Primzahltests und Faktorisierung

Seminararbeit

Simon Bailey, 9818119
Sarah Winkler, 0019379

simon.bailey@uibk.ac.at
csae8654@uibk.ac.at

30. November 2004

Betreuer: Dipl.-Math. Christan Vogt

Inhaltsverzeichnis

1

Theoretische Grundlagen

1.1 Primzahlen und Teilbarkeit

1.1.1 Definition der Primzahlen

Definition 1 (Teilbarkeit) *Man sagt, eine Zahl $a \in \mathbb{Z}$ ist durch eine Zahl $b \in \mathbb{Z}$ teilbar bzw. b teilt a – in Symbolen $b \mid a$ –, wenn eine Zahl $q \in \mathbb{Z}$ existiert, sodass $a = q \cdot b$.*

Definition 2 (Primzahl) *Eine Zahl $p \in \mathbb{Z}$, $p \geq 2$ wird prim genannt, wenn sie nur durch 1 und sich selbst teilbar ist.*

Definition 3 (Größter gemeinsamer Teiler) *Falls ein $m \geq 1$ existiert mit $m \mid a$ und $m \mid b$, und weiters für alle m' mit $m' \mid a$ und $m' \mid b$ gilt, dass $m' \leq m$, so wird m der größte gemeinsame Teiler von a und b genannt. Man schreibt dann $ggT(a, b) = m$. Zwei Zahlen $a, b \in \mathbb{Z}$ heißen relativ prim oder teilerfremd, wenn $ggT(a, b) = 1$.*

Der wohl bekannteste Algorithmus zur Berechnung des größten gemeinsamen Teilers ist der Euklidische Algorithmus. Er beruht auf der folgenden Beobachtung: $ggT(a, b) = ggT(b, a \operatorname{rem} b)$, denn falls z die Zahlen $a = k \cdot b + r$ und b teilt, so teilt z auch r. Der Euklidische Algorithmus wendet diese Tatsache wiederholt an, bis $b = 0$ gilt:

Algorithmus 1 (Euklidischer Algorithmus)
EINGABE: a, b mit $a \geq b$
`while` $(b \neq 0)$
  $(a, b) = (b, a \operatorname{rem} b)$
`return b;`

Der Euklidische Algorithmus benötigt $O(\log n \cdot \log m)$ Bitoperationen, um den größten gemeinsamen Teiler zweier Zahlen m und n zu berechnen.

Für die Konstruktion des Schlüssels wird beim RSA–Algorithmus der sogenannte Erweiterte Euklidische Algorithmus benötigt, der für zwei Zahlen a und b den größten gemeinsamen Teiler $g = ggT(a, b)$ und eine Darstellung $g = a \cdot x + b \cdot y$ zurückgibt.

1.1.2 Hauptsatz der Zahlentheorie

Unter einer Primfaktorzerlegung einer Zahl $n \in \mathbb{Z}$, $n > 0$ versteht man ein Produkt $p_1 \cdot p_2 \ldots p_k = n$, wobei $p_1, \ldots p_k$, $k \geq 0$, (nicht notwendigerweise verschiedene) Primzahlen sind.

Der Hauptsatz der Zahlentheorie besagt nun, dass jede ganze Zahl eine – bis auf Reihenfolge und Vorzeichen – eindeutige Primfaktorzerlegung hat, also $\mathbb{Z}$ ein so genannter *ZPE–Bereich* ist. Diese Aussage wirkt vielleicht wenig beeindruckend und wird oft als selbstverständlich angenommen. Doch ein kleiner Ausflug in einen künstlichen Zahlbereich K zeigt, dass die eindeutige Primfaktorzerlegung nicht selbstverständlich ist.

Sei $K = \{a + bj \mid a, b \in \mathbb{Z}\}$, wobei j eine etwas abgewandelte imaginäre Einheit sein soll, sodass

$j^2 = -3$ gilt. Außerdem definieren wir zwei Verknüpfungen auf K, eine Addition $+$ mit dem neutralen Element $0 + 0j$ und eine Multiplikation $*$ mit dem neutralen Element $1 + 0j$. Die Addition in K soll komponentenweise erfolgen, sodass $(a + bj) + (c + dj) = ((a + b) + j(c + d))$. Die Multiplikation wird als $(a + bj) * (c + dj) = (ac - 3bd) + (ad + bc)j$ definiert.
Wenn wir jetzt die Zahl $4 + 0j$ in irreduzible, d.h. bis auf Multiplikation mit (-1) nicht weiter zerlegbare Faktoren aufspalten wollen, haben wir die nahe liegende Möglichkeit $4 + 0j = (2+0j)*(2+0j)$. Andererseits ergibt aber auch $(1+j)*(1-j) = 1-j^2 = 1-(-3)+0j = 4+0j$. Wie sich leicht zeigen lässt, sind die Faktoren $(1+j)$,$(1-j)$ und $2+0j$ aber nur durch $1, -1$,sich selbst und ihr additiv Inverses teilbar. Also ist die Primfaktorzerlegung in K nicht eindeutig.

Für den Beweis des Hauptsatzes sind die folgenden beiden Beobachtungen nützlich:
Bemerkung
Jede Zahl $n \geq 2$ ist durch eine Primzahl teilbar.

Beweis: Man kann zwei Fälle unterscheiden: Entweder n ist eine Primzahl oder nicht. Falls n prim ist, gilt $n \mid n$ und die Aussage stimmt. Falls n keine Primzahl ist, existiert nach Definition der Primzahlen eine Zahl $n_1 < n$, die n teilt. Falls n_1 prim ist, ist die Aussage bewiesen. Andernfalls muss wiederum eine Zahl $n_2 < n_1$ existieren mit $n_2 \mid n_1$. So erhält man eine absteigende Folge von Teilern von n, die jedoch irgendwann stationär werden muss. Nach einer endlichen Anzahl von Schritten gelangt man zu einem Teiler n_t von n mit $2 \leq n_t < n$, der keinen echten Teiler mehr hat. Also ist n_t prim.

□

Bemerkung
Jede ganze Zahl $n \geq 1$ hat eine Primfaktorzerlegung.

Beweis: Der Beweis erfolgt durch Induktion über n. Falls $n = 1$, so ist das leere Produkt die Primfaktorzerlegung von n. Also kann angenommen werden, dass $n > 1$. Laut der vorigen Bemerkung ist n durch eine Primzahl p_1 teilbar. Wir setzen $n_1 = n/p_1$. Falls $n_1 = 1$, wurde eine Primfaktorzerlegung gefunden: $n = p_1$. Andernfalls wenden wir die Induktionsannahme auf n_1 an, die besagt, dass n_1 eine Primfaktorzerlegung hat. So gelangt man schließlich zu der gewünschten Zerlegung $n = p_1 \cdot n_1 = p_1 \cdot p_2 \cdot n_2 = \ldots = p_1 \cdot p_2 \ldots p_r$.

□

Es bleibt jetzt nur noch zu zeigen, dass die Primfaktorzerlegung für jede Zahl n eindeutig ist. Das erreicht man mit einem Widerspruchsbeweis:

Beweis des Hauptsatzes der Zahlentheorie
Angenommen, $n \geq 1$ sei die kleinste Zahl, die zwei Primfaktorzerlegungen besitzt:

$$n = p_1 \cdot p_2 \cdots p_r = q_1 \cdot q_2 \cdots q_s$$

Es kann weder $s = 0$ noch $r = 0$ sein, denn sonst wäre $n = 1$, und 1 hat nur eine Primfaktorzerlegung, nämlich das leere Produkt. Es muss also $n \geq 2$ sein. Außerdem ist klar, dass die Mengen $\{p_1, \ldots p_r\}$ und $\{q_1, \ldots q_s\}$ disjunkt sein müssen: Gäbe es p_i, q_j mit $p_i = q_j$, so hätte auch die Zahl n/p_i zwei unterschiedliche Primfaktorzerlegungen. Das steht aber im Widerspruch dazu, dass n die kleinste Zahl mit dieser Eigenschaft ist. Man kann also annehmen, dass $p_i < q_j$

und $i = j = 1$. (Ansonsten können die beiden Zerlegungen einfach vertauscht werden, bzw. die Faktoren umgeordnet.) Nun kann die Zahl

$$m = n - p_1 \cdot q_2 \cdots q_s = (q_1 - p_1) \cdot q_2 \cdots q_s$$

konstruiert werden. Es gilt $0 < m < n$. Die Zahl m hat eine Primfaktorzerlegung, in der die Primzahl p_1 nicht vorkommt: $q_1 - p_1$ kann nicht durch p_1 teilbar sein, sonst würde ja gelten $p_1 \mid q_1$. q_1 ist jedoch eine Primzahl, und $p_1 \neq q_1$. $(q_1 - p_1)$ muss jedoch ebenfalls eine Primfaktorzerlegung $p'_1 \cdots p'_t$ besitzen. Dann hat m die Zerlegung $p'_1 \cdots p'_t \cdot q_2 \cdots q_s$, in der p_1 ebenfalls nicht vorkommt. Andererseits muss m aber durch p_1 teilbar sein, denn $m = n - p_1 \cdot q_2 \cdots q_s$, und $p_1 \mid n$. Damit besitzt m aber zwei verschiedene Primfaktorzerlegungen – ein Widerspruch dazu, dass n minimal mit dieser Eigenschaft ist.

□

1.1.3 Wieviele Primzahlen gibt es?

Schon Euklid entdeckte, dass es unendlich viele Primzahlen gibt.
Den *Beweis* dazu führte er indirekt: Unter der Annahme, dass es endlich viele Primzahlen $p_1, p_2, \ldots p_k$ gibt, kann das Produkt aller Primzahlen gebildet werden: $q = p_1 \cdot p_2 \cdots p_k$. q ist durch alle Primzahlen teilbar. Doch die Zahl $q + 1$ ist dann durch keine der Primzahlen $p_1, p_2, \ldots p_k$ teilbar. Nach dem Hauptsatz der Zahlentheorie existiert jedoch eine Primzahlzerlegung für q. Somit kann die Anzahl der Primzahlen nicht endlich sein.

Allerdings nimmt die Dichte der Primzahlen mit zunehmender Größe immer mehr ab, wie die folgende Tabelle veranschaulicht. Die Anzahl der Primzahlen, die kleiner oder gleich n sind, wird mit $\pi(n)$ bezeichnet.

n	$\pi(n)$
1.000	168
1.000.000	78.498
1.000.000.000	50.847.534
1.000.000.000.000	37.607.912.018
1.000.000.000.000.000	29.844.570.422.669
1.000.000.000.000.000.000	24.739.954.287.740.860

Primzahlendichte[1]

Gauss vermutete bereits im 18. Jahrhundert den Zusammenhang $\pi(x) \approx x/\ln(x)$. Der Beweis wurde allerdings erst 1896 von Hadamard und de la Vallée Poussin erbracht. Seitdem ist die folgende Formel als "Prime Number Theorem" bekannt.

$$\lim_{x \to \infty} \frac{\pi(x)}{x/\ln x} = 1$$

Rätselfragen zu Primzahlen

Es gibt viele, teilweise beantwortete Rätselfragen betreffend Primzahlen, unter anderem ...

Gibt es beliebig große Primzahllücken?
Die Abstände zwischen zwei aufeinander folgenden Primzaheln können tatsächlich beliebig groß werden, wie schnell gezeigt werden kann:
Man betrachte die Folge $n!+2, n!+3, n!+4, n!+5, \ldots n!+n$. Jede dieser $n-1$ aufeinander folgenden Zahlen ist zusammengesetzt, denn $n! + k$ ist für $k \leq n$ durch k teilbar. Diese Argumentation funktioniert aber für jedes beliebige n, daher können Lücken zwischen Primzahlen beliebig groß werden.

Gibt es in jedem Intervall zwischen zwei Quadratzahlen eine Primzahl?
Dies ist höchstwahrscheinlich der Fall, der Beweis ist aber ebenfalls ausständig.

Gibt es unendlich viele Primzahlzwillinge, d.h. Primzahlen p_1, p_2 mit $p_1 + 2 = p_2$?
Es sieht tatsächlich so aus, als könnte man beliebig viele solcher Zwillinge wie etwa 17 und 19 finden. Der Beweis dafür steht aber noch aus.

