Algorithmentheorie

Leibniz (1646-1716):
Es kann ein Algorithmus gefunden werden, mit dessen Hilfe jede beliebige Wahrheit ableitbar ist (ars inveniendi), oder ein Verfahren entwickelt werden, anhand dessen von jeder Aussage entschieden werden kann, ob sie wahr oder falsch ist (ars iudicandi).

Hilbert (1900):
"… ein jedes bestimmtes mathematisches Problem einer strengen Erledigung fähig sein müsse, sei es, dass es gelingt, die Beantwortung der gestellten Frage zu geben, sei es, dass die Unmöglichkeit seiner Lösung … dargetan wird …"

Kurt Gödel 1931:
Ein Algorithmus im Hilbertschen (Leibnizschen) Sinne kann nicht existieren.

Algorithmus
Ein Algorithmus ist - intuitiv gesehen - eine Anleitung zur Lösung eines Problems.
"Algorithmus" = "Rechnen nach Art des Algorismi", benannt nach dem arabischen Mathematiker Al Khwarizmi
Abbildung: Werk Al Khwarizmis

Schema eines Algorithmus
Präzisierung des Algorithmen-Begriffs
intuitive Vereinfachung: Ein Verfahren heißt genau dann Algorithmus, wenn es in PASCAL programmiert werden kann. ...bisher kein Gegenbeispiel bekannt

Englisch	algorithm	Holländisch	algoritme	Französisch	algorithme
Russisch	алгоритм	Spanisch	algoritmo	Italienisch	algoritmo, procedura
Dänisch	algoritme	Griechisch	αλγοριΫμοШ	Esperanto	algoritmo
Polnisch	algorytm	Schwedisch	algoritm	Slowenisch	algoritmo
Tschechisch	algoritmus	Finnisch	algoritmi	Portugiesisch	algoritem
Ungarisch	algoritmus	Vietnamesisch	thuật toán		

Nicht berechenbare Probleme
- es existiert kein Algorithmus, der es erlaubt, jedes gegebene Pascal-Programm in ein äquivalentes, aber bzgl. Laufzeit oder Speicherplatzbedarf optimiertes zu überführen

Allgemeines Halteproblem
- es gibt keinen Algorithmus, mit dessen Hilfe festgestellt werden kann, ob ein beliebiges Pascal-Programm terminiert oder nicht

Allgemeines Wortproblem
- es gibt keinen Algorithmus, der es für beliebige formale Sprachen erlaubt festzustellen, ob ein beliebiges Wort über dem Alphabet der Sprache zur Sprache gehört oder nicht
für kontextfreie Sprachen, z.B. Pascal, gibt es solche Algorithmen, sonst könnte der Compiler keine Syntaxanalyse durchführen.

Churchsche These: Jede im intuitiven Sinne berechenbare Funktion ist auch Turing-berechenbar.

Gödelsches Unvollständigkeitstheorem
Der Gödelsche Unvollständigkeitssatz ist einer der wichtigsten Sätze der modernen Logik. Er beschäftigt sich mit der Ableitbarkeit von Aussagen in formalen Theorien. Der Satz zeigt die Grenzen der formalen Systeme ab einer bestimmten Mächtigkeit auf und weist nach, dass man in Systemen wie der Arithmetik nicht alle Aussagen formal beweisen oder widerlegen kann.
1931 bewies Gödel:

Jedes hinreichend mächtige formale System ist entweder widersprüchlich oder unvollständig.

Dies bedeutet u.a.: Es gibt keinen Algorithmus, welcher eine Aussage über natürliche Zahlen trifft und gleichzeitig den Wahrweitswert dieser Aussage ausgibt. d.h. das Hilbertsche Programm ist nicht(!) realisierbar.
Gödels Argumentation läuft auf eine Abzählung aller Sätze innerhalb des formalen Systems hinaus, jeder Satz erhält eine eigene Nummer. Er konstruiert dann mit Hilfe einer Diagonalisierung eine Aussage der Form: "Der Satz mit der Nummer x ist nicht ableitbar" und zeigt, dass es eine Einsetzung für x gibt, so dass x die Nummer dieser Aussage ist.
Gödel bewies auch den Satz

Ein System kann nicht zum Beweis seiner eigenen Widerspruchsfreiheit verwendet werden.
Die Folge daraus ist, dass man die Korrektheit von formalen Systemen als gegeben annehmen muss, sie lassen sich nicht beweisen.

Eine interessante und relativ leicht verständliche Erklärung von Gödels Satz gibt das Buch "Gödel, Escher, Bach" von Douglas Hofstadter.
Da Gödels Ergebnis für die Mathematik ein überraschendes und entscheidendes Ereignis war, wird der Unvollständigkeitssatz von diversen Philosophen, vor allem Subjektivisten, in die von ihnen gewünschte Richtung hin- und hergedeutet.
Diese pseudowissenschaftlichen Versuche sind einfach nur überflüssig.

Algorithmen-Eigenschaften
Allgemeinheit ... Lösung einer Klasse von Problemen, die Auswahl des Einzelfalls erfolgt über Parameter
Determiniertheit ... bei gleichen Eingabewerten und Startbedingungen erfolgt stets dasselber Ergebnis
Determinismus ... zu jedem Zeitpunkt besteht höchstens eine Möglichkeit der Fortsetzung
Terminierung ... nach endlich vielen Schritten hält der Algorithmus für jede Eingabe an
Endlichkeit(Finitheit) ... die Beschreibung eines Algorithmus besitzt endliche Länge (statistische Endlichkeit). Bei der Abarbeitung des Algorithmus entstehende Datenstrukturen und Zwischenergebnisse sind endlich.
Prinzipielle Machbarkeit (Effektivität) ... Jeder Schritt des Algorithmus muss von der zugrundeliegenden Verarbeitungseinheit ausführbar sein (Rechner, Mensch)
Präzises Anweisen ... die elementaren Bestandteile eines Algorithmus heißen Schritte. Die Algorithmen steuern den Ablauf der einzelnen Schritte. Die Schritte und ihre Aufeinanderfolgung müssen unmissverständlich sein. Solche Schritte stellen primitive (relativer Begriff) Grundoperationen dar.

Zeichen, Symbol ... Sinnbild, Element einer vereinbarten Menge, dem eine vereinbarte Bedeutung zukommt.
Grundsymbol ... Symbol, auf das die anderen Symbole einer Menge zurückgeführt werden können.

Alphabet
... die vereinbarte Menge wird meist als endlich angenommen und oft als Alphabet bezeichnet.
Besondere Bedeutung für die Mathematik haben die Zeichen für Ziffern zur Darstellung von Zahlen, die Zeichen für Buchstaben zur Darstellung von variablen Größen, die Operationszeichen wie +, -, *, ...,
die Zeichen für Funktoren und Quantoren und die technischen Zeichen wie Klammern usw.

Numerische Zeichen ... sind auf Ziffern beschränkt.
Alphanumerische Zeichen ... umfassen Ziffern, Buchstaben, Operationszeichen und technische Zeichen.
Wort ... Als Wort bezeichnet man jede endliche Folge von Zeichen aus einem Alphabet A.

Zeichenreihe, Zeichenvorrat
... eine endliche Menge von unterschiedlichen Symbolen heißt Zeichenvorrat.
... ein Zeichenvorrat, in dem eine Reihenfolge für Zeichen definiert ist, heißt Zeichenreihe
... eine Zeichenreihe ist eine Aneinanderreihung von beliebigen Zeichen aus dem Zeichenvorrat:

Bewertung von Algorithmen
Wie gut" löst der Algorithmus das Problem ? Ziel: einen möglichst effizienten Algorithmus zu entwickeln
· Größen zur Bewertung der Effizierung: Laufzeit, Speicherplatzbedarf
· Zur Lösung eines Problems kann es verschiedene Algorithmen geben, die ein unterschiedliches Zeit- und Speicherverhalten aufweisen

Modell zur Laufzeitmessung
... Anzahl der dominanten elementaren Operation werden gezählt
· Bewertung der Effizienz eines Algorithmus ist durch die Funktion f(n) gegeben, die abhängig ist von der Größe und des Untersuchungsgegenstandes (meist n = Anzahl der Elemente in der Datenstruktur)

Untersuchungsfälle
· 3 Fälle werden untersucht
- ungünstigster Fall: worst case ; mittlerer Fall: average case ; günstigster Fall: best case
· Durchschnittsanalyse ist meist sehr schwierig mathematisch darzustellen

Wachstumsrate einiger Funktionen

n	$\log_2 n$	n	n^2	2^n	n^n
2	1	2	4	4	4
4	2	4	16	16	256
10	4	10	100	1000	10^{10}

100	7	100	10^4	10^{30}	10^{200}
1000	10	1000	10^6	10^{300}	10^{3000}

Zeitkomplexität

Die Funktion f(x) heißt von der Ordnung g(x), wenn für eine positive reelle Konstante C und fast alle natürlichen Zahlen n $f(n) \leq C\, g(n)$ gilt, d.h. $f(x) = O(g(x))$
Streben die Grenzwerte von f(x) und g(x) asymptotisch gegeneinander, so ist $f(x) = \Theta(g(x))$
Ein Komplexitätsfunktion der Ordnung $O(n^k)$, die sich asymptotisch wie ein Polynom k.Grades verhält, heißt polynomial. Ein Algorithmus heißt schnell, wenn seine Komplexität höchstens $O(\log^k n)$ ist.

Komplexitätsfunktionen

O(1)	konstant	O(log n)	logarithmisch
O(n)	linear	O(n log n)	Ordnung n log n
$O(n^2)$	quadratisch	$O(n^3)$	kubisch
$O(2^n)$	exponentiell	$O(n^n)$	superexponentiell

Spezielle Funktionen und Algorithmen

Primzahlfunktion $\pi(n) = \Theta(n / \log n)$
Fakultätsfunktion $n! = \Theta(n^{n+0.5})$
Primzahl-Testdivisionen ... $O(\sqrt{n})$
Quick-Sort ... O(n*log n)
Fibonacci-Zahlen $F_n = \Theta(\phi^n) = \Theta(1.618034^n)$
Horner-Schema ... O(n)
Bubble-Sort ... $O(n^2)$
Multiplikation quadr.Matrizen ... $O(n^3)$

Rekursive Algorithmen

Rekursion zählt zu den wichtigsten Prinzipien der Informatik

Beispiele

- Mathematik: 1 ist eine natürliche Zahl - der Nachfolger einer natürlichen Zahl ist wieder eine natürliche Zahl
- Theorie der Programmiersprache: ausdruck = ausdruck + term term := term * faktor

Merkmale rekursiver Algorithmen

... unendliche Menge von Objekten kann durch eine endliche Aussage definiert werden, d.h. es kann unendliche Menge von Berechnungen durch einen endlichen rekursiven Algorithmus beschrieben werden; ohne dass Algorithmus Schleifen enthält

Backtracking Algorithmen

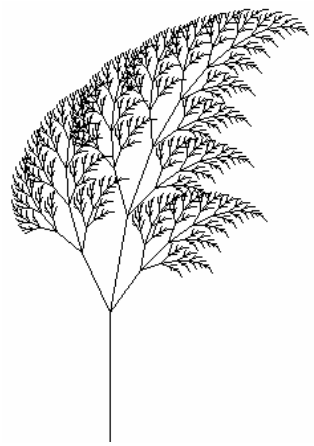
Das Problem ist nicht direkt durch eine Berechnungsvorschrift gelöst, sondern durch Versuch und Nachprüfen (trial und error). Man versucht in Teilschritten zum Ziel zu kommen, Schritte werden aufgezeichnet, falls man in "Sackgasse" kommt, geht man Schritte zurück und löscht Aufzeichnung wieder (backtracking). Die Teilschritte lassen sich meist rekursiv lösen.

Greedy Algorithmus

Als "Greedy" (= gierig) bezeichnet man Methoden, die direkt auf das Ziel zusteuern. Allerdings muss dies aber nicht unbedingt die beste Strategie sein. Zu den Greedy-Algorithmen gehört zum Beispiel das Sortieren durch direkte Auswahl.

Rekursion

Ein Algorithmus heißt rekursiv, wenn er sich selbst mindestens einmal aufruft. (rekursiv (lat.) : zurücklaufend)
Beispiele:
1) Berechne die Summe s_n der ersten n Zahlen!
Lösung: Wenn n = 0, dann summe := 0 sonst summe := n + summe(n-1)
2) Berechne die Quadratzahl ohne zu multiplizieren!
Lösung: Wenn n = 0, dann quad := 0 sonst quad := quad(n-1) + n + n - 1
3) Berechne das Produkt a · b rekursiv!
Lösung: Wenn a = 0, dann prod := 0 sonst prod := prod(a-1, b) + b
4) Bestimme die Quersumme einer Zahl z rekursiv!
Lösung: Wenn z<10 dann quer := z sonst quer := z mod 10 + quer(z div 10)
5) Berechne den Binomialkoeffizienten $\binom{n}{k}$ rekursiv!
Lösung: Wenn k = 0 oder n = k dann bin := 1 sonst bin := bin(n-1, k-1) + bin(n-1, k)
6) Berechne die Fibonacci-Zahlen rekursiv!
Lösung: Wenn t<2 dann fibo := 1 sonst fibo := fibo(t-1) + fibo(t-2)
7) Die Summe zweier natürlicher Zahlen x und y soll rekursiv berechnet werden.
Lösung: Wenn x = 0 dann summe := y sonst summe := summe(x-1, y+1)
8) Umwandlung einer Dezimal- in eine Dualzahl rekursiv.
Lösung: Wenn z = 0 dann dual := '0' sonst dual := dual (z DIV 2) + chr(z MOD 2 + 48)
9) Berechnung der Potenz a^n.
Lösung: Wenn n = 0 dann potenz := 1 sonst potenz := a · potenz (a, n-1)



Rekursiver Algorithmus, Beispiel

Mit rekursiven Algorithmen können interessante Grafiken erstellt werden; z.B. die nebenstehende Zeichnung einer Pflanze.
Der Stamm verzweigt sich in drei Äste, von denen sich jeder wieder in 3 Äste verzweigt usw. Die Rekursion wird wie folgt erreicht:
Der erste Zweig (Linie) wird vom Mittelpunkt der Grundkante unter einem Winkel von 90° zur Horizontalen mit einer wählbaren Anfangslänge gezeichnet.
Wurde ein Zweig (Linie) mit der aktuellen Länge und dem aktuellen Winkel (absoluter Winkel bzgl. der Horizontalen auf der Zeichenfläche) vom Anfangspunkt aus gezeichnet, so wird die Prozedur für das Zeichnen eines Zweiges vom Endpunkt aus erneut aufgerufen, nun aber mit den neuen Werten für Länge und Winkel.
Die Rekursion muss beendet werden, wenn die Länge des Zweiges kleiner oder gleich einem Abbruchwert (z.B. 2) geworden ist.

```
procedure farn(len,alpha:real);
var xa,ya,xe,ye : integer;
begin
  with zeichenflaeche.Canvas do begin
    xa := PenPos.X;    ya := PenPos.Y;
    xe := xa+round(len*cos(alpha));    ye := ya-round(len*sin(alpha));
    lineTo(xe,ye);
    if len>2 then begin
      farn(len*fli,alpha+25*pi/180);     farn(len*fmi,alpha-10*pi/180);
      farn(len*fre,alpha-35*pi/180);
    end;
    moveTo(xa,ya);
  end;
end;
```

Ausschöpfungsverfahren, Brute Force Verfahren

Die Brute-Force-Methode (von engl. brute = animalisch, brutal und force = Kraft, Macht; d.h. "Methode der rohen Gewalt"), das Ausschöpfungsverfahren oder die Exhausitionsmethode ist ein Lösungsverfahren für Probleme aus den Bereichen Algebra, Arithmetik, Informatik, Kryptografie …, die auf dem vollständigen Ausprobieren aller oder möglichst vieler Fälle beruht.
Die Reduktion des Begriffes Brute-Force auf das Knacken von Passwörtern ist nicht korrekt.

Für viele Berechnungsprobleme existieren keine effizienten Algorithmen. Der einfachste Ansatz zu einer algorithmischen Lösung eines Problems besteht darin, einfach alle möglichen Lösungen durchzuprobieren, bis die richtige gefunden ist.
Die Brute-Force-Suche ist relativ einfach zu implementieren und soll die korrekte Lösung finden. Mit der Anzahl der zu probierenden, möglichen Lösungen steigt proportional der rechentechnische Aufwand an Rechenoperationen, meist sogar exponentiell.

In der Spieltheorie, beispielsweise beim Computer-Schach, ist die Brute-Force-Methode eine Strategie, in der der Variantenbaum bis zu einer gewissen Tiefe vollständig durchsucht und analysiert wird. Eine Bewertungsfunktion für jede auftretende Stellung ermöglicht die Wahl des besten Zuges.
Der Aufwand für die Brute-Force-Methode wächst hier exponentiell mit der verwendeten Maximaltiefe der Stellungssuche. Damit setzt die Hardware dieser Methode eine natürliche Grenze. Allerdings sind Hochleistungsrechner (2012) mittlerweile so schnell, dass sie dem Menschen überlegen sind.
Ist das Problem selbst zu komplex, können auch Brute Force Verfahren nicht ausreichen, zum Beispiel beim Go-Spiel.

Multiplikation komplexer Zahlen

$(a+ib)*(c+id) = (t_1 - t_3) + i*(t_1 + t_2)$ mit $t_1 = (a+b)*c$, $t_2=a*(d-c)$, $t_3 = b*(d+c)$

Multiplikation von 2x2-Matrizen

$C = A * B$ … $c_{11} = m_1 + m_2 - m_4 + m_6$ $c_{12} = m_4 + m_5$ $c_{21} = m_6 + m_7$ $c_{22} = m_2 - m_3 + m_5 - m_7$
mit $m_1 = (a_{12} - a_{22})(b_{21} + b_{22})$ $m_2 = (a_{11} + a_{22})(b_{11} + b_{22})$ $m_3 = (a_{11} - a_{21})(b_{11} + b_{12})$
$m_4 = (a_{11} + a_{12})\, b_{22}$ $m_5 = a_{11}\, (b_{12} - b_{22})$ $m_6 = a_{22}\, (b_{21} - b_{11})$
$m_7 = (a_{21} + a_{22})\, b_{11}$

Algorithmus von Stein (ggT-Bestimmung)

ggT(a,b) = 2 ggT(a/2,b/2), wenn a,b gerade
ggT(a,b) = ggT(a/2,b), a gerade, b ungerade
ggT(a,b) = ggT/(a-b,b), wenn a≥b

Ägyptische Bauernmultiplikation

x= a; y=b; z=0; Solange x>0 ist, wiederhole wenn x ungerade dann z=z+y
y=2*y x=[x/2] ; ganzzahlige Division
Ergebnis: z=a*b

Multiplikation (2.Form)

x= a; y=b; Solange x>0 ist, wiederhole z=z+y und x=x-1
Ergebnis: z=a*b

Fibonacci-Zahlen

a=0; b=1; Von i=1 bis n wiederhole b=a+b und a=b-a
Ergebnis: a ist die n.te Fibonacci-Zahl

Halbieren	ungerade?	Verdoppeln
27	ja	245
13	ja	490
6	-	980
3	ja	1960
1	ja	3920
Resultat	=	6615

Binäres Multiplizieren

Im Gegensatz zum schriftlichen Multiplikationsverfahren, so wie es heute gelehrt wird, kommt das binäre Multiplizieren völlig ohne Kenntnis des kleinen 1×1 aus.
Es reicht bereits aus, ganze Zahlen verdoppeln und halbieren und gerade von ungeraden unterscheiden zu können. Dieses Verfahren hat sich erfolgreich bis nach Ägypten verbreitet.
Das Verwunderliche an dem Verfahren ist, dass es ohne die Kenntnis binärer Zahlen nicht zu beweisen ist. Die Sumerer und Babylonier müssen ein großes, aus der Praxis gewachsenes Verständnis für die innere Struktur ganzer Zahlen besessen haben. Dieses Verfahren ist auch unter den Namen abessinische Bauernmultiplikation, ägyptische Multiplikation oder russische Bauernmultiplikation bekannt.

Beispiel 27 × 245 = 6615:
1) die beiden gegebenen Faktoren nach Größe sortieren und in die erste Zeile einer Tabelle schreiben
2) den kleineren Faktor ohne Rest halbieren und untereinander notieren, bis man bei der 1 angelangt ist
3) den größeren Faktor solange verdoppeln und untereinander notieren, bis man neben die 1 gelangt ist
4) diejenigen Zeilen, bei denen der Wert in der Halbierungs-Spalte gerade ist, werden ignoriert und gestrichen
5) die verbleibenden Zahlen aus der Verdopplungs-Spalte werden addiert, das Ergebnis enspricht dem Produkt der oberen beiden Zahlen

Karazuba-Algorithmus

Der Karazuba-Algorithmus ist ein Algorithmus zur Multiplikation zweier ganzer Zahlen. Er wurde 1960 von Anatoli A.Karazuba entwickelt. Das Verfahren ist deutlich schneller als die klassische Multiplikation.
Die Ziffernfolgen der Faktoren x und y seien
$x = x_{2n-1}...x_0$ und $y = y_{2n-1}...y_0$ und
Evtl. wird ein Faktor auf 2n-Stellen erweitert. Jede Ziffernfolge wird in zwei Folgen der Länge n aufgespalten. Es ergeben sich vier Zahlen
$x_h = x_{2n-1}...x_n$ $x_l = x_{n-1}...x_0$
$y_h = y_{2n-1}...y_n$ $y_l = y_{n-1}...y_0$
Mit diesen ergeben sich die Produkte
$a = x_h\, y_h$ $b = x_l\, y_l$
$c = (x_h + x_l)\,(y_h + y_l)$
Für das Produkt erhält man
$xy = a\, 10^{2n} + (c - a - b)\, 10^n + b$
Beste Laufzeitergebnisse ergeben sich bei einer rekursiven Umsetzung des Verfahrens. Statt in zwei Teile, können die zu multiplizierenden Zahlen auch in mehr Teile zerlegt werden. Rekursiv angewandt führt dieses Verfahren zum Toom-Cook-Algorithmus.
Auf der rechten Seite wird der Karazuba-Algorithmus demonstriert. Die Zahlen x_l, x_h, y_l, y_h, a, b und c werden als Binärzahlen verwaltet, d.h. der Algorithmus im Dualsystem mit
$xy = a\, 2^{2m} + (c - a - b)\, 2^m + b$
verwendet.

Wurzelziehen nach Heron

x=a; y=1; Wenn a<1 dann x,y tauschen Solange x>y ist, wiederhole x=(x+y)/2 und y=a/x
Ergebnis: $x = \sqrt{a}$

Euklidischer Algorithmus zur ggt-Bestimmung

x=a; y=b; Solange y>0 ist, wiederhole r=x mod y und x=y und y=r
Ergebnis x

Schnelles Potenzieren von a^b

x=a; y=b; z=1; Solange y>0 ist, wiederhole
Wenn Y ungerade dann z=z*x y=y div 2 und x=x*x
Ergebnis z

Summenfunktion von a bis b

i=a; s=i; Solange i < b ist, wiederhole s=s+i und i=i+1
Ergebnis Summe s

Pascal-Programm: ggT-Verfahren von Stein

```
program stein;
var a,b,d,r,x:longint;     fr:char;
begin
   {Eingabe der zwei Zahlen}     write('1. Zahl: ');readln(a);     write('2. Zahl: ');readln(b);
   {in a muss größere Zahl stehen}
   if a<b then begin       d:=a;a:=b;b:=d;    end;
   x:=1;
   repeat
      {wenn beide Zahlen gerade, dann alle gemeinsamen Faktoren von 2 abtrennen und in x speichern}
     if (a mod 2=0) and (b mod 2=0) then
       begin x:=2*x;a:=a div 2;b:=b div 2; end;
       {nicht gemeinsame Faktoren 2 von a und b abtrennen}
     if (a mod 2=0) and (b mod 2=1) then a:=a div 2;
     if (a mod 2=1) and (b mod 2=0) then b:=b div 2;
       {ggt wäre a}
     if a=b then begin b:=1;x:=x*a;end;
       {entsprechend Stein-Regel Zahlen verringern}
     if a>b then a:=a-b else b:=b-a;
   {Abbruch wenn a oder b gleich 1}
   until (a=1) or (b=1);
   {Ausgabe des ggT}
   writeln('ggT: ',x); writeln;
end.
```

Numerisches Dividieren

Die Bestimmung des Quotienten x = a / b ist näherungsweise ausschließlich mit den Operationen Addition, Subtraktion und Multiplikation sowie der Multiplikation bzw. Division mit einer 2er-Potenz realisierbar. Dabei ist praktisch die lineare Gleichung x b = a zu lösen. Mit der Umwandlung

$x\,b + x = a + x$ $\quad\Rightarrow\quad$ $x = a + x - x\,b$ $\quad\Rightarrow\quad$ $x = (1 - b)\,x + a$

erhält man eine Iterationsgleichung, die für $0 < 1 - b < 0{,}5$ Konvergenz besitzt.

Schritt 1

Durch Multplikation bzw. Division sowohl von a als auch von b wird der Divisor b auf einen Wert zwischen 0,5 und 1 reduziert.

Schritt 2

Beginnend mit der Näherung $x_1 = a$ werden Näherungswerte mit $x_{n+1} = (1 - b)\,x_n + a$ berechnet, bis die gewünschte Genauigkeit erzielt wird.

Binäre Exponentiation, Chandah-sûtra-Exponentiation

Die binäre Exponentiation ist eine effiziente Methode zur Berechnung von natürlichen Potenzen, also Ausdrücken der Form x^k mit einer natürlichen Zahl k.
Dieser Algorithmus wurde bereits um 200 v.u.Z. in Indien entdeckt und ist dem Werk "Chandah-sûtra" niedergeschrieben.
Um um Beispiel x^4 zu berechnen, kann man entweder $x \cdot x \cdot x \cdot x$ rechnen (drei Multiplikationen) oder $y = x \cdot x$, $z = y \cdot y$ (zwei Multiplikationen), also $z = (x^2)^2$.
Ebenso können auch andere ganzzahlige Potenzen durch "fortgesetztes Quadrieren und gelegentliches Multiplizieren" effizient berechnet werden. Dieses Einsparen von Multiplikationen funktioniert sowohl für reelle Zahlen als auch für reellwertige Matrizen und beliebige andere Halbgruppen.

Algorithmus

1) Umwandlung von k in die zugehörige Binärdarstellung.
2) Ersetzen jeder 0 durch Q und jeder 1 durch QM.
3) Streichen des führenden QM-Paares.

Nun wird Q als Anweisung zum Quadrieren und M als Anweisung zum Multiplizieren aufgefasst. Somit bildet die resultierende Zeichenkette von links nach rechts gelesen eine Vorschrift zur Berechnung von x^k.

Beispiel: Es sei k = 23. Die Binärdarstellung von 23 lautet 10111. Daraus ergibt sich nach den Ersetzungen QM Q QM QM QM.

Nach dem Streichen des führenden QM-Paares hat man Q QM QM QM. Das bedeutet für den Rechenvorgang:
"quadriere, quadriere, multipliziere mit x, quadriere, multipliziere mit x, quadriere, multipliziere mit x".
d.h. $(((x^2)^2 \cdot x)^2 \cdot x)^2 \cdot x$
Bei diesem Beispiel werden statt 22 Multiplikationen nur noch 7 benötigt, indem man viermal quadriert und dreimal mit x multipliziert.

Anmerkung: Obwohl der Algorithmus schon über 2000 Jahre alt ist und nach der indischen Quelle bezeichnet werden müsste, wird in der letzten Zeit das Verfahren "Square-and-Multiply-Algorithmus" genannt.

Bitalgorithmus

Bitalgorithmen sind eine iterative Möglichkeit zur Berechnung von elementaren Funktionen f(x), wie der Logarithmus- und Exponentialfunktion. Multiplikation und Division werden ausschließlich mit Potenzen von 2 durchgeführt. Im Prozessor sind diese Operationen als bitweise Verschiebung implementiert.
Gegeben sei die Iterationsvorschrift $x_{n+1} = x_n \cdot (1 + d_n\, 2^{-n})$
mit $x_0 = 1$ und $d_n \in \{0, 1\}$.
Sind alle $d_n = 0$ so sind alle $x_n = 1$. Sind alle $d_n = 1$ gilt $x_\infty = 4{,}768$, d.h. mit der Iterationsvorschrift kann bei geeigneter Wahl der Parameter d_n jede reelle Zahl x im Intervall $1 \le x \le 4{,}768$ dargestellt werden.
Weiterhin gelte die Iterationsvorschrift $y_{n+1} = y_n + d_n \cdot \ln(1 + 2^{-n})$
mit $y_0 = 0$. Für numerische Berechnungen werden die $\ln(1 + 2^{-n})$ durch eine voraus berechnete Tabelle realisiert.
Per Induktion folgt dann sofort, dass $y_n = \ln(x_n)$ für alle n gilt. Analog ergibt sich für den Logarithmus der Bereich $0 \le y = \ln(x) \le 1{,}562$.
Um die Logarithmusfunktion zu berechnen wird in jedem Schritt getestet, ob $x_n\,(1 + 2^{-n}) \le x$ ist. Wenn ja, werden x_{n+1} und y_{n+1} berechnet. Nach n Schritten ist der Funktionswert mit einem Fehler $\Delta\ln(x) < 2^{-n}$ bestimmt.

C-Programm:
```
double Logarithmusfunktion(double t)
{ double x, y,s; int k;
  x=1.0; y=0.0; k=0; s=1.0;
  for(k=0;k<N;k++)
  {  if(x+x*s<=t)
     {  y=y+A[k]; x=x+x*s; }
     s=s*0.5; }
  return y; }
```

Beispiele zu Algorithmen (in Modula 2)

Rekursive Definition der Addition zweier natürlicher Zahlen x,y ; Bedingung $y \ge 0$
```
function p_add(x,y: integer): integer;
var z: integer;
begin
   if y = 0 then return x; else z := p_add(x,dec(y)); return inc(z); end;
end;
```

Rekursive Definition der Multiplikation zweier natürlicher Zahlen x,y ; ; Bedingung $y \ge 0$ / Sehr langsam!
```
function p_mult(x,y: integer): integer;
var  z: integer;
begin
   if y = 0 then return 0; else z := p_mult(x,dec(y)); return p_add(z,x); end;
end;
```

Rekursive Definition der Potenz x^y zweier natürlicher Zahlen x,y ; Bedingung $y \ge 0$
```
function p_pow(x,y: integer): integer;
var z: integer;
begin
   if y = 0 then return 1; else z := p_pow(x,dec(y)); return z · x; end;
end;
```

Zurückführung der Addition von nicht-negativen ganzen Zahlen auf Bit-Operationen
```
function add(x,y: integer): integer;
var x0: integer;
begin
   while y > 0 do x0 := x; x := bit_xor(x,y); y := bit_shift(bit_and(x0,y),1); end;
   return x;
end.
```

Zurückführung der Multiplikation ganzer Zahlen auf Additionen und Bit-Operationen.
Die Faktoren werden als nicht-negativ vorausgesetzt.

```
function mult(x,y: integer): integer;
var z: integer;
begin z := 0;
   while y > 0 do  if odd(y) then z := z + x; end;
      x := bit_shift(x,1); y := bit_shift(y,-1);
   end; return z;
end.
```

Zurückführung der ganzzahligen Division auf Additionen, Subtraktionen und Bit-Operationen.
Die Argumente werden als nicht-negativ vorausgesetzt.

```
function divide(x,y: integer): array[2] of integer;
var quot, b: integer;
begin
   quot := 0; b := 1;
   while y < x do  y := bit_shift(y,1);  b := bit_shift(b,1); end;
   while b > 0 do
      if x >= y then x := x - y; quot := quot + b; end;
      y := bit_shift(y,-1); b := bit_shift(b,-1); end;
   return (quot, x);
end.
```

Langarithmetik

Ziel: Arithmetik für beliebig lange ganze Zahlen

1.Definition

```
pzahl=^zahl;  zahl = array[1..g] of longint; i:integer;
```

je Feldelement werden 1 bis 3 Ziffern der ganzen Zahl gespeichert, (3 empfohlen)

2.Übertragkorrektur nach oben

enthält ein Feldelement einen Wert größer als 999 muss ein Übertrag erfolgen

```
procedure maxkorrektur(x:pzahl);
begin     for i:=1 to g-1 do begin
   if x^[i]>=1000 then begin inc( x^[i+1], x^[i] div 1000 ); x^[i]:= x^[i] mod 1000; end;   end; end;
```

3.Übertragkorrektur nach unten

enthält ein Feldelement einen Wert kleiner als 0 muss ein Übertrag erfolgen

```
procedure minkorrektur(x:pzahl);
begin    for i:=1 to g-1 do begin if x^[i]<0 then begin dec( x^[i+1] ); inc( x^[i],1000 ); end; end;
end;
```

4.Vergleich zweier Zahlen

Ist die Langzahl y größer als x liefert die Funktion false anderfalls true

```
function vergleich(x,y:pzahl):boolean;
begin i:=g; while (x^[i]=y^[i]) and (i>1) do dec(i);
  if (i>0) and (y^[i]>x^[i]) then vergleich:=false else vergleich:=true;  end;
```

5.Addition

Die Summe der langen Zahlen a und b wird in c abgelegt, evtl. ist ein Übertrag erforderlich

```
procedure add(a,b,c:pzahl);
begin     for i:=1 to g do c^[i]:= a^[i]+ b^[i];  end;
```

6.Subtraktion

Die Differenz der langen Zahlen a und b wird in c abgelegt, evtl. ist ein Übertrag erforderlich

```
procedure add(a,b,c:pzahl);
begin      for i:=1 to g do c^[i]:= a^[i] - b^[i];  end;
```

Algorithmus arbeitet nur für a>b korrekt !

7.Multiplikation

Produkt von a und b in c, Übertrag beachten

```
procedure mul(a,b,c:pzahl);
var j,i:byte;
begin    fillchar(c^,sizeof(c^),#0);
  for i:=1 to g-1 do for j:=1 to g-1 do if i+j<=g then c^[i+j-1]:= c^[i+j-1] + a^[i]*b^[j];  end;
```

Sortieren durch direktes Einfügen, Insert-Sort

Grundidee: Aufteilen der Elemente in eine Ziel- und eine Quellfolge
Beginnend bei i=2 wird das Element a[i] aus der Quellfolge herausgenommen, und an die entsprechende Stelle der Zielfolge direkt eingefügt; dann wird i um 1 erhöht; Sortiereigenschaft in a[i].key

Grobalgorithmus

· Element x = a[i].key nach links in Folge wandern lassen, mit jedem Element vergleichen, beginnend bei a[i-1].key

· je nachdem wie der Vergleich ausfällt wird x eingefügt oder x wandert weiter
· Abbruch, falls ein Element a[j] gefunden wurde, dessen Schlüssel > Schlüssel von x ist oder wenn man das linke Ende der Zielfolge erreicht hat.
Analyse der Vergleiche V[min] = n-1, V[max] = ((n+1)*n)/2-1
V[mittel] = (n²+3n-4) / 4
Anzahl der Umspeicherung U[min] = 3(n-1), U[max] = (n²+3n-4) / 2

```
Procedure DirektesEinfügen;
var i : index; x:item;
begin   for i:= 2 to n do begin
        x:= a[i]; J:= i-1;
        while (x.key < a[j] and (j=> 1) do
        begin a[j+1] := a[j]; j:= j-1; end;
        a[j+1] :=x;
        end;
        end;
Procedure DirektesEinfügen2;
var i,j : Index; x  : item;
begin   for i:= 2 to n do begin
        x:= a[i]; a[0]:=x; j:=i-1;
        while x.key < a[j].key do begin
        a[j+1]:= a[j]; j:=j-1;
        end;
        a[j+1]:= x; end;
end;
```

Binäres Einfügen
a[i] muss an entsprechender Stelle in Zielfolge eingefügt werden → Zielfolge ist aber schon sortiert!
→ es wird der Schlüssel des Elements in der Mitte der Zielfolge mit den Schlüsseln von a[i] verglichen
→ je nach Ausgang des Vergleiches wird die linke oder rechte Hälfte der Zielfolge durch weitere Halbierung weiter untersucht bis die Einfügestelle gefunden wurde; binäre Sortierung

Algorithmus
· l und r sind die Intervallgrenzen des Intervalls, das halbiert wird; zu Beginn : l:=1, r:=i-1
· Index des mittleren Elements ist m := (l+r) div 2
· falls Element in linker Hälfte l:=1 und r:= m-1sonst l:=m+1 und r:=i-1
· Halbierung solange l < r

Analyse
Einschubstelle gefunden, wenn a[j].key < x.key < a[j+1].key, d.h. das Suchintervall ist 1. Um ein Suchintervall der Länge 1 bei i-1 Elementen zu erreichen, muss log i-mal halbiert werden.
V = log(1) + log(2) + … + log(n) [ganze Zahlen] = O (n · log n)
· Verbesserung in Anzahl der Vergleiche, aber keine Verbesserung beim Umspeichern

Binäres Einfügen - Programm
· durch die Assymetrie der Intervallbildung (durch DIV-Operation) werden Einfügestellen in der linken Hälfte im Mittel etwas schneller gefunden

```
procedure BinaeresEinfuegen;
var i,j,l,r,m : index ; x :item;
begin   for I:=2 to n do begin x:= a[i]; l:=1; r:=i-1;
        while l<=r do begin m:=(l+r) div 2;
        if x.key < a[m].key then r:=m-1 else l:=m+1;
        end;
        for J:=i-1 downto 1 do a[j+1] := a[j];
        a[j] :=x; end;
end;
```

Sortierung durch direkten Austausch, Bubble-Sort
Grundprinzip: Feld wird (n-1) mal durchsucht
· Bei jeder Suche werden benachbarte Elemente verglichen, ist linkes Element größer als rechtes Element, so werden sie vertauscht.
· der Vergleich beginnt an der rechten Seite der Folge, somit wandert bei einer Durchsuchung das jeweils kleinste Element nach links.
Bubble-Sort … wie eine Blase wandert der größte Wert nach oben

```
program BubbleSort1;
var i,j : index; x:item;
```

```
begin for i:=2 to n do for j:= n downto i do
        if a[j-1] > a[j] then tausch (a[j-1],a[j]);
end.
```

Analyse der Grundversion

Vergleiche sind unabhängig von Vorsortierung V = n(n-1)/2
Umtauschen von Elementen U[min] = 0
U[max] = 3·(n²-n)/2
U[mittel] = 3/4 ·(n²-n)

Analyse der verbesserten Methoden (nächste Seite)

Spezialfälle:
V[min] = n-1 bei BubbleSort 2 und 3 bei sortierten Folgen V[mit] = ((n²-n)·k +ln(n)) /2

Verbesserungen verändern nicht die Anzahl der Tauschoperationen, sondern nur die Anzahl der Vergleiche, aber gerade die Tauschoperationen ist die komplexere der beiden Operationen Verbesserungen zeigen nicht allzu große Wirkungen.
siehe

1.Verbesserung

Eine geordnete Folge, die schon sortiert vorliegt muss nicht noch einmal n-1 Mal durchlaufen werden, sondern kann abgebrochen werden. Das ist der Fall, wenn diese einmal durchläuft und nichts vertauscht wurde.

```
Program Bubblesort2;
var i,j : index; x:item; getauscht : boolean;
begin i:=2; getauscht:= true;
        while (i <= n) and getauscht do begin
        getauscht:= false;
        for j:= n downto i do
        if a[j-1] > a[j] then begin
        tausch (a[j-1],a[j]);
        getauscht := true;
end end end.
```

2.Verbesserung

Es wird bei jedem Suchlauf der Index des letzten Tausches (dieser Index ist ja bereits sortiert) gemerkt, so dass der nächste Durchlauf nur bis zu diesem Index durchgeführt werden braucht

```
Program BubbleSort3;
var i,j,k : index; x:item; getauscht : boolean;
begin i:=2 ; getauscht:=true; k:=1;
        while (i <= n) and getauscht do begin
        getauscht:= false;
        for j:= n downto k do
        if a[j-1] > a[j] then begin
        tausch (a[j-1],a[j]);
        k:=j+1;
        getauscht := true;
end end end.
```

Sortierung durch paarweisen Austausch, Ripple-Sort

Grundprinzip
Feld wird in zwei Schleifen durchsucht
· Bei jeder Suche werden zwei Elemente verglichen, ist das zweite Element (Index j) größer als das erste Element (Index i), so werden sie vertauscht.
Ripple-Sort ... eine Welle durchläuft das zu sortierende Feld

```
program RippleSort;
var i,j : index; x:item;
begin
  for i:=1 to n-1 do
    for j:= i+1 to n do
        if a[j] > a[i] then tausch (a[j],a[i]);
end.
```

Ripple-Sort ist das am häufigsten angewandte Verfahren, da es am offensichtlichsten ist. Sein Laufzeitverhalten ist Bubble-Sort ähnlich, aber in den meisten Fällen etwas ungünstiger.

