Zulassungsarbeit zur wissenschaftlichen Prüfung
für das Lehramt an Gymnasien in Bayern

im Fach Mathematik

über das Thema

Primzahltests
und
Faktorisierungsmethoden

Gestellt von
Prof. Dr. Ernst Kunz

an der Universität Regensburg,

vorgelegt von
Stefan Friedl

Inhaltsverzeichnis

Einleitung

> There is one comforting conclusion which is easy for a real mathematician. Real mathematics has no effects on war. No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems very unlikely that anyone will do so for many years.[1]
> G. H. Hardy: A mathematician's apology, 1940

Der berühmte Zahlentheoretiker Hardy würde diese Zeilen wohl nicht geschrieben haben, hätte er gewußt, daß die meisten jungen Mathematiker aus Oxford und Cambridge, allen voran Alan Turing, in eben diesem Jahr in die Armee einberufen wurden, um den Code der deutschen Chiffriermaschine „Enigma“ zu brechen. Daß ihnen das gelang, sollte sich als einer der kriegsentscheidenden Faktoren herausstellen (vgl. [HS]). Seitdem sind weite Teiler der Zahlentheorie untrennbar mit der Kodierungstheorie verbunden.
Das derzeit wichtigste und weitestverbreitete Verschlüsselungsverfahren ist der vor 20 Jahren entwickelte RSA–Algorithmus. Vor allem im sich stürmisch entwickelnden Datenaustausch über das Internet sind der RSA–Algorithmus bzw. darauf aufbauende Programme wie „Pretty good Privacy“ im Moment die Standardverfahren. Der RSA–Algorithmus verdankt seine Praktikabilität, der Tatsache, daß es Verfahren gibt, mit denen man schnell Primzahlen der Größenordnung 10^{100} finden kann, und er verdankt seine Sicherheit der Unfähigkeit, mit herkömmlichen Faktorisierungsalgorithmen Zahlen der Größenordnung 10^{200} in brauchbarer Zeit zu faktorisieren. Auf der Suche nach schnelleren Faktorisierungsalgorithmen werden Forschungsergebnisse aus Teilbereichen der Mathematik herangezogen, welche auf den ersten Blick keinerlei Zusammenhang mit dem Faktorisierungsproblem aufweisen. Ein Beispiel dafür ist die Theorie der elliptischen Kurven, welcher das Kapitel 5 gewidmet ist.

Ich habe mit dieser Arbeit versucht, eine gründliche Einführung in das Thema „Primzahltests und Faktorisierungsmethoden“ zu geben, welche zwar für einen Leser mit Grundkenntnissen in linearer Algebra verständlich ist, aber dennoch die schwierigen Punkte nicht verschweigt. Da es sich um ein Thema

[1] Deutsche Übersetzung: Es gibt eine tröstliche Schlußfolgerung, welche für einen echten Mathematiker leicht zu treffen ist. Echte Mathematik hat keine Auswirkungen auf Kriege. Bis jetzt hat noch niemand einen kriegerischen Zweck gefunden, dem die Zahlentheorie oder die Relativitätstheorie dienlich sein könnte, und es scheint sehr unwahrscheinlich, daß sich das in absehbarer Zeit ändern wird.

handelt, welches aus der Praxis motiviert ist, und z.B. auch für Schüler von Interesse sein kann, beginnt die Arbeit sofort mit der Schilderung des RSA–Verschlüsselungsalgorithmus, obwohl die (geringen) dafür nötigen mathematischen Kenntnisse erst im darauffolgenden Kapitel 2 vermittelt werden. Allerdings handelt es sich dabei um einfache Tatsachen aus der elementaren Zahlentheorie, welche dem Leser wahrscheinlich schon bekannt sind.
Der RSA–Algorithmus wirft die Frage auf, wie schnell Primzahltests und Faktorisierungsalgorithmen arbeiten. Dieser Frage wird in den folgenden Kapiteln nachgegangen. In Kapitel 2 werden zwei Primzahltests, nämlich der Fermat–Test und der Miller–Rabin–Test, vorgestellt, die beide auf dem kleinen Fermatschen Satz beruhen. Beide arbeiten außergewöhnlich schnell, geben aber keinen Beweis für die Primalität einer Zahl; eine Zahl, die einen solchen Test besteht, ist nur höchstwahrscheinlich eine Primzahl. Für Primzahlen mit einer gewissen Zusatzeigenschaft kann die Primalität mit dem Satz von Pocklington (Satz 2.5) bewiesen werden. Aufbauend auf diesen Satz habe ich einen viel allgemeineren Satz formuliert, welcher die Primalität einer Zahl beweist. Als Korollar davon wird, nachdem die nötigen Kenntnisse über elliptische Kurven eingeführt wurden, in Kapitel 6.2 ein allgemeiner Primzahltest eingeführt.
Zuerst wird jedoch in Kapitel 3 eine Schreibweise vorgestellt, welche es ermöglicht, die Geschwindigkeiten der in dieser Arbeit behandelten Verfahren abzuschätzen. Mit dieser Methode kann gezeigt werden, daß die in Kapitel 4 erläuterten allgemeinen Faktorisierungsalgorithmen, nämlich der Rho–Algorithmus und der Faktor–Basis–Algorithmus, für große Zahlen deutlich langsamer sind als die Primzahltests. Im Anschluß an die Schilderung des Faktor–Basis–Algorithmus wird noch ein kurzer Überblick über die neuesten Faktorisierungalgorithmen und die jüngsten Faktorisierungserfolge gegeben. Insbesondere wird die bisher größte, im September 1997 erfolgte, nichttriviale Faktorisierung angegeben. Den Abschluß des Kapitels 4 bildet der sogenannte $(p-1)$–Faktorisierungsalgorithmus. Dieser ist, analog dem Pocklington–Primzahltest, zwar nur für Zahlen anwendbar, die eine Zusatzbedingung erfüllen, stellt aber einen Ausgangspunkt für weitere Algorithmen dar. Um dies zu zeigen, habe ich einen allgemeinen Faktorisierungssatz formuliert, als dessen Korollar sich verschiedene Faktorisierungsalgorithmen, insbesondere der Lenstra–Algorithmus für elliptische Kurven, ergeben.
Für das Verständnis des Lenstra–Algorithmus, welcher von den in dieser Arbeit vorgestellten Faktorisierungsalgorithmen der modernste und asymptotisch schnellste ist, werden einige Grundlagen aus der Theorie der elliptischen Kurven benötigt. Diese werden in Kapitel 5 bereitgestellt. Die elliptischen Kurven zeichnen sich unter den algebraischen Kurven dadurch aus, daß sie eine Gruppenstruktur besitzen. Dies läßt sich bis auf das Assoziativgesetz

leicht beweisen. Es gibt zwar eine ganze Reihe eleganter Beweise der Assoziativität, diese erfordern aber mehr theoretische Grundlagen als in dieser Arbeit eingeführt werden können. Eine Aufgabenstellung dieser Arbeit war es deshalb, einen rein rechnerischen Beweis zu finden, d.h. die Assoziativität mittels in Kapitel 5 hergeleiteter Additionsformeln zu beweisen. Für die dabei notwendigen langwierigen Berechnungen, welche mit der Hand nicht mehr möglich gewesen wären, habe ich das Computerprogramms „Cocoa“ verwendet. Der Beweis ist wegen seines großen Umfangs aus Kapitel 5 ausgegliedert, ihm wurde das Kapitel 8 gewidmet.
Mit Hilfe der eingeführten Grundtatsachen über elliptische Kurven können jetzt drei Verfahren verstanden werden, welche in Kapitel 6.1 vorgestellt werden. Beim Gordon–Verfahren handelt es sich um einen probabilistischen Primzahltests mit interessanten Eigenschaften, welcher abgesehen von einer kurzen Andeutung in [Ribenboim] in keinem der Lehrbücher erwähnt wird. Bei den anderen beiden Verfahren handelt es sich um den Lenstra–Algorithmus und den Goldwasser–Kilian–Primzahltest; beide sind jeweils mit einem klassischen Algorithmus verwandt, nämlich dem $(p-1)$–Algorithmus bzw. dem Pocklington–Primzahltest, und jeweils ein Spezialfall aus dem von mir formulierten allgemeinen Primzahltest– bzw. Faktorisierungsatz. Der Goldwasser–Kilian–Primzahltest ist in dieser Arbeit der einzige Algorithmus, welcher die Primalität einer beliebigen auch wirklich Primzahl beweisen kann.
Eine weitere Aufgabenstellung war es, die verschiedenen in dieser Arbeit eingeführten Verfahren zu programmieren. Ich habe dafür das Programm „Enigma“ geschrieben, eine kurze Einführung in dieses Programm wird in Kapitel 7.1 gegeben. Die Bedienung des Programms ist so einfach, daß es auch gut im Unterricht an Gymnasien verwendet werden kann, als besonderer Ansporn für Schüler ist auch der RSA–Algorithmus einprogrammiert. Mit Hilfe von „Enigma“ werden in den Kapiteln 7.2 und 7.3 die Geschwindigkeiten der vier verschiedenen in dieser Arbeit vorgestellten Faktorisierungsalgorithmen verglichen.

Die beiden letzten Kapitel dieser Arbeit stammen bis auf zwei zitierte Sätze und eine Graphik vollständig von mir. Die Kapitel 2 bis 6 sind zumeist der Literatur entnommen. Die wichtigsten Bücher waren dabei:

1. Einführungen in das Thema: [Bressoud], [Forster], [Koblitz] und [Riesel],
2. Grundlagen der elliptischen Kurven: [Kunz95] und [Silverman],
3. Grundlagen der Algebra: [Kunz94].

Sind Stellen der Arbeit in diesen Kapiteln einem dieser Bücher entnommen, dann erfolgt keine Literaturangabe. Einige Beweise stammen von mir, ins-

besondere sind alle Sätze aus Kapitel 5 mit Ausnahme von Satz 5.3 von mir bewiesen, ebenso stammt der zweite Teil des Beweises zu Satz 2.15 von mir. In den vorherigen Kapiteln wurden die Sätze 2.13, 2.20, 4.1, 4.4, 4.5 von mir formuliert und bewiesen. Allerdings sind darunter wohl mehrere „Wiederentdeckungen“. Die Abbildungen wurden bis auf Abbildung 6 von mir entworfen.

Zum Schluß möchte ich mich bei all jenen bedanken, die zum Gelingen dieser Arbeit beigetragen haben. Wertvolle Dienste leisteten mir dabei Matthias Aschenbrenner, Josef Hiebl, Dr. Martin Kreuzer, Ben Kriechel, Armin Röhrl sowie meine Eltern und mein Bruder Martin. Mein besonderer Dank gilt Herrn Professor Dr. Ernst Kunz für die glückliche Wahl des Themas und die sehr gute Betreuung.

Notation

In der Arbeit werden folgende, zumeist geläufige, Notationen verwendet:

1. Die Mächtigkeit einer endlichen Menge M wird mit $\sharp(M)$ oder kurz $\sharp M$ bezeichnet. Für eine endliche Gruppe G gilt dann $\sharp(G) = \mathrm{ord}(G)$.

2. Für $x \in \mathbb{R}$ ist $[r] := \max\{n \in \mathbb{Z} \mid n \leq r\}$ definiert.

3. Ist R ein Ring, so bezeichnet R^* die Einheitengruppe des Rings.

4. Sei $n \in \mathbb{N}$. Dann bezeichnet π_n die Abbildung

$$\begin{array}{rcl} \pi_n : \mathbb{Z} & \to & \mathbb{Z}/n\mathbb{Z} \\ a & \mapsto & a + n\mathbb{Z} \end{array}$$

   $\pi_n(a)$ wird, wenn keine Verwechslungsgefahr besteht, auch als $\overline{a}$ geschrieben. Außerdem bezeichnet η_n die Abbildung

$$\begin{array}{rcl} \eta_n : \mathbb{Z}/n\mathbb{Z} & \to & \mathbb{N}_0 \\ \overline{a} = a + n\mathbb{Z} & \mapsto & \min\{b \in a + n\mathbb{Z} \mid b \geq 0\} \end{array}$$

   $\eta_n(\overline{a})$ wird, wenn keine Verwechslungsgefahr besteht, auch als a geschrieben.

5. Restklassen werden immer mit einem Überstrich gekennzeichnet (z.B. $\overline{a}$), ausgenommen davon sind nur $\ldots, -2, -1, 0, 1, 2, \ldots \in \mathbb{Z}/n\mathbb{Z}$.

6. Für $a \in \mathbb{Z}$ ist

$$a \bmod n := \min\{b \in a + n\mathbb{Z} \mid b \geq 0\}$$

7. Besitzt n einen Primteiler p, dann bezeichnet π_{np} die Abbildung

$$\begin{array}{rccc} \pi_{np} : & \mathbb{Z}/n\mathbb{Z} & \to & \mathfrak{p} \\ & x + (n) & \mapsto & x + (p) \end{array}$$

   Für $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$ wird $\pi_{np}(\overline{x})$ auch als $\overline{x}_p$ geschrieben.

8. Ist K ein Körper, so bezeichnet $\mathbb{P}^2(K)$ den zwei dimensionalen projektiven Raum über K.

1 RSA–Verschlüsselungsalgorithmus

Die im Moment vielleicht wichtigste Anwendung für Primzahltests und Faktorisierungsalgorithmen ist der RSA–Verschlüsselungsalgorithmus. Die Sicherheit dieses Algorithmus beruht darauf, daß es zwar mit modernen Computern möglich ist, innerhalb weniger Minuten eine 100–stellige Primzahl zu finden, es aber mit den heutigen Methoden und Computern nicht möglich ist, ein Produkt zweier Primfaktoren der Größenordnung 10^{100} in brauchbarer Zeit zu faktorisieren.

1.1 Allgemeines zur Kodierungstheorie

Um Texte mit mathematischen Methoden zu verschlüsseln, muß man sie zuerst in mathematische Objekte, wie z.B. Zahlen, verwandeln. Dazu wählt man eine Menge M von Buchstaben und Zeichen, beispielsweise könnte man $M = \{A, \ldots, Z\}$ wählen. Mit $n := \sharp(M)$ kann man den Zeichen mittels einer beliebig gewählten bijektiven Abbildung

$$f : M \to \{0, \ldots, n-1\}$$

die Zahlen 0 bis $n-1$ zuordnen. Ein Text bestehend aus s Zeichen wird mittels folgender Abbildung eindeutig durch eine Zahl zwischen 0 und $n^s - 1$ beschrieben:

$$\begin{aligned} f_s : \underbrace{M \times \ldots \times M}_{s-mal} &\to \{0, \ldots, n^s - 1\} \\ (b_1, \ldots, b_s) &\mapsto f(b_1) + nf(b_2) + \ldots + n^{s-1} f(b_s) \end{aligned}$$

Einen beliebigen Text teilt man in Blöcke von s Zeichen auf, die man in Zahlen verwandelt, welche dann verschlüsselt werden. Ist $N \geq n^s - 1$, so können die Zahlen $0, \ldots, n^s - 1$ mittels der Abbildung π_N mit den dazugehörigen Restklassen in $\mathbb{Z}/N\mathbb{Z}$ identifiziert werden. Ein Code ist definitionsgemäß eine bijektive Abbildung $h : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$. Oft ist es nützlich $N > n^s - 1$ zu wählen, da an N gewisse Zusatzforderungen gestellt sein können. Beispielsweise wird beim RSA–Algorithmus verlangt, daß N das Produkt zweier großer Primzahlen ist, was bei $n^s - 1$ i.a. nicht der Fall ist. Mittels η_N werden den Restklassen in $\mathbb{Z}/N\mathbb{Z}$ wieder die Zahlen $0, \ldots, N-1$ zugeordnet. Ist $l \in \mathbb{N}$ so gewählt, daß $n^l \geq N$, so ist die Abbildung f_l^{-1}, welche die Zahlen wieder in Zeichen umwandelt, injektiv. Die gesamte Kodierungsabbildung c eines Textes auf einen anderen Text sieht dann folgendermaßen aus:

$$\begin{aligned} c : M^s \xrightarrow{f_s} \{0, \ldots, n^s - 1\} \hookrightarrow \{0, \ldots, N\} \xrightarrow{\pi_N} \mathbb{Z}/N\mathbb{Z} \xrightarrow{h} \mathbb{Z}/N\mathbb{Z} \\ \xrightarrow{\eta_N} \{0, \ldots, N-1\} \hookrightarrow \{0, \ldots, n^l - 1\} \xrightarrow{f_l^{-1}} M^l \end{aligned}$$

Da die Abbildung injektiv ist, existiert die Umkehrabbildung

$$c^{-1} : c(M^s) \to M^s$$

welche zugleich die Dechiffrierabbildung ist.

1.2 RSA–Verschlüsselung

Eine klassische Verschlüsselungsabbildung ist z.B. durch $h(\bar{t}) = \overline{a}\bar{t} + \bar{b}$ mit $\overline{a} \in (\mathbb{Z}/N\mathbb{Z})^*$ gegeben.[1] Bei einem solchen Kodierungsverfahren muß die Chiffrierabbildung geheimgehalten werden, da mittels dieser die Dechiffrierabbildung $h^{-1}(\bar{t}) = (\bar{t} - \bar{b})\overline{a}^{-1}$ leicht zu bestimmen ist. Im Jahre 1976 schlugen W. Diffie und M. Hellman ein neues Chiffrierungsverfahren vor, nämlich das sogenannte „Public Key"–Verfahren (vgl. [DH]). Man wählt eine Chiffrierabbildung h, von der ohne zusätzliches Wissen nur schwer das Inverse h^{-1} zu bestimmen ist, so daß h ohne Risiko veröffentlicht werden kann. Das bekannteste „Public Key"–Verfahren ist der von Rivest, Shamir und Adleman 1977 entwickelte RSA–Algorithmus (vgl. [RSA]). Ein Benutzer A dieses Kodierungsverfahrens wählt zwei sehr große, verschiedene Primzahlen[2] p_A und q_A (d.h. Primzahlen der Größenordnung 10^{100}), und berechnet $N_A := p_A q_A$.[3] Da A die Faktorisierung von N_A weiß, kann er $\varphi(N_A) = (p_A - 1)(q_A - 1) = N_A + 1 - p_A - q_A$ bestimmen. Als letztes hat A eine Zahl $e_A \in \{2, \ldots, \varphi(N_A)\}$ mit $\mathrm{ggT}(e_A, \varphi(N_A)) = 1$ zu wählen. Wegen dieser Eigenschaft kann A ein $d_A \in \mathbb{N}$ mit $d_A e_A \equiv 1 \bmod \varphi(N_A)$ bestimmen. Die RSA–Verschlüsselungsabbildung h_A lautet dann:

$$\begin{array}{rccc} h_A : & \mathbb{Z}/N_A\mathbb{Z} & \to & \mathbb{Z}/N_A\mathbb{Z} \\ & \bar{t} & \mapsto & \bar{t}^{\,e_A} \end{array}$$

Der Anwender A veröffentlicht die Kodierungsabbildung $h_A(\bar{t}) = \bar{t}^{\,e_A}$. Will jetzt B an A einen Text T, bzw. die entsprechende Restklasse $\bar{t}$ übermitteln, so berechnet er $h_A(\bar{t}) = \bar{t}^{\,e_A}$ und sendet das Ergebnis an A. Nur der Empfänger A besitzt die Umkehrabbildung $h_A^{-1}(\bar{t}) = \bar{t}^{\,d_A}$ und kann damit als einziger den verschlüsselten Text decodieren:[4]

$$h_A^{-1}(h_A(\bar{t})) = (h_A(\bar{t}))^{\,d_A} = \bar{t}^{\,e_A d_A} = \bar{t}$$

[1] Es wird angenommen, daß dieses Verfahren das erste Mal von Cäsar benützt wurde. Er wählte dabei $\overline{a} = 1$ und $\bar{b} = 3$. Allerdings hat Cäsar den Algorithmus vermutlich anders beschrieben.