Ist jede gerade Zahl über 2 Summe zweier Primzahlen?
Diese Behauptung wurde bereits 1742 von Christian Goldbach aufgestellt und ist seitdem als "Goldbachsche Vermutung" bekannt. So ist etwa $36 = 17 + 19$, $52 = 47 + 5$, $28 = 23 + 5$, ... Diese Liste scheint sich beliebig fortsetzen zu lassen, die Vermutung konnte bis dato aber weder bewiesen noch widerlegt werden.

1.1.4 Besondere Primzahlen

Mersenne–Zahlen

Zahlen der Form $M_n = 2^n - 1$ werden, dem französischen Mathematiker Marin Mersenne (8/9/1588 – 1/9/1648) zu Ehren, als Mersenne–Zahlen bezeichnet. Von besonderem Interesse sind Primzahlen, die sich in dieser Form darstellen lassen. Einige solche Primzahlen wurden bereits in Euklids "Elementen" erwähnt, und seit 1996 setzt sich die "Great Internet Mersenne Prime Search" (GIMPS) zum Ziel, möglichst viele weitere Mersenne'sche Primzahlen zu finden.

Zahl p	$2^p - 1$	Jahr	Entdecker
2	3	Antike	Euklid
3	7	Antike	Euklid
5	31	Antike	Euklid
7	127	Antike	Euklid
13	8191	1461	Im *Codex lat. Monac. 14.908* erwähnt
17	131 071	1588	Pietro Antonio Cataldi
19	524 287	1588	Pietro Antonio Cataldi
31	2 147 483 647	1772	Leonhard Euler
		⋮	
23 209	6 987 Dezimalstellen	1979	Curt Noll, unter Verwendung der *CDC-CYBER-174*
44 497	13 395 Dezimalstellen	1979	Harry Nelson, David Slowinski unter Verwendung der *CRAY-1*
		⋮	
20.996.011	6.320.430 Dezimalstellen	17/11/2003	Michael Shafer mit *GIMPS*
24.036.583	7.235.733 Dezimalstellen	15/05/2004	Josh Findley mit *GIMPS*

Übersicht Mersenne'sche Primzahlen[2]

Bis jetzt sind 41 Mersenne'sche Primzahlen bekannt. Es ist noch ungeklärt, ob unendlich viele Primzahlen dieser Form existieren.

Die Tabelle legt bereits die Vermutung nahe, dass $M_n = 2^n - 1$ nur dann eine Primzahl sein kann, wenn auch n prim ist. Dies ist tatsächlich der Fall:

Lemma
Wenn M_n eine Primzahl ist, so ist auch n prim.

Beweis:
Für eine zusammengesetzte Zahl $n = p \cdot q$ ist $M_n = 2^{p \cdot q} - 1 = (2^p)^q - 1$. Aber nach der geometrischen Reihe gilt $(2^p)^q - 1 = (2^p - 1)(2^{p(q-1)} + 2^{p(q-2)} + \ldots + 1) = M_p \cdot \sum_{i=0}^{q-1} 2^{p \cdot i}$. Somit ist M_p ein Teiler von M_n.

□

Die Umkehrung, dass M_p für jede Primzahl p wieder prim ist, gilt jedoch nicht, M_{11} und M_{23} etwa sind keine Primzahlen.
Die Voraussetzung, dass n prim ist, stellt damit zwar ein notwendiges Kriterium für die Primalität von M_n dar, aber kein hinreichendes. Ein vollständiges Kriterium, um die Primalität einer Mersenne–Zahl M_n zu bestimmen, ist der Lucas–Lehmer–Test, der in Abschnitt 2 vorgestellt wird.

Fermat–Zahlen

Eine zweite berühmte Art von Primzahlen sind die Fermat–Zahlen. Der französische Mathematiker Pierre de Fermat (17/8/1601 – 12/1/1665) glaubte, für alle Zahlen n sei $F_n = 2^{2^n} + 1$ prim. Das ist allerdings nicht der Fall, wie Leonhard Euler im Jahre 1732 festgestellt hat, indem er den Faktor 641 von F_5 fand. Die einzigen zur Zeit bekannten Fermat–Zahlen sind

n	$2^{2^n} + 1$
0	3
1	5
2	17
3	257
4	65537

Es ist ungeklärt, ob es noch weitere Fermatzahlen gibt, das Projekt "Distributed Search For Fermat Number Divisors" versucht durch Verwendung ungenutzter Rechenleistung (ähnlich wie GIMPS) Faktorisierungen für Fermat– Zahlen zu finden. Für F_5 bis F_{11} sind bereits vollständige Faktorisierungen bekannt, für F_{12} bis F_{23} sind einzelne Faktoren bekannt [3].

1.2 Modulares Rechnen und der Restklassenring $\mathbb{Z}_n$

Auf der Menge $\mathbb{Z}$ der ganzen Zahlen ist eine Äquivalenzrelation wie folgt definiert:

$$a \equiv b \pmod{n} \quad \Leftrightarrow \quad a \operatorname{rem} n = b \operatorname{rem} n$$

$a \operatorname{rem} n$ bezeichnet dabei den Rest von a bei Division durch n.
Durch die Relation wird die Menge $\mathbb{Z}$ in Äquivalenzklassen geteilt. Wir bezeichnen Restklassen mit $\overline{a}, \overline{b}, \ldots$, wobei $\overline{a} = \{x \in \mathbb{Z} \mid x \operatorname{rem} n = a\}$.

Die Menge der Äquivalenzklassen $\mathbb{Z}_n$ kann als Ring betrachtet werden, indem eine Addition und eine Multiplikation für alle $\overline{a}, \overline{b} \in \mathbb{Z}_n$ folgendermaßen definiert werden:

$$\overline{a} + \overline{b} := \overline{a + b}$$
$$\overline{a} \cdot \overline{b} := \overline{a \cdot b}$$

Behauptung: Diese Verknüpfungen sind wohldefiniert.

Beweis:
Angenommen, $x \in \overline{a}$ und $y \in \overline{b}$. Dann gibt es $m, k \in \mathbb{N}$, sodass $x = m \cdot n + a$ und $y = k \cdot n + b$.
Für die **Addition** gilt dann $x+y = (m \cdot n+a)+(k \cdot n+b) = (m+k)n+(a+b)$. Also $x+y \in \overline{a+b}$.

Für die **Multiplikation** gilt dann $x \cdot y = (m \cdot n+a)(k \cdot n+b) = (m \cdot k \cdot n+m \cdot b+k \cdot a)n+(a \cdot b)$.
Also $x \cdot y \in \overline{a \cdot b}$.
Die Ringaxiome lassen sich leicht nachprüfen, siehe etwa [4].
Anstatt $\overline{a}$ wird im folgenden einfach a verwendet, wenn aus dem Kontext klar ist, dass es sich um ein Element einer Restklasse handelt.

Die Menge $\mathbb{Z}_n^* \subset \mathbb{Z}_n$ ist die Teilmenge der multiplikativ invertierbaren Elemente in $\mathbb{Z}_n$.
Die Menge $\mathbb{Z}_n^*$ ist auch eine multiplikative Untergruppe, denn

- Für $a, b \in \mathbb{Z}_n^*$ gilt $(a \cdot b) \cdot (b^{-1} \cdot a^{-1}) = a \cdot (b \cdot b^{-1}) \cdot a^{-1} = a \cdot 1 \cdot a^{-1} = 1$, sodass $\mathbb{Z}_n^*$ abgeschlossen ist.
- $1 \in \mathbb{Z}_n^*$
- Für $a \in \mathbb{Z}_n^*$ ist auch $a^{-1} \in \mathbb{Z}_n^*$, da a gerade das zu a^{-1} inverse Element ist.

Bemerkung
Es gilt $a \in \mathbb{Z}_n^* \Leftrightarrow a < n$ und $ggT(a, n) = 1$.

Beweis
$\Leftarrow$:

> Angenommen, $ggT(a, n) = 1$. Dann gibt es $x, y \in \mathbb{Z}$ mit $1 = ax + ny$. Daraus folgt $ax \equiv 1 \pmod{n}$, und x ist gerade das zu a inverse Element in $\mathbb{Z}_n$.

$\Rightarrow$:

$a \in \mathbb{Z}_n$ ist also invertierbar, es gibt ein $b \in \mathbb{Z}_n$, sodass $ab \equiv 1 \pmod{n}$. Das bedeutet, es gibt ein $k \in \mathbb{Z}$ mit $ab-1 = nk$. Also ist $ggT(ab, nk) = 1$ und damit auch $ggT(a, n) = 1$.

□

Für $n = p$ prim ist $\mathbb{Z}_p$ daher sogar ein Körper, denn dann ist $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$.

1.2.1 Der Polynomring $\mathbb{Z}_n[X]$

Der Primzahltest von Agrawal, Kayal und Saxena funktioniert mit einer Charakterisierung von Primzahlen, die den Polynomring $\mathbb{Z}_n[X]$ verwendet. Daher wird er an dieser Stelle kurz eingeführt.

Definition 4 (Der Polynomring $\mathbb{Z}_n[X]$) *Formal betrachtet ist $\mathbb{Z}_n[X]$ die Menge der Folgen $(a_n)_{n\in\mathbb{N}}$ mit Koeffizienten $a_i \in \mathbb{Z}_n$, wobei nur endlich viele Koeffizienten $a_i \not\equiv 0$ sind.*

$\mathbb{Z}_n[X]$ bildet mit den folgenden Verknüpfungen einen Ring:

Die **Addition** erfolgt komponentenweise:
$(a_n)_{n\in\mathbb{N}} + (b_n)_{n\in\mathbb{N}} = (c_n)_{n\in\mathbb{N}}$, wobei $c_i := a_i + b_i$ für alle $i \in \mathbb{N}$.
Das neutrale Element der Addition ist die konstante Folge$(0, 0, 0, \ldots) =: 0_{\mathbb{Z}_n[X]}$, das zu $a_n = (a_0, a_1, a_2, \ldots)$ inverse Element ist die Folge $-a_n = (-a_0, -a_1, -a_2, \ldots)$.

Die **Multiplikation** ist gliedweise definiert:
$a_n \cdot b_n = c_n$, wobei $c_i := \sum_{j+k=i} a_j \cdot b_k$.
$\mathbb{Z}_n[X]$ ist sogar unitär mit $1_{\mathbb{Z}_n[X]} = (1, 0, 0, \ldots)$.

Die Ringaxiome lassen sich leicht nachprüfen, siehe etwa [4]. Eine Folge $P \in \mathbb{Z}_n[X]$, $P = (a_0, a_1, a_2 \ldots)$ wird aber meist als Polynom $P = a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots$ geschrieben. $X^i \in \mathbb{Z}_n[X]$ entspricht der Folge $(0, \ldots, 0, 1, 0, \ldots)$, wobei die 1 gerade an der i-ten Stelle steht.

1.3 Nützliche Sätze

1.3.1 Eulers Theorem

Definition 5 (Euler'sche φ-Funktion) *Euler definierte eine Funktion, die einer Zahl n die Anzahl der Elemente zuordnet, die kleiner als n und zu n teilerfremd sind. In Symbolen also* $\varphi(n) = \#\{m \mid ggT(n,m) = 1\} = \#\mathbb{Z}_n^*$.

Satz von Euler
Für $n \in \mathbb{Z}$ mit $1 \leq a < n$, $ggT(a,n) = 1$ gilt:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Beweis: Wenn jede der zu n teilerfremden Zahlen $m_1, m_2, \ldots m_{\varphi(n)}$ mit einer Zahl a multipliziert wird, $1 \leq a < n$ und $ggT(a,n) = 1$, so wird die Menge $\{m_1, m_2, \ldots m_{\varphi(n)}\}$ in sich selbst übergeführt:

$$am_1 \cdot am_2, \cdots am_{\varphi(n)} \equiv a^{\varphi(n)}(m_1 \cdot m_2, \cdots m_{\varphi(n)}) \equiv m_1 \cdot m_2, \cdots m_{\varphi(n)} \pmod{n}$$

Unter Umständen treten die Faktoren auf der rechten Seite in anderer Reihenfolge auf, doch das spielt aufgrund der Kommutativität keine Rolle. Man darf "kürzen" und erhält den Satz von Euler: $a^{\varphi(n)} \equiv 1 \pmod{n}$

Einen andersartigen Beweis kann man mit etwas grundlegender Algebra führen [5]: Die Menge $m_1, m_2, \ldots m_{\varphi(n)}$ bildet die multiplikative Gruppe $\mathbb{Z}_n^*$. Die Ordnung dieser Gruppe ist $\varphi(n)$. Die Ordnung der von einem beliebigen Element $a \in \mathbb{Z}_n^*$ erzeugte Untergruppe teilt nach dem Satz von Lagrange die Ordnung von $\mathbb{Z}_n^*$: $\#a \mid \varphi(n)$. Außerdem gilt für alle $a \in G$ $a^{\#a} \equiv 1$ in allen multiplikativen Gruppen G, und daher $a^{\varphi(n)} \equiv a^{k \cdot \#a} \equiv (a^k)^{\#a} \equiv 1 \pmod{n}$.

