

Studienarbeit

# Faktorisierungsverfahren

vorgelegt von  
**Kay Schönberger**

12. Februar 2007

Institut für Informatik  
Humboldt-Universität zu Berlin

Lehrstuhl  
Komplexität und Kryptografie

Betreuer : Prof. Dr. Johannes Köbler



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
1.1	Motivation . . . . .	5
1.2	Zur Geschichte . . . . .	5
1.3	Komplexität des Problems . . . . .	7
<b>2</b>	<b>Klassische Verfahren</b>	<b>9</b>
2.1	Probedivision . . . . .	9
2.2	Fermat-Faktorisierung . . . . .	10
2.3	Verbesserung nach Lehman . . . . .	12
<b>3</b>	<b>Die Pollard-Rho-Methode</b>	<b>13</b>
3.1	Die probabilistische Idee . . . . .	13
3.2	Floyd's Zyklalgorithmus . . . . .	14
3.3	Die Produktdarstellung . . . . .	15
3.4	Brent's Verbesserung . . . . .	16
<b>4</b>	<b>Das <math>(p-1)</math>-Verfahren</b>	<b>17</b>
4.1	Eine zweite Phase . . . . .	18
4.2	Laufzeitanalyse . . . . .	18
<b>5</b>	<b>Die Methode der elliptischen Kurven</b>	<b>19</b>
5.1	Grundlegende Definitionen . . . . .	19
5.2	Elliptische Kurven mod $p$ . . . . .	20
5.3	Lenstra's Algorithmus . . . . .	21
5.4	Laufzeitanalyse . . . . .	22
5.5	Verbesserungen des Verfahrens . . . . .	24
<b>6</b>	<b>Das Quadratische Sieb</b>	<b>25</b>
6.1	Siebschritt . . . . .	25
6.2	Auswahlschritt . . . . .	27
6.3	Laufzeitanalyse . . . . .	28
6.4	Verbesserungen des Verfahrens . . . . .	29



# 1 Einleitung

## 1.1 Motivation

Die vorliegende Studienarbeit beschäftigt sich mit dem Problem, eine natürliche Zahl in ihre Primfaktoren zu zerlegen. Bis etwa 1970 war dieses Problem nur theoretischer Natur; mit dem Aufkommen leistungsfähiger Computer stand zunehmend die Entwicklung effizienter Verfahren zur Faktorisierung im Mittelpunkt der Forschung.

Letztlich machte die Etablierung des heute weit verbreiteten Public-Key-Verschlüsselungssystems RSA dieses Problem aktuell. Da RSA von der Schwierigkeit der Faktorisierung Gebrauch macht, steht und fällt dessen Sicherheit mit der Effizienz der vorhandenen Faktorisierungsverfahren. Angesichts der Verbreitung von RSA in sicherheitskritischen Bereichen ein nicht zu unterschätzendes Problem.

Tatsächlich sind zur Zeit noch keine effizienten, d. h. Polynomialzeitalgorithmen zur Primfaktorzerlegung bekannt. Die heute gebräuchlichen Verfahren wurden fast alle in den letzten 35 Jahren entwickelt. Trotz enormer Fortschritte erfordern die asymptotisch schnellsten von ihnen immer noch (sub-)exponentiellen Aufwand. Auf der anderen Seite existiert aber noch kein Beweis dafür, daß Faktorisieren wirklich *schwierig* ist.

Eine Besonderheit stellt der Quantenalgorithmus von Shor [Shor94] dar, der auf einem (trotz Forschungserfolgen immer noch eher hypothetischen) Quantencomputer mit Polynomialzeit auskommt.

Allgemein lassen sich Faktorisierungsverfahren in zwei Klassen unterteilen:

Bei Verfahren der ersten Art hängt die erwartete Laufzeit von der Größe des zu findenden Primteilers  $p$  ab. Ein einfaches Beispiel hierfür ist die Probedivision. Bei Verfahren der zweiten Art ist die Größe der Teiler nicht von Belang. Die Laufzeit hängt ausschließlich von der Länge der Eingabezahl ab. Diese werden deswegen auch als „allgemeine“ Verfahren bezeichnet, da die spezielle Struktur der Teiler keine Rolle spielt. Ein Beispiel für ein solches Verfahren ist das quadratische Sieb.

In den nachfolgenden Kapiteln werden mehrere Verfahren und deren mathematische Hintergründe vorgestellt, wobei der Schwerpunkt auf den Verfahren erster Art liegt.

## 1.2 Zur Geschichte

Bereits Euklid (ca. 300 v. Chr.) bewies den Fundamentalsatz der Arithmetik, nachdem jede natürliche Zahl eine (bis auf die Reihenfolge der Faktoren) eindeutige Primfaktorzerlegung besitzt. Seitdem ist auch die Probedivision bekannt, bei der man einfach nacheinander die Primzahlen auf Teilbarkeit testet. Damit war es möglich sehr schnell kleine Faktoren zu finden. Für große Zahlen mit großen Primteilern war das Verfahren aufgrund des Zeitaufwandes aber nicht zu gebrauchen. 1643 beschrieb Pierre de Fermat in einem Brief die Idee, einen Teiler nicht mehr direkt zu konstruieren, sondern die gegebene Zahl als Differenz zweier Quadrate zu schreiben:

Ist nämlich  $N = x^2 - y^2$  für natürliche Zahlen  $x$  und  $y$ , so ist  $N = (x - y)(x + y)$  eine Faktorisierung von  $N$ . Dieses Verfahren funktioniert dann gut, falls zwei Teiler existieren, die etwa gleich groß sind, d. h. in der Nähe von  $\sqrt{N}$  liegen. Ansonsten ist das Verfahren sogar schlechter als die Probedivision.

Lange Zeit waren diese beiden Verfahren die einzigen bekannten Verfahren zur Faktorzerlegung. Erst 1926 stellte Maurice Kraitchik in einer Arbeit zwei neue Ideen vor, die das Fermat-Verfahren modifizierten.

Zum einen wird  $N$  nicht mehr als Differenz von Quadraten dargestellt. Vielmehr werden Kongruenzen  $x^2 \equiv y^2 \pmod{N}$  betrachtet. Zum anderen werden die Kongruenzen nicht direkt, sondern

## 1 Einleitung

durch Kombination geeigneter Reste  $x_i^2 \equiv r_i \pmod{N}$  gewonnen.

Ein Verfahren, das solche Kongruenzen erzeugt, ist die (auf Ideen von Lehmer und Powers basierende) von Morrison und Brillhardt in den 60er Jahren entwickelte Kettenbruchmethode. Diese benutzt die Kettenbruchentwicklung von  $\sqrt{N}$ , bei der viele quadratische Reste erzeugt werden. Das Besondere an dieser Methode ist, daß sie ein „allgemeines“ Verfahren darstellt und auf jede natürliche Zahl angewendet werden kann. Morrison und Brillhardt konnten damit im Jahre 1970 die Zerlegung der siebten Fermatzahl

$$F_7 = 2^{2^7} + 1 = 59649589127497217 \cdot 5704689200685129054721$$

angeben.

Etwa zeitgleich entstanden auch noch andere Ideen, wie man das Problem angehen könnte. Mit Hilfe des randomisiert arbeitenden Rho-Verfahrens konnte John M. Pollard im Jahre 1980 einen 16stelligen Faktor von  $F_8$  finden. Ebenfalls von Pollard stammt das  $(p-1)$ -Verfahren, das sehr schnell Primteiler finden kann, die eine spezielle Struktur aufweisen. Ein ähnliches, aber flexibleres Verfahren wurde 1987 vorgestellt. Dieses benutzt die Theorie elliptischer Kurven, die in einem ganz anderen Zusammenhang entwickelt wurde. Mit deren Hilfe konnten bis heute beachtliche Erfolge erzielt werden.

1981 griff man die Ideen von Kraitchik wieder auf. Da die Bestimmung geeigneter quadratischer Reste sehr zeitaufwendig war (jeder Rest wurde durch Testdivisionen gewonnen, obwohl nur ein Bruchteil der Reste verwendbar war), entwickelte Carl Pomerance ein Siebverfahren, das eine erhebliche Beschleunigung ergab. Dieses quadratische Sieb getaufte Verfahren besaß zwei Vorteile: da die auftretenden Berechnungen aus ganz speziellen Vektoroperationen bestanden, waren Implementierungen auf Spezialrechnern um einiges schneller als auf herkömmlichen Maschinen. Darüberhinaus ließ es sich auch noch gut parallelisieren. 1983 konnte ein Cray-Supercomputer alle Zahlen auf der von Samuel Wagstaff veröffentlichten „wanted list“ [Wag] - einer Liste der zehn begehrtesten Faktorisierungen - innerhalb eines Jahres knacken. Eine Weiterentwicklung, das Multipolynomielle Quadratische Sieb (MPQS) war 1994 in der Lage, die 129stellige Zahl RSA-129 zu faktorisieren.

1983		2006	
Zahl	Stellen im noch nicht faktorierten Teil	Zahl	Stellen im noch nicht faktorierten Teil
$2^{211} - 1$	60	$2^{787} - 1$	204
$2^{251} - 1$	69	$2^{779} + 1$	212
$2^{212} + 1$	54	$2^{772} + 1$	215
$10^{64} + 1$	55	$5^{311} - 1$	173
$10^{67} - 1$	61	$5^{311} + 1$	159
$10^{71} - 1$	71	$10^{229} + 1$	164
$3^{124} + 1$	58	$10^{239} - 1$	228
$3^{128} + 1$	53	$3^{479} + 1$	197
$11^{64} + 1$	67	$6^{281} - 1$	162
$5^{79} - 1$	55	$2^{787} + 1$	169

Die „wanted list“ von Samuel Wagstaff von 1983 bzw. 2006.

Gegen Ende der 80er Jahre waren mehrere Verfahren bekannt, die eine vergleichbare Komplexität aufwiesen. Es kam daher der Verdacht auf, daß man sich nahe an der theoretischen Laufzeitgrenze bewegen würde und es keine wesentlich schnelleren Algorithmen geben könne. 1988 aber zeigte John Pollard, daß man durch eine Modifikation des quadratischen Siebes die mittlere Größe der verwendeten Kongruenzen spürbar verringern konnte. Aus der Idee wurde das Zahlkörpersieb, mit dessen Hilfe man 1990 die neunte Fermatzahl  $F_9$  in Primfaktoren mit 7, 49 und 99 Stellen zerlegen konnte. Basis der Überlegungen war die Untersuchung spezieller Zahlen der Gestalt  $b^n \pm 1$ , die zum speziellen Zahlkörpersieb (SNFS) führten. Es zeigte sich aber, daß man das Verfahren auf jede natürliche Zahl anwendbar machen konnte, was zum allgemeinen Verfahren (GNFS) führte. Der bisherige Rekord mit dem GNFS gelang 2005 mit der Zerlegung von RSA-200 (die 200 Dezimalstellen besitzt).

Wagstaff's Listen machen deutlich, welchen Fortschritt man durch leistungsfähigere Rechner und neuere Verfahren in den letzten Jahrzehnten erzielt hat. Auch RSA war bisher immer ein guter Indikator für die Fortschritte im Bereich der Faktorisierung. Im Rahmen der RSA Challenge [RSA] veröffentlicht die Firma RSA Data Security zusammengesetzte Zahlen, die das Produkt zweier unbekannter, etwa gleichgroßer Primfaktoren (die vorher zufällig ausgewürfelt wurden) sind. Auf die Faktorisierungen dieser Zahlen sind z. T. beträchtliche Geldpreise ausgesetzt.

Mittlerweile gibt es eine Vielzahl von aktiven Projekten zur Faktorisierung, bei denen sogar jeder Interessierte mitwirken kann. Erwähnung finden sollte das (bereits etwa 100 Jahre alte) Cunningham-Projekt [Cun] zur Zerlegung von Zahlen der Form  $b^n \pm 1$  mit Basen  $b = 2, 3, 5, 6, 7, 10, 11$  und  $12$ . [GIMPS] beschäftigt sich mit den Mersennezahlen (Zahlen der Form  $2^p - 1$ , wobei  $p$  prim) und Fermatzahlen werden in [FermS] faktorisiert. Weitere erwähnenswerte Projekte sind [ECMNet] und [NFSNet]. Informationen zum Thema Faktorisieren allgemein erhält man auf der Internetseite des Mersenneforums [MersF].

### 1.3 Komplexität des Problems

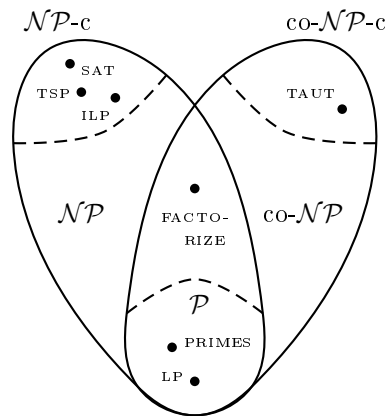
Bei der Aufgabe, eine Liste aller Primfaktoren zu erzeugen, handelt es sich um ein reines Berechnungsproblem. Für eine Analyse des Problems und einen Vergleich mit klassischen Problemen der Komplexitätstheorie betrachtet man das äquivalente Entscheidungsproblem

$$\text{FACTORIZE} := \{(N, k) \mid N \text{ besitzt einen Primteiler } p \leq k\}$$

Man kann sich leicht von der Äquivalenz des Berechnungs- und des Entscheidungsproblems überzeugen: Sind alle Primteiler erzeugt, so ist das Entscheidungsproblem sofort lösbar. Umgekehrt kann man mit Hilfe eines Algorithmus, der das Entscheidungsproblem löst, jeden Faktor  $p$  in Polynomialzeit durch eine binäre Suche über  $k$  finden. Des weiteren gilt

$$\text{FACTORIZE} \in \mathcal{NP} \cap \text{co-}\mathcal{NP}$$

da anhand vorgelegter Faktoren sowohl „ja“- als auch „nein“-Antworten sofort verifiziert werden können. Die Faktoren selbst können in Polynomialzeit auf die Primzahleigenschaft getestet werden.



FACTORIZE in der Komplexitätshierarchie

Da man davon ausgeht, daß  $\mathcal{NP} \neq \text{co-}\mathcal{NP}$  (eine immer noch offene Frage), hält man es für wahrscheinlich, daß FACTORIZE nicht  $\mathcal{NP}$ -vollständig ist wie beispielsweise das Travelling Salesman Problem (TSP). Auf der anderen Seite konnte aber auch nach jahrzehntelanger Forschung noch kein Polynomialzeitalgorithmus angegeben werden. Interessanterweise ist das Problem PRIMES, zu entscheiden, ob eine natürliche Zahl prim ist, in Polynomialzeit lösbar [AKS02]. Diese Tatsache macht Public-Key-Systeme wie RSA möglich, die einerseits schnell erzeugbare Schlüssel benötigen, andererseits aber sicherstellen wollen, daß diese nicht effizient auffindbar sind.