Shaker-Sort

Verbesserung des Bubble-Sort-Verfahrens: die Richtung aufeinanderfolgender Durchläufe wird bei jedem Durchlauf geändert.

Vergleich mit Insert- und Select-Sort:

· alle BubbleSort - Algorithmen sind schlechter
· Shaker Sort dann besser, wenn Folge bereits weitgehend sortiert ist

Sortieren mit abnehmender Schrittweite, Shell-Sort

… basiert auf der Sortierung durch direktes Einfügen
· es werden alle Elemente, die 4 Positionen voneinander entfernt sind, getrennt zusammengefasst und sortiert
· das Gleiche mit allen Elementen die 2 Positionen voneinander entfernt sind
· das Gleiche mit allen Elementen die 1 Position voneinander entfernt sind, d.h. mit allen Elementen
· jede Teilsortierung wird mit dem Sortieren durch direktes Einfügen durchgeführt
· der letzte Schritt ist identisch mit dem Verfahren des direkten Einfügens
· es werden aus direkten Einfügealgorithmen weitere Sortierprozesse vorgeschalten, so dass i.A. vor dem letzten Schritt einer weitgehend geordneten Folge vorliegt
· da direkte Einfügealgorithmen normale Algorithmen sind, werden hier nur noch wenige Umspeicherungen notwendig sein
· die ersten Sortierungen brauchen im Prinzip keine 2^k-Sortierungen zu sein

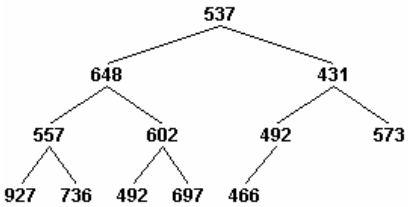
Shell - Sort - Beispiel

· jede Sortierung mit beliebiger Schrittweite sind möglich, die t Schrittweiten werden h[1],h[2],…,h[t] wobei h[t]=1 und h[i+1]<h[i] mit i=1,…,t-1

```
        56    67    23    45    89    34    12    78
K=4     56 <----------------------> 89
        67 <----------------------> 34
        23 <----------------------> 12
        45 <----------------------> 78
        56    34    12    45    89    67    23    78
Sortiere Zahl 1 mit 5, 2 mit 6…
K=2     56 <-> 12  <-> 89  <-> 23
        34 <-> 45  <-> 67  <-> 78
K=1     12    23    34    45    56    67    78    89
```

Analyse

· kompliziert, alle mathematischen Problem sind noch nicht gelöst (1985)
· unklar ist (1985), welche Schrittweiten beste Resultate liefern, denn Vielfaches ist keine beste Lösung!
· Zeitkomplexität $O(n^{1.2})$



Sortieren durch Auswahl mit Auswahlbaum (Williams 1962/64), Heap-Sort

Grundidee

mit n-1 Vergleiche im 1.Schritt darf nicht nur das kleinste Element ermittelt werden, sondern es muss auch mehr Informationen enthalten

Aufbau eines Auswahlbaumes

- mit n/2 Vergleichen wird der kleinste Schlüssel von Paaren von Elementen ermittelt
- mit weiteren n/4 Vergleichen wird der kleinste Schlüssel von Paaren des letzten Ergebnisses ermittelt
--> mit diesen n/2+n/4+n/8+… = n-1 Vergleichen wird ein Auswahlbaum konstruiert, der an der Wurzel der kleinste Schlüssel ist
--> Hinabsteigen entlang des Baumes durch das kleinste Element längs eines markierten Weges, dabei wird das kleinste Element durch Elemente des anderen Zweiges oder durch leere Elemente ersetzt
- nach jedem Herabsteigen --> an Wurzel nächstkleinstes Element
- nach n Schritten Baum leer --> Sortieren beendet

Nachteile: Aufbau der Baumstruktur erfordert mehr Verwaltung, mehr Speicher
Ziel: Baum ohne Löcher -> nur n Speicherplätze

Heap

Definition: Ein Heap ist eine Folge von Schlüsseln h[l],h[l+1],…,h[r] für die gilt:

h[i] <= h[2i] und h[i] <= h[2i+1], für alle i = l,…,r/2

Beispiel siehe

Beispiel zu Heap-Sort

Es soll ein Sortierbaum des Heap aufgebaut werden, der jedes Element nur einmal enthält. Es liege ein Heap mit den Elementen h[l+1], …, h[r] vor. Es soll um ein Element h[l] erweitert werden:

Vorgehensweise

1. h[l] zunächst auf Spitze des Heaps setzen
2. h[l] entlang des Weges der kleineren Elemente, nach dem Wandern nach unten sinken lassen

Es liege ein Heap h[l+1],…,h[r] vor; h[l] soll eingefügt werden; h[i] und h[j] seien die evt. auszutauschenden Elemente

```
VAR i,j:index; x:item;
BEGIN
        i:=l; j:=2*i; x:=a[i];
        WHILE j <= r DO BEGIN
        IF j < r THEN
        IF a[j] < a[j+1] THEN j:=j+1;
        IF x < a[j] THEN BEGIN
        a[i]:=a[j]; i:=j; j:=2*i
        END
        ELSE j:=r+1;
        END;
        a[i]:=x
END;
```

Analyse Heap-Sort

- für kleines n ungünstig
- für große n gut --> besser als Shell-Sort
- Heap-Sort braucht maximal O(n*log(n)) Umspeicherungen

Sortieren durch direktes Auswählen, Select-Sort

Grundidee: es wird das Element mit dem kleinsten Schlüssel gesucht.

· dieses Element wird mit a[1] vertauscht.

· dieser Prozess wird mit den Elementen a[2],…,a[n] wiederholt, dann mit a[3],…,a[n] usw. bis größtes Element a[n] übrigbleibt

Beispiel

	56	67	23	45	89	34	12	78
i=1	12	67	23	45	89	34	56	78
i=2	12	23	67	45	89	34	56	78
i=3	12	23	34	45	89	67	56	78
i=4	12	23	34	45	89	67	56	78
i=5	12	23	34	45	56	67	89	78
i=6	12	23	34	45	56	67	89	78
i=7	12	23	34	45	56	67	78	89

procedure DirekteAuswahl;

```
var i,j,k : index; x:item;
begin   for i:=1 to n-1 do begin k:=i ; x:=a[i];
        for j:=i+1 to n do if a[j].key < x.key then begin k:=j; x:=a[k] end;
        tausch(a[k],a[i]); {a[k]:=a[i]; a[i]:=x;}
        end;
end;
```

Analyse

Select-Sort ist unabhängig von einer Vorsortierung.

V = O (n^2)

U[min] = (n-1),

U[max] = (n^2+5n-6) / 2

U[mittel] = n(ln n +g) = O (n ln n) mit g … Eulersche Konstante g= 0,577216…

Quick-Sort

Grundidee: · Austausch von Elementen über größere Entfernung (Beispiel: umgekehrt geordnete Zahlenfolge kann in n/2 Schritten sortiert werden, wenn 1. mit letztem, 2. mit vorletztem Element getauscht wird)

· es wird ein Element aus der Zahlenfolge ausgewählt, dies sei x

· das Feld wird von links durchsucht, bis ein Element a[i] gefunden wird, das größer ist, als der Schlüssel selbst: a[i].key > x.key, diese beiden Elemente werden ausgetauscht

· das Feld wird von rechts durchsucht, bis ein Element a[i] gefunden wird, das kleiner als der Schlüssel selbst ist: a[i].key < x.key
· dieser Prozess wird solange durchgeführt, bis man sich irgendwo trifft
· das Ergebnis: das Feld ist zerlegt in linken Teil (alle Schlüssel < x.key) und einen rechten Teil (alle Schlüssel > x.key)

Zerlegung in 2 Teile für die gilt:
a[k].key < x.key, für alle k = 1…i-1 a[k].key > x.key, für alle k = j+1…n
a[k].key = x.key, für alle k =i..j

```
PROCEDURE Zerlege(l,r:index);
VAR i,j:Index; x,w:item;
BEGIN  i:=l; j:=r; x:=a[(l+r) DIV 2];
        REPEAT
        WHILE a[i].key < x.key DO i:=i+1;
        WHILE x.key < a[j].key DO j:=j-1;
        IF i<=j THEN BEGIN Tausch(a[i],a[j]); i:=i+1; j:=j-1; END
        UNTIL i>j;
        IF l<j THEN Sort(l,j);
        IF i<r THEN Sort(i,r)
        END;
BEGIN  sort(1,n) END;
```

· auf beide Teile wird Procedure ZERLEGE angewendet
· der Prozess geschieht solange, bis jeder Teil nur noch aus einem Element besteht

Hashfunktion, Streuwertfunktion

Eine Hashfunktion (engl. hash = zerhacken) oder Streuwertfunktion ist eine Abbildung, die zu einer Eingabe aus einer großen Datenmenge eine Ausgabe aus einer kleineren Zielmenge erzeugt, den sogenannten Hashcode oder Hashwert. In der Informatik verwendet man den Begriff Hash-Algorithmus. In der Kryptologie werden Hashcodes verwendet, um den Inhalt eines Dokumentes zu identifizieren. In der Datenspeicherung helfen Hashfunktionen schnell die Speicherstelle von Daten zu finden. Bei Prüfsummen verwendet man Hashwerte, um Übertragungsfehler zu erkennen.
Die Hashwerte sind meist natürliche Zahlen. Eine gute Hashfunktion liefert Werte, die bei unterschiedlichen Eingaben auch unterschiedliche Ausgabewerte liefern.
Eine der effizientesten Hashfunktionen DJBHash wurde von Daniel J. Bernstein entwickelt. Bei Eingabe eines Strings ergibt sich eine natürliche Zahl, der Hash-Code.

```
function DJBHash(const Str : String) : Cardinal; // Bernstein-Algorithmus
var i : Cardinal;
begin Result := 5381;
  for i := 1 to Length(Str) do
 begin
    Result := ((Result shl 5) + Result) + Ord(Str[i]);
  end;
end;
```

Von Justin Sobel stammt eine weitere Hashfunktion:

```
function JSHash(const Str : String) : Cardinal;
var i : Integer;
begin Result := 1315423911;
   for i := 1 to Length(Str) do begin
      Result := Result xor ((Result shl 5) + Ord(Str[i]) + (Result shr 2));
end; end;
```

Hash-Algorithmus

1) Knuths Algorithmus ("The Art of Computer Programming")

```
knuth_hash(unsigned char *str, int len) { unsigned long hash;
     for (hash=len; len--;) { hash = ((hash<<5)^(hash>>27))^*str++; } return hash; }
```

2) Kernigham und Ritchie Algorithmus

```
k_r(unsigned char *str) { unsigned long hash = 0; int c; while (c = *str++) hash += c; return hash; }
```

3) SDBM Algorithmus

```
sdbm(unsigned char *str) { unsigned long hash = 0; int c;
     while (c = *str++) hash = c + (hash << 6) + (hash << 16) - hash; return hash; }
```

4) P.J.Weinberg Algorithmus

```
hashpjw(unsigned char* str) { unsigned long hash=0; unsigned long g;
```

```
    for(;*str;++str) { hash = (hash << 4)+ *str; if(g=hash&0xf0000000) { hash ^= g>>24; hash ^= g; } }
    return hash; }
```

5) OCamls Algorithmus

```
ocaml_hash(unsigned char *str, int len) { unsigned long hash = 0; int i;
    for (i=0; i<len; i++) { hash = hash*19 + str[i]; } return hash; }
```

6) SML Algorithmus

```
sml_hash(unsigned char *str, int len) { unsigned long hash = 0; int i;
    for (i=0; i<len; i++) { hash = 33*hash + 720 + str[i]; } return hash; }
```

7) STL Algorithmus

```
stl_hash(unsigned char *str, int len) { unsigned long hash = 0; int i;
    for (i=0; i<len; i++) { hash = 5*hash + str[i]; } return hash; }
```

8) Javas Algorithmus

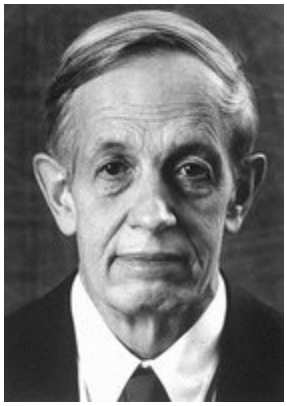
```
java_hash(unsigned char *str, int len) { unsigned long hash = 0; int i;
    for (i=0; i<len; i++) { hash += ((unsigned long)s[i]*31)^(n-1-i); } return hash;}
```

Spieltheorie

Die Spieltheorie ist ein Teilgebiet der Mathematik, um Systeme mit mehreren Akteuren; Spieler, Agenten; zu analysieren, deren Interaktionen denen in Gesellschaftsspielen ähneln.
Die Spieltheorie versucht dabei, das rationale Entscheidungsverhalten in sozialen Konfliktsituationen abzuleiten.

Im Unterschied zur Entscheidungstheorie beschreibt die Spieltheorie Entscheidungssituationen, in denen der Erfolg des Einzelnen nicht nur vom eigenen Handeln, sondern auch von den Aktionen anderer abhängt.
Der Begriff Spieltheorie beruht darauf, dass am Anfang der mathematischen Spieltheorie den Gesellschaftsspielen wie Schach, Mühle, Dame etc. große Aufmerksamkeit gewidmet wurde. Weder ist der Gegenstand der Spieltheorie auf Spiele im gängigen Wortgebrauch beschränkt, noch kann man mit ihrer Hilfe alles, was als Spiel bezeichnet wird analysieren.
Die Spieltheorie ist weniger eine zusammenhängende Theorie als mehr ein Satz von Analyseinstrumenten. Anwendungen findet die Spieltheorie vor allem im Operations Research, in den Wirtschaftswissenschaften, in der Politikwissenschaft, in der Soziologie, in der Psychologie, in der Informatik und seit den 1980ern auch in der Biologie.

Für spieltheoretische Arbeiten wurden bisher acht Wirtschaftsnobelpreise vergeben, welche die große Bedeutung der Spieltheorie für die moderne Wirtschaftstheorie verdeutlichen: 1994 an John Forbes Nash Jr., John Harsanyi und Reinhard Selten, 1996 an William Vickrey und 2005 an Robert Aumann und Thomas Schelling.
Für ihre Erforschung begrenzter Rationalität erhielten Herbert Simon 1978 und Daniel Kahneman 2002 den Nobelpreis. Auch die Nobelpreise an Leonid Hurwicz, Eric S. Maskin und Roger B. Myerson im Jahr 2007 für ihre Forschung auf dem Gebiet der Mechanismus-Design-Theorie stehen in engem Zusammenhang zu spieltheoretischen Fragestellungen.
Quelle: http://de.wikipedia.org/wiki/Spieltheorie



Nash-Gleichgewicht

Das Nash-Gleichgewicht wurde Anfang der 1950er Jahre von John Nash (Abbildung) entwickelt.
Das Nash-Gleichgewicht war ursprünglich ein rein mathematisches Konzept, das sich inzwischen zu einem zentralen Konzept der Sozialwissenschaften entwickelt hat. Nash erhielt 1994 für sein Gleichgewicht den Nobelpreis für Wirtschaftswissenschaften.
Das Nash-Gleichgewicht beschreibt eine Situation in der Spieltheorie, in der sich in einem Spiel mit vollständiger Information keiner der Spieler einen Anreiz, als Einziger von der Gleichgewichtskombination abzuweichen; die Spieler spielen wechselweise beste Erwiderungen. Das Nash-Gleichgewicht wird auch strategisches Gleichgewicht genannt.
Im Nash-Gleichgewicht spielen alle Spieler eine beste Antwort auf das Verhalten der Gegenspieler.

Braess-Paradoxon

Das Braess-Paradoxon der Spieltheorie veranschaulicht, dass eine zusätzliche Handlungsoption bei rationale Entscheidung zu einer Verschlechterung der Situation für alle führen kann. 1968 veröffentlichte der deutsche Mathematiker Dietrich Braess dieses Paradoxon.

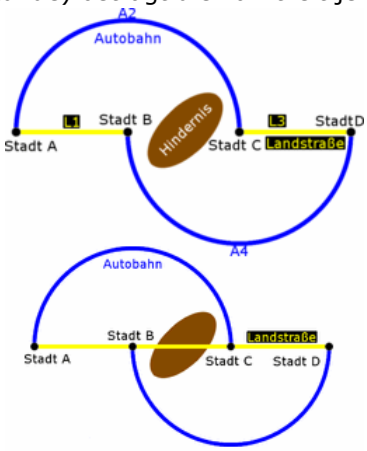
Gegeben sei ein Straßennetz, dass vier Städte A, B, C und D miteinander verbindet. Von A nach C und von B nach D verläuft jeweils eine relativ lange Autobahn, auf der die Fahrtzeit kaum von der Verkehrsdichte abhängt. Bei einer Verkehrsdichte (in Tausend Autos pro Stunde) beträgt die Fahrtzeit je Fahrer

$t_{AC}(x) = t_{BD}(x) = 50 + x$ Minuten

Die Städte A und B sind wie die Städte C und D durch eine Landstraße verbunden, mit einer Fahrtzeit je Fahrer

$t_{AB}(x) = t_{CD}(x) = 0 + 10 x$ Minuten

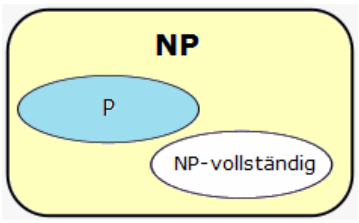
Alle Autofahrer wollen von A nach D fahren, wobei jeder Fahrer den für sich schnellsten Weg wählt. Es stellt sich ein Nash-Gleichgewicht ein, bei dem die Hälfte der Fahrer die Strecke über Stadt B, die andere Hälfte über Stadt C fährt. Bei 6000 Autofahrern fahren somit auf jeder Strecke 3000 Autos. Die Fahrtzeit beträgt für jeden 83 Minuten.



Wird zusätzlich ein Tunnel zwischen den Städten B und C gebaut mit der Fahrzeit

$t_{BC}(x) = 10 + x$ Minuten

so fahren auch Autos direkt von B nach C. Das nun entstehende Gleichgewicht ergibt je 2000 Fahrer wählen die Strecke ABD, die Strecke ACD und die Strecke ABCD.
Durch die vielen Fahrzeuge auf den Landstraßen erhöht sich die Fahrtdauer für alle Fahrer auf 92 Minuten, d.h. länger als ohne die Neubaustrecke.
Ein reales Beispiel für das Braess-Paradoxon ist die umstrittende Waldschössschenbrücke in Dresden. Nach ihrer Eröffnung treten verstärkt Staus und Verzögerungen im Stadtverkehr auf.



Sprachklassen

P-Probleme

Klasse der Probleme, für die ein polynomialer (deterministischer) Algorithmus existiert (Polynomial Time solvable)
Beispiel: binäres Potenzieren, Hornerschema, Quicksort, klassische Matrizenmultiplikation

NP-Probleme

Klasse aller Probleme, deren Lösung durch einen nichtdeterministischen Algorithmus in polynomialer Zeit verifiziert werden können (Nondeterministic Polynomial Time solvable); bisher nur Algorithmen von exponentieller Komplexität bekannt (NP schwer); Beispiel: n-Damen-Problem, Anzahl von Partitionen

NP-completeness(CO-NP) - NP-vollständige Probleme

äquivalente NP-Probleme in dem Sinne: wird nur zu einem dieser Probleme ein polynomialer Algorithmus gefunden, so gehören auch die übrigen Probleme zu P. Diese NP-Probleme heißen NP-vollständig.
Beispiele: SAT-Problem (Satisfiability), TSP-Problem (Travelling Salesman, Rundreise-Problem), SSP-Problem (Subset Sum, Teilsummen), 0/1-Rucksackproblem (Knapsack), Färbungsproblem, Hamiltonkreis, Partitionsproblem, nichtlineare Kongruenz, quadratische diophantische Gleichung, Überdeckungsproblem (Set Covering)

Rucksack-Problem

NP-Problem: Optimierungsaufgabe
... aus einer Gesamtmenge von Gegenständen, welche unterschiedliches Gewicht aber auch unterschiedlichen Wert haben, werden diejenigen ausgewählt und in einen „Rucksack" eingepackt, so dass dieser möglichst gut gefüllt ist aber auch einen größtmöglichen Wert enthält. Beispiel: Bei folgenden Gegenständen

Masse	10	11	12	13	14	15	16	17	18	19
Wert	18	20	17	19	25	21	27	23	25	24

und einem Fassungsvermögen von 120 ergibt sich, dass die Gegenstände mit den Nummern 1, 2, 5, 6, 7, 8, 9 und 10 eingepackt werden sollten. Die Masse dieser Gegenstände beträgt 120, ihr Wert 183.

Teilsummenproblem

es wird versucht aus einer Grundmenge von Zahlen jede Summenmöglichkeit zu ermitteln, welche genau einem vorgegebenen Wert entspricht.
Diese Aufgabenstellung gehört zu den NP-schweren Problemen und benötigt damit auch auf schnellen Rechnern für die Suche aller Lösungen eine gewisse Zeit.
Beispiel: Bei Wahl der Summanden: 7, 14, 18, 23, 22, 26, 28, 31, 35, 39, 42, 55 und 79 und des Maximalwertes 100 ergeben sich 14 verschiedene Teilsummen, beginnend bei 7 + 14 + 18 + 22 + 39

P-NP-Problem

Das P-NP-Problem ist ein ungelöstes Problem der Mathematik und theoretischen Informatik, speziell der Komplexitätstheorie. Es stellt die Frage, ob Probleme existieren, für die eine gegebene Lösung leicht überprüft werden kann, das Finden einer solchen Lösung jedoch extrem schwierig ist.
Dies ist äquivalent zu der Frage, in welcher Beziehung die beiden Komplexitätsklassen P und NP stehen.
Erkannt wurde das Problem zu Beginn der 1970er Jahre aufgrund unabhängig voneinander erfolgter Arbeiten von Stephen Cook und Leonid Levin.
Das P-NP-Problem gilt als eines der wichtigsten offenen Probleme der Informatik und wurde vom Clay Mathematics Institute in die Liste der Millennium-Probleme aufgenommen.

P wird auch als Klasse der praktisch lösbaren Probleme bezeichnet. Probleme aus P sind deterministisch in Polynomialzeit lösbar.
Die Komplexitätsklasse NP ist die Menge aller von nichtdeterministischen Turingmaschinen in Polynomialzeit lösbaren Probleme.

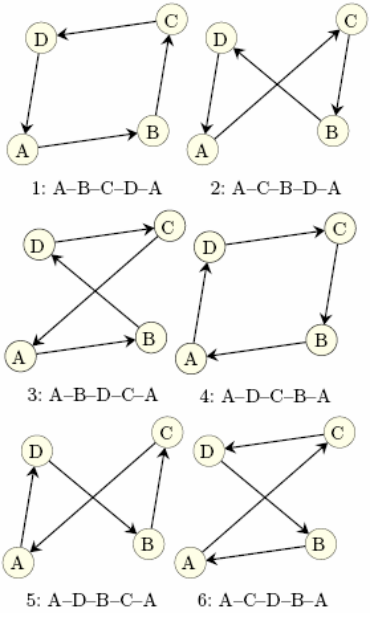
Sehr viele praktische Probleme sind NP-vollständig. Die Lösung des Problems ist daher von großer Bedeutung. Der Beweis von P = NP würde bedeuten, dass für Probleme der bisherigen Klasse NP Algorithmen existieren, die eine Lösung in wesentlich schnellerer Polynomialzeit generieren können.
Mit dem Beweis von $P \neq NP$ wären NP-Probleme endgültig als schwer lösbar klassifiziert. $P \neq NP$ entspricht derzeit der Annahme der meisten Wissenschaftler, und der Beweis wäre weniger folgenschwer als der Beweis von P = NP.

Heiratsproblem

Ein weiteres Problem, welches zu den NP-Problemen (Probleme, die nur von einer nichtdeterministischen Turing-Maschine in polynomialer Zeit gelöst werden können) gezählt werden kann, ist das Heiratsproblem.
Gegeben sind n „heiratswillige" Frauen und Männer. Je Frau und jeder Mann stellt eine Rangfolge der vorhandenen acht Partner auf. Ziel ist es nun, eine für möglichst alle Beteiligten optimale Partnerzuordnung zu berechnen. Wählt man die Eheschließungen so, dass es einen Mann bzw. eine Frau gibt, die nicht miteinander verheiratet sind, aber sich gegenseitig ihren jeweiligen Ehepartnern vorziehen, so nennt man die Heirat unstabil andernfalls stabil. Bei der Berechnung sind damit unstabile Eheschließungen auszuschließen. Die Besonderheit dieses Problems besteht darin, dass Nebenbedingungen auf zwei Ebenen einzuhalten sind.

Rundreiseproblem

Das Problem des Handlungsreisenden (engl. Traveling Salesman Problem, kurz TSP) ist ein kombinatorisches Optimierungsproblem der theoretischen Informatik. Die Aufgabe besteht darin, eine Reihenfolge für den Besuch mehrerer Orte so zu wählen, dass die gesamte Reisestrecke des Handlungsreisenden nach der Rückkehr zum Ausgangsort möglichst kurz ist.
Komplexitätstheoretisch gehört das TSP zur Klasse der NP-äquivalenten Probleme.
Es wird angenommen, dass die schlechteste Laufzeit eines deterministischen Algorithmus, der für dieses Problem stets optimale Lösungen liefert, im besten Fall exponentiell von der Anzahl der Städte abhängt. Schon für wenige Städte kann die benötigte Laufzeit eines solchen Algorithmus viel Zeit beanspruchen. Das Bild zeigt eine optimale Lösung für 15 Städte.



Das Problem des Handlungsreisenden tritt in vielen praktischen Anwendungen auf, beispielsweise in der Logistik oder im Design von Mikrochips.
Im Juni 2001 lösten David Applegate, Robert Bixby, Vasek Chvátal und William Cook das TSP mit 15112 Knoten, eine Reise durch alle Gemeinden Deutschlands. Das gegenwärtig (2007) größte optimal gelöste Rundreiseproblem besitzt 33810 Knoten.

Für das Rundreiseproblem ist 2010 kein effizienter Algorithmus bekannt. Man nimmt an, dass jeder deterministische Algorithmus zur exakten Lösung dieser Probleme mindestens exponentiell viele Rechenschritte ausführen muss.
Das Probieren aller Möglichkeiten ist der einfachste Algorithmus zur exakten Lösung von TSP. Er betrachtet nacheinander alle möglichen Rundreisen, berechnet die Gesamtlänge und ermittelt so die kürzeste Tour. Bei n Städten sind (n-1)!/2 Möglichkeiten zu prüfen.

Berechnungszeiten

Städte	mögliche Rundreisen	Laufzeit	Städte	mögliche Rundreisen	Laufzeit
3	1	1 µs	4	3	3 µs
5	12	6 µs	6	60	60 µs
7	360	360 µs	8	2520	2,5 ms
9	20160	20 ms	10	181440	180 ms
11	1 814400	1,8 s	12	19 958400	20 s
13	239 500800	4 min	14	3113 510400	50 min
15	43589 145600	12 Stunden	16	653837 184000	7,5 Tage

Für diese Tabelle wird angenommen, dass je Sekunde 1 Million Wege geprüft werden.

Müller-Merbach-Verfahren

Das Verfahren nach Müller-Merbach wird zur Berechnung einer optimalen Rundreise durch gegebene Orte verwendet. Dieser Rundreise durch n Orte liegt eine Kostenmatrix C zugrunde. Gesucht ist die kostengünstigste Route durch die n Orte.

Das Verfahren startet mit zwei beliebigen Orten e_i und e_k, durch die eine Rundreise (e_i,e_k,e_i) mit den Kosten $C_{ik}+C_{ki}$ definiert ist. Diese Subtour wird nun aufgebrochen, indem ein zusätzlicher Ort e_j eingeschoben wird. Man erhält dann zwei mögliche Subtouren: (e_i,e_j,e_k,e_i) und (e_i,e_k,e_j,e_i).
Unter diesen wählt man die bessere aus und fährt dann fort, weitere Orte einzufügen. Ist z.B. (e_i,e_j,e_k,e_i) die bessere Variante und wird e_p hinzugenommen, so kann e_p zwischen e_i und e_j, zwischen e_j und e_k oder zwischen e_k und e_i gesetzt werden. Man wählt die beste von den drei Varianten aus, im nächsten Schritt die beste unter vier Möglichkeiten usw.
Dieses Verfahren wird z.B. für die Kostenmatrix C

	e1	e2	e3	e4	e5	e6	e7
e1	0	14	20	10	35	18	5
e2	6	0	7	35	17	9	24
e3	8	35	0	36	27	3	15
e4	21	7	12	0	7	4	26
e5	33	25	6	18	0	19	11
e6	6	2	22	30	9	0	8
e7	24	3	12	5	17	16	0

durchgerechnet, wobei von e_1 und e_2 ausgegangen wird und der Reihe nach die Orte e_i, i=3,… hinzugefügt werden. Die beste Lösung für dieses Verfahren ist im Beispiel die Rundreise $(e_1,e_7,e_4,e_6,e_2,e_5,e_3,e_1)$ mit den Kosten von 47 Einheiten. Der Kostenzuwachs berechnet sich zu $C_{ip} + C_{pj} - C_{ij}$, wenn zwischen die Orte e_i und e_j e_p eingefügt wird.

Kryptologie, Kryptografie

Die Kryptografie ist der Wissenschaftsbereich, welcher sich mit der Verschlüsselung und Entschlüsselung von Daten und den zugehörigen Algorithmen beschäftigt.
Wort aus dem griechischen κρυπτοσ (geheim), λογοσ (das Wort, der Sinn) und γραφειν (schreiben) gebildet

Klassifikation der Sicherheit

Ein Kryptosystem heißt

· absolut (informationstheoretisch) sicher,
wenn nicht genug Information gewonnen werden kann, um den Klartext bzw. den Schlüssel zu rekonstruieren

· analytisch sicher,
wenn es kein Verfahren gibt, mit dem es gebrochen werden kann

· komplexitätstheoretisch sicher,
wenn es keinen Algorithmus gibt, der das System in Polynomialzeit bricht

· praktisch sicher (stark),
wenn kein Verfahren bekannt ist, das das Verfahren mit den verfügbaren Ressourcen brechen kann
Abbildung: Eines der ältesten kryptografischen Werke von 1518, die "Polygraphiae" des Gelehrten Trithemius, zeigt auf der Titelseite, wie der Autor, Benediktinerabt von Sponheim, das Buch in Form eines quadratischen Verschlüsselungsbrettes seinem Kaiser, Maximilian I., überreicht.
Abbildung: Kodierung (London Underground: Jahr der Mathematik 2000 - Oktober)

Symmetrische Kryptosysteme (Chiffresysteme)

Schlüssel und Verfahren zum Ver- und Entschlüsseln der Nachricht sind identisch

monoalphabetische Chiffriersystem
... gleiche Klartextzeichen immer durch gleiche Geheimtextzeichen ersetzen (Cäsar-Chiffre)
polyalphabetische Chiffriersysteme
... verschiedene Geheimtextzeichen für das gleiche Klartextzeichen (Vigenere-Chiffre)
--> weder monoalphabetisch als auch polyalphabetich ist „richtig" sicher, sind über statistische Methoden zu knacken

Substitutionschiffre
Eine Substitutionschiffre ist ein einfaches Verschlüsselungsverfahren, bei dem die Zeichen eines Wortes durch zugeordnete Chiffrezeichen ersetzt werden. Je nach Verfahren kann die Substitution durch ein Zeichen aus einer Chiffretabelle erfolgen (Cäsar-Chiffre), oder es können abwechselnd mehrere Chiffretabellen für die Substitution benutzt werden (Vigenère-Chiffre).

Verschiebechiffre, additive Chiffre
Text verschieben um Schlüssel; besser Verschlüsselung

Tabelle	A	B	C	D	E	…	Kopie
	A	B	C	D	E	F	…
	B	C	D	E	F	G	…
	…						

Weitere Verschlüsselungen: Tauschalgorithmen , Modulaverschlüsselung, Mehralphabet, …

Prinzip von Kerckhoffs
Die Sicherheit eines Kryptosystems beruht nicht auf der Geheimhaltung des Verschlüsselungsalgorithmusses, sondern ausschließlich auf der Geheimhaltung der teilnehmerspezifischen Schlüssels.

One Time Pad
Werden bei der Verschlüsselung die Buchstaben des Klartextes nicht immer durch denselben Buchstaben ersetzt, also z.B. A nicht immer durch C, sondern mal durch Z, mal durch E, mal durch B usw., so ist eine statistische Analyse nicht möglich. Eine solche Verschlüsselung bezeichnet man als polyalphabetische Verschlüsselung.
Ist der Schlüssel eine gleich verteilte Zufallsfolge von Buchstaben, so ist dieses Verschlüsselungsverfahren sogar absolut sicher. Dies liegt daran, dass es genauso viele mögliche Schlüssel wie mögliche Klartexte gibt, jeder Schlüssel gleichwahrscheinlich ist und somit auch jeder aus dem Geheimtext rekonstruierte Klartext gleich wahrscheinlich ist. Niemand kann sagen, ob der Schlüssel Txedubnhw oder Ykrephypi war. Die praktische Bedeutung ist aber leider gering. Eine Verschlüsselung durch Addition einer Zufallsfolge heißt Vernam-Chiffre oder One-Time-Pad.

Stromchiffre
Bei einer Stromchiffre wird die zu verschlüsselnde Nachricht als Datenstrom aufgefasst und zeichenweise mit einem Schlüsselstrom verknüpft.
Der Schlüsselstrom ist meist eine Pseudozufallsfolge, die nur mit Kenntnis des geheimen Schlüssels erzeugt werden kann. Im Gegensatz zu Blockchiffren ist die Verschlüsselung bei Stromchiffren positionsabhängig. Ein beliebiger Abschnitt eines Chiffretextes kann nicht entschlüsselt werden, ohne den vorhergehenden Teil zu entschlüsseln.

Multiplikative Chiffre
Bei dieser multilikativen Chiffierung verwendet man, im Gegensazu zu additiven Chiffren, Multiplikation modulo n; im deutschen Alphabet n = 26.
Jeder Klartextbuchstabe wird mit dem Schlüssel k multipliziert und das Ergebnis modulo 26 umgewandelt.
Zum Beispiel wird für k = 2:
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
Geheimtext: B D F H J L N P R T V X Z B D F H J L N P R T V X Z
Auffallend ist, dass jeweils zwei unterschiedliche Buchstaben dasselbe Produkt ergeben. Damit ist die Chiffre mit k = 2 ungeeignet.
Damit die multiplikative Chiffre bei der Dekodierung eindeutige Resultate liefert, darf es nicht zu Überlappungen und Uneindeutigkeiten in der Chiffrierung kommen. Der Klartext muss mit Hilfe des Schlüssels eindeutig aus dem Geheimtext rekonstruierbar sein.

Damit kommen nur Alphabete in Frage, deren Elemente nur einmal vorkommen. Für 26 Buchstaben können als Faktoren k nur Zahlen verwendet werden, die mit 26 keine gemeinsamen Teiler haben:
1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 und 25
d.h. nur 12 verschiedene Faktoren k.
Zum Beispiel wir mit k = 3:
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
Geheimtext: C F I L O R U X A D G J M P S V Y B E H K N Q T W Z

Eine derartige Chiffrierung ist möglich. Allerdings ist die Schlüsselzahl noch geringer als bei der Verschiebechiffre.

Skytale

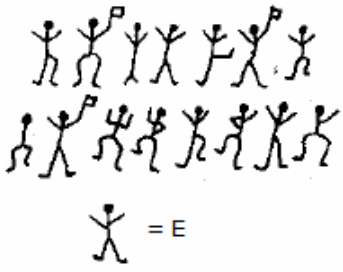
Vor 2500 Jahren verwendete Sparta eine interessante Methode zur Übermittlung geheimer Nachrichten.
Sender und Empfänger mussten beide eine sogenannte Skytale haben. Dies waren zwei Zylinder mit denau dem gleichen Radius.
Der Sender wickelte ein schmales Band aus Pergament spiralförmig um seinen Zylinder und schrieb dann der Länge nach seine Nachricht auf das Band. War nun das Band abgewickelt, konnte die Nachricht nur von einer Person gelesen werden, die einen Zylinder genau desselben Umfangs hatte.

Atbasch-Kodierung

Bei der Atbasch-Kodierung wird einfach der erste Buchstabe des Textes durch den letzten Buchstaben des Alphabets ersetzt, der zweite durch den vorletzten, usw. Natürlich ist dieses Verfahren in keiner Weise sicher.
Historisch ist dieses Verfahren im antiken Judentum nachzuweisen.

Binäres Verschlüsseln

Ein einfaches, schnelles, aber für den praktischen Einsatz nicht sicheres Verschlüsselungssystem ist das binäre Verschlüsseln.
Dazu wird der Originaltext in eine Folge von Binärziffern transformiert und diese Folge mit einer Zufallsbinärfolge bitweise Entweder-Oder verknüpft.
Zu Entschlüsselung wird die gleiche Zufallsfolge wieder Entweder-Oder mit dem chiffrierten Text bearbeitet. Diese Verfahren ist dem Vigenere-Code sehr ähnlich.



Die tanzenden Männchen

Die Möglichkeit, die Kryptoanalyse durch die Verwendung von seltsamen Zeichen als Geheimtextalphabet zu erschweren, hat wenig Sinn, da die Eigenheiten der Sprache dadurch auf die Zeichen übertragen werden.
Dennoch wurde diese Art der monoalphabetische Chiffrierung auch gern von Krimi-Autoren verwendet, denn das Entschlüsseln des Textes ist verhältnismäßig einfach, so dass der Leser noch folgen kann.
Edgar Allan Poe verwendete diese Codierungsvariante in seinem Stück "Der Goldkäfer", Arthur Conan Doyles Sherlock Holmes musste sich in der Kurzgeschichte "Die tanzenden Männchen" (engl. dancing men) damit plagen. Hier ein kleiner Auszug aus dieser Geschichte:

"Holmes hielt das Papier hoch, so dass die Sonne voll darauffiel. Es war eine aus einem Notizbuch herausgerissenen Seite. Die Zeichen waren mit Bleistift gemalt und sahen so aus: (siehe Abbildung) …
Holmes betrachtete sie eine Zeitlang, faltete das Blatt vorsichtig zusammen und steckte es in die Brieftasche. »Das verspricht einen äußerst interessanten und ungewöhnlichen Fall« sagte er. …
Holmes beugte sich einige Minuten lang über den grotesken Fries und sprang dann plötzlich mit einem Ausruf der Überraschung und Bestürzung auf. …
»… Nachdem ich einmal erkannt hatte, dass die Symbole für Buchstaben stehen, und ich die Regeln anwandte, die für alle Arten von Geheimschriften gelten, war die Lösung nicht mehr schwierig.
Die erste Nachricht, die man mir überlies, war so kurz, dass es unmöglich war, mir einiger Sicherheit mehr zu sagen, als dass (untere Abbildung) für E stand.
Wie sie wissen, ist E der im Englischen gebräuchlichste Buchstabe, und er herrscht in einem solchen Maße vor, dass man erwarten kann, ihn selbst in einem kurzem Satz als den häufigsten zu finden. Von den fünfzehn Symbolen der ersten Botschaft kehrte eines viermal wieder, und so war es nur vernünftig, es als E anzunehmen. …«

Goldkäfer-Chiffre

Der Goldkäfer, The Gold-Bug, ist eine Kurzgeschichte von Edgar Allan Poe, in der er im Rahmen einer Schatzsuche ausführlich die Dechiffrierung einer Geheimschrift anhand von Häufigkeitszahlen der einzelnen Buchstaben in englischen Texten erläutert.
Der Geheimtext lautete:

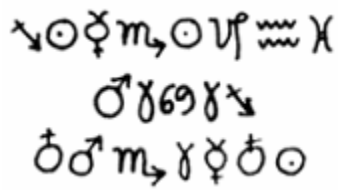
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8
¶60))85;1‡(;:‡*8†83(88)5*†;46(;88*96
?;8)‡(;485);5*†2:*‡(;4956*2(5*—4)8
¶8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
1;48†85;4)485†528806*81(‡9;48;(88;4
(‡?34;48)4‡;161;:188;‡?;

Durch Analyse der Häufigkeit der auftretenden Zeichen und der Buchstabenhäufigkeiten in der englischen Sprache ermittelt Poe als Klartext:

A good glass in the bishop's hostel in the devil's seat
forty-one degrees and thirteen minutes northeast and by north
main branch seventh limb east side shoot from the left eye of the death's-head
a bee line from the tree through the shot fifty feet out.

Interessant ist, dass die verwendete Kryptoanalyse mit einer semantischen Analyse des Textes kombiniert wurde. Im Übrigen finden die Personen der Kurzgeschichte den versteckten Schatz.

