Mathematics Subject Classification: 11Y05, 11Y11, 11A51, 11Y16
Keywords: factorization, algorithms, prime numbers.

Faktorisieren ganzer Zahlen

G. Nebe, Aachen

1 Einleitung

Dieser allgemein gehaltene Übersichtsartikel entstand aus meinem Habilitationsvortrag an der RWTH Aachen. Er soll die modernen Faktorisierungsverfahren kurz darstellen. Die meisten schönen Ideen findet man schon in den klassischen Algorithmen, die dazu entwickelt wurden, Zahlen mit Bleistift und Papier zu faktorisieren. Viele aktuelle, für den Computer entwickelte Faktorisierungsalgorithmen bauen auf diesen Ideen auf.

Schon Euklid wußte, daß sich jede natürliche Zahl als Produkt von Primzahlen darstellen läßt. Diese *vollständige Faktorisierung* ist bis auf die Reihenfolge der Faktoren eindeutig. Streng bewiesen wurde dieser *Fundamentalsatz der Arithmetik* erst von Gauß [6], der das Problem der Berechnung der Faktorisierung als eines der Hauptprobleme der Arithmetik bezeichnet hat. Gauß sah, daß das Faktorisieren sich in 2 Schritte gliedert:
(1) Man teste, ob die zu faktorisierende Zahl zusammengesetzt ist.
(2) Je nach dem Ergebnis von (1) faktorisiere man die zusammengesetzte Zahl oder beweise, daß sie eine Primzahl ist.

Zum ersten Schritt gibt es sehr schnelle Methoden (siehe Abschnitt 2.1) die als Resultat entweder "ist zusammengesetzt" oder "ist sehr wahrscheinlich eine Primzahl" liefern. Für viele Anwendungen genügt es, die Primzahleigenschaft mit hoher Wahrscheinlichkeit zu wissen. Aber auch für rigorose Primzahlbeweise gibt es recht gute Algorithmen (Abschnitt 2.2). Während es beim Faktorisieren genügt, einen nichttrivialen Teiler zu raten, ist hier ein echter Beweis gefragt.

Weiß man nach (1), daß die Zahl N zusammengesetzt ist, so will man in der Regel eine *Faktorisierung* oder *Zerlegung* von N finden, also

$$N = a \cdot b$$

als Produkt zweier ganzer Zahlen $a, b \neq \pm 1$ schreiben. a und b sind meist einfacher zu faktorisieren als N und rekursiv erhält man eine vollständige Faktorisierung von N. Auch das Faktorisieren selbst geschieht in mehreren Etappen. Die Wahl des Verfahrens hängt stark von den Natur der Zahl N ab. Zum Abspalten kleiner Teiler von N werden die Methoden aus Abschnitt 3 angewandt. Für einige Algorithmen ist es auch von Vorteil, wenn man N als Wert eines "kleinen" Polynoms schreiben kann.

In diesem Artikel kann nicht auf alle bekannten Methoden eingegangen werden und die Algorithmen werden oft nur in einer Rohform vorgestellt, die das Erkennen des groben Ablaufs erleichtert. Für detailliertere Beschreibungen aller wichtigen Faktorisierungsalgorithmen und weitere Literaturangaben sei auf die Lehrbücher [4], [14], [15] und [16] sowie die Übersichtsartikel [3], [5] und [9] verwiesen. Ich bedanke mich bei C. Pomerance, der mir relevante Kapitel des Buches [14] vor Drucklegung zur Verfügung gestellt hat.

2 Primzahltests

2.1 Zusammengesetzheits-Tests

Der *kleine Satz von Fermat* sagt aus, daß für jede Primzahl p und jede zu p teilerfremde Zahl a die Kongruenz

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

erfüllt ist. Algebraisch formuliert heißt das, daß der Exponent der multiplikativen Gruppe $(\mathbf{Z}/p\mathbf{Z})^*$ ein Teiler von $p-1$ ist. Einige zusammengesetzte Zahlen N, die sogenannten *Carmichael-Zahlen*, haben auch die Eigenschaft, daß die Ordnung eines jeden Elements von $(\mathbf{Z}/N\mathbf{Z})^*$ ein Teiler von $N-1$ ist. Die kleinste Carmichael-Zahl ist $561 = 3 \cdot 11 \cdot 17$. Es gilt jedoch, daß eine ungerade Zahl N eine Primzahl ist, wenn für alle teilerfremden Zahlen a die Kongruenz

$$\text{(JK)} \quad a^{(N-1)/2} \equiv \left(\frac{a}{N}\right) \pmod{N}$$

gilt, wobei $\left(\frac{\cdot}{N}\right) = \prod_{i=1}^{s} \left(\frac{\cdot}{p_i}\right)^{a_i} : (\mathbf{Z}/N\mathbf{Z})^* \to \{\pm 1\}$ für $N = \prod_{i=1}^{s} p_i^{a_i}$ das Jacobi-Kronecker-Symbol ist. Ist p eine Primzahl und a nicht durch p teilbar, so ist $\left(\frac{a}{p}\right) = 1$, falls a ein Quadrat modulo p ist und -1 sonst. Das Jacobi-Kronecker-Symbol läßt sich mit Hilfe des quadratischen Reziprozitätsgesetzes leicht ausrechnen (siehe [16, Appendix 3]). Ist N zusammengesetzt, so erfüllen mindestens die Hälfte aller Zahlen $a \in \{1, \ldots, N-1\}$ nicht die Kongruenz (JK). Erfüllt N also die Kongruenz (JK) für 50 zufällig ausgewählte Zahlen a, so ist die Wahrscheinlichkeit, daß N zusammengesetzt ist, ungefähr $2^{-50} < 10^{-15}$ und damit im allgemeinen kleiner als die Wahrscheinlichkeit eines Hardware-Fehlers.

Einen schnelleren Test, den sogenannten *Rabin-Miller-Test*, der auch die Berechnung der Jacobi-Kronecker-Symbole vermeidet, liefert das folgende Kriterium: Sei N ungerade und $N - 1 = d2^s$ mit ungeradem d. Dann ist N eine Primzahl, falls für alle zu N teilerfremden Zahlen a entweder $a^d \equiv 1 \pmod{N}$ gilt oder $a^{d \cdot 2^r} \equiv -1 \pmod{N}$ für ein $r = 0, 1, \ldots, s-1$. Ist N zusammengesetzt, so wird diese Bedingung nur von höchstens einem Viertel der Zahlen $a \in \{1, \ldots, N-1\}$ erfüllt.

2.2 Primzahlbeweise

Um zu beweisen, daß N eine Primzahl ist, kann man dieselbe Idee wie im vorangegangenen Abschnitt benutzen: Ist $N = p_1^{a_1} \dots p_s^{a_s}$ für paarweise verschiedene Primzahlen p_i, so ist nach dem chinesischen Restsatz der Ring $\mathbf{Z}/N\mathbf{Z}$ isomorph zu $\oplus_{i=1}^{s} \mathbf{Z}/p_i^{a_i}\mathbf{Z}$ und damit ist auch die Einheitengruppe $(\mathbf{Z}/N\mathbf{Z})^*$ das direkte Produkt der Gruppen $(\mathbf{Z}/p_i^{a_i}\mathbf{Z})^*$. Insbesondere ist der Exponent von $(\mathbf{Z}/N\mathbf{Z})^*$ genau dann durch $N-1$ teilbar, wenn N eine Primzahl ist. Kann man also $N-1$ primfaktorisieren, so bietet sich folgender schneller Primzahltest an:

(1) Sei $N - 1 = q_1^{b_1} \dots q_r^{b_r}$ mit verschiedenen Primzahlen q_j.