[2] Mit modernen Computern benötigt man dafür nur wenige Minuten (vgl. Kapitel 2.5).

[3] Eine Zahl die Produkt zweier etwa gleich großer Primzahlen ist, wird in der Arbeit auch kurz als RSA–Zahl bezeichnet.

[4] Die letzte Gleichheit folgt dabei aus Satz 2.8.

Wie man sieht, kann die Umkehrfunktion nur bestimmt werden, wenn man $\varphi(N_A)$ kennt, was gleichbedeutend ist mit dem Wissen um die Primfaktorzerlegung von N_A. Diese aber weiß nur A, und da es mit den heute zur Verfügung stehenden Faktorisierungsalgorithmen und Computern nicht möglich ist ein Produkt aus zwei Primzahlen der Größenordnung 10^{100}, sprich N_A, in brauchbarer Zeit zu faktorisieren, ist dieses Verfahren sehr sicher.

Das RSA–Verfahren hat zudem den großen Vorteil, daß man die Möglichkeit hat, eine nicht fälschbare „Unterschrift“ zu verwenden. Hat nämlich B ebenfalls eine Kodierungsabbildung h_B gewählt und diese veröffentlicht, so kann B an A einen Text, nämlich die Unterschrift, senden, der durch $h_B^{-1}(\bar{t}) = \bar{t}^{\,d_B}$ verschlüsselt ist. A kann den Text mit Hilfe der bekannten Funktion $h_B(\bar{t}) = \bar{t}^{\,e_B}$ entschlüsseln, denn h_B und h_B^{-1} sind zueinander inverse Abbildungen, die beide als Chiffrier– bzw. Dechiffrierabbildung geeignet sind. A kann also sicher sein, daß der Text von B stammt, denn nur dieser verfügt über die Abbildung h_B^{-1}.

2 Klassische Primzahltests

Bevor jetzt verschiedene Verfahren eingeführt werden, sollen noch einige Begriffe geklärt werden. Ein Verfahren wird als Algorithmus bezeichnet, wenn das Ergebnis, welches das Verfahren liefert, sicher richtig ist, und nicht nur mit einer sehr hohen Wahrscheinlichkeit. Ein Algorithmus heißt probabilistisch, wenn für die Durchführung Zufallszahlen verwendet werden und die Bearbeitungszeit von der Art der Zufallszahlen abhängt. Ein Algorithmus der ohne Zufallszahlen auskommt, wird deterministisch genannt.

Ein Beispiel für einen deterministischen Algorithmus ist folgender Primzahltest:[1]

Algorithmus 2.1 (Probedivisionen–Algorithmus)
Sei n eine auf Primalität zu überprüfende Zahl.

1. Setze $i := 2$.
2. Gilt $i \mid n$, dann handelt es sich bei n um eine zusammengesetzte Zahl. Breche den Algorithmus ab.
3. Ist $i < [\sqrt{n}]$, so setze $i := i + 1$ und gehe zu 2. Andernfalls handelt es sich bei n um eine Primzahl.

Allerdings ist dieser Algorithmus viel zu zeitaufwendig, um für große Zahlen von praktischen Nutzen zu sein.[2] Für praktische Zwecke, wie z.B. den RSA–Algorithmus, verwendet man im Normalfalle Primzahltest–Verfahren, welche zwar nicht 100% Sicherheit bieten, dafür aber außergewöhnlich schnell sind. Für die Beschreibung dieser Verfahren werden einige Hilfsmittel aus der elementaren Zahlentheorie benötigt.

2.1 Der kleine Fermat

Die Sätze die in diesem Kapitel 2.1 vorgestellten werden, stellen die Grundwerkzeuge der elementaren Zahlentheorie dar. Sie werden im folgenden so häufig benötigt, daß sie ohne Verweise zitiert werden.

[1] Der Probedivisionen–Algorithmus kann auch als Faktorisierungsalgorithmus verwendet werden.

[2] Der Algorithmus kann etwas beschleunigt werden, wenn man nur für Primzahlen überprüft ob sie n teilen. Allerdings muß man dafür über eine Liste aller Primzahlen im Bereich $\{2, \ldots, [\sqrt{n}]\}$ verfügen. Für große Zahlen ist dies wiederum inpraktikabel.

Satz 2.1 (Division mit Rest)
Seien $s \in \mathbb{N}_0$ und $t \in \mathbb{N}$, dann gibt es eindeutig bestimmte $f, r \in \mathbb{N}_0$ mit $r < t$, so daß $s = tf + r$.

Satz 2.2
Seien $p, q \in \mathbb{N}$ und sei $d := \mathrm{ggT}(p, q)$, dann gibt es $e, f \in \mathbb{Z}$ mit

$$pe + qf = d$$

Der Satz folgt aus dem folgenden Algorithmus zur Bestimmung von e, f und $d = \mathrm{ggT}(p, q)$ für vorgegebene $p, q \in \mathbb{N}$ (vgl. [BRK, S. 51]):

Algorithmus 2.2 (Euklidischer Algorithmus)
Seien $p, q \in \mathbb{N}$.

1. Setze $r_0 := p$, $r_1 := q$ und $i := 0$.
2. Setze $e_0 := 1$, $e_1 := 0$, $f_0 := 0$ und $f_1 := 1$.
3. Setze $s_{i+2} := \left[\frac{r_i}{r_{i+1}}\right]$ und $r_{i+2} := r_i - r_{i+1}s_{i+2}$.
4. Setze $e_{i+2} := e_i - e_{i+1}s_{i+2}$ und $f_{i+2} := f_i - f_{i+1}s_{i+2}$.
5. Ist $r_{i+2} > 0$, so setze $i := i + 1$ und gehe zu 3.
6. Es ist[1] $d := r_{i+1} = \mathrm{ggT}(r_0, r_1) = \mathrm{ggT}(p, q)$, zudem ist[2]

$$d = r_{i+1} = pe_{i+1} + qf_{i+1}$$

Satz 2.3
Sei $n \in \mathbb{N}$ und sei $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$, dann gilt

$$\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \Leftrightarrow \mathrm{ggT}(a, n) = 1$$

Dieser Satz ist Grundlage für viele Faktorisierungsalgorithmen. Ist nämlich $n \in \mathbb{N}$ und hat man ein $\overline{a} \notin (\mathbb{Z}/n\mathbb{Z})^* \cup \{0\}$ gefunden, so ist $d := \mathrm{ggT}(a, n)$ ein nichttrivialer Teiler von n.

[1] Es gilt $r_{i+1} \mid r_i$, damit aber auch $r_{i+1} \mid r_{i-1}, \ldots, r_{i+1} \mid r_1, r_i \mid r_0$. Außerdem gilt für jeden Teiler t von r_0 und r_1, daß $t \mid r_0, t \mid r_1, \ldots, t \mid r_{i+1}$. D.h. also, daß $r_{i+1} = \mathrm{ggT}(r_0, r_1)$.

[2] Es ist wie man leicht nachprüfen kann, $r_j = pe_j + qf_j$ für $j = 0, 1$. Induktiv kann man zeigen, daß diese Gleichung für $j = 0, \ldots, i-1$ gilt, denn

$$\begin{aligned} r_{j+2} &= r_j - r_{j+1}s_{j+2} = pe_j + qf_j - (pe_{j+1} + qf_{j+1})s_{j+2} = \\ &= p(e_j - e_{j+1}s_{j+2}) + q(f_j - f_{j+1}s_{j+2}) = pe_{j+2} + qf_{j+2} \end{aligned}$$

Satz 2.4 (Chinesischer Restsatz)
Sei $n \in \mathbb{N}$ mit $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$, wobei $p_1, \ldots, p_l$ paarweise verschiedene Primzahlen sind, dann gibt es einen Ringisomorphismus

$$\varphi : \mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \ldots \times \mathbb{Z}/p_l^{e_l}\mathbb{Z}$$

Dabei ist

$$\begin{aligned} \varphi(1) &= (1, \ldots, 1) \\ \varphi(-1) &= (-1, \ldots, -1) \end{aligned}$$

Die einfachsten und schnellsten alternativen Primzahltests beruhen auf dem sogenannten „kleinen Fermat“:

Satz 2.5 (Kleiner Fermat)
Sei G eine endliche abelsche Gruppe, $a \in G$, dann gilt

$$a^{\mathrm{ord}(G)} = e$$

Der Beweis ist zwar bekannt, soll aber wegen der Schönheit des Arguments noch einmal aufgeführt werden:

Beweis:
Seien $g_1, \ldots, g_n$ die Elemente von G. Dann gilt für $i, j \in \{1, \ldots, n\}$ mit $i \neq j$, daß $g_i a \neq g_j a$, und daher ist $ag_1, \ldots, ag_n$ nur eine Umordnung von $g_1, \ldots, g_n$. Man erhält

$$a^n g_1 \ldots g_n = (ag_1) \ldots (ag_n) = g_1 \ldots g_n$$

Daraus folgt, daß $a^n = e$. □

Der kleine Fermat gilt auch für nichtabelsche Gruppen, allerdings ist der Beweis dafür etwas komplizierter, er kann in [Forster, S. 55] nachgelesen werden.

Definition:
Die Eulerfunktion φ ist folgendermaßen definiert:

$$\begin{aligned} \varphi : \mathbb{N} &\to \mathbb{N} \\ n &\mapsto \varphi(n) := \sharp\{b \in \{1, \ldots, n\} \mid \mathrm{ggT}(b, n) = 1\} \end{aligned}$$

Mit Hilfe des chinesischen Restsatzes kann man den folgenden Satz beweisen, welcher die Berechnung von $\varphi(n)$ für ein beliebiges $n \in \mathbb{N} \setminus \{1\}$ ermöglicht (vgl. [Kunz94, S. 78]).

Satz 2.6
Sei $n \in \mathbb{N} \setminus \{1\}$ mit $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$, wobei $p_1, \ldots, p_l$ paarweise verschiedene Primzahlen sind, dann gilt:

$$\varphi(n) = (p_1 - 1)p_1^{e_1 - 1} \cdot \ldots \cdot (p_l - 1)p_l^{e_l - 1}$$

Als Korollar aus dem kleinen Fermat (Satz 2.5) erhält man den folgenden Satz:
Satz 2.7
Sei $n \in \mathbb{N}$ und sei $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^*$. Dann gilt

1. $\varphi(n) = \mathrm{ord}((\mathbb{Z}/n\mathbb{Z})^*)$,
2. $\overline{a}^{\varphi(n)} = 1$,
3. $a^{\varphi(n)} = 1 \bmod n$. Insbesondere gilt für alle $e \in \mathbb{N}$ mit $e \equiv 1 \bmod \varphi(n)$, daß $a^e = a \bmod n$.

Man kann Satz 2.7 noch etwas schärfer formulieren:
Satz 2.8
Seien p, q verschiedene Primzahlen und sei $n := pq$, sowie $k \equiv 1 \bmod \varphi(n)$. Dann gilt für alle $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$, daß $\overline{a}^k = \overline{a}$.

Beweis:
Für $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^*$ gilt die Aussage nach Satz 2.7, für $\overline{a} = 0$ ist die Aussage klar. Sei also $\overline{a} \notin (\mathbb{Z}/n\mathbb{Z})^* \cup \{0\}$. Es ist $k = l\varphi(n) + 1 = l(p-1)(q-1) + 1$ mit einem $l \in \mathbb{Z}$. Dann gilt wegen $a \not\equiv 0 \bmod p$, daß $a^{p-1} \equiv 1 \bmod p$, also

$$a^k \equiv a(a^{p-1})^{l(q-1)} \equiv a1^{l(q-1)} \equiv a \bmod p$$

Analog ist $a^k \equiv a \bmod q$. Es ist also $a^k - a = s_p p = s_q q$ mit $s_p, s_q \in \mathbb{Z}$. Da p, q verschiedene Primzahlen sind, gibt es ein $s_n \in \mathbb{Z}$ mit $a^k - a = s_n n$. □

2.2 Anwendung des kleinen Fermat

Der kleine Fermat gibt eine notwendige Bedingung für die Primalität einer Zahl, denn jede Primzahl p erfüllt die Bedingung, daß $a^{p-1} \equiv 1 \bmod p$ für jedes $a \not\equiv 0 \bmod p$. Allerdings ist sie nicht hinreichend. Beispielsweise ist $3^{90} \equiv 1 \bmod 91$, obwohl $91 = 7 \cdot 13$ keine Primzahl ist. Dies motiviert die folgende Definition:

Definition:

Sei $n \in \mathbb{N}$. Eine Zahl $a \in \{2, \ldots, n-2\}$ heißt Basis zu n, falls $\mathrm{ggT}(a, n) = 1$. Die Menge aller Basen zu n wird mit B_n bezeichnet. Ist n zusammengesetzt und gibt es eine Basis $a \in B_n$ mit

$$a^{n-1} \equiv 1 \bmod n$$

so nennt man n Pseudoprimzahl zur Basis a.

Es stellt sich die Frage, wieviele Pseudoprimzahlen es gibt. Für den Wertebereich bis 10^{10} kann die Zahl der Pseudoprimzahlen zur Basis 2 in Tabelle 1 nachgelesen werden. Für größere Zahlen kann dies allgemein abgeschätzt werden. Für $n \in \mathbb{N}$ bezeichne

$$\begin{aligned} PP_2(n) &:= \sharp\{l \in \{2, \ldots, n\} \mid 2 \in B_l \text{ und } l \text{ ist Pseudoprimzahl zur Basis 2}\} \\ P(n) &:= \sharp\{l \in \{2, \ldots, n\} \mid l \text{ ist Primzahl}\} \end{aligned}$$

Nach [Pomerance81] ist

$$PP_2(n) \leq ne^{-\frac{\ln n \ln\ln\ln n}{2 \ln\ln n}}$$

Eine Abschätzung für $P(n)$ liefert der vielleicht wichtigste Satz der analytischen Zahlentheorie (vgl. [Davenport, S.113]):

Satz 2.9 (Primzahlsatz)
Es gilt

$$\lim_{n\to\infty} \frac{P(n)}{\frac{n}{\ln n}} = 1$$

Bezeichnet $o(1)$ einen Term $f(n)$ mit $\lim_{n\to\infty} f(n) = 0$, dann ist

$$P(n) = (1 + o(1))\frac{n}{\ln n}$$

Wählt man eine beliebige Zahl $n \in \mathbb{N}$ und ist $2^{n-1} \equiv 1 \bmod n$, so gilt für die Wahrscheinlichkeit $W(n)$, daß n nicht prim ist, daß

$$W(n) = \frac{PP_2(n)}{PP_2(n) + P(n)}$$

Man erhält die folgende Tabelle:

n	$P(n)$	$PP_2(n)$	$W(n)$
10^5	$\approx 8.7 \cdot 10^3$	$\approx 1.2 \cdot 10^4$	$\approx 5.8 \cdot 10^{-1}$
10^{10}	$\approx 4.3 \cdot 10^8$	$\approx 1.5 \cdot 10^8$	$\approx 2.6 \cdot 10^{-1}$
10^{20}	$\approx 2.2 \cdot 10^{18}$	$\approx 3.1 \cdot 10^{16}$	$\approx 1.4 \cdot 10^{-2}$
10^{40}	$\approx 1.1 \cdot 10^{38}$	$\approx 2.1 \cdot 10^{33}$	$\approx 1.9 \cdot 10^{-5}$
10^{80}	$\approx 5.4 \cdot 10^{77}$	$\approx 1.5 \cdot 10^{67}$	$\approx 2.8 \cdot 10^{-11}$
10^{160}	$\approx 2.7 \cdot 10^{157}$	$\approx 8.9 \cdot 10^{135}$	$\approx 3.3 \cdot 10^{-22}$

Tabelle 1: Fehlerwahrscheinlichkeit beim Fermat–Test

Der Primzahltest auf Basis des kleinen Fermat sieht dann folgendermaßen aus:

Verfahren 2.3 (Fermat)
Sei n eine auf Primalität[1] zu überprüfende Zahl.

1. Wähle ein $a \in \{2, \ldots, n-2\}$. Ist $\mathrm{ggT}(a, n) \neq 1$, so ist n sicher zusammengesetzt. Breche das Verfahren ab.

2. Berechne $a^{n-1} \bmod n$.

3. Ist $a^{n-1} \not\equiv 1 \bmod n$, dann ist n sicher eine zusammengesetzte Zahl, ist hingegen $a^{n-1} \equiv 1 \bmod n$, so handelt es sich bei n höchstwahrscheinlich um eine Primzahl.

Für praktische Zwecke und große Zahlen ist der Fermat–Test sehr sicher. Zudem geht die Berechnung von $a^{n-1} \bmod n$ sehr schnell von statten, das Computerprogramm „Enigma“[2] benötigt z.B. für die Berechnung von $2^{n-1} \bmod n$ mit $n \approx 10^{100}$ nur einige Sekunden. Im Gegensatz dazu benötigt der Probedivisionen–Algorithmus für den Nachweis der Primalität einer 100–stelligen Primzahl etwa 10^{37} Jahre!

