | 0,35 | ∗mi | 0,23 | t∗d | 0,18 |
| ich | 0,90 | ∗wa | 0,33 | nn∗ | 0,23 | ∗∗i | 0,18 |
| ch∗ | 0,86 | gen | 0,33 | ∗vo | 0,23 | an∗ | 0,18 |
| ein | 0,86 | r∗∗ | 0,32 | ber | 0,23 | n∗a | 0,17 |
| n∗∗ | 0,82 | ∗ha | 0,32 | ∗an | 0,23 | s∗∗ | 0,17 |
| nd∗ | 0,77 | ∗∗s | 0,31 | ne∗ | 0,22 | ∗es | 0,17 |
| der | 0,71 | as∗ | 0,30 | sse | 0,22 | end | 0,17 |
| te∗ | 0,70 | e∗s | 0,30 | it∗ | 0,21 | ver | 0,17 |
| ∗un | 0,70 | sie | 0,29 | e∗d | 0,21 | mit | 0,17 |
| ∗∗d | 0,68 | ∗∗w | 0,29 | nen | 0,21 | ann | 0,17 |
| sch | 0,67 | das | 0,29 | auf | 0,21 | ben | 0,17 |
| und | 0,64 | ∗wi | 0,29 | r∗d | 0,21 | ∗sa | 0,17 |
| ∗di | 0,62 | ter | 0,28 | ∗al | 0,21 | lle | 0,17 |
| die | 0,59 | ste | 0,28 | nic | 0,21 | nte | 0,16 |
| in∗ | 0,59 | ∗zu | 0,28 | war | 0,20 | uer | 0,16 |
| cht | 0,53 | ach | 0,27 | e∗e | 0,20 | n∗w | 0,16 |
| ∗ei | 0,50 | ∗se | 0,27 | nge | 0,20 | her | 0,16 |
| ∗da | 0,50 | nde | 0,27 | tte | 0,20 | wie | 0,16 |
| ∗si | 0,50 | em∗ | 0,26 | sei | 0,20 | e∗a | 0,16 |
| t∗∗ | 0,47 | ∗st | 0,26 | ss∗ | 0,19 | n∗e | 0,16 |
| e∗∗ | 0,46 | r∗s | 0,26 | and | 0,19 | ∗he | 0,16 |
| den | 0,43 | ∗∗e | 0,26 | sen | 0,19 | on∗ | 0,16 |
| ∗er | 0,42 | ∗we | 0,25 | sic | 0,19 | man | 0,16 |
| ine | 0,42 | ∗∗u | 0,25 | aus | 0,19 | ∗ve | 0,16 |
| ∗au | 0,38 | hen | 0,24 | ∗so | 0,19 | uf∗ | 0,15 |
| ∗ge | 0,38 | ∗be | 0,24 | ass | 0,18 | och | 0,15 |
| ten | 0,37 | ∗ma | 0,24 | ar∗ | 0,18 | eit | 0,15 |
| ht∗ | 0,37 | ∗in | 0,24 | ebe | 0,18 | ren | 0,15 |

Tabelle 9

Trigrammhäufigkeiten in Prozenten (Deutsch, mit zusätzlichen Zeichen)

## Bi-und Digrammtabellen für Deutsch mit Berücksichtigung von Umlauten, Leerzeichen usw.

Umlaute (ä, ö, ü, ß) wurden berücksichtigt. Leerzeichen, Satzzeichen, Zeilenwechselzeichen usw. wurden ebenfalls gezählt und unter ∗ zusammengefaßt. Ziffern wurden überlesen. Es kann so z.B kein Paar normaler Buchstaben entstehen, das aus dem letzten Buchstaben eines Wortes und dem ersten Buchstaben des folgenden Wortes besteht. Insgesamt wurden 276784 Symbole des Textes gezählt. Im Vergleich zu den Tabellen 8 und 9 stellen wir fest, daß sich trotz der zusätzlich berücksichtigten Umlaute die Reihenfolge der meisten Paare und Tripel kaum ändert und auch die jeweiligen Werte nur geringfügig abweichen. Eine Ausnahme stellt jedoch vor allem ss dar, der Wert in Tabelle 8 ist 0,55, ss ist jedoch in Tabelle 10 nicht mehr eingetragen.

| Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit |
|---|---|---|---|---|---|---|---|
| ∗∗ | 4,25 | ne | 0,79 | it | 0,42 | tt | 0,27 |
| n∗ | 3,55 | ∗u | 0,79 | ha | 0,42 | ab | 0,27 |
| er | 2,86 | ∗m | 0,77 | ng | 0,41 | ∗t | 0,27 |
| en | 2,76 | be | 0,76 | we | 0,40 | rt | 0,27 |
| e∗ | 2,75 | ∗h | 0,73 | nn | 0,39 | mi | 0,27 |
| ch | 2,41 | es | 0,70 | me | 0,39 | la | 0,26 |
| r∗ | 2,32 | m∗ | 0,70 | wa | 0,39 | ig | 0,26 |
| ∗d | 2,32 | he | 0,70 | li | 0,38 | lt | 0,26 |
| ∗s | 1,89 | di | 0,69 | ∗l | 0,38 | vo | 0,25 |
| t∗ | 1,64 | au | 0,69 | ra | 0,37 | ag | 0,25 |
| te | 1,61 | se | 0,68 | ni | 0,36 | ke | 0,25 |
| de | 1,60 | sc | 0,68 | hr | 0,35 | ih | 0,25 |
| ei | 1,53 | ∗g | 0,67 | g∗ | 0,35 | f∗ | 0,25 |
| in | 1,51 | re | 0,64 | wi | 0,34 | uf | 0,25 |
| ie | 1,51 | ∗n | 0,62 | ma | 0,34 | ns | 0,24 |
| ∗e | 1,30 | ∗b | 0,60 | em | 0,34 | us | 0,24 |
| nd | 1,16 | ht | 0,59 | is | 0,34 | u∗ | 0,24 |
| s∗ | 1,15 | si | 0,58 | ll | 0,33 | ur | 0,24 |
| ge | 1,11 | le | 0,58 | zu | 0,32 | sa | 0,23 |
| ∗a | 1,09 | da | 0,54 | eh | 0,32 | fe | 0,23 |
| ∗w | 1,06 | ∗k | 0,54 | l∗ | 0,31 | ta | 0,23 |
| un | 1,00 | el | 0,50 | nt | 0,30 | ck | 0,23 |
| ic | 0,96 | ∗v | 0,46 | et | 0,30 | na | 0,22 |
| h∗ | 0,93 | ∗z | 0,45 | ri | 0,30 | so | 0,22 |
| d∗ | 0,89 | ∗f | 0,44 | ac | 0,30 | ∗r | 0,22 |
| an | 0,88 | al | 0,44 | on | 0,28 | hi | 0,21 |
| ∗i | 0,87 | as | 0,43 | or | 0,28 | rs | 0,21 |
| st | 0,82 | ar | 0,43 | at | 0,27 | | |

Tabelle 10
(Deutsch, mit Umlauten und zusätzlichen Zeichen)

| Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit |
|---|---|---|---|---|---|---|---|
| en∗ | 2,11 | ht∗ | 0,37 | ∗ih | 0,24 | aus | 0,18 |
| er∗ | 1,69 | es∗ | 0,37 | ∗∗a | 0,24 | t∗d | 0,18 |
| ∗∗∗ | 1,20 | n∗d | 0,37 | ∗ni | 0,24 | ∗∗i | 0,18 |
| ∗de | 0,94 | n∗s | 0,37 | ∗mi | 0,23 | an∗ | 0,18 |
| ie∗ | 0,91 | ∗sc | 0,36 | nn∗ | 0,23 | ver | 0,17 |
| ich | 0,91 | che | 0,35 | ber | 0,23 | n∗a | 0,17 |
| ch∗ | 0,88 | gen | 0,34 | ∗an | 0,23 | ∗es | 0,17 |
| ein | 0,87 | r∗∗ | 0,33 | ∗vo | 0,23 | mit | 0,17 |
| n∗∗ | 0,83 | ∗∗s | 0,32 | ne∗ | 0,22 | ann | 0,17 |
| nd∗ | 0,79 | as∗ | 0,31 | it∗ | 0,22 | ben | 0,17 |
| der | 0,72 | ∗wa | 0,30 | ∗ma | 0,22 | lle | 0,17 |
| te∗ | 0,71 | e∗s | 0,30 | das | 0,21 | ers | 0,17 |
| ∗un | 0,71 | sie | 0,30 | e∗d | 0,21 | nte | 0,17 |
| ∗∗d | 0,69 | ∗∗w | 0,29 | nen | 0,21 | n∗w | 0,16 |
| sch | 0,68 | ∗wi | 0,29 | st∗ | 0,21 | ∗sa | 0,16 |
| und | 0,65 | ∗ha | 0,29 | r∗d | 0,21 | her | 0,16 |
| ∗di | 0,63 | ter | 0,28 | auf | 0,21 | wie | 0,16 |
| die | 0,60 | ach | 0,28 | ∗al | 0,21 | n∗e | 0,16 |
| in∗ | 0,60 | ∗se | 0,28 | nic | 0,21 | ∗he | 0,16 |
| cht | 0,54 | ∗zu | 0,28 | war | 0,21 | on∗ | 0,16 |
| ∗ei | 0,51 | nde | 0,27 | e∗e | 0,21 | e∗a | 0,16 |
| ∗da | 0,51 | em∗ | 0,27 | nge | 0,20 | man | 0,16 |
| ∗si | 0,51 | ∗st | 0,27 | tte | 0,20 | ∗ve | 0,16 |
| t∗∗ | 0,47 | r∗s | 0,26 | sei | 0,20 | uf∗ | 0,16 |
| e∗∗ | 0,47 | ∗∗e | 0,26 | and | 0,19 | och | 0,16 |
| den | 0,43 | ∗we | 0,25 | sic | 0,19 | ren | 0,15 |
| ∗er | 0,43 | hen | 0,25 | ∗so | 0,19 | eit | 0,15 |
| ine | 0,42 | ste | 0,25 | ar∗ | 0,18 | zu∗ | 0,15 |
| ∗au | 0,39 | ∗be | 0,25 | lic | 0,18 | sen | 0,15 |
| ∗ge | 0,39 | ∗∗u | 0,24 | abe | 0,18 | | |
| ten | 0,38 | ∗in | 0,24 | ∗ic | 0,18 | | |

Tabelle 11

Trigrammhäufigkeiten in Prozenten (Deutsch, mit Umlauten und zusätzlichen Zeichen)