(2) Gibt es für alle $j = 1, \dots, r$ Zahlen $c_j \in \{2, \dots, N-1\}$, für welche $c_j^{(N-1)/q_j} \not\equiv 1 \pmod{N}$ und $c_j^{N-1} \equiv 1 \pmod{N}$ ist, so ist die Ordnung von $c_j \in (\mathbf{Z}/N\mathbf{Z})^*$ durch $q_j^{b_j}$ teilbar. Deshalb teilt $N-1$ den Exponenten von $(\mathbf{Z}/N\mathbf{Z})^*$ und N ist eine Primzahl.

Dieser Test setzt eine (fast) vollständige Faktorisierung von $N-1$ voraus, wie sie leider nicht für alle Primzahlkandidaten N erreicht werden kann. Eine berühmte Folge von Zahlen, für die man $N-1$ leicht faktorisieren kann, sind die *Fermat-Zahlen* $F_n := 2^{2^n} + 1$, für die Fermat vermutet hat, sie seien Primzahlen. Es gilt, daß $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$ und $F_4 = 65537$ Primzahlen sind, doch schon 1732 fand Euler den Teiler 641 von F_5. Man kennt bis heute keine weitere Fermat-Primzahl. Das kleinste n, für das man nicht weiß, ob F_n zusammengesetzt ist, ist $n = 31$ (Internetseite [FERMAT]).

Die größten bekannten Primzahlen sind jedoch *Mersenne-Primzahlen* $M_p := 2^p - 1$. Da jeder Teiler d von p den Teiler $X^d - 1$ von $X^p - 1$ in $\mathbf{Z}[X]$ liefert, kann M_p nur für Primzahlen p eine Primzahl sein. Man kennt die ersten 37 Zahlen p, für die M_p eine Primzahl ist - die 37te Mersenne-Primzahl ist $2^{3021377} - 1$. Erst kürzlich wurde die noch größere Mersenne-Primzahl $M_{6972593}$, die mehr als 2 Millionen Dezimalstellen hat, entdeckt, wobei allerdings nicht bekannt ist, ob es dazwischen noch weitere Mersenne-Primzahlen gibt (Internetseite [PRIME]). Die Mersenne-Zahlen haben die Eigenschaft, daß man eine Faktorisierung von $M_p + 1$ (jedoch i.a. nicht von $M_p - 1$) kennt. Um die Faktorisierung von $N+1$ zum Beweis der Primzahleigenschaft auszunutzen, sucht man wieder eine abelsche Gruppe, in der man leicht rechnen kann, und deren Exponent genau dann $N+1$ ist, wenn N eine Primzahl ist. Dazu betrachtet man eine quadratische Erweiterung Q von $\mathbf{Z}/N\mathbf{Z}$ und potenziert in der Faktorgruppe $Q^*/(\mathbf{Z}/N\mathbf{Z})^*$ ([16, S. 107 ff]).

Indem man elliptische Kurven modulo N betrachtet, kann man viele weitere abelsche Gruppen ganz unterschiedlicher Ordnung benutzen, um zu beweisen, daß N prim ist. Dazu sei N teilerfremd zu 6. Man wähle $A, B \in \mathbf{Z}/N\mathbf{Z}$, so daß $4A^3 + 27B^2$ teilerfremd zu N ist. Da man annimmt, daß N prim ist, wird

mit N wie mit einer Primzahl gerechnet. Dann ist

$$E_N = E_N(A,B) = \{[x,y,z] \in \mathbf{P}^2(\mathbf{Z}/N\mathbf{Z}) \mid y^2z \equiv x^3 + Axz^2 + Bz^3 \pmod N\}$$

eine Gruppe. Das Einselement ist $[0,1,0]$. Nach einem Satz von Hasse ist

$$(\sqrt{N}-1)^2 < |E_N| < (\sqrt{N}+1)^2$$

(falls N eine Primzahl ist). Beim *Goldwasser-Kilian-Test* sucht man in E_N ein Element $P = [x,y,z]$ mit $\mathrm{ggT}(z,N) = 1$, welches $P^q = 1 = [0,1,0]$ für eine Primzahl $q > (N^{1/4}+1)^2$ erfüllt. Existiert solch ein Element, dann ist N eine Primzahl: Denn sonst hat N einen Primteiler $p \le \sqrt{N}$. In der Reduktion E_p von E_N modulo p hat das Element P immer noch Ordnung q. Damit ist $|E_p| \ge q > (\sqrt{p}+1)^2$, was dem Satz von Hasse widerspricht. Eine Schwierigkeit bei diesem Test ist es, die Primzahl q zu finden. Deshalb betrachtet man nicht beliebige elliptische Kurven E_N, sondern nur solche mit komplexer Multiplikation. Für diese läßt sich nämlich die Gruppenordnung $|E_N|$ leicht berechnen und man hofft, einen geeigneten Primteiler q von $|E_N|$ zu finden. Die Primzahleigenschaft der meist sehr viel kleineren Zahl q kann man wieder mit geeigneten elliptischen Kurven beweisen. Man erhält so ein *Primzahlzertifikat*, in dem die verwendeten Daten $(N, E_N, |E_N|, q, P)$, $(q, E_q, |E_q|, \ldots), \ldots$ angegeben werden, mit dem sich die wesentlichen Schritte des Primzahlbeweises leicht überprüfen lassen.

3 Finden kleiner und spezieller Teiler

3.1 Trial Division

Den *Trial Division* Algorithmus hat wohl jeder schon einmal angewandt, um durch Kopfrechnen kleine Primfaktoren einer Zahl N zu finden. Die Grundidee ist es, N sukzessive durch die Primzahlen $2, 3, 5, 7, 11, \ldots$ zu teilen.

Dabei kann man alle Primzahlen, die kleiner als eine feste Schranke S sind, zum Beispiel mit dem *Sieb des Eratosthenes* bestimmen. Dazu schreibt man die Zahlen $2, \ldots, S$ in eine Liste. Das erste Element der Liste ist eine Primzahl und wird zur Primzahlliste hinzugefügt. Streicht man alle Vielfachen dieser Primzahl (einschließlich der Zahl selber) aus der Liste, so ist wieder das erste Element eine Primzahl, etc.

Anstelle der Primzahlen kann man auch eine leichter zu berechnende Folge von Zahlen benutzen, die die Primzahlen als Teilfolge besitzt, also z.B. die Folge $2, 3, 6k \pm 1$, mit $k \in \mathbf{N}$.