□

1.3.2 Der Kleine Satz von Fermat

Fermat entdeckte den folgenden Zusammenhang: Falls p prim ist und $p \not| a$

$$a^{p-1} \equiv 1 \pmod{p}$$

Der Kleine Satz von Fermat ist eigentlich ein Spezialfall des Theorems von Euler: Für eine Primzahl p ist die Menge $\{m \mid ggT(m,p) = 1\} = \{1, \ldots p-1\}$, also $\varphi(p) = p - 1$.

2

Algorithmen

2.1 Faktorisierung

Die Faktorisierung einer Zahl n, also das Auffinden der eindeutigen Primfaktorzerlegung $n = p_1^{m_1} \cdot p_2^{m_2} \cdots p_k^{m_k}$ beschäftigt Mathematiker seit der Antike. Eine besondere Bedeutung kommt ihr seit der Verwendung bestimmter kryptographischer Verfahren wie etwa RSA zu, deren Sicherheit auf der hohen Komplexität momentan bekannter Faktorisierungsverfahren beruht. Bis dato sind keine Faktorisierungsmethoden mit polynomialer Zeitkomplexität bekannt.
Die folgende Tabelle enthält eine Übersicht über Faktorisierungsverfahren, deren Komplexität sowie deren historischen Kontext, wobei aber natürlich kein Anspruch auf Vollständigkeit erhoben wird.

Verfahren	**Zeit**	Komplexität
Probedivision	Antike	$O^{\sim}(2^{\frac{n}{2}})$
$(p-1)$–Verfahren	Pollard, 1974	$O^{\sim}(2^{\frac{n}{4}})$
ρ-Methode	Pollard, 1975	$O^{\sim}(2^{\frac{n}{4}})$
Continued Fractions	Morrison, Brillhart, 1981	$exp(O^{\sim}(\sqrt{n}))$
Quadratic Sieve	Pomerance, Kraitchik, Dixon, 1981	$exp(O^{\sim}(\sqrt{n}))$
Elliptic Curves	Lenstra, 1987	$exp(O^{\sim}(\sqrt{n}))$
Number Field Sieve	Pollard u.a. 1990	$exp(O^{\sim}(\sqrt{n}))$

2.1.1 Probedivision

Das naheliegende Verfahren der Probedivison stammt aus der Antike. Es beruht auf der Idee, die zu faktorisierende Zahl n auf Teilbarkeit durch kleinere Zahlen zu testen, indem eine Probedivision durchgeführt wird. Dabei muss n natürlich nicht auf Teilbarkeit durch alle Zahlen $1, \ldots n-1$ getestet werden. Man stellt schnell fest, dass einige Verbesserungen gemacht werden können:

- Die Probedivision n/p muss nur für Primzahlen p durchgeführt werden. Falls n also bereits auf Teilbarkeit durch p getestet wurde, kann die Division durch Vielfache von p ausgelassen werden.
- Wenn n als Produkt $n = p \cdot q$ dargestellt werden kann, ist einer der Faktoren immer kleiner als $\sqrt{n}$. Die Probedivision muss also nur für Zahlen p mit $p \leq \sqrt{n}$ durchgeführt werden, q erhält man durch Division als Quotient $q = n/p$.

Die Anzahl der nötigen Divisionen beträgt dennoch $p_2(n)(\log n)^c$, wobei p_2 der zweitgrößte Primfaktor von n ist, und c eine Konstante. Das Verfahren der Probedivision wird trotzdem in vielen Fällen angewandt, besonders zur Faktorisierung kleiner Zahlen. GIMPS etwa verwendet die Probedivision, um 95% der in Frage kommenden Faktoren zu eliminieren[6], und auch die in MATHEMATICA implementierte `PrimeQ`-Funktion verwendet unter anderem Probedivision durch kleine Primzahlen.

2.1.2 Test auf exakte Potenzen

Viele Faktorisierungsverfahren und Primzahltests funktionieren nur für Eingaben n, die keine exakte Potenz sind, d.h. es existieren keine ganzen Zahlen b, m mit $n = m^b$. Es gibt jedoch relativ

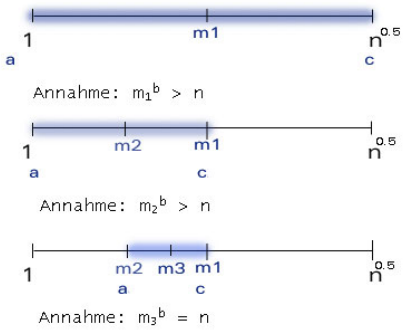

Abbildung 2.1: Test auf exakte Potenzen: Binäre Suche für Basis

einfache Verfahren zum Test auf exakte Potenzen. Im folgenden wird eine Methode exemplarisch vorgestellt.
Als Vorüberlegung können zunächst Schranken für eine mögliche Basis m und einen Exponenten b festgelegt werden: es gilt $2 \leq m \leq \sqrt{n}$ und $2 \leq b \leq \mathrm{ld}(n)\}$.
Die Idee besteht darin, eine Schleife über alle möglichen Exponenten laufen zu lassen, und für jeden möglichen Exponenten b eine Basis m zu suchen. Dazu wird zunächst ein beliebiges $m_0 \in \{2, \ldots \sqrt{n}\}$ gewählt, und die Zahl m_0^b mit n verglichen.

- Falls $m_0^b = n$ gilt, wurden eine passende Basis m und ein Exponent b gefunden, der Algorithmus wird erfolgreich beendet.
- Falls $m_0^b < n$, wird der Vergleich mit einem größeren m_1 wiederholt.
- Falls $m_0^b > n$, wird der Vergleich mit einem kleineren m_1 wiederholt.

Der Vergleich von m_i mit n und die Konstruktion von m_{i+1} wird so lange wiederholt, bis entweder ein passendes m gefunden wurde oder für zwei "Testbasen" m_i, m_{i+1} gilt, dass $m_i - m_{i+1} \leq 1$. Dann gibt es für den Exponenten b keine passende Basis, und b wird inkrementiert. Abbildung 2.1 veranschaulicht die Suche nach dem passenden m.

Es wird also einfach eine binäre Suche für m in eine lineare Suche für b eingebettet. Der Algorithmus sieht folgendermaßen aus:

Algorithmus 2 (Test auf exakte Potenzen)
EINGABE: ganze Zahl $n \geq 2$

```
int a, b, c, m; b=2;
while (2^b ≤ n)
   a = 1; c = ⌈n⌉ + 1;
   while (c − a ≥ 2)
     m = ⌈(a + c)/2⌉;
     p = m^b;
     if (p == n) return ''Primzahlpotenz'' , m, b
```

```
    if (p < n) a = m; else c = m;
  b = b + 1;
return ''keine Primzahlpotenz''
```

Dieser Test auf exakte Potenzen benötigt $O^{\sim}((\log n)^3)$ Bitoperationen. Mit einer anderen Herangehensweise ließe sich ein Test auf exakte Potenzen auch in $O^{\sim}((\log n)^2)$ Bitoperationen durchführen[7].

2.1.3 ρ-Methode nach Pollard

Die Idee dieses 1975 von J. Pollard erfundenen probabilistischen Verfahrens besteht darin, die Berechnung des größten gemeinsamen Teilers zweier Zahlen – ein vergleichsweise einfaches Unterfangen – für die Faktorisierung zu verwenden.

Als Eingabe wird eine zusammengesetzte Zahl n mit (noch unbekannten) Primfaktoren $p, q, \ldots$ benötigt. n darf allerdings keine exakte Potenz sein.
Ziel ist es jetzt, zufällig Zahlen x_i und y_i zu finden, sodass $x_i \equiv y_i \pmod{p}$ für einen Teiler p von n gilt, also $(x_i - y_i) \equiv 0 \pmod{p}$. Dann ist der größte gemeinsame Teiler von n und der Differenz $x_i - y_i$ ein Vielfaches von p, also $ggT((x_i - y_i), n) = k \cdot p$ für ein $k \in \mathbb{N}$. Falls $ggT((x_i - y_i), n) = n$ oder $ggT((x_i - y_i), n) = 1$, erhält man daraus keine weitere Information. Für $k \geq 1$ erhält man mit $ggT((x_i - y_i), n)$ aber einen echten Faktor von n bzw. ein Vielfaches.

Zur iterativen Erzeugung von (Pseudo-)Zufallszahlen x_i wird beim ρ–Verfahren eine Polynomfunktion f verwendet. Der Startwert x_0 kann beliebig gewählt werden. Für x_i, $i > 0$ gilt $x_i = f(x_{i-1}) \operatorname{rem} n$.
Lineare Funktionen sind für f nicht geeignet, aber Polynome mit hohem Grad verschlechtern die Laufzeit. Daher werden meist Funktionen vom Grad 2 verwendet. Eine Ausnahme bildet allerdings $f(x) = x_n^2 - 2 \pmod{n}$. Es erzeugt nach dem ersten Vorkommen von 1 oder -1 nur mehr den Wert -1 und kann daher nicht verwendet werden.

Wie viele Folgenglieder x_i müssen aber berechnet werden, bis eine Kongruenz modulo p vorkommt? Klar ist, dass nach spätestens p Schritten eine Kollision auftritt, also zwei Folgenglieder x_j und x_k mit $x_j \equiv x_k \pmod{p}$ berechnet wurden. Folgendes Lemma gibt genauere Auskunft:

Lemma
Angenommen, es werden aus p (Lotto-)Kugeln zufällig einzelne gezogen (und wieder zurückgelegt). Die zu erwartende Anzahl von Ziehungen, bis eine Kugel zum zweiten Mal gezogen wird, ist in der Größenordnung $O(\sqrt{p})$.
Der *Beweis* kann in [8] nachgeschlagen werden.

In der Praxis müssten daher $O(\sqrt{p})$ Zahlen gespeichert werden, um eine Kollision festzustellen. Floyds Trick zur Zykelfindung ermöglicht alternativ das Auffinden einer Kollision ohne hohen Speicheraufwand:

Floyds Trick zur Zykelfindung

Es wird eine zweite Folge y_i erzeugt, wobei $y_0 = x_0$ und $y_i = f(f(y_{i-1})) \operatorname{rem} n$ für $i > 0$. Damit ist y_i eine Teilfolge von x_i, die "doppelt so schnell" läuft. Für alle Folgenglieder x_i, y_i wird $ggT((x_i - y_i), n)$ berechnet.

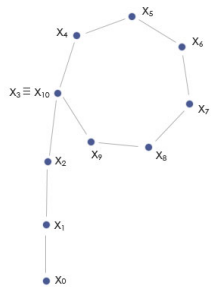

Abbildung 2.2: Pollards ρ-Methode: Zykelentstehung

Wie bald kann jetzt mit $x_i \equiv y_i \pmod{p}$ gerechnet werden? Die unendliche Folge $x_i \equiv f(x_{i-1}) \pmod{n}$ hat Werte in einer endlichen Menge, also wiederholen sich die Folgenglieder nach spätestens p Schritten – es entsteht ein Zykel modulo p: es gibt $t, l \in \mathbb{N}$ mit $t + l \leq p$ und $x_i \equiv x_{i+l} \pmod{p}$ $\forall i \geq t$. Abbildung 2.2 veranschaulicht die Zykelentstehung und macht ersichtlich, woher die Bezeichnung "ρ-Methode" stammt.

Es gilt $x_i \equiv y_i \pmod{p}$ genau dann, wenn $i \geq t$ und $l \mid (2 \cdot i - i) = i$. Der kleinste derartige Index ist $i = t + (-t \operatorname{rem} l) < t + l$, wenn $t > 0$. Falls $t = 0$ gilt, ist $i = l$.
Für die Komplexität der Zykelfindung erhält man daraus mit einigen heuristischen Überlegungen $O^{\sim}(n^{\frac{1}{4}})$.