## Bi-und Digrammtabellen für Englisch ohne Berücksichtigung von Leerzeichen usw.

Leerzeichen, Satzzeichen, Ziffern, Zeilenwechselzeichen usw. wurden überlesen. So kann z.B. ein Paar entstehen aus dem letzten Buchstaben eines Wortes mit dem ersten Buchstaben des folgenden Wortes.

| Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit | Paar | Häufigkeit |
|---|---|---|---|---|---|---|---|
| th | 3,15 | as | 0,67 | ca | 0,44 | wa | 0,32 |
| he | 2,51 | ea | 0,65 | om | 0,44 | ap | 0,31 |
| an | 1,72 | no | 0,65 | em | 0,43 | nc | 0,31 |
| in | 1,69 | ne | 0,64 | ft | 0,42 | sh | 0,30 |
| er | 1,54 | ec | 0,64 | pq | 0,42 | su | 0,30 |
| re | 1,48 | io | 0,63 | si | 0,42 | tr | 0,30 |
| on | 1,45 | rt | 0,63 | us | 0,42 | we | 0,30 |
| es | 1,45 | co | 0,59 | ew | 0,41 | op | 0,29 |
| ti | 1,28 | be | 0,58 | ss | 0,41 | pl | 0,29 |
| at | 1,24 | di | 0,57 | ei | 0,40 | ke | 0,28 |
| st | 1,21 | li | 0,57 | pe | 0,40 | ld | 0,28 |
| en | 1,20 | ra | 0,57 | ac | 0,39 | lo | 0,28 |
| nd | 1,18 | ma | 0,56 | de | 0,39 | mo | 0,28 |
| or | 1,13 | ta | 0,56 | dt | 0,39 | po | 0,28 |
| to | 1,11 | ce | 0,55 | ee | 0,39 | ul | 0,28 |
| nt | 1,10 | ic | 0,55 | il | 0,39 | yo | 0,28 |
| ed | 1,07 | ll | 0,55 | rs | 0,39 | ay | 0,27 |
| is | 1,06 | na | 0,54 | ct | 0,38 | if | 0,27 |
| ar | 1,01 | ro | 0,54 | fo | 0,38 | sw | 0,27 |
| ou | 0,96 | ot | 0,53 | do | 0,37 | mi | 0,26 |
| of | 0,94 | tt | 0,53 | ie | 0,37 | pp | 0,26 |
| te | 0,94 | ve | 0,53 | ni | 0,37 | bu | 0,25 |
| it | 0,88 | ns | 0,51 | os | 0,37 | fe | 0,25 |
| ha | 0,84 | ur | 0,49 | wi | 0,37 | av | 0,24 |
| se | 0,84 | me | 0,48 | ow | 0,36 | sp | 0,24 |
| et | 0,80 | wh | 0,48 | la | 0,34 | ef | 0,23 |
| al | 0,77 | ly | 0,47 | un | 0,33 | go | 0,23 |
| ri | 0,77 | ch | 0,46 | ab | 0,32 | oo | 0,23 |
| ng | 0,75 | el | 0,46 | ds | 0,32 | ht | 0,22 |
| sa | 0,75 | eo | 0,46 | ep | 0,32 | tu | 0,22 |
| hi | 0,72 | ho | 0,46 | ge | 0,32 | | |
| le | 0,72 | da | 0,45 | im | 0,32 | | |
| so | 0,71 | ut | 0,45 | ts | 0,32 | | |

Tabelle 12

Bigrammhäufigkeiten in Prozenten (Englisch, nach O. Phelps Meaker)

| Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit | Tripel | Häufigkeit |
|---|---|---|---|---|---|---|---|
| the | 3,53 | ted | 0,44 | whi | 0,28 | wer | 0,24 |
| ing | 1,11 | ain | 0,43 | ove | 0,28 | ome | 0,24 |
| and | 1,02 | est | 0,42 | tin | 0,28 | een | 0,24 |
| ion | 0,75 | man | 0,40 | ast | 0,28 | lar | 0,24 |
| tio | 0,75 | red | 0,40 | der | 0,28 | les | 0,24 |
| ent | 0,73 | thi | 0,40 | ous | 0,28 | san | 0,24 |
| ere | 0,69 | ive | 0,38 | rom | 0,28 | ste | 0,24 |
| her | 0,68 | rea | 0,38 | ven | 0,28 | any | 0,23 |
| ate | 0,66 | wit | 0,37 | ard | 0,28 | art | 0,23 |
| ver | 0,64 | ons | 0,37 | ear | 0,28 | nte | 0,23 |
| ter | 0,63 | ess | 0,36 | din | 0,27 | rat | 0,23 |
| tha | 0,62 | ave | 0,34 | sti | 0,27 | tur | 0,23 |
| ati | 0,59 | per | 0,34 | not | 0,27 | ica | 0,23 |
| for | 0,59 | ect | 0,33 | ort | 0,27 | ich | 0,23 |
| hat | 0,55 | one | 0,33 | tho | 0,26 | nde | 0,23 |
| ers | 0,54 | und | 0,33 | day | 0,26 | pre | 0,23 |
| his | 0,52 | int | 0,32 | ore | 0,26 | enc | 0,22 |
| res | 0,50 | ant | 0,32 | but | 0,26 | has | 0,22 |
| ill | 0,47 | hou | 0,31 | out | 0,25 | whe | 0,22 |
| are | 0,47 | men | 0,30 | ure | 0,25 | wil | 0,22 |
| con | 0,46 | was | 0,30 | str | 0,25 | era | 0,22 |
| nce | 0,45 | oun | 0,30 | tic | 0,25 | lin | 0,22 |
| all | 0,44 | pro | 0,30 | ame | 0,24 | tra | 0,22 |
| eve | 0,44 | sta | 0,30 | com | 0,24 | | |
| ith | 0,44 | ine | 0,29 | our | 0,24 | | |

Tabelle 13

Trigrammhäufigkeiten in Prozenten (Englisch, aus [4])

# Literaturverzeichnis


[1] *Adleman, L.:* A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography. Proc. IEEE 20th Annual Symp. on Found. of Comp. Sci. 1979, 55–60.