Ein Problem bei der Trial Division ist es, daß man die i.a. recht große Zahl N häufig vergeblich durch kleine Primzahlen zu teilen versucht. Dieses kann man abkürzen, indem man vorher Produkte von Primzahlen, also z.B.

$$P_i := \prod_{\substack{(i-1)\cdot 100 < p < i\cdot 100 \\ p \text{ prim}}} p$$

berechnet und für jedes P_i, den größten gemeinsamen Teiler $\mathrm{ggT}(N, P_i)$ von N und P_i mit dem Euklidischen Algorithmus bestimmt.

Der *Euklidische Algorithmus* wird in vielen Faktorisierungsmethoden als Hilfsmittel benutzt. Er ergibt sich aus der Beobachtung, daß für $N = qP + r$ mit $N, P, r, q \in \mathbf{Z}$ die Beziehung $\mathrm{ggT}(N, P) = \mathrm{ggT}(P, r)$ gilt.

3.2 Der Pollard-ρ-Algorithmus

Der *Pollard-ρ-Algorithmus* hat seinen Namen zum einen von seinem Entdecker, J. Pollard [12], zum anderen von seinem Verhalten: Sei $f \in \mathbf{Z}[X]$ ein Polynom mit ganzen Koeffizienten (z.B. $f(X) = X^2+1$) und $x_0 \in \{0, \ldots, N-1\}$. Die Folge $x_n := f(x_{n-1}) \pmod N$ wird nach einer Vorperiode (dem Aufstrich des Buchstabens ρ) schließlich periodisch, was durch den Kreis im ρ veranschaulicht werden kann. Ist f zufällig gewählt, so erwartet man, daß sowohl die Länge des Aufstrichs als auch der Umfang des Kreises in etwa $\sqrt{N}$ betragen. Ist p ein Teiler von N, so wird die Folge $x_n \pmod p$ i.a. sehr viel eher periodisch mit einer wesentlich kürzeren Periode, sagen wir der Länge k, d.h. der $\mathrm{ggT}(x_n - x_{n+k}, N)$ ist mit großer Wahrscheinlichkeit ein nichttrivialer Teiler von N. Da man weder die Vorperiode noch die Periode der Folge $x_n \pmod p$ kennt und nicht jedes Paar $(n, n+k)$ betrachten möchte, berechnet man zwei Folgen x_n und $y_n := x_{2n}$ parallel und testet in jedem Schritt, ob der $\mathrm{ggT}(x_n - y_n, N)$ ein nichttrivialer Teiler von N ist. Auch hier kann man einige ggT-Berechnungen sparen, indem man direkt das Produkt über mehrere $x_n - y_n$ betrachtet. Es ist erstaunlich, wie schnell man mit diesem Algorithmus, den man sehr leicht z.B. mit MAPLE programmieren kann, kleine Primteiler von N findet.

Pollard-ρ-Algorithmus

(0) Wähle ein Polynom $f(X) \in \mathbf{Z}[X]$.

(1) Wähle $x = y$ zufällig in $\{0, \ldots, N-1\}$.

(2) Setze $x := f(x) \pmod N$, $y := f(y) \pmod N$, $y := f(y) \pmod N$.

(3) Berechne $d := \mathrm{ggT}(x - y, N)$.
Ist $d = N$, so wiederhole das Verfahren mit einen neuen Startwert in (1).
Sonst setze $N := N/d$ und fahre bei (2) fort.

3.3 Der Pollard-$(p-1)$-Algorithmus

Der *Pollard-$(p-1)$-Algorithmus* ist ein Algorithmus, der Primfaktoren von N findet, für die $p-1$ eine glatte Zahl ist. Sei $S \in \mathbf{N}$ fest. Dann nennt man eine Zahl *S-glatt*, wenn all ihre Primteiler $\leq S$ sind und *S-potenzglatt*, wenn alle Primzahlpotenzen, die die Zahl teilen, $\leq S$ sind. Sei V das kleinste gemeinsame Vielfache der Zahlen $\{1, \ldots, S\}$. Hat N einen Primteiler p, für den $p-1$ eine S-potenzglatte Zahl ist, so ist nach dem kleinen Satz von Fermat p ein Teiler

von $\mathrm{ggT}(a^V - 1, N)$ für alle zu N teilerfremden Zahlen a.

Pollard-$(p-1)$-Algorithmus

(0) Wähle Schranken $S < S'$. Bestimme die Primzahlen $p_1 < \ldots < p_r \leq S$ z.B. mit dem Sieb des Eratosthenes und für jede dieser Primzahlen das maximale $e_i \in \mathbf{N}$ mit $p_i^{e_i} \leq S$.

(1) Wähle $a \in \{2, \ldots, N-2\}$, z.B. $a = 2$.

(2) Für $i = 1, \ldots, r$ berechne $a := a^{(p_i^{e_i})} \pmod N$.

(3) Bestimme $d = \mathrm{ggT}(a - 1, N)$.

(4) Ist $d = 1$, so bestimme die Primzahlen $p_{r+1} < \ldots < p_s$ in $(S, S']$ und speichere die Differenzen $d_i := p_i - p_{i-1}$ $(i = r+1, \ldots, s)$ ab. Sei $b := a$. Für $i = r+1, \ldots, s$ setze $a := ab^{d_i}$. Bestimme $d = \mathrm{ggT}(a - 1, N)$.

In der Praxis ist es sinnvoll, während der Berechnung von a^V in Schritt (2) gelegentlich den $\mathrm{ggT}(a-1, N)$ (Schritt (3)) zu bestimmen. Schritte $(1)-(3)$ finden das Produkt der Primteiler p von N, für die $p-1$ eine S-potenzglatte Zahl ist. Ist $p - 1 = qn$ für eine Primzahl $S < q \leq S'$ und eine S-potenzglatte Zahl n, so wird p durch den zusätzlichen Schritt (4) gefunden.

Wie schon bei den Primzahltests kann man den $(p-1)$-Algorithmus auch auf $(p+1)$ [21] ausdehnen. Mit Hilfe von elliptischen Kurven erhält man jedoch Gruppen beliebiger Ordnung zwischen $(\sqrt{p}-1)^2$ und $(\sqrt{p}+1)^2$:

3.4 Die Elliptische-Kurven-Methode

Die *Elliptische-Kurven-Methode* [7] ist eines der drei wichtigsten Arbeitspferde zum Faktorisieren großer Zahlen. Sie stellt den einzigen modernen Faktorisierungsalgorithmus dar, dessen Laufzeit wesentlich von der Größe des zweitgrößten Primteilers von N abhängt. Mit ihr können deshalb sehr viel größere Zahlen faktorisiert werden als mit den beiden anderen wichtigen Verfahren, dem Quadratischen Sieb (Abschnitt 4.3) und dem Zahlkörpersieb (Abschnitt 4.4), sofern ihre Primteiler relativ klein sind. Wie schon beim Primzahltest rechnet man in elliptischen Kurven E_N modulo N. Da N zusammengesetzt ist, bildet die Menge E_N keine Gruppe mehr: Kann man eine Gruppenoperation nicht ausführen, so hat man einen nichttrivialen Teiler von N gefunden. Daraus ergibt sich der folgende Algorithmus, der analog zum Pollard $(p-1)$-Algorithmus abläuft:

Elliptische-Kurven-Methode

(0) Wähle Schranken $S < S'$. Bestimme die Primzahlen $p_1 < \ldots < p_r \leq S$ und für jede dieser Primzahlen das maximale $e_i \in \mathbf{N}$ mit $p_i^{e_i} \leq S$.