Will man den Fermat–Test noch sicherer machen, so wäre der nächstliegende Gedanke, den Test für verschiedene Basen durchzuführen, in der Hoffnung,

[1] In der Fachliteratur wird dieses Verfahren oft als Zusammengesetztheitstest bezeichnet. Denn das einzig sichere Ergebnis, das man erhält, ist, daß eine Zahl zusammengesetzt ist. Eine Zahl, die den Fermat–Test besteht, ist nicht sicher, sondern nur höchstwahrscheinlich prim.

[2] Eine kurze Einführung in dieses Programm wird in Kapitel 7.1 gegeben.

daß deutlich weniger zusammengesetzte Zahlen den Test bestehen. Allerdings gibt es zusammengesetzte Zahlen, die Pseudoprimzahlen zu allen Basen sind. Ein Beispiel dafür ist $561 = 3 \cdot 11 \cdot 17$.

Definition:
Ist $n \in \mathbb{N}$ Pseudoprimzahl zu allen Basen $a \in B_n$, so wird n als Carmichael–Zahl bezeichnet.

Es gibt, wie erst vor wenigen Jahren von Alford, Granville und Pomerance bewiesen wurde, unendlich viele Carmichael–Zahlen (vgl. [AGP]). Genauer gilt, daß $C(n) := \sharp\{l \in \{1, \ldots, n\} \mid l \text{ ist Carmichael-Zahl}\} > n^c$ wobei $c \approx 0.1$.

Für die Charakterisierung der Carmichael–Zahlen wird folgender Satz benötigt:

Satz 2.10
Sei $p > 2$ eine Primzahl und $e \in \mathbb{N}$. Dann gibt es eine Primitivwurzel $\overline{g} \in (\mathbb{Z}/p^e\mathbb{Z})^*$, d.h. es ist $(\mathbb{Z}/p^e\mathbb{Z})^* = \{\overline{g}, \ldots, \overline{g}^{p^{e-1}(p-1)}\}$.

Der Beweis für den Fall $e = 1$ erfolgt in [Forster, S. 62], und für den Fall $e > 2$ erfolgt er in [Forster, S. 68f].

Satz 2.11
Eine ungerade zusammengesetzte Zahl $n \geq 3$ ist genau dann eine Carmichael–Zahl, wenn

1. n quadratfrei ist, d.h. es gibt keine Primzahl p mit $p^2 \mid n$, und
2. für jede Primzahl p mit $p \mid n$ gilt, daß $p - 1 \mid n - 1$.

Beweis:

1. Beweis, daß die Bedingungen hinreichend sind:
   Sei also $n = p_1 \cdot p_2 \cdot \ldots \cdot p_l$, mit paarweise verschiedenen Primzahlen p_i und $n - 1 =: (p_i - 1)s_i$. Dann gilt nach dem chinesischen Restsatz, daß
   $$\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p_1\mathbb{Z} \times \ldots \times \mathbb{Z}/p_l\mathbb{Z}$$
   Für $a \in B_n$ und alle $i \in \{1, \ldots, l\}$ gilt
   $$a^{n-1} \equiv a^{(p_i-1)s_i} \equiv 1 \bmod p_i$$
   Nach dem chinesischen Restsatz ist daher $a^{n-1} \equiv 1 \bmod n$.
2. Beweis, daß die Bedingungen notwendig sind:
   Sei n eine Carmichael–Zahl und p ein Primteiler von n. Dann ist $n = p^e m$ mit einem Exponenten $e \geq 1$ und einer zu p teilerfremden Zahl m. Nach Satz 2.10 existiert eine Primitivwurzel $\overline{g}$ in $\mathbb{Z}/p^e\mathbb{Z}$. Nach dem chinesischen Restsatz gibt es eine Zahl $a \in \{2, \ldots, n-2\}$ mit $a \equiv g \bmod p^e$ und $a \equiv 1 \bmod m$. Nach Konstruktion ist a zu n teilerfremd, d.h. $a \in B_n$. Es ist also $a^{n-1} \equiv 1 \bmod n$. Es folgt, daß $g^{n-1} \equiv a^{n-1} \equiv 1 \bmod p^e$. Da $(\mathbb{Z}/p^e\mathbb{Z})^*$, und somit auch die Primitivwurzel $\overline{g}$, die Ordnung $p^{e-1}(p-1)$ besitzt, folgt daraus, daß $p^{e-1}(p-1) \mid (n-1)$. Wegen $p \mid n$ gilt $p \nmid (n-1)$, es folgt, daß $e = 1$ und $(p-1) \mid (n-1)$.

□

Zudem gilt der folgende Satz:

Satz 2.12
Jede Carmichael–Zahl n besitzt mindestens drei verschiedene Primteiler.

Beweis:
Angenommen, es gibt eine Carmichael–Zahl n mit $n = pq$, wobei p und q verschiedene Primzahlen sind, o.B.d.A. sei $p < q$. Dann ist nach Satz 2.11

$$n - 1 \equiv 0 \bmod (q - 1)$$

andererseits ist

$$n - 1 \equiv (q - 1)p + p - 1 \equiv p - 1 \bmod (q - 1)$$

Da $0 < p - 1 < q - 1$, ist dies ein Widerspruch. □

Wie man an Tabelle 1 erkennen kann, sind Carmichael–Zahlen nicht so viel seltener als Pseudoprimzahlen, als daß man daraus einen signifikant schärferen Primzahltest entwicklen könnte. Die Entwicklung hin zu einem stärkeren Primzahltest muß also in eine andere Richtung gehen.

2.3 Der Miller–Rabin–Test

Mittels des Miller–Rabin–Tests kann man das Risiko, beim kleinen Fermat fälschlicherweise eine zusammengesetzte Zahl als Primzahl zu betrachten, beliebig verkleinern.

Definition:
Sei $n \in \mathbb{N}$ und $a \in B_n$. Man sagt, n erfüllt die Miller–Rabin–Bedingung zur Basis a, wenn gilt:
Ist $n - 1 = 2^s t$, wobei $2 \nmid t$, so ist entweder

$$a^t \equiv 1 \bmod n$$

oder es gibt ein $r \in \{0, \ldots, s - 1\}$ mit

$$a^{2^r t} \equiv -1 \bmod n$$

Mit Hilfe des folgenden Satzes kann man eine schärfere Bedingung für Primzahlen formulieren:

Satz 2.13
Sei p eine Primzahl, dann sind $+1$ und -1 die einzigen Quadratwurzeln von 1 in $\mathfrak{p}$.

Beweis:
Sei $\overline{a}$ eine Quadratwurzel von 1 in $\mathfrak{p}$, dann gilt $a^2 \equiv 1 \bmod p$, d.h. $p \mid (a^2 - 1) = (a+1)(a-1)$. Es folgt $p \mid (a+1)$ oder $p \mid (a-1)$, dies ist gleichbedeutend mit $a \equiv 1 \bmod p$ bzw. $a \equiv -1 \bmod p$. □

Als Korollar erhält man den folgenden Satz:

Satz 2.14 (Miller–Rabin)
Sei p eine Primzahl und $a \in B_p$, dann erfüllt p die Miller–Rabin–Bedingung zur Basis a.

Beweis:
Sei $a \in B_n$ und sei $p - 1 = 2^s t$ mit $2 \nmid t$. Für alle $r \in \mathbb{N}_0$ ist $\overline{a}^{2^r t}$ eine Wurzel von $\overline{a}^{2^{r+1} t}$. Ist also $\overline{a}^{2^{r+1} t} = 1$, so gilt nach Satz 2.13, daß $\overline{a}^{2^r t} = 1$ oder $\overline{a}^{2^r t} = -1$. Da $\overline{a}^{2^s t} = 1$, ist entweder $\overline{a}^{2^r t} = 1$ für alle $r \in \{0, \ldots, s\}$ oder es gibt ein $r \in \{0, \ldots, s-1\}$ mit $\overline{a}^{2^r t} = -1$. □

Allerdings gibt es auch jetzt wieder die Möglichkeit, daß eine zusammengesetzte Zahl die Bedingung zu einer Basis erfüllt:

Definition:
Sei n eine zusammengesetzte ungerade Zahl. Erfüllt n die Miller–Rabin–Bedingung zu einer Basis $a \in B_n$, so sagt man, n ist starke Pseudoprimzahl zur Basis a.

Ein Beispiel für eine starke Pseudoprimzahl zur Basis 2 ist $2047 = 23 \cdot 89$.

Die folgende Tabelle (welche [BRK, S. 155 und S. 166] entnommen ist) gibt einen Überblick über die Häufigkeit der verschiedenen Zahlentypen:

n	Anzahl der ungeraden Zahlen $l \in \{2, \ldots, n-2\}$ mit l			
	prim	Pseudoprimzahl zur Basis 2	starke Pseudoprimzahl zur Basis 2	Carmichael–Zahl
10^3	168	3	0	1
10^4	1229	22	5	7
10^5	9592	78	16	16
10^6	78498	245	46	43
10^7	664579	750	162	105
10^8	5761455	2057	488	255
10^9	50847534	5597	1282	646
10^{10}	455052511	14884	3291	1547

Abbildung 1: Anzahl von Pseudoprimzahlen und Carmichael–Zahlen.

Vergleicht man die 3. und die 4. Spalte von Tabelle 1, so sieht man, daß starke Pseudoprimzahlen nicht sehr viel seltener sind als „normale“ Pseudoprimzahlen. Den Fortschritt in Sachen Sicherheit bringt der folgende Satz:

Satz 2.15
Ist n eine ungerade zusammengesetzte Zahl, dann gibt es höchstens $\frac{n-1}{4}$ Basen zu denen n eine starke Pseudoprimzahl ist.

Zum Beweis des Satzes wird folgendes Lemmas benötigt:

Lemma 2.16
Sei G eine zyklische Gruppe mit m Elementen. Dann gibt es für ein $k \in \mathbb{N}$ genau $d := \mathrm{ggT}(k, m)$ Elemente $h \in G$, mit $h^k = 1$.

Beweis:
Sei g ein Erzeuger von G, d.h. $G = \{g, \ldots, g^m = 1\}$. Für ein Element $h = g^j$ mit $j \in \{1, \ldots, m\}$ gilt $h^k = g^{jk} = 1$ genau dann, wenn $m \mid jk$, d.h. $\frac{m}{d} \mid j\frac{k}{d}$. Dies ist wegen $\mathrm{ggT}(\frac{m}{d}, \frac{k}{d}) = 1$ gleichbedeutend mit $\frac{m}{d} \mid j$, d.h. $j \in \{\frac{m}{d}, \ldots, (d-1)\frac{m}{d}, m\}$. □

Jetzt zum Beweis von Satz 2.15:

Beweis:
Sei n eine ungerade zusammengesetzte Zahl, dann setze

$$\begin{aligned} P &:= \{a \in B_n \mid n \text{ ist starke Pseudoprimzahl zur Basis } a\} \\ A &:= \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid n \text{ ist starke Pseudoprimzahl zur Basis } a\} \cup \{-1, 1\} \\ C &:= \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{n-1} = 1\} \end{aligned}$$

Da $\pi_n(P) \subset A$ genügt es zu zeigen, daß $\sharp A \leq \frac{n-1}{4}$. Offensichtlich ist $A \subset C$.

Angenommen n ist nicht quadratfrei. Es gibt also eine Primzahl p mit $p^e \mid n$ wobei $e \geq 2$. Für $\overline{a} \in C$ gilt, daß insbesondere $a^{n-1} \equiv 1 \bmod p^2$. Setze $D := \{\overline{a} \in (\mathbb{Z}/p^2\mathbb{Z})^* \mid \overline{a}^{n-1} = 1\}$. Aus Satz 2.10 folgt, daß $(\mathbb{Z}/p^2\mathbb{Z})^*$ eine zyklische Gruppe mit $p(p-1)$ Elementen ist. Nach Lemma 2.16 gilt mit $d := \mathrm{ggT}\,(p(p-1), n-1)$, daß $\sharp(D) = d$. Da $p \mid n$, folgt $p \nmid (n-1)$ und daher $p \nmid d$, d.h. $d \leq p-1$. Wegen $\sharp(C) \leq \frac{n}{p^2} d$ folgt $\sharp(C) \leq (p-1)\frac{n}{p^2}$. Für den Anteil ergibt sich wegen $p \geq 3$, daß

$$\frac{\sharp A}{n-1} \leq \frac{\sharp C}{n-1} \leq \frac{(p-1)\frac{n}{p^2}}{n-1} \leq \frac{p-1}{p^2-1} = \frac{1}{p+1} \leq \frac{1}{4}$$

Im folgenden sei angenommen, daß n quadratfrei ist, d.h. es ist $n = p_1 \cdot \ldots \cdot p_l$ ein Produkt aus paarweise verschiedenen ungeraden Primzahlen. Nach dem chinesischem Restsatz gibt es einen Isomorphismus

$$\psi : (\mathbb{Z}/n\mathbb{Z})^* \stackrel{\cong}{\Rightarrow} (\mathbb{Z}/p_1\mathbb{Z})^* \times \ldots \times (\mathbb{Z}/p_l\mathbb{Z})^*$$

Es seien $\overline{g}_i \in (\mathbb{Z}/p_i\mathbb{Z})^*$ Primitivwurzeln von $(\mathbb{Z}/p_i\mathbb{Z})^*$.

1. Fall: n ist eine Carmichael–Zahl:
   Nach Satz 2.11 gilt $(p_i - 1) \mid (n-1)$ für alle i, d.h. $n - 1 =: 2^{s_i} u_i (p_i - 1)$ mit ungeraden Zahlen u_i. O.B.d.A. sei $s := s_1 \leq s_2 \leq \ldots \leq s_l$. Aus $a^{\frac{n-1}{2^s}} \equiv a^{(p_i-1)u_i 2^{s_i - s}} \equiv 1 \bmod p_i$ folgt $a^{\frac{n-1}{2^s}} \equiv 1 \bmod n$ für alle $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^*$.
   (a) Fall: $s = s_i$ für alle i.
   Dann ist $\frac{n-1}{2^{s+1}} = u_i \frac{p_i - 1}{2}$ ein ungerades Vielfaches von $\frac{p_i-1}{2}$. Offensichtlich ist

$$A \subset B := \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{\frac{n-1}{2^{s+1}}} = \pm 1\}$$

   Für $\overline{a} = \psi^{-1}(\overline{g}_1^{k_1}, \ldots, \overline{g}_l^{k_l})$ mit $k_i \in \{1, \ldots, p_i - 1\}$ gilt $a^{\frac{n-1}{2^{s+1}}} \equiv 1 \bmod n$ genau dann, wenn für alle $i \in \{1, \ldots, l\}$

$$a^{\frac{n-1}{2^{s+1}}} \equiv g_i^{k_i \frac{n-1}{2^{s+1}}} \equiv g_i^{k_i u_i \frac{p_i - 1}{2}} \equiv 1 \bmod p_i$$

d.h., wenn k_i gerade für alle i. Analog ist $a(k_1,\ldots,k_l)^{\frac{n-1}{2^{s+1}}} \equiv -1 \bmod n$ genau dann, wenn wenn k_i ungerade für alle i. Nach Satz 2.12 ist $l \geq 3$, d.h.

$$\begin{aligned}\sharp(A) &\leq \sharp(B)\\ &= \sharp\{(k_1,\ldots,k_l) \mid k_i \text{ ungerade für alle } i \text{ und } k_i \in \{1,\ldots,p_i-1\}\}\\ &\quad +\sharp\{(k_1,\ldots,k_l) \mid k_i \text{ gerade für alle } i \text{ und } k_i \in \{1,\ldots,p_i-1\}\}\\ &= 2(\tfrac{1}{2^l}(p_1-1)\cdot\ldots\cdot(p_l-1)) \leq \tfrac{1}{4}\varphi(n) \leq \tfrac{n-1}{4}\end{aligned}$$

(b) Fall: $s = s_1 < s_l$.
Dann ist $\frac{n-1}{2^{s+1}} = u_l 2^{s_l-s-1}(p_l-1)$ ein Vielfaches von p_l-1, insbesondere geradzahlig. Es ist also $a^{\frac{n-1}{2^{s+1}}} = 1 \bmod p_l$ für alle zu n teilerfremden $a \in \{1,\ldots,n-1\}$, es gibt deshalb kein $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^*$ mit $\overline{a}^{\frac{n-1}{2^{s+1}}} = -1$. Also gilt

$$A \subset B_1 := \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{\frac{n-1}{2^{s+1}}} = 1\}$$

B_1 ist insbesondere eine Untergruppe von $(\mathbb{Z}/n\mathbb{Z})^*$. Aus $\frac{n-1}{2^{s+1}} = u_1\frac{p_1-1}{2}$ folgt, daß

$$\psi^{-1}(\overline{g}_1,1,\ldots,1)^{\frac{n-1}{2^{s+1}}} = \psi^{-1}(\overline{g}_1^{\frac{n-1}{2^{s+1}}},1,\ldots,1) = \psi^{-1}(-1,1,\ldots,1) \neq \pm 1$$

also $\psi^{-1}(\overline{g}_1,1,\ldots,1) \in A \setminus B_1$, d.h. B_1 ist sogar eine echte Untergruppe von $(\mathbb{Z}/n\mathbb{Z})^*$. Deshalb ist $\sharp(B_1) \leq \frac{1}{2}\varphi(n)$.[1] Zudem gilt

$$A \subset B_2 := \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{\frac{n-1}{2^{s+2}}} = \pm 1\} \subset B_1$$

Für das Element $\overline{a} = \psi^{-1}(\overline{g}_1^2,1,\ldots,1)$ gilt $\overline{a}^{\frac{n-1}{2^{s+1}}} = 1$, und

$$\overline{a}^{\frac{n-1}{2^{s+2}}} = \psi^{-1}(\overline{g}_1^{2\frac{n-1}{2^{s+2}}},1,\ldots,1) = \psi^{-1}(-1,1,\ldots,1) \neq \pm 1$$

also $\overline{a} \in B_1 \setminus B_2$, deshalb ist B_2 eine echte Untergruppe von B_1 und es folgt, daß

$$\sharp(A) \leq \sharp(B_2) \leq \frac{1}{2}\sharp(B_1) \leq \frac{1}{4}\varphi(n) \leq \frac{n-1}{4}$$

2. Fall: n ist keine Carmichael–Zahl.
Es ist also C eine echte Untergruppe von $(\mathbb{Z}/n\mathbb{Z})^*$. Setze $d := \frac{\sharp((\mathbb{Z}/n\mathbb{Z})^*)}{\sharp(C)}$.