[2] *Bach, E.:* How to generate factored random numbers. SIAM J. Computing **12** (1988), 179–193.

[3] *Barany, I. und und Füredi, Z.:* Mental Poker with Three or More Players. Technical Report, Mathematical Institute of the Hungarian Academy of Sciences 1983.

[4] *Bauer, F. L.:* Kryptologie. Springer, Berlin 1993.

[5] *Beker, H. und Piper, F.:* Cipher Systems. Northwood Books, London 1982.

[6] *Biham, E. und Shamir, A.:* Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology **4** (1991), 3-72.

[7] *Biham, E. und Shamir, A.:* Differential Cryptanalysis of the Full 16-round DES. In: Advances of Cryptology CRYPTO '92, Lecture Notes in Computer Science **740**, Springer, New York 1992, 487–496.

[8] *Blakley, B. und Blakley, G. R.:* Security of Number Theoretic Public Key Cryptosystems Against Random Attack. Cryptologia **2** (1978), 305–321, **3** (1979), 29–42, **3** (1979), 105–118.

[9] *Blakley, G. R. und Borosh, I.:* Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages. Comp. & Math. with Applic. **5** (1979), 169–178.

[10] *Blum, M.:* Three Applications of the Oblivious Transfer: 1. Coin Flipping by Telephone, 2. How to Exchange Secrets, 3. How to Send Certified Electronic Mail. Dept. EECS, Univ. of California, Berkeley, Calif. 1981.

[11] *Blum, M. und Rabin, M. O.:* How to Send Certified Electronic Mail. Dept. EECS, Univ. of California, Berkeley, Calif. 1981.

[12] *Bressoud, D. M.:* Factorization and Primality Testing. Springer, New York 1989.

[13] *Cook, S. A.:* The complexity of theorem-proving procedures. Proc. 3rd Ann. ACM Symp. on Theory of Computing, ACM, New York 1971, 151–158.

[14] *Davida, G. I., Wells, D. L. und Kam, J. B.:* A Database Encryption Scheme with Subkeys. TR-CS-78-8, Dept. of Electrical Eng. and Computer Science, The Univ. of Wisconsin, Milwaukee, Wis. 1979.

[15] *Denning, D. E.:* Cryptography and Data Security. Addison-Wesley, Reading, Mass. 1982.

[16] *Dickson, L. E.:* Einführung in die Zahlentheorie. Teubner, Leipzig 1931.

[17] *Diffie, W.:* Cryptograhic Technology: Fifteen Year Forecast. BNR Inc., Mountain View, Calif. 1981.

[18] *Diffie, W. und Hellman, M.:* New Directions in Cryptography. IEEE Trans. on Info. Theory **IT-22** (1976), 644–654.

[19] *Diffie, W. und Hellman, M.:* Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer **10** (1977), 74–84.

[20] *Diffie, W., Strawczynski, L., O'Higgins, B. und Steer, D.:* An ISDN secure telephone unit. In: Proceedings, National Communications Forum 1987, Rosemont, IL, Sept. 28–30, **41**, book 1, National Engineering Consortium, Chicago 1987, 473–477.

[21] *Even, S., Goldreich, O. und Lempel, A.:* A Randomized Protocol for Signing Contracts. Comm.ACM **28** (1985), 637–647.

[22] *Fiat, A., Shamir, A.:* How to prove yourself: practical solutions to identification and signature problems. Proceedings of CRYPTO 86. Lecture Notes in Computer Science **263**, Springer, New York 1987, 186–194.

[23] *Fortune, S. und Merritt, M.:* Poker Protocols. In: Advances in Cryptology. Proceedings of CRYPTO 84. Lecture Notes in Computer Science **196**, Springer, New York 1985, 454–464.

[24] *Franz, E. und Pfitzmann, A.:* Einführung in die Steganographie und Ableitung eines neuen Stegoparadigmas. Informatik-Spektrum **21**(1998), 183–193.

[25] *Friedman, W. F.:* Methods for the Solution of Running Key Ciphers. Riverbank Publication **16**, Riverbanks Labs, Geneva, Ill. 1918.

[26] *Friedman, W. F.:* The Index of Coincidence and Its Applications in Cryptography. Riverbank Publication **22**, Riverbank Labs, Geneva, Ill. 1920.

[27] *Fumy, W. und Rieß, P.:* Kryptographie. Entwurf, Einsatz und Analyse symmetrischer Kryptoverfahren. Oldenbourg, München 1994.

[28] *Gaines, H. F.:* Cryptanalysis. Dover, New York 1956.

[29] *Garey, M. R. und Johnson, D. S.:* Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman and Company, San Francisco 1979.