(1) Wähle zufällige $x, y, A \in \{0, \ldots, N-1\}$.
Setze $B := (y^2 - x^3 - Ax) \pmod N$.
Ist $d := \mathrm{ggT}(4A^3 + 27B^2, N)$ ein nichttrivialer Teiler von N, so gibt d aus.
Ist $d = N$, so wähle neue x, y, A.
Setze $P := [x, y, 1] \in E_N := E_N(A, B)$.

(2) Rechne in der elliptischen Kurve E_N:
Für $i = 1, \ldots, r$ berechne $P := P^{(p_i^{e_i})} = [p_i^{e_i}]P \in E_N$, und brich ab, falls während der Berechnung ein nichttrivialer Teiler d von N gefunden wird.

(3) Bestimme die Primzahlen $p_{r+1} < \ldots < p_s$ in $(S, S']$ und speichere die Differenzen $d_i := p_i - p_{i-1}$ $(i = r+1, \ldots, s)$ ab.
Sei $Q := P$. Für $i = r+1, \ldots, s$ berechne $P := PQ^{d_i} = P + [d_i]Q \in E_N$, und brich ab, falls während der Berechnung ein nichttrivialer Teiler d von N gefunden wird.

(4) Wähle die nächste Kurve in Schritt (1).

Ein offensichtlicher Vorteil gegenüber dem Pollard-$(p-1)$-Algorithmus ist, daß in Gruppen ganz unterschiedlicher Ordnung in $(p+1-2\sqrt{p}, p+1+2\sqrt{p})$ gerechnet wird, so daß mit großer Wahrscheinlichkeit eine der Gruppenordnungen S-glatt ist. Der größte Faktor, der mit der Elliptische-Kurven-Methode gefunden wurde, hat 49 Stellen und ist ein Teiler von $6^{250} + 1$ (siehe Internetseite [ECM]).

4 Faktorisieren beliebiger Zahlen

Hat man mit den Algorithmen aus Abschnitt 3 alle kleinen Primteiler von N abgespalten, und ist N immer noch zusammengesetzt, so muß nun schweres Geschütz aufgefahren werden. Es wird dabei angenommen, daß N keine kleinen Teiler mehr hat, also insbesondere ungerade ist. Ist N eine nichttriviale Potenz, etwa $N = n^k$ mit $k > 1$, so läßt sich der nichttriviale Teiler $n \in \mathbf{N}$ von N durch näherungsweises Wurzelziehen bestimmen. Hat N keine Primteiler $\leq N^{1/K}$, so muß man für k nur die Zahlen $2, \ldots, K-1$ überprüfen. Also ist N keine Potenz.

4.1 Fermat

Ist $N = a \cdot b$ eine zusammengesetzte ungerade Zahl, so sind auch a und b ungerade und $x := \frac{a+b}{2}$ und $y := \frac{a-b}{2}$ sind ganze Zahlen. Also ist

$$N = (x+y) \cdot (x-y) = x^2 - y^2$$

die Differenz zweier Quadrate. Da $y^2 \geq 0$ ist, ist also $x \geq \lceil \sqrt{N} \rceil$. Daraus ergibt sich der folgende Faktorisierungsalgorithmus:

(1) Setze $x := \lceil \sqrt{N} \rceil$ und $z := x^2 - N$.

(2) Ist $z = y^2$ ein Quadrat, so gib die Faktorisierung $N = (x-y)\cdot(x+y)$ aus. Sonst setze $x := x+1$ und $z := z+2x+1$ und wiederhole (2).

Aus diesem Fermatschen Faktorisierungsalgorithmus entstand die grundlegende Idee für die meisten modernen Faktorisierungsmethoden. Gauß und Legendre beobachteten nämlich, daß es genügt, ein Vielfaches von N als Differenz von zwei Quadraten zu schreiben

$$x^2 - y^2 = (x+y)\cdot(x-y) = kN$$

für ein $k \in \mathbf{N}$. In dem Ring $\mathbf{Z}/N\mathbf{Z}$ der Restklassen modulo N sind dann y und $\pm x$ Wurzeln von x^2. Hat N genau d verschiedene Primteiler, so gibt es in $\mathbf{Z}/N\mathbf{Z}$ genau 2^d verschiedene Wurzeln von x^2. Also ist die Chance, daß $y \not\equiv \pm x \pmod N$ und damit der größte gemeinsame Teiler $\mathrm{ggT}(x-y, N)$ ein nichttrivialer Teiler von N ist, gleich $(2^d-2)/2^d$, insbesondere $\geq 1/2$ für $d \geq 2$, vorausgesetzt, x ist "unabhängig" von y.

Die Algorithmen, die in den nächsten 3 Abschnitten beschrieben werden, streben eine solche Lösung von $x^2 \equiv y^2 \pmod N$ mit $y \not\equiv \pm x \pmod N$ an. Das Grundprinzip ist bei allen Algorithmen gleich: Man bestimmt genügend viele quadratische Reste modulo N, also ganze Zahlen z_i, so daß $z_i \equiv x_i^2 \pmod N$ für ein $x_i \in \mathbf{N}$ $(i = 1, \ldots, m)$ ist, in der Hoffnung, daß das Produkt einiger der Zahlen z_i ein Quadrat einer natürlichen Zahl ist,

$$\prod_{i\in I} z_i = y^2$$

für eine Teilmenge $I \subseteq \{1, \ldots, m\}$ und ein $y \in \mathbf{N}$. Setzt man $x := \prod_{i\in I} x_i$, so ist $x^2 - y^2$ durch N teilbar. Die Schwierigkeit ist nur, geeignete Zahlen z_i zu finden, so daß ein Teilprodukt über sie ein Quadrat ist. Dazu sucht man glatte Zahlen, also Zahlen z_i, deren sämtliche Primteiler kleiner als eine fest vorgegebene Schranke S sind, deren optimale Wahl von der Größe von N und von dem verwendeten Algorithmus abhängt. Sind $\{p_1, \ldots, p_r\}$ die Primzahlen $\leq S$, so kann man die S-glatten Zahlen als

$$z_i = (-1)^{e_{i0}} \prod_{j=1}^{r} p_j^{e_{ij}}$$

schreiben. Man erhält so eine Matrix $E = (e_{ij} \pmod 2) \in \mathbf{F}_2^{m\times(r+1)}$. Jede Linearkombination des Nullvektors aus den Zeilen von E liefert eine Menge I mit $\sum_{i\in I} e_{ij} \equiv 0 \pmod 2$ für alle $j = 0, \ldots, r$, also ein Quadrat $y^2 = \prod_{i\in I} z_i$.