Darüberhinaus liegt das Faktorisierungsproblem in der Klasse  $\mathcal{BQP}$ , da es auf einem Quantencomputer in  $\mathcal{O}(n^3)$  gelöst werden kann [Shor94].

## 1 Einleitung

## 2 Klassische Verfahren

### 2.1 Probedivision

Das älteste und zugleich einfachste Verfahren, Teiler einer zusammengesetzten Zahl zu finden, ist die Probedivision. Hierzu dividiert man die gegebene Zahl  $N$  nacheinander durch 2, 3, 4, usw., bis man einen oder alle Teiler gefunden hat. Die Rechtfertigung hierfür ist, daß 50% der natürlichen Zahlen gerade (d. h. durch 2 teilbar) und bereits 77% aller Zahlen einen der Trivialteiler 2, 3, 5 oder 7 enthalten. Ein solches Vorgehen macht natürlich nur Sinn, wenn  $N$  wirklich zusammengesetzt und nicht etwa eine Primzahl ist. Ist das nicht bekannt, so wendet man vorher einen Primtest - z. B. Fermat- oder Rabintest - an. Die folgenden Überlegungen führen noch zu einigen Einsparungen der nötigen Divisionen.

**Satz 2.1.1** *Ist  $N \in \mathbb{N}$  zusammengesetzt, dann existiert ein Primteiler  $p$  von  $N$  mit  $p \leq \sqrt{N}$ .*

*Beweis.* Nach Voraussetzung ist  $N = ab$  mit  $a, b \neq 1$ . Ist  $a \leq \sqrt{N}$ , dann ist für jeden Primteiler  $p$  von  $a$  die Behauptung erfüllt. Ist  $a > \sqrt{N}$ , so muß aber  $b < \sqrt{N}$  gelten, da sonst  $N = ab > \sqrt{N} \cdot \sqrt{N} = N$ . Dann leistet jeder Primteiler  $p$  von  $b$  das Verlangte.  $\square$

Demnach genügt es, nur alle Primzahlen bis  $\sqrt{N}$  auf Teilbarkeit zu testen. Sind die betrachteten  $N$  nicht zu groß, so kann man dazu eine vorher erstellte Liste aller in Frage kommenden Primzahlen verwenden. Ansonsten bietet sich folgende Überlegung an, um nicht jede Zahl testen zu müssen: Ist 2 bereits als Teiler ausgeschlossen, so sind es auch alle Vielfachen von 2, d. h. es kommen nur noch alle ungeraden Zahlen in Frage. Die gleiche Überlegung gilt für alle Vielfachen von 3. Genauer bedeutet das, nur die Reste  $r$  zu berücksichtigen, für die  $r \equiv 1 \pmod{2}$  bzw.  $r \equiv 1 \pmod{3}$  oder  $r \equiv 2 \pmod{3}$  gilt. Zusammengenommen also die Reste  $r$  mit  $r \equiv 1 \pmod{6}$  und  $r \equiv 5 \pmod{6}$  - was allen Zahlen der Form  $6k \pm 1$  entspricht.

Allgemein gilt: Für die ersten  $l$  Primzahlen  $p_1, \dots, p_l$  sei  $p_l\# := \prod_{i=1}^l p_i$  (die sogenannten Primorials). Dann sind nur diejenigen Reste modulo  $p_l\#$  zu berücksichtigen, die teilerfremd zu  $p_l\#$  sind. Das sind genau  $\varphi(p_l\#)$  viele, mit der nachfolgend definierten Funktion.

**Definition 2.1.2** *Die Funktion*

$$\varphi(N) := \#\{1 \leq x < N \mid \gcd(x, N) = 1\}$$

*heißt EULERSche  $\varphi$ -Funktion.*

Modulo  $p_3\# = 2 \cdot 3 \cdot 5 = 30$  wären also die Reste  $\{1, 7, 11, 13, 17, 19, 23, 29\}$  bzw.  $\{\pm 1, \pm 7, \pm 11, \pm 13\}$  zu überprüfen. Das heißt, man hat nur noch  $\frac{8}{30} \approx 27\%$  aller Zahlen zu testen. Diese Idee kann man natürlich beliebig weiterverfolgen, allerdings steht der Nutzen in keinem Verhältnis zum hinzukommenden Aufwand. Die Berücksichtigung der 7 würde bereits 48 verschiedene Reste nach sich ziehen, wobei man den zu testenden Anteil auf gerade einmal  $\frac{48}{210} \approx 23\%$  drückt.

Man benötigt also etwa  $\frac{2}{4}$  Schritte, um einen Primfaktor  $p$  zu finden. Bei Erfolg sollte der Test für den gefundenen Faktor wiederholt werden, um auch Primpotenzen ausfindig zu machen.

Bevor man andere Verfahren anwendet, wird man zunächst kleine Teiler durch Testen ausschließen, da die Probedivision für solche Teiler sehr schnell ist (bzw. andere Verfahren mit kleinen Teilern Probleme haben). Mit vertretbarem Aufwand ist dies bis etwa  $10^{12}$  möglich. Das Verfahren besitzt somit eine Komplexität von  $\mathcal{O}(p)$  bzw.  $\mathcal{O}(\sqrt{N})$ , wobei sämtliche Primteiler gefunden werden.

Erwähnung finden sollte eine wichtige Variante der Probedivision, die Euklid-Faktorisierung: Man testet nicht mehr direkt eine Zahl auf Teilbarkeit, sondern wählt ein  $1 < m < N$  und berechnet

$\gcd(m, N)$  mit dem Euklidischen Algorithmus. Enthält nämlich  $m$  einen Teiler, der auch in  $N$  vorkommt, so ist auch  $1 < \gcd(m, N) < N$  und wir haben einen Teiler gefunden. Wie man das entsprechende  $m$  findet, hängt vom jeweiligen Verfahren ab. Wie wir später noch sehen werden wird diese Form von Test bei allen gebräuchlichen Faktorisierungsmethoden verwendet, insbesondere da der Euklidische Algorithmus vergleichsweise schnell arbeitet.

## 2.2 Fermat-Faktorisierung

Die 1643 von Fermat in einem Brief (an einen nicht genau bekannten Adressaten) beschriebene Methode hat das Ziel, die gegebene Zahl als Differenz zweier Quadrate zu schreiben. Dann ist

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

eine Faktorzerlegung. Natürlich darf dabei  $N$  weder gleich  $(x + y)$  noch  $(x - y)$  sein, sonst hätten wir nichts gewonnen. Schreibt man die Gleichung um, so erhält man  $x^2 - N = y^2$ . Es ist nun naheliegend, solche  $x$ -Werte zu testen, für die die Differenz  $x^2 - N$  möglichst klein wird, da dann die Chance, daß sie ein Quadrat darstellt, größer ist. Genau das war auch die Idee von Fermat: man definiert eine Folge  $x_k$  mit

$$x_0 := \lceil \sqrt{N} \rceil \quad \text{und} \quad x_k := x_0 + k$$

und testet nacheinander die Differenzen  $z_k := x_k^2 - N$ . Um nicht jedesmal eine vollständige Quadrierung durchzuführen, nutzte Fermat die Beziehung

$$x_{k+1}^2 = (x_k + 1)^2 = x_k^2 + 2x_k + 1$$

aus, es folgt

$$z_{k+1} = z_k + 2x_k + 1$$

Dies läßt sich natürlich recht einfach berechnen.

### Beispiel

Als einfaches Beispiel soll  $N = 94883$  faktorisiert werden. Zunächst ist  $\sqrt{N} = 308.03\dots$ , also  $x_0 = 309$ . Dann ergeben sich für die Differenz  $x_k^2 - N$  folgende Werte:

$k$	$x_k$	$z_k = x_k^2 - N$
0	309	598 = 2 · 13 · 23
1	310	1217 = 1217
2	311	1838 = 2 · 919
3	312	2461 = 23 · 107
4	313	3086 = 2 · 1543
5	314	3713 = 47 · 79
6	315	4342 = 2 · 13 · 167
7	316	4973 = 4973
8	317	5606 = 2 · 2803
9	318	6241 = 79 <sup>2</sup>

Nach zehn Schritten finden wir also  $N = 318^2 - 79^2 = (318 - 79)(318 + 79) = 239 \cdot 397$ .

Allgemein muß man aber nicht aus jedem  $z_k$  die Wurzel ziehen. Eine Quadratzahl kann nur auf die Ziffern 0, 1, 4, 5, 6 und 9 enden. Des weiteren wußte Fermat, daß vor den Resten 1, 4 und 9 wiederum nur gerade, vor der 6 nur ungerade Ziffern stehen dürfen. Dazu sind noch 00 und 25 als letzte Ziffernpaare möglich. In unserem Beispiel kämen demnach nur die Zahlen 2461 und 6241 als Quadrate in Frage. Für eine Programmierung würde es sich anbieten, Reste modulo einer Zweierpotenz zu betrachten, da diese Reste besonders schnell auszurechnen sind. Modulo 8 gibt es nur die quadratischen Reste 0, 1, und 4. Modulo 16 sind es gar nur 0, 1, 4 und 9, d. h. anhand der letzten vier Bit kann man 75% aller Zahlen sofort als Nichtquadrate identifizieren, ohne erst die

Wurzel ziehen zu müssen.

Im Gegensatz zur Probedivision arbeitet dieses Verfahren „von oben nach unten“: Beginnend bei  $\sqrt{N}$  testet man absteigend mögliche Kandidaten auf Teilbarkeit bzw. in der Produktdarstellung  $(x+y)(x-y)$  aufsteigend verschiedene  $y$ -Werte. Es fällt auf, daß dabei nicht alle  $y$ -Werte getestet, sondern etliche übersprungen werden. Falls also Teiler existieren, die ausreichend nahe bei  $\sqrt{N}$  liegen, so werden diese viel schneller gefunden als mit einer einfachen Testdivision.

**Lemma 2.2.1** Sei  $N = p \cdot q$  ungerade. Dann findet das Fermat-Verfahren  $p$  und  $q$ .

*Beweis.* Sei o. B. d. A.  $p \geq q$ , dann sind auch  $p$  und  $q$  ungerade und wir können

$$x := \frac{p+q}{2} \quad y := \frac{p-q}{2}$$

setzen. Dann gilt  $N = x^2 - y^2$  und wegen  $x = \frac{1}{2}(p+q) \geq \sqrt{p \cdot q} = \sqrt{N}$  wird  $x$  in jedem Fall getestet. □

**Satz 2.2.2** Sei  $N = p \cdot q$  ungerade und  $|p-q| \leq c \cdot \sqrt[4]{N}$  für eine positive Konstante  $c$ . Dann findet das Fermat-Verfahren  $p$  und  $q$  in  $\mathcal{O}(c^2)$  Schritten.

*Beweis.* Sei wieder o. B. d. A.  $p \geq q$ . Für die Lösung  $N = (x+y)(x-y)$  haben wir also wieder  $y = \frac{p-q}{2}$ . Nach dem Lemma findet das Fermat-Verfahren im Schritt  $k$  die Lösung, d. h.  $z_k = x_k^2 - N = y^2$ . Es gelten zunächst folgende Vorbemerkungen:

1. Es ist  $\sqrt{N} \leq x_0 \leq \sqrt{N} + 1$  und somit  $N \leq x_0^2 \leq N + 2\sqrt{N} + 1$ .
2. Nach Voraussetzung gilt  $z_k = y^2 \leq \frac{c^2}{4}\sqrt{N}$ .
3. Wir können annehmen, daß  $N$  kein Quadrat ist (sonst wäre bereits  $N = x_0^2$  eine Lösung), d. h. es ist  $z_0 \geq 1$ .
4. Aus der Analysis ist bekannt, daß  $\sqrt{1+x} \leq 1 + \frac{x}{2}$  für  $x \geq 0$  gilt.

Aus der expliziten Darstellung  $z_k = z_0 + 2kx_0 + k^2$  und den Vorbemerkungen 1. bis 4. erhalten wir

$$\begin{aligned} k &= \sqrt{x_0^2 + z_k - z_0} - x_0 \\ &\leq \sqrt{N + 2\sqrt{N} + 1 + \frac{c^2}{4}\sqrt{N} - 1} - \sqrt{N} \\ &= \sqrt{N + (2 + \frac{c^2}{4})\sqrt{N}} - \sqrt{N} \\ &= \sqrt{N} \left( \sqrt{1 + \frac{8+c^2}{4\sqrt{N}}} - 1 \right) \\ &\leq \sqrt{N} \left( 1 + \frac{8+c^2}{8\sqrt{N}} - 1 \right) \\ &= 1 + \frac{c^2}{8} \end{aligned}$$

$k$  ist also durch eine Konstante nach oben beschränkt. □

#### Anmerkung

Die Näherung  $\sqrt{1+x} \approx 1 + \frac{x}{2}$  ist um so genauer, je kleiner  $|x|$  ist. Ist also  $c \ll \sqrt[4]{N}$ , so erhalten wir mit  $k \approx 1 + \frac{c^2}{8}$  eine recht gute Näherung für die erforderliche Anzahl von Schritten. Im Regelfall wird dagegen  $|p-q|$  sogar groß gegenüber  $\sqrt{N}$  sein; dann ist  $k \approx \frac{|p-q|}{2} + \frac{N}{|p-q|} - \sqrt{N}$  die bessere Näherung.

## 2 Klassische Verfahren

Im ungünstigsten Fall ist  $|p - q| = \frac{N}{c}$  für ein kleines  $c$ , d. h. wir haben einen sehr kleinen und einen sehr großen Primteiler. Dann ist

$$\begin{aligned} k &= \sqrt{x_0^2 + z_k - z_0} - x_0 \\ &\geq \sqrt{N + \frac{N^2}{4c^2} - 2\sqrt{N}} - \sqrt{N} - 1 \\ &= \frac{N}{2c} \sqrt{1 + \frac{4c^2}{N} - \frac{8c^2}{N\sqrt{N}}} - \sqrt{N} - 1 \\ &\approx \frac{N}{2c} - \sqrt{N} \end{aligned}$$

Mit  $k \approx \frac{|p-q|}{2} - \sqrt{N}$  wären wir sogar bedeutend schlechter als die Probedivision. Für große  $N$  ist der Algorithmus also ungeeignet, da es dann sehr unwahrscheinlich ist, daß der günstige Fall  $|p - q| \leq c \cdot \sqrt[4]{N}$  mit nicht zu großem  $c$  eintritt. Zu Fermat's Zeiten konnte man (in Kombination mit der Probedivision) aber sehr gut Zahlen bis etwa  $10^6$  vollständig zerlegen.