[30] *General Services Administration:* Telecommunications: Compatibility Requirements for Use of the Data Encryption Standard. Proposed Federal Standard 1026, General Services Administration Washington, D.C. 1977.

[31] *Goldwasser, Sh. und Micali, S.:* Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information. In: Proc. 14th Annual ACM Symp. on Theory of Computing 1982, 365–377.

[32] *Goldwasser, Sh., Micali, S. und Rackoff, C.:* The knowledge complexity of interactive proof-systems. SIAM Journal of Computing **18**(1989), 186–208.

[33] *Goldwasser, S., Micali, S. und Rivest, R.:* A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Computing **17** (1988), 281–308.

[34] *Goldwasser, S., Micali, S. und Wigderson, A.:* Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In: Proceedings of FOCS 86, 174–187.

[35] *Goldwasser, S., Micali, S. und Yao, A.:* Strong signature schemes. In: Proc. 15th Ann. ACM Symp. on Theory of Computing 1983, 431–439.

[36] *Guillou, L. C., Quisquater, J.-J.:* A paradoxical identity-based signature scheme resulting from zero-knowledge. In: CRYPTO 88, Lecture Notes in Computer Science **403**, Springer, New York 1990, 216–231.

[37] *Hammer, C.:* High Order Homophonic Ciphers. Cryptologia **5** (1981), 231–242.

[38] *Hellman, M. E.:* An Extension of the Shannon Theory Approach to Cryptography. IEEE Trans. on Info. Theory **IT-23** (1977), 289–294.

[39] *Hill, L. S.:* Cryptography in an Algebraic Alphabet. Am. Math. Monthly **36** (1929), 306–312.

[40] *Hoffman, L. J.:* Modern Methods for Computer Security and Privacy. Prentice-Hall, Englewood Cliffs 1977.

[41] *Hornfeck, B.:* Algebra. de Gruyter, Berlin 1969.

[42] *Horster, P.:* Kryptologie. Reihe Informatik 47. BI Wissenschaftsverlag, Mannheim 1985.

[43] *Kahn, D.:* The code breakers. Macmillan, New York 1967.

[44] *Kari, J.:* A crytanalytic observation concerning systems based on language theory. Discrete Applied Mathematics **21** (1988), 265–268.

[45] *Kari, J.:* Observations concerning a public-key cryptosystem based on iterated morphisms. Theoretical Computer Science **66** (1989), 45–53.

[46] *Kasiski, F. W.:* Die Geheimschriften und die Dechiffrirkunst. Mittler & Sohn 1863.

[47] *Knuth, D.:* The Art of Computer Programming, Volume B / Seminumerical Algorithms, Second Edition. Addison-Wesley, Reading 1980.

[48] *Koblitz, N.:* A Course in Number Theory and Cryptography. Springer, New York 1987.

[49] *Konfelder, L. M.:* On the Signature Reblocking Problem in Public-Key Cryptosystems. Comm. ACM **21** (1978).

[50] *Kranakis, E.:* Primality and Cryptography. Teubner - Wiley, Stuttgart - Chichester 1986.

[51] *Lempel, A.:* Cryptology in Transition. Computing Surveys **11** (1979), 285–303.

[52] *Lenstra, H. W.:* Integer Programming with a Fixed Number of Variables. Mathematics of Operations Research **8** (1983), 538–548.

[53] *Lipton, R. J.:* How to Cheat at Mental Poker. Comp. Sci. Dept., Univ. of Calif., Berkeley, Calif. 1979.

[54] *Lipton, R. J.:* An Improved Power Encryption Method. Comp. Sci. Dept., Univ. of Calif., Berkeley, Calif. 1979.

[55] *MasterCard, VISA:* SET Secure Electronic Transaction Specification, Book 1: Business Description, Book 2: Programmar's Guide, Book 3: Formal Protocoll Definitions. Version 1.0, May 31, 1997 (http://www.setco.org/set_specifications.html).

[56] *Merkle, R. C. und Hellman, M. E.:* Hiding Information and Signatures in Trapdoor Knapsacks. IEEE Trans. on Info. Theory **IT-24** (1978), 525–530.

[57] *Micali, S., Shamir A.:* An improvement of the Fiat-Shamir identification and signature scheme. In: CRYPTO 88, Lecture Notes in Computer Science **403**, Springer, New York 1990, 244–247.

[58] *Naor, M. und Yung, M.:* Universal one-way hash functions and their cryptographic applications. In: Proc. 21st Ann. ACM Symp. on Theory of Computing 1989, 33–43.

[59] *National Bureau of Standards:* Data Encryption Standard, FIPS PUB **46**, National Bureau of Standards, Washington, D.C. 1977.

[60] *Niemi, V. und Renvall, A.:* Cryptographic protocols and voting. In: Results and Trends in Theoretical Computer Science. Lecture Notes in Computer Science **812**, Springer, New York 1994, 307–316.

[61] *Nurmi, H.:* Cryptographic Protocols for Auctions and Bargaining. In: Results and Trends in Theoretical Computer Science. Lecture Notes in Computer Science **812**, Springer, New York 1994, 317–324.

[62] *Nurmi, H. und Salomaa, A.:* A Cryptographic Approach to the Secret Ballot. Behavioral Science **36** (1991), 34–40.

[63] *Nurmi, H. und Salomaa, A.:* Cancellation and reassignment of votes in secret ballot elections. European Journal of Political Economy **9** (1993), 427–435.