Es folgt ein Algorithmus zur Auffindung eines Faktors noch der ρ-Methode. f bezeichnet dabei die Polynomfunktion.

Algorithmus 3 (ρ-Methode nach Pollard)
EINGABE: ganze Zahl $n \geq 3$, nicht prim und keine exakte Potenz
Wähle $x_0 \in \{0, \ldots, n-1\}$ *zufällig*
$y_0 = x_0$; $i = 0$;
do
  $i++$;
  $x_i = f(x_{i-1}) \bmod n$;
  $y_i = f(f(y_{i-1})) \bmod n$;
  $g = ggT(x_i - y_i, n)$;
  if $(1 < g < n)$
   return g;

Beispiel zur ρ-Methode nach Pollard
Wir versuchen, für die Eingabe $n = 91 = 13 \cdot 7$ eine Faktorisierung zu finden. Es wird die Polynomfunktion $f(x) = x^2 + 3 \pmod{n}$ verwendet.

x_i	$y_i = x_{2 \cdot i}$	$(x_i - y_i) \pmod{n}$	$ggT((x_i - y_i) \operatorname{rem} n, \text{n})$
29	29	0	91
25	82	34	1
82	52	30	1
84	77	7	7

Wir starten mit dem (völlig beliebigen) Wert $x_0 = y_0 = 29$. Im ersten Schritt erhalten wir natürlich kein nützliches Ergebnis, im zweiten und dritten Schritt ist der größte gemeinsame Teiler von n und $x_i - y_i$ 1, was auch nichts über einen Faktor aussagt. Bei x_4 und y_4 taucht als Differenz aber bereits 7 auf, was tatsächlich ein Faktor ist.

2.2 Primzahltests

Viele Faktorisierungsalgorithmen, wie etwa Pollards ρ–Verfahren, benötigen als Eingabe eine zusammengesetzte Zahl. Auch einige Verschlüsselungsmethoden arbeiten mit "großen" Primzahlen. Für solche Fälle werden möglichst effiziente Primzahltests benötigt. Im folgenden werden drei probabilistische Algorithmen vorgestellt und ein Einblick in einen deterministischen Algorithmus sowie den Lucas–Lehmer–Test für Mersenne–Zahlen gegeben.

2.2.1 Der Fermat–Test

Der einfachste probabilistische Primzahltest basiert auf dem Kleinen Fermatschen Theorem: Für eine Primzahl p und $1 \leq a < p$ gilt demnach

$$a^{p-1} \equiv 1 \pmod{p}$$

Dieses Kriterium ist jedoch nicht hinreichend: Falls für eine Zahl n gilt, dass $a^{n-1} \equiv 1 \pmod{n}$, so ist sie nicht notwendig prim. In diesem Zusammenhang haben sich folgende Bezeichnungen etabliert:

Definition 6 (F–Zeuge) *Eine Zahl a, $1 \leq a \leq n$, heisst ein F–Zeuge für n, falls $a^{n-1} \not\equiv 1 \pmod{n}$. Genauer gesagt: a ist Zeuge für die Nichtprimheit von n.*

Definition 7 (F–Lügner) *Für eine zusammengesetzte Zahl n nennt man eine Zahl a, $1 \leq a \leq n$, einen F–Lügner, falls $a^{n-1} \equiv 1 \pmod{n}$. Die Zahl n wird dann auch pseudoprim zur Basis a genannt.*

Die Zahlen 1 und $n-1$ sind trivialerweise F–Lügner für alle ungeraden zusammengesetzten Zahlen n, da $1^{n-1} \equiv 1 \pmod{n}$ und $(n-1)^{n-1} \equiv (-1)^{n-1} \equiv 1 \pmod{n}$ für alle n gilt.

Bemerkung:
Für eine ganze Zahl $n \geq 2$ gilt:

a) Falls für $1 \leq a < n$ gilt, dass $a^r \equiv 1 \pmod{n}$ für ein $r \geq 1$ gilt, dann ist $a \in \mathbb{Z}_n^*$.

b) Falls $a^{n-1} \equiv 1 \pmod{n}$ für alle Zahlen a, $1 \leq a < n$, gilt, dann ist n eine Primzahl.

Beweis:

a) Da $a \cdot a^{r-1} \equiv 1 \pmod{n}$, ist a^{r-1} gerade das zu a inverse Element.

b) Aus (a) folgt, dass alle Zahlen a, $1 \leq a < n$ in $\mathbb{Z}_n$ invertierbar sind. Also ist n eine Primzahl.

Aus b) der Bemerkung folgt, dass es immer F–Zeugen für eine zusammengesetzte Zahl n gibt. Aus (a) ersieht man allerdings, dass dies gerade die zu n teilerfremden Zahlen sind. Für Zahlen n, die aus wenigen Primfaktoren bestehen, gibt es also nur sehr wenige F–Zeugen – genau $(n-1)-\phi(n)$ Elemente. Falls zum Beispiel $n = pq$ für p,q prim, sind das $p + q - 2$ Zahlen.
Trotzdem kann ein erster Algorithmus für den Primzahltest folgendermassen definiert werden:

Algorithmus 4 (Fermat–Test)
EINGABE: Ungerade, ganze Zahl $n \geq 3$.
`Wähle` a `zufällig aus` $\{2, 3, \ldots n-2\}$
`if` $(a^{n-1} \not\equiv 1 \pmod{n})$
`return ''vielleicht prim'';`
`else`
`return ''zusammengesetzt'';`

Falls $n \geq 3$ eine zusammengesetzte, ungerade Zahl ist, ist die Wahrscheinlichkeit für den Rückgabewert "zusammengesetzt" *meistens* grösser als $\frac{1}{2}$. Warum?
Wie die obige Bemerkung zeigt, ist die Menge der F–Lügner für n, L_n^F, eine Teilmenge der $\mathbb{Z}_n^*$. L_n^F ist jedoch sogar eine Untergruppe, da

a) L_n^F unter der Multiplikation mod(n) abgeschlossen ist: für $a \in L_n^F$ und $b \in L_n^F$ ist auch das Produkt $ab \in L_n^F$, da $a^{n-1} \cdot b^{n-1} \equiv 1 \cdot 1 \equiv 1 \pmod{n}$.

b) $1 \in L_n^F$ und

c) L_n^F unter der Inversenbildung abgeschlossen ist: $1 \equiv 1^{n-1} \equiv (a \cdot a^{-1})^{n-1} \equiv a^{n-1} \cdot (a^{-1})^{n-1} \equiv 1 \cdot (a^{-1})^{n-1} \equiv (a^{-1})^{n-1} \pmod{n}$.

Wenn es zumindest einen F–Zeugen in $\mathbb{Z}_n^*$gibt, ist L_n^F sogar eine echte Untergruppe der $\mathbb{Z}_n$*. Es muss $\#L_n^F$ ein Teiler von $\#\mathbb{Z}_n^*$ sein, wobei $\#\mathbb{Z}_n^* = \phi(n) \leq n-2$. Man erhält also als Abschätzung $\#L_n^F \leq (n-2)/2$. Die Wahrscheinlichkeit, dass eine beliebig aus $\{2, 3, \ldots n-2\}$ gewählte Zahl a in $L_n^F \setminus \{1, n-1\}$ liegt, beträgt also maximal

$$\frac{(n-2)/2 - 2}{n-3} = \frac{(n-6)}{2(n-3)} < \frac{1}{2}$$

Durch Wiederholung von Algorithmus 1 kann die Wahrscheinlichkeit für den (falschen) Rückgabewert "vielleicht prim" bei Eingabe einer zusammengesetzten, ungeraden Zahl $n \geq 3$ beliebig verkleinert werden. Bei m–maliger Wiederholung erhält man als Wahrscheinlichkeit für den Rückgabewert "vielleicht prim"$P = \frac{1}{2^m}$.

Die Laufzeit des Algorithmus wird durch das Potenzieren bestimmt, es werden $O(\log n)$ arithmetische Operationen und $O((\log n)^3)$ Bitoperationen benötigt.

Ist der Fermat–Test also *der* probabilistische Primzahltest? Für zufällig ausgewählte Zahlen n funktioniert der Test recht gut. Doch leider gibt es seltene, besonders störrische Zahlen, für die alle Basen $a \in \mathbb{Z}_n^*$ F–Lügner sind. Es gibt sogar unendlich viele von diesen Saboteuren! Diese Zahlen werden zu Ehren des amerikanischen Mathematikers R.D. Carmichaeal *Carmichaelzahlen* genannt. Falls eine Carmichaelzahl n in den Fermat–Test gelangt, kann nur dann "zusammengesetzt" zurückgegeben werden, wenn a zufällig ein Teiler von n ist. Die Wahrscheinlichkeit für den falschen Rückgabewert "vielleicht prim" beträgt

$$\text{P(Ergebnis "vielleicht prim")} = \frac{\#\text{Teiler von } n}{\#\text{mögliche Basen}} = \frac{\phi(n) - 2}{n-3} > \frac{\phi(n)}{n} = \prod_{p \text{ prim},\ p|n} 1 - \frac{1}{p}$$

Falls n aus nur wenigen Faktoren besteht, liegt dieser Wert sehr nahe bei 1. Der Fermat–Test hat somit ein entscheidendes Handicap: Für Zahlen eines bestimmten Typs funktioniert er mit "an Sicherheit grenzender Wahrscheinlichkeit" nicht.

2.2.2 Der Strong Pseudoprimality Test

Miller und Rabin haben unabhängig voneinander einen weiteren zufallsbasierten Primzahltest entwickelt. Er beruht auch auf dem Satz von Fermat, verwendet aber außerdem nicht triviale Quadratwurzeln von 1 zur Charakterisierung der Primzahlen. Die hier besprochene Variante gibt für Carmichaelzahlen mit einer Wahrscheinlichkeit $\geq \frac{1}{2}$ sogar einen echter Teiler zurück.

Definition 8 (Quadratwurzel von 1) *Zahlen a mit $1 \leq a < n$, für die gilt, dass $a \cdot a \equiv 1 \pmod{n}$, werden Quadratwurzeln von 1 modulo n genannt.*

Bemerkung:
Falls p eine Primzahl ist, und a, $1 \leq a < p$, eine Quadratwurzel von 1 modulo p ist, dann ist entweder $a \equiv 1 \pmod{p}$ oder $a \equiv p-1 \pmod{p}$.
Beweis:
Zerlege $a^2 - 1 \pmod{p}$ in $(a+1)(a-1) \pmod{p}$. Also gilt $a^2 - 1 \equiv 0 \equiv (a+1)(a-1) \pmod{p}$. Damit teilt p das Produkt $(a+1)(a-1)$, was bedeutet, dass es einen der Faktoren teilen muss: $p \mid (a+1)$ oder $p \mid (a-1)$. Daher gilt $a + 1 \equiv 0 \pmod{p}$ oder $a - 1 \equiv 0 \pmod{p}$, also $a \equiv 1 \pmod{p}$ oder $a \equiv -1 \equiv p - 1 \pmod{p}$.

□

Falls n nicht prim ist, kann es mehrere verschiedene Zahlen $a \not\equiv 1$,$a \not\equiv n-1$ geben, für die $a^2 \equiv 1$ gilt.

Zum Beispiel sind die Quadratwurzeln von 1 modulo $119 = 17{\cdot}7$ die Menge $Q_{119} = \{1, 50, 69, 118\}$. Noch mehr Quadratwurzeln von 1 gibt es modulo $273 = 13{\cdot}7{\cdot}3$: $Q_{273} = \{1, 64, 92, 118, 155, 181, 209, 272\}$. Mit dem Chinesischen Restsatz kann man feststellen [1], dass es für ein n, das aus k verschiedenen Primfaktoren $p_1, p_2, \ldots p_k$ zusammengesetzt ist, immer 2^r Quadratwurzeln von 1 modulo n gibt. Falls n nicht aus sehr vielen Primfaktoren besteht, ist es daher eher hoffnungslos, mit zufällig gewählten a's nicht triviale Quadratwurzeln von 1 zu finden.