## 2.3 Verbesserung nach Lehman

Eine Verbesserung der Fermat-Faktorisierung für Zahlen mit genau zwei Primteilern wurde 1974 von R. S. Lehman vorgestellt. Sie beruht auf folgendem Satz (für eine detaillierte Beschreibung des Verfahrens und den Beweis des Satzes siehe [Leh74]).

**Satz 2.3.1 (Lehman)** *Ist  $N = p \cdot q$  eine ungerade natürliche Zahl mit Primteilern  $p$  und  $q$  und ist  $1 \leq r < \sqrt{N}$  wobei  $\sqrt{\frac{N}{r+1}} \leq p \leq \sqrt{N}$ , so gibt es natürliche Zahlen  $x$ ,  $y$  und  $k$  mit den folgenden Eigenschaften:*

1.  $x^2 - y^2 = 4kN$
2.  $x \equiv 1 \pmod{2}$ , falls  $k$  gerade;  $x \equiv k + N \pmod{4}$ , falls  $k$  ungerade
3.  $\sqrt{4kN} \leq x \leq \sqrt{4kN} + \frac{\sqrt{\frac{N}{k}}}{4(r+1)}$

Daraus läßt sich ein Algorithmus zum Faktorisieren einer zusammengesetzten Zahl  $N$  ableiten. Zunächst wählt man  $r$  so, daß der Term  $\frac{\sqrt{\frac{N}{k}}}{4(r+1)}$  möglichst klein und die Schranke  $\sqrt{\frac{N}{r+1}} < p$  mit hoher Wahrscheinlichkeit eingehalten wird, auf der anderen Seite aber  $r$  nicht zu groß gerät (da im folgenden etwa  $r$  Operationen erforderlich sind). Als optimale Wahl erweist sich  $r = \sqrt[3]{N}$ .

Nun macht man eine Probedivision bis  $r$ . Findet sich kein Teiler, so sind die Voraussetzungen des Satzes erfüllt. Für alle  $1 \leq k \leq r$  prüft man für jedes  $x$  aus dem im Satz angegebenen Intervall, ob  $y^2 := x^2 - 4kN$  eine Quadratzahl ist. Ist dies der Fall, so gilt  $x^2 \equiv y^2 \pmod{N}$  und es liefern  $\gcd(x + y, N)$  und  $\gcd(x - y, N)$  die gesuchten Teiler  $p$  und  $q$ .

Für ein festes  $k$  sind offensichtlich  $\mathcal{O}(N^{\frac{1}{6}} k^{-\frac{1}{2}})$  verschiedene  $x$  zu testen. Über alle  $k$  gemittelt sind dies  $\mathcal{O}(N^{\frac{1}{6}} \cdot \frac{1}{\sqrt[3]{N}} \sum_{k=1}^{\sqrt[3]{N}} k^{-\frac{1}{2}}) = \mathcal{O}(N^{\frac{1}{6}} \cdot \frac{1}{\sqrt[3]{N}} (\sqrt[3]{N})^{\frac{1}{2}}) = \mathcal{O}(N^{\frac{1}{6}} \cdot N^{-\frac{1}{6}}) = \mathcal{O}(1)$ , so daß der Durchlauf

der  $x$ -Werte für die Komplexität nicht wesentlich ist. Das Verfahren liefert also bereits in  $\mathcal{O}(\sqrt[3]{N})$  die Faktorzerlegung. Da das deterministisch geschieht, besitzt das Verfahren einen gewissen theoretischen Wert - praktisch ist es aber bedeutungslos.

### Anmerkung

Wir sind bisher davon ausgegangen, daß Berechnungen wie  $\sqrt{x^2 - 4kN}$  in konstanter Zeit ausgeführt werden können - was natürlich nicht der Fall ist. Bei der Betrachtung von Zahlen der Länge  $k$  haben beispielsweise die Elementaroperationen Addition und Multiplikation einen Aufwand von  $\mathcal{O}(k)$  bzw.  $\mathcal{O}(k \cdot \ln k \cdot \ln \ln k)$ , das Wurzelziehen ist mit  $\ln k$  Multiplikationen zu realisieren. Die Laufzeit des Lehman-Verfahrens würde demnach etwa um einen Faktor  $\ln N$  vergrößert. Ab hier soll aber nicht mehr darauf Rücksicht genommen werden, die angegebenen Laufzeiten beziehen sich immer auf die Elementaroperationen.

## 3 Die Pollard-Rho-Methode

Die 1975 von John Pollard [Pol75] erfundene Rho-Methode ist ein Monte-Carlo-Verfahren, das auf einer probabilistischen Überlegung beruht. Ein Erfolg kann deswegen zwar nicht garantiert werden, dennoch wird ein Faktor  $p$  recht zuverlässig nach etwa  $\sqrt{p}$  Schritten gefunden. Gegenüber der Probedivision eine spürbare Verbesserung, da somit Faktoren doppelter Länge bei gleicher Rechenzeit gefunden werden können.

### 3.1 Die probabilistische Idee

Als Basis des Verfahrens dient die folgende statistische Überlegung: Aus einer Urne mit  $p$  verschiedenen Kugeln werden nacheinander Kugeln gezogen und wieder zurückgelegt. Man kann nun danach fragen, wie oft man im Mittel ziehen muß, bis sich eine Ziehung wiederholt hat (d. h. man erhält eine Kugel, die man schon in einer früheren Ziehung hatte).

Eine heuristische Überlegung ist folgende: Im zweiten Versuch hat man offensichtlich eine Chance von  $\frac{1}{p}$ , die erste Ziehung zu wiederholen. Schlägt der Versuch fehl, so hat man im dritten aber bereits  $\frac{2}{p}$ , im vierten  $\frac{3}{p}$  usw. Die Chance für die ersten  $k$  Schritte steigt also etwa proportional zur Summe der ersten  $k$  Einzelwahrscheinlichkeiten (falls  $p$  hinreichend groß ist), d. h. ist quadratisch in  $k$ . Wir müßten also nach ungefähr  $\sqrt{p}$  Ziehungen auf die erste Wiederholung treffen. Genauer gilt

**Satz 3.1.1** Für die Wahrscheinlichkeit  $P_k$ , daß bis zur  $k$ -ten Ziehung keine Wiederholung auftritt, gilt

$$P_k = \prod_{i=0}^{k-1} \left(1 - \frac{i}{p}\right)$$

*Beweis.* Induktion: Für  $k = 1$  erhalten wir erwartungsgemäß  $P_1 = 1$ . Sind bereits  $k$  Ziehungen erfolgt, so verbleiben  $p - k$  „günstige“ Möglichkeiten für die nächste Ziehung, es folgt also

$$P_{k+1} = P_k \left(\frac{p-k}{p}\right) = P_k \left(1 - \frac{k}{p}\right) = \prod_{i=0}^k \left(1 - \frac{i}{p}\right)$$

□

Daraus erhalten wir

$$\log P_k = \log \prod_{i=0}^{k-1} \left(1 - \frac{i}{p}\right) = \sum_{i=0}^{k-1} \log \left(1 - \frac{i}{p}\right)$$

Ist  $k \ll p$ , was man annehmen darf, so kann man die Näherung  $\log(1-x) \approx -x$  verwenden:

$$-\log P_k \approx \sum_{i=0}^{k-1} \frac{i}{p} = \frac{k(k-1)}{2p}$$

$$P_k \approx e^{-\frac{k(k-1)}{2p}}$$

Für den kritischen Wert  $P_k = \frac{1}{2}$  erhalten wir also  $k \approx 1.2\sqrt{p}$ , d. h. unsere heuristische Vorhersage. Eine „Anwendung“ dieser Tatsache ist das Geburtstagsparadoxon: Es genügt bereits, nur  $1.2\sqrt{365} \approx 23$  Gäste zu einer Feier einzuladen, damit es sich lohnt, darauf zu wetten, daß zwei von ihnen am selben Tag Geburtstag haben - was sicherlich der Intuition widerspricht.

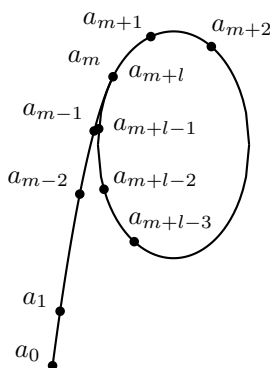
### 3.2 Floyd's Zyklalgorithmus

Für unser Faktorisierungsproblem sei wieder  $N$  eine natürliche Zahl und  $p$  ein zu findender Teiler dieser Zahl. Wir wählen eine Zufallszahlenfolge  $(a_k)$  von Resten modulo  $N$ . Obwohl wir  $p$  nicht kennen, können wir die Folge  $(a_k)$  modulo  $p$  betrachten: nach den angeführten Untersuchungen genügen  $i \approx \sqrt{p}$  Folgenglieder, so daß wir eine Wiederholung  $a_i \equiv a_j \pmod{p}$ ,  $i > j$  vorfinden. Es fällt sofort auf, daß dann  $\gcd(a_i - a_j, N)$  den Faktor  $p$  liefert, es sei denn, es gilt auch  $a_i \equiv a_j \pmod{N}$ , was aber sehr unwahrscheinlich ist.

Das Problem ist nur, daß wir  $i$  und  $j$  nicht kennen. Da wir auch nicht jedes Paar  $(i, j)$  testen können (dies sind etwa  $i^2 \approx p$  viele), greifen wir zu folgendem Trick: Wir wählen keine zufällige, sondern eine *deterministische* Folge  $(a_k)$ , von der wir aber annehmen können, daß sie sich hinreichend zufällig verhält. Der gewünschte Effekt dieser Forderung ist

1. Die betreffende Folge  $(a_k)$  mündet in eine Periode, sobald sich ein  $a_j$  wiederholt. (Determinismus)
2. Für die Periodenlänge  $l$  gilt:  $l \approx \sqrt{p}$ . (Pseudozufälligkeit)

Veranschaulicht man das Verhalten der Folge (nach Durchlauf einer Vorperiode der Länge  $m$  schließt sich eine Periode der Länge  $l$  an) bildlich, so ergibt sich eine Schleife, die vom Aussehen dem griechischen  $\rho$  ähnelt - welches dem Verfahren auch dessen Namen gab.



Verlauf der Pseudozufallsfolge mod  $p$

Für das deterministische Verhalten benötigen wir eine rekursive Vorschrift  $a_{i+1} = f(a_i)$ , die das Nachfolgeelement modulo  $p$  eindeutig bestimmt. Diese soll die Pseudozufälligkeit gewährleisten aber nicht zu aufwendig zu berechnen sein. Man sieht leicht, daß eine lineare Rekursion  $f(x) = x + c$  nicht günstig ist, da dann die Periodenlänge gleich  $p$  ist. Empirische Untersuchungen haben gezeigt, daß die Rekursion  $f(x) = x^2 + c$  beide Forderungen gut erfüllt, sofern  $c$  nicht die Werte 0 oder -2 annimmt (siehe [For96]). Gebräuchlich ist die Vorschrift  $f(x) = x^2 + 1$ , für die die Periodenlänge sogar höchstens  $\frac{1}{2}(p+1)$  sein kann (siehe [AB82]). Die Periode kann man nun schnell mit Hilfe des nächsten Satzes finden.

**Satz 3.2.1 (Zyklensatz von Floyd)** Für eine periodische Folge  $(a_k)$  mit Vorperiode  $m$  und Periode  $l$  ist das kleinste  $i$  mit  $a_i \equiv a_{2i} \pmod{p}$  ein Vielfaches von  $l$ .

*Beweis.* Gilt  $a_i \equiv a_{2i} \pmod{p}$ , so ist  $(a_k)$  spätestens von  $a_{2i}$  an periodisch. Umgekehrt gilt  $a_i \equiv a_j \pmod{p}$  für  $j = i + kl$  mit Periodenlänge  $l$  und  $i > m$ . Dann hat das erste  $i$ , für das  $a_i \equiv a_{2i} \pmod{p}$  zutrifft, die Gestalt  $i = l(1 + \lfloor \frac{m}{l} \rfloor)$ . Ist  $l > m$ , so folgt  $i = l$ , für  $l < m$  ist  $i$  nur ein Vielfaches von  $l$ . □

Daraus ergibt sich *Floyd's Zyklalgorithmus*: Wir berechnen  $\gcd(a_{2i} - a_i, N)$  aufsteigend. Hierzu kann man entweder die Folgen  $a_i$  und  $a_{2i}$  parallel berechnen oder (auf Kosten von Speicherplatz)

die Werte für  $a_i$  im Speicher halten. Weiterhin bietet es sich an, den Euklidischen Algorithmus nicht nach jedem Schritt zu starten. Vielmehr berechnet man

$$\gcd\left(\prod_{i=i_0}^{i_0+n} (a_{2i} - a_i)\right)$$

für  $n$  aufeinanderfolgende Reste. Als günstig erweist sich z. B.  $n = 10$  (das optimale  $n$  hängt von der Länge der Eingabezahl ab). Das kann das Verfahren um einen Faktor zwei bis drei beschleunigen, birgt aber auch die Gefahr, daß plötzlich mehr als ein Faktor im Zwischenprodukt erscheint. Da man nun davon ausgehen kann, daß die Vorperiode nicht sehr viel größer als die Periode sein wird, findet man nach  $l$  (oder einem kleinen Vielfachen von  $l$ ) Schritten den Teiler  $p$ .

### Beispiel

Beispielsweise ist  $N = 134023$  zu faktorisieren. Wir wählen  $f(x) = x^2 + 1$ ,  $a_0 = 1$  und erhalten folgende Werte:

$i$	$a_i$	$a_{2i}$	$a_{2i} - a_i$	$\gcd(a_{2i} - a_i, N)$
1	2	5	3	1
2	5	677	672	1
3	26	78931	78905	1
4	677	52126	51449	1
5	56261	27878	105640	1
6	78931	53398	108490	1
7	43607	1799	92215	1
8	52126	128904	76778	1
9	71598	35178	97603	1
10	27878	26317	132462	223

Im zehnten Schritt taucht plötzlich der Faktor 223 auf, so daß wir die vollständige Faktorisierung  $N = 223 \cdot 601$  erhalten. Experimentiert man mit dem Parameter  $c$  etwas herum, so macht man folgende Beobachtung: Für  $c = 2$  hätten wir nur sieben Schritte benötigt, für  $c = 3$  nach 14 Schritten den (größeren) Faktor 601 zuerst erhalten. Bei  $c = 4$  hätten wir mit insgesamt 27 Schritten sogar relatives Pech gehabt. Dadurch wird deutlich, daß es sich eben um eine Monte-Carlo-Methode handelt.