[1] Bekanntlich gilt für eine echte Untergruppe H einer endlichen Gruppe G, daß $\sharp(H) \leq \sharp(G)$ (vgl. [Kunz94, S. 130]).

Ist $d \geq 4$, so folgt $\frac{\sharp(A)}{n-1} \leq \frac{\sharp(C)}{n-1} \leq \frac{1}{4}$. Es sind also nur noch die Fälle $d = 2$ bzw. $d = 3$ zu betrachten. Aus dem chinesischen Restsatz und Lemma 2.16 folgt, daß

$$\frac{\sharp((\mathbb{Z}/n\mathbb{Z})^*)}{\sharp(C)} = \prod_{i=1}^{l} \frac{p_i - 1}{\mathrm{ggT}(p_i - 1, n-1)}$$

Es gibt also nur ein i, so daß $(p_i - 1) \nmid (n-1)$, o.B.d.A sei $i = 1$. Für $i \geq 2$ setze $n - 1 =: 2^{s_i} u_i (p_i - 1)$ mit $2 \nmid u_i$, o.B.d.A. sei $s := s_2 \leq \ldots \leq s_l$. Für $i = 0, \ldots, s+1$ setze

$$\begin{aligned} D_i &:= \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{\frac{n-1}{2^i}} = 1\} \\ E_i &:= \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^{\frac{n-1}{2^i}} = \pm 1\} \end{aligned}$$

Offensichtlich sind D_i und E_i (für $i > 0$) Untergruppen von C. Gilt $D_i = C$ so ist wegen $A \subset D_i = C$ insbesondere $A \subset E_{i+1} \subset D_i$. Folgende zwei Fälle werden unterschieden:

(a) $D_i = C$ für $i \in \{0, \ldots, s\}$.
Es ist also $A \subset E_{s+1}$. Aus $\frac{n-1}{2^s} = u_2(p_2 - 1)$ folgt für $\overline{a} := (1, \overline{g}_2, 1, \ldots, 1)$, daß

$$\begin{aligned} \overline{a}^{\frac{n-1}{2^s}} &= \psi^{-1}(1, \overline{g}_2^{u_2(p_2-1)}, 1, \ldots, 1) = \psi^{-1}(1, 1, \ldots, 1) = 1 \\ \overline{a}^{\frac{n-1}{2^{s+1}}} &= \psi^{-1}(1, \overline{g}_2^{u_2 \frac{(p_2-1)}{2}}, 1, \ldots, 1) = \psi^{-1}(1, -1, \ldots, 1) \neq \pm 1 \end{aligned}$$

Es ist also $\overline{a} \in D_s \setminus E_{s+1}$, damit ist E_{s+1} eine echte Untergruppe von D_s, es folgt $\sharp(A) \leq \sharp(E_{s+1}) \leq \frac{1}{2}\sharp(D_s) = \frac{1}{2}\sharp(C) \leq \frac{n-1}{4}$.

(b) Sonst.
Wegen $D_0 = C$ gibt es ein $j \in \{0, \ldots, s-1\}$ mit $D_j = C$ aber $D_{j+1} \neq C$. Sei $\overline{a} = \psi^{-1}(\overline{a}_1, \ldots, \overline{a}_l) \in D_j \setminus D_{j+1}$. Für $i \geq 2$ ist

$$a_i^{\frac{n-1}{2^{j+1}}} \equiv a_i^{2^{s_i - j - 1} u_i (p_i - 1)} \equiv 1 \bmod p_i$$

Es ist also $\overline{a}^{\frac{n-1}{2^{j+1}}} = \psi^{-1}(\overline{a}_1^{\frac{n-1}{2^{j+1}}}, 1, \ldots, 1)$. Da $\overline{a} \in D_j \setminus D_{j+1}$ folgt $\overline{a}_1^{\frac{n-1}{2^{j+1}}} \neq 1$, d.h. $\overline{a}^{\frac{n-1}{2^{j+1}}} \neq \pm 1$. Es ist also $\overline{a} \notin E_{j+1}$, d.h. E_{j+1} ist eine echte Untergruppe von $D_j = C$. Es folgt $\sharp(A) \leq \sharp(E_{j+1}) \leq \frac{1}{2}\sharp(C) \leq \frac{n-1}{4}$.

□

Wählt man k paarweise verschiedene Basen $a_1, \ldots, a_k \in B_n$ und erfüllt n die Miller–Rabin–Bedingung zu allen k Basen, so ist die Wahrscheinlichkeit, daß es sich bei n um keine Primzahl handelt, höchstens $\frac{1}{4^k}$.

Damit kann nun das Miller–Rabin–Verfahren formuliert werden:

Verfahren 2.4 (Miller–Rabin)
Sei n eine Zahl, die auf Primalität überprüft werden soll.

1. Wähle k paarweise verschiedene Zahlen $a_1, \ldots, a_k \in \{2, \ldots, n-2\}$. Ist $\mathrm{ggT}(a_i, n) \neq 1$ für ein $i \in \{1, \ldots, k\}$ so ist n sicher zusammengesetzt. Breche das Verfahren ab.

2. Bestimme die Zerlegung $n - 1 = 2^s t$, wobei $2 \nmid t$.

3. Setze $i := 1$.

4. Berechne $a_i^{n-1} \bmod n$, ist $a_i^{n-1} \not\equiv 1 \bmod n$, so ist n keine Primzahl. Breche das Verfahren ab.

5. Berechne $a_i^t \bmod n$. Ist $a_i^t \equiv 1 \bmod n$, so gehe zu 7.

6. Für $l = 1, \ldots, s$ berechne sukzessive $a_i^{2^l t} \equiv (a_i^{2^{l-1} t})^2 \bmod n$. Gibt es ein l, so daß $a_i^{2^l t} \equiv 1 \bmod n$ und $a_i^{2^{l-1} t} \not\equiv \pm 1 \bmod n$, so ist n nicht prim. Breche das Verfahren ab.

7. Ist $i < k$, dann setze $i := i + 1$ und gehe zu 4. Andernfalls ist die Wahrscheinlichkeit, daß es sich bei n um keine Primzahl handelt, höchstens $\frac{1}{4^k}$.

Unter der Voraussetzung, daß die verallgemeinerte Riemannsche Vermutung gilt, kann das Miller–Rabin–Verfahren sogar zum Beweisen der Primalität einer Zahl verwendet werden. Es gilt nämlich der folgende Satz (vgl. [Miller]):

Satz 2.17
Gilt die verallgemeinerte Riemannsche Vermutung, und erfüllt eine Zahl n die Miller–Rabin–Bedingung zu allen Basen $a \in \{2, \ldots, [2(\ln n)^2]\}$ so ist n eine Primzahl.

2.4 Der Pocklington–Primzahl–Test

Mit folgendem Satz ist es möglich, den Beweis der Primalität einer Zahl auf den Beweis der Primalität einer kleineren Zahl zurückzuführen:

Satz 2.18
Sei $n \in \mathbb{N}$. Gibt es eine Primzahl $p > \sqrt{n} - 1$ mit $p \mid (n-1)$, und gibt es ein $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$ mit

1. $\overline{a}^{\,n-1} = 1$, und

2. $\left(\overline{a}^{\frac{n-1}{p}} - 1\right) \in (\mathbb{Z}/n\mathbb{Z})^*$,

dann ist n prim.

Beweis:
Angenommen n ist nicht prim. Dann existiert eine Primzahl $q \leq \sqrt{n}$, welche n teilt. Da p eine Primzahl ist und $p > \sqrt{n} - 1 \geq q - 1$, ist $\mathrm{ggT}(p, q-1) = 1$. Es existiert also ein $u \in \mathbb{N}$ mit $up \equiv 1 \bmod (q-1)$. Dann gilt in $\mathbb{Z}/q\mathbb{Z}$, daß

$$\overline{a}_q^{\frac{n-1}{p}} = \overline{a}_q^{up\frac{n-1}{p}} = \overline{a}_q^{u(n-1)} = ((\overline{a}^{n-1})_q)^u = 1$$

D.h. $\left(\overline{a}^{\frac{n-1}{p}} - 1\right) \in q(\mathbb{Z}/n\mathbb{Z})$, im Widerspruch zu 2. □

Die Bedeutung des Satzes liegt darin, daß er eine hinreichende Bedingung für die Primalität einer Zahl gibt, im Gegensatz zu den notwendigen Bedingungen, die der kleine Fermat bzw. der Satz von Miller–Rabin geben.

Aufbauend auf Satz 2.18 kann folgender deterministischer Algorithmus formuliert werden:

Algorithmus 2.5 (Pocklington)
Sei n eine Zahl, die auf Primalität überprüft werden soll.

1. Ist n klein (etwa $n < 10^9$), so überprüfe n mittels des Probedivisionen–Algorithmus (Algorithmus 2.1) auf Primalität. Breche den Pocklington–Algorithmus ab.

2. Wähle eine Schranke $A \leq \sqrt{n}$.

3. Finde alle Primfaktoren $q < A$ von $n-1$. Sei p die Zahl, die man erhält, wenn man $n-1$ durch die maximal möglichen Potenzen der gefundenen Primfaktoren teilt. Ist $p \leq \sqrt{n} - 1$, oder besteht p den Fermat–Test zu einer beliebig gewählten Basis nicht, so breche den Algorithmus ab. Es wurde kein Ergebnis gefunden.

4. Wähle ein $a \in \{1, \ldots, n-1\}$.

5. Ist $a^{n-1} \not\equiv 1 \bmod n$, so ist n sicher zusammengesetzt, breche den Algorithmus ab.

6. Andernfalls setze $d := \mathrm{ggT}(a^{\frac{n-1}{p}} - 1, n)$. Ist $d = 1$, so setze $n := p$ und starte erneut den Algorithmus.[1] Ist $d := n$, so gehe zu 4. und wähle ein anderes a, andernfalls ist d ein echter Teiler von n. Breche den Algorithmus ab.

[1] Da $p \mid (n-1)$ und $2 \mid (n-1)$, ist $p \leq \frac{n-1}{2}$. D.h. der Algorithmus wird sich maximal $([\log_2 n] + 1)$–mal selbst aufrufen, bis der Nachweis der Primalität von n auf den Nachweis der Primalität einer Zahl zurückgeführt ist, welche im Bereich des Probedivisionen–Algorithmus liegt.

Es kann also vorkommen, daß der Algorithmus ohne Ergebnis abgebrochen wird, obwohl n prim ist. Dieser Fall tritt dann auf, wenn $n-1$ von keiner großen Primzahl geteilt wird, die Wahrscheinlichkeit dafür ist etwa 30%.[1] Wird hingegen $n-1$ von einer großen Zahl p geteilt, welche nach dem kleinen Fermat höchstwahrscheinlich eine Primzahl ist, so ist noch die Primalität von p zu beweisen. Dies geschieht durch Anwenden des Algorithmus auf p. Dabei ist im Durchschnitt $\ln p \approx 0.65 \ln n$.[1] Der Algorithmus wird irgendwann einmal bei einer Zahl m stoppen, für die $m-1$ nicht die gewünschten Eigenschaften hat. Erwartungsgemäß gilt dabei

$$\ln m = \left(1-\frac{2}{3}\right)\sum_{n=0}^{n=\infty}\left(\frac{2}{3}\right)^n (0.65)^n \ln n = \frac{1-\frac{2}{3}}{1-0.65\cdot\frac{2}{3}}\ln n \approx 0.59 \ln n$$

Der Beweis der Primalität von n ist auf den deutlich einfacheren Beweis der Primalität von m zurückgeführt.

Satz 2.18 ist ein Spezialfall eines viel allgemeineren Satzes. Zur Einführung dieses Satzes wird folgende Defintion benötigt:

Definition:
Seien $s, n \in \mathbb{N}$ und $G_n \subset (\mathbb{Z}/n\mathbb{Z})^s$. Zudem gebe es eine Menge $T \subset G_n \times G_n$ und Abbildungen

$$\begin{array}{rrcl} \oplus: & T & \to & G_n \\ \theta: & (G_n \times G_n)\setminus T & \to & \{l \in \{2,\ldots,n-1\} \mid l \text{ ist Teiler von } n\} \end{array}$$

$(G_n, \oplus, \varphi)$, oder kurz G_n, wird als n–Pseudogruppe bezeichnet, wenn die teilweise Verknüpfung $\oplus$ über ein neutrales Element $O = (\overline{o}_1, \ldots, \overline{o}_s) \in T$ verfügt, und wenn für jeden Primteiler p von n die Abbildung[2]

$$\begin{array}{rrcl} \pi_{np}: & G_n & \to & (\mathbb{Z}/p\mathbb{Z})^s \\ & A=(\overline{a}_1,\ldots,\overline{a}_s) & \mapsto & (\overline{a}_{1p},\ldots,\overline{a}_{sp}) =: A_p \end{array}$$

die Eigenschaft hat, daß $G_p := \{P_p \mid P \in G_n\}$ eine Gruppenstruktur $\oplus_p$ besitzt, und daß $A_p \oplus_p B_p = (A \oplus B)_p$, sofern $A \oplus B$ definiert ist, d.h. sofern $(A, B) \in T$.

[1] Vergleiche Abbildung 6.

[2] Neben $\pi_{np} : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ wird also noch eine weitere Abbildung mit π_{np} bezeichnet. Im folgenden dürfte allerdings immer klar sein, welche der beiden Abbildungen gemeint ist.

Die letzte Eigenschaft bedeutet, daß folgendes Diagramm kommutiert:

$$\begin{array}{ccc} (A,B) & \xrightarrow{\oplus} & A \oplus B \\ \Big\downarrow{\scriptstyle \pi_{np} \times \pi_{np}} & & \Big\downarrow{\scriptstyle \pi_{np}} \\ (A_p, B_p) & \xrightarrow{\oplus_p} & A_p \oplus_p B_p \end{array}$$

Es folgt, daß O_p das neutrale Element in der Gruppe ist.

Eine n–Pseudogruppe G_n verfügt also insbesondere über eine teilweise Verknüpfung $\oplus$ und eine Funktion, welche, falls die Verknüpfung nicht definiert ist, einen nichttrivialen Teiler von n liefert.

Bemerkung 2.19
Wie man sich leicht vergewissern kann, ist $(\mathbb{Z}/n\mathbb{Z})^*$ mit $T = (\mathbb{Z}/n\mathbb{Z})^*$ und $\overline{a} \oplus \overline{b} := \overline{a}\,\overline{b}$ eine n–Pseudogruppe.

Für die Anwendungen in den verschiedenen Sätzen benötigt man eine Definition von „$k \odot A$“. Dies ist allerdings nicht ganz trivial, denn ebenso wie es verschiedene Möglichkeiten gibt, $\overline{a}^k$ zu berechnen, gibt es verschiedene Möglichkeiten „$k \odot A$“ zu berechnen. Da G_n keine Gruppe ist, müssen die verschiedenen Ergebnisse für „$k \odot A$“ nicht übereinstimmen.

Definition:
Sei G_n eine n–Pseudogruppe und sei $k = (k_l \dots k_0)_2 \in \mathbb{N}$ sowie $A \in G_n$. Sofern die folgenden Verknüpfungen definiert sind, setze $2 \odot A := A \oplus A$ und induktiv $2^j \odot A := (2^{j-1} \odot A) \oplus (2^{j-1} \odot A)$, sowie

$$kA := k \odot A := \bigoplus_{j=0}^{l} 2^j \odot A$$

Anders ausgedrückt, $k \odot A$ soll mittels der Methode des sukzessiven Quadrierens berechnet werden.

Satz 2.20
Sei G_n eine n–Pseudogruppe und sei $m := \sharp G_n$. Es gebe eine Primzahl p welche m teilt und mit der Eigenschaft, daß $p > \min\{\sharp G_q \mid q \text{ ist Primteiler von } n\}$ falls n zusammengesetzt ist. Gibt es ein $A \in G_n$ mit den Eigenschaften

1. $m \odot A = O$, und
2. $\left(\frac{m}{p}\right) \odot A$ ist definiert,
3. mit $(\overline{b}_1, \dots, \overline{b}_s) := \left(\frac{m}{p}\right) \odot A$ sind $(\overline{b}_1 - \overline{o}_1), \dots, (\overline{b}_s - \overline{o}_s) \in (\mathbb{Z}/n\mathbb{Z})^*$,

so ist n eine Primzahl.

Beweis:
Angenommen n ist nicht prim, dann existiert eine Primzahl q welche n teilt und mit $m_q := \sharp G_q < p$. Da p eine Primzahl ist, folgt $\mathrm{ggT}(p, m_q) = 1$. Es existiert daher ein $u \in \mathbb{N}$ mit $up \equiv 1 \bmod m_q$. Dann gilt, daß

$$\begin{aligned}\left(\left(\tfrac{m}{p}\right) \odot A\right)_q &= \left(\tfrac{m}{p}\right) \odot_q A_q = up \odot_q \left(\tfrac{m}{p}\right) \odot_q A_q = \\ &= u \odot_q (m \odot_q A_q) = u \odot_q (m \odot A)_q = O_q\end{aligned}$$

Es ist also $\pi_{nq}\left(\left(\frac{m}{p}\right) \odot A\right) = O_q$, d.h. $(\overline{b}_1 - \overline{o}_1, \ldots, \overline{b}_s - \overline{o}_s) \in q(\mathbb{Z}/n\mathbb{Z})$. Dies aber ist ein Widerspruch zu 3. □

Satz 2.18 erhält man als Korollar, indem man $(\mathbb{Z}/n\mathbb{Z})^*$ als n–Pseudogruppe wählt. Eine weitere Anwendung des Satzes wird in Kapitel 6.2 beschrieben.

2.5 Auffinden von Primzahlen gegebener Größenordnung

Für den RSA–Verschlüsselungsalgorithmus werden zwei verschiedene, große Primzahlen benötigt, wenn möglich sogar mit bestimmten Zusatzeigenschaften. Zum Beispiel sollten, wie in Kapitel 4.3 einsichtig wird, beide Primzahlen die Eigenschaften besitzen, daß $p - 1$ eine große Primzahl als Teiler besitzt.