Zurück zum Fermat–Test: Man kann für Testkandidaten n natürlich voraussetzen, dass n ungerade ist. Also enthält $n-1$ mindestens einmal den Primfaktor 2 und kann als $n - 1 = m \cdot 2^\nu$ geschrieben werden, $\nu \geq 1$. Daraus folgt $a^{n-1} = a^{m \cdot 2^\nu}$. Man kann jetzt $a^{n-1} \pmod{n}$ in $\nu + 1$ Schritten berechnen. Dazu bezeichnet man:

$$b_0 = a^m \pmod{n}$$
$$b_i = b_{i-1}^2 \pmod{n} \text{ für } i = 1, \ldots \nu$$

Als Beispiel hier die Berechnungsfolge für $n = 325$ und verschiedene Basen a:

a	$b_0 = a^{81}$	$b_1 = a^{162}$	$b_2 = a^{324}$
2	252	129	66
7	307	324	1
32	57	324	1
49	324	1	1
65	0	0	0
126	1	1	1
201	226	51	1
224	274	1	1

Die Zahlen $a^{n-1} \pmod n$ in der letzten Spalte bezeichnen gerade die für den Fermattest ausschlaggebenden Werte. So sind also 2 und 65 F–Zeugen für 325, während 7, 32, 49, 126, 201 und 224 F–Lügner sind. Soweit nichts Neues.
Durch die schrittweise Berechnung mit Basis 201 stellen wir aber fest, dass 51 eine nicht triviale Quadratwurzel von 1 ist, ebenso 274. Spätestens jetzt ist klar, dass 325 keine Primzahl sein kann.

Wie kann die Berechnungsfolge allgemein aussehen?

- Wenn bereits $b_0 = 1$, ist auch $b_i = 1$ für alle $1 \leq i \leq \nu$ und es kann keine Aussage über die Zusammengesetztheit von n getroffen werden. Dieser Fall trifft im obigen Beispiel auf $a = 126$ zu. (Im folgenden Algorithmus führt dies zu Rückgabe in Schritt c.)
- Es kann ein $j \in \{1, ...\nu - 1\}$ geben, sodass $b_j = n-1$ und $b_i \neq 1$ für alle $0 \leq i < j$. Nach der Zahl $n-1$ tauchen natürlich nur mehr Einsen auf, wie im Beispiel bei $a = 7$ und $a = 32$. Es kann ebenfalls nichts über die Zusammengesetztheit von n ausgesagt werden. (Abbruch mit Rückgabe "vielleicht prim" in Schritt f)
- Falls keine Eins in der Berechnungssequenz mit Basis a steht, also auch $b_\nu \neq 1$, ist a F–Zeuge für n. (Abbruch in Schritt e des Algorithmus)
- Wenn für alle $i \geq j$ ab einem Berechnungsschritt $j > 0$ gilt, dass $b_i = 1$, dann ist b_{j-1} eine nicht triviale Quadratwurzel von 1, und n ist mit Sicherheit nicht prim. Im obigen Beispiel trifft dies bei $a = 201$ und $a = 224$ zu. (Der folgende Algorithmus bricht dementsprechend in Schritt f ab.)

Diese Beobachtungen werden für einen Algorithmus verwendet:

Algorithmus 5 (Strong Pseudoprimality Test)
EINGABE: Ungerade, ganze Zahl $n \geq 3$.

a) *Wähle a zufällig aus* $\{2, 3, \ldots n-2\}$

b) $g = ggT(a, n)$
`if` $(g > 1)$
`return` g;

c) *Finde eine ungerade Zahl* m *und* $\nu \in \mathbb{N}$ *für die Zerlegung* $n = m \cdot 2^\nu$
$b_0 = a^m \pmod n$
`if` $(b_0 == 1)$
`return` ‘‘*vielleicht prim*’’

```
d) for (i = 0; i ≤ ν; i++)
       b_i = b_{i-1}^2 (mod n)
e) if (b_ν == 1)
       j = min{0 ≤ i < ν | b_{i+1} == 1}
   else
       return ''zusammengesetzt''
f) g = ggT(b_j + 1, n)
   if (g == 1) ∨ (g == n − 1)
       return ''vielleicht prim''
   else
       return g
```

Für die Analyse des Algorithmus wird folgende Charakterisierung von Carmichaelzahlen benötigt:

Lemma
Jede Carmichaelzahl n ist quadratfrei, d.h. kein Primfaktor p von n kommt in der Primfaktorzerlegung von n mehr als einmal vor.
Der *Beweis* ist in [7] zu finden.

Behauptung
Algorithmus 5 gibt für alle eingegebenen Primzahlen n "vielleicht prim" zurück. Falls n zusammengesetzt und keine Carmichaelzahl ist, wird mit Wahrscheinlichkeit $\geq \frac{1}{2}$ "zusammengesetzt" zurückgegeben. Bei Eingabe einer Carmichaelzahl n wird mit Wahrscheinlichkeit $\geq \frac{1}{2}$ sogar ein Teiler von n gefunden.

Beweis
Fall 1: n ist zusammengesetzt und keine Carmichaelzahl.

> Das letzte Glied in der Folge der b_i ist $b_\nu \equiv a^{m2^\nu} \equiv a^{n-1}$. Wie in der Analyse des Fermattest erläutert, ist a mit Wahrscheinlichkeit $\geq \frac{1}{2}$ F–Zeuge und es wird in Schritt e "zusammengesetzt" zurückgegeben.

Fall 2: n ist eine Primzahl.

> In diesem Fall ist $b_\nu \equiv 1 \pmod{n}$. Wenn $b_0 \equiv 1 \pmod{n}$, wird in Schritt c "wahrscheinlich prim" zurückgegeben. Sonst hat man $b_j \neq 1$ und $b_{j+1} \equiv 1 \pmod{n}$. Nachdem es in $\mathbb{Z}_n$ aber keine nicht trivialen Quadratwurzeln von 1 gibt, muss gelten $b_j \equiv -1 \pmod{n}$. Damit wird g in Schritt f auf n gesetzt, und es wird korrekt "wahrscheinlich prim" zurückgegeben.

Fall 3: n ist eine Carmichaelzahl.

> Wir setzen P als die Menge aller Primfaktoren von n. Nachdem n quadratfrei ist, also alle Faktoren verschieden sind, ist $n = \prod_{p \in P} p$.
> Wir betrachten jetzt die folgende Menge von Indizes: $I = \{i \mid 0 \leq i \leq \nu \text{ und } \forall u \in \mathbb{Z}_n^* \text{ ist } u^{2^i m} = 1\}$. Da n eine Carmichaelzahl ist, gilt $a^{n-1} \equiv 1$ für alle $a \in \mathbb{Z}_n^*$. Also muss $\nu \in I$ sein. Aber $0 \notin I$, denn da m ungerade ist, ist $(-1)^m \equiv -1 \neq 1$. Falls $i \in I$, sind außerdem auch alle $j \in I$ mit $i < j \leq \nu$. I ist also eine echte Teilmenge

von $\{0, \dots n\}$, und es gibt ein $l < \nu$, sodass $l \notin I$ und $l+1, \dots l+k = \nu \in I$.
Mit diesem Index l können wir nun die folgende Menge definieren:
$G = \{u \in \mathbb{Z}_n^* \text{ mit } u^{2^l m} = \pm 1\} \subset \mathbb{Z}_n^*$
Damit gilt für alle $u \in G$, dass $u^{2^{l+1}m} = 1$. G ist die Menge der Basen u, für die $u^{2^l m}$ eine triviale Quadratwurzel von 1 ist. Wenn wir eine derartige Basis zufällig erwischen, haben wir Pech, und geben "wahrscheinlich prim" zurück. Die Wahrscheinlichkeit, ein $a \in G$ zu erwischen, ist aber $\leq \frac{1}{2}$:
G ist eine Untergruppe von $\mathbb{Z}_n^*$, denn

- für $a, b \in G$ ist auch $ab \in G$: $(ab)^{2^l m} \equiv a^{2^l m} b^{2^l m} \equiv \pm 1 \cdot \pm 1 \equiv \pm 1$
- $1 \in G$
- für $a \in G$ ist auch $a^{-1} \in G$, denn $(a^{-1})^{2^l m} \equiv (a^{2^l m})^{-1} \equiv \pm 1$

G ist außerdem eine echte Teilmenge von $\mathbb{Z}_n^*$, wie die folgende Konstruktion zeigt:
Es gibt ein $p \in P$ und ein $b \in \mathbb{Z}$, das relativ prim zu p ist. Da n quadratfrei ist, sind p und n/p relativ prim, und wir können den Chinesischen Restsatz folgendermaßen anwenden: es gibt ein $c \in \mathbb{Z}$, sodass $c \equiv b \pmod p$ und $c \equiv 1 (\text{mod } n/p)$. Dann ist $c \pmod n \in \mathbb{Z}_n^* \setminus G$.
Nachdem G also eine echte Untergruppe von $\mathbb{Z}_n^*$ ist, gilt $\#G \leq \frac{1}{2}\#\mathbb{Z}_n^*$ und die Wahrscheinlichkeit, ein $a \in G$ zu erwischen, ist $\leq \frac{1}{2}$.
Wir behaupten außerdem, dass der Algorithmus einen echten Teiler von n zurückgibt, wenn die gewählte Basis $a \in \mathbb{Z}_n^* \setminus G$: Nachdem $b_{l+1} \equiv a^{2^{l+1}m} \equiv 1 \pmod n$, gilt auch $b_{l+1} \equiv 1 \pmod p$ für alle Primfaktoren $p \in P$. Da es in $\mathbb{Z}_p$ keine nicht trivialen Quadratwurzeln von 1 gibt, muss $b_l \equiv 1$ oder $b_l \equiv -1 \pmod p$ sein. Aber $b_l \pmod n$ ist weder 1 noch -1, also kommen beide Möglichkeiten tatsächlich vor. Damit gibt es mindestens einen Primfaktor p von n mit $b_l \equiv -1 \pmod p$, und man erhält in Schritt f

$$g = ggT(b_l + 1, n) = \prod_{p \in P \text{ mit } a^{2^l m} \equiv -1 \pmod p} p$$

□

Damit ist die Wahrscheinlichkeit, bei Eingabe einer zusammengesetzten Zahl n das falsche Resultat "vielleicht prim" zu erhalten, $\leq \frac{1}{2}$. Durch Wiederholung des Tests lässt sich die Wahrscheinlichkeit für eine falsche Rückgabe jedoch beliebig verkleinern. Bei m–maliger Wiederholung des Strong Pseudoprimality Tests beträgt sie $\frac{1}{2^m}$, was für große m gegen 0 geht.
Der Miller–Rabin–Algorithmus benötigt (bei einmaliger Anwendung)$O^\sim((\log n)^2)$ Bitoperationen unter Verwendung von schneller Multiplikation[7].

2.2.3 Der Solovay–Strassen-Test

Eine andere Variante eines probabilistischen Primzahltests haben D.H. Lehmer von der *University of California* in Berkeley zusammen mit Robert M. Solovay vom *California Institute of Technology* und (unabhängig) Volker Strassen von der *Eidgenössisch Technischen Hochschule Zürich* entwickelt. Qualitativ betrachtet unterscheidet sich das Verfahren nicht vom *Strong Pseudoprimality Test*: Es kann lediglich die Nichtprimheit einer Zahl mit Sicherheit bestätigt werden. Falls

der Test eine Zahl als *"vielleicht prim"* erklärt, so trifft dies (bei einmaliger Durchführung) mit einer Wahrscheinlichkeit von $P \geq \frac{1}{2}$ zu. Solovay und Strassen meinen, ihr Verfahren sei so etwas wie eine "Monte–Carlo–Methode" [2]. Das Verfahren basiert auf dem Legendre–Symbol und dem Jacobi–Symbol, die aufgrund dem Quadratischen Reziprozitätsgesetz schnell zu berechnen sind.