Aufgrund des vergleichsweise hohen Aufwandes für die Berechnung des größten gemeinsamen Teilers lohnt sich das Rho-Verfahren erst bei vier- oder höherstelligen Teilern. In vertretbarer Zeit lassen sich Faktoren bis etwa  $10^{20}$  finden. Beispielsweise gelangen Brent und Pollard im Jahre 1980 die aufsehenerregende Faktorisierung von  $F_8 = 2^{2^8} + 1$ . Sie fanden 1238926361552897 als Teiler und benötigten (mit einer auf die Fermatzahlen zugeschnittenen Rekursionsvorschrift) zwei Stunden Rechenzeit. Mit unseren Standardvorgaben  $f(x) = x^2 + 1$  und  $a_0 = 1$  bräuchten wir 14816648 Iterationen (was gut im Rahmen der theoretischen Vorhersage liegt) - auf einem aktuellen Rechner kostet diese Berechnung gerade einmal wenige Minuten.

## 3.3 Die Produktdarstellung

Nachdem seine zugrundeliegende Idee vorgestellt wurde, kann es nicht schaden, das Verfahren aus einer anderen Perspektive zu betrachten. Die Differenzen  $(a_{2n} - a_n)$  - die wir ja auf einen mit  $N$

### 3 Die Pollard-Rho-Methode

gemeinsamen Teiler testen - können auch folgendermaßen geschrieben werden:

$$\begin{aligned}
 a_{2n} - a_n &= (a_{2n-1}^2 + c) - (a_{n-1}^2 + c) \\
 &= (a_{2n-1}^2 - a_{n-1}^2) \\
 &= (a_{2n-1} + a_{n-1}) \cdot (a_{2n-1} - a_{n-1}) \\
 &= (a_{2n-1} + a_{n-1}) \cdot ((a_{2n-2}^2 + c) - (a_{n-2}^2 + c)) \\
 &= (a_{2n-1} + a_{n-1}) \cdot (a_{2n-2}^2 - a_{n-2}^2) \\
 &= (a_{2n-1} + a_{n-1}) \cdot (a_{2n-2} + a_{n-2}) \cdot (a_{2n-2} - a_{n-2}) \\
 &= \dots \\
 &= (a_{2n-1} + a_{n-1}) \cdot (a_{2n-2} + a_{n-2}) \cdot \dots \cdot (a_n + a_0) \cdot (a_n - a_0) \\
 &= (a_n - a_0) \cdot \prod_{i=0}^{n-1} (a_{n+i} + a_i)
 \end{aligned}$$

Es werden also „hochgradig multiplikative“ Reste getestet. Im Gegensatz zur Probedivision, die pro Iteration nur einen Kandidaten auf Teilbarkeit testet, ist es also möglich, mindestens  $n$  Zahlen parallel zu testen - allerdings muß man hierfür beim Rho-Verfahren erst einmal  $n$  Iterationen absolviert haben. Da die erzeugten Faktoren scheinbar zufällig verteilt und damit voneinander unabhängige Werte darstellen hat man bis zum  $n$ -ten Schritt etwa  $n^2$  Zahlen getestet - was wiederum die Laufzeit von  $\sqrt{p}$  Schritten für einen Faktor  $p$  bestätigt.

#### Parallelisierbarkeit

Die Produktdarstellung hilft auch bei der Beantwortung der Frage nach der Parallelisierbarkeit des Verfahrens: offensichtlich ist die Berechnung *einer* Rho-Sequenz nicht verteilt lösbar, da jedes Folgenglied  $a_{i+1}$  vom Vorgänger  $a_i$  abhängt. Wir können aber  $n$  unabhängige Folgen parallel berechnen lassen. Da  $k$  Iterationen äquivalent zu  $k^2$  Testdivisionen sind, haben wir damit ein Äquivalent von  $n \cdot k^2$  Testdivisionen. Das ist viel schlechter als die (theoretische) Parallelisierung einer Sequenz, die mit einem Äquivalent von  $(n \cdot k)^2 = n^2 \cdot k^2$  Tests aufwarten würde. Eine Parallelisierung ist also möglich, jedoch geht dabei der Vorteil des Verfahrens verloren.

### 3.4 Brent's Verbesserung

Richard P. Brent gab in [Brent80] eine Möglichkeit zur Beschleunigung des Verfahrens an. Dies gelang ihm, indem er einen schnelleren Zyklenfindungsalgorithmus angab. Der Geschwindigkeitsgewinn liegt allerdings bei nur 24%. Grundlage ist nachfolgendes Lemma.

**Lemma 3.4.1** *Sei  $(a_k)$  eine periodische Folge mit Vorperiodenlänge  $m$  und Periodenlänge  $l$ . Sei weiter  $k(n) := 2^{\lceil \log_2 n \rceil}$ . Dann gibt es ein  $n$ , so daß  $a_n = a_{k(n)-1}$ .*

*Beweis.* Wir wählen  $n = 2^{\lceil \log_2 \max(m+1, l) \rceil} + l - 1$ . Da  $l - 1 < 2^{\lceil \log_2 \max(m+1, l) \rceil}$  (d. h. durch Addition von  $l - 1$  wird die nächsthöhere Potenz von 2 nicht überschritten), folgt  $k(n) = 2^{\lceil \log_2 \max(m+1, l) \rceil}$ . Da wegen  $k(n) - 1 \geq m$  das Glied  $a_{k(n)-1}$  bereits in der Periode liegt und  $n - (k(n) - 1) = l$  ist, folgt die Behauptung. □

Aufgrund der Definition gilt  $k(n) \leq n < 2k(n)$ . Brent konnte nun zeigen, daß gewisse solche  $n$  sogar die Relation  $\frac{3}{2}k(n) \leq n < 2k(n)$  erfüllen (siehe hierzu auch [List95]). Das kleinste ist  $n_0 = 3$  (für  $m = 0$  und  $l = 1$ ) oder hat die Gestalt

$$n_0 = 2^{\lceil \log_2 \max(m+1, l) \rceil} + l \left\lceil \frac{k(n) + 1}{l} \right\rceil - 1$$

d. h. hat die Größenordnung von  $\max(m, l)$ . Der verbesserte Algorithmus würde also die Werte  $a_{2^k-1}$  speichern und für  $3 \cdot 2^{k-1} \leq i < 2^{k+1}$  den  $\gcd(a_i - a_{2^k-1}, N)$  berechnen. Insbesondere müssen wir also nicht mehr zwei Folgen parallel berechnen.

## 4 Das $(p-1)$ -Verfahren

Das im folgenden vorgestellte Verfahren stammt ebenfalls von Pollard. Zur Vorbereitung benötigen wir aber noch etwas Zahlentheorie.

**Definition 4.0.2** Seien  $N, B \in \mathbb{N}$ .  $N$  heißt  $B$ -glatt, wenn alle Primteiler von  $N$  kleiner oder gleich  $B$  sind;  $B$ -potenzglatt, wenn dies für alle Primpotenzen gilt.

Beispielsweise ist  $45 = 3^2 \cdot 5$  5-glatt und 9-potenzglatt (aber nicht 5-potenzglatt).

**Satz 4.0.3 (Kleiner Satz von Fermat)** Ist  $p$  eine Primzahl und  $p \nmid a$  für ein beliebiges  $a \in \mathbb{N}$ . Dann gilt  $p \mid a^{p-1} - 1$ .

*Beweis.* Da  $a$  und  $p$  teilerfremd sind, durchlaufen die Vielfachen  $1a, 2a, \dots, (p-1)a$  die gesamte Gruppe  $\mathbb{F}_p^*$ , d. h. es gilt  $1 \cdot 2 \cdot \dots \cdot (p-1) \equiv 1a \cdot 2a \cdot \dots \cdot (p-1)a \pmod{p}$ . Aufgrund der Kürzungsregel folgt  $1 \equiv a^{p-1} \pmod{p}$ . □

Allgemeiner ist der Satz von Euler: Aus  $\gcd(a, m) = 1$  folgt  $m \mid a^{\varphi(m)} - 1$  für beliebige natürliche Zahlen  $a$  und  $m$ .

Aus dem Satz von Fermat läßt sich nun ein einfaches Verfahren zur Faktorzerlegung ableiten: Ist

$$p-1 = n_1 \cdot n_2 \cdot \dots \cdot n_k, \quad n_1 \geq n_2 \geq \dots \geq n_k \quad (4.1)$$

die Primfaktorzerlegung von  $p-1$  und  $B \geq n_1$  eine Schranke, für die  $p-1$   $B$ -potenzglatt ist, dann gilt für jede natürliche Zahl  $m$ , die durch alle Zahlen kleiner oder gleich  $B$  teilbar ist

$$p-1 \mid m \quad \text{und} \quad a^m \equiv 1 \pmod{p}$$

Beispielsweise würde  $m = B!$  diese Eigenschaft erfüllen. Gilt dies nicht für alle Primteiler  $p$  von  $N$  (was bei nicht zu großem  $B$  sehr wahrscheinlich ist), so ist insbesondere  $a^m \not\equiv 1 \pmod{N}$  und wir erhalten mit  $\gcd(a^m - 1, N)$  einen nichttrivialen Teiler von  $N$ . Es stellt sich nur die Frage, wie aufwendig  $a^m \pmod{N}$  zu berechnen ist. Um allgemein den Rest

$$b \equiv a^m \pmod{N}$$

für große  $m$  auszurechnen, zieht man die Binärdarstellung  $m = \sum_{i \in I} 2^i$  (für eine Indexmenge  $I$ ) heran. Dann ist

$$b \equiv a^m \equiv a^{\sum_{i \in I} 2^i} \equiv \prod_{i \in I} a^{2^i} \pmod{N} \quad (4.2)$$

Die Reste  $a^{2^i}$  erhält man durch schrittweises Quadrieren mod  $N$ . Wegen  $|I| \leq \lceil \log_2 m \rceil$  wächst die Zahl der erforderlichen Quadrierungen (und somit der Gesamtaufwand) logarithmisch mit  $m$ .

$B$  muß natürlich so gewählt sein, daß  $B \geq n_1$ . Mit der vereinfachten Wahl  $m = B!$  kann man nun aufsteigend  $a_{i+1} = a_i^{i+1} \pmod{N}$  berechnen und erhält  $a^m \equiv a_B \pmod{N}$ .

### Beispiel

Wir wählen  $N = 10^{34} + 9$ . Der kleinere der beiden Primteiler ist  $p = 2532184185301$  und es ist  $p-1 = 2^2 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 17 \cdot 53 \cdot 89 \cdot 1367$ , d. h.  $p-1$  ist 1367-potenzglatt. Wir finden also  $\gcd(2^{1367!} - 1, N) = p$ .

In der Praxis ist es nicht nötig,  $B!$  als Exponenten  $m$  zu wählen, da die Forderung nach Potenzglattheit i. a. zu stark ist. Es genügt, in  $m$  alle Primteiler von  $p-1$  zu erfassen, wobei nur von den kleineren auch Potenzen zu berücksichtigen sind (es ist unwahrscheinlich, daß große Primteiler von  $p-1$  in echten Potenzen auftreten).

## 4.1 Eine zweite Phase

Das bisherige Vorgehen kann man folgendermaßen auffassen: Alle Berechnungen erfolgen in der multiplikativen Gruppe  $\mathbb{F}_p^*$ . Potenziert man nun ein Element aus dieser Gruppe mit einem der Primteiler  $n_1, \dots, n_k$  von (4.1), so landet man in einer (deutlich kleineren) Untergruppe  $U$ . Da ausschließlich potenziert wird, ist sichergestellt, daß man  $U$  nicht mehr verläßt - bis man 1 als neutrales Element erhält.

Hat man nun bereits  $a^{n_2 \cdots n_k} =: b$  (oder ein Vielfaches dieser Zahl) berechnet, so ist es nicht mehr nötig, mit weiteren Primzahlen zu potenzieren, da keine kleinere Untergruppe (bis auf die 1) mehr existiert. Stattdessen wird aufsteigend  $b^{q_i}$  für Primzahlen  $q_i$  berechnet, bis man  $q_j = n_1$  gefunden hat. Wegen  $b^{q_{i+1}} = b^{q_i + d_i} = b^{q_i} \cdot b^{d_i}$  benötigt man dazu nur die entsprechenden Primzahldifferenzen  $d_i$ . Die zugehörigen  $b^{d_i}$  können einmalig berechnet und gespeichert werden - was eine erhebliche Beschleunigung bewirkt, da nur noch eine Multiplikation ausgeführt wird.

Das Verfahren kann also um eine zweite Phase (auch Continuation genannt) erweitert werden: Neben der Schranke  $B = B_1$  führt man noch eine Schranke  $B_2$  ein, bis zu der man nach dem verbleibenden Primteiler  $n_1$  sucht. Da das deutlich effizienter ist, kann  $B_2$  meist viel größer als  $B_1$  gewählt werden; üblich ist hier z. B.  $B_2 = 50B_1$ . Voraussetzung für den Erfolg ist allerdings die Bedingung  $n_2 \leq B_1$ .

Ein weiterer zu erwähnender Vorteil ist die mögliche Parallelisierbarkeit, da die Werte  $b^{q_i}$  unabhängig voneinander berechnet werden können.

## 4.2 Laufzeitanalyse

Offensichtlich liegt die Laufzeit in  $\mathcal{O}(n_1 \ln n_1)$ . Daher wird im folgenden eine Abschätzung für den größten Primteiler  $n_1$  gegeben.

**Satz 4.2.1 (Primzahlsatz)** Sei  $\pi(x) := \#\{\text{Primzahl } p \mid p \leq x\}$ . Dann gilt  $\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\ln(x)} = 1$ .