Edgar Allan Poe interessierte sich für Kryptoanalyse. Ihm zugesandte monoalphabetische Geheimschriften konnte er stets entschlüsseln. Er veröffentlichte außerdem theoretische Schriften zum Thema Geheimschrift.



Eco-Chiffre

Im berühmten Roman "Der Name der Rose" (1980) von Umberto Eco finden der Franziskaner William von Baskerville und der junge Adson von Melk ein Schriftstück mit den links abgebildeten Zeichen.
William von Baskerville erkennt eine Chiffre mit Tierkreiszeichensymbolen. Außerdem gelingt es ihm, den griechischen Text und seinen kryptischen Inhalt zu entschlüsseln. Wie, verschweigt leider Umberto Eco.

"Als ich fertig war, nahm William meine Tafel, hielt sie hoch und musterte die Kopie, leider ohne seine Augengläser. »Zweifellos eine Geheimschrift, die wir entziffern müssen«, sagte er. »Die Zeichen sind schlecht gemalt, und vielleicht hast du sie in deiner Kopie noch mehr verzerrt, aber es handelt sich fraglos um ein Alphabet aus Tierkreiszeichen. Sieh hier, in der ersten Zeile haben wir« – er hielt die Tafel mit gestreckten Armen weit von sich und kniff die Augen zusammen – »Schütze, Sonne, Merkur, Skorpion …«"
…
"Die besten Traktate über Kryptographie sind Werke ungläubiger Gelehrter, ich hatte in Oxford die Möglichkeit, mir einige davon vorlesen zu lassen. Roger Bacon sagte zu Recht, daß der Erwerb des Wissens mit dem Erlernen der Sprachen beginnt. Abu Bakr Ahmad ben Ali ben Washiyya an-Nabati schrieb vor Jahrhunderten ein Buch der unbezähmbaren Begierde des Frommen, die Rätsel der alten Schriften zu lösen. Darin finden sich viele Regeln zur Bildung und zur Entzifferung von Geheimschriften, die man zu magischen Zwecken benutzen kann, aber auch zur Verschlüsselung der Korrespondenz zwischen zwei Armeen oder zwischen einem König und seinen Botschaftern."

Gartenzaunkodierung

Bei der Gartenzaumkodierung wird die Nachricht, hier "das ist ein test", in n Reihen aufgeteilt, und diese in Form einer Zickzack-Linie (Gartenzaun) untereinander aufgeschrieben.
Die kodierte Nachricht ergibt sich, in dem die Buchstaben der einzelnen Zeilen aneinandergefügt werden.
Im abgebildeten Beispiel ist n = 3 und aus der Nachricht "das ist ein test" wird der Codetext "dsntaititssee"

Sind n Zeilen vorgegebenen, so enthält (mit jeweils i = 0,1,2,…)
die oberste Zeile alle Buchstaben mit den Positionen 1 + (2n-2) i
die 2.Zeile alle Buchstaben mit den Positionen 2 + (2n-2) i und ((2n-2)-0) + (2n-2) i
die 3.Zeile alle Buchstaben mit den Positionen 3 + (2n-2) i und ((2n-2)-1) + (2n-2) i
die 4.Zeile alle Buchstaben mit den Positionen 4 + (2n-2) i und ((2n-2)-2) + (2n-2) i …
die k.Zeile alle Buchstaben mit den Positionen k + (2n-2) i und ((2n-2)-(k-2)) + (2n-2) i …
die letzte Zeile alle Buchstaben mit den Positionen n + (2n-2) i

Buchseitenkode, Buch-Chiffre

Ein einfaches aber sehr wirkungsvolles Verfahren ist die Verschlüsselung von Texten, indem man anstelle der Wörter nur Seitenzahlen, Zeilenzahlen und Wortpositionen verschickt. Die Wörter stammen dabei aus einem bestimmten Buch. Diese einfache Methode wurde lange Zeit auch von Geheimdiensten genutzt.

Zum Beispiel können die Geheimtexte
255 17 1 53 29 2 34 20 8 181 28 5
163 23 10 253 31 2 94 8 7 236 12 7
ohne die genaue Kenntnis des Buches, des Verlages und auch der Auflage nur mit extremen Aufwand decodiert werden.
Jeder Block besteht aus drei Zahlen. Die erste Zahl jedes Blocks gibt die Seite an, auf der das verwendete Wort zu finden ist. Die zweite Zahl steht für die Zeile, die dritte für die Position des Wortes in der Zeile. Jeder Block steht für ein komplettes Wort.
Damit ist diese Verschlüsselung nicht nur relativ sicher, sondern auch kurz.

Der obige Geheimtext wurde mit der 5.Auflage (1982) der Kriminalgeschichten "Wer stiehlt schon Unterschenkel?" von Gert Prokop beim "Verlag Das Neue Berlin" verschlüsselt.

Da es für den Programmnutzer schwierig sein wird, dieses spannende Buch zu erhalten, wird hier der Klartext angegeben. Der erste Klartext lautet "Wir benutzen diese Methode"
der zweite "Anschließend spielen wir Computer"
Obwohl in beiden Nachrichten "wir" auftritt, kann es unterschiedlich verschlüsselt werden, da dieses häufige Wort in dem Buch mehrfach auftritt.
Eine spezielle Form der Buch-Chiffre ist die Ottendorf-Chiffre, bei der Angaben der Form 1:23:45 ein Buch, die Seitenzahl und Wortzahl kodieren.

I	C	H	B	I	N
D	E	R	D	O	K
T	O	R	E	I	S
E	N	B	A	R	T

1	2	3	4
E	S	W	A
R	S	C	H
O	N	D	U
N	K	E	L

1	2	3	4
W	C	D	E
A	H	U	L
S	S	N	K
E	R	O	N

Transpositionsverfahren

Beim Transpositionsverfahren werden die Zeichen des Klartextes nicht durch andere ersetzt, sondern lediglich in der Reihenfolge vertauscht.
Als Grundlage dient eine Matrix in die das zu chiffrierende Wort zeilenweise eingetragen wird. Die Anzahl der Spalten ist der geheime Schlüssel. Daraus und aus der Gesamtlänge des Klartextes ergibt sich die Anzahl der Zeilen in der Matrix. Der Geheimtext wird durch spaltenweises Auslesen der Matrix gebildet.

Beispiel (obere Abbildung) mit Matrix mit Schlüssel 6
spaltenweise ausgelesen: IDTECEONHRRBBDEAIOIRNKST
Das noch im Jahr 1917 von der deutschen Reichwehr eingesetzte Drehraster basiert auf diesem Verfahren.
In dieser einfachen Form wird dieses Verschlüsselungsverfahren auch hebräisches Verfahren genannt.

Einfache Spalten-Transposition

Alternativ dazu kann man die Matrix auch nach einer vorgegebenen Permutation spaltenweise auslesen.
Beispiel (mittlere Abbildung) Schlüssel 4 und Auslesepermutation 3421
permutiert ausgelesen: WCDE AHUL SSNK ERON

Doppelte Spalten-Transposition

Wird der durch einfache Spalten-Transposition gewonnene Chiffre erneut in die Matrix eingeschrieben und nochmals ausgelesen, erhöht man die Sicherheit des Geheimtextes.
Beispiel der mittleren Abbildung weiter mit Schlüssel 4 und Auslesepermutation 3421
erneut permutiert ausgelesen: DUNO ELKN CHSR WASE

	1	2	3	4
4	5	6	7	8
3	13	14	15	16
2	1	2	3	4
1	9	10	11	12

Weitere Möglichkeiten von Verschlüsselungsverfahren mit Transposition sind:

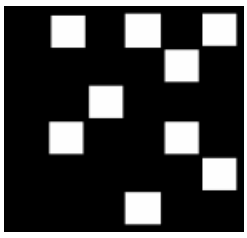
Gemischte Zeilen- und Spalten-Transpositionen

Bei dieser Abwandlung des Verfahrens wird nicht nur das Auslesen, sondern auch das Einschreiben in die Matrix permutiert.
Ein Beispiel mit einem Schlüsel 4, der Einschreibepermutation der Zeilen 2413 und der Auslesepermutation der Spalten 2143 ist in der Abbildung dargestellt.
Als Chiffre wird ausgelesen
6 14 2 10 5 13 1 9 8 16 4 12 7 17 3 11

Block-Transposition

Der Klartext wird hierzu in gleichgroße Blöcke unterteilt, die jeweils mit dem gleichen Transpositionsverfahren chiffriert weren.
Zum Beispiel wird mit dem Schlüssel 4 und der Permuation 3421 aus dem Klartext
ESWA RSCH ONDU NKEL
der chffrierte Text
WASE CHSR DUNO ELKN

Mit der Kenntnis des Schlüssels, des verwendeten Transpositions-Verfahrens und der Länge des Geheimtextes kann die Matrix wiederhergestellt und rückwärts ausgelesen werden. Dabei erhält man den Klartext zurück. Ohne Kenntnis des Schlüssels greift man auf Kryptoanalyse und Häufigkeitsstatistiken



Fleißnersche Schablone

Die Fleißnersche Schablone ist ein Verschlüsselungsverfahren, bei dem eine Transposition mit einer Schablone den Klartext verschlüsselt.
Die Schablone wurde von dem österreichischen Oberst Eduard Fleißner von Wostrowitz 1881 in seiner Schrift "Neue Patronengeheimschrift" veröffentlicht.
Bekannt wurde das Verfahren durch Jules Verne. In seinem Roman "Mathias Sandorf" nutzte er 1885 die Schablonen.
Die Fleißnersche Schablone besteht aus einem Quadrat, aus dem kleinere Quadrate ausgeschnitten sind. Die Schablone wird auf ein Blatt Papier gelegt und jeweils ein Buchstabe des Klartextes in ein ausgeschnittenes Quadrat eingetragen.

Sind alle Felder gefüllt, wird die Schablone um 90° gedreht und die nächsten Buchstaben in die Lücken geschrieben.
Wird der Vorgang noch zweimal wiederholt, entsteht ein Quadrat mit Buchstaben. Ist die Nachricht länger, wird ein neues Quadrat begonnen.

Es können eine Vielzahl verschiedener Fleißnerscher Schablonen konstruiert werden.
Mathematische Bedingungen für die Erzeugung solcher Schablonen sind:
1) Die Anzahl der Felder der Schablone ist durch 4 teilbar, da die Schablone viermal aufgelegt wird
2) Ein Viertel aller Felder wird ausgeschnitten, wobei keine Symmetrie innerhalb der ausgeschnittenen Felder auftreten darf

Die Original-Schablone hatte 36 Felder, von denen 9 ausgeschnitten sind.
Die Schablonen können erzeugt werden, indem man ein Viertel der Matrix mit den Werten 1 bis 4 füllt und diese unter zyklischer Verschiebung der Ziffern dreimal um 90° in den jeweils nächsten Quadranten dreht. Ausgeschnitten werden alle Felder mit der gleichen Ziffer, z.B. der 1. Geht man von gleichmäßiger Verteilung über alle vier Quadranten aus, ergibt sich die Anzahl der so möglichen Schablonen durch:

$$n = 4 \cdot \binom{9}{3} \cdot 3 \cdot \binom{6}{2} \cdot 2 \cdot \binom{4}{2} \cdot \binom{2}{2} = 181440$$

Von den 181440 mögliche Schablonen sind nicht alle gut geeignet, da mitunter Felder nebeneinander liegen und der Text einfacher lesbarer wird.
Für Schablonen in beliebiger Größen mit N x N Feldern gilt bei gleichmäßiger Verteilung:

$$n = \prod_{k=0}^{3} (4-k) \cdot \binom{[N^2 (4-k)/16]}{[[N^2 (4-k)/16] / (4-k)]}$$

Die Anzahl der Felder N² muss größer 4 sein. Unter [x] ist die größte ganze Zahl kleiner gleich x zu verstehen.

Blockweise Verschlüsselung

Bei der blockweisen Verschlüsselung wird der Klartext zunächst in eine Bitfolge umgewandelt. Aus dieser Folge werden jeweils Dreierblöcke entsprechend einem Codierungsschlüssel transformiert und das Ergebnis wieder in Zeichen des Geheimtextes verwandelt.

Beispiel Codierungsschlüssel:
0 → 1 ; 1 → 6 ; 2 → 0 ; 3 → 7 ; 4 → 3 ; 5 → 4 ; 6 → 2 ; 7 → 5

Klartext	e	b	e	n
ASCII-Code	1 0 1	9 8	1 0 1	1 1 0
Bitfolge	01100101	01100010	01100101	01101110
Dreierblöcke	011 001 010	110 001 001	100 101 011 …	
Dezimalzahlen	3 1 2	6 1 1	4 5 3 3 …	
	↓ ↓ ↓	↓ ↓ ↓	↓ ↓ ↓ ↓	
zugeordnete Zahlen	7 6 0	2 6 6	3 4 7 7 …	
Dreierblöcke	111 110 000	010 110 110	011 100 111 …	
Dualzahlen	11111000	00101101	10011100	11111101
Dezimalzahlen	2 4 8	4 5	1 5 6	2 5 3
Geheimtext	°	–	£	2

Die Entschlüsselung verläuft in umgekehrter Reihenfolge, d.h. mit dem entgegengesetzten Codierungsschlüsel.

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	j
3	kq	l	m	n	o
4	p	r	s	t	u
5	v	w	x	y	z

Polybius-Chiffre

Der griechische Historiker Polybius (200-118 v.u.Z.) beschrieb im Buch X seiner "Geschichte" ein einfaches Verschlüsselungsverfahren.
Die Idee besteht in der Ersetzung eines jeden Buchstabens durch zwei Ziffern von 1 bis 5. Die Ziffernkombination wird aus der links abgebildeten Tabelle entnommen. Zum Beispiel wird ein a durch 11, ein r durch 42 kodiert.

Polybius beabsichtigte dabei nicht, eine Nachricht sicher zu verstecken. Ursprüngliches Ziel war es, einen Text so zu verändern, dass dieser über optische Telegraphen mit Hilfe von brennenden Fackeln übermittelt werden konnte. Ein e = 15 konnte mit einer Fackel links und fünf Fackeln rechts gesendet werden.
Zu Beginn bestand die Kodierungstabelle aus den 24 griechischen Buchstaben und einem Zeichen zu Beginn und am Ende einer Nachricht. Später wurde das System auf die lateinischen Buchstaben umgestellt.
Jahrhunderte später wurde die Polybius-Chiffre zur Playfair-Chiffre weiterentwickelt.

Cäsar-Chiffre

Gegeben sei ein Alphabet mit 26 Buchstaben (A...Z). Der Klartext "WINFUNKTION" soll verschlüsselt werden.

Die einfachste Form der Verschlüsselung ist, jeden Buchstaben des Klartextes durch z.B. den drittnächsten Buchstaben im Alphabet zu ersetzen (gemäß der alphabetischen Reihenfolge und zyklisch, d.h. auf Z folgt wieder A). Das Ergebnis ist der Geheimtext ZLQIXQNWLRQ !
Mathematisch entspricht diese Verschlüsselung einer buchstabenweise "Addition" des Textes DDDDDDDDDDD zum Klartext. Werden die Buchstaben entsprechend der alphabetischen Reihenfolge von 0 bis 25 numeriert, so ergibt sich die Summe zweier Buchstaben aus der Summe dieser Nummern modulo 26.

```
.       W  I  N  F  U  N  K  T  I  O  N
+       D  D  D  D  D  D  D  D  D  D  D
.       Z  L  Q  I  X  Q  N  W  L  R  Q
```

Der Empfänger kann aus dem Geheimtext den Klartext wieder zurückgewinnen. Er muss dazu wissen, mit welchem Algorithmus die Verschlüsselung vorgenommen wurde, und er muss den Schlüssel D kennen. Durch Umkehrung des Verschlüsselungsalgorithmus (Subtraktion) unter Verwendung des richtigen Schlüssels ergibt sich wieder der Klartext.

Beim Cäsar-Chiffre handelt es sich um eine monoalphabetische Substitution. Jedem Buchstaben wird ein eindeutiger anderer Buchstabe zugeordnet. Diese Zuordnung basiert auf der zyklischen Rotation des Alphabets um eine feste Anzahl von Zeichen, dabei folgt auf z wieder a. Die Anzahl der verschobenen Zeichen ist dann der Schlüssel, mit dem ver- bzw. entschlüsselt wird.

JavaScript-Texte zum Ver- und Entschlüsseln

```
 1:  var alphabet = "abcdefghijklmnopqrstuvwxyz"; var alphaLen = alphabet.length;
 2:  function cesar_encrypt (key, text) {
 3:    var i, n    = 0;  var chiffre = "";
 4:    key  = key % alphaLen;
 5:    text = text.toLowerCase();
 7:    for (i=0; i < text.length; i++) {   n = alphabet.indexOf(text.charAt(i));
 7:       if (n >= 0) chiffre = chiffre + alphabet.charAt((n+key) % alphaLen);
 8:       else chiffre = chiffre + text.charAt(i); }
 9:    return chiffre; }

10:  function cesar_decrypt (key, chiffre) {
11:    var i, n = 0; var text = "";
12:    key     = key % alphaLen;
13:    chiffre = chiffre.toLowerCase();
14:    for (i=0; i < chiffre.length; i++) {  n = alphabet.indexOf(text.charAt(i));
15:       if (n >= 0)  text = text + alphabet.charAt((alphaLen+(n-key)) % alphaLen);
16:       else  text = text + text.charAt(i); }
17:    return text; }
```

ROT13-Verfahren

ROT13 ist eine Verschiebechiffre auf dem Alphabet {A, …, Z}, bei der jeder Buchstabe durch denjenigen ersetzt wird, der im Alphabet 13 Stellen weiter steht.
Der Name leitet sich aus dem Begriff "rotiere um 13 Zeichen" ab. Da das lateinische Alphabet ohne Sonderzeichen genau 26 Buchstaben hat, kann die Ver- und Entschlüsselung mit demselben Algorithmus erreicht werden, nämlich mit der Rotation durch das Alphabet um 13 Stellen.
ROT13 ist damit nur ein Sonderfall der Cäsar-Chiffre und damit für die Verschlüsselung von Nachrichten ungeeignet.

Dennoch wird das Verfahren vielfach in Newsgroups zur Unkenntlichmachung von Texten eingesetzt. Als Verschlüsselung ist das trivial, es hat lediglich zum Ziel, bestimmte Inhalte, die eventuell als kompromittierend empfunden werden, nicht direkt lesbar zu machen.
Durch die Entschlüsselung wird das Lesen zu einem bewussten Akt.

Monotoner Cäsar-Code

Der monotone Cäsar-Code ist eine Verschiebechiffre auf dem Alphabet {A, …, Z, a, …, z}, bei der jeder Buchstabe um so viele Stellen zyklisch im Alphabet verschoben wird, wie seiner Position im jeweiligen Text entspricht.
Dieses Verfahren ist ein einfacher Sonderfall der Cäsar-Chiffre und damit für die Verschlüsselung von geheimen Nachrichten ungeeignet.
Interessant ist das Verfahren nur dahin gehend, dass die Entschlüsselung nicht trivial ist, sondern ein gewisses Maß an Aufwand erfordert.
Es hat, wie das ROT13-Verfahren, lediglich zum Ziel, bestimmte Inhalte, die eventuell als kompromittierend empfunden werden, nicht direkt lesbar zu machen.

Augustus-Chiffre

In seiner Autobiografie beschreibt der römische Kaiser Tiberius Claudius Nero Germanicus (10 v.u.Z.-54 u.Z.) ein Verschlüsselungsverfahren, dass von Augustus (Gaius Octavius, 63 v.u.Z.-14 u.Z.) verwendet worden sein soll.

Grundlage der Methode ist das Cäsar-Verfahren. Allerdings wird nicht jeder Buchstabe um den gleichen Wert im Alphabet zyklisch verschoben.
Vielmehr nutzt Augustus zur Festlegung der Codes die Buchstabenfolge eines von ihm gewählten Buches. Der erste Buchstabe des Klartextes wird damit mit dem ersten Buchstaben des Buches verknüpft, ebenso die zweiten, dritten usw. Als Geheimtext wird die Differenz der Positionen beider Buchstaben im Alphabet notiert. Ist die Differenz negativ wird 26 addiert.
Zur Dekodierung muss dann ein Zeichen des Geheimtextes nur um die entsprechende Differenz verringert werden, evtl. plus 26.

Beispiel: Der Klartext "Das ist ein Test" soll mit den ersten Buchstaben aus Wilhelm Buschs "Max und Moritz" verschlüsselt werden. Es wird

Klartext:	D A S I S T E I N T E S T
Codetext:	A C H W A S M U S S M A N
Geheimtext:	23 2 15 14 8 25 8 12 5 25 8 8 20

Mit der Verwendung eines unterschiedlichen Codes für jedes Zeichen wird das Verfahren polyalphabetisch und ähnelt dem Prinzip des Vigenere-Codes, der erst Jahrhunderte Jahre später entwickelt wurde.
In einer modernen Erweiterung werden alle darstellbaren Zeichen, inkl. Ziffern und Sonderzeichen, betrachtet. In diesem Fall wird das Augustus-Verfahren durchaus schwierig zu knacken, vorausgesetzt der Codetext ist nicht bekannt. Diese Variante wird hier im Programm demonstriert.

Kamasutra-Chiffre

Die Kamasutra-Chiffre ist eine der ältesten Substitutionsmethoden. Die Chiffre wurde im Kamasutra um etwa 400 v.u.Z. beschrieben. Die Absicht lag darin, Frauen beizubringen, wie sie geheime Nachrichten vor Schnüfflern schützen konnten. Die in Europa übliche Praxis, das "Vatsyayana Kamasutra" auf ein Erotiklehrbuch zu reduzieren, ist, gelinde gesagt, Blödsinn.
Bei der Kamasutra-Chiffre sind Klartext- und das Chiffretextalphabet identisch. Der Schlüssel der Chiffre ergibt sich durch die Permutation des Alphabets. Bei 26 Buchstaben existieren 25! = 15 511 210 043 330 985 984 000 000 mögliche Schlüssel, da kein Buchstabe in sich selbst überführt werden kann.
Anschließend wird das Alphabet in zwei Hälften aufgeteilt und untereinandergeschrieben. Damit kann man Buchstaben wie bei Atbash einander leicht zuordnen.

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

Der Buchstabe F wird zu einem C und ein B wird durch ein G ersetzt. Das Wort "BEISPIEL" würde somit zu "GMYPSYMA" chiffriert. Trotz der großen Schlüsselzahl lässt sich die Chiffre mit einer Häufigkeitsanalyse relativ schnell brechen.

Affine Chiffre

Die affine Chiffre ist ein Verschlüsselungsverfahren. Bei diesem Verfahren wird der Klartext Buchstabe für Buchstabe nach einer bestimmten mathematischen Formel verschlüsselt. Die affine Chiffre lässt sich ohne Aufwand berechnen, ist jedoch nicht besonders sicher.
Einerseits gibt es nur eine begrenzte Anzahl geheimer Schlüssel, sodass diese alle durchprobiert werden können. Andererseits kann der Geheimtext entschlüsselt werden, sobald die Verschlüsselung von nur zwei Zeichen bekannt ist.

Sender und Empfänger einigen sich vor Verwendung der affinen Chiffre auf einen geheimen Schlüssel. Dieser Schlüssel ist ein Zahlenpaar (a,b), wobei a eine der Zahlen 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 oder 25 und b eine beliebige Zahl von 0 bis 25 ist. Für a gilt dabei a < b und ggT(a,b) = 1 und b ist die Anzahl der Buchstaben des Alphabets.
Zur Verschlüsselung werden die Buchstaben des Alphabets fortlaufend durchnummeriert: A = 0, B = 1, …
Man verschlüsselt jeden Buchstaben einzeln, indem man seinen Zahlenwert x mit a multipliziert und anschließend b addiert. Das Ergebnis teilt man dann durch die Anzahl der Buchstaben des Alphabets, im lateinischen also 26.
Der bei dieser Division auftretende Rest ist die Nummer des verschlüsselten Buchstabens.
Zum Beispiel wird für den Buchstaben S (= 18) mit dem Schlüssel (7,15): (7·8+15) mod 26 = 11. Das S damit durch das L (= 11) ersetzt.

Entschlüsselung
Zu jeder Zahl a des Schlüssels existiert eine zweite Zahl a^{-1}, die der folgenden Tabelle entnommen werden kann.

a	1 3 5 7 9 11 15 17 19 21 23 25
a^{-1}	1 9 21 15 3 19 7 23 11 5 17 25

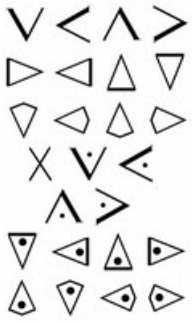
Mit den Zahlen a^{-1} und b des Schlüssels lässt sich der Geheimtext wieder entschlüsseln. Man nimmt dazu den Zahlenwert y des Geheimtextbuchstabens, berechnet a^{-1} (y - b) und teilt das Ergebnis wieder durch die Anzahl Buchstaben. Auch bei dieser Division tritt ein Rest auf, der die Nummer des Buchstabens vor der Verschlüsselung angibt.
Bei a = 1 handelt es sich um die als Caesar-Verschlüsselung bekannte Verschiebechiffre.

Alphabetum Kaldeorum

Das Alphabetum Kaldeorum ist eine der bekanntesten Geheimschriften des Mittelalters. Sein Name verweist auf das Volk der Chaldäer, die in der mittelalterlichen Ideenwelt für geheimnisvolles und magisches Wissen standen.
Überliefert ist es in einer Handschrift aus dem Jahre 1428, die sich heute in der Universitätsbibliothek München befindet; seine Ursprünge liegen jedoch in früherer Zeit, wie einige erhaltene Beispiele für die praktische Verwendung belegen.

Das Alphabetum Kaldeorum war in erster Linie zur Verschlüsselung diplomatischer Korrespondenz gedacht; sein Zeichenvorrat weist darauf hin, dass überwiegend lateinische Texte chiffriert wurden: u und v werden gleichgesetzt; w war als doppeltes v zu schreiben; j fehlt.
Für häufig auftretende Buchstaben sieht das Alphabetum Kaldeorum mehrere unterschiedliche Zeichen vor, die parallel verwendet wurden und eine Entschlüsselung nach der klassischen Häufigkeitsmethode verhindern sollten.
Ergänzend wurden in die chiffrierten Texte oft sogenannte "nulla" eingeschoben, sinnlose Zeichen, die wie Buchstaben aussahen und das Entschlüsseln durch Unbefugte zusätzlich erschwerten.

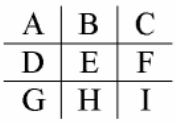
Als möglicher Urheber des Alphabetum Kaldeorum gilt Herzog Rudolf IV. von Österreich (1339-1365), der den Zeichen eine indische Herkunft zuschrieb.
Sogar die Grabplatte Rudolfs im Wiener Stephansdom trägt eine mittels Alphabetum Kaldeorum verschlüsselte Inschrift, die lediglich Namen und Titel des Herzogs wiedergibt.



Templer-Chiffre

Der Templer-Orden wurden im Jahr 1118 gegründet. Ihr voller Name war "Arme Ritterschaft Christi und des salomonischen Tempels zu Jerusalem (Pauperes commilitones Christi templique Salomonici Hierosalemitanis)". Nachdem sie immer größere Macht erlangten, stieg die Notwendigkeit, Nachrichten über Geld- und Reliquientransporte zu chiffrieren. Dadurch entstanden verschiedene Formen der Templerchiffre.

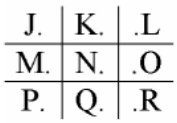
Alle Formen der Templer-Chiffre sind sehr einfach. Jeder Klartextbuchstabe bekommt ein festgelegtes Zeichen zugewiesen. Einen Schlüssel gibt es nicht.
Ein Alphabet A-Z kann in einer Variante der Templer-Chiffre aussehen, wie in der Abbildung. Da die Templer-Chiffre nur für 25 Zeichen vorgesehen ist, werden für das deutsche Alphabet I und J gleich gesetzt.
Die Templer-Chiffre kann mit einer Häufigkeitsanalyse in kürzester Zeit geknackt werden.
Dadurch, dass jedes Klartextzeichen einem anderen, unveränderlichen Geheimtextzeichen zugewiesen wird, ist die Templer-Chiffre ein Beispiel für monoalphabetische Substitution.



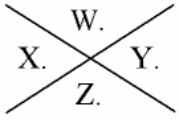
Freimaurer-Chiffre

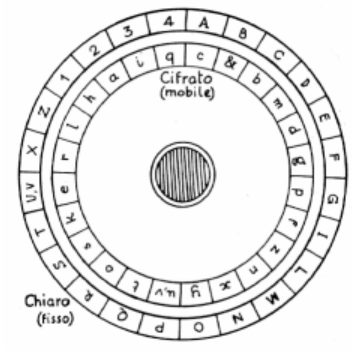
Auf einem Friedhof neben dem New Yorker Geschäftsviertel befindet sich der Grabstein des Freimaurers James Leason (gest. 28.9.1794). Auffällig ist, dass sich am oberen Teil des Grabsteines eine Reihe von Zeichen befinden, die nur gelesen werden können, wen man im Besitz des Freimaurer-Schlüssels ist.

Bei der Freimaurer-Chiffre handelt es sich um eine sehr einfache Ersetzung der Buchstaben durch Zeichen. Einen variablen Schlüssel gibt es nicht. Die Ersetzung basiert auf dem Schema der Abbildung.
Danach steht zum Beispiel < für ein U, |_ für ein C, •> für ein X und so weiter.



Diese monoalphabetische Verschlüsselung kann mit einer Häufigkeitsanalyse in kürzester Zeit geknackt werden.





Alberti-Kodierung

Leon Battista Alberti (1404-1472) beschrieb in seinem Werk "Trattato della cifra" ein auf dem Cäsar-Code basierendes Verschlüsselungsverfahren, dass allerdings wesentlich schwerer zu knacken ist.

Zur Kodierung benutzt er eine Chiffrierscheibe. Diese besteht aus zwei konzentrischen, beweglichen Kreisen. Auf den äußeren Kreis sind 24 Zeichen alphabetische sortiert geschrieben, lateinische Großbuchstaben außer dem H, K, W und Y, sowie die Ziffern 1, 2, 3 und 4.
Auf der inneren Scheibe findet man in zufälliger Reihenfolge 23 lateinische Kleinbuchstaben und das Zeichen &. Die Buchstaben u und w werden durch v vertreten, das j durch das i. Die Scheiben können gegeneinander verdreht werden.
Zur Kodierung eines Klartextes wird das & auf ein beliebiges Zeichen eingestellt. Jeder Klartextbuchstabe wird auf dem inneren Kreis gesucht und durch das Zeichen auf der äußeren Scheibe ersetzt.
Zum Beispiel wird bei der in der Abbildung gezeigten Einstellung aus CHEMNITZ
der kodierte Text A1VDM3QL

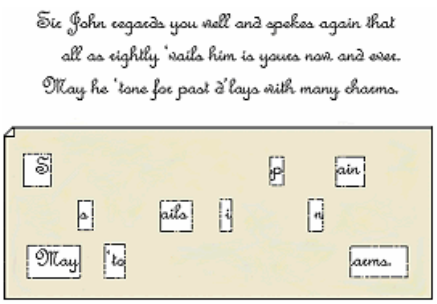
Durch die veränderte Reihenfolge der inneren Zeichen wird, obwohl monoalphabetisch, dennoch anspruchsvoller als beim Cäsar-Kode verschlüsselt. Das Knacken ist nur durch eine Häufigkeitsanalyse möglich. Alberti beschreibt eine zusätzliche Idee. Nach der Verschlüsselung eines Zeichens soll die Scheibe um ein Zeichen weiter gedreht werden. Damit entsteht ein polyalphabetisches Verschlüsselungsverfahren; das erste in der Geschichte beschriebene.

Bacon-Chiffre

Die Bacon-Chiffre wurde von Francis Bacon (1561-1626) entwickelt und stellt ein Steganographieverfahren dar.
Jedem Buchstaben des Klartextes wird ein fünfstelliger Code zugeordnet:

Buchstabe	Code	Buchstabe	Code	Buchstabe	Code
A	aaaaa	I, J	abaaa	R	baaaa
B	aaaab	K	abaab	S	baaab
C	aaaba	L	ababa	T	baaba
D	aaabb	M	ababb	U, V	baabb
E	aabaa	N	abbaa	W	babaa
F	aabab	O	abbab	X	babab
G	aabba	P	abbba	Y	babba
H	aabbb	Q	abbbb	Z	babbb

Das Wort "Mathe" wäre kodiert: "ababb aaaaa baaba aabbb aabaa". Diese Kodierung wird anschließend in einem Text versteckt.
1.Möglichkeit: Für die einzelnen Buchstaben eines Textes werden zwei unterschiedliche Schriftarten verwendet. Eine Schriftart steht für den Buchstaben a, die andere für den Buchstaben b.
Francis Bacon entwickelte dazu eine Handschrift aus 21 Zeichen, die für jeden Groß- und Kleinbuchstaben zwei unterschiedliche Darstellungen enthielt.
2.Möglichkeit: Im Text werden alle "a" als Kleinbuchstaben und die "b" als Großbuchstaben geschrieben.
Die Kodierung des Wortes "Mathe" ergibt dann zum Beispiel:
"dAs HÖchste LebEn isT MAthEmatik."
Bacon's Chiffre ist eine der ersten Anwendungen des Binärsystems in Europa.



Cardan-Gitter, Cardan-Schablone

1550 wurde von dem dem italienischen Mathematiker Gerolamo Cardano (1501-1576) ein Verschlüsselungsverfahren erfunden, dass das ihm benannte Cardan-Gitter, die Cardan-Schablone, benutzt. Das Verfahren gehört zu den Steganografieverfahren und war zu Zeit Cardanos eine wichtige Kodierungsmethode.

Auf eine Unterlage aus Papier wird ein Gitter gezeichnet. Einige der Felder der entstandenen Tabelle werden ausgeschnitten. Im Ergebnis erhält man eine Schablone mit Öffnungen, das individuelle Cardan-Gitter.

Zur Verschlüsselung eines Klartextes wird das individuelle Cardan-Gitter auf Papier gelegt und an den leeren Stellen der Klartext eingetragen. Nach der Entfernung der Schablone wird der Rest der Tabelle mit beliebigen Daten gefüllt. Zur Entschlüsselung der Botschaft benötigt man das individuelle Cardan-Gitter. Heute wird die Cardano-Verschlüsselung in modernisierter Form für Online-Banking genutzt.

IDVQ	I	O	A	B	C	D	F	G	H	L
	V	E	M	N	P	Q	R	S	T	X
OFER	I	O	A	B	C	D	F	G	H	L
	X	V	E	M	N	P	Q	R	S	T
AGMS	I	O	A	B	C	D	F	G	H	L
	T	X	V	E	M	N	P	Q	R	S
BHNT	I	O	A	B	C	D	F	G	H	L
	S	T	X	V	E	M	N	P	Q	R
CLPX	I	O	A	B	C	D	F	G	H	L
	R	S	T	X	V	E	M	N	P	Q

Bellaso-Chiffre

1553 veröffentlichte der italienische Mathematiker G.B.Bellaso ein kryptologisches Werk, in dem er eine neue Art von Chiffre vorstellte. Da ihm die Schwäche des monoalphabetischen Cäsar-Codes klar war, versuchte er eines der ersten polyalphabetischen Verschlüsselungsverfahren zu konstruieren.
Dabei wird mit vier verschiedenen Kodierungstabellen gearbeitet. Ausgehend von einem Schlüsselsatz in Latein (U=V, W=V und J=I), z.B. OPTARE MELIORA, wird für jedes Wort des Klartextes ein Buchstabe des Schlüsselsatzes gewählt und nach diesem die Kodierungstabelle aus der linken Abbildung.

Beispiel:
Der Klartext "DAS IST EIN TEST" soll mit der Bellaso-Chiffre kodiert werden.
Für das 1.Wort DAS wird die Tabelle für O gewählt. Es werden D zu Q, A zu M, S zu G.
Das 2.Wort IST wird mit der P-tabelle (oPtare) verschlüsselt und ergibt ROA, das 3.Wort mit der T-Tabelle sowie das 4.Wort mit der A-Tabelle.
Insgesamt erhält man als Geheimtext: QMG ROA CSF IBLI; polyalphabetisch da z.B. die S verschieden kodiert werden.

Beispiel 2: Klartext: "Inviare truppe domani"

Tabelle	O	P	T
Klartext	I N V I A R E	T R U P P E	D O M A N I
Geheimtext	X C O X E G A	A I C H H D	M T D X F S

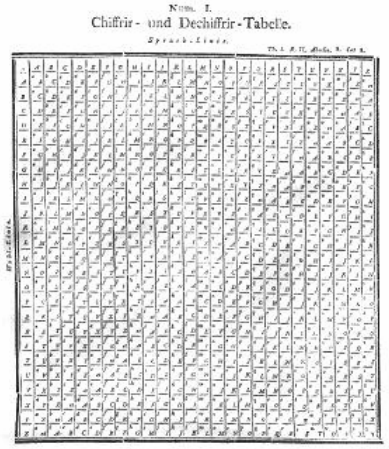
Die Bellaso-Chiffre ist ein direkter Vorgänger des Vigenere-Codes.
siehe http://critto.liceofoscarini.it/critto/bellaso.htm

Vigenere-Code

Gleiche Zeichen im Klartext haben nicht gleiche Zeichen im codierten Text
· Text und Schlüsseltext in Zahlen umsetzen (Schlüsselwort)
· diese zeichenweise addieren (evt. Schlüsseltext mehrfach aneinander setzen)
· falls die Summe größer ist als die Anzahl der zur Verfügung stehenden Zeichen, wird der Rest der Summe bei Division durch diese Zahl als Ergebnis genommen werden, sonst die Zahl selbst
· die erhaltenen Zahlen werden wieder in Zeichen umgesetzt und bilden dann den Geheimtext
· die Entschlüsselung funktioniert umgekehrt, also als Subtraktion des Schlüssels vom Geheimtext.

.	A	B	C	D	E	F	...
A	A	B	C	D	E	F	...
B	B	C	D	E	F	G	...
C	C	D	E	F	G	H	...
D	D	E	F	G	H	I	...
E	E	F	G	H	I	J	...
F	F	G	H	I	J	K	...
...	...	...	...	...	...	...	...

Ergibt sich als Differenz eine negative Zahl, so wird die Anzahl der zur Verfügung stehenden Zeichen addiert
Nach einer Modifizierung durch Lewis Caroll kann zum Ver- und Entschlüsseln eine Vigenere-Tafel genutzt werden:
Steht ein Zeichen des Textes in Zeile i und das zugehörige Zeichen des Schlüsselwortes in Spalte j der Vigenere-Tafel, so kann das verschlüsselte Zeichen im Schnittpunkt der Zeile und Spalte abgelesen werden. Die Entschlüsselung erfolgt in umgekehrter Reihenfolge.


Num. I.
Chiffrir- und Dechiffrir-Tabelle.

Johann Ludwig Klübers "Lehrbuch der Kryptographik", 1809 bei Cotta in Tübingen erschienen, zeigt in seinem Kapitel über "Buchstabenchiffre" ein sogenanntes "Vigenére-Quadrat", mithin eine polyalphabetische Verschlüsselung.

Anhand eines Schlüsselwortes wird in der vertikalen Buchstabenreihe der Klartextbuchstabe aufgesucht; von ihm aus trifft man horizontal in der obersten Reihe auf den zugehörigen Geheimtextbuchstaben.
Im Streben nach möglichst kurzen C-Programmen hat Bruno Van Wilder von der Londoner Universität das nachfolgende Programm zum Kodieren und Dekodieren nach Vigenere geschaffen:

```
main(int
n,char**a){for(n=0;putchar(a[2][n]?(a[2][n]%32+(**a%2*2-1)*
     (a[1][n++%(a[2] -a[1] -1)]%32-1)+25)%26+97:10)-10;);}
```

Quelle: http://www.iwriteiam.nl/SigProgC.html

Diese Chiffre wurde von Blaise de Vigenère (1523-1596) entwickelt. Statt einem werden 26 Alphabete verwendet, bei denen jedes Alphabet durch eine zyklische Rotation um 1 zum Vorhergehenden erzeugt wird.