Nach dem Primzahlsatz (Satz 2.9) ist die Wahrscheinlichkeit, daß eine beliebige Zahl n eine Primzahl ist, ungefähr $\frac{1}{\ln n}$. Sucht man also eine Primzahl p von der Größenordnung n, so wird man durchschnittlich etwa $[\ln n]$ Zahlen untersuchen müssen, bis man eine Primzahl findet. Zum Beispiel bedeutet das für $n = 10^{100}$, daß man im Durchschnitt 230 Zahlen überprüfen muß. Um eine beliebige Primzahl dieser Größenordnung zu finden, benötigt das Programm „Enigma“ etwa $230 \cdot 4,87s \approx 19\,\mathrm{min}$. Diese Zeit kann deutlich reduziert werden. Denn die Berechnungsdauer von $a^{n-1} \bmod n$ hängt nicht davon ab, ob n eine Primzahl ist oder nicht. Überprüft man zuerst, ob n durch kleine Primzahlen teilbar ist, kann man viele Zahlen, die „offensichtlich“ zusammengesetzt sind, schon vorher aussieben. Die Zeitersparnis kann mit Hilfe des folgenden Satzes berechnet werden:

Satz 2.21
Seien $p_1, \ldots, p_k$ paarweise verschiedene Primzahlen. Dann gilt für die Wahrscheinlichkeit W, daß eine beliebige Zahl n keine dieser Primzahlen als Teiler besitzt, daß

$$W = \left(1 - \frac{1}{p_1}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right)$$

Beweis:
Für $k = 1$ ist die Aussage klar. Der Fall $k > 1$ folgt aus der Tatsache, daß die Wahrscheinlichkeiten W_i dafür, daß eine Zahl n von p_i nicht geteilt wird, stochastisch unabhängig sind. □

Es ergibt sich folgende Tabelle:

Anzahl von Primzahlen k	Wahrscheinlichkeit W, daß eine Zahl n nicht von den k kleinsten Primzahlen geteilt wird
10	15.8 %
100	8.9 %
1000	6.2 %
10000	4.9 %

Folgendes Verfahren bietet sich an, um schnell eine Primzahl der Größenordnung n zu finden:

Verfahren 2.6
Sei n eine Zahl, in deren Größenordnung eine Primzahl gefunden werden soll.

1. Wähle eine Zahl $k \in \mathbb{N}$.
2. Berechne $a_i := n \bmod p_i$ für $i = 1, \ldots, k$, wobei $\{p_1, \ldots, p_k\}$ die Menge der k kleinsten Primzahlen sei.
3. Ist $a_i = 0$ für ein $i \in \{1, \ldots, k\}$, dann gehe zu 4., ansonsten überprüfe ob n einen Primzahltest wie etwa den kleinen Fermat besteht. Wenn ja, dann breche das Verfahren ab.
4. Setze $n := n + 1$ und $a_i := (a_i + 1) \bmod p_i$ für $i = 1, \ldots, k$. Gehe zu 3.

Für $l = 100$ muß man nur noch jede 11. Zahl auf Primalität überprüfen. Da die Berechnung in 2. nur einmalig erfolgt, bedeutet dies eine deutliche Zeitersparnis.

3 Geschwindigkeitsabschätzung von Algorithmen

In diesem Kapitel wird eine Schreibweise eingeführt, die es ermöglicht, die Geschwindigkeit von Algorithmen abzuschätzen und zu vergleichen.
Für den Rest des Kapitels sei $b \in \mathbb{N} \setminus \{1\}$ fest gewählt.

Notation:
Eine Zahl $n \in \mathbb{N}$ wird bzgl. der Basis b, d.h. in der b–adischen Darstellung folgendermaßen geschrieben:

$$n = (n_{k-1} \dots n_0)_b$$

wobei $n_i \in \{0, \dots, b-1\}$ eindeutig festgelegt sind durch die Bedingung[1]

$$n = n_{k-1} b^{k-1} + \dots + n_1 b + n_0$$

Beispielsweise ist $100 = (100)_{10} = (1100100)_2 = (202)_7$.

Definition:
Die Addition bzw. Multiplikation von zwei b–Ziffern wird als b–Rechenschritt, oder auch kurz als b–Schritt, bezeichnet. Ist $b = 2$, so sagt man anstatt 2–Schritt auch Bit–Operation.
Die Geschwindigkeit eines Verfahrens hängt vor allem davon ab, wieviele Rechenschritte es benötigt.[2] Um die Anzahl der Bit–Operationen eines Verfahrens abschätzen zu können, müssen zuerst einige grundlegende Tatsachen dargelegt werden:
Betrachte folgenden Algorithmus zur Addition von x und y:
Algorithmus 3.1
Seien $x = (x_{k-1} \dots x_0)_b$ und $y = (y_{k-1} \dots y_0)_b$ zwei Zahlen, die addiert werden sollen.

1. Setze $i := 0$, $u_0 := 0$.
2. Setze $z_i := x_i + y_i + u_i$.
3. Ist $z_i \geq b$, dann setze $u_{i+1} := 1$ und $z_i := z_i - b$.
4. Setze $i := i + 1$. Ist $i \leq k$, dann gehe zu 2., ansonsten ist

$$z := (z_k \dots z_0)_b = x + y$$

[1] Zudem ist $n_i := 0$ für $i \geq k$.

[2] Es gibt noch viele weitere geschwindigkeitsbeeinflußende Faktoren, wie z.B. Verschiebungen im Speicher. Diese werden allerdings bei den folgenden Geschwindigkeitsabschätzungen nicht berücksichtigt.

Hier, wie bei den weiteren Geschwindigkeitsabschätzungen, wird der Rechenbedarf für den Schritt $i := i + 1$ vernachlässigt. Man erhält dann, daß der Algorithmus maximal $3(k + 1)$ b–Schritte benötigt. Darauf aufbauend kann man den Rechenbedarf des folgenden Multiplikationsalgorithmus abschätzen:

Algorithmus 3.2
Seien $x = (x_{k-1} \dots x_0)_b$ und $y = (y_{k-1} \dots y_0)_b$ zwei, Zahlen die multipliziert werden sollen.

1. Setze $j := 0$.
2. Setze $i := 0$, $u_0 := 0$.
3. Berechne $f := x_i \cdot y_j + u_i$.
4. Ist $f = (f_1, f_0)_b$, so setze $u_{i+1} = f_1$ und $e_i := f_0$.
5. Setze $i := i + 1$. Ist $i \leq k$, dann gehe zu 3.
6. Setze $z_j := (e_l, \dots, e_0, \underbrace{0, \cdots, 0}_{j-\text{mal}})_b$.
7. Setze $j := j + 1$. Ist $j \leq k$, dann gehe zu 2.
8. Setze $z := z_0 + \ldots + z_l$. Dann ist $z = x \cdot y$.

Für den 3. Schritt, welcher $(k + 1)^2$–mal erfolgt, benötigt man 5 b–Schritte. Im Schritt 8 werden $(k+1)$ Zahlen der maximalen Länge $2k+1$ addiert. Dafür werden weitere $3(k + 1)(2k + 1)$ b–Schritte benötigt. Insgesamt benötigt man also maximal $11(k + 1)^2$ b–Schritte für die Multiplikation zweier k–stelliger Zahlen.

Eine gröbere, aber auch besser lesbare Zeitabschätzung ermöglicht die folgende Notation:

Notation:
Seien $f, g : \mathbb{R}^+ \to \mathbb{R}^+$ beliebige Funktionen. Gibt es eine Konstante $C \in \mathbb{R}^+$, so daß $f(x) < Cg(x)$ für alle $x \in \mathbb{R}^+$, so schreibt man $f(x) = O(g(x))$. Gilt sogar, daß $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$, dann schreibt man $f(x) = o(g(x))$.

Seien $x = (x_{k-1} \dots x_0)_b, y = (y_{k-1} \dots y_0)_b \in \mathbb{N}$. Dann kann die Zeitabschätzungen der Algorithmen 3.1 und 3.2 auch so formuliert werden:

$$\begin{array}{lcll} \text{Zeit(Berechnung von } x + y) & = & O(k) & \text{Bit–Operationen}^1 \\ \text{Zeit(Berechnung von } x \cdot y) & = & O(k^2) & \text{Bit–Operationen} \end{array}$$

Man kann auch leicht zeigen, daß gilt:

$$\begin{array}{lcll} \text{Zeit(Berechnung von } x-y) & = & O(k) & \text{Bit–Operationen} \\ \text{Zeit(Berechnung von } x/y) & = & O(k^2) & \text{Bit–Operationen} \end{array}$$

Es gibt Algorithmen welche für die Multiplikation großer Zahlen deutlich weniger Zeit verbrauchen. Ein Beispiel dafür ist der folgende, von Karatsuba und Ofman 1962 entwickelte, Multiplikationsalgorithmus (vgl. [KO]):

Algorithmus 3.3 (Karatsuba und Ofman)
Seien x und y zwei k–stellige Zahlen, die multipliziert werden sollen. Sind x und y einstellige Zahlen, dann multipliziere sie wie üblich. Andernfalls:

1. Schreibe x und y folgendermaßen:

   $$x =: x_1 b^m + x_0, \quad y =: y_1 b^m + y_0$$

   wobei $m := [\frac{k}{2}] + 1$ und $x_0, y_0 < b^m$.

2. Berechne $t_1 := x_1 y_1$, $t_2 := x_0 y_0$ sowie $t_3 := (x_1 + x_0)(y_1 + y_0)$ mittels des gleichen Algorithmus. Dann ist[2]

   $$xy = t_1 b^{2m} + (t_3 - t_2 - t_1) b^m + t_2$$

Jede Multiplikation zweier k–stelliger Zahlen wird durch drei Multiplikationen von zwei $\left(\left[\frac{k}{2}\right] + 1\right)$–stelligen Zahlen ersetzt. Ist $l := [\ln_2(k)] + 1$, so benötigt man, wie man leicht durch Induktion zeigen kann, 3^l b–Schritte für die Multiplikation von x und y. Es ergibt sich eine Gesamtzeit von $O(3^{\ln_2 k}) = O(k^{\frac{\ln 3}{\ln 2}}) = O(k^{1.59})$, denn die Zeit für die Additionen bzw. die Multiplikationen mit b^m bzw. b^{2m} wird von dieser Abschätzung dominiert.[3]

Dies ist allerdings noch nicht der schnellste Algorithmus; durch Verwendung von Fourier–Transformationen konnten Schönhage und Strassen einen Multiplikationsalgorithmus entwickeln, welcher zwei k–stellige Zahlen x, y mit nur $O(k \ln k \ln\ln k)$ Bit–Operationen multipliziert. Allerdings er erst sinnvoll für Zahlen mit mehreren hundert Stellen. Der Algorithmus wird ausführlich in [Forster] beschrieben. Bei den folgenden Geschwindigkeitsabschätzungen wird weiterhin davon ausgegangen, daß man für die Multiplikation $O(k^2)$ Bit–Operationen benötigt.

[1] Genauer müßte man sagen, es gibt Algorithmen, so daß die Addition von x und y mit $O(k)$ Bit–Operationen erfolgt.

[2] Denn $xy = x_1 y_1 b^{2m} + (x_1 y_0 + x_0 y_1) b^m + x_0 y_0 = t_1 b^{2m} + (t_3 - t_2 - t_1) b^m + t_2$

[3] Die Multiplikation mit einer Potenz von b entspricht einer Verschiebung der Ziffern in der b–adischen Notation. Man benötigt dafür also keine Rechnungen.

Konvention:
Ein ϵ in der O–Notion bedeutet, daß der Ausdruck für jedes $\epsilon > 0$ richtig ist.

Beispielsweise ist $O(\ln n) = O(n^\epsilon)$, denn $\lim_{n\to\infty} \frac{\ln n}{n^\epsilon} = 0$ für alle $\epsilon > 0$. Für den Multiplikationsalgorithmus von Schönhage und Strassen ergibt sich also eine Geschwindigkeitsabschätzung von $O\left(k^{(1+\epsilon)}\right)$ Bit–Operationen.

Mit den bisher erfolgten Abschätzungen kann man nun leicht die Geschwindigkeit des Probedivisionen–Algorithmus abschätzen:

Satz 3.1
Sei $n \in \mathbb{N}$. Überprüft man n mittels des Probedivisionen–Algorithmus auf Primalität, so benötigt man $O((\ln n)^2\sqrt{n}\,) = O(n^{\frac{1}{2}+\epsilon})$ Bit–Operationen.[1]

Als weiteres kann die Geschwindigkeit des Euklidischen Algorithmus abgeschätzt werden:

Satz 3.2
Seien $p, q \in \mathbb{N}$ mit $p > q$ gegeben. Unter Verwendung des Algorithmus 2.2 benötigt man $O((\ln p)^3)$ Bit–Operationen für die Berechnung von $\mathrm{ggT}(p, q)$ und die Bestimmung von r, s, so daß $pr + qs = \mathrm{ggT}(p, q)$.

Beweis:
Die Bezeichnungen seien dieselben wie bei Algorithmus 2.2. Ist $r_{j+1} > \frac{1}{2}r_j$ so folgt, daß $r_j = 1 \cdot r_{j+1} + r_{j+2}$, also $r_{j+2} = r_j - r_{j+1} < \frac{1}{2}r_j$. Ist dagegen $r_{j+1} \leq \frac{1}{2}r_j$, so folgt ebenfalls $r_{j+2} < r_{j+1} \leq \frac{1}{2}r_j$.
Man benötigt also $O(\ln a)$ Divisionen, bis man bei $r_{i+2} = 0$ angelangt ist. Da man für jede Division $O((\ln a)^2)$ Bit–Operationen benötigt, ergibt sich eine Gesamtzeit von $O((\ln a)^3)$ Bit–Operationen. □

Sei $n \in \mathbb{N}$. Dann kann der Ring $\mathbb{Z}/n\mathbb{Z}$ mit dem Ring $\{0, \ldots, n-1\}$ identifiziert werden, wobei die Verknüpfungen auf $\{0, \ldots, n-1\}$ als Verknüpfung in $\mathbb{Z}$ mit anschließender Reduktion modulo n definiert sind. Eine Geschwindigkeitsabschätzung für den Ring $\{0, \ldots, n-1\}$ ist also auch für den Ring $\mathbb{Z}/n\mathbb{Z}$ gültig. Zur Zeitabschätzung für Rechnungen im Ring $\mathbb{Z}$ kommt also noch die Zeitabschätzung für die Reduktion modulo n hinzu. Bei der Addition bzw. Subtraktion ist das Zwischenergebnis im Bereich $[-n+1, 2n-2]$, d.h. die Reduktion besteht aus höchstens einer Addition und benötigt $O(\ln n)$ Bit–Operationen. Bei der Multiplikation benötigt man für die Reduktion modulo

[1] Der Satz müßte genauer folgendermaßen formuliert werden: Es gibt Multiplikations– und Additionsalgorithmen, so daß man die Primalität eines gegebenen $n \in \mathbb{N}$ mittels des Probedivisionen–Algorithmus mit $O((\ln n)^2\sqrt{n}) = O(n^{\frac{1}{2}+\epsilon})$ Bit–Operationen beweisen kann. Im weiteren sollen alle derartigen Sätze so verstanden werden. Der Satz schließt aber nicht aus, daß es nicht schnellere Verknüpfungsalgorithmen gibt.

n eine Division, d.h. $O((\ln n)^2)$ Bit–Operationen. Insgesamt gilt also für die Rechnungen im Ring $\mathbb{Z}/n\mathbb{Z}$:

$$\begin{array}{lcll} \text{Zeit(Berechnung von } \overline{x}+\overline{y}) & = & O(\ln n) & \text{Bit–Operationen} \\ \text{Zeit(Berechnung von } \overline{x}-\overline{y}) & = & O(\ln n) & \text{Bit–Operationen} \\ \text{Zeit(Berechnung von } \overline{x}\cdot\overline{y}) & = & O((\ln n)^2) & \text{Bit–Operationen} \end{array}$$

Eine Abweichung von den Zeitabschätzungen für den Ring $\mathbb{Z}$ ergibt sich bei der Bestimmung des Inversen. Das Inverse eines Elementes $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$ erhält man durch Anwendung von Algorithmus 2.2 mit $p := a$ und $q := n$. Dann ist $\overline{a}^{-1} = \overline{r}$. Nach Satz 3.2 ergibt sich folgende Geschwindigkeitsabschätzung:

$$\text{Zeit(Berechnung von } \overline{x}\cdot\overline{y}^{-1}) \;=\; O((\ln n)^3) \text{ Bit–Operationen}$$

Damit kann der folgende Satz bewiesen werden:

Satz 3.3
Sei $n \in \mathbb{N}$ und $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$. Für die Berechnung von $\overline{a}^{n-1}$ bzw. die Berechnung von $a^{n-1} \bmod n$ werden $O((\ln n)^3)$ Bit–Operationen benötigt.

Zum Beweis dieses Satzes betrachte den folgenden Algorithmus:

Algorithmus 3.4 (Sukzessives quadrieren)
Sei $n \in \mathbb{N}$ und $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$. Ist $n-1 = (n_{k-1} \dots n_0)_2$.[1] so berechne folgendermaßen $\overline{a}^{n-1}$:

1. Setze $\overline{a}_0 := \overline{a}$.
2. Ist $n_0 = 1$, so setze $\overline{e} := \overline{a}$, sonst setze $\overline{e} := 1$.
3. Wiederhole die Schritte 3.1 und 3.2 für $i = 1, \dots, k-1$:
   (a) Setze $\overline{a}_i := \overline{a}_{i-1}^2$.
   (b) Ist $n_i = 1$ so setze $\overline{e} := \overline{e} \cdot \overline{a}_i$.

Es ist $\overline{a} = \overline{a}^{n-1}$.[2]

Man benötigt für die $k = O(\ln n)$ Schritte jeweils maximal zwei Multiplikationen, das entspricht $O((\ln n)^2)$ Bit–Operationen. Insgesamt ergibt sich also eine Zeit von $O((\ln n)^3)$ Bit–Operationen für die Berechnung von $\overline{a}^{n-1}$.

[1] Ist n nicht in der 2–adischen Schreibweise gegeben, so werden $O((\ln n)^2)$ Bit–Operationen zur Umwandlung benötigt (vgl. [Koblitz, S. 9]).

[2] Denn $\overline{e} = \overline{a}^{2^{k-1} n_{k-1}} \dots \overline{a}^{2 n_1} \overline{a}^{n_0} = \overline{a}^{\sum_{i=0}^{k-1} n_i 2^i} = \overline{a}^{n-1}$.