Eine Beschleunigung des Verfahrens kann man mit der Strategie der "Large-Prime Variations" erzielen. Dazu betrachtet man nicht nur die S-glatten Zahlen z_i, sondern auch *S-semiglatte Zahlen*, also solche von der Form $z_i = q_i d_i$ für eine S-glatte Zahl d_i und $S < q_i \leq S^2$, so daß q_i keine Primteiler $\leq S$ hat; denn dann ist q_i notwendigerweise eine Primzahl. Die Matrix E erhält dabei

keine zusätzlichen Spalten, sondern man merkt sich die jeweiligen Werte q_i. Ist $\prod_{i \in I} z_i$ ein Quadrat, so kommt jedes q_i mit einer geraden Vielfachheit vor. Man merkt sich also die erste zu q_i gehörende Zeile, addiert sie zu allen anderen Zeilen mit demselben $q_j = q_i$, multipliziert die zugehörigen z_j mit dem entsprechenden z_i und streicht dann die zu z_i gehörende Zeile aus E.

4.2 Der Kettenbruchalgorithmus

Gehen wir noch einmal zurück zum Fermatschen Faktorisierungsalgorithmus: Ist in Schritt (2) $z = x^2 - N$ kein Quadrat, so liefert dies eine Einschränkung für die möglichen Primteiler von N: Dann ist nämlich $z \equiv x^2 \pmod{N}$ also ist auch $z \equiv x^2 \pmod{p}$ für jeden Primteiler p von N, also z ein Quadrat modulo p. Mit Hilfe des quadratischen Reziprozitätsgesetzes erhält man Kongruenzen (modulo $4z$) für p und schließt so etwa die Hälfte der Primzahlen als Teiler von N aus. Um solche a priori Eigenschaften der Primteiler von N zu erhalten, suchten schon Legendre und Gauß nach kleinen quadratischen Resten modulo N. Dazu benutzten sie die Kettenbruchentwicklung von $\sqrt{kN}$ für kleine natürliche Zahlen k. Ist nämlich $z = x^2 - kN$ klein, so ist x eine gute Approximation von $\sqrt{kN}$. Bei der Kettenbruchentwicklung schreibt man

$$\sqrt{N} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{\ddots}}}$$

mit $c_i \in \mathbf{Z}_{\geq 0}$. Dazu setzt man $u_0 := \sqrt{N}$, $c_i := \lfloor u_i \rfloor$, $u_{i+1} = \frac{1}{u_i - c_i}$ $(i = 0, 1, \ldots)$. Man kann nachrechnen, daß $u_i = c_i + \frac{\sqrt{N} - P_{i+1}}{Q_i}$, wobei $(-1)^i Q_i = A_i^2 - B_i^2 N$ ein Quadrat modulo N ist und $2\sqrt{N} > Q_i > 0$ gilt ([16, Appendix 8]). Insbesondere sind die $(-1)^i Q_i$ kleine quadratische Reste modulo N.

Kleine Zahlen sind mit größerer Wahrscheinlichkeit glatt als große. Daher nutzten Morrison und Brillhart [10] die Kettenbruchentwicklung zur Bestimmung genügend vieler glatter quadratischer Reste $z_i = (-1)^i Q_i$ modulo N:

Kettenbruchalgorithmus.

(0) Bestimme eine geeignete Schranke S und eine kleine natürliche Zahl k. Setze $j := 1$.

(1) Bestimme die Kettenbruchentwicklung von $\sqrt{kN}$.

(2) Teste jedesmal, ob Q_i eine S-glatte Zahl ist durch Trial Division durch die Primzahlen $p_1, \ldots, p_r \leq S$.

(3) Falls Q_i aus (2) S-glatt ist, so schreibe die Exponenten von $z_j := (-1)^i Q_i = A_i^2 - B_i^2 kN$ modulo 2 als eine neue Zeile der Matrix E und setze $x_j := A_i$ und $j := j + 1$.

(4) Teste, ob E einen Kern hat: Ist $\sum_{j\in I} e_{jl} \equiv 0 \pmod 2$ für alle $l = 0, \ldots, r$, so setze $y = \sqrt{\prod_{j\in I} z_j}$ und $x := \prod_{j\in I} x_j$ und teste, ob $\mathrm{ggT}(x - y, N)$ ein nichttrivialer Teiler von N ist.

(5) Ist die Periode der Kettenbruchentwicklung von $\sqrt{kN}$ erreicht, so wähle ein neues k in Schritt (0).

4.3 Das Quadratische Sieb

Ein Schwachpunkt beim Kettenbruchalgorithmus ist, daß ein Glattheitstest teuer ist und man in Schritt (2) viele quadratische Reste vergeblich auf Glattheit testet. 1982 hat C. Pomerance [13] einen Algorithmus veröffentlicht, der die Erzeugung der quadratischen Reste und damit auch den Glattheitstest vereinfacht. Das *Quadratische Sieb* erzeugt quadratische Reste modulo N als Werte eines Polynoms. Sei also $f(X) := X^2 - N \in \mathbf{Z}[X]$ und setze $z_i := f(x_i) = x_i^2 - N$ für $x_i \in [\sqrt{N} - M, \sqrt{N} + M] \cap \mathbf{N}$. Um die S-glatten (oder S-semiglatten) quadratischen Reste unter den z_i zu finden, geht man wie beim Sieb des Eratosthenes vor. Ist nämlich p ein Teiler von $f(x_i)$, so teilt p auch alle $f(x_i + k \cdot p)$ mit $k \in \mathbf{Z}$. Um Divisionen zu vermeiden, speichert man die Werte $R_i := \log|f(x_i)|$ ab. Für alle Primzahlen $p \leq S$ berechnet man ein $x(p)$ mit $p \mid f(x(p))$ und subtrahiert $\log(p)$ von allen R_i, für die $x_i \equiv \pm x(p) \pmod p$ ist. Ist nach dem Durchlaufen aller Primzahlen

$$R_i = \log(|f(x_i)|) - \sum_{\substack{p\leq S,\ p \text{ prim} \\ x_i \equiv \pm x(p) \pmod p}} \log(p)$$

kleiner als eine vorgegebenen Schranke, so ist z_i ein guter Kandidat für eine S-glatte Zahl. Man testet nur solche z_i auf Glattheit und bestimmt gleichzeitig die Exponentenvektoren und damit die Matrix E analog zu Schritt (2)-(4) des Kettenbruchalgorithmus.

Da Sieben sehr viel schneller ist als Trial Division, ist das Quadratische Sieb dem Kettenbruchalgorithmus überlegen. Ein Problem beim Quadratischen Sieb ist, daß die Werte $f(x_i)$ schnell groß werden und deshalb schlechte Chancen haben, glatt zu sein. Deshalb werden i.a. gleich mehrere Polynome betrachtet [20], damit M und folglich die Zahlen z_i klein gehalten werden können. Das so entstehende Verfahren, das sogenannte *Multiple Polynomial Quadratic Sieve*, ist im Moment einer der besten allgemeinen Faktorisierungsalgorithmen. Mit ihm können z.B. 67- bis 88-stellige Zahlen in 0,4 bis 12 CPU-Stunden faktorisiert werden [1].