Mit Hilfe diesen Satzes (der 1896 von Hadamard und de la Vallée Poussin bewiesen wurde) kann man beispielsweise den mittleren Abstand der Primzahlen kleiner oder gleich  $N$  zu  $\ln N$  abschätzen. Ein anderes wichtiges Resultat ist

**Satz 4.2.2** Sei  $\Omega(N) := \#\text{Primfaktoren von } N$ . Dann ist im Mittel  $\Omega(N) = \ln \ln N$ .

Für eine genauere Herleitung siehe [HaRa17]. Für eine natürliche Zahl  $N$  sei  $N = p_1 \cdot \dots \cdot p_s$  die Primfaktorzerlegung, wobei  $p_1 \geq p_2 \geq \dots \geq p_s$ . Die Zahl  $N/p_1$  hat also  $(s-1)$  Primfaktoren, was zum Ansatz

$$\begin{aligned} s-1 &\approx \ln \ln \frac{N}{p_1} = \ln(\ln N - \ln p_1) \\ &= \ln(\ln N \cdot (1 - \frac{\ln p_1}{\ln N})) = \ln \ln N + \ln(1 - \frac{\ln p_1}{\ln N}) \\ &\approx s + \ln(1 - \frac{\ln p_1}{\ln N}) \end{aligned}$$

führt. D. h. es gilt  $-1 \approx \ln(1 - \frac{\ln p_1}{\ln N})$ , woraus man die Abschätzung  $\ln p_1 \approx (1 - \frac{1}{e}) \ln N \approx 0.632 \ln N$  gewinnt. Für den größten Primteiler können wir also  $p_1 \approx N^{0.632}$ , für den zweitgrößten  $N^{0.632/e} \approx N^{0.233}$  usw. erwarten.

Da die Laufzeit etwa proportional zu  $p_1$  ist, ergibt sich also  $\mathcal{O}(p^{0.632})$  als *mittlere* Laufzeit, wobei die genaue Größe natürlich stark variieren kann. Man vergleiche mit dem Rho-Verfahren, welches eine Laufzeit von  $\mathcal{O}(p^{0.5})$  besitzt. Mit dem  $(p-1)$ -Verfahren konnten auch schon recht große Teiler gefunden werden (eine Liste erreicht man über [ECMNet]).

Ein ähnliches Verfahren, daß die Zerlegung von  $p+1$  benutzt, wurde von Williams vorgeschlagen [Will82].

# 5 Die Methode der elliptischen Kurven

Dieses 1987 von Hendrik Lenstra, jr. [Len87] erfundene Verfahren funktioniert nach demselben Prinzip wie das  $(p-1)$ -Verfahren, hat aber entscheidende Vorteile. Berechnungen werden nicht mehr in  $\mathbb{F}_p^*$ , sondern auf einer elliptischen Kurve über  $\mathbb{F}_p$  ausgeführt.

## 5.1 Grundlegende Definitionen

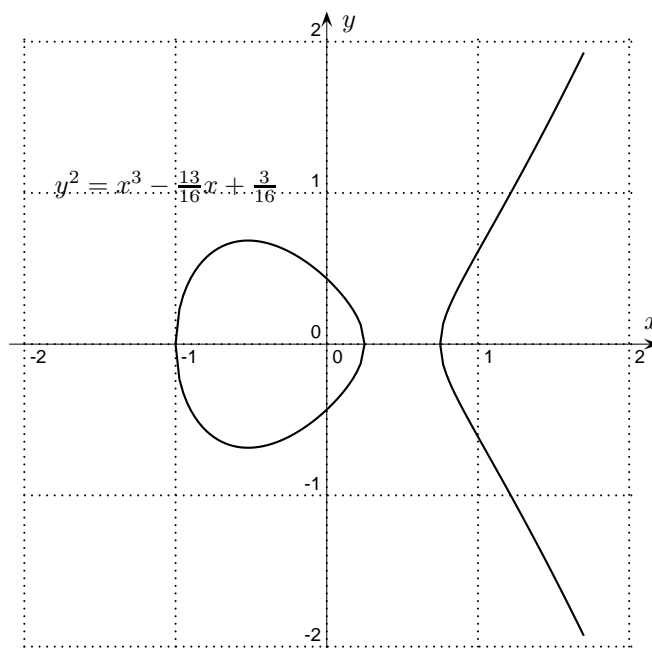
**Definition 5.1.1** Sei  $\mathbb{K}$  ein Körper der Charakteristik  $\neq 2, 3$  und  $x^3+ax+b$  ( $a, b \in \mathbb{K}$ ) ein Polynom ohne mehrfache Nullstellen. Eine elliptische Kurve  $E$  über  $\mathbb{K}$  ist die Menge  $(x, y) \in \mathbb{K}$  mit

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

und einem Punkt  $O$ .

### Beispiel

Wählt man als Körper  $\mathbb{K} = \mathbb{R}$ , so ergibt sich eine Kurve wie in der folgenden Abbildung.



Elliptische Kurve über  $\mathbb{R}$

Mit Hilfe der Cardano'schen Formeln sieht man, daß das Polynom  $x^3 + ax + b$  keine Mehrfachnullstellen besitzt, falls  $4a^3 + 27b^2 \neq 0$ . Sei zunächst  $\mathbb{K} = \mathbb{R}$ , dann können folgende Operationen definiert werden:

**Definition 5.1.2** Sei  $\mathbb{K} = \mathbb{R}$  und  $P, Q$  Punkte auf  $E$ .

1. Ist  $P = O$ , dann definiere  $-P := O$  und  $P + Q := Q$ , d. h.  $O$  ist neutrales Element. Im folgenden ist  $P = (x, y)$  und  $Q \neq O$ .
2.  $-P := (x, -y)$

## 5 Die Methode der elliptischen Kurven

3. Haben  $P$  und  $Q$  verschiedene  $x$ -Koordinaten, schneidet die Gerade  $\overline{PQ}$   $E$  in genau einem Punkt  $R$ . Dann  $P + Q := -R$ . Ist  $\overline{PQ}$  Tangente an  $P$  bzw.  $Q$ , dann ist  $P = R$  bzw.  $Q = R$
4. Aus  $P = (x, y)$  und  $Q = (x, -y)$  folgt  $P + Q = O$
5. Ist  $P = Q$  und  $R$  Schnittpunkt der Tangente an  $P$  bzw.  $Q$ , dann ist  $P + Q := -R$

Man kann zeigen, daß  $E$  eine (additive) abelsche Gruppe ist, und zwar unabhängig von der konkreten Wahl des Körpers  $\mathbb{K}$  [For96]. Auf einer solchen Kurve ist es also möglich, beliebige Punkte zu addieren: Seien dazu  $P = (x_1, y_1)$ ,  $Q = (x_2, y_2)$  sowie  $P, Q \neq O$  und  $Q \neq -P$ , dann kann man die Summe  $P + Q = (x_3, y_3)$  bestimmen über

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = -y_1 + m(x_1 - x_3)$$

wobei

$$m := \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{wenn } P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & \text{wenn } P = Q \end{cases}$$

Über  $\mathbb{K} = \mathbb{R}$  ist  $m$  gerade der Anstieg der Verbindungsgeraden  $\overline{PQ}$  (bzw. im Falle  $P = Q$  der Tangente).

### Beispiel

Es sind  $E : y^2 = x^3 + x + 6$  und  $P = (2, 4)$ ,  $Q = (3, 6)$  gegeben. Als Koordinaten für die Summe ergeben sich  $x_{P+Q} = \left(\frac{6-4}{3-2}\right)^2 - 2 - 3 = -1$  und  $y_{P+Q} = -4 + 2 \cdot (2 - (-1)) = 2$ , d. h.  $P + Q = (-1, 2)$ .

Wie in jeder anderen Gruppe kann man zu einem Gruppenelement  $P$  die Ordnung definieren, die im weiteren Verlauf große Bedeutung bekommt.

**Definition 5.1.3** Die Ordnung  $\text{ord}(P)$  eines Punktes  $P$  ist die kleinste natürliche Zahl  $n$ , für die  $n \cdot P = O$ .

## 5.2 Elliptische Kurven mod $p$

Sei wieder  $N$  natürliche Zahl und  $p$  ein (unbekannter) Primteiler von  $N$ . Dann kann man die elliptische Kurve  $E_p$  über  $\mathbb{K} = \mathbb{F}_p$  mit  $P = (x, y) \in \mathbb{Z}^2$  betrachten:

$$P \bmod p = (x \bmod p, y \bmod p)$$

Da uns  $p$  nicht bekannt ist - wir aber dennoch Punktadditionen ausführen wollen - werden im folgenden alle Operationen modulo  $N$  ausgeführt. Die so berechneten Reste kann man als Vertreter der Reste mod  $p$  (also der Punkte auf der elliptischen Kurve) ansehen:

$$P \bmod N = (x \bmod N, y \bmod N)$$

Für eine Addition müssen  $(x_2 - x_1)$  und  $2y_1$  modulo  $p$  (bzw.  $N$ ) invertiert werden. Die Invertierung mod  $N$  ist hierbei wieder konform mit der Invertierung mod  $p$ . Realisiert wird sie mit dem Erweiterten Euklidischen Algorithmus.

### Beispiel

$E : y^2 = x^3 + x + 6$ ,  $N = 187$ ,  $P = (-1, 2)$ . Um  $2P = P + P$  zu berechnen, muß  $2y_1 \equiv 4 \pmod{187}$  invertiert werden: Wegen  $4^{-1} \equiv 47 \pmod{187}$  ist

$$m = (3(-1)^2 + 1) \cdot 47 = 1$$

$$x_{2P} = 1^2 - (-1) - (-1) = 3$$

$$y_{2P} = -2 + 1(-1 - 3) = -6$$

d. h.  $2P = (3, -6)$ .

## 5.3 Lenstra's Algorithmus

Da wir mod  $N$  rechnen, arbeiten wir nicht auf einer elliptischen Kurve (und auch nicht in einer Gruppe). Es kann nämlich sein, daß eine Punktaddition nicht ausführbar ist, falls die Addition als Ergebnis das neutrale Element von  $E_p$  ergibt. Gemäß der Additionsvorschrift sind dann die  $x$ -Koordinaten beider Punkte identisch oder die  $y$ -Koordinaten gleich 0. In diesem Fall müßte also  $\gcd(x_2 - x_1, N) \neq 1$  oder  $\gcd(2y_1, N) \neq 1$  gelten:

**Satz 5.3.1** *Seien  $P_1 = (x_1, y_1)$  und  $P_2 = (x_2, y_2)$ . Dann gilt:  $\gcd(x_2 - x_1, N) > 1$  ( $\gcd(2y_1, N) > 1$ )  $\Leftrightarrow \exists$  Primteiler  $p \mid N: P_1 + P_2 \bmod p = O \bmod p$*

Der Algorithmus von Lenstra spekuliert nun darauf, genau diesen Fall herbeizuführen, da wir dann einen nichttrivialen Teiler von  $N$  gefunden haben. Um das neutrale Element von  $E_p$  zu erzeugen, geht man nach demselben Schema vor wie im  $(p-1)$ -Verfahren: Ausgehend von einem Kurvenpunkt  $P$  berechnet man ein Vielfaches  $m \cdot P$ , wobei  $m$  alle Primzahlen kleiner oder gleich einer vorgegebenen Schranke  $B$  als Teiler enthält (bzw. analog zum  $(p-1)$ -Verfahren auch höhere Potenzen der kleineren Primzahlen). Wenn alle Primteiler  $n_1, \dots, n_k$  der Punktordnung  $\text{ord}(P)$  diese Schranke nicht überschreiten, haben wir  $O$  berechnet. In der Praxis wählt man auch hier zwei unterschiedliche Schranken  $B_1$  und  $B_2$ . Um das Vielfache  $m \cdot P$  schnell auszurechnen, bedient man sich desselben Algorithmus wie in (4.2).

Insgesamt ergibt sich also folgendes Vorgehen:

### Algorithmus

1. Wähle eine Kurve  $E: y^2 = x^3 + ax + b$  mit  $a, b \in \mathbb{Z}$ , so daß  $\gcd(4a^3 + 27b^2, N) = 1$ , einen Startpunkt  $P \in E$  und zwei Schranken  $B_1, B_2 \in \mathbb{N}$ , wobei  $B_1 \leq B_2$ .
2. Wähle  $m \in \mathbb{N}$  so, daß alle Primzahlen  $\leq B_1$  Teiler von  $m$  sind, wobei kleine Primzahlen mit höherer Vielfachheit vorkommen sollten.
3. Phase 1: Berechne  $Q := m \cdot P$  durch schnelles Potenzieren. Ist dabei eine Addition  $P_1 + P_2$  nicht möglich, so liefert  $\gcd(x_2 - x_1, N)$  (bzw.  $\gcd(2y_1, N)$ ) sehr wahrscheinlich den Primfaktor  $p$ . ( $\rightarrow$  Abbruch)
4. Phase 2: Berechne aufsteigend  $q \cdot Q$  für alle Primzahlen  $q \in (B_1, B_2]$ . Ist dabei wieder eine Addition  $P_1 + P_2$  nicht ausführbar, so bekommt man über  $\gcd(x_2 - x_1, N)$  (bzw.  $\gcd(2y_1, N)$ ) vermutlich den Primfaktor  $p$ . ( $\rightarrow$  Abbruch)  
Bei Mißerfolg wird mit 1. fortgefahren.

Für Phase 2 genügt es, die Vielfachen von  $2 \cdot Q$  vorzuberechnen und für eine spätere Addition zu speichern, da man nur die den Primzahlabständen  $p_{i+1} - p_i$  entsprechenden Punkte  $(p_{i+1} - p_i) \cdot Q$  addieren muß.

Die Hoffnung besteht also darin, durch ständige Variation der Kurvenparameter im ersten Schritt Einfluß auf die Ordnung  $\text{ord}(P)$  des Startpunktes  $P$  nehmen zu können. Unter der Annahme, daß die zufällige Wahl der Parameter zu einer zufälligen Ordnung führt (was noch nicht bewiesen werden konnte), bekommt man also nach hinreichend vielen Versuchen eine recht hohe Chance, den Primfaktor  $p$  zu finden. Das ist der wesentliche Unterschied zum  $(p-1)$ -Verfahren, bei dem die Ordnung der zugrundeliegenden Gruppe  $\mathbb{F}_p^*$  unveränderbar ist.

Eine Aussage über die Anzahl der Kurvenpunkte (und damit über die Gruppenordnung) gibt

**Satz 5.3.2 (Hasse)** *Sei  $p$  eine Primzahl und  $E_p$  die zugehörige elliptische Kurve über  $\mathbb{F}_p$ . Dann gilt  $||E_p| - p - 1| < 2\sqrt{p}$*

Es ist also  $|E_p| \approx p \approx \text{ord}(P)$ , d. h. die Ordnung ist etwa so groß wie  $p$ . Experimente haben ergeben, daß bei zufälliger Wahl der Parameter alle Werte innerhalb des Hasseintervalls mit etwa gleicher Wahrscheinlichkeit angenommen werden.