Mit Satz 3.3 kann man auch eine Geschwindigkeitsabschätzung für den kleinen Fermat und das Miller–Rabin–Verfahren geben: Die Verfahren 2.3 und 2.4 benötigen für ein gegebenes $n \in \mathbb{N}$ jeweils $O((\ln n)^3)$ Bit–Operationen. Dies zeigt auch die Schwäche der O–Notation, denn obwohl die O–Notation die gleiche Geschwindigkeit angibt, ist der Miller–Rabin–Test z.B. bei einer gewünschten Fehlerwahrscheinlichkeit von weniger als 10^{-20} um den Faktor $33 \approx [\log_4(10^{20})]$ langsamer.

Zum Abschluß des Kapitels soll noch eine Geschwindigkeitsabschätzung des RSA–Algorithmus gegeben werden. Sollen die beiden zu suchenden Primzahlen von der Größenordnung n sein, so muß man im Durchschnitt $\ln n$ Zahlen auf Primalität überprüfen, bis man eine Primzahl gefunden hat. Es ergibt sich also eine Gesamtzeit von $O((\ln n)^4)$ Bit–Operation für das Auffinden der beiden Primzahlen. Sei eine Zeichenmenge M und ein Text T der Länge l gegeben. Der Text muß dann in $\left[\frac{l}{\log_M(2n)}\right] + 1$ Blöcke aufgeteilt werden. Zur Verschlüsselung eines Blocks benötigt man $O((\ln(2n))^3)$ Bit–Operationen. Es ergibt sich damit eine Gesamtzeit von

$$O\left(\left[\frac{l}{\log_M(2n)}\right]\right) O((\ln(2n))^3) = O(l(\ln n)^2)$$

Bit–Operationen. Für die Verschlüsselung eines langen Textes ist der RSA–Algorithmus damit ungeeignet. Tatsächlich wird z.B. beim weitverbreiteten Programm „Pretty Good Privacy“ der RSA–Algorithmus nur zur Verschlüsselung des Schlüssels eines klassischen Decodierverfahrens verwendet, mit welchem der eigentliche Text verschlüsselt ist.

4 Klassische Faktorisierungsalgorithmen

Der bekannteste Algorithmus, eine Zahl zu faktorisieren, ist der schon erwähnte Probedivisionen–Algorithmus (Algorithmus 2.1). Dabei handelt es sich um einen deterministischen Faktorisierungsalgorithmus; hat nämlich eine Zahl n einen Primteiler p, so wird dieser nach p Probedivisionen sicher gefunden. Die in diesem Kapitel beschriebenen Algorithmen sind nicht mehr deterministisch, sondern nur noch probabilistisch, der Erfolg hängt von Zufallselementen ab. Es gibt keine Garantie, daß die Algorithmen nach einer nur von n abhängigen Zeit einen Teiler p finden werden. Aber es ist sehr wahrscheinlich, daß sie deutlich schneller sind, als der Probedivisionen–Algorithmus.

Im gesamten Kapitel 4 sei $n \in \mathbb{N}$ eine zu faktorisierende Zahl.

4.1 Der Rho–Algorithmus

Der Rho–Algorithmus wurde 1975 von J. Pollard entwickelt (vgl. [Pollard]). Er hat den Vorteil, daß er leicht zu programmieren ist und „kleine“ Primfaktoren, d.h. Primfaktoren bis etwa 10^{10}, schnell findet, während er asymptotisch, d.h. zum Auffinden von größeren Primfaktoren, zu langsam ist.

Der Ausgangspunkt ist, daß man rekursiv eine Zufallsfolge[1] von Zahlen erzeugt. Am günstigsten geschieht dies mittels eines einfachen Polynoms $f(X)$, z.B. $f(X) = X^2 + a$ mit $a \neq 0, -2$ [2]. Ausgehend von einer Zahl x_1 definiert man rekursiv für $i \geq 2$ eine Folge von Zahlen:

$$x_i := f(x_{i-1}) \bmod n$$

Diese Zahlen werden ziemlich zufällig über den Bereich 0 bis $n-1$ verteilt. Hat n einen Primteiler p, so liegen die Zahlen

$$y_i := x_i \bmod p$$

nur im Bereich 0 bis $p-1$, so daß es irgendwann einmal dazu kommen wird, daß sich die Zahlen y_i wiederholen, d.h. daß es $i_1 < i_2$ gibt, mit $y_{i_1} = y_{i_2}$. Dann gilt aber für $l := i_2 - i_1$ und alle $i \geq i_1$, daß $y_{i+l} = y_i$, d.h. die Werte von y_i wiederholen sich ab $i = i_1$ mit der Periodenlänge l.[3] Ist aber $y_i = y_{i+l}$, so ist auch

Abbildung 2

[1] Es handelt sich dabei eigentlich um eine sogenannte Pseudo–Zufallsfolge. Eine kurze Einführung in die Theorie der Zufallsfolgen wird in [Forster, S. 72] gegeben.

[2] Mit $a = 0$ bzw. $a = -2$ wird keine echte Zufallsfolge erzeugt (vgl. [Forster, S. 108]).

[3] Die Abbildung 2 gibt eine bildliche Darstellung der y_i. Sie erklärt insbesondere, warum der Algorithmus nach dem griechischen Buchstaben ρ benannt ist.

$x_i \equiv x_{i+l} \equiv x_{i+2l} \equiv \ldots \bmod p$, d.h. $d := \mathrm{ggT}(x_i - x_{i+l}, n) \neq 1$. Im Normalfall ist $x_i \neq x_{i+l}$ also $d \neq n$, d.h. man hat mit d einen echten Teiler von n gefunden.

Das Problem ist, daß man weder den Beginn der Schleife weiß, noch die Periodenlänge l kennt. R. H. Brent schlug 1980 vor, von folgenden Differenzen den ggT mit n zu berechnen:

$$\begin{array}{rcl} x_1 & - & x_3 \\ x_3 & - & x_6 \\ x_3 & - & x_7 \\ x_7 & - & x_{12} \\ x_7 & - & x_{13} \\ & \vdots & \end{array}$$

allgemein also von den Differenzen

$$x_{2^k-1} - x_j, \text{ mit } 2^{k+1} - 2^{k-1} \leq j \leq 2^{k+1} - 1$$

Da die Differenz der Indizes immer nur um 1 zunimmt, wird irgendwann einmal der Fall $l \mid (2^k - 1 - j)$ auftreten. Dann befindet man sich auf der Schleife.

Der Algorithmus sieht dann folgendermaßen aus:

Algorithmus 4.1 (Pollard's Rho–Algorithmus)
Sei n eine Zahl von der ein Teiler gefunden werden soll.[1]

1. Wähle $a \in \{0, \ldots, n-1\}$ mit $a \neq 0, -2$ und ein $x_0 \in \mathbb{N}$.
2. Setze $k := 0$.
3. Berechne $x_{i+1} := x_i^2 + a \bmod n$ für $i = 2^k - 1, \ldots, 2^{k+1} - 1$.
4. Berechne $d_j := \mathrm{ggT}(x_{2^k-1} - x_j, n)$ für $j = 2^{k+1} - 2^{k-1}, \ldots, 2^{k+1} - 1$.[2] Gibt es ein j mit $d_j \neq 1, n$, so breche den Algorithmus ab, man hat einen Teiler d von n gefunden.
5. Setze $k := k + 1$ und gehe zu 3.

[1] Damit der Algorithmus nicht ins Leere läuft, sollte man n zuerst, z.B. mittels des kleinen Fermat, auf Zusammengesetztheit überprüfen.

[2] Der Algorithmus kann beschleunigt werden, indem man das Produkt von mehreren Differenzen $x_{2^{k-1}} - x_j$ bildet und erst dann den größten gemeinsamen Teiler mit n bestimmt. Dadurch spart man sich einen Großteil der ggT Berechnungen.

Es bleibt noch die Frage zu klären, wie schnell dieser Algorithmus ist. Die Wahrscheinlichkeit W, daß $y_1, \ldots, y_k$ paarweise verschieden sind, ist

$$W = \left(1 - \frac{1}{p}\right)\left(1 - \frac{2}{p}\right)\ldots\left(1 - \frac{k-1}{p}\right) \approx \left(1 - \frac{k}{2p}\right)^{k-1} \approx e^{-\frac{k(k-1)}{2p}}$$

Die beiden Näherungen lassen sich aus den folgenden beiden Tatsachen ableiten:

1. Sind $a, b \ll 1$, so ist $(1-a)(1-b) \approx (1-\frac{a+b}{2})^2$.
2. Ist $a \ll 1$, so ist $\ln(1-a) \approx -a$, d.h. $(1-a)^{k-1} = e^{\ln(1-a)(k-1)} = e^{-a(k-1)}$.

Die Wahrscheinlichkeit W ist $\frac{1}{2}$, wenn

$$\frac{k(k-1)}{2p} = \ln 2$$

Für $\frac{k}{k-1} \approx 1$ ist dies gleichbedeutend mit[1]

$$k \approx \sqrt{2p \ln 2} \approx 1.18\sqrt{p} \leq 1.18 n^{\frac{1}{4}}$$

Nach Satz 3.2 benötigt man $O((\ln n)^3)$ Bit–Operationen für die Berechnung des ggT, es ergibt sich also eine Gesamtzeit von $O((\ln n)^3 n^{\frac{1}{4}}) = O(n^{\frac{1}{4}+\epsilon})$ Bit–Operationen.

4.2 Der Faktor–Basis–Algorithmus

Der Faktor–Basis–Algorithmus basiert auf einer Idee von Kraitchik aus den 30er Jahren. Systematisiert wurde das Verfahren dann von Morrison und Brillhardt in den 70er Jahren (vgl. [MB]).

Die grundlegende Idee hinter diesem Algorithmus ist folgende: Man sucht $s, t \in \mathbb{N}$ mit $s^2 \equiv t^2 \bmod n$, aber $s \not\equiv \pm t \bmod n$. Dann gilt $s^2 - t^2 \equiv 0 \bmod n$, d.h. $(s-t)(s+t) = s^2 - t^2 = kn$ für ein $k \in \mathbb{N}$. Wegen $s \not\equiv \pm t \bmod n$ hat man mit $\mathrm{ggT}(s-t, n)$ oder auch $\mathrm{ggT}(s+t, n)$ einen nichttrivialen Teiler von n gefunden. Die Hoffnungen, solche s und t finden zu können, beruhen auf dem folgenden Satz:

[1] Dieses Wahrscheinlichkeitsproblem ist als das „Geburtstagsproblem“ bekannt: wieviele Menschen müssen in einem Raum sein, damit die Wahrscheinlichkeit, daß zwei Menschen den gleichen Geburtstag haben, größer $\frac{1}{2}$ ist. Mit $p = 365$ ergibt sich als Antwort, daß sich $k = 1.18\sqrt{365} \approx 23$ Menschen in dem Raum befinden müssen.

Satz 4.1
Sei n eine ungerade Zahl mit l verschiedenen Primteilern. Sei $\overline{b} \in (\mathbb{Z}/n\mathbb{Z})^*$. Dann besitzt $\overline{b}$ entweder keine Wurzel oder 2^l Wurzeln.

Beweis:
Sei $\overline{b} \in (\mathbb{Z}/n\mathbb{Z})^*$. Angenommen $\overline{b}$ besitzt eine Wurzel $\overline{a}_1$, d.h. es ist $\overline{a}_1^2 = \overline{b}$. Es ist zu beweisen, daß $\overline{b}$ genau 2^l verschiedene Wurzeln $\overline{a}_1, \ldots, \overline{a}_{2^l}$ besitzt. Da $\overline{a}_i\,\overline{a}_1^{-1}$ eine Wurzel von 1 ist, kann man o.B.d.A. annehmen, daß $\overline{b} = 1$.

1. Fall: $l = 1$.
   Dann ist $n = p^e$ mit einer Primzahl p. Ist $\overline{a}$ eine Wurzel von 1, dann folgt $p^e \mid (a^2 - 1) = (a+1)(a-1)$. Es existieren also $e_1, e_2, l_1, l_2 \in \mathbb{N}$, so daß

   $$\begin{aligned} a - 1 &= l_1 p^{e_1} \\ a + 1 &= l_2 p^{e_2} \end{aligned}$$

   Daraus folgt $2 = l_2 p^{e_2} - l_1 p^{e_1}$. Aus $p > 2$ folgt $e_1 = 0$ oder $e_2 = 0$, d.h. $\overline{a} = 1$ oder $\overline{a} = -1$. Es gibt also genau 2 Wurzeln.

2. Fall: $l \neq 1$:
   Ist $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$, wobei $p_1, \ldots, p_l$ paarweise verschiedene Primzahlen sind, dann folgt aus dem chinesischen Restsatz, daß

   $$\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \ldots \times \mathbb{Z}/p_l^{e_l}\mathbb{Z}$$

   Jede Wurzel von 1 ist das direkte Produkt von Wurzeln von 1 in den Ringen $\mathbb{Z}/p_i^{e_i}\mathbb{Z}$. Die 2^l verschiedenen Kombinationen von $+1$ und -1 ergeben 2^l verschiedene Wurzeln von 1 in $\mathbb{Z}/n\mathbb{Z}$.

□

Man kann also geeignete s und t nur finden, wenn n keine Primzahlpotenz ist. Diesen Fall kann man folgendermaßen seperat abhandeln:

Man überprüft für $i = 2, \ldots, [\log_2(n)]$, ob $\sqrt[i]{n}$ ein echter Teiler von n ist. Die benötigte Zeit für jede einzelne Überprüfung beträgt $O((\ln n)^3 \ln i)$ Bit–Operationen, als Gesamtzeit erhält man $O((\ln n)^4 \ln\ln n)$ Bit–Operationen (vgl. [Gerhard, S. 80]).

Im folgenden sei n eine Zahl mit mindestens zwei verschiedenen Primteilern.

Die einfachste Methode, geeignete s und t zu finden, geht auf Fermat zurück. Man bestimmt zunächst $r := [\sqrt{n}]$. Mit $i = 1, 2, \ldots$ berechnet man dann $x_i := (r+i)^2 - n$. Ist x_i ein Quadrat, d.h. ist $x_i = y_i^2$ für ein $y_i \in \mathbb{N}$, so

ist $(r+i)^2 \equiv y_i^2 \bmod n$. Diese Methode funktioniert hervorragend, wenn n das Produkt zweier nahezu gleich großer Primzahl ist. Im allgemeinen Fall benötigt man jedoch, wie beim Probedivisionen–Algorithmus, $O(n^{\frac{1}{2}+\epsilon})$ Bit–Operationen.

Einen deutlich komplizierteren, aber auch sehr viel erfolgreicheren, Weg geht die Faktor–Basis–Methode:

Definition:
Für $k \in \mathbb{N}$ sei $F_k = \{p_1, \ldots, p_k\}$ die Menge der k kleinsten Primzahlen, dabei sei $p_1 < p_2 < \ldots < p_k$. Eine Zahl b heißt F_k–Zahl, wenn $p_1, \ldots, p_k$ die einzigen Primteiler von $b^2 \bmod n$ sind.

Hat man $k+1$ verschiedene F_k–Zahlen $b_1, \ldots, b_{k+1}$ gefunden, so definiert man für $i = 1, \ldots, k+1$ jeweils

$$b_i^2 \bmod n =: \prod_{j=1}^{k} p_j^{e_{ij}}$$

sowie den Vektor[1]

$$\overline{e}_i := (\overline{e}_{i1}, \ldots, \overline{e}_{ik}) \in \mathbb{F}_2^k$$

Die Vektoren $\overline{e}_1, \ldots, \overline{e}_{k+1}$ sind linear abhängig, man kann daher eine nichttriviale Linearkombination von $\overline{e}_1, \ldots, \overline{e}_{k+1}$ der Form $\overline{\lambda}_1\overline{e}_1 + \ldots + \overline{\lambda}_{k+1}\overline{e}_{k+1} = 0$ finden.
Dann gilt mit

$$s := \prod_{i=1}^{k+1} b_i^{\lambda_i} \quad \text{und} \quad t := \prod_{j=1}^{k} p_j^{\frac{1}{2}\sum_{i=1}^{k+1}\lambda_i e_{ij}}$$

daß

$$s^2 \equiv \prod_{i=1}^{k+1} b_i^{2\lambda_i} \equiv \prod_{i=1}^{k+1}\prod_{j=1}^{k} p_j^{\lambda_i e_{ij}} \equiv \prod_{j=1}^{k} p_j^{\sum_{i=1}^{k+1}\lambda_i e_{ij}} \equiv t^2 \bmod n$$

Sofern $s \not\equiv \pm t \bmod n$, hat man mit $\mathrm{ggT}(s+t, n)$ oder auch $\mathrm{ggT}(s-t, n)$ einen echten Teiler von n gefunden.

Die einfachste einigermaßen aussichtsreiche Methode, F_k–Zahlen zu finden, ist

$$b = \left\lceil \sqrt{in} \right\rceil + j \text{ mit } i, j = 1, 2, \ldots$$

zu wählen. Denn dann ist

$$b^2 \bmod n = \left(\left\lceil \sqrt{in} \right\rceil^2 + j^2 + 2j \left\lceil \sqrt{in} \right\rceil \right) \bmod n \approx j^2 + 2j \left\lceil \sqrt{in} \right\rceil$$

D.h. $b^2 \bmod n$ ist von der Größenordnung $\sqrt{n}$, also verhältnismäßig klein, und die Wahrscheinlichkeit ist daher relativ groß, daß sich $b^2 \bmod n$ durch die k kleinsten Primzahlen faktorisieren läßt.

[1] Dies ist der Grund, warum die Menge F_k auch als Faktor–Basis bezeichnet wird.

Es wird im folgenden eine grobe Skizze für eine Zeitabschätzung des Faktor–Basis–Algorithmus gegeben.

Definition:
Eine Zahl $a \in \mathbb{N}$ heißt B–glatt, wenn a das Produkt von Primzahlen $p \leq B$ ist. Die Zahl a heißt B–potenzglatt, wenn a B–glatt ist und alle Primzahlpotenzen, welche a teilen, kleiner oder gleich B sind. Es bezeichne

$$\psi(n, B) := \sharp\{a \in \{2, \ldots, n\} \mid a \text{ ist } B\text{–glatt}\}$$

Für $n \in \mathbb{R}^+$ setze $L(n) := e^{\sqrt{\ln n \ln \ln n}}$.