4.4 Das Zahlkörpersieb

Rekorde im Faktorisieren hat man mit dem *Zahlkörpersieb* [8] erzielt. Dabei wird in Teilringen algebraischer Zahlkörper gerechnet, die $\mathbf{Z}/N\mathbf{Z}$ als ein

homomorphes Bild haben. Man bestimmt zwei irreduzible, der Einfachheit halber hier als normiert vorausgesetzte, Polynome $f(X), g(X) \in \mathbf{Z}[X]$ vom Grad n bzw. l und ein $m \in \mathbf{Z}$ mit $f(m) \equiv 0 \pmod{N}$ und $g(m) \equiv 0 \pmod{N}$. Weiter seien $\alpha, \beta \in \mathbf{C}$ Nullstellen von f bzw. g, $f(\alpha) = g(\beta) = 0$. Dann induziert die Abbildung $\alpha \mapsto m$ (bzw. $\beta \mapsto m$) einen Ringhomomorphismus von $\mathbf{Z}[\alpha] \to \mathbf{Z}/N\mathbf{Z}$ (bzw. $\mathbf{Z}[\beta] \to \mathbf{Z}/N\mathbf{Z}$). Man sucht Zahlen $v_i, w_i \in \mathbf{Z}$ mit $\mathrm{ggT}(v_i, w_i) = 1$, so daß gewisse Teilprodukte

$$(*) \quad \prod_{i \in I}(v_i - w_i\alpha) = \gamma^2 \ (\gamma \in \mathbf{Z}[\alpha]) \ \text{ und } \prod_{i \in I}(v_i - w_i\beta) = \delta^2 \ (\delta \in \mathbf{Z}[\beta])$$

Quadrate sind. Ist nämlich $\gamma = c_0 + c_1\alpha + \ldots + c_{n-1}\alpha^{n-1}$ und $\delta = d_0 + d_1\beta + \ldots + d_{l-1}\beta^{l-1}$, so erfüllen $x := \sum_{i=0}^{n-1} c_i m^i$ und $y := \sum_{j=0}^{l-1} d_j m^j$ die Kongruenz

$$x^2 \equiv \prod_{i \in I}(v_i - w_i m) \equiv y^2 \pmod{N}.$$

Um geeignete Quadrate wie in $(*)$ zu finden, sucht man $v_i, w_i \in \mathbf{Z}$ für die $(v_i - w_i\alpha)$ und $(v_i - w_i\beta)$ "glatte" Zahlen sind. Dabei will man möglichst das Rechnen im Zahlkörper $\mathbf{Q}[\alpha]$ (bzw. $\mathbf{Q}[\beta]$) vermeiden. Eine notwendige Bedingung für $(*)$ ist, daß die Normen $\prod_{i \in I} \mathrm{Norm}(v_i - w_i\alpha) = \mathrm{Norm}(\gamma)^2$ und $\prod_{i \in I} \mathrm{Norm}(v_i - w_i\beta) = \mathrm{Norm}(\delta)^2$ Quadrate in $\mathbf{Z}$ sind.

Nun ist $\mathrm{Norm}(v - w\alpha) = w^n f(v/w)$ ein ganzzahliges Polynom in v und w. Hält man z.B. w fest, so kann man Zahlen v, für welche $\mathrm{Norm}(v - w\alpha) \cdot \mathrm{Norm}(v - w\beta)$ glatt ist, durch Sieben der Polynome $w^n f(X/w)$ und $w^l g(X/w) \in \mathbf{Z}[X]$ analog wie im Quadratischen Sieb bestimmen.

Das Aufstellen der Matrix E aus den Exponentenvektoren der glatten $v_i - w_i\alpha$ (und $v_i - w_i\beta$) muß verfeinert werden. Die Spalten von E werden mit Primidealen von $\mathbf{Z}[\alpha]$ indiziert. Teilt eine rationale Primzahl p die Norm $\mathrm{Norm}(v - w\alpha)$, so gibt das $r \in \{0, \ldots, p-1\}$ mit $v \equiv wr \pmod{p}$ das Primideal $\wp_r$ in $\mathbf{Z}[\alpha]$ über p an, welches das Ideal $(v - w\alpha)$ teilt. Anstelle von nur einer Spalte für jedes p hat die Matrix E also für jedes vorkommende (p, r) eine Spalte. Jedes Element aus dem Kern von E entspricht nun einem Produkt $\prod_{i \in I}(v_i - w_i\alpha)$, das Quadrat eines Ideals in $\mathbf{Z}[\alpha]$ ist. Auch beim Kettenbruchalgorithmus hatten wir noch eine zusätzliche Spalte für das Vorzeichen eingeführt. In $\mathbf{Z}[\alpha]$ muß man sehr viel mehr Einheiten als nur ± 1 berücksichtigen. Außerdem ist im allgemeinen nicht jedes Ideal in $\mathbf{Z}[\alpha]$ von einem Element erzeugt. Beide Probleme kann man gleichzeitig, ohne viel Rechnen in $\mathbf{Z}[\alpha]$, mit Hilfe von sogenannten quadratischen Charakteren erschlagen: Dazu wählt man z.B. Primideale $\mathcal{Q}$ in $\mathbf{Z}[\alpha]$, deren Norm eine Primzahl größer als S ist. Damit ist sichergestellt, daß die S-glatten $v - w\alpha$ auf Einheiten modulo $\mathcal{Q}$ abgebildet werden. Für jedes dieser Primideale $\mathcal{Q}$ erhält die Matrix E eine weitere Spalte, in die man einträgt, ob $(v_i - w_i\alpha)$ auf ein Quadrat in $\mathbf{Z}[\alpha]/\mathcal{Q}$ abgebildet wird (Eintrag 0) oder auf ein Nichtquadrat (Eintrag 1). Wählt man genügend viele solche Ideale $\mathcal{Q}$, so entspricht jedes Element aus dem Kern von E mit

großer Wahrscheinlichkeit einem Quadrat in $\mathbf{Z}[\alpha]$. Analog verfährt man mit den $(v - w\beta)$, was ungefähr ebensoviele Spalten in E liefert. Um ein Element aus dem Kern von E zu finden, braucht man jetzt sehr viel mehr Paare (v_i, w_i), für die $\mathrm{Norm}(v_i - w_i\alpha)\mathrm{Norm}(v_i - w_i\beta)$ glatt ist. Die Wahrscheinlichkeit, solche glatten Zahlen zu finden, ist jedoch so viel größer, daß der Nachteil der zusätzlichen Spalten in E und auch der zusätzliche Rechenaufwand in Zahlkörpern für große Zahlen N aufgewogen werden. Wie auch schon beim Quadratischen Sieb ist der zeitaufwendigste Teil des Algorithmus, das Sieben, sehr leicht parallelisierbar und kann problemlos auf mehrere Rechner verteilt werden.