**Beispiel**

$N = 48551357401$  soll faktorisiert werden. Wir wählen als Schranke  $B = 15$  (um das Beispiel einfach zu halten verzichten wir hier auf die zweite Schranke  $B_2$ ),  $m = 360360 = 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13$  (d. h.  $m$  ist 15-potenzglatt). An Stelle der direkten Darstellung  $m = 1010111111110101000_2$  können wir auch die Kurzform  $m = 101100000000101000_2 - 10000000_2$  verwenden. Dann gilt  $m = 2^3 + 2^5 - 2^7 + 2^{15} + 2^{16} + 2^{18}$ . Als elliptische Kurve dient  $E : y^2 = x^3 + 12x - 12$  mit dem Startpunkt  $P = (1, 1)$ . Nun wird fortschreitend berechnet (mit dem schnellen Potenzierungsalgorithmus wie in (4.2)):

$$2^1 P = (36413518105, 18206758625)$$

$$2^2 P = (1043472969, 824250396)$$

$$2^3 P = (11302484541, 10152896351)$$

$$\vdots$$

$$2^{17} P = (42439169295, 19606302175)$$

$$2^{18} P = (10947169822, 37588903780)$$

Jetzt kann das Produkt  $m \cdot P$  berechnet werden:

$$2^3 P + 2^5 P = (16392437179, 41251256536)$$

$$2^3 P + 2^5 P - 2^7 P = (35239456057, 23440611854)$$

$$2^3 P + 2^5 P - 2^7 P + 2^{15} P = (23303441326, 46657314722)$$

$$2^3 P + 2^5 P - 2^7 P + 2^{15} P + 2^{16} P = (13486387219, 27823258603)$$

$$2^3 P + 2^5 P - 2^7 P + 2^{15} P + 2^{16} P + 2^{18} P = \underbrace{(13486387219, 27823258603)}_{x_1}$$

$$+ \underbrace{(10947169822, 37588903780)}_{x_2}$$

Die Addition schlägt fehl wegen  $x_1 - x_2 \equiv 2539217397 \pmod{N}$  und  $\gcd(2539217397, N) = 54547$ . Wir finden also tatsächlich die Faktorisierung  $N = 54547 \cdot 890083$ . Ein kurzer Test ergibt, daß  $m^* = m/13 = 27720$  nicht zum Erfolg führt, d. h. 13 muß also ein Teiler von  $\text{ord}(P)$  sein.

## 5.4 Laufzeitanalyse

Da der Algorithmus nur zum Erfolg führen kann, wenn  $\text{ord}(P)$  bezüglich der selbstgewählten Schranke  $B$  (bzw.  $B_1$  und  $B_2$ ) potenzglatt ist, stellt sich die Frage, wie viele Kurven berechnet werden müssen bzw. wie  $B$  gewählt werden sollte, bis der gewünschte Fall eintritt. Antwort gibt ein Satz aus der Zahlentheorie, der eine Aussage über die Verteilung glatter Zahlen macht ([CEP83]).

**Satz 5.4.1 (Canfield/Erdős/Pomerance)** Gegeben sei die Glattheitsfunktion

$$\psi(x, y) := \#\{n \in \mathbb{N} \mid n \leq x, n \text{ besitzt keinen Primfaktor } p > y\}$$

Dann gilt  $\psi(x, y) = x \cdot u^{-u(1+o(1))}$ , wobei  $u := \frac{\ln x}{\ln y}$ .

Man kann den Satz folgendermaßen auffassen: Ist  $N \leq x$  zufällig gewählt und  $y \ll x$ , so ist die *Wahrscheinlichkeit*, daß  $N$  keinen Primfaktor  $> y$  besitzt (d. h.  $y$ -glatt ist), etwa gleich  $u^{-u}$ . Sind beispielsweise  $x = 10^{25}$  und  $y = 10^{10}$ , so beträgt die Wahrscheinlichkeit für  $y$ -Glattheit  $\left(\frac{25}{10}\right)^{-25/10} \approx 0.1012$ .

Für Lenstra's Algorithmus bedeutet das: Unter der Annahme, daß sich die Gruppenordnung in Abhängigkeit der Kurvenparameter *zufällig* verhält, müßten bei Wahl von  $k = 10^{10}$  in der Erwartung zehn Kurven berechnet werden, bis  $\text{ord}(P) \approx 10^{25}$   $k$ -glatt ist. Nach dem Satz von Hasse

liegt der gesuchte Primteiler  $p$  in derselben Größenordnung wie  $\text{ord}(P)$ , d. h. wir bekommen eine Aufwandsabschätzung für das Finden eines beliebigen Primteilers  $p$ .

Wie sollte nun  $k$  gewählt werden, um den Aufwand zu minimieren? Für die Berechnung von  $P^k!$  pro Kurve durch schnelles Potenzieren sind  $k \cdot \ln k$  Multiplikationen erforderlich. Mit  $u := \frac{\ln p}{\ln k}$  sind nach dem Satz von Canfield/Erdős/Pomerance ungefähr  $u^u$  Kurven zu testen. Für die zu erwartende Laufzeit  $L(p)$  ergibt sich also

$$\begin{aligned} L(p) &= u^u \cdot k \cdot \ln k \\ &= u^u \cdot p^{1/u} \cdot \frac{1}{u} \cdot \ln p \\ &= u^{u-1} \cdot p^{1/u} \cdot \ln p \end{aligned} \tag{5.1}$$

Gesucht ist nun jenes  $u_{\text{opt}}$ , für das  $L(p)$  minimal wird. Da  $L$  differenzierbar von  $u$  abhängt, erhält man das Minimum über den Ansatz  $\frac{dL}{du} = 0$ , d. h.

$$u^{u-3} \cdot p^{1/u} \cdot (\ln p \cdot (u^2 \cdot \ln u + u(u-1)) - (\ln p)^2) = 0$$

Vereinfacht ergibt sich

$$\begin{aligned} u^2 \cdot \ln u + u(u-1) &= \ln p \\ u \sqrt{\ln u - \frac{1}{u} + 1} &= \sqrt{\ln p} \end{aligned}$$

Aus  $p \rightarrow \infty$  folgt erkennbar  $u_{\text{opt}} \rightarrow \infty$ . Um eine Aussage über das asymptotische Verhalten von  $u_{\text{opt}}$  (in Abhängigkeit von  $p$ ) zu gewinnen, genügt es demnach,

$$u \sqrt{\ln u} = \sqrt{\ln p}$$

anzunehmen. Setzt man in den störenden rechten Faktor der linken Seite die Näherung  $u \approx \sqrt{\ln p}$  ein, so ergibt sich

$$u_{\text{opt}} = u(p) = \mathcal{O} \left( \sqrt{\frac{2 \ln p}{\ln \ln p}} \right)$$

Mit der Kurznotation  $q := \ln p$  und  $u = u_{\text{opt}} = \sqrt{\frac{2q}{\ln q}}$  eingesetzt in (5.1) kommt man zur Abschätzung

$$\begin{aligned} L(p) &= (2q)^{\frac{1}{2} \sqrt{\frac{2q}{\ln q}}} \cdot \ln q \cdot \left( \frac{q}{\ln q} \right)^{\frac{1}{2} (\sqrt{\frac{2q}{\ln q}} + 1)} \\ &= \exp \left( \frac{1}{2} \ln(2q) \sqrt{\frac{2q}{\ln q}} + \ln \ln q + \frac{1}{2} \ln \frac{q}{\ln q} \left( \sqrt{\frac{2q}{\ln q}} + 1 \right) \right) \\ &= \exp \left( \frac{1}{2} \ln q \sqrt{\frac{(2+o(1))q}{\ln q}} + \frac{1}{2} \ln q \sqrt{\frac{(2+o(1))q}{\ln q}} \right) \\ &= \exp \left( \sqrt{(2+o(1)) \ln p \cdot \ln \ln p} \right) \end{aligned}$$

In [Brent99] wird diese Laufzeit über einen etwas modifizierten Weg hergeleitet.

Es stellt sich die Frage, wie  $u$  (und damit auch  $B$ ) gewählt werden sollte, da der fragliche Primfaktor  $p$  ja nicht bekannt ist. In der Praxis nimmt man eine bestimmte Größe (z. B.  $10^{15}$ ) für  $p$  an und berechnet für das zugehörige  $B$  entsprechend viele Kurven (so daß ein  $p$  dieser Größenordnung sehr wahrscheinlich gefunden wird). Selbst wenn  $p$  deutlich größer ist, erhält man eine gewisse Chance; ansonsten wechselt man zu einer höheren Größenordnung (etwa  $10^{20}$ ). Gegenüber dieser ist der Aufwand für den kleineren Test dann vernachlässigbar klein.

Praktische Untersuchungen haben weiterhin ergeben, daß das optimale Verhältnis  $B_2/B_1$  von den Zeiten  $t_1, t_2$  für Phase 1 bzw. 2 abhängt. Demnach sollte  $B_2$  gegenüber  $B_1$  so gewählt werden, daß sich ein Verhältnis von 3 : 2 bis 2 : 1 zwischen  $t_1$  und  $t_2$  ergibt.

## 5.5 Verbesserungen des Verfahrens

Der oben vorgestellte Algorithmus macht die Idee und das grundsätzliche Vorgehen deutlich, wird in der Praxis so aber nicht umgesetzt, da er viel zu ineffizient ist. Im folgenden werden die drei wesentlichsten Verbesserungen beschrieben, Details erfährt man in [Mül95], [AtMo92].

Da in Phase 2 aufsteigend Punkte der Form  $q \cdot Q$  für Primzahlen  $q \in (B_1, B_2]$  berechnet werden, müssen hier ungefähr  $\frac{B_2}{\ln B_2}$  Punktadditionen ausgeführt werden. Das ist aber nicht notwendig, da man sich folgendermaßen behelfen kann: Für ein fest gewähltes  $w \in \mathbb{N}$  ist die Darstellung

$$\begin{aligned} q &= vw - u \\ q \cdot Q &= vw \cdot Q - u \cdot Q \end{aligned}$$

mit  $1 \leq u \leq w$  eindeutig. Wählt man nun  $w$  in der Größenordnung  $\sqrt{B_2}$ , so sind  $u$  und  $v$  in ihrer Größe durch ungefähr  $\sqrt{B_2}$  beschränkt. Die Vielfachen  $vw \cdot Q$  sowie  $u \cdot Q$  können vorberechnet werden, was nur  $2\sqrt{B_2}$  Punktadditionen kostet. Ist  $q \cdot Q = O$ , so sind die  $x$ -Koordinaten  $x_1, x_2$  von  $vw \cdot Q$  und  $u \cdot Q$  kongruent modulo  $p$ . Man kann also direkt  $\gcd(x_1 - x_2, N)$  testen, ohne die Differenz der Punkte auszurechnen. Analog zum Pollard-Rho-Verfahren kann man mehrere dieser  $(x_1 - x_2)$  aufmultiplizieren und erst dann den gemeinsamen Teiler bestimmen.

Die in 5.1.1 definierten elliptischen Kurven liegen in der sogenannten Weierstraß-Form vor. Eine andere Parametrisierung dieser Kurven geht im wesentlichen auf Peter L. Montgomery zurück. Kurven mit dieser Parametrisierung haben die Gestalt

$$by^2z = x^3 + ax^2z + xz^2$$

die man auch als Montgomery-Chudnowsky-Form bezeichnet. Man kann Punkte  $(x, y, z)$  dieser Form recht leicht in die Weierstraß-Form transformieren. Der Vorteil dieser Darstellung besteht darin, Punkte ganz ohne teure Invertierung addieren zu können. Es sei daran erinnert, daß bei einer Punktaddition die Invertierung modulo  $N$  die aufwendigste Operation ist. Das Rechnen in Montgomery-Chudnowsky-Darstellung bringt also eine deutliche Beschleunigung für Phase 1.

Eine weitere Verbesserung bekommt man durch die Verwendung sogenannter Kurvengeneratoren. Ein solcher Generator erzeugt aus einem Zufallswert  $\sigma$  Kurvenparameter  $a, b$  derart, daß die Ordnung der zugehörigen Kurve immer durch ein bestimmtes  $k$  teilbar ist. Das ist essentiell für die Elliptische-Kurven-Methode, da dann die Glattheitsforderung viel eher erfüllt ist (der Aufwand für das Finden eines Primteilers  $p$  ist dann so groß wie der Aufwand für einen Primteiler der Größe  $p/k$  ohne Generator). Bekannt sind Generatoren für  $k = 8, 12, 16$  (siehe auch [BCDH00]). Der folgende 12-Generator für die Montgomery-Chudnowsky-Parametrisierung (der sich nach praktischen Untersuchungen sogar fast so gut verhält wie ein 24-Generator) wird beispielsweise in *gmp-ecm* verwendet: Sei  $\sigma \notin \{0, 1, 5\}$  zufällig gewählt und seien

$$u := \sigma^2 - 5, \quad v := 4\sigma$$

Dann erhält man als Kurvenparameter

$$a = \frac{(v - u)^3(3u + v)}{4u^3v} - 2$$

und mit  $(u^3, -, v^3)$  einen gültigen Startpunkt.  $b$  und die  $y$ -Koordinate des Startpunktes werden bei Berechnungen auf der Kurve nicht benötigt. Die Verwendung von Generatoren macht sich gerade beim Finden kleinerer Primfaktoren deutlich bemerkbar.

### Parallelisierbarkeit

Mit ECM können nun sehr effizient Faktoren  $< 10^{30}$  gefunden werden. Da sämtliche berechneten Kurven unabhängig voneinander sind, läßt sich das Verfahren aufwandslos parallelisieren. Für die Berechnung von  $k$  Kurven können beispielsweise  $k$  parallel (und völlig unabhängig) arbeitende Rechner herangezogen werden. Dadurch ist es möglich, auch Primfaktoren  $> 10^{50}$  mit vertretbarem Aufwand zu finden [ECMNet]. Die zu erwartende Gesamtlaufzeit steigt aber sehr schnell an: Das Aufwandsverhältnis zwischen einem Primfaktor mit 100 und einem mit 50 Dezimalstellen beträgt etwa  $10^6$ .