JavaScript-Texte zum Ver- und Entschlüsseln

```
 1:  var alphabet = "abcdefghijklmnopqrstuvwxyz"; var alphaLen = alphabet.length;
 2:  function vigenere_encrypt (key, text) {
 3:    var i, j = 0; var zp, kp  = 0; var chiffre = "";
 4:    if (key != "") {  key  = key.toLowerCase();  text = text.toLowerCase();
 5:       for (i=0; i < text.length; i++) { zp = alphabet.indexOf(text.charAt(i));
 6:          if (zp >= 0) { kp = alphabet.indexOf(key.charAt(j));
 7:             chiffre = chiffre + alphabet.charAt((zp+kp) % alphaLen);
 8:             j = (j+1) % key.length;
 9:          } else chiffre = chiffre + text.charAt(i); } }
10:    return chiffre; }

11:  function vigenere_decrypt (key, chiffre) {
12:    var i, j   = 0;  var zp, kp = 0;  var text   = "";
13:    if (key != "") {  key     = key.toLowerCase();  chiffre = chiffre.toLowerCase();
14:       for (i=0; i < chiffre.length; i++) {  zp = alphabet.indexOf(chiffre.charAt(i));
15:          if (zp >= 0) {  kp = alphabet.indexOf(key.charAt(j));
16:             text = text + alphabet.charAt((alphaLen+(zp-kp)) % alphaLen);
17:             j = (j+1) % key.length;
18:          } else text = text + text.charAt(i); } }
19:    return text; }
```

Gronsfeld-Verfahren

Eine Variante des Vigenere-Codes ist die von Gronsfeld vorgeschlagene Verinfachung.
Dieses Verfahren benutzt nur 10 der 26 möglichen Schlüssel des Vigenere-Verfahrens, also nur die Schlüsselalphabete A bis J, die in diesem Verfahren durch die Zahlen 0 bis 9 dargestellt werden. Damit kann man anstatt eines Schlüsselwortes auch eine Codezahl verwenden.

Schlüssel von Gronsfeld

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

Dieses Verfahren ist allerdings wesentlich leichter zu knacken, da es für jeden Buchstaben nur 10 verschiedene Verschlüsselungsmöglichkeiten gibt.
Ein Z im verschlüsselten Text kann somit z.B. nur den Buchstaben Q-Z entsprechen. Niemals können die Buchstaben A-P durch ein Z verschlüsselt werden. Damit ist die Leistungsfähigkeit des Verfahrens sehr eingeschränkt.

Kasiski-Friedman-Test

Mit der kombinierten Methode von Kasiski und Friedman ist es möglich, Vigenere-Chiffren zu brechen. Dabei wird die Tatsache ausgenutzt, dass bei diesem Chiffrierverfahren das Schlüsselwort periodisch verwendet wird. Es treten also Wiederholungen von Teilfolgen im Schlüsseltext auf, wenn gleiche Klartextfolgen mit gleichen Schlüsselfolgen verschlüsselt worden sind. Der Abstand solcher übereinstimmender Teilfolgen mit der Länge l > 2 im Schlüsseltext ist ein Vielfaches der Schlüssellänge. Gibt es mehrere sich wiederholende Schlüsseltextfolgen, dann muss die Schlüssellänge den größten gemeinsamen Teiler der Abstände teilen. Diese Überlegung wird Kasiski-Text genannt. Man muss aber die Möglichkeit in Betracht ziehen, dass solche Übereinstimmungen auch durch Zufall entstanden sein könnten und damit das Ergebnis verfälschen würden.
Während der Kasiski-Text die Schlüsselwortlänge nur bis auf Vielfache und Teiler liefert, gibt der Friedman-Test die Größenordnung der Schlüsselwortlänge an. Für die Schlüsselwortlänge l eines Vigenere-verschlüsselten Klartextes in deutscher Sprache mit einem Schlüsseltext der Länge n (Zeichenzahl) gilt

$$l = 0.0377\ n\ /\ ((n-1)\ IC - 0.0385\ n + 0.0762)$$

Dabei ist IC der Koinzidenzindex des Schlüsseltextes, der sich wie folgt aus den Anzahlen n_i der Buchstaben a_i des Schlüsseltextes berechnen lässt

$$IC = \text{Summe}\ (n_i\ (n_i - 1))\ /\ (n\ (n-1))\ ,\ \text{Summenbildung über alle Buchstaben}$$

Zur Ermittlung des Schlüsselwortes schreibt man den Schlüsseltext der Länge n in l Spalten. Es genügt nun, spaltenweise das Äquivalent der Buchstaben E zu finden, da die Spalten bei der Vigenere-Chiffre durch eine Verschiebechiffre entstanden sind.

Statistische Analyse

Für jede natürliche Sprache gibt es Verteilungen der Häufigkeiten von Einzelbuchstaben, Buchstabenpaaren, Worten usw. Zum Beispiel ist in der deutschen Sprache E der häufigste Buchstabe.

Buchstaben	Gesamthäufigkeiten	Buchstaben	Gesamthäufigkeiten
E, N	27,18 %	I, S, R, A, T	34,48 %
D, H, U, L, C, G, M, O, B, W, F, K, Z	36,52 %	P, V, J, Y, X, Q	1,82 %

Für ausreichend lange Schlüsseltexte ist es unter Ausnutzung der Häufigkeitsverteilungen möglich, monoalphabetische monographische Substitutionen zu brechen.

Prozentuale Häufigkeit der Buchstaben in deutschen Texten

A	6.51 %	B	1.89 %	C	3.06 %	D	5,08 %
E	17.40 %	F	1.66 %	G	3.01 %	H	4.76 %
I	7.55 %	J	0.27 %	K	1.21 %	L	3.44 %
M	2.53 %	N	9.78 %	O	2.51 %	P	0.79 %
Q	0.02 %	R	7.00 %	S	7.27 %	T	6.51 %
U	4.35 %	V	0.67 %	W	1.89 %	X	0.03 %
Y	0.04 %	Z	1.13 %				

Prozentuale Häufigkeit der Buchstaben in englischen Texten

A	8.56 %	B	1.39 %	C	2.79 %	D	3.78 %
E	13.04 %	F	2.89 %	G	1.99 %	H	5.28 %
I	6.27 %	J	0.13 %	K	0.42 %	L	3.39 %
M	2.49 %	N	7.07 %	O	7.97 %	P	1.99 %
Q	0.12 %	R	6.77 %	S	6.07 %	T	10.45 %
U	2.49 %	V	0.92 %	W	1.49 %		
X	0.17 %	Y	1.99 %	Z	0.08 %		

N-Gramme

Um Kryptoanalyse betreiben zu können, nutzt man die Gesetzmäßigkeiten der jeweiligen Sprache. Auftretende Muster sind die Art und Weise, wie sich Buchstaben in einem Wort wiederholen. Solche Muster bleiben bei der monoalphabetischen Chiffrierung erhalten, d.h. enthält ein Geheimtext keine Muster, so ist er nicht durch monoalphabetischer Chiffrierung entstanden.
Die Häufigkeiten der Einzelbuchstaben hängen vom Genre des Textes ab. So schreibt Beutelsbacher:

"Ein von Zitaten strotzender zoologischer Text über den Einfluß von Ozon auf die Zebras im Zentrum von Zaire wird eine andere Häufigkeitsverteilung ausweisen, als ein Traktat über die amourösen Adventüren des Balthasar Matzbach am Rande des Panamakanals."

Die Verschlüsselung des Wortzwischenraumes führt zu einem Alphabet mit 27 Buchstaben. Da die Leerstelle im Deutschen nach dem e das häufigste Zeichen und leicht zu finden ist, lässt der professionelle Chiffrierer diesen Zwischenraum oft weg. Dies erschwert die Kryptoanalyse allerdings nur unwesentlich.

Häufigkeiten von n-Grammen
N-Gramme sind Kolonnen von n Buchstaben. Die Tabelle zeigt die Häufigkeiten für einige Bigramme.

Bigramm	engl. Häufigkeit in %	Bigramm	dt. Häufigkeit in %
th	3,15	en	3,88
he	2,51	er	3,75
an	1,72	ch	2,75
in	1,69	te	2,26
er	1,54	de	2,00
re	1,48	nd	1,99
on	1,45	ei	1,88
es	1,45	ie	1,79
ti	1,28	in	1,67
at	1,24	es	1,52

Im Deutschen kommt ch sehr häufig vor, nahezu niemals hc. Des Weiteren kommen ei und ie gleichhäufig vor.

Die häufigsten Trigramme sind

deutsch	Häufigkeit in %	englisch	Häufigkeit in %
ein	1,22	the	3,53
ich	1,11	ing	1,11
nde	0,89	and	1,02
die	0,87	ion	0,75
und	0,87	tio	0,75
der	0,86	ent	0,73
che	0,75	ere	0,69

Häufige Viergramme:
deutsch icht, keit, heit, chon, chen, cher, urch, eich, …

Aufschluß über den Ursprung des Textes kann auch die mittlere Wortlänge geben:

deutsch	5,9	italienisch	4,5
englisch	4,5	spanisch	4,4
französisch	4,4	russisch	6,3

aber auch die zehn häufigsten Wörter:

deutsch	die, der, und, den, am, in, zu, ist, dass, es
englisch	the, of, and, to, a, in, that, it, is, I
französisch	de, il, le, et, que, je, la, ne, on, les
italienisch	la, di, che, il, non, si, le, una, lo, in
spanisch	de, la, el, que, en, no, con, un, se, sa

Koinzidenzindex einer Sprache

Um den Koinzidenzindex (Kappa κ) einer Sprache zu bestimmen, schreibt man zwei gleichlange unterschiedliche Texte untereinander und zählt alle Spalten mit gleichen Buchstaben. Anschließend teilt man diese Zahl durch die Anzahl der Buchstaben in einer Zeile.

Für zwei Texte gleicher Länge $T_x = x_1x_2...x_n$ und $T_y = y_1y_2...y_n$
definiert man $\kappa(T_x, T_y) = \Sigma_{i=1}^{n} \delta(x_i, y_i)/n$
wobei $\delta(x, y) = 1$ für $x = y$ anderfalls 0 ist.
Günstiger ist die Formel $\kappa = \Sigma_{i=1}^{26} n_i^2 / (n (n-1))$
wobei n die Gesamtzahl der Buchstaben ist und n_i die absolute Häufigkeit der einzelnen Buchstaben.

Es zeigt sich, dass jede Sprache ihr eigenes Kappa hat:

deutsch 7,62 %	spanisch 7,75 %	englisch 6,61 %
japanisch 8,19 %	französisch 7,78 %	russisch 5,29%
italienisch 7,38 %		

Kommt jeder Buchstabe bei einem Alphabet aus 26 Buchstaben mit der gleichen Wahrscheinlichkeit von 1/26 vor, so ergibt sich der Wert $\kappa = 1/26 = 3{,}85$ %.
Durch Zerschneiden des Geheimtextes in zwei Hälften und der Ermittlung des Kappa der beiden so entstandenen Texte kann festgestellt werden, mit welcher Sprache der Klartext verfasst wurde oder ob es eine Kunstsprache ist.
"Ein ideal für die Chiffrierung vorbereiteter Klartext ist orthographisch falsch, sprachlich knapp und stilistisch grauenhaft." Bauer

Lesbarkeitsindex

Ein Lesbarkeitsindex ist eine Gleichung, mit der die Lesbarkeit eines Textes bestimmt wird. Die ersten Lesbarkeitsformeln wurden für Englisch konstruiert.

Flesch Reading Ease

Der Flesch Reading Ease FRE ist ein numerischer Wert für die Lesbarkeit. Je höher der Wert, desto leichter verständlich der Text.

FRE = 206,835 - 1,015 ASL - 84,6 ASW

Die durchschnittliche Satzlänge ASL (Average Sentence Length) ergibt sich, indem die Anzahl der Wörter im Text durch die Anzahl der Sätze des Textes dividiert wird.
Die durchschnittliche Silbenanzahl pro Wort ASW (Average Number of Syllables per Word) ergibt sich, indem die Silbenanzahl des gesamten Textes durch die Anzahl der Wörter im Text dividiert wird.

Durch Toni Amstad wurde der Index auf die deutsche Sprache übertragen. Da deutsche Wörter im Allgemeinen länger als englische sind, ergibt sich FRE(Deutsch) = 180 - ASL - 58,5 ASW
Mit dem ermittelten Wert kann eine ungefähre Einordnung eines Textes gegeben werden:

FRE von … bis	Lesbarkeit	verständlich für
0-30	sehr schwer	Akademiker
30-50	schwer	
50-60	mittelschwer	
60-70	mittel	13-15jährige Schüler
70-80	mittelleicht	
80-90	leicht	
90-100	sehr leicht	11jährige Schüler

Shannon-Entropie

Die Entropie ist ein Maß für den mittleren Informationsgehalt oder auch Informationsdichte einer Nachricht; eine Analogie zur Entropie in der Thermodynamik. Die informationstheoretische Entropie geht auf Claude E.Shannon zurück, die er 1948 einführte.
Die Entropie wird mit einem großen Eta H bezeichnet.

Die Entropie H einer diskreten Zufallsvariable X über einem endlichen Alphabet Z wird wie folgt definiert: Es sei z aus Z, m die Mächtigkeit von Z und ist p_z P(X=z) die Wahrscheinlichkeit, mit der das Zeichen z des Alphabets auftritt. Dann ist $H = \Sigma_{i=1}^{m} p_i \log_2 p_i$
Summanden mit der Wahrscheinlichkeit 0 werden nicht berücksichtig.
Die Entropie ist ein Maß für den mittleren Informationsgehalt pro Zeichen einer Quelle, die ein System oder eine Informationsfolge darstellt.
Je mehr Zeichen im Allgemeinen von einer Quelle empfangen werden, desto mehr Information erhält man und gleichzeitig sinkt die Unsicherheit über das, was hätte gesendet werden können.

Shannon-Entropie

Die Entropie ist ein Maß für den mittleren Informationsgehalt oder auch Informationsdichte einer Nachricht; eine Analogie zur Entropie in der Thermodynamik. Die informationstheoretische Entropie geht auf Claude E.Shannon zurück, die er 1948 einführte.
Die Entropie wird mit einem großen Eta H bezeichnet.

Die Entropie H einer diskreten Zufallsvariable X über einem endlichen Alphabet Z wird wie folgt definiert: Es sei z aus Z, m die Mächtigkeit von Z und ist p_z P(X=z) die Wahrscheinlichkeit, mit der das Zeichen z des Alphabets auftritt. Dann ist $H = \Sigma_{i=1}^{m} p_i \log_2 p_i$
Summanden mit der Wahrscheinlichkeit 0 werden nicht berücksichtig.
Die Entropie ist ein Maß für den mittleren Informationsgehalt pro Zeichen einer Quelle, die ein System oder eine Informationsfolge darstellt.
Je mehr Zeichen im Allgemeinen von einer Quelle empfangen werden, desto mehr Information erhält man und gleichzeitig sinkt die Unsicherheit über das, was hätte gesendet werden können.

Worttabellen

Für eine erfolgreiche Kryptoanalyse ist nicht nur die Häufigkeit auftretender Buchstaben, sondern auch die in der jeweiligen Sprache besonders oft auftretenden Wörter von Bedeutung. Die nachfolgenden Listen geben die in deutsch, englisch und französisch auf häufigsten auftretenden Wörter wieder; geordnet von links nach rechts.
Die Tabellen wurden durch Auswertung führender Medien erstellt. Berücksichtigt wurden außer normalen Begriffen auch Abkürzungen sowie die Klein- und Großschreibung.

Häufigste Wörter in der deutschen Sprache

der	die	und	in	den	von	zu	das
mit	sich	des	auf	für	ist	im	dem
nicht	ein	Die	eine	als	auch	es	an
werden	aus	er	hat	daß	sie	nach	wird
bei	einer	Der	um	am	sind	noch	wie
einem	über	einen	Das	so	Sie	zum	war
haben	nur	oder	aber	vor	zur	bis	mehr
durch	man	sein	wurde	sei	In	Prozent	hatte
kann	gegen	vom	können	schon	wenn	habe	seine
Mark	ihre	dann	unter	wir	soll	ich	eines
Es	Jahr	zwei	Jahren	diese	dieser	wieder	keine
Uhr	seiner	worden	Und	will	zwischen	Im	immer
Millionen	Ein	was	sagte	Er	gibt	alle	DM
diesem	seit	muß	wurden	beim	doch	jetzt	waren
drei	Jahre	Mit	neue	neuen	damit	bereits	da
Auch	ihr	seinen	müssen	ab	ihrer	Nach	ohne
sondern	selbst	ersten	nun	etwa	Bei	heute	ihren
weil	ihm	seien	Menschen	Deutschland	anderen	werde	Ich
sagt	Wir	Eine	rund	Für	Aber	ihn	Ende
jedoch	Zeit	sollen	ins	Wenn	So	seinem	uns
Stadt	geht	Doch	sehr	hier	ganz	erst	wollen
Berlin	vor allem	sowie	hatten	kein	deutschen	machen	lassen
Als	Unternehmen	andere	ob	dieses	steht	dabei	wegen
weiter	denn	beiden	einmal	etwas	Wie	nichts	allerdings
vier	gut	viele	wo	viel	dort	alles	Auf
wäre	SPD	kommt	vergangenen	denen	fast	fünf	könnte
nicht nur	hätten	Frau	Am	dafür	kommen	diesen	letzten
zwar	Diese	großen	dazu	Von	Mann	Da	sollte

würde	also	bisher	Leben	Milliarden	Welt	Regierung	konnte
ihrem	Frauen	während	Land	zehn	würden	stehen	ja
USA	heißt	dies	zurück	Kinder	dessen	ihnen	deren
sogar	Frage	gewesen	erste	gab	liegt	gar	davon
gestern	geben	Teil	Polizei	dass	hätte	eigenen	kaum
sieht	große	Denn	weitere	Was	sehen	macht	Angaben
weniger	gerade	läßt	Geld	München	deutsche	allen	darauf
wohl	später	könne	deshalb	aller	kam	Arbeit	mich
gegenüber	nächsten	bleibt	wenig	lange	gemacht	Wer	Dies
Fall	mir	gehen	Berliner	mal	Weg	CDU	wollte
sechs	keinen	Woche	dagegen	alten	möglich	gilt	erklärte
müsse	Dabei	könnten	Geschichte	zusammen	finden	Tag	Art
erhalten	Man	Dollar	Wochen	jeder	nie	bleiben	besonders
Jahres	Deutschen	Den	Zu	zunächst	derzeit	allein	deutlich
Entwicklung	weiß	einige	sollten	Präsident	geworden	statt	Bonn
Platz	inzwischen	Nur	Freitag	Um	pro	seines	Damit
Montag	Europa	schließlich	Sonntag	einfach	gehört	eher	oft
Zahl	neben	hält	weit	Partei	meisten	Thema	zeigt
Politik	Aus	zweiten	Januar	insgesamt	je	mußte	Anfang
hinter	ebenfalls	ging	Mitarbeiter	darüber	vielen	Ziel	darf
Seite	fest	hin	erklärt	Namen	Haus	An	Frankfurt
Gesellschaft	Mittwoch	damals	Dienstag	Hilfe	Mai	Markt	Seit
Tage	Donnerstag	halten	gleich	nehmen	solche	Entscheidung	besser
alte	Leute	Ergebnis	Samstag	Daß	sagen	System	März
tun	Monaten	kleinen	lang	Nicht	knapp	bringen	wissen
Kosten	Erfolg	bekannt	findet	daran	künftig	wer	acht
Grünen	schnell	Grund	scheint	Zukunft	Stuttgart	bin	liegen
politischen	Gruppe	Rolle	stellt	Juni	sieben	September	nämlich
Männer	Oktober	Mrd	überhaupt	eigene	Dann	gegeben	Außerdem
Stunden	eigentlich	Meter	ließ	Probleme	vielleicht	ebenso	Bereich
zum Beispiel	Bis	Höhe	Familie	Während	Bild	Ländern	Informationen
Frankreich	Tagen	schwer	zuvor	Vor	genau	April	stellen
neu	erwartet	Hamburg	sicher	führen	Mal	Über	mehrere
Wirtschaft	Mio	Programm	offenbar	Hier	weiteren	natürlich	konnten
stark	Dezember	Juli	ganze	kommenden	Kunden	bekommen	eben
kleine	trotz	wirklich	Lage	Länder	leicht	gekommen	Spiel
laut	November	kurz	politische	führt	innerhalb	unsere	meint
immer wieder	Form	Münchner	AG	anders	ihres	völlig	beispielsweise
gute	bislang	August	Hand	jede	GmbH	Film	Minuten
erreicht	beide	Musik	Kritik	Mitte	Verfügung	Buch	dürfen
Unter	jeweils	einigen	Zum	Umsatz	spielen	Daten	welche
müßten	hieß	paar	nachdem	Kunst	Euro	gebracht	Problem
Noch	jeden	Ihre	Sprecher	recht	erneut	längst	europäischen
Sein	Eltern	Beginn	besteht	Seine	mindestens	machte	Jetzt
bietet	außerdem	Bürger	Trainer	bald	Deutsche	Schon	Fragen
klar	Durch	Seiten	gehören	Dort	erstmals	Februar	zeigen
Titel	Stück	größten	FDP	setzt	Wert	Frankfurter	Staat
möchte	daher	wolle	Bundesregierung	lediglich	Nacht	Krieg	Opfer
Tod	nimmt	Firma	zuletzt	Werk	hohen	leben	unter anderem
Dieser	Kirche	weiterhin	gebe	gestellt	Mitglieder	Rahmen	zweite

Häufigste Wörter in der englischen Sprache

the	of	to	and	a	in	for	is
The	that	on	said	with	be	was	by
as	are	at	from	it	has	an	have
will	or	its	he	not	were	which	this
but	can	more	his	been	would	about	their
also	they	million	had	than	up	who	In
one	you	new	A	I	other	year	all
two	S	But	It	company	into	U	Mr.
system	some	when	out	last	only	after	first
time	says	He	years	market	no	over	we
could	if	people	percent	such	This	most	use
because	any	data	there	them	government	may	software
so	New	now	many	used	program	systems	three
do	Inc	between	billion	what	through	per	make

before	should	these	down	under	made	Corp	companies
much	work	users	officials	like	business	support	just
each	those	her	well	since	she	information	both
being	your	products	president	share	sales	where	say
get	network	against	We	while	even	If	week
price	group	way	during	state	still	same	then
number	For	They	our	very	shares	next	including
computer	set	applications	off	available	York	how	prices
take	him	American	using	must	part	back	expected
good	stock	And	high	another	several	months	end
report	IBM	product	industry	don't	own	four	major
need	power	management	control	did	less	rate	file
called	United	As	Soviet	There	second	without	want
day	found	month	reported	see	development	help	cent
public	today	Co	money	case	interest	programs	PC
trading	based	five	cost	days	large	problems	too
service	line	user	federal	few	At	include	told
does	going	plan	C	technology	increase	until	performance
go	provide	might	results	offer	least	run	small
it's	operating	current	Windows	Bush	National	police	different
quarter	my	memory	according	plans	long	ago	around
version	disk	rose	Friday	early	problem	think	President
process	One	services	costs	oil	home	area	higher
Mr	former	Monday	however	among	spokesman	board	already
earlier	announced	standard	issue	rates	Tuesday	application	within
little	economic	When	members	office	files	recent	States
later	Wednesday	better	West	past	On	show	come
chief	whether	That	pay	change	know	unit	six
put	agreement	cents	Some	example	Department	Thursday	Pounds
hard	added	House	local	company's	law	world	right
production	You	value	order	trade	financial	possible	far
access	design	point	However	political	become	features	fell
late	It's	news	San	important	Systems	move	director
military	total	court	here	foreign	began	Washington	N
operations	future	best	average	growth	tax	lower	meeting
left	drive	issues	give	us	country	official	big
held	executive	level	general	investment	John	An	came
didn't	chairman	US	find	me	close	June	range
given	interface	These	Bank	buy	To	Japanese	real
took	place	bank	making	includes	result	firm	With
database	model	reports	half	every	points	March	never
enough	provides	yesterday	times	addition	No	life	common
changes	DOS	doesn't	further	top	decision	compared	likely
due	period	open	markets	call	capital	name	national
closed	recently	proposed	single	able	lot	sold	South
package	test	customers	British	cut	sell	International	full
running	equipment	computers	index	working	Congress	contract	paper
workers	vice	study	others	System	research	low	makes
Federal	city	May	got	family	policy	investors	record
loss	received	April	Exchange	code	graphics	agency	increased
manager	keep	look	often	designed	European	earnings	environment
July	job	third	water	net	banks	analysts	strong
party	economy	away	dollar	taken	developed	continue	allow
Microsoft	key	either	security	project	agreed	though	Japan
rather	countries	plant	along	Apple	action	After	screen
war	processing	employees	included	asked	special	field	energy
old	deal	offers	nearly	weeks	debt	She	charges
Union	needs	effect	income	uses	again	Computer	East
similar	Europe	near	create	form	main	free	largest
return	machine	hours	yet	text	almost	All	man
required	hardware	private	allows	killed	international	known	things
base	bonds	groups	list	Mac	done	means	additional
gas	areas	trying	force	isn't	funds	lost	once
can't	server	Texas	OS	City	calls	space	annual

Häufigste Wörter in der französischen Sprache

de	la	le	et	les	des	en	un
du	une	que	est	pour	qui	dans	a
par	plus	pas	au	sur	ne	se	Le

ce	il	sont	La	Les	ou	avec	son
Il	aux	d'un	En	cette	d'une	ont	ses
mais	comme	on	tout	nous	sa	Mais	fait
été	aussi	leur	bien	peut	ces	y	deux
A	ans	l	encore	n'est	marché	d	Pour
donc	cours	qu'il	moins	sans	C'est	Et	si
entre	Un	Ce	faire	elle	c'est	peu	vous
Une	prix	On	dont	lui	également	Dans	effet
pays	cas	De	millions	Belgique	BEF	mois	leurs
taux	années	temps	groupe	ainsi	toujours	société	depuis
tous	soit	faut	Bruxelles	fois	quelques	sera	entreprises
F	contre	francs	je	n'a	Nous	Cette	dernier
était	Si	s'est	chez	L	monde	alors	sous
actions	autres	Au	ils	reste	trois	non	notre
doit	nouveau	milliards	avant	exemple	compte	belge	premier
s	nouvelle	Elle	l'on	terme	avait	produits	cela
d'autres	fin	niveau	bénéfice	toute	travail	partie	trop
hausse	secteur	part	beaucoup	Je	valeur	croissance	rapport
USD	aujourd'hui	année	base	Bourse	lors	vers	souvent
vie	l'entreprise	autre	peuvent	bon	surtout	toutes	nombre
fonds	point	grande	jour	va	avoir	nos	quelque
place	grand	personnes	plusieurs	certains	d'affaires	permet	politique
cet	chaque	chiffre	pourrait	devrait	produit	l'année	Par
rien	mieux	celui	qualité	France	Ils	Ces	s'agit
vente	jamais	production	action	baisse	Avec	résultats	Des
votre	risque	début	banque	an	voir	avons	qu'un
qu	elles	moment	qu'on	question	pouvoir	titre	doute
long	petit	d'ailleurs	notamment	FB	droit	qu'elle	heures
cependant	service	Etats-Unis	qu'ils	l'action	jours	celle	demande
belges	ceux	services	bonne	seront	économique	raison	car
situation	Depuis	entreprise	me	nouvelles	n'y	possible	toutefois
tant	nouveaux	selon	parce	dit	seul	qu'une	sociétés
vient	jusqu	quatre	marchés	mise	seulement	Van	semble
clients	Tout	Cela	serait	fort	frais	lieu	gestion
font	quand	capital	gouvernement	projet	grands	réseau	l'autre
données	prendre	plan	points	outre	pourtant	Ainsi	ni
type	Europe	pendant	Comme	mesure	actuellement	public	dire
important	mis	partir	parfois	nom	n'ont	veut	présent
passé	forme	autant	développement	mettre	grandes	vue	investisseurs
D	trouve	maison	mal	l'an	moyen	choix	doivent
NLG	direction	Sur	simple	période	enfants	dollars	personnel
assez	programme	général	banques	eux	semaine	président	personne
européenne	moyenne	tard	loi	petite	certaines	savoir	loin
explique	plupart	jeunes	cinq	contrat	Banque	valeurs	seule
rendement	nombreux	fonction	offre	client	activités	eu	environ
ministre	cadre	sens	étaient	sécurité	recherche	Paris	sorte
décembre	Son	suite	davantage	ensuite	janvier	donne	vrai
cause	d'abord	conditions	suis	juin	peine	certain	septembre
sommes	famille	l'indice	pris	laquelle	directeur	qu'en	propose
gens	derniers	étant	fut	chose	portefeuille	obligations	afin
différents	technique	Aujourd'hui	ailleurs	P	l'ensemble	américain	ventes
Selon	rue	livre	octobre	vraiment	sein	Or	dollar
Enfin	haut	Plus	petits	porte	tel	durée	domaine
aurait	jeune	présente	passe	PC	lorsque	choses	puis
Vous	aucun	l'un	n'en	tandis	coup	existe	propre
carte	crise	importante	atteint	revenus	montant	forte	ici
s'il	Quant	vu	rapidement	j'ai	ville	etc	mars
s'en	mon	premiers	bas	marque	véritable	ligne	longtemps
propres	devant	passer	départ	pu	total	série	quoi
particulier	concurrence	élevé	position	connu	principe	tendance	court
n	pages	évidemment	résultat	aura	parmi	Sans	américaine
face	trouver	durant	femmes	construction	désormais	distribution	telle
difficile	autour	européen	pratique	centre	vendre	juillet	mai
région	sociale	filiale	film	h	besoin	mode	Pas
représente	réalité	femme	vaut	Tél	aucune	hommes	donner
titres	l'Europe	nombreuses	différentes	moyens	formation	chiffres	Générale

dix	prochain	l'Etat	genre	bureau	communication	participation	gros
pourquoi	estime	devient	réalisé	création	novembre	l'évolution	pourra
semaines	consommation	faible	terrain	site	droits	moitié	puisque
Du	reprise	compris	projets	avril	vont	call	donné
simplement	six	firme	perte	Bien	Philippe	sait	prend
vite	via	stratégie	vos	jeu	J	petites	marketing
presque	Michel	manque	réaliser	financiers	Car	Comment	voiture
chef	constitue	Internet	J'ai	enfin	net	charge	nature
second	payer	actuel	Elles	investissements	dispose	financier	d'achat

Diceware

Diceware ist eine einfaches Verfahren, sichere und leicht erinnerbare Passwörter mit Hilfe eines Würfels zu erzeugen.
Für die Erzeugung des Passwortes werden mehrere Wörter aus einer speziellen Wortliste ausgewählt und aneinander gehängt. Die Worte werden mittels eines Würfels, der hier als Zufallszahlengenerator dient, ermittelt. Für jedes Wort werden fünf Würfelwürfe gebraucht. Anhand dieser Zahl wird das zugehörige Wort aus der Wortliste ausgewählt. Derzeit existieren Wortlisten für mehrere Sprachen.
Trotz der großen Länge der Passphrasen kann man sich gut an sie erinnern, da die einzelnen Wörter als Einheiten gemerkt werden können. Um eine ausreichende Sicherheit zu erhalten, sollten mindestens fünf Wörter zu einer Passphrase zusammengeführt werden.

Beispiel: Es wird mit einem Würfel fünf mal gewürfelt. Die Zahlen lauten 4,3,1,4 und 2.
Das zugehörige Wort wird in der Wortliste nachgeschlagen. Im Beispiel das Wort "merken".
Die zwei Schritte werden vier mal wiederholt. Insgesamt ergeben sich z.B. folgende Paare:

43142	merken	15613	boom
22543	ekd	66445	zonen
51615	ragt		

Das Passwort lautet merkenboomekdzonenragt.

Jedes Diceware-Wort fügt 12,9 Bit Entropie zu einer Passphrase hinzu. Dies entspricht $\log_2(65)$ Bit. Fünf Wörter entsprechen 64 Bit und werden als Minimum angesehen, um als sicher zu gelten.
Rechts kann nach Eingabe der Zahlenfolge das deutsche Wort in der Liste nachgeschlagen werden.

Autokey-Chiffre

Die Autokey-Chiffre wurde ebenfalls von Blaise de Vigenère entworfen.
Sie basiert auf demselben Verfahren wie die Vigenère-Chiffre, enthält aber eine Modifikation, welche die Sicherheit der Chiffre verbessert. Diese Modifikation basiert auf Ideen von Gerolamo Cardano.
Die Autokey-Chiffre umgeht, dass das Schlüsselwort periodisch wiederholt wird. Wenn das Schlüsselwort kürzer ist als der Klartext, so wird einfach der Klartext an das Schlüsselwort angehängt.

Klartext: DIES IST EIN GEHEIMER TEXT
Schlüsselwort: KEY
Schlüssel: KEYDIESISTEINGEHEIMERT
Geheimtext: NMCVQ WLMAG KMUKM TIZFI OM

Die Sicherheit der Autokey-Chiffre ist höher einzustufen als die der Vigenère-Chiffre, denn eine Mustersuche mit dem Kasiski-Friedman-Test führt bei der Autokey-Chiffre zu keinem Ergebnis.
Auf der anderen Seite ist die Chiffre sehr unsicher, wenn dem Angreifer Teile des Klartextes bekannt sind, da der Klartext ebenfalls im Schlüssel vorkommt.
Außerdem lassen sich Buchstaben durch analytische Methoden auffinden. Da der Klartext Teil des Schlüssels ist, kann man davon ausgehen, dass natürliche Sprache im Schlüssel verwendet wird. Wird jetzt des Weiteren von einem deutschen Text ausgegangen, so wäre sowohl im Schlüssel als auch im Klartext das E der am häufigsten vorkommende Buchstabe. Dadurch ergibt sich die Kombination von Klartext E und Schlüssel E mit dem daraus resultierenden Geheimtextbuchstaben I, am häufigsten. Nach diesem Verfahren lässt sich auch auf andere Buchstaben im Klartext schlussfolgern.
Quelle: http://www.cryptool-online.org

Playfair-Kodierung

Die Playfair-Verschlüsselung ist ein klassisches Verschlüsselungsverfahren, bei dem jedes Buchstabenpaar des Klartextes durch ein anderes Buchstabenpaar ersetzt wird.
Dieses Verfahren wurde 1854 von Sir Charles Wheatstone erfunden. Sein Bekannter, Lord Lyon Playfair (Abbildung), unter dessen Namen sie berühmt

wurde, empfahl diese Methode zur Benutzung beim britischen Militär. Das Verfahren wurde mehrfach zu Kriegszwecken eingesetzt.
Zum Zeitpunkt ihrer Erfindung war die Playfair-Kodierung ein sehr sicheres Verfahren. Ab 1915 gelang es das Verfahren zu knacken.

Aus einem Schlüsselwort oder -satz wird ein vertauschtes Alphabet mit 36 Zeichen (25 Buchstaben, 10 Ziffern ein Leerzeichen) erzeugt.
Das Schlüsselwort wird dazu zeilenweise in eine 6×6-Matrix eingetragen, wobei bereits eingetragene Buchstaben übersprungen werden. Danach werden die noch fehlenden Zeichen in alphabetischer Reihenfolge ergänzt. Die entstehende Anordnung ist das Playfair-Quadrat.

Zur Verschlüsselung werden immer zwei nebeneinanderliegende Buchstaben durch Buchstaben des Playfair-Quadrates ersetzt.
Stehen beide Originalzeichen in der gleichen Spalte oder in der gleichen Zeile, werden jeweils die unteren beziehungsweise rechten Nachbarbuchstaben als Geheimbuchstaben genommen.
Stehen die beiden Buchstaben des Klartext-Bigramms hingegen in unterschiedlichen Zeilen und Spalten, so ersetzt man den ersten Klarbuchstaben durch den in derselben Zeile aber in der Spalte des zweiten liegenden. Der zweite Klarbuchstabe wird durch den in derselben Zeile aber in der Spalte des ersten Klarbuchstabens ersetzt. Das Klartextpaar bildet also die diagonal gegenüber liegenden Ecken eines Rechtecks. Das Geheimtextpaar wird aus den übrigen beiden Ecken dieses Rechtecks erzeugt.

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	--··	ä	·-·-
ö	---·	ü	··--	0	-----
1	·----	2	··---	3	···--
4	····-	5	·····	6	-····
7	--···	8	---··	9	----·

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

Pollux-Kodierung

Die Pollux-Kodierung basiert auf dem Morse-Kode.
In der ersten Verschlüsselungsrunde wird der Klartext mittels Morse-Alphabet (obere Tabelle) in eine Folge von Punkten, Strichen und Leerzeichen verwandelt.
Anschließend wird jeder Punkt, jeder Strich und jedes Leerzeichen durch eine Ziffer oder einen Buchstaben ersetzt.
Dabei wird zufällig eines der Zeichen aus der unteren Tabelle ausgewählt.
Zum Beispiel kann ein Punkt durch die Ziffern 2, 3, 8 oder 9 bzw. durch die Buchstaben A, F, G, M, N, Q, S, W, Y kodiert werden. Im Ergebnis entsteht ein polyalphabetisch verschlüsselter Geheimtext.

Zum Beispiel wird aus dem Klartext "Pollux-Kodierung" der Geheimtext
3149577759193081990927649316189345434647754895230969486821072547 95
oder
8743641462423627390894073816483940484647161990230869420987572071 96

Wird die Zuordnung der Ziffern und Buchstaben für jede Verschlüsselung verändert, so ergibt sich ein relativ schwer "knackbares" Kodierungsverfahren. Eine weitere Erhöhung der Sicherheit ergibt sich durch eine zusätzliche Transposition der Zeichen des Geheimtextes.
Hauptnachteil des Verfahrens ist, dass der Geheimtext im Durchschnitt die drei- bis vierfache Länge besitzt.

Jefferson-Walze

Die Jefferson-Walze (engl. wheel cipher) war ein Hilfsmittel zur Verschlüsselung von Nachrichten. Die Walze wurde 1790 von Thomas Jefferson entwickelt.

Das Chiffrenrad war ein 5 cm dicker und 14 cm langer Zylinder aus Holz. Dieser Zylinder bestand aus 36 nummerierten Scheiben, deren Randflächen in 26 Abschnitte für die 26 Buchstaben aufgeteilt sind. Zur Chiffrierung wurde eine spezielle Reihenfolge der Scheiben eingestellt. Der Text wurde in einer Zeile eingestellt. Die kodierte Nachricht wurde dann aus einer der anderen 25 Zeilen abgelesen.
Der Empfänger wählte die gleiche Scheibenreihenfolge und in einer Zeile die kodierte Nachricht. Unter den anderen 25 Zeilen befindet sich dann auch die Klarbotschaft.
Bei den ursprünglich 36 Scheiben existierten $36! = 4 \cdot 10^{41}$ Möglichkeiten, diese anzuordnen.
Die Jefferson-Walze wurde zu seiner Lebenszeit nicht genutzt. Erst im 2.Weltkrieg verwendete die US-Armee eine auf 25 Scheiben reduzierte Variante.

Sorge-Kode

Während des 2.Weltkrieges übermittelte Dr.Richard Sorge (Kodename RAMSAY) geheime Nachrichten über die Kriegsvorbereitungen der Japaner an die UdSSR. Der verwendete Kode war wie folgt aufgebaut.

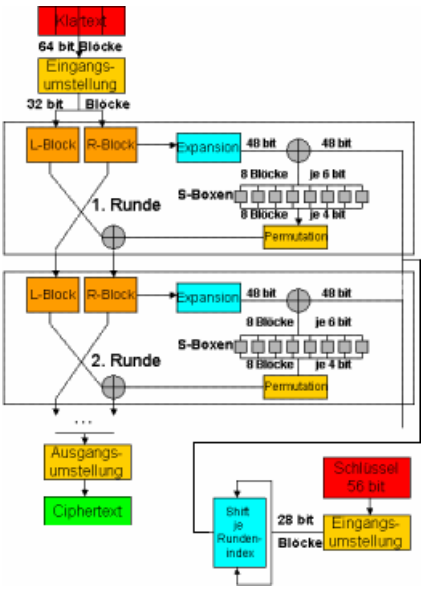
1. Substitution: Die Klartextzeichen wurden durch Zahlen ersetzt.

A	B	C	D	E	F	G	H	I	J
5	87	80	83	3	92	95	98	1	84
K	L	M	N	O	P	Q	R	S	T
88	93	96	7	2	85	89	4	0	6
U	V	W	X	Y	Z	•	/		
82	99	91	81	97	86	90	94		

Zum Beispiel ergibt sich für die Nachricht
DAL .DE R/SO WJE TISC HE/ FERN E/OS TEN/ KANN /AL S/SI CHE R/V OR/E INEM /ANG RIF F/J APA NS/E RACH TET/ WER DEN. RAM SAY •
die in Fünfergruppen aufgeteilte Zahlenkolonne
83593 90833 49402 91843 61080 98394 92347 39420 63794 88577 94593 09401 80983 49499 24943 17396 94579 54192 92948 45855 70943 45809 86369 49134 83379 04596 05979 0
Bei Überstand der 5erGruppen wurden diese weggelassen oder aufgefüllt, im Beispiel die 0 weggelassen.
Anschließend wurde eine Chiffriertabelle aus dem "Statistisches Jahrbuch des Deutschen Reiches" gewählt, z.B.