[64] *Nurmi, H., Salomaa, A. und Santean, L.:* Secret Ballot Elections in Computer Networks. Computers & Security **10** (1991), 553–560.

[65] *OConner, L. J. und Seberry, J.:* Cryptographic Significance of the Knapsack Problem. Aegean Park Press, Laguna Hills, Calif. 1988.

[66] *O'Higgins, B., Diffie, W., Strawczynski, L. und de Hoog, R.:* Encryption and ISDN — a natural fit. In: Proceedings International Switching Symp., Phoenix, AZ, March 16–20, 1987, A11.4.1–7.

[67] *Ohta, K., Okamoto, T.:* A modification of the Fiat-Shamir scheme. In CRYPTO 88, Lecture Notes in Computer Science, Springer **403**, New York 1990, 232–243.

[68] *Peleg, S. und Rosenfeld, A.:* Breaking Substitution Ciphers Using a Relaxation Algorithm. CACM **22** (1979), 598–605.

[69] *Pohlig, S. und Hellman, M.:* An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance. IEEE Trans. on Info. Theory **IT-24** (1978), 106–110.

[70] *Rabin, M. O.:* Probabilistic Algorithms. In: J. F. Traub (Ed.): Algorithms and Complexity - New Directions and Recent Results. New York 1976. 21–39.

[71] *Rabin, M. O.:* Digitalized Signatures and Public-Key Functions as Intractable as Factorization. MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, Mass., 1979.

[72] *Rabin, M. O.:* Probabilistic Algorithms for Primality Testing. Journal of Number Theory **12** (1980), 128–138.

[73] *Rabin, M. O.:* Exchange of Secrets. Dept. of Applied Physics, Harvard Univ., Cambridge, Mass. 1981.

[74] *Rademacher, H.:* Lectures on Elementary Number Theory. Blaisdell, New York 1964.

[75] *Renvall, A.:* ANDOS: A simple protocol for secret selling of secrets. EATCS Bulletin **47** (1992), 200–205.

[76] *Rivest, R. L.:* Remarks on a Proposed Crytanalytic Attack of the M.I.T. Public Key Cryptosystem. Cryptologia **2** (1978), 62–65.

[77] *Rivest, R. L.:* A Description of a Single-Chip Implementation of the RSA Cipher. Lambda **1** (1980), 14–18.

[78] *Rivest, R. L., Shamir, A. und Adleman, L.:* A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Comm. ACM **21** (1978), 120–126.

[79] *Rothkopf, M. H., Teisberg, Th. J. und Kahn, E. P.:* Why Are Vickrey Auctions Rare? Journal of Political Economy **98** (1990), 94–109.

[80] *Rivest, R. L., Shamir, A. und Adleman, L.:* Mental Poker. MIT Technical Report 1978.

[81] *Salomaa, A.:* Trapdoors and protocols: recent trends in cryptography. In: *H. Maurer (Ed.):* Überblicke Informationsverarbeitung 1984, BI Wissenschaftsverlag, Mannheim 1984, 275–320.

[82] *Salomaa, A.:* Computation and Automata. Chapter 7. Cambridge University Press, Cambridge 1985.

[83] *Salomaa, A.:* Public-Key Cryptography. Springer, Berlin 1990.

[84] *Salomaa, A. und Welzl, E.:* A cryptographic trapdoor based on iterated morphisms. Manuskript 1983.

[85] *Salomaa, A. und Yu, Sh.:* On a public-key cryptosystem based on iterated morphisms and substitutions. Theoretical Computer Science **48** (1986), 283–296.

[86] *Schneier, B.:* Angewandte Kryptographie. Protokolle, Algorithmen und Sourcecode in C. Addison-Wesley, Bonn 1996.

[87] *Schroeppel, R. und Shamir, A.:* A $TS^2 = O(2^n)$ Time/Space Tradeoff for certain NP-Complete Problems. Proc. IEEE 20th Ann. Symp. on Found. of Computer Science 1979.

[88] *Schuster, R., Färber, J. und Eberl, M.:* Digital Cash. Zahlungssysteme im Internet. Springer, Berlin 1997.

[89] *Sedlak, H.:* Konzept und Entwurf eines Public-Key-Code Kryptographie-Prozessors. Diplomarbeit, TU Braunschweig 1985.

[90] *Sedlak, H.:* The RSA cryptography processor. EUROCRYPT 87, Amsterdam 1987, Proceedings, Lecture Notes in Computer Science 304.

[91] *Sedlak, H. und Golze, U.:* An RSA cryptography processor. Microprocessing and Microprogramming **18** (1986), 583–590.

[92] *Shamir, A.:* A Fast Signature Scheme. MIT/LCS/TM-107, MIT Lab. for Computer Science, Cambridge, Mass. 1978.

[93] *Shamir, A.:* A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. Proc. 23rd IEEE Symp. on Found. of Computer Science 1982, 145–152.

[94] *Shamir, A., Rivest, R. L. und Adleman, L.:* Mental Poker. In: D. E. Klarner (ed.): Mathematical Gardner, Wadsworth International 1981, 37–43.

[95] *Shannon, C. E.:* A Mathematical Theory of Communication. Bell Syst. Tech. J. **27** (1948), 379–423 und 623–656.

[96] *Shannon, C. E.:* Communication Theory of Secrecy Systems. Bell Syst. Tech. J. **28** (1949), 656–715.