Definition 9 (Quadratzahl modulo m) *Für $n \geq 2$ und $a \in \mathbb{Z}$ mit $ggT(a,n) = 1$ heißt a eine Quadratzahl modulo n oder ein (quadratisches) Residuum modulo n, wenn $a \equiv x^2 \pmod{n}$ für ein $x \in \mathbb{Z}$. Wenn kein derartiges x existiert, wird a im folgenden als Nichtquadratzahl oder (quadratisches) Nichtresiduum bezeichnet.*

Beispiel: Wir betrachten die Quadratzahlen modulo 13 bzw. modulo 15 und modulo 12. Nachdem gilt, dass $x^2 \bmod (m) = (x \pmod m)^2$, müssen für x jeweils nur Werte von $1...m-1$ betrachtet werden.

x	$x^2 \bmod (13)$
1	1
2	4
3	9
4	3
5	12
6	10
7	10
8	12
9	3
10	9
11	4
12	1

x	$x^2 \bmod (15)$
1	1
2	4
3	9
4	1
5	10
6	6
7	4
8	4
9	6
10	10
11	1
12	9
13	4
14	1

x	$x^2 \bmod (12)$
1	1
2	4
3	9
4	4
5	1
6	0
7	1
8	4
9	9
10	4
11	1

Beispiele für Quadratzahlen modulo 13, 15, 12

Im Fall $m = 13$ gibt es 6 Quadratzahlen und 6 Nichtquadratzahlen. Das gilt allgemein für $m = p$ prim. Nachdem $x^2 \equiv (-x)^2 \equiv (p-x)^2 \pmod p$, gibt es maximal $\frac{p-1}{2}$ Quadratzahlen. Andererseits müssen die Quadrate von $1, 2, \ldots \frac{p-1}{2}$ alle verschieden sein. Falls nämlich $x^2 \equiv y^2 \pmod p$ für $1 \leq x \leq y < \frac{p}{2}$, dann gilt $p \mid (y^2 - x^2) = (y+x)(y-x)$. Also muss p mindestens einen der Faktoren teilen, was nur für $x = y$ stimmen kann.
Daraus folgt, dass jede Quadratzahl a modulo p, $a \neq 0$ genau zwei Quadratwurzeln $\{x, p-x\}$ hat. Falls p eine Primzahl ist, stellt es sich als recht einfach heraus, die Quadratzahlen modulo p zu finden:

Lemma (Eulers Kriterium)
Für $p > 2$ prim und $a \in \mathbb{Z}_p^*$ gilt, dass

$$a^{\frac{p-1}{2}} = \begin{cases} 1 & \text{wenn } a \text{ eine Quadratzahl modulo } p \text{ ist} \\ -1 & \text{wenn } a \text{ eine Nichtquadratzahl modulo } p \text{ ist} \end{cases}$$

Die Quadratzahlen modulo p bilden eine Untergruppe von $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ der Ordnung $\frac{p-1}{2}$.

Beweis:
Für die Quadratzahlen $x^{2\cdot i} \in \mathbb{Z}_p^*$ gilt nach dem Kleinen Satz von Fermat $(x^{2\cdot i})^{\frac{p-1}{2}} \equiv (x^i)^{p-1} \equiv 1 \pmod{p}$, für die Nichtquadratzahlen x^{2i+1} dagegen $(x^{2\cdot i+1})^{\frac{p-1}{2}} = x^{\frac{p-1}{2}} \cdot (x^i)^{p-1} \equiv x^{\frac{p-1}{2}} \cdot 1 \pmod{p}$. $x^{\frac{p-1}{2}}$ ist eine Quadratwurzel von 1. Nachdem es in $\mathbb{Z}_p^*$ aber keine nicht trivialen Quadratwurzeln von 1 gibt, muss $x^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ sein, und damit für alle Nichtquadratzahlen $(x^{2\cdot i+1})^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ gelten.
Die Gruppe $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ ist zyklisch. Es gibt also ein erzeugendes Element g, sodass $\mathbb{Z}_p^* = \{1, g, g^2, g^3, \ldots g^{p-2}\}$. Die von g^2 erzeugte Untergruppe der Quadratzahlen mit den Elementen $\{1, g^2, g^4, \ldots\}$ ist ebenfalls zyklisch und hat Ordnung $\frac{p-1}{2}$, denn $(g^2)^{\frac{p-1}{2}} \equiv g^{p-1} \equiv 1 \pmod{p}$.

□

Definition 10 (Legendre–Symbol) *Für eine Primzahl $p \geq 3$ und $a \in \mathbb{Z}$ definiert man:*

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{wenn } a \text{ eine Quadratzahl modulo } p \text{ ist} \\ -1 & \text{wenn } a \text{ eine Nichtquadratzahl modulo } p \text{ ist} \\ 0 & \text{wenn } a \text{ ein Vielfaches von } p \text{ ist} \end{cases}$$

als das Legendre–Symbol von a und n.

Definition 11 (Jacobi–Symbol) *Für eine ganze Zahl $n \geq 3$ mit der Primfaktorzerlegung $n = p_1 p_2 \cdots p_r$ und $a \in \mathbb{Z}$ ist das Jacobi–Symbol von a und n als*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \cdots \left(\frac{a}{p_r}\right)$$

definiert.

Wenn n eine Primzahl ist, stimmt das Jacobi–Symbol von a und n mit dem Legendre–Symbol überein, also kann problemlos dieselbe Notation verwendet werden.
Das Jacobi–Symbol besitzt einige interessante Eigenschaften, wie z.B. die Multiplikativität in beiden Komponenten und den Zusammenhang zwischen $\left(\frac{m}{n}\right)$ und $\left(\frac{n}{m}\right)$, der durch das *Gesetz der Quadratischen Reziprozität* ausgedrückt wird. Im folgenden einige praktische Rechenregeln:

Lemma (Rechenregeln für das Jacobi–Symbol)
Für n, m ungerade ganze Zahlen und $a, b \in \mathbb{Z}$ gilt

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

b) $\left(\frac{a}{n\cdot m}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{a}{m}\right)$

c) $\left(\frac{a+cn}{n}\right) = \left(\frac{a}{n}\right)$ und insbesondere $\left(\frac{a}{n}\right) = \left(\frac{a \pmod{n}}{n}\right)$ für alle $c \in \mathbb{Z}$

d) $\left(\frac{a \cdot b^2}{n}\right) = \left(\frac{a}{n}\right)$ für $ggT(b, n) = 1$

e) $\left(\frac{a}{n \cdot m^2}\right) = \left(\frac{a}{n}\right)$ für $ggT(a, m) = 1$

f) $\left(\frac{2^{2k} \cdot a}{n}\right) = \left(\frac{a}{n}\right)$ für $k \geq 1$

g) $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$

h) $\left(\frac{0}{n}\right) = 0$ und $\left(\frac{1}{n}\right) = 1$

Lemma (Quadratisches Reziprozitätsgesetz)
Für ungerade ganze Zahlen $n, m \geq 3$ gilt

$$\left(\frac{m}{n}\right) = \begin{cases} \left(\frac{n}{m}\right) & \text{wenn } n \equiv 1 \text{ oder } m \equiv 1 \pmod 4 \\ -\left(\frac{n}{m}\right) & \text{wenn } n \equiv 3 \text{ und } m \equiv 3 \pmod 4 \end{cases}$$

Bemerkung
Für eine ungerade Zahl $n \geq 3$ gilt

$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{wenn } n \equiv 1 \text{ oder } n \equiv 7 \pmod 8 \\ -1 & \text{wenn } n \equiv 3 \text{ oder } n \equiv 5 \pmod 8 \end{cases}$$

Die *Beweise* dieser beiden Lemmata und der Bemerkung sind in [7] zu finden.
Diese Eigenschaften ermöglichen einen effizienten Algorithmus für die Berechnung des Jacobi–Symbols $\left(\frac{m}{n}\right)$, der logarithmische Komplexität aufweist [7].

Die Eigenschaften des Jacobi–Symbols und das Kriterium von Euler führen nun zu folgender Charakterisierung von Primzahlen:

Lemma (Testkriterium)
Wenn p eine ungerade Primzahl ist, dann gilt

$$a^{(p-1)/2} \cdot \left(\frac{a}{p}\right) \equiv 1 \pmod p, \text{ für alle } a \in \{1, \ldots p-1\}$$

Umgekehrt besagt das Lemma, dass eine ungerade Zahl $n \geq 3$, für die ein $a \in \{2, \ldots n-1\}$ existiert, sodass $a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \not\equiv 1 \pmod n$, mit Sicherheit zusammengesetzt ist.
Ähnlich wie in Abschnitt 2.1 definiert man daher

Definition 12 (E–Zeuge, E–Lügner) *Sei n eine ungerade, zusammengesetzte Zahl. Eine Zahl a, $1 \leq a < n$ wird E–Zeuge für n genannt, wenn $a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \not\equiv 1 \pmod n$. Andernfalls wird a als E–Lügner bezeichnet.*

Entsprechend kann nun der Primzahltest von Solovay und Strassen formuliert werden:

Algorithmus 6 (Solovay–Strassen–Test)
EINGABE: Ungerade, ganze Zahl $n \geq 3$.
Wähle a zufällig aus $\{2, 3, \ldots n-2\}$
if $(a^{(n-1)/2} \cdot (\frac{a}{n}) \not\equiv 1 \pmod{n})$
  return ''*zusammengesetzt*''
else
  return ''*vielleicht prim*''

Bemerkung:
Für Primzahlen wird immer "wahrscheinlich prim" zurückgegeben. Bei Eingabe einer zusammengesetzten Zahl ist die Wahrscheinlichkeit für die Ausgabe "vielleicht prim"$< \frac{1}{2}$.

Beweis:
Die Aussage über die Wahrscheinlichkeit eines falschen Rückgabewerts lässt sich ähnlich wie beim Fermattest zeigen.
Zunächst ist jeder E–Lügner a auch ein F–Lügner: Falls n zusammengesetzt ist und $a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \equiv 1 \pmod{n}$, dann gilt $\left(\frac{a}{n}\right) \in \{1, -1\}$ und $a^{(n-1)/2} \in \{1, -1\}$. Also ist $a^{n-1} \equiv 1$. Andererseits ist jeder F–Zeuge auch ein E–Zeuge, denn wenn $a^{(n-1)} \not\equiv 1$ ist, ist auch die Quadratwurzel $a^{(n-1)/2} \not\equiv 1 \pmod{n}$. Nachdem das Jacobi–Symbol nur Werte in $\{-1, 0, 1\}$ annimmt, gilt $a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \not\equiv 1 \pmod{n}$.

Die Menge der E–Lügner für n, L_n^E, ist eine Untergruppe von $\mathbb{Z}_n^*$, denn

- Für $a, b \in L_n^E$ gilt $(a \cdot b)^{(n-1)/2} \cdot \left(\frac{a \cdot b}{n}\right) \pmod{n} \equiv (a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \pmod{n}) \cdot (b^{(n-1)/2} \cdot \left(\frac{b}{n}\right) \pmod{n})$
- $1 \in L_n^E$
- Für $a \in L_n^E$ gilt $1 \equiv 1^{(n-1)/2} \cdot \left(\frac{1}{n}\right) \equiv (a \cdot a^{-1})^{(n-1)/2} \cdot \left(\frac{a \cdot a^{-1}}{n}\right) \equiv a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \cdot (a^{-1})^{(n-1)/2} \cdot \left(\frac{a^{-1}}{n}\right) \equiv 1 \cdot (a^{-1})^{(n-1)/2} \cdot \left(\frac{a^{-1}}{n}\right) \equiv (a^{-1})^{(n-1)/2} \cdot \left(\frac{a^{-1}}{n}\right)$.

Um zu zeigen, dass L_n^E eine *echte* Untergruppe von $\mathbb{Z}_n^*$ ist, muss mindestens ein E-Zeuge für n gefunden werden. Dazu unterscheiden wir zwei Fälle:
Fall 1:

> n ist durch p^2 teilbar, wobei $p \geq 3$ eine Primzahl ist. Dann kann folgendermaßen ein F–Zeuge (und damit ein E–Zeuge) konstruiert werden:
> n kann als $n = m \cdot p^k$ geschrieben werden, wobei m ungerade ist und nicht durch p teilbar sein soll. Wenn $m = 1$ gilt, setze $a = 1 + p$. Falls $m \geq 3$, gibt es nach dem Chinesischen Restsatz ein a, $1 \leq a < p^2 \cdot m \leq n$, mit
>
> $$a \equiv 1 + p \pmod{p^2} \text{ und}$$
> $$a \equiv 1 \pmod{m}$$
>
> Wir zeigen jetzt, dass a tatsächlich ein F–Zeuge in $\mathbb{Z}_n^*$ ist. Es gilt in beiden Fällen $a - (1 + p) \equiv 0 \pmod{p^2}$, also teilt p^2 (und p) den Ausdruck $a - (1 + p)$, deshalb ist $a \equiv 1 \pmod{p}$, d.h. $ggT(a, p) = 1$. Außerdem ist der größte gemeinsame Teiler

$ggT(a, m) = 1$, da $a \equiv 1 \pmod{m}$, deshalb gilt $ggT(a, n) = 1$ und $a \in \mathbb{Z}_n^*$.
Es bleibt noch zu zeigen, dass n ein F–Zeuge ist. Für einen Widerspruchsbeweis nehmen wir an, $a^{n-1} \equiv 1 \pmod{n}$. Da $p^2 \mid n$, gilt auch $a^{n-1} \equiv 1 \pmod{p^2}$. Andererseits erhält man unter Verwendung des Binomischen Lehrsatzes

$$a^{n-1} \equiv (1+p)^{n-1} \equiv 1 + (n-1)p + \sum_{2 \le i \le n-1} \binom{n-1}{i} p^i \equiv 1 + (n-1)p \pmod{p^2}$$

Dann müsste $(n-1)p \equiv 0 \pmod{p^2}$ sein, also $p^2 \mid (n-1)p$. Das kann aber nicht sein, denn p teilt nicht $n - 1 = p^k \cdot m - 1$.