## 6 Das Quadratische Sieb

Bereits in den 20er Jahren des letzten Jahrhunderts wurde die Fermatmethode, die zu zerlegende Zahl als Differenz zweier Quadrate zu schreiben, in den Arbeiten von M. Kraitchik weiterentwickelt. Wesentliches Ziel war das Finden einer Kongruenz

$$x^2 \equiv y^2 \pmod{N}$$

Das ist identisch mit der Darstellung

$$(x + y)(x - y) \equiv 0 \pmod{N}$$

Ist diese Kongruenz nichttrivial, d. h. sind  $(x + y) \not\equiv 0 \pmod{N}$  und  $(x - y) \not\equiv 0 \pmod{N}$ , so erhält man zwei echte Teiler durch Berechnung von  $t_1 = \gcd(x + y, N)$  und  $t_2 = \gcd(x - y, N)$ . Die entscheidende Idee besteht nun darin, nicht direkt zwei kongruente Quadrate zu suchen, sondern solche aus quadratischen Resten zu kombinieren. Diese Idee wurde ab 1982 von Carl Pomerance ([Pom85], [Pom96]) zu einem systematischen Verfahren weiterentwickelt. In den folgenden Abschnitten wird die genaue Vorgehensweise beschrieben.

Als einführendes Beispiel gehen wir für  $N = 517631$  wie bei Fermat vor und erzeugen (beginnend bei  $x_0 = \lceil \sqrt{N} \rceil$ ) aufsteigend die quadratischen Reste  $x_i^2 \equiv r_i \pmod{N}$ . Wir erhalten u. a. folgende Werte:

$$724^2 \equiv 5 \cdot 7 \cdot 11 \cdot 17 \pmod{N}$$

$$739^2 \equiv 2 \cdot 5 \cdot 7 \cdot 11 \cdot 37 \pmod{N}$$

$$741^2 \equiv 2 \cdot 5^2 \cdot 17 \cdot 37 \pmod{N}$$

Man kann ohne Mühe erkennen, daß das Produkt der drei rechten Seiten ein Quadrat ergibt. Nach Zusammenmultiplizieren erhalten wir

$$(724 \cdot 739 \cdot 741)^2 \equiv (2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 17 \cdot 37)^2 \pmod{N}$$

$$473961^2 \equiv 351126^2 \pmod{N}$$

Das ist eine nichttriviale Kongruenz, mit der man sofort die Zerlegung  $N = 431 \cdot 1201$  erhält.

Ziel ist es also, quadratische Reste derart zu kombinieren, daß ihre Primfaktoren im Produkt in gerader Potenz erscheinen. In der so gefundenen Kongruenz kann dann jeder Primfaktor  $p$  von  $N$  mit theoretisch gleicher Wahrscheinlichkeit in den Faktoren  $(x + y)$  und  $(x - y)$  enthalten sein; bei insgesamt  $k$  Primfaktoren ist die Wahrscheinlichkeit, eine nichttriviale Kongruenz gefunden zu haben, also gleich  $1 - \left(\frac{1}{2}\right)^{k-1}$ .

Das Verfahren läßt sich nun in zwei Phasen unterteilen: Im *Siebschritt* werden hinreichend viele quadratische Reste  $r_i$  erzeugt, während diese im *Auswahlschritt* (auch *Matrixschritt* genannt) zu einem Quadrat kombiniert werden. Beide Teilschritte werden im folgenden vorgestellt.

### 6.1 Siebschritt

Wir wollen uns nun damit beschäftigen, wie wir geeignete quadratische Reste  $r_i$  erzeugen können. Wie im Beispiel vorgeführt wurde, werden wir dazu die Primfaktorzerlegung aller  $r_i$  heranziehen. Es ist klar, daß wir keine großen Primfaktoren in den Resten haben wollen. Denn um solche Reste zu einem Quadrat kombinieren zu können, müssen alle Primfaktoren in gerader Anzahl vorkommen - desto größer die Faktoren also sind, desto unwahrscheinlicher ist es, geeignete Reste zu finden, die diese enthalten. Wir werden also nur solche Reste benutzen, die bezüglich einer Schranke  $B$  glatt sind. Dazu definieren wir uns den Begriff der *Faktorbasis*.

**Definition 6.1.1** Sei  $B \in \mathbb{N}$ . Eine Menge  $F = \{-1, p_1, \dots, p_{k-1}\}$  mit Primzahlen  $p_1, \dots, p_{k-1} \leq B$  heißt Faktorbasis bezüglich  $B$ .

Quadratische Reste, die wir verwenden wollen (sogenannte Relationen), müssen also über der Faktorbasis zerlegbar sein. Da auch negative Reste erwünscht sind, wird  $-1$  in die Faktorbasis mit aufgenommen. Um quadratische Reste zu erzeugen, definieren wir uns ein *Siebpolynom*  $Q(i)$  der Form

$$Q(i) = \left(i + \lfloor \sqrt{N} \rfloor\right)^2 - N$$

Das Siebpolynom bestimmt im wesentlichen die Elemente der Faktorbasis  $F$ , da nicht jede Primzahl  $p_j$  ein Teiler von  $Q(i)$  sein kann. Um das zu sehen, benötigen wir noch die Definition des Legendre-Symbols.

**Definition 6.1.2** Sei  $p$  eine ungerade Primzahl und  $a \in \mathbb{Z}$ . Dann ist das Legendre-Symbol  $\left(\frac{a}{p}\right)$  definiert durch

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & , \text{ wenn } p \mid a \\ 1 & , \text{ wenn } a \text{ quadratischer Rest modulo } p \text{ ist} \\ -1 & , \text{ wenn } a \text{ kein quadratischer Rest modulo } p \text{ ist} \end{cases}$$

Eine Primzahl  $p$  kann nur dann ein Teiler von  $Q(i)$  sein, falls  $(i + \lfloor \sqrt{N} \rfloor)^2 \equiv N \pmod{p}$ , woraus  $\left(\frac{N}{p}\right) = 1$  folgt. Beim Aufbau der Faktorbasis wird also für jede Primzahl  $\leq B$  das Legendre-Symbol ausgewertet. Für große Primzahlen kann das durch Ausnutzung der Rechenregeln für das Legendre-Symbol (zu finden z. B. in [For96]) geschehen.

Für betragskleine  $i$  erzeugt  $Q(i)$  quadratische Reste in der Größenordnung  $\sqrt{N}$ . Wichtig für die Laufzeit des späteren Auswahlsschrittes ist, daß die so gewonnenen Reste nicht zu groß werden. Daher definieren wir das *Siebintervall*

$$I_\varepsilon = \{i \in \mathbb{Z} \mid -N^\varepsilon \leq i \leq N^\varepsilon\}$$

mit  $0 \leq \varepsilon \leq \frac{1}{2}$ . Für alle  $i \in I_\varepsilon$  ist damit gewährleistet, daß

$$Q(i) = \mathcal{O}(N^{\frac{1}{2}+\varepsilon})$$

Ein fixiertes  $\varepsilon$  ist aber nicht nötig, für  $N \rightarrow \infty$  kann man  $\varepsilon$  abklingen lassen, so daß die erzeugten Reste von der Größenordnung  $N^{\frac{1}{2}+o(1)}$  sind.

Wie wird nun am einfachsten festgestellt, ob  $Q(i)$  über der Faktorbasis zerfällt? Der naive Ansatz,  $Q(i)$  durch alle Zahlen in  $F$  zu dividieren, ist zu aufwendig. Stattdessen bedient man sich eines Siebverfahrens, das auf Richard Schroepfel zurückgeht. Man prüft leicht nach, daß

$$p^s \mid Q(i) \Leftrightarrow p^s \mid Q(x + l \cdot p^s), \quad l \in \mathbb{Z}, s \in \mathbb{N}$$

Das führt auf die Idee, einmalig die Kongruenz

$$Q(i) \equiv 0 \pmod{p_j^s} \tag{6.1}$$

für alle in Frage kommenden Potenzen der  $p_j$  aus der Faktorbasis zu lösen und danach alle weiteren Lösungen durch Addition von  $l \cdot p_j^s$  zu erhalten. Da  $Q(i)$  ein quadratisches Polynom ist, besitzt (6.1) - falls  $\left(\frac{N}{p_j}\right) = 1$  - modulo einer ungeraden Primzahl  $p_j$  genau zwei Wurzeln, die man effizient über den Shanks-Tonelli-Algorithmus bestimmen kann. Darüberhinaus kann man zeigen (beispielsweise in [For96]), daß dann auch modulo jeder Potenz  $p_j^s$  eindeutige Wurzeln existieren, die ebenfalls effizient ausrechenbar sind.

Im Siebschritt wird nun eine Tabelle mit Zeileneinträgen  $r_i = Q(i)$  und Spalteneinträgen  $p_j$  erstellt. Für alle erzeugten Reste wird dann für jedes  $p_j$  und  $s \in \mathbb{N}$  die Kongruenz  $Q(i) \equiv 0 \pmod{p_j^s}$  gelöst und die Lösungen  $i + l \cdot p_j^s$  in der Tabelle markiert - man kommt hier also ohne rechenintensive

Auswertung von  $Q(i)$  aus. Um herauszufinden, ob ein Rest über der Faktorbasis zerlegbar ist, können die Tabelleneinträge  $r_i$  parallel durch  $p_j$  dividiert werden - falls der entsprechende Eintrag markiert („gesiebt“) wird. Ist am Ende des Siebschrittes ein Rest gleich 1, dann konnte er vollständig über  $F$  zerlegt werden.

Für unser Eingangsbeispiel  $N = 517631$ , die Faktorbasis  $F = \{-1, 2, 5, 7, 11, 13, 17, 23, 37\}$  und das Siebintervall  $I = [-24, 24]$  ergibt sich für die ausgesiebten Reste die folgende Tabelle:

$i$	$x_i$	$Q(i)$	-1	2	5	7	11	13	17	23	37
-24	695	-34606	1	1	-	-	3	1	-	-	-
-15	704	-22015	1	-	1	1	-	-	1	-	1
-10	709	-14950	1	1	2	-	-	1	-	1	-
-2	717	-3542	1	1	-	1	1	-	-	1	-
2	721	2210	-	1	1	-	-	1	1	-	-
5	724	6545	-	-	1	1	1	-	1	-	-
15	734	21125	-	-	3	-	-	2	-	-	-
20	739	28490	-	1	1	1	1	-	-	-	1
22	741	31450	-	1	2	-	-	-	1	-	1

Siebtabelle für  $N = 517631$ 

## 6.2 Auswahlsschritt

Sind hinreichend viele Relationen  $Q(i_1), \dots, Q(i_l)$  gefunden (wie viele genau benötigt werden wird später noch näher erklärt), muß eine Auswahlmenge  $\{i_{m_1}, i_{m_2}, \dots, i_{m_n}\} \subseteq I$  gefunden werden, so daß die Kongruenz

$$Q(i_{m_1})Q(i_{m_2}) \cdots Q(i_{m_n}) \equiv (x_{i_{m_1}} x_{i_{m_2}} \cdots x_{i_{m_n}})^2 \pmod{N}$$

erfüllt ist. Damit die linke Seite zu einem Quadrat kombiniert werden kann, müssen die darin enthaltenen Elemente der Faktorbasis in gerader Potenz vorkommen. Hierfür ist die genaue Größe der Exponenten  $s_{ij}$  nicht wichtig - sondern nur, ob sie gerade oder ungerade ist. Gesucht sind also geeignete  $z_1, \dots, z_l \in \{0, 1\}$ , so daß

$$s_{1j}z_1 + s_{2j}z_2 + \dots + s_{lj}z_l \equiv 0 \pmod{2}$$

gilt. Da das für alle  $j$  (d. h. für alle Faktorbasiselemente gleichzeitig) erfüllt sein muß, gelangt man zu einem Gleichungssystem

$$Az \equiv 0 \pmod{2} \tag{6.2}$$

mit den Komponenten

$$A := \begin{pmatrix} s_{11} & \cdots & s_{l1} \\ \vdots & \ddots & \vdots \\ s_{1k} & \cdots & s_{lk} \end{pmatrix} \pmod{2}, \quad z := \begin{pmatrix} z_1 \\ \vdots \\ z_l \end{pmatrix}$$

wobei dem Vektor  $z$  die Auswahlfunktion zukommt; er wählt genau die Relationen aus, die am Ende die gesuchte Lösung ergeben.

Bei (6.2) handelt es sich um ein lineares Gleichungssystem über  $\mathbb{F}_2$ , das z. B. mit Gaußelimination gelöst werden kann. Um sicherzustellen, daß mindestens eine nichttriviale Lösung existiert, muß  $l > k$  gelten (was aber nicht zwingend notwendig ist). Je größer also die Faktorbasis ist, desto mehr Relationen müssen gesammelt werden und desto umfangreicher wird das entsprechend zu lösende Gleichungssystem ausfallen.

Im Beispiel ergibt sich für die Matrix  $A$  und den Lösungsvektor  $z$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{und} \quad z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Die gesuchte Lösung erhält man also durch Kombination der Relationen 6, 8 und 9 ( $i = 5, 20, 22$ ).

### 6.3 Laufzeitanalyse

Nach einer Untersuchung von Carl Pomerance schlägt das Sieben von insgesamt  $n$  gewonnenen quadratischen Resten mit  $n \cdot \ln \ln B$  zu Buche. Da man für jede Primzahl  $p$  aus der Faktorbasis  $n/p$  Markierungen bzw. Divisionen benötigt, erhält man durch Anwendung des Primzahlsatzes

$$\sum_{p \in F} \frac{1}{p} = \mathcal{O} \left( \int_2^B \frac{dt}{t \cdot \ln t} \right) = \mathcal{O}(\ln \ln B)$$

also einen Aufwand von  $\ln \ln B$  für jede Operation.