[97] *Simmons, G. J. (Ed.):* Contemporary Cryptology. The Science of Information Integrity. IEEE Press, New York 1992.

[98] *Sinkov, A.:* Elementary Cryptanalysis. Math. Assoc. Am. 1966.

[99] *Stinton, D. R.:* Cryptography: Theory and Practice. CRC Press, Boca Raton, Florida 1995.

[100] *Tedrick, T.:* Fair Exchange of Secrets (extended abstract). In: CRYPTO 1984. Lecture Notes in Computer Science **196** (1984), 434–438.

[101] *Tuchman, W.:* Hellman Presents No Shortcut Solution to the DES. IEEE Spectrum **16** (1979), 40–41.

[102] *Vickrey, W.:* Counter Speculation, Auctions and Competitive Sealed Tenders. Journal of Finance **16**,8–37.

[103] *Wobst, R.:* Abenteuer Kryptologie. Methoden, Risiken und Nutzen der Datenverschlüsselung. Addison-Wesley, Bonn 1997.

[104] *Yao, A. C.-C:* Protocols for Secure Computations. Proc. of the 23rd IEEE Symp. on Found. of Computer Science 1982, 160–164.

# Index


absolut sicher, 7
absolute Dichte, 5
absolute Rate, 5
Äquivokation, 6
Alberti, Leon Battista, 22
Altersvergleich-Protokoll, 93
anagraming, 16
ANDOS-Protokoll, 103
asymmetrische Chiffre, 4
Auktionen, 95
Authenzitätsanforderungen, 3

Beaufort-Chiffre, 24
  Variante, 24
bedingte Entropie, 6
Blockchiffre, 140
  Chiffre-Block-Chaining, 142
  mit Teilschlüsseln, 142

Cäsar-Chiffre, 2, 11, 13, 17
Chiffre, 1
  uneingeschränkt sicher, 2
  asymmetrische, 4
  Beaufort-Chiffre, 24
    Variante, 24
  berechnungs-sicher, 2
  Blockchiffre, 140
  Cäsar-Chiffre, 2, 11, 13, 17
  DES, 34–43
  durch affine Transformation, 17, 19
  durch polynomiale Transformation, 18
  einfache Substitution, 16
  Exponentiationschiffre, 60
    Pohlig-Hellman-Schema, 61
    RSA-Schema, 62
  Friedhofchiffre, 18
  Hill-Chiffre, 33
  homophone Substitution, 16, 21
  Merkle-Hellman-Knapsack, 127–128
  mit laufendem Schlüssel, 30
  mit periodischer Substitution, 22
  mit T0L-System, 137–140
  Playfair-Chiffre, 32
  Pohlig-Hellman-Schema, 61
  polyalphabetische Substitution, 16, 22
  Polygramm- Substitution, 16, 32
  Produktchiffre, 34
  RSA-Schema, 62
  Shamir-Knapsack, 131–133
  Spaltentransposition, 15
  Substitutionschiffre, 1, 10
  symmetrische, 3
  Transpositionschiffre, 1, 15
  Vernam-Chiffre, 32
  Verschiebechiffre, 17
  Vigenère-Chiffre, 23
  Zaun, 1
  zufällige, 12, 14
Chiffre-Block-Chaining, 142
Chiffretext, 1
Chiffretextraum, 2
Chiffriertransformation, 2
chinesischer Restesatz, 54, 116, 142

Dechiffriertransformation, 2
DES, 13, 34–43
Dichte, 5
digitale Signatur, 58, 106
duale Signatur, 106

Eindeutigkeitsdistanz, 11, 16, 19, 24
einfache Substitution, 16
einfacher Knapsack, 125
  Lösung, 126
Einschreiben, Protokoll für, 86
Einweg-Hahfunktion
  SHA, 106
Einweg-Hashfunktion, 105
Entropie, 4
  bedingte, 6
Entscheidungsproblem, 115
Erste-Preis-Auktion (Protokoll), 96
Euklidischer Algorithmus, 51
Eulersche Funktion, 49
Exponentiation, schnelle, 47
Exponentiationschiffre, 60

Pohlig-Hellman-Schema, 61
RSA-Schema, 62

Fermat, Satz von, 50
Fingerabdruck, 106
Friedhofchiffre, 18

Geheimes Verkaufen von Geheimnissen (Protokoll), 103
Geheimhaltungsanforderungen, 3
Geschäftsabschluß, 98
Graph-Isomorphismus-Problem, 124

Häufigkeitsverteilung (Tabellen), 20, 147–158
Hashfunktion, 105
SHA, 106
Hill-Chiffre, 33
homophone Substitution, 16, 21

Information Protection System, 141
Informationsgehalt, 4
Inversenbildung, 51–52
ISDN-Telefone, sichere, 92