Seite 193 Zeile 7 Spalte 5 ist … (19375)

Die Zahlenkolonne wurde mit dem Schlüssel ohne Übertrag addiert:
83593 90833 49402 91843 61080 98394 92347 … 92948 45855 70943 45809 86369 49134 83379 04596 05979 35635 51303 24932 10010 78191 12106 21169 … 15112 84112 13865 86318 09150 65213 43724 43839 92727 18128 41136 63334 01853 39171 00490 13406 … 07050 29967 83708 21117 85419 04347 26093 26093 97696
Für den zu übermittelten Schlüssel wurde die Summe aus der 4.Gruppe, der drittletzten Gruppe und dem Schlüssel ohne Übertrag gebildet 01853 + 26093 + 19375 = 36111
Diese Schlüsselgruppe wurde immer an den Anfang des Spruches gesetzt.



Data Encryption Standard

Entwicklung

- Anfang der 70er Jahre von IBM aus dem Vorgängersystem Lucifer
- August 1974 beim US-amerikanischen Normenbüro eingereicht
- 1977 als US-Verschlüsselungsstandard genormt
- DEA (Data Encryption Algorithm) Norm X3.92
- 1994 mittels linearer Kryptoanalyse wird ein DES-Schlüssel von 12 HP9735-Workstations in 50 Tagen ermittelt
- 1998, 15. Juli … ein weniger als 250000 US-Dollar teurer Spezialcomputer knackt eine mit DES verschlüsselte Nachricht in 56 Stunden (brute-force)

Algorithmus

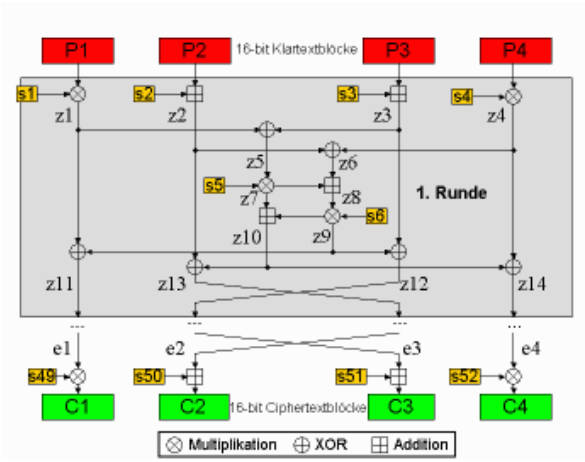
- Block-Produkt-Chiffre aus Permutationen und Substitutionen
- 16 Iterationsschritte ("Runden"); Umwandlung 64-Bit-Klartext-Block in einen 64-Bit-Chiffratblock
- Schlüssel 64 Bit lang (56 relevant, 8 Paritätsbits)

Da die Sicherheit des DES nicht mehr gegeben war - seit Ende 1998 dürfen US-amerikanische Regierungsstellen kein einfaches DES mehr verwenden -, entschied man sich zu einer anderen Einsatzform, dem Triple-DES oder 3DES. Man verschlüsselt die Information mit DES und dem Schlüssel 1, dann entschlüsselt man mit dem Schlüssel 2 und schließlich verschlüsselt man erneut mit Schlüssel 1. Die theoretische Schlüssellänge beträgt daher 3·56=168 bit.

DES-Algorithmus

Im DES-Algorithmus wird die zu verschlüsselnde Information in 64 bit Klartextblöcke unterteilt. Der erste Klartextblock wird in einer schlüsselunabhängigen Umstellung in zwei Blöcke zu je 32 bit geteilt, dem L- und R-Block.
Es folgen nun die 16 Runden. Der R-Block der ersten Runde wird zum L-Block der zweiten Runde. Der L-Block der ersten Runde wird mit einer Funktion addiert, die den R-Block der ersten Runde und den Schlüssel verwendet. Diese Funktion, das zentrale Element des DES wird auch Rundenschlüssel genannt. Das Ergebnis ist der R-Block der zweiten Runde. In der 16.Runde werden allerdings der R- und L-Block nicht mehr über Kreuz vertauscht. Es folgt eine Abschlusspermutation, welche zur ersten Permutation invers ist.
Der Rundenschlüssel, welcher bei jeder Runde erneut berechnet wird, setzt sich aus dem 32 bit R-Block und dem 48-bit Teilschlüssel zusammen.
Die 32 bit des R-Blocks werden durch Duplizierung bestimmter Bitstellen auf 48 bit expandiert.

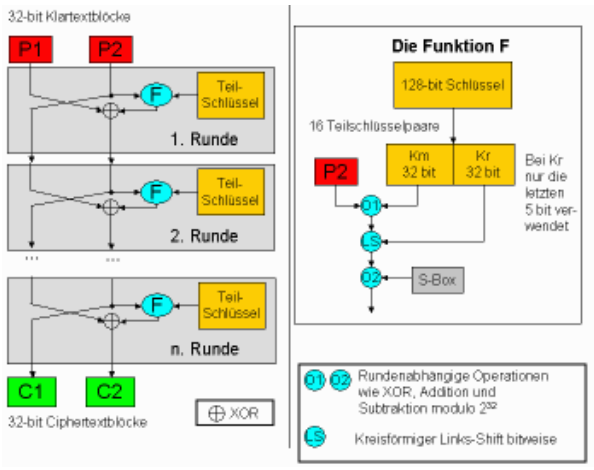
Der 56 bit Schlüssel wird nach einer festen Regel umgestellt und in zwei 28 bit Blöcke geteilt. Diese Blöcke werden je nach Runde um einen oder zwei Stellen nach links verschoben. Aus den dann vorliegenden Blöcken wird der 48 bit Teilschlüssel für die jeweilige Runde ermittelt.
Der expandierte R-Block und der Teilschlüssel werden modulo 2 addiert. Das Ergebnis wird in acht Gruppen zu je 6 bit aufgeteilt. Jeder 6 bit Block wird in eine Substitutionsbox (S-Boxen) für 4 verschiedene nichtlineare Substitutionen gegeben. Als Ergebnis erhält man in jeder der 8 Boxen eine 4 bit Ausgabe. Diese werden miteinander verbunden und abschließend nach einer festen Regel umgestellt. Damit steht der Rundenschlüssel zur Verfügung. Dieser bildet dann mit dem L-Block den R-Block der nächsten Runde.



IDEA-Algorithmus (International Data Encryption Algorithm)

Der IDEA-Algorithmus wurde 1991 von Lai und Massay zum Patent vorgelegt. Wie beim DES-Algorithmus handelt es sich um ein symmetrisches Verschlüsselungsverfahren; IDEA ist ein potentieller Nachfolger für DES.

Der Algorithmus ist insbesondere als Bestandteil des bekannten Softwarepakets PGP (Pretty Good Privacy) zur Verschlüsselung von e-mails bekannt geworden. Im Unterschied zu DES wurde nicht nur der Algorithmus veröffentlicht, sondern auch seine Entwurfsgrundlagen. Ziel war die Verwendung möglichst einfacher Operationen (Addition modulo 2, Addition modulo 2^{16}, Multiplikation modulo 2^{16+1}).
Mit IDEA kann man 64-Bit-Klartextblöcke verschlüsseln und bei Wahl der Teilschlüssel in umgekehrter Reihenfolge wieder entschlüsseln. Zur Verschlüsselung wird jeder 64-Bit-Klartextblock in vier Teilblöcke von je 16 Bit aufgeteilt. IDEA benutzt 128-Bit-Schlüssel, aus denen 52 Teilschlüssel von je 16 Bit erzeugt werden. In 8 Verschlüsselungsrunden werden jeweils 6 dieser Teilschlüssel benötigt; die restlichen 4 Teilschlüssel werden in einer Ausgabetransformation mit den vier Textblöcken verknüpft und abschließend zu einem 64-Bit-Schlüsseltextblock zusammengesetzt.
IDEA ist etwa doppelt so schnell wie DES, in Hardware jedoch schwieriger zu implementieren. Öffentlich sind keine erfolgreichen Angriffe gegen IDEA bekannt geworden. Angriffe durch Ausprobieren aller Schlüssel bleiben bei der Schlüssellänge von 128 Bit wirkungslos.



CAST-Algorithmus

Der Algorithmus CAST wurde von Carlisle Adams und Stafford Tavares Anfang der neunziger Jahre entwickelt. CAST-5 ist der Standardalgorithmus (Default-Wert) in PGP und GnuPG. CAST ist ebenfalls wie IDEA patentiert (Entrust Technologies), darf aber frei verwendet werden.
Bei CAST handelt es sich um ein DES-ähnliches Substitutions-Permutations-Netzwerk (SPN). Es werden 64-bit Textblöcke mit einem variablen Schlüssel von 40 bis 128 bit (in 8-bit-Inkrementen) verarbeitet. Ab einer Schlüssellänge von 80 bit sind 16 Runden zu durchlaufen.
Ein 64-bit Klartextblock wird in zwei 32-bit Blöcke P1 und P2 zerlegt. P1 (oder auch linker Block genannt) wird mit einem aus P2 (oder auch rechter Block genannt), den Teilschlüsseln und den S-Blöcken in einer Funktion F gebildeten Wert XOR-verknüpft, um dann selbst zum rechten Block der nächsten Runde zu werden. P2 wird unmittelbar ohne mathematische Behandlung zum linken Block der nächsten Runde.
Die angesprochene Funktion F benötigt 16 Teilschlüsselpaare zu je 32 bit (Km und Kr). Vom Teilschlüssel Kr werden lediglich die letzten 5 bit verwendet; er wird für den kreisförmigen Links-Shift benötigt). In der Operation
O1 wird der rechte 32-bit Block (P2) mit dem 32-bit-Teilschlüssel Km verknüpft. Das Ergebnis wird einem kreisförmigen Links-Shift mit Kr unterzogen und anschließen in der Operation O2 mit den S-Boxen verknüpft. Die Operationen O1 und O2 sind rundenabhängig. Insgesamt werden drei verschiedene Typen unterschieden. Als mathematische Operationen werden Addition und Subtraktion modulo 2^{32} und XOR-Verknüpfungen verwendet.

Apparantly RC4

Ron's Code oder der Rivest Cipher No.4 (RC4) ist ein Verfahren zur Stromchiffrierung. Es wurde 1987 von Ronald L.Rivest für RSA Data Security Inc. entwickelt und lange Jahre geheimgehalten.
Im September 1994 veröffentlichte eine anonyme Person einen Algorithmus, der zu RC4 identische Ergebnisse erzeugte und daher "apparantly RC4" ist. Der Quelltext verbreitete sich schnell.

Im Gegensatz zu DES ist die Schlüssellänge variabel und kann bis zu 2048 Bit (256 Zeichen) betragen, wobei bereits 128 Bit als sicher gelten. Es wird immer ein Byte auf einmal verschlüsselt. Der Algorithmus ist einfach und sicher und kann effizient programmiert werden.

```
var RC4_sbox = new Array (256);
function RC4_crypt (key, text) { var i, j, k = 0; var temp = 0; var t = 0; var rtext = "";
        for (j = 0; j < 256; j++) RC4_sbox[j] = j;
        j = 0;
        for (i=0; i < 256; i++) { j = (j + RC4_sbox[i] + key.charCodeAt(i % key.length)) % 256;
    temp = RC4_sbox[i]; RC4_sbox[i] = RC4_sbox[j]; RC4_sbox[j] = temp; }
        for (k=0; k < text.length; k++) { i = (i + 1) % 256; j = (j + RC4_sbox[i]) % 256;
    temp = RC4_sbox[i]; RC4_sbox[i] = RC4_sbox[j]; RC4_sbox[j] = temp;
    t = (RC4_sbox[i] + RC4_sbox[j]) % 256;
    rtext =  retxt + String.fromCharCode(text.charCodeAt(k) ^ RC4_sbox[t]); }
return rtext; }
```

Der Algorithmus kann unverändert zur Ver- und Entschlüsselung genutzt werden kann.
Zuerst wird eine S-Box initialisiert. Das gefüllte Feld wird mit Hilfe des Schlüssels untereinander vertauscht.
Danach wird für jeden Buchstaben des Eingabetextes ein Zufallsbyte aus der S-Box ermittelt, wobei die Felder des Array erneut vertauscht werden.
Am Ende werden das Zufallsbyte und das Textzeichen XOR miteinander verknüpft.

Kryptosysteme mit öffentlichem Schlüssel

Jeder Kommunikationsteilnehmer T befindet sich im Besitz von zwei Schlüsseln :
1. einem öffentlichen Schlüssel ET zur Verschlüsselung sowie
2. einem geheimen Schlüssel DT zur Entschlüsselung.

Alle Teilnehmer in einem asymmetrischen Kommunikationssystem verwenden den selben Verschlüsselungsalgorithmus e und den selben Entschlüsselungsalgorithmus d.
Für jede Nachricht m in einem asymmetrischen Verschlüsselungssystem gilt: **d(e(m,ET),DT) = m**
Nachrichtenaustausch zwischen zwei Teilnehmern A und B in einem asymmetrischen Kryptosystem:
1. Der Benutzer A muss sich den öffentlichen Schlüssel EB des Teilnehmers B aus einer öffentlichen Datei heraussuchen.
2. Der Benutzer A verschlüsselt das Kryptogramm mit Hilfe des Verschlüsselungsalgorithmus e und des öffentliches Schlüssels EB des Teilnehmers B und sendet das Kryptogramm. **c = e(m,EB)**
über einen beliebigen offenen (also durchaus unsicheren) Kanal an B.
3. Benutzer B entschlüsselt nun das erhaltene Kryptogramm c mit dem Entschlüsselungsalgorithmus d und seinem geheimen Schlüssel DB. **m = d(c, DB)**

Anforderungen an asymmetrische Kryptosysteme

1. Bei gegebenem m und ET sollte c = e(m,ET) leicht berechenbar sein.
2. Bei gegebenem Kryptogramm c sollte es rechnerisch nicht möglich sein, die Nachricht m zu erschließen, dies bedeutet konkret, dass aus Kenntnis des öffentlichen Schlüssels ET der geheime Schlüssel DT nicht zu erschließen sein darf
3. Wenn das Kryptogramm c und der geheime Schlüssel DT bekannt sind, sollte die Nachricht m leicht wieder zu bestimmen sein.

Rivest-Shamir-Adlemann-System

Mit dem 1978 veröffentlichten und am 20. September 1983 als US Patent 4405829 eingetragenen RSA-Algorithmus lieferten die Mathematiker Ronald Rivest, Adi Shamir und Leonard Adlemann die wohl wichtigste und bekannteste Implementierung eines asymmetrischen Kryptosystems.
Seine Sicherheit hängt im Wesentlichen von der Vermutung ab, dass es (im Moment) keine schnelle Möglichkeit gibt, Zahlen zu Faktorisieren, die das Produkt zweier hinreichend großer Primzahlen sind.
1. Wahl von zwei großen Primzahlen p und q mit den Forderungen
    a) p-1, p+1, q-1 und q+1 enthalten keine kleinen Primteiler
    b) |p-q| darf nicht klein sein
    c) p und q sollten mehr als 42 Dezimalstellen besitzen
    d) p*q sollte deutlich mehr als 129 Dezimalstellen besitzen
2. öffentlicher Parameter n=p*q
3. Berechnung $\phi(n) = (p-1)*(q-1)$
4. Wahl eines öffentlichen Parameters e mit $ggT(e,\phi(n)) = 1$

5. Berechnung des geheimen Parameters d mit e * d = 1 (mod $\phi(n)$)
6. Ermittlung von k mit $e*d = 1 + k\phi(n)$
Anmerkung: Das RSA-Verfahren funktioniert auch, wenn p und q keine Primzahlen, sondern zueinander teilerfremde Carmichael-Zahlen sind.

Gültigkeit des RSA-Verfahrens: Für alle $x \in Z_n$ gilt $x^{ed} \equiv x \pmod n$.
Nachweis:
Für x = 0 ist die Aussage trivial. Sei nun $x \neq 0$.
Fall 1: x ist weder durch p noch durch q teilbar. Wegen $ed + 1 + z\,\phi(n)$
für ein geeignetes $z \in Z$ gilt: $x^{ed} = x^{1+z\phi(n)} = x(x^{\phi(n)})^z$
Der Satz von Euler besagt, $x^{\phi(n)} \equiv 1$ dass (mod n) ist, und es folgt:
$x^{ed} = x(x^{\phi(n)})^z \equiv x\ 1^z = x \pmod n$.
Fall 2: x ist durch p, aber nicht durch q teilbar. Wegen $\phi(n) = \phi(p)\phi(q)$ wird
$x^{ed} = x^{1+z\phi(n)} = x(x^{\phi(q)})^{\phi(p)z}$
und $x^{ed} = x(x^{\phi(q)})^{\phi(p)z} \equiv x\ 1^{\phi(p)z} = x \pmod q$.
Daraus folgt, dass q ein Teiler von x^{ed}-x ist. Weiterhin ist auch p ein Teiler von x^{ed}-x, da p ein Teiler von x. Also ist auch n = pq ein Teiler von x^{ed}-x und daraus folgt $x^{ed} \equiv x \pmod n$.
Fall 3: x ist durch q, aber nicht durch p teilbar: wie Fall 2.
Fall 4: x ist durch p und durch q teilbar: nicht möglich, denn $x \in Z_n$, also x < n = pq.

Nachrichtenverschlüsselung unter RSA

Nachricht m wird in $c = m^e \bmod n$ verschlüsselt und c abgesendet

Empfänger

Berechnung von $m' = c^d \bmod n$
m' ist dann Originalnachricht

Die Sicherheit des Verfahrens beruht darauf, dass es praktisch unmöglich ist, große Zahlen in Primfaktoren zu zerlegen:

Zahl	Rechenzeit	Zahl	Rechenzeit
154stellig	1 Jahr	308stellig	10^6 Jahre
462stellig	10^{11} Jahre	616stellig	10^{15} Jahre

Die Zeiten beziehen sich auf einen Superrechner, den INTEL PARAGON XP/S-5. Dieser Rechner besitzt eine Leistungsfähigkeit von 5,4 GFLOPS (5,4 Giga-FLOPS = 5,4 Milliarden floating point operations per second).

Exponentieller Schlüsseltausch von Diffie und Hellman (1976)

Öffentlich große Primzahl p und Hilfszahl s < p
Teilnehmer A geheime Zahl a<p Teilnehmer B geheime Zahl b<p
A berechnet $\alpha = s^a \bmod p$ B berechnet $\beta = s^b \bmod p$
α und β werden öffentlich ausgetauscht

Berechnung des Schlüssels k für symmetrisches Codierungsverfahren, wie z.B. DES

A: $k = \beta^a \bmod p$ B: $k = \alpha^b \bmod p$

RSA-Zahl

Als RSA-Zahlen werden große, aus zwei Primfaktoren bestehende, natürliche Zahlen bezeichnet, die von der Firm RSA Data Security Inc. zum Test der Sicherheit des RSA-Verfahrens veröffentlicht wurden. 1977 wurde eine 129stellige RSA-Zahl veröffentlicht, von der Rivest, Shamir und Adleman überzeigt waren, dass das Faktorisieren und damit das Brechen dieses RSA-Codes: "would take millions of years to break".

$$114381625757888867669235779976146612010218296\cdot$$
$$\cdots 7212423625625618429357069352457338978305971$$
$$\cdots 2356395870505898907514759929002687954354 1$$
$$= 34905295108476509491478496199038981334177764$$
$$\cdots 6384933878439908205773276913299326\cdots$$
$$\cdots 6709549961988190834461413177642967992\cdots$$
$$\cdots 942539798288533$$

1994 wurde die Zahl mit Hilfe des Verfahrens mehrfacher quadratischer Siebe mit extremen Rechenaufwand faktorisiert. Durch Leylad, Atkins und Graff wurde die links angegebene Zerlegung ermittelt.
Der gegenwärtige (2002) Weltrekord im Faktorisieren besteht in der Zerlegung einer RSA-155 Zahl in ihre zwei 78stelligen Primfaktoren. Nach monatelanger Vorarbeit auf rund 300 PCs und Workstations erreichten am 22.8.1999 te Riele und Peterson nach abschließenden 224 Stunden Rechnung auf einem Cray C916 das Rekordergebnis.

InversMod-Funktion

Zur Bestimmung des geheimen Parameters d mit e * d = 1 (mod m) des RSA-Verfahrens wird über den erweiterten Euklidischen Algorithmus eine InversMod-Funktion realisiert.

Iterativer Algorithmus

```
function invers_mod(e,m:integer):integer;
var m0,x0,x1,y0,y1,xx,yy,q,r,sign:integer;
begin
  m0:=m; x0:=1; x1:=0; y0:=0; y1:=1; sign:=1;
  while e <> 0 do Begin q:=m div e;   r:=m mod e;   m:=e;   e:=r;   xx:=x1;   yy:=y1;
```

```
   x1:=q*x1+x0;   y1:=q*y1+y0;   x0:=xx;    y0:=yy;    sign:=-sign;  End;
  y0:=-sign*y0; if y0 < m then y0:=m0+y0; result:=y0;
end;
```

Rekursiver Algorithmus

```
function invers_mod0(e,m: integer): integer;
var v: integer;
procedure ggTerw(a,b: integer; var u,v: integer);
var u0, v0:integer;
begin if b = 0 then Begin u := 1; v := 0; End  else Begin ggTerw(b, a mod b, u0, v0); u := v0; v := u0 -
(a div b)*v0; End;
end;
begin  ggTerw(e,m,result,v); if result < 0 then result := result + m; end;
```

ElGamal-Verfahren

Das ElGamal-Verfahren ist ein asymmetrisches Verschlüsselungsverfahren, ähnlich RSA. Es wurde 1985 von Taher ElGamal veröffentlicht.
Die Sicherheit beruht auf der Schwierigkeit, diskrete Logarithmen zu bestimmen.
Ausgang ist eine große Primzahl Z, für die (Z-1)/2 ebenfalls prim ist. Zwei weitere Zahlen y und g, die größer als Eins, aber kleiner als die Zahl Z sind, werden gewählt.
Der öffentliche Schlüssel p wird berechnet aus $p = y^g \bmod Z$.
p, y und Z sind öffentlich und werden auch im öffentlichen Schlüssel verwendet. g dagegen ist geheim und darf nie veröffentlicht werden.

Verschlüsselung

Die Klartextnachricht wird in eine Zahl transformiert. Ist sie größer als Z, so muss sie in Teile zerlegt werden.
Der Absender benötigt noch zufällige Zahl x, die größer Eins, aber kleiner Z-1 sein muss und keinen gemeinsamen Teiler mit Z haben soll.
Hilfsgrösse c_1: $c_1 = y^x \bmod Z$ Chiffrierung $c_2 = (k\ p^x) \bmod Z$ mit k= Klartextteil
Außer der chiffrierten Nachricht c_2 wird der Wert c_1 übermittelt.

Entschlüsselung

Der Empfänger weiss nichts von der Hilfszahl x, benötigt diese aber für die Rückrechnung auch nicht.
Für die Entschlüsselung gilt $k= (c_2\ c_1^{(Z-1-g)}) \bmod Z$

Nachweis des ElGamal-Verfahrens

Schlüsselvergabe $c = g^d \bmod p$ kurz: $c \equiv g^d\ (p)$
Verschlüsselung $h = g^a \bmod p$ kurz: $h \equiv g^a\ (p)$
$y = (x \cdot c^a) \bmod p$ kurz: $y \equiv x \cdot c^a\ (p)$
Nachweis $y \cdot h^{p-1-d} \equiv x \cdot c^a \cdot h^{p-1-d} \equiv x \cdot (g^d)^a \cdot (g^a)^{p-1-d} \equiv x \cdot g^{da} \cdot g^{ap-a-ad} \equiv$
$\equiv x \cdot g^{ad+ap-a-ad} \equiv x \cdot g^{ap-a} \equiv x \cdot g^{a(p-1)} \equiv x \cdot (g^{p-1})^a \equiv$
$\equiv x \cdot 1^a$ nach dem kleinen Satz von Fermat: $g^{p-1} \equiv 1\ (p)$
$\equiv x$

Pretty Good Privacy, PGP

Pretty Good Privacy (PGP) ist ein von Phil Zimmermann entwickeltes Programm zur Verschlüsselung und zum Unterschreiben von Daten.
PGP benutzt ein sogenanntes Public-Key-Verfahren.
Genutzt wird ein öffentlicher Schlüssel, mit dem jeder die Daten für den Empfänger verschlüsseln kann, und ein privater geheimer Schlüssel, den nur der Empfänger besitzt und der durch ein Kennwort geschützt ist. Nachrichten an einen Empfänger werden mit seinem öffentlichen Schlüssel verschlüsselt und können dann ausschließlich durch den privaten Schlüssel des Empfängers entschlüsselt werden.

Die erste Version wurde 1991 geschrieben und verwendete einen RSA-Algorithmus zur Verschlüsselung der Daten. Spätere Versionen benutzten den Diffie-Hellman-Schlüsselaustausch (DH/DSS).
Bei PGP wird nicht die ganze Nachricht asymmetrisch verschlüsselt. Stattdessen wird die eigentliche Nachricht symmetrisch und nur der verwendete Schlüssel asymmetrisch verschlüsselt (Hybride Verschlüsselung). Dazu wird jedes Mal ein symmetrischer Schlüssel zufällig erzeugt. Die ersten PGP-Versionen verwendeten die IDEA-Verschlüsselung mit 128 Bit Schlüssellänge.
Dieser symmetrische Schlüssel wird dann per RSA- oder Elgamal-Kryptosystem mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und der Nachricht hinzugefügt.
Da PGP, ursprünglich Freeware, mittlerweile vermarktet wird, nutzen viele Anwender das kostenlose GNU Privacy Guard.
siehe dazu http://www.gnupg.org/index.de.html

Advanced Encryption Standard (AES)

Im Herbst 1997 startete das NIST (National Institute of Standards and Technology) einen weltweiten Wettbewerb zur Entwicklung eines neuen Standards zur symmetrischen Blockchiffrierung von Daten, den Advanced Encryption Standard (AES).
Aus 15 Kandidaten wurde fünf Finalisten ausgewählt. Nach Prüfung wurde im Oktober 2000 Rijndael als neuer Standard ausgewählt. Geplant ist eine Lebensdauer von 20 bis 30 Jahren für den AES. Wahrscheinlich wird die Lebensdauer weit höher sein.

Elliptic Curve Cryptography (ECC)

Elliptische Kurven werden in der Mathematik untersucht und seit kurzem verstärkt angewendet. Elliptische Kurven werden zur Faktorisierung, bei Primzahlnachweisen und in der Public-Key-Kryptographie eingesetzt.
Als Verfahren der Verschlüsselung sind sie geeignet, die Standards DES (Data Encryption Standard) und RSA (Rivest, Shamir, Adleman) abzulösen, weil es wesentlich kürzere Schlüssel benötigt.
Ein 160 Bit langer ECC-Schlüssel bietet beispielsweise die gleiche Sicherheit wie ein 1024 Bit langer RSA-Schlüssel.

Rijndael

Im Oktober 2000 wurde das Verschlüsselungsverfahren Rijndael der belgischen Kryptologen Joan Daemen und Vincent Rijmen zum Advanced Encryption Standard (AES) erklärt und soll Nachfolger des DES werden.
Rijndael ist eine symmetrische Blockchiffre mit variablen Block- und Schlüssellängen von 128 bis 256 Bit. Die Verschlüsselung wird in mehreren Runden durchgeführt, wobei deren Anzahl n von der Schlüssel- und Blocklänge abhängig ist.

Anzahl Runden n	Blocklänge in Bit		
Schlüssellänge in Bit	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

Für jedes Verschlüsselungsrunde wird ein Rundenschlüssel erzeugt. Dazu werden an den Anwenderschlüssel 4-Byte-Wörter angehängt.
Zuerst wird der Klartextblock mit dem Anwenderschlüssel durch XOR (bitweise Addition) verknüpft.
Danach werden n-1 Runden durchgeführt:
1) Substitution: Jedes Byte eines Blockes wird substituiert
2) Permutation: Die Bytes eines Blockes werden in einer vierzeiligen Matrix eingetragen und reihenweise zyklisch verschoben.
3) Diffusion: Die Bytes der permutierten Matrix des Blockes werden nochmals spaltenweise permutiert.
4) Schlüsselverknüpfung: Der Block wird durch XOR mit dem Rundenschlüssel verknüpft.
Die abschließende Runde wird ohne 3) Diffusion durchgeführt.

Bisher widerstand Rijndael allen kryptografischen Attacken.
Benötigt ein Computer zum Knacken von DES eine Sekunde, so wären für Rijndael mit 128-Bit-Schlüssel rund 150 Billionen Jahre notwendig.

Blowfish

Das Verschlüsselungsverfahren Blowfish wurde von Bruce Schneier entwickelt und 1994 veröffentlicht. Es handelt sich um eine Blockchiffre, die auf Feistel-Netzwerken basiert. Die Blockgröße beträgt 64 bit, die Schlüssellänge ist variabel und kann von 32 bis zu 448 bit betragen.
Die Umsetzung kann sehr effizient vorgenommen werden, da nur XOR-Operationen und Additionen von 32-Bit-Worten verwendet werden. Auf 32-Bit-Prozessoren ist Blowfish wesentlich schneller ist als z.B. DES.

Blowfish verwendet 18 Rundenschlüssel P mit je 32 Bit, und vier S-Boxen mit je 256 = 28 Einträgen von je 32 Bit. Die Initialisierung der P-Werte und der vier S-Boxen erfolgt mit einer fixen Zahlenfolge, die aus der hexadezimalen Ziffernfolge der Kreiszahl π abgeleitet wird. Die Nachkommastellen von π sind pseudozufällig und unabhängig vom restlichen Algorithmus, damit sind die Anforderungen an eine nutzbare Konstante erfüllt.
Der Algorithmus gilt als sicher, da bisher keine Methode gefunden wurde, mit der die Verschlüsselung effektiv geknackt werden konnte.

Steganografie

Steganografie ist ein Verfahren zur Aufnahme von Textmitteilungen (oder anderen Daten) in ein Bild. Dabei wird als Vorlage ein Foto genutzt, dessen Bildinformation mit dem zu kodierenden Text verändert wird. Die Veränderung ist dabei so gering, dass es dem menschlichen Auge schwer fällt, die Unterschiede

zum Ausgangsbild zu erkennen. Zur Dekodierung werden Originalbild und verändertes Bild wieder verglichen.
Schon früher wurden steganographische Verfahren verwendet um Nachrichten in unverfänglicheren Informationen zu verstecken. So kann man z.B. mit Zitronensäure einen Brief unsichtbar zwischen die Zeilen eines ganz anderen Briefes schreiben und die Schrift später mit Wärme wieder sichtbar machen. Oder man kann bestimmte Worte des unverfänglichen Briefes mit Nadelstichen markieren, die dann natürlich nur gegen Licht zu entdecken sind. Eine andere Methode ist in der "Geschichte zweier Städte" von Charles Dickens beschrieben. Die Frauen in der Geschichte stricken Informationen in linke und rechte Maschen ein - anstatt der Werte 0 und 1 als Bitwerte.
An die Steganografie werden folgenden Forderungen gestellt:
1. Keine offensichtlichen Veränderungen der Trägerdaten, d.h. die versteckten Daten sollten nur minimal wahrnehmbar sein.
2. Die versteckten Daten sollten allen Veränderungen wiederstehen, egal ob durch Rauschen, Filtern, Verschlüsselung, unscharfe, d.h. verlustbehaftete, Kompression, Verkleinerung des Trägermaterials, Drucken, oder durch Umwandlung in andere Dateiformate.
3. Das Verschlüsseln der Daten mit asymmetrischen Verfahren (RSA, PGP) vor dem Verstecken ist erwünscht. Das versteckt die Daten besser, da nach dem Verschlüsseln ein "weißes Rauschen" übrig bleibt, macht sie aber deshalb nicht schwerer zugänglich.
4. Fehlerkorrektur sollte die Korrektheit der Daten sicherstellen.
5. Die versteckten Daten sollten mit so viel Redundanz gespeichert werden, dass man mit Hilfe von Fehlerkorrektur alle Daten wieder herstellen kann, wenn ein Bit verloren geht.
Eine der ersten einfachen Anwendungen der Steganografie war Bacon's Chiffre, bei der mittels Dualsystem ein Text in einem anderen versteckt wurde.

Steganografie, Quelltext

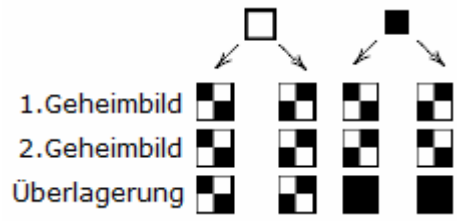
Delphi-Quelltext zur Ver- und Entschlüsselung mittels Steganografie. In dem String s befindet sich der Text, Bitmap ist das Bild, BitmapX beim Dekodieren das unveränderte Bild.

Kodieren

```
with Bitmap do begin
    for i := 1 to Length(s) do begin  Maske := 80hex;
      for j := 1 to 8 do begin CharData := Byte(s[i]) and Maske;
        if (CharData <> 0) then begin PixelData := Canvas.Pixels[x, y] xor 1;Canvas.Pixels[x, y] :=
PixelData;
       end;
       x := (x + 1) mod Bitmap.Width; if (x = 0) then Inc(y); Maske := Maske shr 1;  end; end; end;
```

Dekodieren

```
s:=''; maske := 80hex; ch := 0;
  for y := 0 to Bitmap.Height -1 do begin for x := 0 to Bitmap.Width -1 do begin
     if (Bitmap.Canvas.Pixels[x, y]<> BitmapX.Canvas.Pixels[x, y]) then
        ch := ch or maske; maske := maske shr 1;
       if maske = 0 Then begin s := s + char(ch); maske := 80hex; ch := 0;  end; end; end;
```

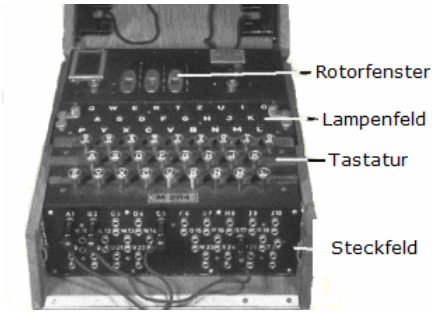


Visuelle Kryptografie

Dieses kryptografische Verfahren wurde erstmals 1994 von Adi Shamir und Moni Naor beschrieben.
Die zu verschlüsselnde Nachricht liegt in Form eines Schwarz-Weiß-Bildes vor. Für jedes Pixel der Nachricht werden je vier Pixel in zwei Teilbilder erzeugt, gemäß dem Schema in der Abbildung.

Es entstehen zwei Bilder, deren vertikale und horizontale Auflösung doppelt so hoch ist wie die der Nachricht und deren Inhalt aus zufällig gesetzten schwarzen und weißen Pixeln besteht.

Die Entzifferung erfolgt durch Überlagerung der Teilbilder. Praktisch geschieht dies, indem man das eine Teilbild auf eine transparente Folie und das andere auf ein Blatt Papier druckt. Ein schwarzes Pixel der Nachricht wird in der Überlagerung durch vier schwarze Pixel repräsentiert. Ein weißes Pixel hingegen durch zwei weiße und zwei schwarze. Der Kontrast der rekonstruierten Nachricht verringert sich um 50% im Vergleich zum Original. Dennoch ist der Inhalt erkennbar.
Da die Entschlüsselung allein durch visuelle Wahrnehmung geschieht, spricht man von visueller Kryptografie.
Das Verfahren ist nicht knackbar. Für jedes Pixel der Nachricht wird zufällig eine von zwei Kodierungen gewählt, so dass der sich Inhalt der Teilbilder nicht von einem zufälligen Rauschen unterscheidet. Ein Teilbild allein lässt keinen Rückschluss auf die ursprüngliche Nachricht zu. Auch dann nicht, wenn Vermutungen über den Inhalt der Nachricht oder große Rechenkapazitäten vorhanden sind. Damit bietet das Verfahren die kryptografische Sicherheit eines One-Time-Pads.



Enigma

Die Enigma (griechisch αινιγμα = Rätsel, Geheimnis) ist die Chiffriermaschine, die die faschistische, deutsche Wehrmacht während des Zweiten Weltkrieges zur Verschlüsselung von Funksprüchen verwendete.
Die Enigma wurde 1919 durch den Niederländer Hugo A.Koch erfunden. Der Erfolg der Maschine begann, nachdem die deutsche Kriegsmarine 1926 nach genauen Prüfungen eine große Anzahl der Maschinen kaufte.

Die Enigma sah fast wie eine Schreibmaschine aus. Sie hatte auch die Größe einer Schreibmaschine und wog 20 bis 30 Kilogramm. Sie besteht aus 4 Hauptteilen:

Die Tastatur ist eine Schreibmaschinentastatur. Das Lampenfeld sieht fast so aus wie die Tastatur. Alle Buchstaben des Alphabets sind darauf angeordnet und neben jedem Buchstaben ist eine kleine Lampe. Wenn ein Buchstabe zum Verschlüsseln in die Enigma eingegeben wird, so leuchtet der verschlüsselte Buchstabe im Lampenfeld auf.
Die Enigma gehört zu den Rotormaschinen. Die Rotoren sind elektrisch isolierte Scheiben, die an jeder Seite eine bestimmte Anzahl von Schleifkontakten haben. Jeder Kontakt auf der einen Seite der Scheibe war mit einem Kontakt auf der anderen Seite der Scheibe verbunden.
Das Steckfeld wurde benutzt, um vor und nach der Benutzung der Enigma Buchstabenpaare zu vertauschen. Dafür wurden zwei Buchstaben auf dem Steckfeld miteinander verbunden. Die nicht gesteckten Buchstaben werden ganz normal durch die Rotoren verändert.

Enigma-Gleichung

Die Enigma-Gleichung beschreibt die bei der Verschlüsselung mithilfe der Rotor-Schlüsselmaschine Enigma auftretende Buchstabenvertauschung (Permutation):

$$c_i = (T \bullet S_z \bullet U \bullet S_z^{-1} \bullet T^{-1})\ (p_i)$$

Diese Gleichung gibt den Zusammenhang an zwischen
- einem Klartextbuchstaben p_i und
- dem dazugehörigen Geheimtextbuchstaben c_i an

sowie den die Permutation und damit die Verschlüsselung bewirkenden kryptografischen Elementen der Maschine:
- dem "Steckerbrett", das eine feste involutorische Substitution T verursacht,
- dem Walzensatz, der aus zumeist drei rotierenden Walzen besteht, die eine polyalphabetische Substitution S_z hervorrufen, sowie
- der feststehenden Umkehrwalze (UKW), die eine weitere involutorische Permutation U erzeugt.