Fall 2:

n ist ein Produkt aus lauter verschiedenen Primfaktoren. Wir können $n = p \cdot m$ schreiben, wobei p eine ungerade Primzahl ist und $p \nmid m$. Für eine beliebige Nichtquadratzahl modulo p, $b \in \mathbb{Z}_n^*$, gilt $\left(\frac{b}{p}\right) \equiv -1$. Anwendung des Chinesischen Restsatzes liefert die Existenz eines Elements a, $1 \le a < n$, mit $a \equiv m \pmod{p}$ und $a \equiv 1 \mod(m)$.

Dieses a liegt in $\mathbb{Z}_n^*$ und a ist ein E–Zeuge:
Aus den Bedingungen für a ist klar, dass $ggT(p, a) = 1$ und $ggT(m, a) = 1$. Also ist auch $ggT(n, a) = 1$, und $n \in \mathbb{Z}_n^*$.
Angenommen, n sei E–Lügner. Wenn man das Jacobi–Symbol

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{m}\right) = \left(\frac{b}{p}\right) \cdot \left(\frac{1}{m}\right) = (-1) \cdot 1 = -1$$

unter Verwendung der Rechenregeln auswertet, stellt man fest, dass $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$ sein müsste, um $a^{(n-1)/2} \cdot \left(\frac{a}{n}\right) \equiv 1 \pmod{n}$ zu erhalten. Nachdem m ein Teiler von n ist, muss auch $a^{\frac{n-1}{2}} \equiv -1 \pmod{m}$ gelten. Das widerspricht allerdings der Voraussetzung, dass $a \equiv 1 \pmod{m}$. Also muss a eine E–Zeuge sein.

Die E–Lügner L_n^E bilden also eine echte Untergruppe von $\mathbb{Z}_n^*$, sodass $\#L_n^E \le \frac{1}{2}\#\mathbb{Z}_n^*$, und die Wahrscheinlichkeit, als Basis a einen E–Lügner zu erwischen, $\le \frac{1}{2}$ ist.

□

Daher gilt auch hier, dass die Wahrscheinlichkeit für ein falsches Resultat bei Eingabe einer zusammengesetzten Zahl beliebig reduziert werden kann, indem der Test entsprechend oft wiederholt wird.
Der Berechnungsaufwand des Solovay–Strassen–Algorithmus liegt bei $O^{\sim}((\log n)^2)$ Bitoperationen unter Verwendung von schnellen Multiplikationsmethoden[7].

2.2.4 "Primes is in P"–Test

Unter PRIMES wird das Problem verstanden, die Primalität einer Zahl $n \in \mathbb{N}$ zu bestimmen. Es war lange ungeklärt, ob ein deterministischer Algorithmus existiert, der PRIMES in polynomialer

Zeit löst. Im August 2002 wurde von M. Agrawal, N. Kayal und N. Saxena in dem Paper "PRIMES *is in P*" ein deterministischer Algorithmus mit polynomialer Zeitkomplexität vorgeschlagen.
Beim Fermat–Test, dem *Strong Pseudoprimality Test* und dem Solovay–Strassen–Verfahren wird ein notwendiges Kriterium geprüft, um eine Aussage über die Primalität einer Zahl zu treffen. Für einen deterministischen Algorithmus wird aber ein hinreichendes Kriterium benötigt. Dazu wurde nun Agrawal, Kayal und Saxena folgende Charakterisierung vorgeschlagen:

Lemma:
Für $a < n$, a teilerfremd zu n und n keine exakte Primzahlpotenz gilt

$$n \text{ prim} \Leftrightarrow (X+a)^n \equiv X^n + a \text{ in } \mathbb{Z}_n[X]$$

Beweis
Nach dem Binomischen Lehrsatz gilt

$$(X+a)^n = X^n + \sum_{0<i<n} \binom{n}{i} a^i X^{n-i} + a^n$$

mit dem Binomialkoeffizienten

$$\binom{n}{i} = \frac{n \cdot (n-1) \cdots (n-i+1)}{i \cdot (i-1) \cdots 2 \cdot 1}$$

$\Rightarrow$:

Der Zähler ist $\equiv 0 \pmod{n}$, da er durch n teilbar ist. Nach Voraussetzung ist n aber eine Primzahl, und somit ist der Nenner $\not\equiv 0$, da alle Faktoren teilerfremd zu n sind (weil n ja prim ist). Also gilt $(X+a)^n = X^n + a^n$. Nach dem Satz von Fermat ist $a^n = a^{n-1} \cdot a \equiv 1 \cdot a \equiv a \pmod{n}$.
Damit erhält man $(X+a)^n = X^n + a$ in $\mathbb{Z}_n[X]$.

$\Leftarrow$:

Diese Richtung wird indirekt gezeigt: Angenommen, n ist zusammengesetzt. Dann gibt es einen Primfaktor p von n. Für den Koeffizienten von X^{n-p} gilt $\binom{n}{p} \cdot a^p = \frac{n \cdot (n-1) \cdots (n-p+1)}{p!} \cdot a^p$
Der Faktor n im Zähler ist durch p teilbar, ebenso der Nenner, aber a^p nicht. Der gesamte Ausdruck $\frac{n \cdot (n-1) \cdots (n-p+1)}{p!} \cdot a^p$ ist also nicht durch p, daher auch nicht durch n teilbar.
In $\mathbb{Z}_n[X]$ gilt daher $(X+a)^n \not\equiv X^n + a$, was ein Widerspruch zur Annahme ist. n muss also eine Primzahl sein.

□

Naheliegend wäre nun, für einen Testkandidaten n und ein beliebiges a relativ prim zu n mit dem "repeated squaring"–Verfahren die Koeffizienten des Polynoms $(X+a)^n$ zu berechnen und aus dem Ergebnis auf die Primalität von n zu schließen. Dafür ist jedoch leider ein beträchtlicher Aufwand nötig: Im Zuge der Berechnung können viele Koeffizienten $\neq 0$ auftreten, das im letzten

Durchgang zu quadrierende Polynom hat Grad $\frac{(n)}{2}$ und bis zu $\frac{(n+1)}{2}$ Monome. Die Anzahl der nötigen arithmetischen Operationen liegt in $O(n)$, die Komplexität damit schlechter als bei der Probedivision.

Man berechnet $(X+a)^n$ daher nicht direkt, sondern vergleicht die Reste der beiden Polynome bei Division durch ein anderes Polynom X^r-1, wobei r "geschickt" gewählt wird.
In diesem Fall rechnet man mit Polynomen, deren Grad kleiner als r ist. Wenn außerdem gilt, dass r in $O((\log n)^c)$ bleibt, ist der Rechenaufwand nur polynomial von n abhängig.

Wenn n eine Primzahl ist, gilt $(X+a)^n \equiv X^n + a \pmod{X^r-1}$ für alle r und jedes n.
In umgekehrter Richtung wäre es praktisch, den Ausdruck $(X+a)^n \pmod{X^r-1}$ nur für ein einziges, nicht zu großes r und ein einziges a berechnen zu müssen. Das reicht jedoch leider nicht aus, um die Primalität von n zu bestätigen bzw. zu widerlegen. Man kann allerdings die folgende Aussage verwenden:

Lemma:
Für $n, r \in \mathbb{Z}$ gilt: n ist prim, wenn die folgenden Bedingungen erfüllt sind.

- $n \geq 3$, n ist keine exakte Potenz
- $r < n$ ist eine Primzahl
- Für alle a, $2 \leq a \leq r$ gilt $a \not| n$
- $\mathrm{ord}_r(n) > 4(\log n)^2$
- Für alle a, $1 \leq a \leq 2\sqrt{n}$ gilt $(X+a)^n \equiv X^n + a \pmod{X^r-1}$

Daraus erhält man den folgenden Algorithmus mit Komplexität $O(c \cdot (\log n)^{12} \cdot (\log\log n)^d)$ für Konstanten c, d[7], [9].

Algorithmus 7 (Algorithmus von Agrawal, Kayal und Saxena)
EINGABE: $n \geq 3$, keine exakte Potenz `r=2 while (r<n)`

```
  if (r | n)
    return ‘‘zusammengesetzt’’
  if (r prim)
    if (n^i ≢ 1 (mod r) für alle 1 ≤ i ≤ 4(log n)^2)
      break;
 r++;
if (r == n)
 return ‘‘prim’’;
for (a = 1; a ≤ 2√r · logn)
 if (in Z_n[X])(X + a)^n (mod X^r − 1) ≢ X^n + a (mod X^r − 1))
    return ‘‘zusammengesetzt’’
return ‘‘prim’’
```

2.2.5 Der Lucas–Lehmer–Test

Der Lucas-Lehmer-Test bietet ein vollständiges Kriterium, um festzustellen, ob eine Mersenne–Zahl prim ist.
Wie in Abschnitt 1.1.4 gezeigt wurde, können Mersenne-Zahlen $M_n = 2^n - 1$ nur dann prim sein, wenn n eine Primzahl ist. Weiters gilt

Lemma (Lucas–Lehmer–Kriterium)
Für eine Primzahl n setzt man die Lucas–Lehmer–Folge

$$s_0 = 4$$
$$s_{k+1} = s_k^2 - 2 \text{ für } k = 0, \ldots n-2$$

M_n ist genau dann eine Primzahl, wenn s_{n-2} durch M_n teilbar ist.

Der *Beweis* ist in [10] zu finden.
Entsprechend kann ein Algorithmus formuliert werden:

Algorithmus 8 (Lucas–Lehmer–Test)
EINGABE: Ungerade, ganze Zahl $n \geq 3$.

```
s=4;
for (int i = 0; i ≤ n − 2; i++)
   s=s*s-2
if (s rem(2^n − 1) == 0)
  return ‘‘prim’’;
else
  return ‘‘zusammengesetzt’’;
```

3

Anwendungen

3.1 Public Key Cryptography und RSA

Unter Kryptographie versteht man die Wissenschaft der Verschlüsselung von Nachrichten zur vertraulichen Kommunikation.

3.1.1 Symmetrische vs. Asymmetrische Verfahren

Symmetrische Verfahren

Die Verwendung von symmetrischen Verfahren war über Jahrhunderte die einzig bekannte Möglichkeit zur Verschlüsselung von Nachrichten: Man verwendet dabei einen Schlüssel, um eine Nachricht zu "codieren". Die verschlüsselte Nachricht und der Schlüssel werden auf getrennten Wegen zum Empfänger geschickt, und der Empfänger benutzt den Schlüssel, um die Nachricht zu dechiffrieren. Der Schlüssel für Ver- **und** Entschlüsselung ist also derselbe und muss beiden Kommunikationspartnern bekannt sein. Die Sicherheit des Verfahrens hängt daher maßgeblich von der sicheren Übertragung des Schlüssels ab.

Wenn man zur Verdeutlichung des symmetrischen Verfahrens als Analogie die verschlüsselte Nachricht als Botschaft in einer Kiste und den Schlüssel als Schlüssel für die Kiste betrachtet, funktioniert das Prinzip folgendermaßen: die in der Kryptographie mittlerweile berühmte Protagonistin Alice legt ihre Nachricht in die Kiste und versperrt die Kiste. Dann werden Kiste und Schlüssel auf getrennten Wegen zu Bob, ihrem ebenso bekannten Kommunikationspartner, geschickt, der die Kiste öffnet. Es besteht aber die Gefahr, dass eine neugierige Spionin wie Eve Kiste und Schlüssel abfängt und die Botschaft ebenfalls lesen kann.