Folgendes Lemma (vgl. [LL90]) ist der Spezialfall eines der wichtigsten Theoreme der analytischen Zahlentheorie (vgl. [CEP]):

Satz 4.2
Seien $a \in \mathbb{R}^+$ und $n \in \mathbb{N}$ mit $n > L(n)^a$. Dann gilt

$$\psi(n, L(n)^a) = nL(n)^{-\frac{1}{2a}+o(1)}$$

Beim Faktor–Basis–Algorithmus gilt dann $L(n)^a = p_k$. Die Geschwindigkeit des Algorithmus hängt von der geeigneten Wahl von k ab. Eine genaue Abschätzung der Geschwindigkeit des Algorithmus in Abhängigkeit von k ist nicht möglich. Greift man aber auf einige sehr plausible Vermutungen zurück, kann gezeigt werden, daß man am schnellsten arbeitet, wenn k so gewählt wird, daß $p_k \approx \sqrt{L(n)}$, bzw. $k \approx \frac{\sqrt{L(n)}}{\ln \sqrt{L(n)}}$. Es ergibt sich eine Gesamtzeit von

$$O\left(L(n)^{\sqrt{2}+\epsilon}\right) = O\left(e^{(\sqrt{2}+\epsilon)C\sqrt{\ln n \ln \ln n}}\right)$$

Bit–Operationen für das Auffinden eines Faktors mittels des Faktor–Basis–Algorithmus (vgl. [Pomerance96, S. 1477]). Diese Zeitabschätzung dominiert die Zeit, die man benötigt, um zu überprüfen, ob n eine Primzahlpotenz ist.

Der Algorithmus sieht dann folgendermaßen aus:

Algorithmus 4.2 (Faktor–Basis)
Sei n eine Zahl, von der ein Teiler gefunden werden soll.[1]

1. Überprüfe für $i = 2, \ldots, [\log_2(n)]$, ob $\sqrt[i]{n}$ ein echter Teiler von n ist. Wenn ja, dann breche den Algorithmus ab.
2. Setze $k := \left[\frac{\sqrt{L(n)}}{\ln \sqrt{L(n)}}\right]$. Sei $F_k := \{p_1, p_2, \ldots, p_k\}$ die Menge der k kleinsten Primzahlen. Wähle eine obere Schranke C.

[1] Vgl. Anmerkung zum Rho–Algorithmus.

3. Setze $l := 0$. Im folgenden bezeichne l die Anzahl der schon gefundenen F_k–Zahlen.

4. Setze $i := 1$.

5. Setze $j := 1$.

6. Setze $b_{ij} := \left[\sqrt{in}\,\right] + j$. Ist b_{ij} keine F_k–Zahl, so gehe zu 9.

7. Setze $l := l + 1$. Ist
$$b_{ij}^2 \bmod n =: \prod_{h=1}^{k} p_h^{e_{lh}}$$
für $e_{lh} \in \mathbb{N}_0$, so setze $\overline{e}_l := (\overline{e}_{l1}, \ldots, \overline{e}_{lk}) \in \mathbb{F}_2^k$. Speichere $b_l := b_{ij}$ und $\overline{e}_l$.

8. Ist $l = k + 1$, d.h. hat man $k + 1$ F_k–Zahlen gefunden, so finde z.B. mittels des Gaußschen Eliminierungsalgorithmus (vgl. [Cohen, S. 47]) eine nichttriviale Linearkombination der Form $\overline{\lambda}_1\overline{e}_1 + \ldots + \overline{\lambda}_{k+1}\overline{e}_{k+1} = 0$. Setze
$$s := \prod_{i=1}^{k+1} b_i^{\lambda_i} \quad \text{und} \quad t := \prod_{j=1}^{k} p_j^{\frac{1}{2}\sum_{i=1}^{k+1}\lambda_i e_{ij}}$$
Ist $d := \mathrm{ggT}(s + t, n) \neq n$, dann hat man einen nichttrivialen Teiler von n gefunden. Breche den Algorithmus ab. Ansonsten setze $l := 0$.

9. Ist $b_{ij}^2 \bmod n > C\sqrt{n}$, dann setze $i := i + 1$ und gehe zu 5., ansonsten setze $j := j + 1$ und gehe zu 6.

Es gibt eine ganze Reihe von leicht zu verwirklichenden Verbesserungen:

1. Es ist $b_{ij}^2 = \left(\left[\sqrt{in}\,\right] + j\right)^2 = (b_{ij-1} + 1)^2 = b_{ij-1}^2 + 2b_{ij-1} + 1$. Man kann also eine Multiplikation durch drei Additionen ersetzen.

2. Um zu überprüfen, ob b_{ij} eine F_k–Zahl ist, wird man der Reihe nach $e_1, \ldots, e_k$ bestimmen. Dabei sollte man die sogenannte „early abort strategy“ anwenden. Ist nämlich $\sqrt{b_{ij}^2 \bmod n} > \prod_{h=1}^{k/4} p_h^{e_h}$, so ist b_{ij} höchstwahrscheinlich keine F_k–Zahl; man kann sich die Bestimmung von $e_{\frac{k}{4}+1}, \ldots, e_k$ ersparen.

3. In Schritt 7 wird man nicht $l := 0$ setzen, sondern nur eine F_k–Zahl $\overline{b}_i$ mit $\overline{\lambda_i} \neq 0$ streichen und diese durch eine neue F_k–Zahl ersetzen.

4. Den Faktor C sollte man am Anfang klein wählen und dann mit der Zeit vergrößern.

Die Verbesserungen beschleunigen zwar den Algorithmus z.T. beträchtlich, ändern aber nichts an der Geschwindigkeitsabschätzung in der O–Notation. Der Faktor–Basis–Algorithmus ist mit $O\left(e^{(\sqrt{2}+\epsilon)C\sqrt{\ln n \ln\ln n}}\right)$ benötigten Bit–Operationen für große n deutlich schneller als der Rho–Algorithmus, welcher $O(e^{\frac{1}{4}(1+\epsilon)\ln n})$ Bit–Operationen benötigt. Allerdings hat der Faktor–Basis–Algorithmus, im Gegensatz zum Rho–Algorithmus, den Nachteil, daß das Aufsuchen eines kleinen Faktors genauso lange dauert, wie das Aufsuchen eines sehr großen Teilers (d.h. Teiler der Größenordnung $\sqrt{n}$).
Mit Abstand am längsten dauert beim Faktor–Basis–Algorithmus die Suche nach F_k–Zahlen. Hier setzen die verschiedenen Verbesserungen an. Die zwei bekanntesten und erfolgreichsten sind der Kettenbruch–Algorithmus (vgl. [Forster, S. 189]) und das quadratische Sieb (vgl. [Koblitz, S. 160]). Das quadratische Sieb wurde 1981 von Pomerance entwickelt. Es war mit einer Geschwindigkeit von $O(e^{(1+\epsilon)\sqrt{\ln n \ln\ln n}})$ Bit–Operationen lange Zeit der schnellste Algorithmus, bevor es 1993 vom Zahlkörpersieb überholt wurde, welches von A. K. und H. W. Lenstra entwickelt wurde. Mit Hilfe von schwierigen algebraischen Hilfsmitteln werden beim Zahlkörpersieb s, t mit $s^2 \equiv t^2 \bmod n$ gefunden. Dafür werden nach einer heuristischen Zeitabschätzung $O\left(e^{(C+\epsilon)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}}\right)$ Bit–Operationen benötigt. Für das spezielle Zahlkörpersieb, das nur auf Zahlen angewendet werden kann, welche in der Nähe einer großen Potenz liegen, ist $C = \left(\frac{32}{9}\right)^{\frac{1}{3}} \approx 1.523$; für den allgemeinen Fall ist $C = \left(\frac{64}{9}\right)^{\frac{1}{3}} \approx 1.923$. Durch eine Abwandlung des Algorithmus konnte Coppersmith den Wert C auf $C = \frac{1}{3}(92 + 26\sqrt{13})^{\frac{1}{3}} \approx 1.902$ senken. Eine ausführliche Beschreibung des Zahlkörpersiebs wird in [LL93] gegeben, die Weiterentwicklung von Coppersmith wird in [Coppersmith] dargestellt.

Die folgende Tabelle gibt für die verschiedenen Algorithmen eine Übersicht über die zu erwartenden Zeiten bei der Faktorisierung von großen Zahlen. Dabei wurden der konstante Faktor der O–Notation ebenso wenig berücksichtigt wie das „ϵ“ in den Termen.

[1] Die Geschwindigkeit des quadratischen Siebs entspricht der des Lenstra–Algorithmus, welcher in Kapitel 6.1 erläutert wird.

Größen–ordnung	Benötigte Bit–Operationen für Faktorisierungen mit dem			
	Rho–Algorithmus	Faktor–Basis–Algorithmus	quadratischen Sieb[1]	Zahlkörpersieb
10^5	$1.8 \cdot 10$	$1.8 \cdot 10^3$	$2.0 \cdot 10^2$	$2.4 \cdot 10^3$
10^{10}	$3.1 \cdot 10^2$	$1.7 \cdot 10^5$	$4.9 \cdot 10^3$	$1.1 \cdot 10^5$
10^{20}	$1.0 \cdot 10^5$	$1.4 \cdot 10^8$	$5.9 \cdot 10^5$	$1.8 \cdot 10^7$
10^{40}	$1.0 \cdot 10^{10}$	$3.4 \cdot 10^{12}$	$7.3 \cdot 10^8$	$1.5 \cdot 10^{10}$
10^{80}	$1.0 \cdot 10^{20}$	$1.1 \cdot 10^{19}$	$2.9 \cdot 10^{13}$	$1.2 \cdot 10^{14}$
10^{160}	$1.0 \cdot 10^{40}$	$4.4 \cdot 10^{28}$	$1.8 \cdot 10^{20}$	$2.2 \cdot 10^{19}$

Die Tabelle zeigt, daß sich die moderneren Algorithmen erst bei sehr großen Zahlen durchsetzen, bei Zahlen also, die zumeist außerhalb des Bereichs eines Hobbyprogrammierers liegen. In der Realität überholt das Zahlkörpersieb das quadratische Sieb etwa bei Zahlen der Größenordnung 10^{100}. Ein ausführlicherer Überblick über die Entwicklung der modernsten Faktorisierungsalgorithmen wird in [Pomerance96] gegeben.

Die Faktorisierung von RSA–Zahlen hat eine lange Geschichte. Im August 1977 publizierten die Erfinder des RSA–Algorithmus einen mit RSA codierten Text, wobei die zu faktorisierende Zahl 129 Stellen besaß. Der berühmte Buchautor Martin Gardner schätzte damals, daß die Faktorisierung 10^{15} Jahre dauern würde. Nichts zeigt deutlicher die stürmische Entwicklung der Computertechnologie und der Faktorisierungsalgorithmen als die Tatsache, daß es tatsächlich nur 17 Jahre dauerte bis der Text entschlüsselt war.[1] Ein weiterer Prüfstein für Faktorisierungsalgorithmen sind die von der RSA Data Security, Inc herausgegebenen „RSA challenge numbers“. Als bisher größte dieser Zahlen wurde im April 1996 eine 130–stellige Zahl mit Hilfe des Zahlkörpersiebs faktorisiert. Die dafür benötigte Zeit betrug nur etwa 15% der Zeit, die das quadratische Sieb benötigt hätte. Als nächstgrößere RSA–Zahl steht mit

21290246318258757547497882016271517497806703963277216278233383
21538194998405649591136657385302191831678310738799531723088956
9230873441936471

eine 140–stellige Zahl auf der Liste der „RSA challenge numbers“.[2]

Am 4. September 1997 gab das holländische CWI–Institut bekannt[3], daß

12862480745292006410890272858478665047052244641713578041943 4
68146981010437930160098645877829367241942440036761175181846 7

[1] Der Text lautete übrigens „The magic words are squeamish ossifrage“.

[2] Quelle: challenge-rsa-list@rsa.com

[3] Quelle: http://www.cwi.nl/cwi/Latest_News.html

$$
\begin{aligned}
&1309490252184868543062502487621047944788065325791142443946 93\\
=\ &7885391524799599235834738707297251587966475388837188632621 81\\
&413347391236469 \cdot\\
&\cdot 1631178452565029206875585436566970202474113644621203858931 60\\
&9458045532501874726047433264764355226803788 97
\end{aligned}
$$

Die oben angegebene Faktorisierung von $\frac{12^{167}+1}{13}$, einer 180–stelligen Zahl, erfolgte nach nur 12 Tagen Berechnung auf 85 SGI/Cray Computern mit Hilfe des speziellen Zahlkörpersiebs und stellt die größte bisher erfolgte nichttriviale Faktorisierung[4] dar.

4.3 Der $(p-1)$–Algorithmus von Pollard

Der $(p-1)$–Algorithmus wurde 1974, ebenso wie der Rho–Algorithmus, von J. Pollard entwickelt.

Satz 4.3
Sei $a \in \{1, \ldots, n-1\}$ mit $\mathrm{ggT}(a, n) = 1$. Besitzt n einen Primteiler p, und ist $k \in \mathbb{N}$ so gewählt, daß $(p-1) \mid k$, dann gilt für $f := (a^k - 1) \bmod n$, daß $d := \mathrm{ggT}(f, n) \neq 1$. D.h. entweder ist $d = n$, oder d ist ein nichttrivialer Teiler von n. Es ist $d = n$ genau dann, wenn $a^k \equiv 1 \bmod n$.

Beweis:
Wegen $(p-1) \mid k$ folgt aus dem kleinen Fermat, daß $a^k \equiv 1 \bmod p$, und daher $p \mid f$ sowie $p \mid \mathrm{ggT}(f, n)$. □

Will man aus Satz 4.3 einen Faktorisierungsalgorithmus ableiten, so hängt der Faktorisierungserfolg von der geschickten Wahl von k ab. Am besten wählt man k so, daß k möglichst viele Teiler besitzt, in der Hoffnung, daß es einen Primteiler p gibt, so daß $p-1$ unter den Teilern von k ist. Eine Zahl k hat besonders viele verschiedene Teiler, wenn man sie ein Produkt von relativ kleinen Primzahlen ist. In der Praxis wird deshalb normalerweise

$$k(B) := \prod_{p \leq B, p \text{ prim}} p^{\left[\frac{\ln B}{\ln p}\right]} = \mathrm{kgV}((\,;1), 2, \ldots, [B])$$

mit einem geeigneten B gewählt.

Der Algorithmus sieht folgendermaßen aus:

Algorithmus 4.3 (Pollard's $(p-1)$–Algorithmus)
Sei n eine zu faktorisierende Zahl.[1]

1. Setze $B := 10$.

[4]Unter einer nichttrivialen Faktorisierung soll hier die Faktorisierung einer Zahl verstanden sein, welche keinen kleinen Primfaktor besitzt.

[1]Vgl. Anmerkung zum Rho–Algorithmus.

2. Wähle ein $a \in \{2, \ldots, n-2\}$. Ist $\mathrm{ggT}(a, n) \neq 1$, so hat man einen Teiler von n gefunden. Breche den Algorithmus ab.

3. Setze $f := a^k - 1 \bmod n$ mit $k := \prod_{p \leq B, p \text{ prim}} p^{\left\lfloor \frac{\ln B}{\ln p} \right\rfloor}$.[2]

4. Berechne $d := \mathrm{ggT}(f, n)$. Ist d ein nichttrivialer Teiler von n so breche den Algorithmus ab.

5. Ist $B < \sqrt{n}$, so setze $B := B \cdot 10$ und gehe zu 2.

[2]Um k nicht berechnen zu müssen, setzt man in der Praxis

$$f := \left(\ldots \left(a^{p_1^{\left\lfloor \frac{\ln B}{\ln p_1} \right\rfloor}} \right) \ldots \right)^{p_l^{\left\lfloor \frac{\ln B}{\ln p_l} \right\rfloor}} - 1 \bmod n$$

wobei $\{p_1, \ldots, p_l\}$ die Menge aller Primzahlen kleiner oder gleich B bezeichne.

Für eine Geschwindigkeitsabschätzung ist folgender Satz grundlegend:

Satz 4.4
Setzt man $k := \prod_{p \leq B, p \text{ prim}} p^{\left[\frac{\ln B}{\ln p}\right]}$ für ein $B \in \mathbb{N}$ und ist $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$, so werden $O(B(\ln n)^2)$ Bit-Operationen für die Berechnung von $\overline{a}^{\,k}$ benötigt.

Beweis:
Mit der Methode des sukzessiven Quadrierens (vgl. Algorithmus 3.4) benötigt man maximal $2[\ln k + 1]$ Multiplikationen, dabei ist

$$\begin{array}{rcccl} \ln k & = & \ln\left(\prod_{p \leq B, p \text{ prim}} p^{\left[\frac{\ln B}{\ln p}\right]}\right) & = & \sum_{p \leq B, p \text{ prim}} \left[\frac{\ln B}{\ln p}\right] \ln p \\ & \approx & \sum_{p \leq B, p \text{ prim}} \ln B & \approx & \frac{B}{\ln B} \ln B \; = \; B \end{array}$$

Die vorletzte Gleichheit folgt aus dem Primzahlsatz (Satz 2.9). Da man für eine Multiplikation $O((\ln n)^2)$ Bit–Operationen benötigt, erhält man eine Gesamtzeit von $O((\ln n)^2) \cdot B = O(B(\ln n)^2)$ Bit-Operationen. □

Im ungünstigsten Fall ist $n = pq$ eine RSA–Zahl. Die Wahrscheinlichkeit, daß $p - 1 \approx \sqrt{n}$ keinen Primteiler größer $n^{\frac{1}{4}}$ besitzt, ist etwa 30%.[1] Mit $B = n^{\frac{1}{4}}$ benötigt man $O((\ln n)^2 n^{\frac{1}{4}})$ Bit–Operationen für die Berechnung von $\overline{a}^{\,k(B)}$ und für die Wahrscheinlichkeit W, daß $\mathrm{ggT}(a^{k(B)} - 1, n) \neq 1$, gilt $W \approx 1 - (1 - 0.3)^2 = 0.51$.