Da der die Maschine durchfließende elektrische Strom nach Passieren der Umkehrwalze den Walzensatz in umgekehrter Richtung durchläuft, und abschließend noch einmal durch das Steckerbrett fließt, treten die beiden Terme S_z^{-1} und T^{-1} am Ende der Gleichung in invertierter Form erneut auf. Die Gleichung gilt für den Verschlüsselungs- und Entschlüsselungsvorgang.
Die Gleichung wurde vom 27jährigen polnischen Kryptoanalytiker Marian Rejewski bei seiner Arbeit in der polnischen Dechiffrierstelle im Jahre 1932 aufgestellt und stellte die Grundlage für die Ermittlung der von den faschistischen, deutschen Militärs streng geheim gehaltenen Walzenverdrahtung der drei Walzen I bis III sowie der Umkehrwalze A der Enigma dar. Damit legte Rejewski die entscheidenden Voraussetzungen für die Entzifferung des geheimen militärischen Nachrichtenverkehrs der faschistischen Wehrmacht im Zweiten Weltkrieg.
Abbildung: der polnische Mathematiker Marian Rejewski (1932)

Geschichte der Kryptografie bis 1620

Jahr	Ereignis
600 v.u.Z.	in Palästina werden Texte mit der Atbasch-Kodierung verschlüsselt.
500 v.u.Z.	die Griechen verschlüsseln Nachrichten mit Hilfe der Skytale.
200 v.u.Z.	der griechische Mathematiker Polybious beschreibt erstmals sein System.
44 v.u.Z.	Julius Cäsar schrieb vertrauliche Botschaften in dem nach ihm benannten Cäsar-Code.
ab 500	in Europa beginnt die "Dunkle Zeit der Kryptografie". Kryptografie wurde der schwarzen Magie zugeordnet. Im Gegensatz dazu blüht die Kryptographie im persischen Raum auf.
855	im arabischen Raum erscheint das erste Buch über Kryptologie. Abu 'Abd al-Raham al-Khahil ibn Ahmad ibn'Amr ibn Tammam al Farahidi al-Zadi al Yahamadi beschreibt die geglückte Entschlüsselung eines für den byzantinischen Kaiser bestimmten griechischen Codes.
1397	auf Wusch Clemens VII. erfindet Gabrieli di Lavinde den Nomenklatur-Code. Dieser Nomenklatur-Code wurde wegen seiner Einfachheit in den nächsten 450 Jahren vor allem in diplomatischen Kreisen verwendet.
1412	eine 14-bändige arabische Enzyklopädie beschreibt kryptografische Methoden. Dabei wird neben der Substitution und der Transposition, erstmals die Methode der mehrmaligen

	Substitution an einem Klartextzeichen erwähnt.
1466	Leon Battista Alberti (1404-1472) veröffentlicht das Buch "Modus scribendi in ziferas", indem erstmals die von ihm erfundenen Chiffrierscheiben erwähnt. Albertis zahlreiche kryptologischen Leistungen beruhen auf der Tatsache, das er Sekretär jener Behörde war, die sich am päpstlichen Hof mit Geheimschriften befasste. Er wird als "Vater der Kryptografie" bezeichnet.
1518	im deutschsprachigen Raum erscheint das erste gedruckte Buch über Kryptologie von Johannes Trithemius.
1586	"Tractié de Chiffre" des französischen Diplomaten Blaise de Vigenère erscheint. Seine Verschlüsselungsmethode, die später nach ihm als Vigenère-Code benannt wurde, wird der Öffentlichkeit zugänglich gemacht. Dieser Code ist der bekannteste unter allen polyalphabetischen Algorithmen.
1628	Antione Rissignol wird der erste bezahlte Kryptoanalytiker, nachdem seine Entschlüsselung einer feindlichen Botschaft die Belagerung Realmonts durch die Hugenotten beendete. Seitdem werden Kryptoanalytiker für Kriegszwecke missbraucht.
1700	in Russland wird eine Code-Tabelle von 2000-3000 Silben und Worten zur Chiffrierung von Botschaften verwendet.
1795	Thomas Jefferson entwickelt den ersten Chiffrierzylinder "wheel cypher", die Jefferson-Walze.
1854	der Engländer Charles Babbage erfindet erneut einen Chiffrierzylinder.
	der englische Physiker Charles Wheatstone erfand einen Chiffre, der mit einer 5x5 Matrix arbeitet. Lord Lyon Playfair Baron von St. Anrews macht diesen Code in England bekannt. Der Code erhält den Namen Playfair-Code.
1860	Friedrich Kasiski und William F.Friedmann entwickeln statistische Methoden zur Kryptoanalyse.
1863	Kasiski (1805-1881) veröffentlicht sein kryptologisches Werk "Die Geheimschriften und die Dechiffrierkunst", in dem er als erster ein Verfahren zur Lösung von polyalphabetischen Chiffren vorschlug.
1883	"La Cryptographie militaire" von Auguste Kerkhoff von Nieuwendhoff erscheint. Das Werk gilt als Meilenstein in der Kryptographie der Telegraphenzeit und enthält die "Prinzipien von Kerkhoff" für die strategische Kryptologie.
1891	der französische Major Etienne Bazeries erfindet den Bazeries-Zylinder.
1917	Gilbert S.Vernam entdeckt und entwickelt das "One-Time-Pad".
1921	in den USA wird die erste Chiffriermaschine nach dem Rotor-Prinzip gebaut.
1922	Jeffersons Walze wird von der US-amerikanischen Marine weiterentwickelt und im 2.Weltkrieg genutzt.
1923	der deutsche Ingenieur Arthur Scherbius entwickelt die Rotormaschine "Enigma".
1925	William F. Friedman entwickelt einen Test zur Abschätzung der Schlüssellänge von Vigenère-Chiffren. Zusammen mit dem Kasiski-Test gilt der Friedmann-Test als einer der Klassiker der Kryptoanalyse.
1926	die deutsche Reichsmarine führt den Funkschlüssel C ein, eine Codierung der Nachrichten mit einer Enigma vom Typ C
ab 1939	die faschistische, deutsche Wehrmacht nutzt die Enigma
	England übernimmt die bereits 1938 von polnischen Kryptologen begonnene Versuche, die Enigma, mit Hilfe sogenannter "Bomben", zu entschlüsseln.
1941	die japanische Angriffsmeldung für den 2.Weltkrieg wird decodiert, d.h. der Angriff auf Pearl Harbor war den USA vorher bekannt
ab 1950	weltweit werden eigene Chiffriermaschinen entwickelt
1975	Diffie und Hellmann zeigen, dass Public-Key-Verfahren theoretisch möglich sind.
1977	das von IBM entwickelte DES (Data Encryption Standard) wird in den USA zum Standardverfahren erhoben.
1978	das RSA-Verfahren wird veröffentlicht. Es ist das erste praktisch einsetzbare Public-Key-Verfahren.
1985	Goldwasser, Micali und Racoff stellen Zero-Knowledge-Verfahren vor.
1990	Xueija Lai und James Massey entwickeln das IDEA-Verfahren, das z.B. in der Kryptologiesoftware PGP (Pretty Good Privacy) eingesetzt wird.
2000	Das von den belgischen Kryptologen Joan Daemen und Vincent Rijmen entwickelte symmetrische Blockchiffreverfahren Rijndael wurde zum Advanced Encryption Standard (AES).

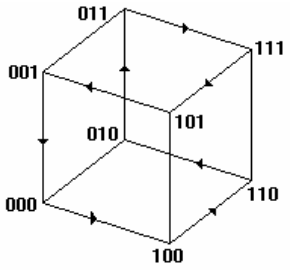
Gray-Code

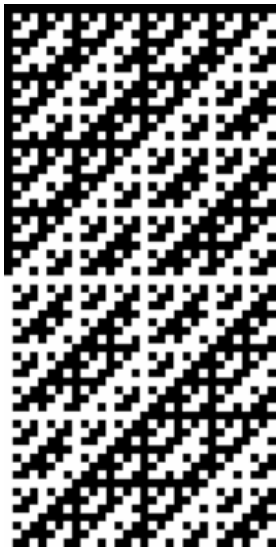
Der Gray-Code ist ein technisch wichtiger Code, bei welchem sich nachfolgende Codewörter um genau ein Bit unterscheiden.

Ermittlung des Gray-Codes

```
PROGRAM  gray;
```

```
TYPE  bit=0..1;
VAR   b,g,i,j,k,n : INTEGER;
      a : ARRAY[1..8] OF BIT; kk,kp: STRING;
BEGIN
   write('Codezahl: '); readln(n);
   IF n<2 THEN n:=3;   IF n>8 THEN n:=8;   g:=1;
   FOR i:=1 TO n DO g:=g*2;
   FOR i:=1 TO n DO a[i]:=0; i:=1; kk:='';
   REPEAT
    b:=i; k:=1;
   WHILE not odd(b) DO BEGIN b:=b div 2; inc(k) END;
    a[k]:=1-a[k];
    FOR j:=1 TO n DO BEGIN str(a[j],kp); kk:=kk+kp END;
    kk:=kk+', ';
    IF i mod 3=0 THEN BEGIN writeln(kk); kk:='' END;
    inc(i);
   UNTIL i=g; IF kk<>'' THEN writeln(kk);
END.
```



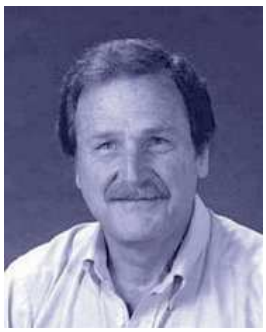


Hadamard-Code

Der Hadamard-Code ist ein binärer Blockcode zur Fehlererkennung und Fehlerkorrektur. Er wurde nach dem französischen Mathematiker Jacques Hadamard benannt.
Für große Blocklängen n haben die Hadamard-Codes eine schlechte Informationsrate, können aber viele Fehler korrigieren.

Der Code basiert auf den Hadamard-Matrizen. Ist H eine Hadamard-Matrix vom Rang 2^n, so werden die Codewörter konstruiert, indem die Zeilen von H und -H als Codewörter verwendet werden. Dabei werden die Einträge -1 durch 0 ersetzt.
Auf diese Art entstehen 2^{n+1} Codewörter der Länge 2^n. Da da die Zeilen von H orthogonal sind, unterscheiden sich zwei verschiedene Zeilen einer Hadamard-Matrix an 2^{n-1} Stellen. Damit ist der Hammingabstand 2^{n-1}.
Der Code kann höchstens $t = 2^{n-2}-1$ Fehler korrigieren. Wird ein Wort v empfangen, wird es zuerst in einen 1/-1 Vektor umgewandelt, indem alle Nullen durch -1 ersetzt werden. Nun wird vH^T berechnet. Der Eintrag mit dem höchsten Absolutwert korrespondiert mit der Zeile, die als Codewort verwendet wurde. Ist dieser Wert positiv, dann stammt das Wort aus H, ist er negativ, dann stammt das Wort aus -H.

Ein Hadamard-Code wurde 1971 in der Mariner 9 Mission zur Bildübertragung vom Mars verwendet. Die Datenwörter in dieser Mission waren 6 Bit lang und repräsentierten 64 Grauwerte. Die größte verwendbare Datenlänge war 30 Bit.
Anstelle eines Wiederholungscodes wurde der [32,6,16] Hadamard-Code verwendet; siehe Abbildung. Es konnten 7 Bit pro Wort korrigiert werden, 8 Bit Fehler wurden noch erkannt. Ein wichtiger Grund für die Verwendung dieses Codes war dessen effizienter Decodierungsalgorithmus. Die Entschlüsselungsmaschine wurde "Green Machine" genannt. Diese führte eine Schnelle Fourier-Transformation durch, die die Entschlüsselung um den Faktor 3 beschleunigte.



Huffman-Kodierung

Der in der Datenverarbeitung sehr weit verbreitete ASCII-Code kodiert jedes Zeichen mit 8 Bit. Beim Huffman-Verfahren (nach David Huffman 1925-1999, Abbildung) dagegen werden häufig vorkommende Zeichen mit wenigen Bits, selten vorkommende Zeichen dafür mit mehr Bits kodiert. Dies spart Speicherplatz gegenüber der ASCII-Kodierung.

Die Huffman-Kodierung beschreibt ein Verfahren, mit dessen Hilfe sich ein binärer Codebaum aufbauen lässt. Der von David Huffman 1952 beschriebene Algorithmus stellt dabei sicher, dass sich die Häufigkeit, mit der das jeweilige Zeichen auftritt, in der Kodelänge wiederspiegelt.
Zur Festlegung, welches Zeichen mit wie vielen Bits kodiert werden soll, müssen Informationen über die Zeichenhäufigkeiten vorhanden sein, wofür drei Möglichkeiten existieren:

1) statisch: die Zeichenhäufigkeiten werden vorher festgelegten Tabellen entnommen
2) dynamisch: die Daten werden einmal ganz gelesen, um die dort geltenden Häufigkeiten zu bestimmen
3) adaptierend: Es wird mit festen Vorgaben begonnen (z.B.: alle Zeichen treten gleich oft auf, oder das e ist das häufigste Zeichen, usw..), die während der Kodierung an die realen Daten angepasst werden.

Das Verfahren nach Huffman sorgt nicht nur für eine Kodierung, die eindeutig ist, sondern auch optimal, d.h. die Länge des kodierten Textes wird minimal.
Huffman-Kodes finden als Bestandteil von verschiedenen Datenformaten Anwendung, z.B. ZIP, GZIP oder JPEG.

Huffman-Kodierung, Beispiel

Beschreibung des Verfahrens siehe
Das Wort "abrakadabra" soll mit dem Huffman-Verfahren kodiert werden.
1. Die dynamische Bestimmung der Zeichenhäufigkeiten ergibt:
5 mal "a" 2 mal "b" 2 mal "r" 1 mal "k" 1 mal "d"
2. Bestimmung der Huffman-Codes
Zur Ermittlung der Huffman-Kodierung baut man einen Binärbaum auf. Dazu beginnt man mit den beiden seltensten Zeichen und kodiert diese mit 0 und 1, d.h. k mit 0 und d mit 1. Die Kodierung lautet bis jetzt:
k: 0 d: 1
Nun fasst man die Häufigkeiten dieser beiden Zeichen zusammen und sucht wieder die beiden seltensten. Das sind dann r und k mit d zusammen. Hiervon wird wieder eines mit 0, das andere mit 1 kodiert. Existiert für ein Zeichen schon ein Code, so wird die 0 oder 1 einfach davorgestellt. Die Kodierung lautet dann z.B.:
r:0 k: 10 d: 11
Wiederholung der Verfahrens:
b: 0 r: 10 k: 110 d: 111
Bei der letzten Wiederholung dieses Verfahrens wird a mit 0 kodiert und alle anderen erhalten eine vorangestellte 1:
a: 0 b: 10 r: 110 k: 1110 d: 1111
Das Wort "abrakadabra" wird dann also mit 01011001110011110101100 kodiert.

Dekodierung

Bei der oben dargestellen Kodierung wurde ein Binärbaum aufgestellt, wobei ein Pfad nach links einer Kodierung mit 0 und ein Pfad nach rechts einer Kodierung mit 1 entspricht.
Zur Dekodierung liest man einfach Bit für Bit ein und und wandert dabei durch den Binärbaum. Wenn ein Zeichen als Blatt gefunden wurde, ist dieses dekodiert und man beginnt mit dem nächsten Bit wieder an der Wurzel des Baumes.

Ergebnis

Die obige Kodierung benötigt für das Wort "abrakadabra" 23 Bit. Eine Kodierung in 8-Bit-ASCII verschlingt 88 Bit. Natürlich genügen für 5 verschiedene Zeichen auch 3 Bit, was bei 11 Zeichen insgesamt 33 Bit verbrauchen würde.

Lauflängenkodierung, RLE-Kodierung

Die Lauflängenkodierung (engl. Run-length encoding, RLE) ist ein einfacher verlustfreier Kompressionsalgorithmus für digitale Daten. Das Verfahren ist gut geeignet, Wiederholungen von gleichen Werten verkürzt darzustellen. Liegt eine Wiederholung vor, wird die Anzahl der Wiederholungen sowie der wiederholte Wert gespeichert.

Als Beispiel sei ein Schwarz-Weiß-Bildschirm gegeben, auf dem sich schwarzer Text auf weißem Hintergrund befindet. Dabei sind lange Folgen von weißen Punkten nur selten durch einzelne schwarze Punkte unterbrochen.
Bezeichnet man weiße Punkte mit "W" und schwarze mit "S", so ergibt sich z.B. in einer einzelnen Bildzeile die folgende Information: WWWWWWWWWWSWWWWWWWWWWSSS
Nach Anwendung der Lauflängenkodierung erhält man: 10W1S10W3S

In der Realität arbeitet man nicht mit Zeichen, sondern mit Bytes, die durch zweistellige Hexadezimalzahlen beschrieben werden.
Wählt man für Weiß 00 und für Schwarz FF, so ergibt sich folgendes Bitmuster:
000000000000000000FF000000000000000000FFFFFF
Die ursprünglich verwendeten 24 Buchstaben wurden somit in acht Buchstaben komprimiert, was einer Kompression auf ca. 33% der ursprünglichen Größe entspricht.
Bei einer ungünstigen Verteilung der Werte kann allerdings auch eine Vergrößerung der Datenmenge vorkommen.
Die unterschiedlichen Kompressionsprogramme speichern die komprimierte Datenfolge im Gegensatz zum obigen Beispiel in binärer Form ab, die teilweise sehr unterschiedlich sein kann.
Bekannte Dateiformate, die die Lauflängenkodierung anwenden, sind ältere Grafikformate wie beispielsweise Windows Bitmap, GEM Image, Targa oder PCX.

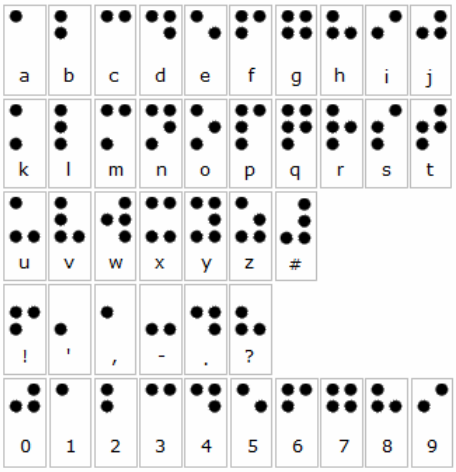
Lewenstein-Distanz

Die Lewenstein-Distanz Δ (auch Levenshtein-Distanz) zwischen zwei Zeichenketten ist die minimale Anzahl von Einfüge-, Lösch- und Ersetz-Operationen, um die erste Zeichenkette in die zweite

umzuwandeln. Benannt ist die Distanz nach dem sowjetischen Wissenschaftler Wladimir Lewenstein, der sie 1965 einführte.
Mathematisch ist die Lewenstein-Distanz eine Metrik auf dem Raum der Symbolsequenzen. Die Lewenstein-Distanz wird zur Bestimmung der Ähnlichkeit von Zeichenketten in der Rechtschreibprüfung genutzt.
Für die Lewenstein-Distanz Δ gilt:

- Δ beträgt mindestens den Unterschied der Längen beider Zeichenketten
- Δ beträgt höchstens die Länge der längeren Zeichenkette
- Δ ist genau dann 0, wenn beide Zeichenketten identisch sind
- Δ ist nicht größer als die Hamming-Distanz plus dem Längenunterschied der Zeichenketten

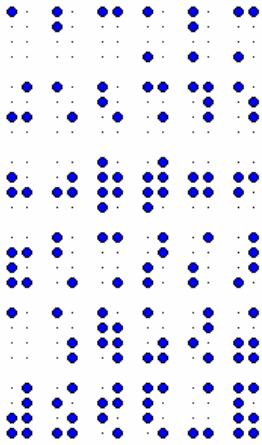
Quelle: Wikipedia



Brailleschrift

Die Brailleschrift wird von Sehbehinderten und Blinden benutzt. Sie wurde 1820 von dem französischen Arzt Louis Braille entwickelt. Die Schrift arbeitet mit Punktmustern, die in das Papier gepresst sind, so dass sie als Erhöhung mit den Fingerspitzen abgegriffen werden können.

Sechs Punkte, drei in der Höhe mal zwei Punkte in der Breite, bilden das Raster für Kombinationen, mit denen die Buchstaben dargestellt werden.
Bei sechs Punkten ergeben sich 64 Kombinationsmöglichkeiten; das Leerzeichen inbegriffen.
Für die Ausgabe von Brailleschrift durch den Computer werden Braillezeilen verwendet. Da für die Arbeit am Computer mehr Zeichen notwendig sind, als sich mit sechs Punkten festlegen lassen, wird hier auch oft noch eine vierte Zeile hinzugefügt, so dass acht Punkte zur Verfügung stehen (Computerbraille, Eurobraille). Auf diese Weise erhält man 256 Kombinationen. Die Codierung der Standardzeichen bleibt dabei jedoch gleich, die letzte Zeile bleibt lediglich leer.



Eurobraille

Eurobraille ist eine 8-Punkte-Computerbrailleschrift, die 1986 von verschiedenen europäischen Sprachen für die Computerbedienung definiert wurde.
Jedes Zeichen am Computer wird durch ein einziges Braillezeichen wiedergegeben und umgekehrt entspricht jedes Braillezeichen genau einem Bildschirmzeichen. Es besteht aus drei Tabellen von je 256 = 2^8 Zeichen, die drei Computerzeichensätze abbilden.
Durch die Erweiterung det klassischen Braille-Schrift um 2 Punkte konnte eine ein-eindeutige Zuordnung zum ASCII-Zeichensatz erreicht werden. An einem internationalen ISO-Standard wird gearbeitet.
Die als Auswahl abgebildeten Zeichen sind von rechts oben nach links unten:

a b c A B C
0 1 2 3 4 5
( ) [ ] = +
± ² ³ @ \ _
a á à â ä ã
å ø ð þ æ ç

Die 26 kleinen Grundbuchstaben (ohne Umlaute und ß) sind der 6-Punkte-Schrift gleich. Die 3 kleinen Umlautbuchstaben sind ebenfalls gleich, erhalten jedoch zusätzlich den Punkt 8. Die 26 Großbuchstaben entsprechen den kleinen, erhalten jedoch zusätzlich den Punkt 7. Die Ziffern 1 bis 9 bestehen aus den ersten 9 Kleinbuchstaben, jeweils mit einem zusätzlichen Punkt 6.
Weitere Gemeinsamkeiten und Unterschiede zwischen der 6- und 8-Punkte-Schrift werden zum Beispiel erläutert auf http://www.braille.ch/eb-erl-g.htm

Morsealphabet

Zur Übermittlung von Nachrichten mit Hilfe eines Telegraphen führte Samuel Morse (1791-1872, Abbildung) 1838 das nach ihm benannte Alphabet ein. Mit einigen Modifizierungen wurde es 1851 zum internationalen Morse-Alphabet erweitert. Das Morse-Alphabet basiert auf der Idee der Huffman-Kodierung: Häufige Zeichen erhalten kurze Codes, seltenere Zeichen lange.

a	• -	b	- • • •	c	- • - •	ch	- - - -	d	- • •
e	•	f	• • - •	g	- - •	h	• • • •	i	• •
j	• - - -	k	- • -	l	• - • •	m	- -	n	- •
o	- - -	p	• - - •	q	- - • -				

r	•-•	s	•••	t	-	u	••-	v	•••-	w	•--
x	-••-	y	-•--	z	--••	ä	•-•-	ö	---•	ü	••--
á	•--•-	ñ	--•--	0	-----	1	•----	2	••---	3	•••--
4	••••-	5	•••••	6	-••••	7	--•••	8	---••	9	----•
Punkt	•-•-•-	Komma	--•---	Fragezeichen		••--••		Doppelpunkt		---•••	
Apostroph	•-----•	Anführungsstriche		•-••-•							
Bindestrich	-•••-	Klammer		-•--•-							
Bruchstrich	-•••-	Trennung		•-••-							
Unterstreichung		••--•-									
Anfangszeichen	-•-•-	Warten	•-•••			Irrung	••••••				
Verstanden	•••-•	Aufforderung zum Geben		-•-							
Schluss der Übermittlung		•-•-•	Schluss des Verkehrs			•••-•-					

Leetspeak 1337

Leetspeak ist eine reine Schriftsprache und bezeichnet im Netzjargon das Ersetzen von Buchstaben durch ähnlich aussehende Ziffern oder Zeichen.
Die häufige Schreibweise 1337 für Leetspeak entstand aus dem englischen Wort Elite. Es wurde "leet" abgekürzt, was im Leetspeak als 1337 geschrieben wird.

Leetspeak kann schwer zu lesen sein und ist dadurch als eine Art Geheimcode bestimmter Gruppen der Computerszene zu betrachten.
Ursprünglich ernst gemeint, wird sie heute jedoch fast nur noch selbstironisch genutzt.
Streng genommen ist Leetspeak eine Substitutionschiffre, die allerdings auf einfache Weise dekodiert werden kann.
Je nach Art der Ersetzung der Ausgangsbuchstaben unterscheidet man verschiedene Leetspeak-Varianten, die teilweise für den Menschen nicht ohne Weiteres zu lesen sind.

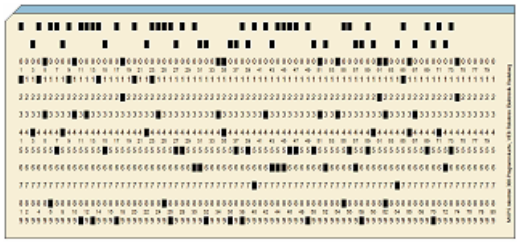
Varianten von Leetspeak:
1) Die Buchstaben LREASTG werden zu 1234579.
2) ABCDEFGHIJKLNOPRSTUXY werden zu ^8(Ð&ƒ64!¿<12*°®57µ%¥'
3) ABCDEFGHIJKLMNOPQRSTUVWXYÄÖÜ werden zu
/-\ |> (|) € |*|-| 6 /-/][_| |< 1 /\/\ /\\/ [] |* 0_ 2 § 7 [_] \/ \A/ }{ `/ "/_ °A° °O° °U°

ASCII-Tabelle

American Standard Code for Information Interchange (ASCII) ist eine 7-Bit-Zeichenkodierung und bildet eine Variante von ISO 646 sowie die Grundlage für spätere mehrbittige Zeichensätze und -kodierungen.
ASCII wurde im Jahr 1967 erstmals als Standard veröffentlicht und im Jahr 1986 zuletzt aktualisiert. Die Zeichenkodierung definiert 128 Zeichen, bestehend aus 33 nicht-druckbaren sowie 95 druckbaren.
Die Zeichen umfassen das lateinische Alphabet in Groß- und Kleinschreibung, die zehn arabischen Ziffern sowie einige Satz- und Steuerzeichen. Der Zeichenvorrat entspricht weitgehend dem einer Tastatur oder Schreibmaschine für die englische Sprache.

Code	Z.	Code	Z.	Code	Z.	Code	Z.	Code	Zeichen
32	Space	33	!	34	"	35	#	36	
37	%	38	&	39	'	40	(	41	)
42	*	43	+	44	,	45	-	46	.
47	/	48	0	49	1	50	2	51	3
52	4	53	5	54	6	55	7	56	8
57	9	58	:	59	;	60	<	61	=
62	>	63	?	64	@	65	A	66	B
67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L
77	M	78	N	79	O	80	P	81	Q
82	R	83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_	96	`
97	a	98	b	99	c	100	d	101	e
102	f	103	g	104	h	105	i	106	j
107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t
117	u	118	v	119	w	120	x	121	y
122	z	123	{	124	|	125	}	126	~
127		128							

Lochkartencode, Lochkarte

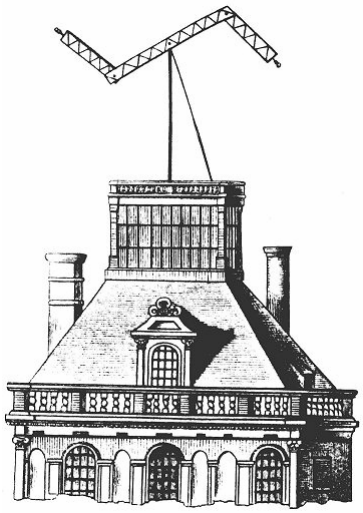


Eine Lochkarte ist eine Pappkarte, in die Löcher zur Informationsspeicherung, -verarbeitung und -übertragung eingestanzt werden. Dabei dient die Lochkarte als dauerhafter mechanischer Speicher für Dateneingaben.
Die Lochkarte wurde im Allgemeinen maschinell beschrieben und ausgelesen, kann aber auch manuell entziffert werden.
Die maschinelle Datenspeicherung geschieht durch den Lochkartenlocher, die Auslesung durch einen Lochkartenleser.
Bei dem R300 erfolgte das Kartenlesen fotoelektrisch mit einer Geschwindigkeit von 270 Zeichen/s, d.h. etwas mehr als 3 Karten je Sekunde. Der Lochkartenleser EC6016 erreichte sogar 1000 Karten/Minute!

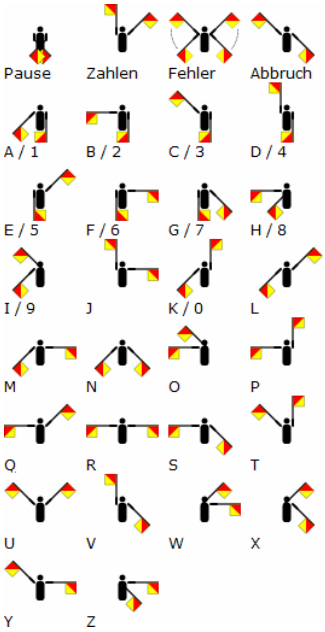
Für die Kodierung der Zeichen wird ein spezieller Lochkartencode verwendet. Auf einer Karte können maximal 80 Zeichen gespeichert werden, je Kartenspalte ein Zeichen.
Über den Zeilen 0 bis 9 mit je 80 Spalten befinden sich zwei zusätzliche Zeilen A und B.
Auf den Lochkarten des DDR-Großcomputers R300 wurden die Ziffern 0 bis 9 durch ein Loch an der Stelle 0 bis 9 markiert. Für die Buchstaben galt:
A … I : Loch bei A und 1 bis 9
J … R : Loch bei B und 1 bis 9
S … Z : Loch bei 0 und 2 bis 9
Für die Sonderzeichen gab es spezielle Codes.

Telegrafenalphabet

Unter optischer Telegrafie versteht man die Nachrichtenübermittlung über große Entfernungen mit Hilfe optischer oder einer Kombination von optischen mit akustischen Vorrichtungen.
Bereits in der Antike dienten Rauch- und Feuerzeichen zur Übermittlung von Nachrichten. Der griechische Dichter Aischylos beschrieb in seinem Drama "Agamemnon", wie die Nachricht vom Sieg der Griechen über Troja im Jahre 1184 v.u.Z. mit einer Feuerzeichenkette von Troja in das 555 km entfernte Argos gelangte.
Dem französischen Techniker Claude Chappe gelang eine technisch praktikable, optische Telegrafie-Vorrichtung, basierend auf der Zeichenübermittlung mit Hilfe von schwenkbaren Signalarmen. An einem fünf Meter hohen Mast mit zwei Querbalken war jeweils ein weiterer schwenkbarer Balken am Ende der Querbalken angebracht, der je nach Position unterschiedliche Buchstaben signalisierte.
Abbildung: Chappe-Telegraph



Die erste erste reguläre Telegrafenlinie zwischen Paris und Lille wurde 1794 eingerichtet, 22 Stationen auf 270 km. Die Laufzeit für die Übertragung eines einzelnen Buchstabens lag bei beeindruckenden zwei Minuten.
Jeder der beiden Signalarme konnte sieben verschiedene Stellungen einnehmen, die Querbalken noch jeweils zwei, d.h. insgesamt 7 x 2 x 7 x 2 = 196 verschiedene Zeichen.



Flaggenalphabet, Winkeralphabet

Das Flaggen- oder Winkeralphabet (Semaphore) dient zur optischen Nachrichtenübermittlung zwischen Schiffen. Mit der Entwicklung des Sprechfunks verlor es an Bedeutung, wird jedoch noch heute militärisch genutzt, da es nur schwer abhörbar ist.
Es existiert unter anderem ein internationales und ein deutsches Winkeralphabet.
Die einzelnen Buchstaben des lateinischen Alphabets werden beim Winkeralphabet durch die Stellung beschrieben, in der der Winker zwei Flaggen hält. Die Flaggen sind meist quadratisch und entweder diagonal Gelb-Rot geteilt (Flagge Oscar) oder rote Flaggen, die ein kleineres, zentriertes weißes Quadrat enthalten. Die Abbildungen zeigen die Signale, wie sie der Empfänger sieht.

Zahlen werden durch das "Zahlen folgen"-Signal angekündigt und

entsprechen den ersten zehn Buchstaben des Alphabets, Alfa bis Juliett. Die Zahl endet mit dem nächsten "Pause"-Signal. siehe auch http://www.anbg.gov.au/flags/semaphore.html

Internationales Flaggenalphabet

Das internationale Flaggenalphabet wird in der Schifffahrt verwendet, um Nachrichten auf optischem Wege durch Signalflaggen zwischen Schiffen auszutauschen.
Bei dem internationalen Flaggenalphabeten wird jeder Buchstabe des lateinischen Alphabets durch eine unterschiedlich farbig gestaltete Flagge signalisiert. Des Weiteren existieren Flaggen zur Signalisierung von Ziffern. Die meisten Flaggen des Flaggenalphabets besitzen darüber hinaus noch weitere spezielle Bedeutungen, zum Beispiel signalisiert Flagge A des internationalen Flaggenalphabets Taucher unten.

Im Gegensatz zum internationalen Flaggenalphabet wird beim Winkeralphabet (Semaphore) ein Buchstabe durch die Stellung von zwei Flaggen signalisiert.
Das heutige Flaggenalphabet wurde 1901 international verbindlich.

Buchstabe,deutsch,international

A	Anton	Amsterdam	B	Berta	Baltimore	C	Cäsar	Casablanca
D	Dora	Dänemark	E	Emil	Edison	F	Friedrich	Florida
G	Gustav	Gallipoli	H	Heinrich	Havanna	I	Ida	Italia
J	Julius	Jerusalem	K	Kaufmann	Kilogramm	L	Ludwig	Liverpool
M	Martha	Madagaskar	N	Nordpol	New York	O	Otto	Oslo
P	Paula	Paris	Q	Quelle	Quebec	R	Richard	Roma
S	Siegfried	Santiago	T	Theodor	Tripoli	U	Ulrich	Uppsala
V	Viktor	Valencia	W	Wilhelm	Washington	X	Xanthippe	Xanthippe
Y	Ypsilon	Yokohama	Z	Zeppelin	Zürich	Ch	Charlotte	
Sch	Schule		Ä	Ärger		Ö	Ökonom	
Ü	Übermut							

Abweichungen in Österreich

Ch	Christine	K	Konrad	Ö	Österreich
Ü	Übel	X	Xaver	Z	Zürich

Abweichungen in der Schweiz

A	Anna	Ä	Äsch	D	Daniel
J	Jakob	K	Kaiser	L	Leopold
M	Marie	N	Niklaus	Ö	Örlikon
P	Peter	R	Rosa	S	Sophie
X	Xaver	Y	Yverdon	Z	Zürich

Internationale Buchstabiertafel

Das ICAO-Alphabet ist ein seit dem 1.März 1956 gültiges Merkwortalphabet für den Flugfunksprachverkehr.
Es wurde von der Flugsicherungskommission der ICAO ausgearbeitet, um Schwierigkeiten im internationalen Luftverkehr zu vermeiden. Auch in der Seefahrt findet dieses Alphabet Anwendung.
Das ICAO-Alphabet ist ein wichtiger Grundstein für die Kommunikation. Neben der Verwendung im Sprechfunk dient es vielfältigen Zwecken wie der Bezeichnung von Positionen, Meldepunkten oder Schlüsselwörtern.

Ziffer/Buchstabe	Wort	Ziffer/Buchstabe	Wort	Ziffer/Buchstabe	Wort
0	Zero	1	One	2	Two
3	Three	4	Four	5	Five
6	Six	7	Seven	8	Eight
9	Niner	A	Alpha	B	Bravo
C	Charlie	D	Delta	E	Echo
F	Foxtrot	G	Golf	H	Hotel
I	India	J	Juliett	K	Kilo
L	Lima	M	Mike	N	November
O	Oscar	P	Papa	Q	Quebec
R	Romeo	S	Sierra	T	Tango
U	Uniform	V	Victor	W	Whiskey
X	X-ray	Y	Yankee	Z	Zulu

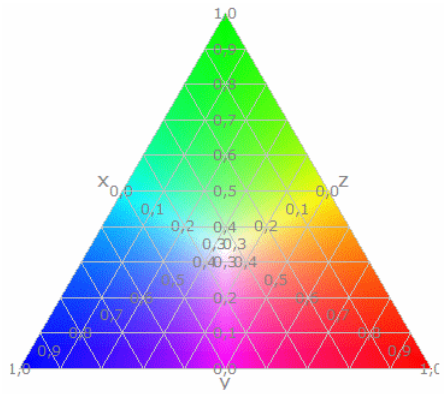
Eine weitere Regel gibt es für das Dezimaltrennzeichen, ganz gleich ob es ein Punkt oder ein Komma ist, wird es stets Decimal ausgesprochen.

QR-Code

Der QR-Code (engl. Quick Response = schnelle Antwort) ist ein zweidimensionaler Code, der von der japanischen Firma Denso Wave im Jahr 1994 entwickelt wurde.

Der QR-Code wurde zur Markierung von Baugruppen und Komponenten für die Logistik in der Automobilproduktion des Toyota-Konzerns entwickelt.
Der Code besteht aus einer quadratischen Matrix aus schwarzen und weißen Punkten, die die kodierten Daten binär darstellen. Eine spezielle Markierung in drei der vier Ecken des Quadrats gibt die Orientierung vor. Die Daten im QR-Code sind durch einen fehlerkorrigierenden Code geschützt. Dadurch wird der Verlust von bis zu 30 % des Codes toleriert, d.h. er kann auch dann noch dekodiert werden.

Im Code enthalten sind die Versionsinformation und das benutzte Datenformat. Der Datenteil enthält die kodierten Daten in redundanter Form. Zur Feldbegrenzung enthält der QR-Code in nur drei seiner Ecken ein bestimmtes Muster. Über das fehlende Muster in der vierten Ecke erkennt das Lesegerät die Orientierung. Mit zunehmender Größe des Codes werden weitere Muster hinzugefügt, um die Ausrichtung des Codes besser erkennbar zu machen. Zwischen den drei Hauptpositionsmarkierungen befindet sich eine Linie aus einer Folge streng abwechselnder Bits, worüber sich die Matrix definiert.



Maxwell-Dreieck

Durch den schottischen Physiker James Clark Maxwell (1831-1879) wurde das nach ihm benannte Farbdreieck eingeführt.
Dieses Dreieck demonstriert die Farbmischung aus den drei Grundfarben Blau, Rot und Grün. Diese drei Farben bezeichnet man als additive Primärfarben, die in etwa gleicher Farbintensität zusammen weißes Licht ergeben.
Die Mischfarben ergeben sich durch Addition der Grundfarben, die prozentual gewichtet werden.

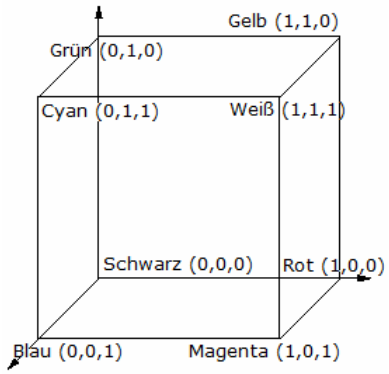
Nach Definition ist die Summe der Primärfarben stets 1.

Farbkreis nach Itten

Der Farbkreis nach Johannes Itten wurde nach Anforderungen der Kunstpädagogik entwickelt.
Der Farbkreis besteht aus zwölf Farben mit jeweils eigenem Charakter und eindeutiger Beziehung zu den anderen Farben.

Ausgangspunkt ist die Festlegung von drei Grundfarben: Rot, Gelb und Blau. Im Farbkreis Ittens werden diese Farben im Zentrum angeordnet und als Farben erster Ordnung bezeichnet. Durch Mischung von jeweils zwei dieser Grundfarben entstehen die Farben zweiter Ordnung: Orange, Violett und Grün.
Durch Mischen der Farben zweiter Ordnung mit ihren benachbarten Farben erster Ordnung erhält man die Farben dritter Ordnung, die sechs Zwischenstufen.

Die Farben werden so aufeinander abgestimmt, dass die Farbabstände möglichst gleich erscheinen, dass gegenüberliegende Farben jeweils komplementär sind und dass eine Bewegung von oben hell nach unten dunkel erfolgt.

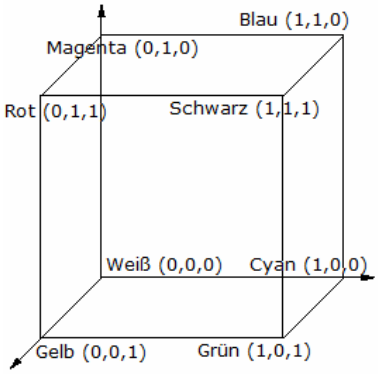


RGB-Modell (Rot, Grün, Blau), additiv

Zur Ansteuerung der dreifarbigen Phosporschicht mit roten, grünen, blauen Phosphorpunkten eines Monitors oder Fernsehers bietet sich das RGB-Modell an.
Es handelt sich um ein additives Farbmodell, da das von den Phosphorpunkten ausgehende Licht addiert wird.
Die typische Darstellung durch einen Einheitswürfel ist in der Abbildung zu sehen, dem RGB-Würfel.
Häufig werden die drei RGB-Werte statt im reellen Wertebereich [0, 1] in 256 Abstufungen als ganze Zahlen im Bereich {0, 1, …, 255} angegeben, die in einem Byte kodiert werden. Dadurch ergibt sich die Darstellung einer Farbe in drei RGB-Bytes.

R	G	B	hexadezimal	Farbe
0	0	0	000000	Schwarz
255	0	0	FF0000	Rot
0	255	0	00FF00	Grün

0	0	255	0000FF	Blau
255	255	0	FFFF00	Gelb
255	0	255	FF00FF	Magenta
0	255	255	00FFFF	Cyan
255	128	128	FF8080	Hellrot
128	255	128	80FF80	Hellgrün
128	128	255	8080FF	Hellblau
64	64	64	404040	Dunkelgrau
128	128	128	808080	Grau
192	192	192	C0C0C0	Hellgrau
255	255	255	FFFFFF	Weiß



CMY-Modell (Cyan, Magenta, Yellow), subtraktiv

Bei Farbdrucken empfängt das Auge nur solche Anteile des weißen Lichts, die reflektiert werden. Zur Erklärung bietet sich ein subtraktives Modell an.
Ein CMY-Tripel beschreibt, wieviel von den Grundfarben Cyan, Magenta, Yellow reflektiert bzw. von den Grundfarben Rot, Grün, Blau absorbiert wird.