Asymmetrische Verfahren

1976 präsentierten Diffie und Hellman ein asymmetrisches Verfahren zur Verschlüsselung von Nachrichten, dessen Sicherheit nicht vont der sicheren Schlüsselübertragung abhängt.
Die Idee des asymmetrischen Verfahrens besteht darin, zwei verschiedene Schlüssel zu verwenden: ein sogenannter öffentlicher Schlüssel dient zum Verschlüsseln der Nachricht. Zum Entschlüsseln verwendet der Empfänger der Nachricht einen privaten Schlüssel, der nur ihm bekannt ist.

In der Analogie mit Kiste und Schlüssel bedeutet das folgendes: Wenn Bob von Alice eine vertrauliche Nachricht erhalten will, schickt Bob an Alice auf öffentlichem Weg eine offene Kiste sowie den öffentlichen (in der folgenden Grafik roten) Schlüssel. Alice legt ihre Nachricht in die Kiste und versperrt die Kiste mit dem öffentlichen Schlüssel. Dann wird die Kiste an Bob gesandt, der den privaten (blauen) Schlüssel verwendet, um die Kiste zu öffnen. Eve kann die Kiste natürlich trotzdem abfangen. An den Schlüssel zum Entsperren kommt sie aber schwer heran – er wird schließlich nicht übertragen. Falls die Kiste mit einem leichten Schloss versehen ist, könnte Eve der Botschaft vielleicht mit allerhand Werkzeugen zu Leibe rücken. Bei einem ausgefeilten Tresor mit stabilem Schloss wird das aber erhebich schwieriger, wenn die Nachricht nicht zerstört werden soll.
Um dieses Schloss möglichst unzerstörbar zu halten, werden nun beim RSA–Algorithmus "sehr große" Primzahlen verwendet.

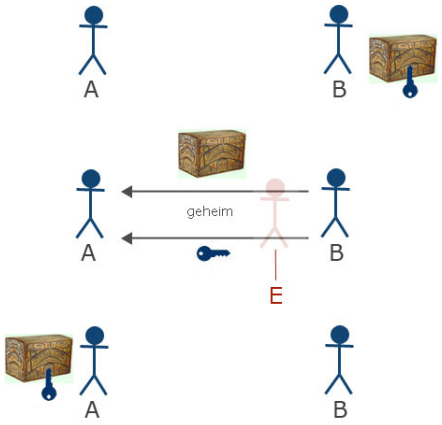

Abbildung 3.1: Prinzip der symmetrischen Verschlüsselung

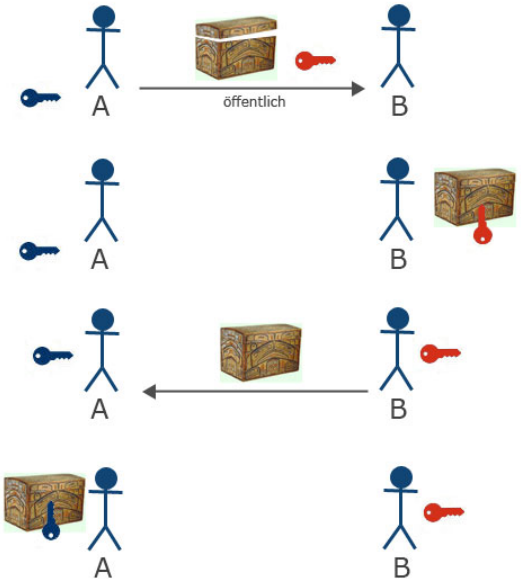

Abbildung 3.2: Prinzip der asymmetrischen Verschlüsselung

3.1.2 Das RSA–Verfahren

1978 entwickelten Rivest, Shamir und Adleman den mittlerweile berühmten RSA–Algorithmus.

Die Vorgangsweise zur Konstruktion des öffentlichen und privaten Schlüssels ist die folgende:

a) Wähle zwei "große" Primzahlen p, q wobei $p \neq q$

b) Berechne die Zahl $N = p \cdot q$, sowie $\varphi(N) = (p-1)(q-1)$

c) Wähle $e \in \{2, \ldots, \varphi(N) - 2\}$ relativ prim zu $\varphi(N)$

d) Veröffentliche (N, e) als Public–Key

e) Berechne $d \in \{2, \ldots, \varphi(N) - 2\}$, sodass $e \cdot d \equiv 1 \pmod{\varphi)(N}$. Dazu kann der Erweiterte Euklidische Algorithmus folgendermaßen verwendet werden: Für $ggT(e, \varphi(N)) = 1$ wird eine Linearkombination $1 = e \cdot d + k \cdot \varphi(N)$ berechnet. Damit ist $1 \equiv e \cdot d \pmod{\varphi(N)}$

f) Verheimliche (N, d) als Private–Key

Wenn Alice Bob eine Nachricht schicken möchte, verschlüsselt sie ihre Botschaft m (in Form einer Zahl) mit dem öffentlichen Schlüssel N und e wie folgt:
$m \rightsquigarrow c = m^e \pmod{N}$
Alice schickt die verschlüsselte Botschaft c an Bob, der seinen privaten Schlüssel benutzt, um die Nachricht zu erhalten:
$c \rightsquigarrow m^* = c^d \pmod{N}$

Unter Verwendung des Satzes von Euler wird klar, dass tatsächlich $m = m^*$ gilt:

$$\begin{aligned} m^* &\equiv c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \cdot m^{ed-1} \equiv m \cdot m^{\frac{ed-1}{\varphi(N)}\varphi(N)} \\ &\equiv m(m^{(\ldots)})^{\varphi(N)} \equiv m \cdot 1 \equiv m \pmod{N} \end{aligned}$$

Der RSA-Algorithmus kann auch zur Authentifizierung verwendet werden: Wenn Bob eine Nachricht m signieren und an Alice schicken möchte, verwendet er seinen privaten Schlüssel d, um $c = m^d$ zu berechnen. Anschließend schickt er c an Alice, die mit dem öffentlichen Schlüssel e die Botschaft berechnet: $m = c^e$.

Sicherheit der RSA–Verschlüsselung

Falls Eve die geheime Nachricht abfängt, kann sie zunächst nichts mit dieser anfangen, da sie nicht über den privaten Schlüssel d verfügt.
Sie könnte allerdings versuchen, aus dem öffentlichen Schlüssel N und e auf d zu schließen, indem sie N faktorisiert. Dann wären ihr p, q und daraus folgend $\varphi(N)$ bekannt, und sie könnte wie Bob bei der Konstruktion des Schlüssels den Erweiterten Euklidischen Algorithmus verwenden, um d zu finden.
Der kritische Punkt bei der Sicherheit von RSA ist also die Effizienz momentan verfügbarer Faktorisierungsalgorithmen.

4

System

4.1 Mathematica

4.1.1 Nützliche Funktionen in Mathematica

Die folgenden Funktionen wurden bei der Realisierung des RSA–Algorithmus in MATHEMATICA verwendet. Für weitere Details zur Implementierung der einzelnen Funktionen siehe [11].

`Prime[n]`
: liefert die n-te Primzahl. Mathematica verwendet dazu "sparse caching and sieving". Für große n werden Schätzungen mit Hilfe der Primzahldichte vorgenommen und der Lagarias-Miller-Odlyzko-Algorithmus verwendet.

`PrimeQ[n]`
: überprüft, ob n eine Primzahl ist. Dazu wird zunächst Probedivision verwendet, um die Teilbarkeit durch kleine Faktoren zu überprüfen. Dann wird der Miller–Rabin *Strong Pseudoprimality Test* verwendet, und schließlich der Lucas–Test.

`Mod[m, n]`
: liefert den Rest der Division m/n, das Ergebnis liegt also in der Menge $\{0, n-1\}$.

`PowerMod[a, b, c]`
: liefert den Rest der Division a^b/c. Es arbeitet für große Zahlen aber wesentlich effizienter als `Mod[m, n]`, weil die Berechnung von a^b durch schrittweise Multiplikation und Restberechnung erfolgt. Für negative b wird das modulare Inverse zurückgegeben, falls es existiert.

`EulerPhi[n]`
: liefert als Ergebnis $\varphi(n)$, also die Kardinalität der Menge aller $m < n$, mit m teilerfremd zu n.

`GCD[n1, n2, ...]`
: liefert den größten gemeinsamen Teiler der Zahlen $n1, n2, \ldots$.

`ExtendedGCD[n1, n2, ...]`
: ist eine Implementierung des erweiterten Euklidischen Algorithmus, der den größten gemeinsamen Teiler der Parameter als Linearkombination berechnet. Es wird eine Liste $\{g, \{r_1, r_2, \ldots\}\}$ zurückgegeben. g ist der größte gemeinsame Teiler von $n_1, n_2, \ldots$, und es gilt $g = r_1 \cdot n_1 + r_2 \cdot n_2 + \ldots$.

In MATHEMATICA sind weitere Funktionen implementiert, die im Abschnitt über Algorithmen erwähnt wurden:

`PrimeQ[n]`
: überprüft, ob n eine Primzahl ist. Dazu wird zunächst Probedivision verwendet, um die Teilbarkeit durch kleine Faktoren zu überprüfen. Dann wird der Miller–Rabin *Strong Pseudoprimality Test* verwendet, und schließlich der Lucas-Test.

`Mod[m, n]`
: liefert den Rest der Division m/n, das Ergebnis liegt also in der Menge $\{0, n-1\}$.

`FactorInteger[n]`
: gibt die Primfaktoren von n und deren Exponenten als Liste zurück. Der Algorithmus sollte für n mit bis zu 50 Dezimalstellen ohne Probleme funktionieren.

`JacobiSymbol[m,n]`
: gibt den Wert des Jacobi–Symbols $\left(\frac{m}{n}\right)$ bzw. des Legendre-Symbols (für eine Primzahl m) zurück.

`PrimePi[n]`
: gibt $\pi(n)$ zurück, also die Anzahl der Primzahlen, die kleiner oder gleich n sind.

4.1.2 Implementierung von RSA in Mathematica

Im Folgenden wurde der RSA-Algorithmus in MATHEMATICA implementiert:

```
p=Prime[Random[Integer,{10^7, 10^8}]]
q=Prime[Random[Integer,{10^7, 10^8}]]
n=p q
phi = EulerPhi[n]
e = Random[Integer,{2,phi}];
While[GCD[e,phi] ≠ 1, e = Random[Integer, {2,phi-1}]]
d = Mod[ExtendedGCD[e,phi][[2]][[1]], phi]
cryptmessage = PowerMod[MESSAGE, e, n]
decryptmessage = PowerMod[cryptmessage, d, n]
```

Literaturverzeichnis

[1] J. Derbyshire. *Prime Obsession*. Plume, London u.a., 2004.

[2] C. Pomerance. *Primzahlen im Schnelltest*. Spektrum der Wissenschaft, 1983.

[3] *Distributed Search for Fermat Number Divisors*. Internet. http://www.fermatsearch.org (accessed 22.12.04).

[4] S. Bosch. *Algebra*. Springer, Berlin u.a., 2004.

[5] *Satz von Euler*. Internet. http://www.matheboard.de/lexikon/Satz_von_Euler,definition.htm (accessed 22.10.04).

[6] *Great Internet Mersenne Prime Search (GIMPS)*. Internet. http://www.mersenne.org (accessed 22.12.04).

[7] M. Dietzfelbinger. *Primality Testing in Polynomial Time*. Springer, Berlin u.a., 2004.

[8] J. von zur Gathen und J. Gerhard. *Modern Computer Algebra*. Cambridge University Press.

[9] N.Saxena und N. Kayal M. Agrawal. PRIMES *is in* P. Internet. http://www.cse.iitk.ac.in/news/primality.pdf (accessed 22.1.05).

[10] M. E. Pohst. *Der Lucas–Lehmer Test*. 2004. Internet. http://www.math.tu-berlin.de/~kant/Mersenne/MersenneVortrag.pdf (accessed 21.12.04).

[11] MATHEMATICA *5.1 Documentation*. Internet. http://documents.wolfram.com/mathematica/ (accessed 17.12.04).