Kasiski-Methode, 28–30
Klartext, 1
Klartextraum, 2
Knapsack-Problem, 115
0-1-Knapsack, 125
einfacher Knapsack, 125
Lösung, 126
Trapdoor-Knapsack, 126
Koinzidenzindex, 26–28
Komplexitätsklassen
$NP$, 115
$P$, 115
Komplexitätstheorie, 115
kongruente Zahlen, 45
Kopf-Poker-Protokoll, 72
Kryptoanalysis, 2
Kryptoanalytiker, 2
Kryptographie, 1
kryptographisches System, *siehe* Kryptosystem
Kryptologie, 2
Kryptosystem, 2
Authenzitätsanforderungen, 3
Chiffretextraum, 2
Chiffriertransformation, 2
Dechiffriertransformation, 2
DES, 13
Ein-Schlüssel-Kryptosysteme, 3
Geheimhaltungsanforderungen, 3
Klartextraum, 2
Public-Key-Kryptosystem, 2, 57–60
Authenzität, 57, 68
brechbarer vollständiger Knapsack, 134
Brechen des Merkle-Hellman-Knapsacks, 128–131
Geheimhaltung, 57, 68
Geheimhaltung und Authentizität, 58, 68
Merkle-Hellman-Knapsack zur Geheimhaltung, 127–128
Shamir-Knapsack zur Authentizität, 131–133
Schlüsselraum, 2
Zwei-Schlüssel-Kryptosysteme, 3

Merkle-Hellman-Knapsack, 127–128
message digest, 106
modulare Arithmetik, 46

$\mathbb{N}$, Menge der natürlichen Zahlen, 45
$\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, 45
$NP$, Komplexitätsklasse, 115
$NP$-vollständig, 116

Oblivious-Transfer-Protokoll, 75
allgemeines, 84
1-von-2, 81
one-time-pad, 31

$P$, Komplexitätsklasse, 115
perfekte Geheimhaltung, 7, 9
Playfair-Chiffre, 32
Pohlig-Hellman-Schema, 61
polyalphabetische Substitution, 16, 22
Polygramm-Substitution, 16, 32
primitive Wurzel, 89
Primzahlen, Suche großer, 64
Produktchiffre, 34
Protokoll
ANDOS, 103

digitale Signatur, 106
duale Signatur, 106
Erste-Preis-Auktion, 96
für sichere ISDN-Telefone, 92
für Wahlen
Änderung der Stimme, 102
Stimmabgabe, 99
Verbesserung von falschen Zuordnungen, 101
Geheimes Verkaufen von Geheimnissen, 103
Geschäftsabschluß, 98
Kopf-Poker, 72
Oblivious-Transfer-Protokoll, 75
allgemeines, 84
1-von-2, 81
Schlüsselaustausch, 90
Schlüsselverteilung, 91–92
SET
*payment authorization*, 110
*payment capture*, 111
*purchase request*, 108
Vertragsunterzeichnungsprotokoll, 85
Werfen von Münzen, 80
Zero-Knowledge-Protokoll
zum Nachweis der Kenntnis einer Quadratwurzel, 117
zum Nachweis eines Graphisomorphismus, 124
zur Dreifärbung eines Graphen, 120
zur Identifikation, 119
zum Altersvergleich, 93
zum Versenden von Einschreiben, 86
Zweite-Preis-Auktion, 97
Public-Key-Kryptosystem, 2, 57–60
Authenzität, 57, 68
brechbarer vollständiger Knapsack, 134
Brechen des Merkle-Hellman-Knapsacks, 128–131
Geheimhaltung, 57, 68
Geheimhaltung und Authentizität, 58, 68
Merkle-Hellman-Knapsack zur Geheimhaltung, 127–128
mit T0L-System, 137–140
RSA-Schema, 62, 68–69
Shamir-Knapsack zur Authentizität, 131–133

quadratische Kongruenz, 116

Rate, 5
Redundanz, 5
reduzierte Menge der Reste modulo $n$, 49
Replay, 140
Rest einer Zahl, 45
quadratischer Rest modulo $n$, 72
reduzierte Menge der Reste modulo $n$, 49
Restklasse modulo $n$, 45
Restesatz, chinesischer, 54, 116, 142
Restklasse modulo $n$, 45
RSA-Schema, 62

Satz von Fermat, 50
Schlüssel, 1
Schlüsseläquivokation, 11
Schlüsselaustauschprotokoll, 90
Schlüsselaustauschsystem, 90
Schlüsselraum, 2
Schlüsselverteilung, Protokoll, 91–92
schnelle Exponentiation, 47
SET, 108–113
*payment authorization*-Protokoll, 110
*payment capture*-Protokoll, 111
*purchase request*-Protokoll, 108
Shamir-Knapsack, 131–133
Signatur, 106
duale, 106
Spaltentransposition, 15
Steganographie, ii
Substitutionschiffre, 1, 10
einfache, 16
homophone, 16, 21
periodische, 22
polyalphabetische, 16, 22
Polygramm-, 16, 32
Suche großer Primzahlen, 64
symmetrische Chiffre, 3

Transpositionschiffre, 1, 15
Trapdoor-Knapsack, 126

Vernam-Chiffre, 32

Verschiebechiffre, 17
Vertragsunterzeichnungsprotokoll, 85
Vigenère-Chiffre, 23
Vigenère-Tableau, 25

Wahlprotokoll
    Änderung der Stimme, 102
    Stimmabgabe, 99
    Verbesserung von falschen Zuordnungen, 101
Wurzelberechnung einer quadratischen Gleichung, 76–80

Zaun, 1
Zero-Knowledge-Protokoll
    zum Nachweis der Kenntnis einer Quadratwurzel, 117
    zum Nachweis eines Graphisomorphismus, 124
    zur Dreifärbung eines Graphen, 120
    zur Identifikation, 119
Zertifikat, 107
Zertifikate, 108
$\mathbb{Z}_n$, Restklasse modulo $n$, 45
zufällige Chiffre, 12, 14
Zweite-Preis-Auktion (Protokoll), 97