Es gilt: (0,0,0) absorbiert nichts bleibt Weiß
(0,0,1) absorbiert Blau bleibt Gelb
(0,1,0) absorbiert Grün bleibt Magenta
(1,0,0) absorbiert Rot bleibt Cyan
(0,1,1) absorbiert Cyan bleibt Rot
(1,0,1) absorbiert Magenta bleibt Grün
(1,1,0) absorbiert Gelb bleibt Blau
(1,1,1) absorbiert alles bleibt Schwarz

Beispiel: (0,1,0) Magenta gemischt mit (0,0,1) Gelb ergibt (0,1,1) Rot ; (1,0,0) Cyan gemischt mit (0,0,1) Gelb ergibt (1,0,1) Grün ; (1,0,0) Cyan gemischt mit (0,1,0) Magenta ergibt (1,1,0) Blau
Die Umrechnung zwischen dem CMY-und dem RGB-Modell erfolgt in Vektorschreibweise über die Subtraktionen

[R, G, B] = [S, S, S] - [C, M, Y] [C, M, Y] = [W, W, W] - [R, G, B]

wobei die Vektoren [S,S,S] im CMY-Modell und [W,W,W] im RGB-Modell gleich [1,1,1] sind.

YIQ-Modell

Beim 1953 in den USA eingeführten NTSC-System (National Television Standards Committee) werden die Farben durch die Farbparameter YIQ beschrieben, wobei Y die Helligkeit darstellt. Die Umrechnung zum RGB-Modell erfolgt durch die Formeln

Y = 0,299 R + 0,587 G + 0,114 B I = 0,596 R - 0,274 G - 0,322 B
Q = 0,211 R - 0,522 G + 0,311 B R = Y + 0,956 I + 0,623 Q
G = Y - 0,272 I - 0,648 Q B = Y - 1,105 I + 1,705 Q

Beim europäischen PAL-System (Phase Alternating Line) werden statt der Parameter I und Q die Farbdifferenzen R - Y und B - Y übertragen. Die Konvertierung der Farbinformation in Monochrom-Darstellung erfolgt in beiden Systemen durch Auswertung des Helligkeitsparameters Y.

YUV-Modell

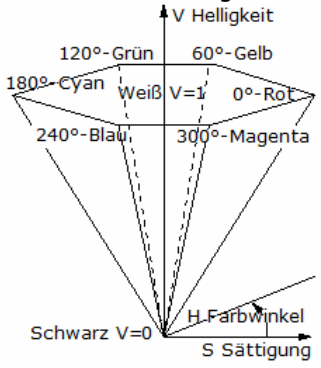
Ein Farbwert wird beschrieben durch ein YUV-Tripel, wobei Y die Helligkeit (Luminanz) bezeichnet und U,V Farbdifferenzen (Chrominanz). Die Helligkeitsempfindung resultiert zu 59 % aus dem Grünanteil, zu 30 % aus dem Rotanteil und zu 11 % aus dem Blauanteil:

$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$

In den Farbdifferenzen; mit begründeten Normierungsfaktoren; ist die restliche Information codiert:

$U = 0{,}493 \cdot (B - Y)$ $V = 0{,}877 \cdot (R - Y)$

Der Vorteil dieses Farbmodells liegt darin begründet, dass in der Y-Komponente das Bild als Matrix von Grauwerten vorliegt und separat von der Farbinformation weiterverarbeitet werden kann.



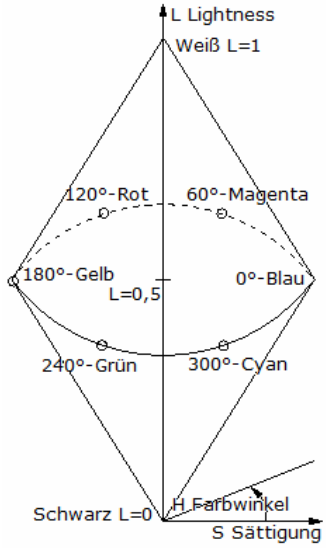
HSV-Modell

Zur intuitiven Beschreibung einer Farbe eignet sich das HSV-Modell, welches jede Farbe durch ein Tripel Hue = Farbton; Saturation = Sättigung; Value = Helligkeit beschreibt.
Projiziert man den RGB -Würfel längs der Weiß-Schwarz-Diagonale, so entsteht ein Sechseck, dass die Basis einer Pyramide bildet.
Die Wahl des Farbtons (hue) geschieht durch Angabe eines Winkels (0° = Rot).
Der Parameter V liegt zwischen 0 und 1 und bestimmt die Intensität der Farbe; dargestellt durch die Senkrechte. Der Parameter S liegt zwischen 0 und 1 und bestimmt die Reinheit der Farbe (Entfernung von der

Senkrechten). Die Farbselektion kann erfolgen, indem z.B. zunächst eine reine Farbe ausgewählt wird (H = α, V = 1, S = 1). Das Hinzumischen von Weiß zur Erzeugung von Pastellfarben erfolgt durch Reduktion von S bei konstantem V. Das Hinzumischen von Schwarz erfolgt durch Reduktion von V bei konstantem S.
Die Achse V entspricht der Diagonalen im RGB-Würfel durch die Punkte Schwarz und Weiß. Daher ist der Wert für V gleich dem Maximum der RGB-Intensitäten. Die Werte H und S können aus der Position des Punktes in jenem Sechseck berechnet werden, das durch Projektion des kleinsten, den RGB-Punkt beinhaltenden Würfels erzeugt wird.

V = max (R, G, B) mi = min (R, G, B)
S = (V - mi) / V H = (1 + (V-R)/(V-mi)) β

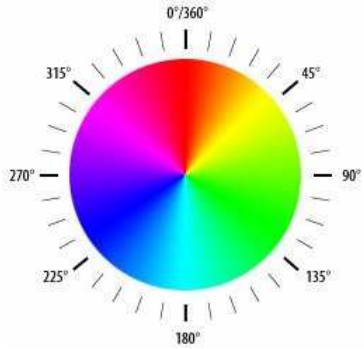
Der Winkel β wird aus dem Maximum und dem Minimum der Farbanteile ermittelt.



HSL-Modell

Das HSL-Modell (hue, saturation, lightness) ist dem HSV-Modell ähnlich. Die Farbe (hue) wird ebenfalls als Winkel in einem Farbenkreis angegeben. Die Reihenfolge der Farben auf diesem Kreis ist identisch der beim HSV-Modell, jedoch liegt beim HSL-Modell die Farbe Blau bei 0°, um 120° verdreht.
Auch der Parameter S(aturation) entspricht dem HSV-Modell und liegt im Bereich [0,1]. Der Wert S = 1 entspricht maximaler Reinheit, und für S = 0 erhält man die Grauskala.
Der Parameter L(ightness) entspricht in dem Parameter V(alue), wobei allerdings aus der Pyramide durch Hochziehen des Punktes S = 0, V = 1 ein Doppelkegel entsteht. Die Werte L = 0 und L = 1 beschreiben die Farben Schwarz und Weiß, und die Farben maximaler Sättigung liegen auf der Ebene durch L = 0,5.

HSL-RGB-Umrechnung



Die Umrechnung einer Farbe aus dem HSL-Modell (hue, saturation, lightness, Abbildung: hue-Kreis) in das RGB-System ist etwas aufwendiger.
Dazu seien die drei Parameter h, s und l als rationale Zahl im Bereich [0; 1] gegeben. Dann wird:

```
Wenn s = 0 dann r = g = b = l; sonst
Wenn (l < 0.5) dann t2 = l · (1 + s) sonst t2 = (l + s) - (l · s);
      t1 = 2 · l - t2;    tr = h + 1/3;
      wenn (tr > 1) dann tr = tr-1 sonst tg = h;
      tb = h - 1/3;
      wenn (tb < 0) dann tb = tb+1;
//Rot
      wenn (tr < 1/6) dann r = t1 + (t2-t1) · 6 · tr;
      sonst wenn (tr < 0.5) dann r = t2;
      sonst wenn (tr < 2/3) dann r = t1 + (t2-t1) · (2/3-tr) · 6;
      sonst r = t1;
//Grün
      wenn (tg < 1/6) dann g = t1 + (t2-t1) · 6 · tg;
      sonst wenn (tg < 0.5) dann g = t2;
      sonst wenn (tg < 2/3) dann g = t1 + (t2-t1) · (2/3-tg) · 6;
      sonst g = t1;
//Blau
      wenn (tb < 1/6) b = t1 + (t2-t1) · 6 · tb;
      sonst wenn (tb < 0.5) dann b = t2;
      sonst wenn (tb < 2/3) dann b = t1 + (t2-t1) · (2/3-tb) · 6;
            sonst b = t1;
colorRGB.r = int(255·r) ; colorRGB.g = int(255·g) ; colorRGB.b = int(255·b);
```

Negativbild

Ist ein Bild im RGB-Modell gegeben, so kann das Negativ des Bildes erzeugt werden, in dem jeder Farbanteil F (Rot, Grün, Blau) durch das Komplement 255 - A ersetzt wird.
Quelltext für ein Bild "flower.bmp":

```
ColorRGB image[200][133];
int main(int argc, char *argv[])
{ screen(200, 133, 0, "RGB Color");
loadBMP("flower.bmp",image[0], 200, 133);
ColorRGB color; //the color for the pixels
```

```
for(int x = 0; x < w; x++)
for(int y = 0; y < h; y++)
{ color.r = 255 - image[x][y].r;
color.g = 255 - image[x][y].g;
color.b = 255 - image[x][y].b;
pset(x, y, color); }
redraw();
return 0; }
```

Graustufenbild

Ist ein Farbbild im RGB-Modell gegeben, so kann ein Graustufenbild erzeugt werden, in dem jedem Pixel der Mittelwert der drei Farbanteile Rot, Grün, Blau zugewiesen wird.
Quelltext für ein Bild "flower.bmp":

```
ColorRGB image[200][133];
int main(int argc, char *argv[])
{       screen(200, 133, 0, "RGB Color");
loadBMP("flower.bmp",image[0], 200, 133);
ColorRGB color; //the color for the pixels
for(int x = 0; x < w; x++)
for(int y = 0; y < h; y++)
{       color.r = image[x][y].r - 50;
color.g = image[x][y].g - 50;
color.b = image[x][y].b - 50;
color.r = color.g = color.b = (color.r + color.g + color.b) / 3;
pset(x, y, color);      }
redraw();
return 0;       }
```

Die einfache Formel Graustufenbild = 1/3 x Rot + 1/3 x Grün + 1/3 x Blau erzeugt allerdings keine sehr guten Schwarzweißbilder.
Werden die Farben entsprechend der menschlichen Wahrnehmung, die erhöhte Grünempfindlichkeit des menschlichen Auges, gewichtet, so wird:

Graustufenbild = 0,3 x Rot + 0,59 x Grün + 0,11 x Blau

Farbwahrnehmung

Obwohl das menschliche Auge das wichtigste Sinnesorgan des Menschen darstellt, ist es doch in seiner Leistungsfähigkeit etwas begrenzt und kann sich mit den Augen einiger Tiere nicht vergleichen.
Insbesondere die Wahrnehmung der Farben ist begrenzt. Nach der Theorie des Farbensehens von Young (1773-1829) und von Helmholtz (1821-1894) wird diese Dreikomponententheorie oder Theorie des trichromatischen Sehens genannt. Dabei befinden sich in der Netzhaut des menschlichen Auges drei Typen von Zapfen vor, welche für unterschiedliche Frequenzen des Lichts empfindlich sind. Die maximale Empfindlichkeit ergibt sich für

· Blauviolettrezeptor bei 450 nm
· Grünrezeptor bei 530 nm und
· Rotrezeptor bei 570 nm

Trifft Licht nun auf die Netzhaut, werden diese Rezeptoren unterschiedlich stark gereizt und diese Reize im Gehirn zu dem Farbeneindruck "gemixt". Einheitlich gefärbte Flächen kann der Mensch sehr gut unterscheiden. Wird die Fläche aber mit einem Raster von zwei verschiedenen Farbpunkten überzogen, tritt der Effekt ein, dass man bei hinreichend kleinen Punkten die zwei verwendeten Farben nicht mehr exakt trennen kann und eine Mischfarbe wahrnimmt. Dieser Effekt wird z.B. beim Druck von Photos und Abbildungen in Zeitungen und Zeitschriften verwendet, wo diese Abbildungen aus sehr kleinen Punkten der drei Grundfarben Rot, Grün und Blau zusammengesetzt werden. Dieses Verfahren wird ebenso bei der Erzeugung des farbigen Fernsehbildes oder bei einem Computer-Farbdrucker verwendet.

Automatentheorie, Regeln und Zeichenketten

Die Beschäftigung mit Sprachen und Sprachumfängen sind Thema der Automatentheorie. Dort werden Computer einfach nur Automaten genannt. Man kann das Ganze dann auch so nennen: Die Automatentheorie beschäftigt sich mit den Möglichkeiten und Grenzen der automatischen Realisierbarkeit von natürlicher Intelligenz, wobei hier Intelligenz alles umfasst, was Nachdenken erfordert: also das Erringen des Nobelpreises ebenso wie das Errechnen des maximal verfügbaren Budgets für den nächsten Schallplattenkauf. Hintergrund dieser automatischen Realisierbarkeit von natürlicher Intelligenz sind die formalen Sprachen.

Formale Sprache

Formale Sprachen sind Mengen aus Regeln und Grundzeichen.
Die Regeln besagen dabei, wie aus diesen verfügbaren Grundzeichen Zeichenketten gebildet werden können, oder sie können entscheiden, welche Zeichenketten zu einer bestimmten Sprache gehören und welche nicht.
Oder nochmals anders gesagt: Die Regeln, die eine bestimmte Sprache ausmachen, können zur Analyse und zur Synthese von Zeichenketten verwendet werden. Bei der Analyse wird eine bestimmte Zeichenkette auf ihre Bestandteile reduziert. Sind die Bestandteile Elemente der Grundzeichen, dann gehört die Zeichenkette zu der Sprache, die durch die Regeln beschreibbar ist; sind sie nicht Elemente der Grundzeichen, dann gehört die Zeichenkette nicht zu dieser Sprache.
Bei der Synthese wird umgekehrt vorgegangen: Die Regeln bilden aus den Grundzeichen Zeichenketten, und alle Zeichenketten, die aus den Grundzeichen durch Anwendung der Regeln gebildet werden können, gehören zu dieser Sprache. Folglich, da die Kombinationsmöglichkeiten zwischen Grundzeichen und Regeln endlos zu sein scheinen, kann eine anscheinend nicht-abbrechende Menge von Wörtern einer bestimmten Sprache erzeugt werden.
Je nach Komplexitätsstufe der Sprache, das heißt ihrer Regeln, ist also die Anzahl der erzeugbaren Wörter kleiner oder größer - bis hin zu universellen Sprachen, von denen man glaubt, dass in ihnen sogar natürliche Intelligenz höherer Stufe darstellbar sei.

Sprachen

Es lassen sich insgesamt vier Sprachstufen unterscheiden. Mit wachsender Komplexität und Umfang sind das die regulären Sprachen, die kontextfreien Sprachen, die kontextsensitiven Sprachen und die unbeschränkt-allgemeinen Sprachen (d.h. die rekursiv-aufzählbaren).
In der Automatentheorie wird gezeigt, dass in diesen Grammatiken von Stufe zu Stufe unterschiedliche Regelformen zur Anwendung kommen. Jeder dieser Stufen oder Sprachen und Regelformen entspricht dann genau einem Automat. Das sind dann mit wachsender Komplexität die schon genannten Automaten: der endliche Automat mit/ohne Ausgabe, der Kellerautomat, der linear beschränkte Automat und die Turing-Maschine.

Regel

Eine Regel ist ganz allgemein eine bestimmte, für einen konkreten Fall festgelegte Handlungsanweisung. Sie verbindet zwei Aussagen miteinander unter dem Primat der Wenn-dann-Beziehung:

"Wenn x, dann y".

Eine Regel ist damit eine Handlungsvorschrift, wie sie in Kochrezepten oder Computerprogrammen vorkommt. Es sind Handlungsanleitungen, die mehr oder weniger mechanisch ausgeführt werden sollen, also eine Sache genau für Computer. **A -> B.**
Diese Regel sagt aus: "Wenn A, dann B" bzw. "Aus A folgt B" bzw. "Wer A hat, der hat auch B" bzw. "Aus A kann man B ableiten".

Reguläre Sprachen

Die regulären Sprachen sind elementarste Stufe der Sprachen überhaupt, die von Computern verarbeitet werden.
Sie verfügen nur über rechts- und linksrekursive Regeln (genauer: rechts- und linkslineare Regeln).
Rechts- und linksrekursive bzw. -lineare Regeln besitzen folgende Form:

rechtsrekursive Regel $A \rightarrow aA$
linksrekursive Regel $B \rightarrow Bb$

Bei diesen Regeln kommt es nicht auf die Groß- und Kleinbuchstaben selbst an und nicht darauf wie sie heißen, sondern wichtig ist nur, dass auf der linken Regelseite nur Großbuchstaben und dass auf der rechten Seite der ersten Regel ein Kleinbuchstabe links vom Großbuchstaben (rechtslinear) bzw. auf der rechten Seite der zweiten ein Kleinbuchstabe rechts vom Großbuchstaben (linkslinear) stehen muss.

Bei den Großbuchstaben handelt es sich um sogenannte nicht-terminale Symbole (bzw. Variable), also um die Elemente VN = {A,B}; bei den Kleinbuchstaben um sogenannte terminale Symbole (bzw. Konstante), also um die Elemente VT = {a,b}.
Nicht-terminale und terminale Symbole unterscheiden sich darin, dass die ersteren so lange durcheinander ersetzt werden müssen, bis die entstehende Zeichenfolge nur noch aus terminalen Symbolen besteht.
Man verfügt über eine Anzahl von Regeln, zum Beispiel der folgenden Art:

$a \rightarrow a$; $b \rightarrow b$; $A \rightarrow Aa$; $B \rightarrow Ab$; $S \rightarrow Ba$

Mithilfe dieses Regelapparates P lassen sich folgende Zeichenketten erzeugen oder analysieren:

S => Ba => Aba => Aaba => Aaaba => aaaba
aaaaba => Aaaaba => Aaaba => Aaba => Aba => Ba => S

Im ersten Fall wurde ausgehend vom Start-Symbol S durch schrittweise Regel-Anwendung eine Zeichenfolge erzeugt; im zweiten Fall wurde eine beliebige Zeichenfolge durch Anwendung des Regelapparates bis auf das Start-Symbol S reduziert. Im ersten Fall handelt es sich daher um eine Synthese (Generierung), im zweiten um eine Analyse von Zeichenfolgen.

Kann im Laufe einer Analyse einer Zeichenfolge durch keine der möglichen Regel-Anwendungen das Start-Symbol S erreicht werden, so können genau zwei Fälle vorliegen:
Erstens kann es sein, dass der vorliegende konkrete Regelapparat P nicht in der Lage ist, diese Zeichenfolge zu analysieren; dann muss ein anderer Regelapparat P' aus rechts- und linksrekursiven Regeln gefunden werden, der diese Aufgabe erledigt.

Zweitens besteht aber auch die Möglichkeit, dass eine Zeichenfolge vorliegt, die durch keinen möglichen Regelapparat von links- und rechts-rekursiven Regeln analysiert werden kann. In diesem Fall ist der Bereich von Wörtern und Sprachen, der durch links- und rechtsrekursive Regeln erfasst werden kann, überschritten.

Damit haben die regulären Sprachen, die auf der Basis von links- und rechtsrekursiven Regeln darstellbar sind, nur einen begrenzten und endlichen Umfang. Eine dieser regulären Sprachen kann jeweils durch einen solchen Regelapparat P beschrieben werden. Für die Veranschaulichung dieser regulären Sprachen reichen die sogenannten endlichen Automaten A mit bzw. ohne Ausgabe vollkommen aus.
Man kann dafür auch sagen: **L(G) = L(A),**
das heißt: Die Sprache L über G ist gleich der Sprache L über A.

Oder: Die durch eine Grammatik der Form **G = [VT, VN, P, S]**
symbolisierte Sprache ist gleich der Sprache, die aufgrund derselben Regeln durch einen Automaten A mit bzw. ohne Ausgabe überprüft bzw. erzeugt werden kann.
Eine solche Grammatik ist folglich ein Quatrupel der Form G = [VT, VN, P, S] und besteht aus der Menge der terminalen Symbole VT, der Menge der nicht-terminalen Symbole VN, dem Regelapparat P und dem Start-Symbol S.

Grenzen der regulären Sprachen

Es existieren bestimmte formale Sprachen, die auf der Basis von links- und rechtsrekursiven Regeln nicht dargestellt werden können.
Es sind Sprachen höherer Komplexität. Die Grenzen der regulären Sprachen sind aber zugleich die Grenzen des endlichen Automaten mit bzw. ohne Ausgabe. Denn es gilt: L(G) = L(A).
Deshalb wird hier eine solche Sprache angegeben, die über die regulären Sprachen und den Sprachumfang dieser endlichen Automaten hinausgeht. Eine solche Sprache ist zum Beispiel die Sprache mit der Form: **$L(G) = \{x \mid x = a^n b^n, n \in N\}$**
über dem Alphabet E = {a, b}. Worte dieser Sprache sind Zeichenfolgen, die stets über dieselbe Anzahl von a- und b-Elementen verfügen: ab, aabb, aaabbb, usw.

Um die Worte dieser Sprache $L(G) = \{x \mid x = a^n b^n, n \in N\}$ erzeugen bzw. analysieren zu können, müsste dieser Automat in der Lage sein, sich die Anzahl der a-Elemente zu merken, um sie anschließend auf die b-Elemente zu übertragen.
Diese Aufgabe leitet zu den kontextfreien Sprachen und dem zugehörigen Kellerautomat über.

Kontextfreie Sprachen

Die kontextfreien Sprachen verfügen nicht nur über links- und rechts-rekursive Regeln, sondern zusätzlich über die zentralrekursive Regel.
Damit können in kontextfreien Grammatiken insgesamt drei Regelformen auftreten:
linksrekursive Regel $A \rightarrow A\ a$
rechtsrekursive Regel $A \rightarrow a\ A$
zentralrekursive Regel $A \rightarrow a\ A\ b$

Dabei kommt es nicht auf die Bezeichnung von Klein- und Großbuchstaben an, sondern darauf, dass links nur ein Großbuchstabe und niemals Kleinbuchstaben stehen dürfen und rechts die Verteilung von Kleinbuchstaben wie angegeben erfolgen kann. Rechts der Regeln darf eine beliebige Folge von terminalen und nicht-terminalen Symbolen stehen.
Mit Hilfe dieser drei Regelformen lässt sich für die Sprache **$L(G) = \{x \mid x = a^n b^n, n \in N\}$**
eine Regelbeschreibung und damit eine Grammatik finden. Der zu dieser Sprache gehörige Regelapparat P besitzt nur zwei Regeln: **$A \rightarrow ab$** **$A \rightarrow aAb$**
Die Wörter dieser Sprache werden zentral von der Mitte her gebildet oder so analysiert, dass zunächst von der Mitte ausgehend stets ein a- und ein b-Element entfernt wird. Durch die Entfernung immer nur eines a- und eines b-Elements wird sichergestellt, dass nur solche Zeichenfolgen als Wörter dieser Sprache analysiert werden können, die aus gleich vielen a- und b-Elementen bestehen:

$aaaabbbb \rightarrow aaabbb \rightarrow aabb \rightarrow ab \rightarrow A$
$A \rightarrow ab \rightarrow aabb \rightarrow aaabbb \rightarrow aaaabbbb \rightarrow \ldots$

Ein Kellerautomat analysiert eine solche Zeichenfolge, indem er für jedes gelesene a-Element einen Eintrag in einem speziellen Speicherbereich macht. Wenn er schließlich zum Lesen der b-Elemente kommt, löscht er für jedes gelesene b-Element eines der gespeicherten a-Elemente, bis weder ein weiteres b-Element gelesen noch ein weiteres a-Element noch gelöscht werden muss.

Kontextfreie Grammatik

In jedem anderen Fall wäre entweder dieser spezielle Speicherbereich nach dem Lesen aller b-Elemente noch nicht leer oder der Speicherbereich wäre bereits leer, ohne dass alle b-Elemente bereits gelesen wurden. In den letzten beiden Fällen würde der Kellerautomat die Verarbeitung der Zeichenfolge vorzeitig abbrechen. Mit diesem Abbruch würde er anzeigen, dass das Eingabewort nicht zu dieser Grammatik gehört.

$$\mathbf{L(G) = \{x \mid x = a^n b^n, n \in N\}}$$

ist eine kontextfreie Grammatik, die durch Kellerautomaten dargestellt werden kann.
Der Grund dieser Fähigkeit des Kellerautomaten liegt darin begründet, dass dieser eine gewisse Merkfähigkeit besitzt, der an einen Stack erinnert.
D.h., er kann sich durch Lesen und sogenanntes Abkellern der a-Elemente ihre Zahl merken, um sie anschließend feldweise mit den b-Elementen zu vergleichen. Damit gehören die kontextfreien Grammatiken einschließlich der Grammatiken der regulären Sprachen zum Sprachumfang des Kellerautomaten.

Grenzen der kontextfreien Sprachen und Sprachumfang des Kellerautomaten

Zwar besitzt der Kellerautomat einen größeren Sprachumfang als der endliche Automat mit und ohne Ausgabe, jedoch kann auch für den Kellerautomaten eine Sprache angegeben werden, für die dieser Automat keinen Akzeptor darstellt.
Eine solche Sprache hat zum Beispiel mit dem Alphabet E = {a, b, c} folgende Form:

$L(G) = \{x|x = a^n b^n c^n, n \in N\}$.

Die Worte dieser Sprache sind beispielsweise abc, aabbcc, aaabbbccc, usw. Diese Worte besitzen stets die gleiche Zahl an a-, b- und c-Elementen.

Beim Kellerautomaten rührt die Nichtakzeptanz dieser Sprache vom Bau seines Speichers her, denn er wird beim Lesen der a-Elemente deren Anzahl in seinem Speicher vermerken, um sie dann auf die folgenden b-Elemente zu übertragen.
Dies hat zur Folge, dass er danach keine Möglichkeit mehr besitzt, die verbleibenden c-Elemente zu überprüfen, denn nach Überprüfung der b-Elemente ist durch Löschen der a-Elemente dieser Speicher wieder leer.

Kontextsensitive Sprachen

Die Sprache mit der allgemeinen Wortform $a^n b^n c^n$ gehört zu den kontextsensitiven Sprachen.
Diese Sprachen zeichnen sich dadurch aus, dass auf der rechten und linken Regelseite beliebig viele terminale und/oder nicht-terminale Symbole stehen dürfen. Damit können kontextsensitive Grammatiken Regeln enthalten, die nur umgebungsabhängig angewendet werden können.
Damit liegen für die kontextsensitiven Sprachen und ihre Regeln keine Einschränkungen mehr vor, allerdings mit Ausnahme der einen, dass bei der Analyse von Zeichenfolgen das Wort stets kürzer und bei der Synthese bzw. Generierung das Wort stets länger werden muss.
Die allgemeinen Regelformen in kontextsensitiven Grammatiken haben daher folgendes Aussehen:

linksrekursive Regeln	$A \rightarrow Aa$
rechtsrekursive Regeln	$B \rightarrow bB$
zentralrekursive Regeln	$A \rightarrow aAb$
kontextsensitive Regeln	$aA \rightarrow Aa$, $Aa \rightarrow Aa$, $aAb \rightarrow Aa$

Ein Regelapparat, der die oben angeführte Sprache mit der allgemeinen Wortform "$a^n b^n c^n$" analysieren bzw. generieren kann, hat auf der Basis der terminalen und nicht-terminalen Symbole VT und VN die Form P:

VT = {a, b, c,} VN = {A, B, S}
P: $S \rightarrow aB$; $S \rightarrow aSA$ $B \rightarrow bc$; $cA \rightarrow Ac$
$BA \rightarrow bBc$

Mit Hilfe dieses Regelapparats kann die oben angegebene Sprache analysiert oder deren Worte generiert werden. Beispiel:
$S \rightarrow aSA \rightarrow aaSAA \rightarrow aaaBAA \rightarrow aaabBcA \rightarrow aaabBAc \rightarrow aaabbbBcc \rightarrow aaabbbccc \rightarrow aaabbBcc \rightarrow aaabBAc \rightarrow aaabBcA \rightarrow aaaBAA \rightarrow aaSAA \rightarrow aSA \rightarrow S$

Es lässt sich zeigen, dass für jede kontextsensitive Grammatik eine "Turing-Maschine" im Sinne eines linear beschränkten Automaten existiert, so dass gilt: $L(G) = L(LA)$;
denn der linear beschränkte Automat geht so vor, dass er zunächst durch elementeweises Löschen von links nach rechts die Anzahl der a- und b-Elemente miteinander vergleicht.
Folgt auf das zuletzt gelöschte b-Element ein c-Element und ist dann ebenso kein a-Element mehr vorhanden, dann stimmt die Zahl der a- und b-Elemente überein.
Dasselbe Verfahren wird nach Wiederauffüllen der a- und b-Elemente dann auf die b-und c-Elemente angewandt. Folgt schließlich auf das letzte c-Element ein Leerzeichen und ist dann auch kein b-Element mehr vorhanden, dann ist gezeigt, dass die Anzahlen aller drei Elemente übereinstimmen.
Das analysierte Wort gehört damit zu dieser Sprache L(G).

Differenzierung in linear beschränkten Automaten und Turing-Maschine

Der linear beschränkte Automat unterscheidet sich von einer Turing-Maschine darin, dass er ein zweiseitig beschränktes Band besitzt. Die Turing-Maschine besitzt hingegen nur ein einseitig beschränktes Band.

Endlicher Automat ohne Ausgabe

Ein endlicher Automat ohne Ausgabe wird durch drei Mengen definiert: durch das Eingabealphabet E, seine Zustände S und die Überführungsfunktion.
Das Eingabealphabet umfasst diejenigen Zeichen, die der Automat liest und verarbeitet.
Die Menge der Zustände S umfasst diejenigen Zustände, die der Automat im Laufe seiner Verarbeitung einnimmt. Zu dieser Menge gehören auch der Anfangszustand und die Menge der Endzustände. Das Programm eines solchen Automaten besteht aus dreiteiligen Regeln mit folgender Form:

s_i, a_i, s_j

Der Parameter s_i bezeichnet die Regelnummer. Dabei bilden alle Regeln mit derselben Ordnungszahl i zusammen einen Maschinenzustand.
Der Parameter a_i stellt die Aktion dar. Er stimmt im aktuellen Zustand s_i mit dem Eingabealphabetszeichen auf dem Leseband überein.
Der Parameter s_j steht für eine neue Regel, zu der von der aktuellen Regel aus verzweigt werden soll. Er steht also für eine Verzweigungsregel.

Die Zeichenverarbeitung eines endlichen Automaten ohne Ausgabe erfolgt in folgender Weise:
Bevor die neue Regel s_j ermittelt werden kann, wird das Leseband um ein Feld nach links bewegt. Dann wird vom Leseband das neue Zeichen a_i gelesen und zusammen mit der Nummer s_j der aktuellen Regel eine neue Regel gesucht.
Wird eine solche Regel aus der Kombination von s_j und a_i gefunden, wird diese Regel die neue aktuelle Regel, während das Leseband erneut um ein Feld nach links bewegt wird usw.
Mit der Verzweigungsregel s_j der nun aktuellen Regel beginnt dieser Verarbeitungszyklus von Neuem. Der endliche Automat muss also in seinem Regelapparat P von Verarbeitungsschritt zu Verarbeitungsschritt eine Regel finden, die in den ersten beiden Parametern eine Kombination aus Verweisungsregel der bisher aktuellen Regel und dem neu eingelesenen Eingabealphabetszeichen darstellt.
Ist diese Regel vorhanden, kann der Automat zu dieser neuen Regel übergehen und seinen Verarbeitungsprozess fortsetzen; wenn nicht, bricht der Automat die Bearbeitung des Lesebandes vorzeitig ab.
Bei Abbruch des Verarbeitungsprozesses hat sich dann gezeigt, dass die auf dem Leseband niedergeschriebene Zeichenfolge nicht zu der im Regelapparat des Automaten beschriebenen Grammatik gehört.

Endlicher Automat ohne Ausgabe

Ein endlicher Automat ohne Ausgabe wird durch drei Mengen definiert: durch das Eingabealphabet E, seine Zustände S und die Überführungsfunktion.
Das Eingabealphabet umfasst diejenigen Zeichen, die der Automat liest und verarbeitet.
Die Menge der Zustände S umfasst diejenigen Zustände, die der Automat im Laufe seiner Verarbeitung einnimmt. Zu dieser Menge gehören auch der Anfangszustand und die Menge der Endzustände. Das Programm eines solchen Automaten besteht aus dreiteiligen Regeln mit folgender Form:

s_i, a_i, s_j

Der Parameter s_i bezeichnet die Regelnummer. Dabei bilden alle Regeln mit derselben Ordnungszahl i zusammen einen Maschinenzustand.
Der Parameter a_i stellt die Aktion dar. Er stimmt im aktuellen Zustand s_i mit dem Eingabealphabetszeichen auf dem Leseband überein.
Der Parameter s_j steht für eine neue Regel, zu der von der aktuellen Regel aus verzweigt werden soll. Er steht also für eine Verzweigungsregel.

Die Zeichenverarbeitung eines endlichen Automaten ohne Ausgabe erfolgt in folgender Weise:
Bevor die neue Regel s_j ermittelt werden kann, wird das Leseband um ein Feld nach links bewegt. Dann wird vom Leseband das neue Zeichen a_i gelesen und zusammen mit der Nummer s_j der aktuellen Regel eine neue Regel gesucht.
Wird eine solche Regel aus der Kombination von s_j und a_i gefunden, wird diese Regel die neue aktuelle Regel, während das Leseband erneut um ein Feld nach links bewegt wird usw.
Mit der Verzweigungsregel s_j der nun aktuellen Regel beginnt dieser Verarbeitungszyklus von Neuem. Der endliche Automat muss also in seinem Regelapparat P von Verarbeitungsschritt zu Verarbeitungsschritt eine Regel finden, die in den ersten beiden Parametern eine Kombination aus Verweisungsregel der bisher aktuellen Regel und dem neu eingelesenen Eingabealphabetszeichen darstellt.
Ist diese Regel vorhanden, kann der Automat zu dieser neuen Regel übergehen und seinen Verarbeitungsprozess fortsetzen; wenn nicht, bricht der Automat die Bearbeitung des Lesebandes vorzeitig ab.

Bei Abbruch des Verarbeitungsprozesses hat sich dann gezeigt, dass die auf dem Leseband niedergeschriebene Zeichenfolge nicht zu der im Regelapparat des Automaten beschriebenen Grammatik gehört.

Endlicher Automaten ohne Ausgabe-Beispiel

Für einen endlichen Automaten ohne Ausgabe kann auf der Regelform der vorhergehenden Seiten basierend folgendes kleines Programm angegeben werden:

```
000,1,000   ;  PROGRAMM: Test auf 1er-Gruppen, die max. durch eine Null getrennt sind
000,0,010   ;
010,1,000   ;  "010" ist elementare if-Anweisung: wenn 1, dann "000"
010,s,030   ;  wenn 0, dann "030"
030,s,030   ;  "s" ist Stopp-Zeichen
```

Der Text nach dem Semikolon stellt dabei einen Kommentar dar, der beim Einlesen in den Programmeditor nicht berücksichtigt wird.
Die Aufgabe des Programms besteht darin, Gruppen von Eins-Elementen danach zu testen, ob sie nur durch ein Leer-Element (hier die Null) getrennt sind.
Die Bandinschrift vor Verarbeitungsbeginn kann damit folgendes Aussehen aufweisen:

111110111111111110110111110s

Das "s" symbolisiert den Stop-Zustand. Vor Arbeitsbeginn steht der Lesekopf über der ersten linken Eins. Mit sukzessiver Kopfbewegung nach rechts wird Regel um Regel des Programms abgearbeitet.
Gelingt die Verarbeitung im Ganzen, so wird der gesamte Prozess im Zustand "s" abgeschlossen. Damit gehört dieses Wort zu der durch den Regelapparat symbolisierten Sprache.

Endlicher Automat mit Ausgabe

Zwischen den endlichen Automaten mit und ohne Ausgabe bestehen wesentliche Unterschiede. Der wichtigste Unterschied ist, dass der endliche Automat mit Ausgabe zusätzlich über eine Ausgabeeinheit verfügt.
Die Ausgabeeinheit verfügt über ein Ausgabeband, auf dem in Abhängigkeit von Automatenzustand und eingelesenem Zeichen ein Ausgabezeichen feldweise eingetragen wird.
Das Ausgabeband ist identisch mit dem Leseband, nur dass über diesem statt einem Lese- ein Schreibkopf (S-Kopf) fest angebracht ist. Auch dieses Ausgabeband kann nur von rechts nach links bewegt werden, so dass die Ausgabezeichen auf ihm von links nach rechts eingetragen werden.

Aufgrund des zusätzlichen Bandes ist der endliche Automat mit Ausgabe durch folgende Mengen definiert:
Diese sind neben dem Eingabealphabet E und der Menge der Zustände S (einschließlich der Anfangs- und Endzustände) das Ausgabealphabet Z.
Zusätzlich gibt es bei diesem Automaten neben der Überführungsfunktion u die Ausgabefunktion g.
Ebenso wie die Funktion u kann die Funktion g als Cartesisches Produkt dargestellt werden: Sie gibt an, aufgrund welcher spezifischen kartesischen Produktbildung der Automat eine Ausgabe erzeugt.

Die Ausgabefunktion wird nicht näher festgelegt, da es verschiedene Definitionen gibt. Je nach Festlegung spricht man von einem Mealy-, einem Moore- oder einem trivialen bzw. Medwedew-Automaten. Gemäß dieser Reihenfolge haben diese drei endlichen Automaten mit Ausgabe folgende zu unterscheidenden Ausgabefunktionen:

$g: E \times S \rightarrow Z$ $g: S \rightarrow Z$ $g: E \rightarrow Z$

Damit kann der endliche Automat mit Ausgabe durch die folgende Menge M beschrieben werden:

$M = (E, S, Z, u, g, sa)$.

Dabei ist E das Eingabealphabet, S die Menge der Zustände, Z das Ausgabealphabet und sa der Anfangszustand, während u die bereits bekannte Überführungsfunktion und g die Ausgabefunktion darstellt. Damit besitzt der endliche Automat mit Ausgabe eine vierteilige Regelform:

$s_i,\ a_h,\ z_k,\ s_j$

Die Regelnummer ist s_i, die Verzweigungsnummer s_j. Die Variable a_h stellt das Einlesezeichen dar, z_k das Ausgabezeichen, das von der Maschine im Zustand s_i und beim Einlesen von a_k ausgegeben wird.

Beim endlichen Automaten mit Ausgabe steht im Zentrum des Interesses die Frage, welches Wort nach einem erfolgreichen Programmlauf auf dem Schreibband ausgegeben wird. Dabei korrespondiert die Grammatik der Eingabesprache einer anderen Grammatik als Ausgabesprache.
Es wird ein Wort der Worteingabe-Grammatik genau einem anderen der Ausgabe-Grammatik zugeordnet. Gehört das Eingabewort nicht zur Grammatik der Eingabesprache, so kann ihm auch keines der Ausgabe-Sprache zugeordnet werden. Das heißt beide Sprachen stehen in einem maschinellen Übersetzungsverhältnis zueinander.

Beispielprogramm

Dieses sehr einfache Programm für den endlichen Automaten mit Ausgabe basiert auf der zu diesem Automaten eingeführten Regelform:

000,1,0,000 ; PROGRAMM: Umkehrung der Ein-/Ausgabe
000,0,1,000
000,s,s,000

Es hat zur Aufgabe, Einer- und Null-Elementgruppen des Eingabebandes in Null- und Einer-Elementgruppen zu verkehren, so dass zwischen Eingabe- und Ausgabeband ein komplementäres Wortverhältnis bezüglich der Symbole Eins und Null besteht.
Zu Arbeitsbeginn hat das Leseband zum Beispiel folgende Inschrift:

11111000111100111000 0s

Nach der Verarbeitung dieser Bandinschrift durch das Programm steht auf dem Schreibband das komplementäre Wort:

00000111000011000111 1s

Damit wird die Verarbeitung im Zustand s, dem Stop-Fall, beendet.

Keller-Automat

Die endlichen Automaten mit und ohne Ausgabe können nur relativ zu einem bestimmten Maschinenzustand einen Zeichenvergleich zwischen Maschinenzustand und eingelesenem Zeichen durchführen.
Da sie nur über die Merkfähigkeit des unmittelbar vom Leseband gelesenen Zeichens verfügen, sind sie nicht in der Lage, ein Zeichen für einen späteren Verarbeitungsschritt aufzubewahren.