Der aktuelle Rekord im Faktorisieren nichtspezieller Zahlen wurde am 22. August 1999 mit dem Zahlkörpersieb aufgestellt. H. te Riele und andere fanden in Gemeinschaftsarbeit die Faktorisierung einer 155-stelligen Zahl aus der RSA-challenge-list (Internetseite [RSA]) als Produkt von zwei 78-stelligen Primzahlen. Die Auswahl des Polynoms f (vom Grad 5) benötigte 100 MIPS Jahre, also weniger als ein halbes Jahr CPU-Zeit auf modernen Prozessoren. Es soll kleine Werte im Siebbereich annehmen und ungewöhnlich viele Nullstellen modulo kleiner Primzahlen haben. g wurde als $X - m$ gewählt. Die Gesamtzeit beim Sieben betrug 8000 MIPS Jahre (35,7 CPU Jahre) jedoch nur 3 1/2 Monate Kalenderzeit, da auf sehr vielen Rechnern gleichzeitig gesiebt wurde. Das Aufstellen der Matrix E, die Berechnung von Elementen aus dem Kern und das Wurzelziehen gingen im Vergleich zum Sieben sehr schnell.

Dies ist aber noch nicht die größte Zahl, die mit dem Zahlkörpersieb faktorisiert wurde. Am 8. April 1999 wurden der 93-stellige und der 118-stellige Primteiler der Zahl $N = 10^{211} - 1$ gefunden. Dazu wurde die spezielle Form von N zur geschickten Wahl der Polynome $f(X) = 10X^6 - 1$ und $g(X) = X - 10^{35}$ $(m = 10^{35})$ benutzt.

5 Wie schnell kann Faktorisieren sein?

Die meisten Methoden zum Faktorisieren einer Zahl N benutzen glatte Zahlen. Dabei ist die Häufigkeit glatter Zahlen nicht nur für eine Komplexitätsanalyse des Algorithmus von Bedeutung, sondern auch für die optimale Auswahl der Schranken. Sei $\psi(M, S)$ die Anzahl S-glatter Zahlen $\leq M$. Um T verschiedene S-glatte Zahlen zu finden, muß man also in etwa $MT/\psi(M, S)$ zufällige Zahlen $\leq M$ erzeugen. Definiert man

$$L_M[v, c] := \exp(c \cdot \ln(M)^v \ln(\ln(M))^{1-v}),$$

so ist nach [2, Theorem 10.1]

$$\frac{MT}{\psi(M, S)} \geq L_M[1/2, \sqrt{2} + o(1)]$$

für $M \to \infty$, falls $T = S^{1+o(1)}$, wobei Gleichheit nur für $S = L_M[1/2, \sqrt{2}/2 + o(1)]$ gilt. Der Kettenbruchalgorithmus und auch das Quadratische Sieb erzeu-

gen quadratische Reste modulo N in der Größenordnung $M \sim \sqrt{N}$. Da man in etwa so viele S-glatte Zahlen benötigt, wie es Primzahlen $\leq S$ gibt (also $S/\ln(S) \sim S^{1+o(1)}$), ist die optimale Schranke also $S = L_{\sqrt{N}}[1/2, \sqrt{2}/2 + o(1)]$. Da die übrigen Operationen vernachlässigt werden können, ist unter der Annahme, daß die erzeugten quadratischen Reste sich bezüglich der Glattheit wie zufällige Zahlen benehmen, die erwartete Laufzeit für den Kettenbruchalgorithmus und das Quadratische Sieb $L_N[1/2, 1 + o(1)]$. Auch die Elliptische-Kurven-Methode findet, unter einer plausiblen Annahme, den kleinsten Primteiler p von N mit $L_p[1/2, 2 + o(1)]$ arithmetischen Operationen [7]. Wegen der Verteilung glatter Zahlen hat man bis zur Entdeckung des Zahlkörpersiebs angenommen, daß der Exponent $\frac{1}{2}$ nicht unterboten werden könne. Unter gewissen Annahmen läßt sich jedoch zeigen, daß die heuristische Laufzeit des Zahlkörpersiebs $L_N[1/3, (64/9)^{1/3} + o(1)]$ ist [2].

Bis heute kennt man noch keinen deterministischen Faktorisierungsalgorithmus mit subexponentieller Laufzeit. Setzt man die verallgemeinerte Riemannsche Vermutung voraus, so ist Shanks Klassengruppenalgorithmus [18] ein deterministisches Faktorisierungsverfahren mit Komplexität $O(N^{1/5+o(1)})$ ($O(N^{1/4+o(1)})$ ohne diese Voraussetzung). Dabei wird in der Idealklassengruppe von $\mathbf{Q}[\sqrt{-N}]$ mit Hilfe der Bijektion zwischen eigentlichen Idealklassen und reduzierten binären definiten quadratischen Formen gerechnet. Elemente der Ordnung 2 in dieser Gruppe liefern Zerlegungen von N, vgl. [14, Section 5.6.4].

Eine sehr interessante Plausibilitätsbetrachtung, die durch den Primzahlsatz motiviert ist und die Existenz einer Faktorisierungsmethode, deren Laufzeit polynomial in der Länge $\log(N)$ der eingegebenen Zahl ist, nahezulegen versucht, findet man in [16, S. 221 ff]. Die Entwicklung eines polynomialen Faktorisierungsalgorithmus ist zum einen praktisch interessant, da man damit das weitverbreitete Kryptographieverfahren RSA [17] angreifen kann, zum anderen ist es komplexitätstheoretisch von fundamentaler Bedeutung, wenn man zeigen kann, daß es keinen solchen polynomialen probabilistischen Algorithmus gibt, der mit herkömmlichen Computern auskommt.

P. Shor hat nämlich einen solchen Algorithmus für Quantencomputer entworfen [19]. Momentan ist der Quantencomputer zwar nicht mehr als ein theoretisches Modell, jedoch widerspricht es keinem physikalischen Gesetz, daß solche Computer in hinreichender Größe gebaut werden könnten. Anstelle der gewöhnlichen bits 0/1 gibt es im Quantencomputer sogenannte qubits $|0\rangle$ und $|1\rangle$, die einen 2-dimensionalen komplexen Vektorraum erzeugen. Die Zustände sind nicht mehr 0-1-Folgen der Länge n, sondern Vektoren $\sum_{s=0}^{2^n-1} a_s V_s$ in $\mathbf{C}^{2^n}$, wobei V_s das der Binärdarstellung von s entsprechende Tensorprodukt der qubits $|0\rangle$ und $|1\rangle$ ist. Die Zahlen $a_s \in \mathbf{C}$ erfüllen $\sum_{s=0}^{2^n-1} |a_s|^2 = 1$, und $|a_s|^2$ gibt die Wahrscheinlichkeit an, den Zustand V_s zu beobachten. Die elementaren Operationen sind Multiplikationen mit *Elementarmatrizen*; das sind gewisse unitäre $2^n \times 2^n$-Matrizen, die bis auf einen 4×4 Block die Identität sind. Ein Bestandteil des

Shorschen Faktorisierungsalgorithmus ist die *Quantum Fourier Transformation*

$$\mathrm{QFT}_n : \mathbf{C}^{2^n} \to \mathbf{C}^{2^n}; V_s \mapsto \frac{1}{2^{n/2}} \sum_{r=0}^{2^n-1} \exp(\frac{2\pi i s r}{2^n}) V_r.$$