Da nur Reste verwendet werden, die über der Faktorbasis zerlegbar (mithin  $B$ -glatt) sind, kommen hier die in Kapitel 5 verwendete Glattheitsfunktion  $\psi$  und der Satz von Canfield, Erdős und Pomerance ins Spiel. Sind alle gewonnenen Reste kleiner oder gleich  $x$ , so ist der Quotient  $\psi(x, B)/x$  praktisch gleichbedeutend mit der Chance, über der Faktorbasis zerlegbar zu sein.  $x/\psi(x, B)$  kann man dann als die Anzahl der notwendigen Versuche auffassen, um eine solche Relation zu finden. Nach den vorangegangenen Untersuchungen braucht man ungefähr  $|F|$  Relationen (also etwa  $\pi(B)$  viele), was zur Gesamtlaufzeit

$$L(x) = \ln \ln B \cdot \pi(B) \cdot x \cdot \psi(x, B)^{-1}$$

führt. Nach dem Primzahlsatz ist  $\pi(B) = \mathcal{O}(B/\ln B)$ , so daß man das Produkt  $\ln \ln B \cdot \pi(B)$  vereinfacht durch  $B$  nach oben abschätzen kann. Mit der Darstellung  $B = x^{1/u}$  ergibt sich demnach

$$L(x) = x^{1/u} \cdot x \cdot \psi(x, x^{1/u})^{-1}$$

Jetzt kann Satz 5.4.1 angewendet werden:

$$L(x) = x^{1/u} u^u$$

Da wir das optimale  $B$  finden wollen, ist das Minimum dieses Ausdrucks zu berechnen. Das führt wieder auf den Ansatz  $\frac{dL}{du} = 0$ , d. h.

$$\begin{aligned} x^{1/u} u^{u-2} (u^2(\ln u + 1) - \ln x) &= 0 \\ u^2(\ln u + 1) &= \ln x \end{aligned}$$

Da  $u^2$  der bestimmende Term der rechten Seite ist, gilt  $u \approx \sqrt{\ln x}$ , wir bekommen also

$$u^2(\ln \sqrt{\ln x} + 1) \approx \ln x$$

mit dem Ergebnis

$$u_{\text{opt}} = \mathcal{O} \left( \sqrt{\frac{2 \ln x}{\ln \ln x}} \right)$$

Das entspricht interessanterweise derselben Wahl wie bei der Methode der elliptischen Kurven (siehe Kapitel 5). Eingesetzt bedeutet das für die Gesamtlaufzeit

$$\begin{aligned} L(x) &= \exp\left(\frac{1}{u} \ln x + u \ln u\right) \\ &= \exp\left(\ln x \sqrt{\frac{\ln \ln x}{2 \ln x}} + \frac{1}{2} \ln \frac{2 \ln x}{\ln \ln x} \sqrt{\frac{2 \ln x}{\ln \ln x}}\right) \\ &\approx \exp\left(\sqrt{\frac{1}{2} \ln x \cdot \ln \ln x} + \frac{1}{2} \ln \ln x \sqrt{\frac{2 \ln x}{\ln \ln x}}\right) \\ &= \exp\left(\sqrt{2 \ln x \cdot \ln \ln x}\right) \end{aligned}$$

Wie im Abschnitt über den Siebschritt ausgeführt wurde, sind die Relationen durch  $N^{\frac{1}{2}+o(1)}$  beschränkt, so daß man mit

$$\begin{aligned} L(N) &= \exp\left(\sqrt{2 \ln(N^{\frac{1}{2}+o(1)}) \cdot \ln \ln(N^{\frac{1}{2}+o(1)})}\right) \\ &= \exp\left(\sqrt{(1+o(1)) \cdot \ln N \cdot \ln \ln N}\right) \end{aligned}$$

die Laufzeit in Abhängigkeit von der Eingabe  $N$  erhält. Die zugehörige Wahl von  $B$  ist dann

$$B = \exp\left(\left(\frac{1}{2} + o(1)\right) \sqrt{\ln N \cdot \ln \ln N}\right)$$

Im Auswahlschritt wird zu insgesamt  $l$  gesammelten Relationen das entsprechende Gleichungssystem gelöst. Der Gauß-Algorithmus besitzt eine Laufzeit von  $\mathcal{O}(l^3)$  für eine quadratische  $l \times l$ -Matrix. Wegen  $l \approx B$  wäre dann der Aufwand mit  $B^3$  deutlich größer als im Siebschritt.

Stattdessen benutzt man zwei andere Verfahren [Pom02], das Verfahren von Wiedemann und das von Montgomery [Mo95] entwickelte Block-Lanczos-Verfahren. Beide besitzen eine Laufzeit von  $B^{2+o(1)}$ , also im wesentlichen die des Siebschrittes. Letzteres arbeitet sehr gut mit dünn besetzten Matrizen und kann den benötigten Speicherplatz deutlich reduzieren. In der Praxis macht die Dauer des Auswahltrittes nur einen Bruchteil der Gesamtzeit aus.

## 6.4 Verbesserungen des Verfahrens

Eine wesentliche Eigenschaft des Quadratischen Siebes ist seine gute Parallelisierbarkeit. Das lineare Sieben von Zahlen der Form  $x + l \cdot p^s$  kann unter Verwendung von  $n$  Rechnern auf das Sieben aller Zahlen der Gestalt  $x + n \cdot l \cdot p^s$  (mit unterschiedlichen Initialwerten  $x$ ) verteilt werden. Wie schon bemerkt wurde spielt der (kaum verteilt lösbare) Auswahlritt eine untergeordnete Rolle. Um die Effizienz des Verfahrens noch weiter zu erhöhen, wurden im Laufe der Zeit mehrere Verfeinerungen entwickelt, die z. T. auch heuristischer Natur sind.

In der Praxis hat sich beispielsweise gezeigt, daß das Betrachten beliebiger Primpotenzen  $p^s$  im Siebschritt nur für kleine Primzahlen  $p$  lohnenswert ist, so daß man sich auf diese beschränkt. Das kann dazu führen, daß einige „gute“ Relationen nicht gefunden werden, was man aber dennoch in Kauf nimmt.

Eine andere Möglichkeit stellt das *logarithmische Sieb* dar. Es sei daran erinnert, daß die Siebtabelle zunächst mit den quadratischen Resten  $r_i$  aufgefüllt wird, die im Siebschritt bei einer Markierung durch den betrachteten Primteiler  $p_j$  dividiert werden. Einen zerlegbaren Rest erkennt man dann daran, daß er nach dem Siebschritt zu 1 reduziert wurde. Da Divisionen relativ aufwendig sind (und in diesem Fall sogar die Dauer des Siebschrittes wesentlich bestimmen), behilft man sich, indem man die Tabelle stattdessen mit den auf geringe, aber hinreichende Genauigkeit berechneten Logarithmen  $\ln r_i$  initialisiert. Das Markieren eines Primteilers  $p_j$  gestaltet sich dann in Form der Subtraktion von  $\ln p_j$ . Einen zerlegbaren Rest erkennt man dann dadurch, daß er nach dem Sieben zu 0 reduziert wurde. Eine aufwendige Division kann also durch eine schnelle Subtraktion ersetzt werden - allerdings auf Kosten der Exaktheit.

**Mehrfachpolynome und Selbstinitialisierung**

In der Grundversion benutzen wir ein festes Siebpolynom der Form  $Q(x) = x^2 - N$ . Dieses hat einen erkennbaren Nachteil: desto größer das Siebintervall  $I$  ist, desto größer geraten auch die von  $Q$  erzeugten quadratischen Reste. Es wird dann immer unwahrscheinlicher, daß selbige über der Faktorbasis zerlegbar sind. Als einen Ausweg stellten J. A. Davis und D. B. Holdridge 1983 das *Multiple Polynomial Quadratic Sieve* (MPQS) vor. In dieser Variante werden mehrere Siebpolynome der Form

$$Q(x) = (ax + b)^2 - N$$

betrachtet. Die Parameter  $a$  und  $b$  werden dabei so gewählt, daß  $b^2 - N$  durch  $a$  teilbar ist. Dann gilt nämlich

$$Q(x) = (ax + b)^2 - N = a^2x^2 + 2abx + b^2 - N = a \cdot (ax^2 + 2bx + c)$$

Mit einem solchen Polynom werden also Reste erzeugt, die immer durch  $a$  teilbar sind - und sich somit „glatter“ verhalten. Man kann also verschiedene Werte  $a$  wählen und bekommt mehrere Siebpolynome, für die jeweils der Siebschritt durchgeführt werden kann. Als Nachteil muß man allerdings in Kauf nehmen, daß ein großes  $a$  die Werte  $Q(x)$  auch schneller ansteigen läßt.

Um zu einem gegebenen  $a$  passende Werte für  $b, c$  zu finden, muß die Kongruenz

$$b^2 \equiv N \pmod{a}$$

gelöst werden. Für Primzahlen  $a$  aus der Faktorbasis existieren zwei Lösungen dieser Gleichung (da wir sichergestellt haben, daß  $\left(\frac{N}{a}\right) = 1$ ). Die Idee des *Self Initializing Quadratic Sieve* (SIQS) besteht nun darin,  $a$  als Produkt von Elementen der Faktorbasis zu wählen, da es dann mehrere Lösungen für  $b$  gibt und dadurch auch mehr Polynome zur Verfügung stehen. Mit dieser Idee konnte das Verfahren nochmals verbessert werden.

**Das Zahlkörpersieb**

Basierend auf einer Idee von J. M. Pollard wurde ab 1988 das *Special Number Field Sieve* (SNFS) für spezielle Zahlen der Form  $r^e \pm s$  für kleine positive  $r, s$  entwickelt [Len93]. Mit dieser Methode konnte erstmals die neunte Fermatzahl faktorisiert werden.

Das Verfahren geht von einem irreduziblen Polynom  $f(x)$  aus und benutzt als Grundlage den Ringhomomorphismus  $\varphi : \mathbb{Z}[\alpha] \rightarrow \mathbb{Z}/N\mathbb{Z}$ , wobei  $\alpha$  eine Nullstelle von  $f$  ist. Mit Hilfe von  $f, \varphi$  und einigen Eigenschaften algebraischer Zahlkörper ist es möglich, quadratische Reste zu erzeugen, die deutlich kleiner sind als beim klassischen Quadratischen Sieb. In einer späteren Version, dem *General Number Field Sieve* (GNFS) konnte dieses Verfahren auf beliebige natürliche Zahlen  $N$  erweitert werden. Mit einer heuristischen Laufzeit von

$$L(N) = \exp\left((c + o(1)) \sqrt[3]{\ln N \cdot (\ln \ln N)^2}\right)$$

(wobei  $c = \sqrt[3]{32/9}$  bei SNFS und  $c = \sqrt[3]{64/9}$  bei GNFS) ist es das momentan asymptotisch schnellste bekannte Faktorisierungsverfahren. Wie auch bei den meisten anderen neueren Verfahren konnte diese Laufzeit aber bis heute nicht exakt bewiesen werden (für eine ausführliche Analyse des Verfahrens siehe [SS02])

Der bislang größte Erfolg des Zahlkörpersiebes ist die Faktorisierung von RSA-200. F. Bahr, M. Boehm, J. Franke und T. Kleinjung von der Universität Bonn konnten im Mai 2005 (nach eineinhalb Jahren Rechenzeit) die vollständige Zerlegung in zwei 100stellige Primteiler angeben.

# Literaturverzeichnis

- [AKS02] Manindra Agrawal, Neeraj Kayal and Nitin Saxena: *PRIMES is in P*. 2002.
- [AB82] J. Arney and E. A. Bender: *Random mappings with constraints on coalescence and number of origins*, Pacific Journal of Math. 103, 269-294 (1982).
- [AtMo92] A.O.L. Atkin, F. Morain: *Finding Suitable Curves for The Elliptic Curve Method of Factorization*. Preprint (1992).
- [BCDH00] R. P. Brent, R. E. Crandall, K. Dilcher, C. van Halewyn: *Three New Factors of Fermat Numbers*. Preprint (2000).
- [Brent80] Richard P. Brent: *An Improved Monte Carlo Factorization Algorithm*. Nordisk Tidskrift for Informationsbehandling (BIT) 20, 176-184 (1980).
- [Brent99] Richard P. Brent: *Factorization of The Tenth Fermat Number*. Math. Comp. 68, 429-451 (1999).
- [CEP83] E. R. Canfield, P. Erdős, C. Pomerance: *On a Problem of Oppenheim concerning Factorisatio Numerorum*. Journal of Number Theory 17, 1-28 (1983).
- [Cun] The Cunningham Project: <http://homes.cerias.purdue.edu/~ssw/cun/>.
- [ECMNet] The ECMNet Project:  
<http://www.loria.fr/~zimmerma/records/ecmnet.html>.
- [FermS] Distributed Search for Fermat Number Divisors:  
<http://www.fermatsearch.org/>.
- [For96] Otto Forster: *Algorithmische Zahlentheorie*. Vieweg 1996.
- [GIMPS] Projekt GIMPS: <http://prime.haugk.co.uk/>.
- [HaRa17] G. H. Hardy, S. Ramanujan: *The normal number of prime factors of a number n*. Quart. J. Math. 48, 76-92 (1917).
- [Leh74] R. S. Lehman: *Factoring Large Integers*. Math. Comp. 28, 637-646 (1974).
- [Len87] Hendrik W. Lenstra: *Factoring integers with elliptic curves*. Annals of Mathematics 127, 649-673 (1987).
- [Len93] A. K. Lenstra, H. W. Lenstra: *The Development of the Number Field Sieve*. Springer-Verlag 1993.
- [List95] Klaus List: *Faktorisierungsalgorithmen und ihre Implementierung im Computeralgebrasystem DERIVE*. Technische Universität Wien 1995.
- [MersF] Mersenneforum: <http://www.mersenneforum.org/>.
- [Mo95] Peter L. Montgomery: *A block Lanczos algorithm for finding dependencies over GF(2)*. Lecture Notes in Computer Science 921, 106-120. Springer-Verlag 1995.
- [Mül95] Andreas Müller: *ECM. Eine FFT-Continuation für die elliptische Kurvenmethode*. 1995.
- [NFSNet] NFSNet: <http://www.nfsnet.org/>.

## Literaturverzeichnis

- [Pet88] Ivars Peterson: *Mathematische Expeditionen*. Spektrum Akademischer Verlag 1998.
- [Pol75] John M. Pollard: *A Monte Carlo Method for Factorization*. Nordisk Tidskrift for Informationsbehandling (BIT) 15, 331-334 (1975).
- [Pom85] Carl Pomerance: *The Quadratic Sieve Factoring Algorithm*. Lecture Notes in Computer Science 209, 169-182 (1985).
- [Pom96] Carl Pomerance: *A Tale of Two Sieves*. Notices of the Amer. Math. Soc. 43, 1473-1485 (1996).
- [Pom02] Carl Pomerance: *Smooth numbers and the quadratic sieve*. Preprint (2001).
- [RSA] RSA Factoring Challenge:  
<http://www.rsasecurity.com/rsalabs/node.asp?id=2093>.
- [Shor94] P. W. Shor: *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. Proceeding of the 35th International Symposium on the Foundations of Computer Science, Santa Fe, IEEE Computer Society Press, 124-134 (1994).
- [SS02] Katja Schmidt-Samoa: *Das Number Field Sieve. Entwicklung, Varianten und Erfolge*. Universität Kaiserslautern 2002.
- [Wag] Samuel Wagstaff: <http://homes.cerias.purdue.edu/~ssw/cun/want97>.
- [Wiki] Wikipedia-Artikel: <http://de.wikipedia.org/wiki/Faktorisierungsverfahren>.
- [Will82] H. C. Williams: *A  $p+1$  Method of Factoring*. Math. Comp. 39, 225-234 (1982).