Der Kellerautomat besitzt einen sogenannten Keller. Er hat, wie der endliche Automat mit Ausgabe, zwei Bänder: ein Leseband mit einem darüber angebrachten Lesekopf (L-Kopf), das feldweise nur von links nach rechts bewegt werden kann, und ein zweites Band, der sogenannte Keller, der von ihm als Speicherband, d.h. als Lese- und Schreibband, verwendet werden kann. Über ihm ist ein Schreib-Lese-Kopf (SL-Kopf) angebracht. Das Kellerband hat ein unteres Ende, während es nach oben offen ist. Es kann in beiden Richtungen bewegt werden.

Das Kellerband funktioniert wie ein Stack. Der Automat kann immer nur ganz oben auf dem Kellerband ein Zeichen einfügen, und er kann auch immer nur das oberste der abgekellerten Zeichen lesen und löschen, niemals aber ein darunterliegendes.
Ein Kellerautomat kann damit durch die folgende Menge KA beschrieben werden:

$$KA = (E, S, K, s_a, k_a, u, S^*)$$

Dabei ist E das Eingabealphabet, S die Menge der Zustände, K das Kelleralphabet, s_a der Anfangszustand, k_a das Kellerstartzeichen, S die Menge der Endzustände und u die Überführungsfunktion.
Es ist dabei zu beachten, dass das leere Zeichen k ein Element von E ist und damit auch nicht in einem Eingabewort vorkommen kann. Dagegen gehört das leere Zeichen zur Menge K. Der SL-Kopf kann es auf das Speicherband schreiben, um damit ein über das Leseband eingegebenes und abgespeichertes Zeichen wieder zu löschen.
D.h., es gibt kein leeres Zeichen, sondern nur eine Kellerbandbewegung ohne Zeichenverarbeitung bzw. das Löschen eines Zeichens des Kelleralphabets mit einer Bandbewegung.

Die Überführungsfunktion des Kellerautomaten hat die Form: $u: (E \cup |\langle|) \times S \times K \rightarrow S \times K^*$
Es wird ein geordnetes Tripel (e_i, s_j, k_u), bestehend aus einem Zeichen des Eingabebandes, dem aktuellen Zustand des Automaten und einem auf dem Kellerband unter dem SL-Kopf eingetragenen Zeichen, in das geordnete Paar (s_n, t) überführt. Dabei ist s_n der neue Zustand und t als Element von K* ein Wort, das durch eine Verkettung des letzten Eingabezeichens mit dem zuletzt abgekellerten Zeichen gebildet und durch das oberste Kellerzeichen ersetzt wird.

Zu Beginn der Abarbeitung befindet sich der Kellerautomat im Zustand s_a, der Lesekopf steht über dem linken Zeichen des Lesebandes und der SL-Kopf über dem untersten Zeichen k_a des Kellers.
Steht bei der Bearbeitung eines Eingabewortes W unter dem Lesekopf das Zeichen e_j, unter dem SL-Kopf das Zeichen k_u und befindet sich der Kellerautomat gerade im Zustand s_j, dann kann es zu folgenden Vorgängen kommen:

1) falls es in u kein Tripel (e_i,s_j,k_u) oder (í, s_j, k_u) gibt, bleibt der Kellerautomat stehen;
2) falls es in u eine Zuordnung $(e_i, s_j, k_u) \rightarrow (s_n, k_t)$ gibt, dann geht der Kellerautomat in den Zustand s_n über, k_u wird durch k_t ersetzt und das Eingabeband um ein Feld nach links bewegt; dann steht der SL-Kopf über dem letzten Zeichen von k_t;

3) falls die obige Zuordnung statt e_i das leere Zeichen í bzw. kein Zeichen enthält, erfolgen dieselben Vorgänge wie zuvor, außer dass das Eingabeband nicht bewegt wird;

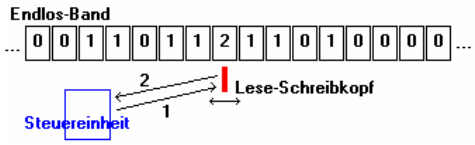
4) falls auf der rechten Seite einer Zuordnung das Paar (sn,í) steht, so geht der Kellerautomat in den Zustand s_n über und schreibt das Zeichen í auf das Kellerband; er löscht das dort stehende Zeichen k_u und bewegt danach das Kellerband um ein Feld nach oben;

5) falls s_n ein Element aus S* ist und das Speicherband leer ist, bleibt der Kellerautomat stehen.

Ein Eingabewort W ∈ E*; d.h. "W ist ELEMENT von E*"; aus der Sprache L(KA) gehört demnach dann zu einem Kellerautomaten KA, wenn dieser beginnend auf dem linken Zeichen von W und auf dem untersten Feld des leeren Speicherbandes eine Reihe von Zuständen durchläuft und schließlich auf dem am weitesten rechts stehenden Zeichen des Wortes W stehen bleibt, dabei der Kellerspeicher leer ist und KA einen Endzustand aus der Menge S* erreicht hat.

Turing-Maschine

David Hilbert, ein hervorragender Mathematiker, präsentierte nach der Jahrhundertwende eine Liste ungelöster mathematischer Probleme der damaligen Zeit:
1.Ist die Mathematik vollständig in einem technischen Sinn, dass nämlich jede ihrer Aussagen zumindest im Prinzip entweder bewiesen oder widerlegt werden kann?
2.Ist die Mathematik widerspruchsfrei, oder können falsche Aussagen mittels logischer Beweisführung hergeleitet werden?
3.Ist die Mathematik entscheidbar, d.h., gibt es ein Verfahren, das für jede Aussage deren Wahrheit bzw. Falschheit feststellt?
Kurt Gödel konnte zeigen, dass sich die beiden ersten Aussagen gegenseitig ausschließen:
Entweder ist das Axiomensystem der Mathematik hinreichend vollständig, dann führt es unvermeidlich zu Widersprüchen, oder man hält es widerspruchsfrei, dann deckt es aber nicht alle Aussagen ab, es bleibt unvollständig. Erst 1936 konnte Alan M. Turing (1912 - 1954) das dritte Problem lösen: es ist unlösbar. Die Bedeutung dieser Antwort geht über die Lösung dieses speziellen Problemes weit hinaus, denn sie machte die exakte Definition des Begriffes "Verfahren" oder "Algorithmus" erforderlich. Nach Turings Vorstellung muss ein Verfahren rein mechanisch, ohne höhere Einsicht, ausgeführt werden können.



Eine Turing-Maschine besteht aus ...
1. einem unendlich langen Band. Dieses Band besteht aus Zellen. Jede Zelle speichert genau ein Zeichen. Diese Zeichen entstammen einem Bandalphabet A,
2. einem Lese-Schreib-Kopf. Er kann vom Band jeweils genau ein Zeichen lesen bzw. auf das Band schreiben. In jedem Takt bewegt er sich genau eine Zelle nach links oder rechts bzw. bleibt an der gleichen Stelle stehen. Die Menge der Bewegungen ist damit {L,R,S}.
3. einem Leitwerk. Dieses Leitwerk steuert die Arbeit des Lese-Schreibkopfes. Im Leitwerk ist das Programm (die "Turing-Tafel") gespeichert. Das Leitwerk befindet sich ständig im aktuellen Zustand und kann diesen entsprechend dem Programm ändern. Die Menge aller Zustände sei Z.

Maschinenlauf

Zu Beginn befindet sich das Leitwerk im Zustand Z0. Folgende Schrittfolge steuert die Maschine:
1. Lesen des Zeichens auf dem Band, z.B. 1
2. Lesen der Turing-Tafel für Zustand Z1, z.B. [0,0,S | 0,1,R | 2,0,S | 3,0,S]
3. Bestimmung des Befehls entsprechend des Zeichens, im Beispiel neues Zeichen = 0, neuer Zustand = Z1, Bewegung nach R
4. Eintragen des neuen Zeichens, im Beispiel 0
5. Ausführung der Bewegung, im Beispiel eine Zelle nach rechts
6. Beginn bei 1. mit neuem Zustand, im Beispiel Z1
Die Maschine stopt im Zustand 0

Turing-Tafel (ggT-Bestimmung)

Zeichen	0	1	2	3	Zeichen	0	1	2	3
Zustand Z0	0,0,S	0,1,R	2,0,S	3,0,S	Zustand Z1	1,2,R	1,1,R	2,1,S	3,1,S
Zustand Z2	0,5,L	2,3,L	2,2,R	3,2,R	Zustand Z3	0,4,R	3,2,R	2,3,L	3,3,L
Zustand Z4	0,2,L	1,2,L	1,4,R	0,4,R	Zustand Z5	0,6,R	1,2,R	0,5,L	1,5,L
Zustand Z6	0,0,S	1,6,R	2,6,S	3,6,S					

Die nachfolgende Turing-Maschine hat das Ziel so viele wie möglich "Einsen" zu produzieren.

Turing-Tafel	0,1,S	1,0,S	1,2,R	1,3,L
	0,1,L	0,4,L	1,1,L	1,0,L
	1,2,L	1,5,R	0,4,R	0,2,R
Startinschrift	Keine	ergibt 1915 Einsen	2133493 Iterationen	
	>000111	ergibt 112 Einsen	7585	
	>0001111	ergibt 2184 Einsen	2779919	
	>0001111111	ergibt 890 Einsen	461646	

Turing-Maschine-Beispiel 2

Das Beispiel ist ein besonders für den linear-beschränkten Automaten geeignetes Programm, das zur Zeichensuche dient.
Die Verarbeitung beginnt irgendwo zwischen zwei Einsen auf dem Schreib-Lese-Band, um sukzessive die Position der nächstliegenden ersten Eins zu finden.
Das Band hat z.B. vor Arbeitsbeginn folgendes Aussehen:

```
0000000000000000100000000000000000000000100000000
```

Das zugehörige Programm besitzt folgenden Regelapparat

```
000,0,1,010     ; PROGRAMM: Suche der nächsten rechts oder links stehenden Eins
010,1,l,020
020,0,l,030
030,0,1,040
040,1,r,050
050,0,r,050
050,1,0,060
060,0,r,070
070,1,s,200     ; Ende rechts
070,0,1,100
100,1,l,110
110,0,l,110
110,1,0,120
120,0,l,130
130,1,s,200     ; Ende links
130,0,1,040
```

Turing-Maschine-Addition

Addition zweier Einser-Zahlengruppen:

```
000,0,r,000  ;  PROGRAMM: Addition zweier Einsen-Zahlengruppen
000,1,r,010
010,1,r,010
010,0,r,020
020,1,r,020
020,0,l,040
040,1,0,040  ; Eins der zweiten Zahl in Null verwandelt
040,0,l,050
050,1,l,050
050,0,1,060  ; die Null zwischen den Zahlen in Eins verwandelt
060,1,l,070
070,1,l,070
070,0,s,070
```

Vor Arbeitsbeginn hat das Band zum Beispiel folgende Struktur, wobei der SL-Kopf irgendwo auf den links stehenden Nullen steht:

```
  0000000000011111111110111111000000000
```

Die Addition erfolgt nun so, dass die Null zwischen den Einsergruppen in eine Eins verwandelt wird, wobei zuvor noch die am weitesten rechts stehende Eins in Null verwandelt wurde.
Das Ergebnis in diesem Falle wäre dann die Zahl 15, wobei der SL-Kopf über der ersten Null nach der letzten rechten Eins stehen bleibt.

Turing-Maschine-Subtraktion

Subtraktion zweier Einser-Zahlengruppen:

```
000,*,l,001  ; PROGRAMM: Subtraktion zweier Einsen-Zahlengruppen
001,1,0,002   002,0,l,003   003,1,l,003
003,*,l,004   011,*,r,012   004,0,l,004
012,0,r,012   004,1,0,005   012,*,r,013
005,0,l,006   013,1,r,013   006,*,r,017
013,*,1,014   006,1,r,007   014,1,l,014
007,0,r,007   014,*,l,015   007,*,r,008
015,0,l,015   008,1,r,008   015,*,l,016
008,0,l,009   016,0,l,016   009,1,0,002
016,1,0,011   009,*,l,010   016,*,r,017
010,0,l,010   017,0,1,018   010,1,0,011
018,1,r,017   011,0,r,011   017,*,r,019
019,0,1,020   019,*,r,021   020,1,r,019
021,1,r,021   021,*,s,021
```

Hier fungiert als Leerzeichen der Stern "*". Also heißt vor Verarbeitungsbeginn die Bandinschrift für die Subtraktion "5 - 3" wie folgt:

```
              |
**************11111*111***************
```

Der SL-Kopf steht vor Arbeitsbeginn über dem ersten Stern nach der letzten Eins. Die TM-Maschine arbeitet nun so, dass sie nach dem gleichmäßigen Streichen der Einsen in den beiden Zahlengruppen die verbliebenen der linken Zahlengruppe im freien Sternfeld rechts außen einträgt. Danach werden die beiden Zahlengruppen wieder hergestellt, so dass sich folgende Bandinschrift als Ergebnis ergibt:

```
          |
****************11111*111*11****************
```

Turing-Maschine-Division

Das Programm dient der Modulo3-Division mit dem Faktor drei. Das Programm besitzt folgenden Regelapparat:

```
        000,*,l,000    ; PROGRAMM: Modulo-3-Division (mit Rest-Ausgabe)
        000,0,l,000      000,1,0,001 ; 1.Zahl auf Null
        006,*,1,007      000,*,r,020 ; Rest       007,1,l,007
        001,0,r,001      007,*,l,008      001,*,r,002
        008,*,l,008      002,*,1,003      008,0,l,009
        003,1,l,003      009,0,l,009      003,*,l,004
        009,1,0,010 ; 3.Zahl=0            004,0,l,004      009,*,r,020 ; Rest
        004,1,0,005 ; 2.Zahl=0            010,0,r,010      004,*,r,020 ; Rest
        010,*,r,011      005,0,r,005      011,1,r,011
        005,*,r,005      011,*,1,012      005,1,r,006
        012,1,r,013      006,1,r,006      013,*,l,014
        014,1,*,015      015,*,l,014      014,*,l,000 ; erneute Schleife
        020,0,1,021      020,*,r,022      021,1,r,020
        022,1,r,022      022,*,s,023
```

Leerzeichen ist auch hier der Stern "*". Im Anfangszustand sieht die Bandinschrift zum Beispiel folgendermaßen aus:

```
          |
*****************11111111****************
```

Diese TM-Maschine zur Modulo3-Division überträgt nun immer gestrichene Einsen in das freie rechte Feld. Bei der Anzahl drei werden diese wieder gestrichen, um von Neuem zu beginnen. Zum Schluss steht diejenige Anzahl von Einsen im rechten Feld, die Rest dieser Modulo-Division sind. Dann wird die Einsen-Zahlengruppe wieder aufgefüllt. Die Bandinschrift sieht dann folgendermaßen aus:

```
           |
***************11111111*11*****************
```

Turing-Maschine, der linear beschränkte Automat

Ein linear beschränkter Automat unterscheidet sich von einer Turing-Maschine nur durch ein zweiseitig begrenztes Band.
Die Turing-Maschine verfügt nur noch über ein Band, das zugleich als Schreib- und Leseband (SL-Band) verwendet wird. Während das Band der Turing-Maschine auf einer Seite begrenzt ist, ist es auf der anderen Seite beliebig lang. Es kann von der Turing-Maschine beliebig weit nach rechts und wieder nach links bewegt werden.
Über dem Band ist ein Schreib-Lese-Kopf (SL-Kopf) angebracht, durch den der Automat Zeichen in jedes in endlicher Zeit erreichbare Feld setzen, löschen oder durch Gehen zum benachbarten Feld nach rechts oder links übergehen kann.

Damit kann die Turing-Maschine einen bestimmten Bereich seines Bandes als Speicherabschnitt verwenden. Die Turing-Maschine ist nicht darauf angewiesen, immer nur das oberste abgekellerte Zeichen vom Speicher zu holen.
Vielmehr können beliebige Zeichen vom Band gelesen und gelöscht werden. Damit besitzt die Turing-Maschine gegenüber dem Kellerautomaten ein Speicherband, auf dem jederzeit an beliebiger Stelle notierte Zeichen zugänglich sind und beliebig Zeichen gelöscht und gesetzt werden können.
Eine Turing-Maschine T ist durch folgende Menge bestimmt: T = (E, S,B, s_a, ä, u)
Dabei bezeichnet E die Menge der Bandzeichen, S die Menge der internen Zustände der Maschine, B die Menge der Kopfbewegungen über dem Schreib-Lese-Band (SL-Band), s_a den Anfangszustand, ä die Menge der Endzustände und u die Überführungsfunktion.
Die Überführungsfunktion hat dabei die folgende Form: u: $(e_i, s_j) \rightarrow (e_p, s_q, b_r)$

Diese Funktion sagt aus, dass die Maschine aufgrund eines bestimmten Zustands und eines bestimmten gelesenen Bandzeichens in einen neuen Zustand übergeht, indem $b_r \in B$ durchgeführt wird, d.h. es wird eine neue Bandbeschriftung an der betreffenden Stelle erzeugt oder der Schreib-Lese-Kopf um eine Feldstelle nach rechts oder links bewegt.
In diesem neuen Feld liegt dann die Eingabe e_p im Zustand s_q vor. Wenn die Turing-Maschine denn Zustand $s_r \in$ ä erreicht, bleibt sie stehen.

Brainfuck, BF

Brainfuck (kurz: BF) ist eine sogenannte esoterische Programmiersprache, die der Schweizer Urban Müller im Jahre 1993 entwarf. Eine esoterische Programmiersprache soll ungewöhnliche Sprachkonzepte darstellen und hat nichts mit Esoterik zu tun.

BF ist geeignet, um Grundlagen der Computertechnik zu erlernen und besitzt ein extrem einfaches Sprachkonzept. Müllers konstruierte eine einfache Turing-vollständige Sprache, welche mit einem möglichst kleinen Compiler (auf dem Amiga 240 Byte) übersetzt werden kann.
Der Brainfuck-Compiler für x86-Linux von Brian Raiter ist nur 171 Byte groß; für MS-DOS erreichte Bertram Felgenhauer nur 98 Byte.
BF besitzt acht Befehle, jeweils bestehend aus einem einzigen Zeichen:

Zeichen	C-Äquivalent	Erklärung
>	++ptr;	inkrementiert den Zeiger
<	--ptr;	dekrementiert den Zeiger
+	++*ptr;	inkrementiert den aktuellen Zellenwert
-	--*ptr;	dekrementiert den aktuellen Zellenwert
.	putchar(*ptr);	gibt den aktuellen Zellenwert als ASCII-Zeichen auf der Standardausgabe aus
,	*ptr = getchar();	liest ein Zeichen von der Standardeingabe und speichert dessen ASCII-Wert in der aktuellen Zelle
[	while (*ptr) {	springt nach vorne, hinter den passenden] -Befehl, wenn der aktuelle Zellenwert 0 ist
]	}	springt zurück, hinter den passenden [-Befehl, wenn der aktuelle Zellenwert nicht 0 ist

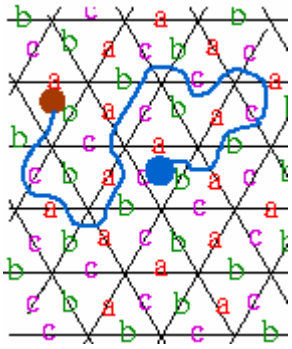
Andere im Quelltext vorkommende Zeichen werden ignoriert.
Für verschiedene BF-Umgebungen wurde Turing-Vollständigkeit bewiesen.
Bei einer BF-Variante mit endlicher Zellgröße sowie endlicher Feldgröße handelt es sich, wie bei jedem realen Computer, um einen endlichen Automaten.

Berechenbare Zahl

Zu den berechenbaren Zahlen werden die Zahlen gezählt, bei denen alle Dezimalstellen mit einer Berechnungsvorschrift erzeugt werden können.
Überraschend ist es, dass es nicht berechenbare Zahlen gibt. Mit der Church-Turing-These kann man für den Begriff der Berechnungsvorschrift eine Turingmaschine wählen.

Definition

Eine reelle Zahl r heißt genau dann berechenbar, wenn es eine Turing-Maschine gibt, die für jedes $\varepsilon > 0$ eine Zahl q ausgibt, so dass $|r - q| < \varepsilon$
gilt, die also die Zahl r beliebig genau annähern kann.
Alle natürlichen, rationalen und algebraischen Zahlen sind berechenbar, aber auch einige transzendente Zahlen, z.B. die Kreiszahl π oder die Eulersche Zahl e.



Alphabet und kürzeste Wege

Gegeben sei ein Alphabet V = {a, b, c} auf dem Worte gebildet werden sollen. Gleichzeitig werden die zugelassenen Worte durch eine Dreiecksparkettierung gekennzeichnet. Ein Startpunkt werde durch einen roten Punkt markiert, der Endpunkt durch einen blauen Punkt. Jede Seite der Dreiecke ist durch einen Buchstaben des Alphabets gekennzeichnet. Überschreitet ein Weg von Rot nach Blau eine Dreiecksseite, so wird dieser Buchstabe dem Wort hinzugefügt.

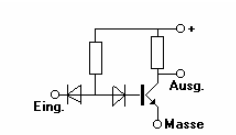
In der Darstellung würde sich das Wort cbcababacabcbccacacbab ergeben.
Zulässig ist ein Wort in dieser Sprache L, wenn es einen kürzesten Weg von Rot nach Blau beschreibt.
Das Beispielwort gehört damit nicht zur Sprache, jedoch "baca", das den gleichen Weg beschreibt.

Um aus einer beliebigen Buchstabenfolge ein zulässiges Wort zu konstruieren, müssen Transformationsregeln angewandt werden. Ist ø das leere Wort, so besteht das Regelsystem aus

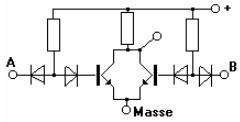
aa → ø	bb → ø	cc → ø
ababab → ø	acacac → ø	bababa → ø
bcbcbc → ø	cacaca → ø	cbcbcb → ø
abab → ba	acac → ca	baba → ab
bcbc → cb	caca → ac	cbcb → bc

Logische Schaltungen

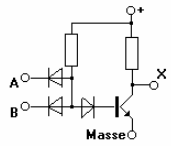
Negation-Schaltung
Das eingehende Signal wird negiert.

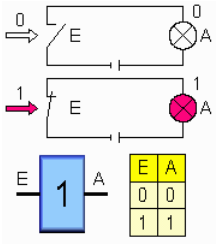


NAND - Schaltung
Bei X liegt kein Signal an, wenn an A und an B ein Signal anliegt.

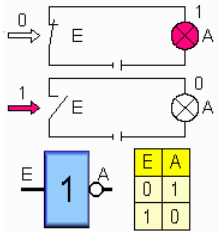


NOR-Schaltung
An X liegt nur dann ein Signal an, wenn sowohl an A als auch an B kein Signal anliegt.

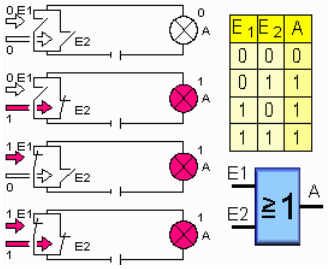




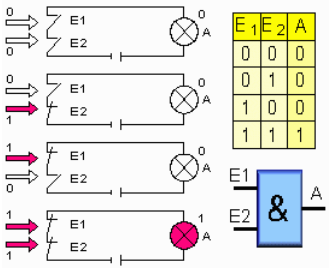
Identitätsschaltung
Am Ausgang liegt 1, wenn am Eingang 1 liegt.



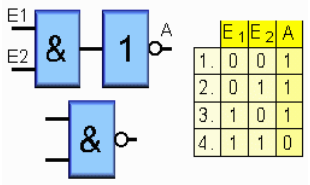
NOT-Schaltung
Am Ausgang liegt 1, wenn am Eingang nicht 1 liegt.



OR-Schaltung
Am Ausgang liegt 1, wenn an einem oder mehreren Eingängen 1 liegt.

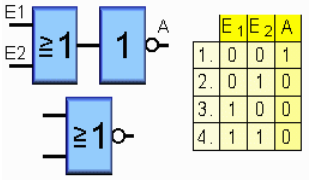


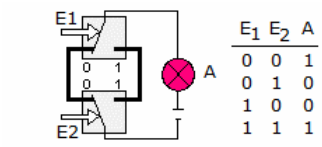
AND-Schaltung
Am Ausgang liegt 1, wenn an E1 und E2 1 liegt, also an allen Eingängen 1 liegt



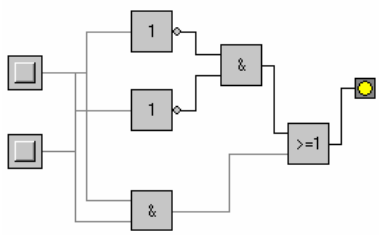
NAND-Schaltung
Der Ausgang zeigt 1, wenn nicht beide Eingänge den Zustand 1 haben; die Negation der AND-Schaltung

NOR-Schaltung
Der Ausgang zeigt 1, wenn beide Eingänge den Zustand 0 haben; die Negation der OR-Schaltung

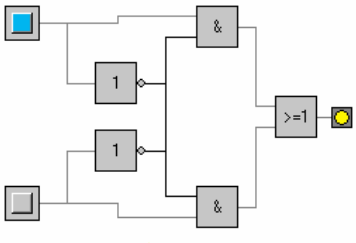




Äquivalenzschaltung
Bei einer Äquivalenzschaltung zeigt der Ausgang eine 1, wenn alle Eingänge den gleichen Zustand aufweisen, d.h. alle auf 0 oder 1 gesetzt sind.



Antivalenzschaltung, EXOR-Schaltung
Bei einer Antivalenzschaltung bzw. EXOR-Schaltung zeigt der Ausgang eine 1, wenn an einem und nur an einem Eingang der Wert 1 liegt.

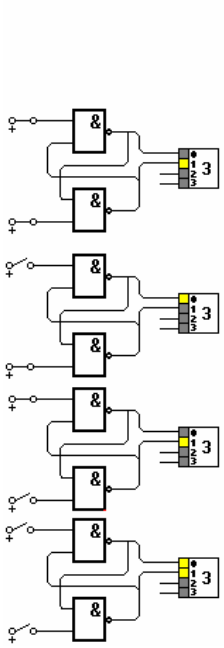


Das Verhalten der Antivalenzschaltung unterscheidet sich von dem einer OR-Schaltung. Wenn beide Eingänge auf 1 liegen, ist der Ausgang beim OR auf 1, bei der Antivalenz auf 0. Sie schließt also den Fall aus.
Man bezeichnet die Antivalenz daher als "Exclusiv-OR" oder "XOR".

E_1 E_2 A
0 0 0
0 1 1
1 0 1
1 1 0

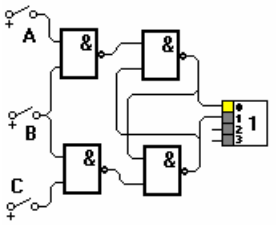
Grundflipflop

Eine wesentliche Aufgabe in der Digitaltechnik ist es, Informationen zu speichern. Beschränken wir uns dabei zunächst auf ein Bit, die kleinste Informationseinheit. Es sind also Schaltungen erforderlich, die ein Bit in irgendwelchen Zuständen speichern können. Diese Schaltungen werden Flipflops genannt.
Wir gehen aus von der folgenden Schaltung:
(Abbildung 1) Beide Eingänge liegen also auf H, einer der Ausgänge ebenso. Nun öffnen wir, u.U. auch nur kurzzeitig, den Eingang a:
(Abbildung 2) Wir sehen, dass nun Ausgang c auf H liegt. Öffnen wir nun kurzzeitig Eingang b, dann liegt Ausgang d auf H:
(Abbildung 3) Kurzzeitiges Öffnen von Eingang a bringt also Ausgang c auf H, kurzzeitiges Öffnen von Eingang b bringt Ausgang d auf H.
(Abbildung 4) Öffnen wir beide Eingänge, dann erhalten wir einen Ausgangszustand, der zur Informationsspeicherung unbrauchbar ist.
Durch geeignete Verschaltung muss daher immer gewährleistet sein, dass nie beide Eingänge gleichzeitig offen sind.



Wahrheitswertetabelle

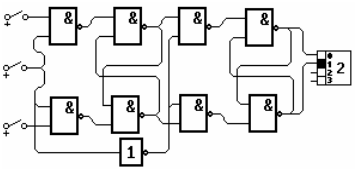
a	b	c	d
L	L	X	X
L	H	H	L
H	L	L	H
H	H	Z	Z



Dabei steht X für ein beliebiges, also undefiniertes Ausgangssignal, und Z für den vorhergehenden Zustand, der also erhalten blieb.

Auffangflipflop

Beim Grundflipflop wirkt sich eine Änderung an den Eingängen unmittelbar an den Ausgängen aus. Dies ist häufig unerwünscht. Eine Eingangsänderung soll sich erst dann am Ausgang auswirken, wenn ein spezielles Taktsignal aktiv ist.
Wir erweitern das Grundflipflop um zwei NANDs, die den Eingängen vorgeschaltet werden:
Der mittlere Schalter ist nun für das Taktsignal zuständig. Schließen wir jetzt Eingang C (durch die Negation in den vorgeschalteten NANDs ist die Logik an den Eingängen hier gerade umgekehrt wie am Grundflipflop), bewirkt dies noch keine Änderung am Ausgangszustand des Flipflops.
Erst wenn wir zusätzlich noch ein Taktsignal geben, schaltet das Flipflop an seinen Ausgängen um. Das Umschalten in den anderen Ausgangszustand des Flipflops erfolgt ebenso nur bei vorhandenem Taktsignal.
Sind beide Eingänge geschlossen und es liegt Taktsignal an, dann ist der Ausgangszustand undefiniert, diese Situation muss also schaltungstechnisch verhindert werden.



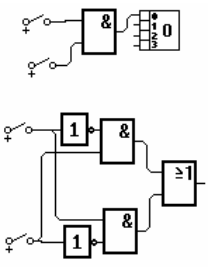
Master-Slave-Flipflop

Das Auffang-Flipflop wird nun noch einmal erweitert, denn in der Regel liegt das Taktsignal länger an als die Zeit, die gebraucht wird, bis eine Eingangsänderung an den Ausgängen erscheint.
Wir schalten zwei Auffang-Flipflop hintereinander und erhalten damit eine Entkoppelung der Eingänge und Ausgänge:
Das Taktsignal des Ausgangflipflops ist über ein NOT mit dem Taktsignal des Eingangflipflops verbunden. Wenn eines der Flipflops gerade in der Lage ist, eine Informationseinheit aufzunehmen, dann ist das andere Flipflop gerade blockiert und umgekehrt. Ist der Eingang A geschlossen, zeigt sich keine Änderung am Ausgang, da die an den Eingängen liegende Informationen momentan nur im Eingangsflipflop gespeichert werden kann, das Ausgangsflipflop ist blockiert und behält den alten Zustand bei. Erst wenn der Taktschalter geöffnet wird, schaltet der Ausgang um.
Die Länge des Taktsignals darf also nun beliebig sein (anders als beim Auffang-Flipflop), denn erst mit der fallenden Flanke des Taktsignals schalten die Ausgänge um.
Das Umklappen in den anderen Zustand erfolgt natürlich analog, erst mit der fallenden Flanke des Taktsignals, also dem Öffnen des Taktschalters, werden die Ausgänge umschalten. Sind beide Eingänge geschlossen und es liegt ein Taktsignal an: führt wieder zu einem undefinierten Zustand an den Ausgängen, weshalb diese Situation schaltungstechnisch ausgeschlossen sein muss.

Schieberegister

Die Hintereinanderschaltung von taktgesteuerten Flipflops heißt Schieberegister. Damit können Dualzahlen, die mehr als eine Stelle umfassen, gespeichert werden und taktgesteuert seriell ausgelesen werden.

Halbaddierer



Eine grundlegende Rechenoperation ist die Addition. Es soll im folgenden eine Schaltung entwickelt werden, die im Zweiersystem addiert.
Zunächst kann diese Schaltung eventuelle Überträge vorhergehender Stellen nicht verarbeiten, man spricht daher von einem Halbaddierer.

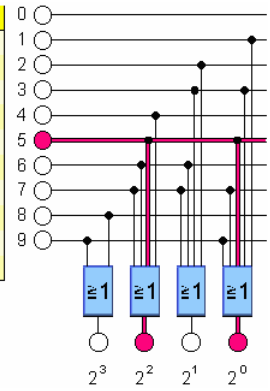
Bestimmung des Übertrags

Der Übertrag ist offenbar genau dann H, wenn beide Eingänge H sind. Dies ist eine AND-Verknüpfung.

Bestimmung der Summe

Die Summe ist genau dann H, wenn entweder a oder b auf H liegen. Es handelt sich hier also um ein ausschließendes Oder, im Fachjargon exklusives OR oder XOR. Dies ist vom einfachen OR, das bei den Grundgattern vorgestellt wurde, zu unterscheiden.

Taste	Ausgänge
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1



Dual-Dezimal-Codierer

Aufgabe

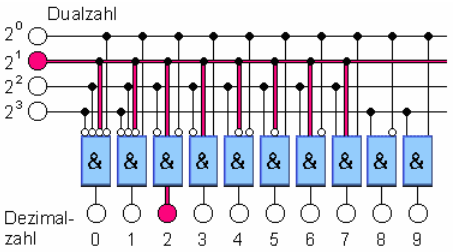
Umwandlung einer Dualzahl in eine Dezimalzahl

Prinzip

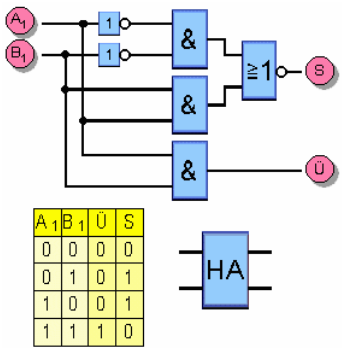
Vier Eingangsleitungen, welche die Dualzahl repräsentieren, sind derart mit je einer Ausgangsleitung zu verbinden, dass nur die zu der Dezimalzahl zugehörige Ausgangsleitung das Signal 1 erhält

Dezimal-Dual-Codierer

Aufgabe: Umwandlung einer einstelligen Dezimalzahl in eine Dualzahl



Prinzip: Jede einstellige Dezimalzahl lässt sich eindeutig in eine vierstellige Dualzahl umwandeln



A_1	B_1	Ü	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

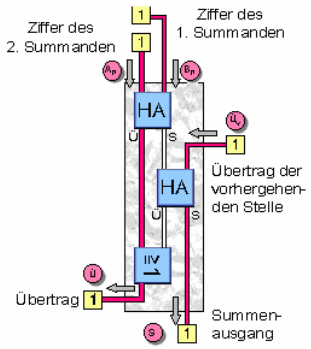
Halbadder

Aufgabe

Addition von 2 einstelligen Dualzahlen

Ergebnis

Mit einem Halbaddierer können 2 einstellige Dualzahlen addiert werden



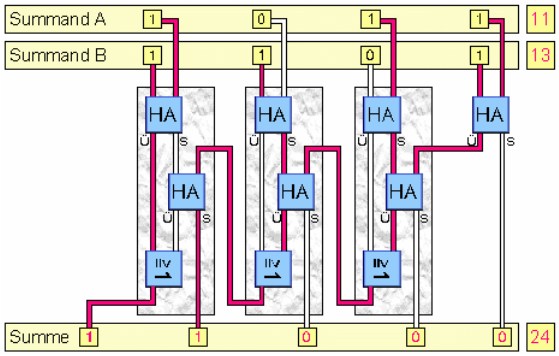
Volladder

Aufgabe

Addition von 2 einstelligen Dualzahlen mit Verarbeitung der Stellen mit dem Übertrag der vorhergehenden Stelle

Prinzip

Man benötigt einen ersten Halbaddierer, der die beiden Summandenziffern addiert, einen zweiten Halbaddierer, welcher für die Addition des Summenausgangs des ersten Halbaddierers mit dem Übertrag aus der letzten Ziffernaddition zuständig ist und eine OR-Schaltung, die für die Addition der beiden Übertragsausgänge der zwei Halbaddierer genügt, da hier kein weiterer Übertrag auftreten kann.



Jeder Volladdierer besitzt
3 Eingänge: Übertrag der letzte Zeile, Summand A; Summand B
2 Ausgänge: Summenausgang, Übertrag

Addierwerk

Aufgabe

Addition von zwei mehrstelligen Dualzahlen

Prinzip

Für jede Stelle wird die Stellensumme und der Übertrag zur nächsthöheren Stelle durch einen Volladdierer ermittelt. Eine Ausnahme bildet die letzte Stelle. Hier genügt ein Halbaddierer, da kein vorhergehender Übertrag auftreten kann.

P-adische Darstellung reeller Zahlen

Der Bruch 22/7 = 3,142857 142857,… entspricht in Potenzschreibweise der unendlichen Reihe

$$22/7 = 3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2} + 2 \cdot 10^{-3} + 8 \cdot 10^{-4} + \ldots$$

Dabei liegt das Dezimalsystem, also Basis B = 10, zugrunde. Statt B = 10 kann jede natürliche Zahl ≥ 2 als Basis verwendet werden. $B \in \aleph$, $B \geq 2$ und $Z_B = \{0, 1, 2, \ldots, B\text{-}1\}$ eine Menge von B Elementen, denen wir in der angeschriebenen Reihenfolge die Zahlenwerte 0,1,2,…,B-1 zuordnen. Dann konvergiert nach dem Majorantenkriterium jede Reihe der Form

$$b_n \cdot B^n + \ldots + b_0 \cdot B^0 + b_{-1} \cdot B^{-1} + b_{-2} \cdot B^{-2} + \ldots,\ n \in \aleph,\ b_k \in Z_B$$

für alle $n \leq k$, und ihre Summe ist eine positive reelle Zahl.

Komplementärzahl

Gegeben sei ein Zahlensystem zur Basis g mit den Ziffern {0, 1, 2, …, g-1}. Unter der Komplementaritätsbeziehung versteht man die Tatsache, dass es zu jeder Zahl a < g dann eine Komplementärzahl (g - a - 1) existiert.
Zum Beispiel hat die 6 im Dezimalsystem die Komplementärzahl 3. Im Binärsystem entspricht die Null der Komplementärzahl 1 und die 1 entspricht der Null. Allgemein ist die Komplentärzahl der Null stets die Zahl g-1, die Komplementärzahl von 1 entsprechend g-2.
Die vierstellige Zahl 10(g-2)(g-1)
wird Repräsentantin des Zahlensystems zur Basis g genannt. Die 1089 ist die Repräsentantin des Dezimalsystems, die 1001 die im Dualsystem. Jede andere vierstellige Zahl im System zur Basis g, bei der die Summen aus erster und vierter sowie zweiter und dritter Ziffer gerade g-1 ergeben, wird Mutuante genannt.
Im Dezimalsystem sind zum Beispiel 2178, 3267, 4356, … Mutuanten.

Einheiten der Datendarstellung

Bit ... kleinste Einheit der Datendarstellung, kann mögliche Werte 0 / 1, falsch / wahr, O / L, false / true annehmen
Byte ... Zusammenfassung von 8 bit zu einem Zeichen, maximal $2^8 = 256$ verschiedene Zeichen darstellbar
1 KByte = 2^{10} Byte = 1024 Byte 1 MByte = 2^{20} Byte = 1048576 Byte
Binary coded decimal (BCD) Standard ... jede Dezimalstelle einer Zahl wird für sich als 4-Bit-Dualzahl (Tetrade) codiert. Zur Codierung der Ziffern von 0 bis 9 werden nur 10 Tetraden benötigt, 6 Pseudotetraden (1010, 1011, 1100, 1101, 1110, 1111) werden nicht benötigt.
IEEE-Standard ... standardisierte Zahlendarstellung in Rechnern; ist in den meisten Programmiersprachen und auf vielen Computersystemen realisiert

Gleitkommaformat nach IEEE-Standard 754-1985 (DIN IEC 60559 1991)

Die Mantissen sind normalisiert, d.h. die Ziffer ganz links ist nicht Null.
1) einfache Genauigkeit (32 Bit) |1 | 8 | 23 Bit |
Vorzeichen Exponent Bruch
Beispiel: 10#13.75 = 2#1.10111*2^3
Die erste 1 von 1,10111 wird weggelassen und nur 10111 im 23-Bit-Feld gespeichert; zu dem Exponenten von 2 wird 127 addiert, das Ergebnis dann im 8-Bit-Feld gespeichert; Überschuß-127.
Formal (S = Vorzeichen, f = Nachkommastellen der normalisierten Mantisse, E = Exponent):

$$Z = (-1)^S \cdot (1.f) \cdot 2^{E\text{-}Bias}$$

2) doppelte Genauigkeit (64 Bit)
|1| 11 | 52 Bit |
Vorzeichen Exponent Bruch