QFT_n kann als Produkt von $\binom{n}{2}$ Elementarmatrizen geschrieben werden. Zum Faktorisieren von N benutzt man wieder die alte Fermatsche Idee. Man wählt ein zufälliges $x \in \{2, \ldots, N-2\}$. Der Algorithmus bestimmt die Ordnung g von x in $(\mathbf{Z}/N\mathbf{Z})^*$. Ist g gerade und $x^{g/2} \not\equiv -1 \pmod N$, so ist $\mathrm{ggT}(x^{g/2} - 1, N)$ ein nichttrivialer Teiler von N. Dazu wird in einem 2^{3L}-dimensionalen Vektorraum, d.h. mit qubits der Länge $3L$ gearbeitet, wobei $L := \lceil \log_2(N) \rceil$. Der Anfangszustand ist

$$\frac{1}{2^L} \sum_{s=0}^{2^{2L}-1} V_s \otimes V_0$$

wobei der zweite Tensorfaktor L qubits lang ist. Mit $O(L^3)$ elementaren Operationen berechnet man daraus

$$\frac{1}{2^L} \sum_{s=0}^{2^{2L}-1} V_s \otimes V_{x^s \pmod N}$$

und führt dann QFT_{2L} auf den ersten $2L$ qubits aus, was

$$\frac{1}{2^{2L}} \sum_{s=0}^{2^{2L}-1} \sum_{r=0}^{2^{2L}-1} \exp(\frac{2\pi i r s}{2^{2L}}) V_r \otimes V_{x^s \pmod N}$$

liefert. Dieser Zustand wird gemessen. Dabei ist die Wahrscheinlichkeit, $V_r \otimes V_{x^j}$ zu beobachten, gleich

$$\frac{1}{2^{4L}} | \sum_{s \equiv j \pmod g} \exp(\frac{2\pi i r s}{2^{2L}}) |^2.$$

Falls die Einheitswurzeln in der Summe in verschiedene Richtungen zeigen, ist diese Zahl sehr klein. Also beobachtet man mit großer Wahrscheinlichkeit solche r mit $gr \sim d2^{2L}$ für ein $d \in \mathbf{N}$. Daraus läßt sich die Ordnung g von $x \in (\mathbf{Z}/N\mathbf{Z})^*$ durch Runden von $\frac{r}{2^{2L}}$ ermitteln. Der Algorithmus benötigt asymptotisch $O(L^3)$ elementare Operationen und ist damit polynomial in der Anzahl der Ziffern von N.

Literatur

[1] *H. Boender, H. J. J. te Riele:* Factoring integers with large-prime variations of the quadratic sieve. Experiment. Math. **5** (1996), 257-273.

[2] *J. P. Buhler, H. W. Lenstra, Jr, C. Pomerance:* Factoring integers with the number field sieve. [8], 50-94.

[3] *R. P. Brent:* Parallel algorithms for integer factorisation. in J. H. Loxton (Hrsg.): Number theory and cryptography, Cambridge University Press (1990), 26-37.

[4] *H. Cohen:* A course in computational algebraic number theory. Springer Graduate Text in Mathematics **138** (third printing 1996)

[5] *R.-M. Elkenbracht-Huizing:* Historical background of the number field sieve factoring method. Nieuw archief voor wiskunde, Vierde serie Deel 14 no 3 (1996), 375-389

[6] *C. F. Gauß:* Disquisitiones arithmeticae. Yale university Press, New Haven, 1966

[7] *H. W. Lenstra, Jr:* Factoring integers with elliptic curves. Ann. of Math. **126** (1987), 649-673.

[8] *A. K. Lenstra, H. W. Lenstra, Jr. (Hrsg.):* The development of the number field sieve. Springer Lecture Notes in Math. **1554** (1993)

[9] *P. L. Montgomery:* A survey of modern integer factorization algorithms. CWI Quaterly, Volume 7 (4) (1994), 337-365

[10] *M. A. Morrison, J. Brillhart:* A method of factoring and the factorization of F_7. Math. Comp. **29** (1975), 83-205.

[11] *J. Pollard:* Theorems of factorization and primality testing. Proc. Cambridge Philos. Soc. **76** (1974), 521-528.

[12] *J. Pollard:* Monte Carlo method for factorization. BIT 15 (1975), 331-334.

[13] *C. Pomerance:* Analysis and comparison of some integer factoring algorithms. In H. W. Lenstra, Jr. and R. Tijdeman (Hrsg.) Computational Methods in Number Theory, Part I, Mathematisch Centrum Amsterdam (1982), 89-139.

[14] *C. Pomerance, R. Crandall:* Primes. A computational perspective. Springer (2000)

[15] *P. Ribenboim:* The NEW book of prime number records. Springer (1996)

[16] *H. Riesel:* Prime numbers and computer methods of factorization. Second edition, Birkhäuser (1994)

[17] *R. L. Rivest, A. Shamir, L. Adleman:* A method of obtaining digital signatures and public-key cryptosystems. Comm. Assoc. Compu. Mach. **21** (1978), 120-126.

[18] *D. Shanks:* Class number, a theory of factorization and genera. Proc. Symp. Pure Math. **20** A.M.S. Providence, R.I. (1969), 415-440.

[19] *P. W. Shor:* Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Computing **26** (1997), 1484-1509.

[20] *R. D. Silverman:* The multiple polynomial quadratic sieve. Math. Comp. **48** (1987), 329-339.

[21] *H. C. Williams:* A $p+1$ method of factoring. Math. Comp. **39** (1982), 225-234.

Einige interessante Internetseiten

The number theory web http://www.numbertheory.org/ntw/
Auf dieser Seite findet man unter anderem viele homepages von Zahlentheoretikern. Alle weiteren hier aufgeführten Internetseiten sind von dieser Seite aus gelinkt.

The prime pages [PRIME]: http://www.utm.edu/research/primes
Eine sehr ansprechende Seite mit vielen Links. Natürlich findet man dort Informationen über Primzahlrekorde, Primzahltests, Mersenne-Primzahlen, Software und eine lange Literaturliste. Aber man kann auch Primzahlen hören, sehen oder raten. (z.B. unter "Aesthetics of the Prime Numbers Sequence")

Faktorisieren Eine informative Internetseite zum Faktorisieren mit elliptischen Kurven ist
[ECM]: http://www.loria.fr/~zimmerma/records/ecmnet.html
Dort findet man Literatur zum Faktorisieren mit elliptischen Kurven sowie Links zu anderen Seiten, unter anderem den Link NFSNET, wo man auch einige Literatur zum Zahlkörpersieb bekommen kann.
Eine Liste von schwer zu faktorisierenden Zahlen erhält man unter
[RSA]: http://www.rsa.com/rsalabs/html/factoring.html
Über den aktuellen Stand der Faktorisierung von Fermat-Zahlen informiert
[FERMAT]: http://vamri.xray.ufl.edu/proths/fermat.html

Gabriele Nebe
Lehrstuhl B für Mathematik
RWTH Aachen
Templergraben 64
D-52062 Aachen
gabi@math.rwth-aachen.de

Eingegangen 28.10.1999