Der Algorithmus wird also nur dann sehr schnell sein, wenn n kleine Primteiler besitzt oder wenn man Glück hat, und $p - 1$ glatt ist.[2]

Analog zum allgemeinen Primzahltestsatz aus Kapitel 2.4 kann, basierend auf Satz 4.3, ein allgemeiner Faktorisierungssatz formuliert werden:

Satz 4.5
Sei G_n eine n–Pseudogruppe mit neutralem Element $O = (\overline{o}_1, \ldots, \overline{o}_s)$, und $A \in G_n$. Angenommen n besitzt einen Primteiler p, und $k \in \mathbb{N}$ ist so gewählt, daß $\sharp(G_p) \mid k$. Ist $k \odot A$ definiert, dann gilt mit $(\overline{b}_1, \ldots, \overline{b}_s) := k \odot A$, daß $(\overline{b}_1 - \overline{o}_1, \ldots, \overline{b}_s - \overline{o}_s) \in p(\mathbb{Z}/n\mathbb{Z})^s$.

Beweis:
Aus $\sharp(G_p) \mid k$ folgt $k \odot_p A_p = O_p$. Es ist $(k \odot A)_p = k \odot_p A_p$, d.h. entweder ist $k \odot A$ nicht definiert, oder es ist $(k \odot A)_p = O_p$. □

[1] Vergleiche Abbildung 6.

[2] Eine Zahl wird in der mathematischen Umgangsprache als glatt bezeichnet, wenn sie keinen großen Primteiler besitzt.

Ist $k \odot A$ definiert, so ist $d := \mathrm{ggT}(o_1 - b_1, n) \neq 1$, d.h. man hat, falls $d \neq n$, einen nichttrivialen Teiler von n gefunden. Ist $k \odot A$ hingegen nicht definiert, so liegt dies daran, daß man bei der Berechnung von $k \odot A$ eine Addition $C \oplus D$ durchführt, welche nicht definiert ist. Dann hat man aber mit $\theta(C, D)$ ebenfalls einen nichttrivialen Teiler von n gefunden. Hat man also k so gewählt, daß $k \mid G_p$, so findet man mit der Berechnung von $k \odot A$ bzw. mit $\mathrm{ggT}(o_1 - b_1, n)$ fast immer einen nichttrivialen Teiler von n.

Satz 4.3 erhält man als Korollar aus Satz 4.5, indem man $G_n := (\mathbb{Z}/n\mathbb{Z})^*$ als n–Pseudogruppe wählt. Ist $n - 1 = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ mit paarweise verschiedenen Primzahlen $p_1, \ldots, p_l$, so besagt der Satz also, daß der $(p-1)$–Algorithmus dann zu langsam ist, wenn keine der Gruppenordnungen von $(\mathbb{Z}/p\mathbb{Z})^*$ glatt ist. Denn diese Gruppen sind die einzigen, die für G_p in Frage kommen. Eine weitere Anwendungsmöglichkeit ist, $G_n := \{\zeta \in \mathbb{F}_{n^2}^* \mid \zeta^{n+1} = 1\}$ zu wählen. Dies ist der sogenannte $(p+1)$–Algorithmus (vgl. [Forster, S. 148]). Den Durchbruch schafft dieser Satz allerdings durch den Übergang zu den elliptischen Pseudokurven. Für das Verständnis dieser Anwendung von Satz 4.5 müssen jedoch einige Grundtatsachen über elliptische Kurven eingeführt werden. Dies erfolgt im nächsten Kapitel.

5 Elliptische Kurven

5.1 Grundlagen

Es existiert eine vielfältige Literatur zum Thema elliptische Kurven[1], da diese auf Grund ihrer Gruppeneigenschaft eine herausragende Stellung unter den algebraischen Kurven einnehmen. Eine sehr gute Einführung in die Theorie der elliptischen Kurven wird in [Silverman] gegeben. In diesem Buch werden elliptische Kurven als ein Paar (E, O) definiert, wobei E eine Kurve von Geschlecht 1 und $O \in E$ ist. Für die Zwecke dieser Arbeit ist dies eine zu allgemeine Definition, zudem würde die Einführung der für diese Definition benötigten Theorie den Umfang dieser Arbeit sprengen. Deswegen wird im weiteren mit der folgenden, elementareren Definition gearbeitet:

Definition:
Sei $\overline{K}$ ein algebraisch abgeschlossener Körper mit Char $\overline{K} > 3$. Seien $a, b \in \overline{K}$ mit der Eigenschaft, daß $4a^3 + 27b^2 \neq 0$ [2]. Die Menge

$$E := E\left(\overline{K}\right) := E_{a,b}\left(\overline{K}\right) := \{(x : y : z) \in \mathbb{P}^2\left(\overline{K}\right) \mid y^2z = x^3 + axz^2 + bz^3\}$$

wird als projektive elliptische Kurve bezeichnet.[3] Man sagt, E ist durch die Weierstrass-Gleichung $Y^2Z = X^3 + aXZ^2 + bZ^3$ gegeben. Die Punktmenge

$$E^{aff} := E^{aff}\left(\overline{K}\right) := E^{aff}_{a,b}\left(\overline{K}\right) := \{(x, y) \in \overline{K}^2 \mid y^2 = x^3 + ax + b\}$$

wird als affine elliptische Kurve, welche durch die Weierstrass–Gleichung $Y^2 = X^3 + aX + b$ gegeben ist, bezeichnet.

Setzt man $g_\infty := \{(x : y : z) \in \mathbb{P}^2\left(\overline{K}\right) \mid z = 0\}$, die sogenannte unendlich ferne Gerade, so sieht man, daß jede elliptische Kurve nur einen Punkt in g_∞ besitzt, nämlich $O := (0 : 1 : 0)$ [4], den sogenannten unendlich fernen Punkt. Mittels der bijektiven Abbildung

$$\begin{array}{rccl} \pi: & E_{a,b}\left(\overline{K}\right) & \to & E^{aff}_{a,b}\left(\overline{K}\right) \cup \{O\} \\ & (x : y : z) & \mapsto & (\frac{x}{z}, \frac{y}{z}) \text{ falls } z \neq 0 \\ & (0 : 1 : 0) & \mapsto & O \end{array}$$

[1] Die elliptischen Kurven verdanken ihren Namen den elliptischen Integralen aus der Funktionentheorie, welche wiederum bei der Berechnung der Bogenlänge einer Ellipse auftreten. Der genauere Zusammenhang kann in [Silverman, S. 146ff] nachgelesen werden.

[2] Die Einschränkung auf den Fall $4a^3+27b^2 \neq 0$ wird beim Beweis zu Satz 5.2 ersichtlich.

[3] Mit der Beschränkung auf diese Definition verliert man nicht viel gegenüber der allgemeineren Definition der elliptischen Kurve, welche in [Silverman] gegeben wird, denn jede elliptische Kurve in der allgemeinen Definition über einem algebraisch abgeschlossenen Körper $\overline{K}$ mit Char $\overline{K} > 3$ ist isomorph zu einer solchen elliptischen Kurve. Genauer formuliert und bewiesen wird dies im Buch von Silverman (vgl. [Silverman, S. 63]).

[4] Denn ist ein Punkt $(x : y : 0) \in E$ so folgt $x^3 = 0$, d.h. $x = 0$.

wird im weiteren $E_{a,b}\left(\overline{K}\right)$ mit $E_{a,b}^{aff}\left(\overline{K}\right) \cup \{O\}$ identifiziert. Analog wird im folgenden mittels der Abbildung

$$\begin{array}{rccc} \pi: & \mathbb{P}^2\left(\overline{K}\right) \setminus g_\infty & \to & \overline{K}^2 \\ & (x:y:z) & \mapsto & (\frac{x}{z}, \frac{y}{z}) \end{array}$$

$\mathbb{P}^2\left(\overline{K}\right) \setminus g_\infty$ mit $\overline{K}^2$ identifiziert.

Für den Rest des Kapitels bezeichne $\overline{K}$ einen algebraisch abgeschlossenen Körper mit $\operatorname{Char}\overline{K} > 3$ und $K \subseteq \overline{K}$ einen Teilkörper von $\overline{K}$.

Definition:
Sind $a, b \in K$, so sagt man, $E_{a,b}\left(\overline{K}\right)$ ist über K definiert. Zudem wird

$$E(K) := E_{a,b}(K) := E_{a,b}\left(\overline{K}\right) \cap \mathbb{P}^2(K)$$

als die Menge der K–rationalen Punkte der elliptischen Kurve $E\left(\overline{K}\right)$ bezeichnet. Es ist

$$E(K) = \{(x, y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \{O\}$$

Eine Gerade schneidet eine elliptische Kurve im Normalfall in drei verschiedenen Punkten, sollte dies nicht der Fall sein, dann liegt das daran, daß zwei oder drei Punkte zusammenfallen. Man sagt dann, ein solcher Schnittpunkt besitzt Schnittmultiplizität 2 bzw. 3. Genauer:

Satz 5.1
Jede (projektive) Gerade schneidet eine elliptische Kurve in 3 Punkten (mit Schnittmultiplizitäten[1] gezählt).

Beweis:
Sei $Y^2Z = X^3 + aXZ^2 + bZ^3$ die Weierstrass–Gleichung einer elliptischen Kurve und $dX + eY + fZ = 0$ die Gleichung einer (projektiven) Geraden, d.h. $(d, e, f) \neq (0, 0, 0)$.

1. Fall: $e = 0$.
   Ist $d = 0$, so lautet die Geradengleichung $Z = 0$, d.h. g ist die unendlich ferne Gerade. Dann ist $O = (0 : 1 : 0)$ der einzige Schnittpunkt. Er besitzt die Schnittmultiplizität 3. Sei im folgenden $d \neq 0$.
   Setze $g := -\frac{f}{d}$. Dann ist $X = gZ$; eingesetzt in die Weierstrass–Gleichung ergibt sich $Y^2Z = g^3Z^3 + agZ^3 + bZ^3$. Diese Gleichung

[1] Zur allgemeinen Definition der Schnittmultiplizität siehe [Kunz95, S. 38].

erfüllen der unendlich ferne Punkt $O = (0 : 1 : 0)$ sowie die Punkte $(gz : y : z)$ mit $z \neq 0$ und $\left(\frac{y}{z}\right)^2 = g^3 + ag + b$. Da $\overline{K}$ algebraisch abgeschlossen ist, existiert eine Wurzel $h \in \overline{K}$ von $g^3 + ag + b$. Man erhält die beiden Schnittpunkte $P_1 = (g : h : 1)$ und $P_2 = (g : -h : 1)$. Ist $P_1 = P_2$, dann besitzt dieser Punkt die Schnittmultiplizität 2.

2. Fall: $e \neq 0$.
In diesem Fall erfüllt der unendlich ferne Punkt die Geradengleichung nicht. Der Fall kann also im Affinen betrachtet werden. Die Geradengleichung lautet dann $Y = g + hX$ mit $g := -\frac{f}{e}$ und $h := -\frac{d}{e}$. Für die Schnittpunkte gilt die folgende Bedingung:

$$(g + hX)^2 - X^3 - aX - b = Y^2 - X^3 - aX - b = 0$$

Da $\overline{K}$ algebraisch abgeschlossen ist, gibt es $x_1, x_2, x_3 \in \overline{K}$ mit

$$(g + hX)^2 - X^3 - aX - b = (X - x_1)(X - x_2)(X - x_3)$$

Man erhält drei Schnittpunkte $P_1 = (x_1, g + hx_1)$, $P_2 = (x_2, g + hx_2)$, $P_3 = (x_3, g + hx_3)$. Jedoch können 2 Punkte oder sogar alle 3 identisch sein. Die Gerade und die Kurve in diesem Punkt schneiden sich in diesem Punkt mit Schnittmultiplizität 2 bzw. 3.

□

Definition:
Sei $E_{a,b}\left(\overline{K}\right)$ eine elliptische Kurve und $P \in E_{a,b}\left(\overline{K}\right)$. Eine Gerade t durch P, welche im Punkt P eine Schnittmultiplizität ungleich 1 besitzt, heißt Tangente an $E\left(\overline{K}\right)$ im Punkt P.

In Satz 5.2 wird bewiesen, daß eine elliptische Kurve in einem Punkt P genau eine Tangente besitzt. Ab sofort wird deshalb von *der* Tangente im Punkt P gesprochen.

Konvention:
Ist E eine elliptische Kurve und sind $P, Q \in E$, so bezeichne $g(P, Q)$ die Gerade durch P und Q, falls $P \neq Q$, und die Tangente in P, falls $P = Q$. Schneidet eine Gerade g die Kurve E in den Punkten $P_1, \ldots, P_l$ ($l \in \{1, 2, 3\}$) mit den Schnittmultiplizitäten $s_1, \ldots, s_l$ ($s_i \in \{1, 2, 3\}$) so sagt man, g schneidet E in den Punkten $\underbrace{P_1, \ldots, P_1}_{s_1\text{-mal}} \ldots \underbrace{P_l, \ldots, P_l}_{s_l\text{-mal}}$.[1]

[1] Satz 5.1 besagt, daß $\sum_{i=1}^{l} s_i = 3$.

Jetzt verfügt man über die nötigen Grundlagen, um eine Verknüpfung auf den elliptischen Kurven zu definieren:

Definition:
Auf einer elliptischen Kurve $E\left(\overline{K}\right)$ werde folgendermaßen eine Verknüpfung

$$\begin{aligned} + : E_{a,b}\left(\overline{K}\right) \times E_{a,b}\left(\overline{K}\right) &\rightarrow E_{a,b}\left(\overline{K}\right) \\ (P, Q) &\mapsto P + Q \end{aligned}$$

definiert: Seien $P, Q \in E\left(\overline{K}\right)$. Die Gerade $g(P, Q)$ schneidet die elliptische Kurve in den Punkten P, Q, R (mit Schnittmultiplizitäten gezählt). Die Gerade $g(R, O)$ [1] schneidet die elliptische Kurve in den Punkten O, R, S (mit Schnittmultiplizitäten gezählt). Setze $P + Q := S$.

Abbildung 3: Addition der Punkte P und Q auf den $\mathbb{R}$–rationalen Punkten der affinen elliptischen Kurve, welche durch die Weierstrass–Gleichung $Y^2 = X^3 - X$ gegeben ist. Den unendlich fernen Punkt muß man sich als unendlich weit „oben“ bzw. „unten“ vorstellen.

Um die Verknüpfung durch Formeln auszudrücken, benötigt man die Tangentengleichung für einen beliebigen Punkt auf der Kurve. Für deren Er-

[1] Die Wahl des Punktes O für die Definition der Addition ist willkürlich, tatsächlich kann jeder Punkt auf der Kurve für die Definition der Addition verwendet werden. Nach [Kunz95, S. 89] sind die daraus entstehenden Gruppen sogar isomorph, allerdings läßt sich mit dem Punkt O die Addition am einfachsten durch Formeln beschreiben (vgl. Satz 5.3).

mittelung benötigt man ein algebraisches Hilfsmittel, nämlich die formale Ableitung in Polynomringen:

Definition:
Sei R ein Integritätsring und $R[X]$ der Polynomring in der Unbestimmten X über dem Ring R. Sei $f \in R[X]$. Dann läßt sich f folgendermaßen darstellen:

$$f = \sum_{j=0}^{k} a_j X^j$$

mit $a_j \in R$. Die partielle Ableitung von f nach X ist dann wie folgt definiert:

$$\frac{\partial f}{\partial X} := \sum_{j=1}^{k} j a_j X^{j-1}$$

Die Analysis legt die Vermutung nahe, daß die Steigung α der Tangente durch „ $\alpha = \frac{\frac{\partial f(X,Y)}{\partial Y}}{\frac{\partial f(X,Y)}{\partial X}}$ “ gegeben ist. Tatsächlich gilt der folgende Satz:

Satz 5.2
Sei $E_{a,b}\left(\overline{K}\right)$ eine elliptische Kurve und $P = (x : y : z) \in E_{a,b}\left(\overline{K}\right)$.

1. Setzt man

   $$f := f(X, Y, Z) := Y^2 Z - X^3 - aXZ^2 - bZ^3$$

   so lautet die Gleichung der Tangente im Punkt P folgendermaßen:

   $$\begin{aligned} \frac{\partial f}{\partial X}(x,y,z) \cdot X + \frac{\partial f}{\partial Y}(x,y,z) \cdot Y + \frac{\partial f}{\partial Z}(x,y,z) \cdot Z &= 0 \text{ d.h.} \\ (-3x^2 - az^2)X + 2yzY + (y^2 - 2axz - 3bz^2)Z &= 0 \end{aligned} \quad (1)$$

2. Ist $P = (x, y) \neq O$, und setzt man

   $$f := f(X, Y) := Y^2 - X^3 - aX - b$$

   so lautet die Gleichung der Tangente im Punkt P folgendermaßen:

   $$\begin{aligned} \frac{\partial f}{\partial X}(x,y) \cdot (X - x) + \frac{\partial f}{\partial Y}(x,y) \cdot (Y - y) &= 0 \text{ d.h.} \\ (-3x^2 - a)(X - x) + 2y(Y - y) &= 0 \end{aligned} \quad (2)$$

Insbesondere besitzt die Kurve genau eine Tangente im Punkt P.

Der Satz gilt allgemeiner für algebraische Kurven die durch eine Gleichung $f(X, Y) = 0$ mit $f(X, Y) \in \overline{K}[X, Y]$ gegeben sind. Der allgemeine Fall kann in [Kunz95, S. 52] nachgelesen werden.

Beweis:
Nach dem Beweis zu Satz 5.1 ist die Gerade welche durch die Gleichung $Z = 0$ gegeben ist, die einzige Gerade durch O mit Schnittmultiplizität größer 1. Sie ist also die einzige Tangente im Punkt O. Sei im folgenden $P = (x, y) = (x : y : 1) \in E_{a,b}\left(\overline{K}\right) \setminus \{O\}$. Wie man sich leicht vergewissern kann, sind die angegebenen Tangentengleichungen für den projektiven bzw. den affinen Fall unter der oben angegebenen Identifizierung von $\mathbb{P}^2\left(\overline{K}\right) \setminus g_\infty$ mit $\overline{K}^2$, äquivalent. Es genügt also, jeweils einen der beiden Fälle zu zeigen.

1. Fall: $y = 0$:
   Ist h eine Wurzel von $x^3 + ax + b$, so schneidet nach dem 1. Fall des Beweises zu Satz 5.1 die Gerade $X = xZ$ die Kurve in den Punkten O, $P_1 = (x : h : 1)$ und $P_2 = (x : -h : 1)$. Nachdem $h^2 = y^2 = 0$ folgt, daß $P_1 = P_2 = P$, d.h. $P = (x : 0 : 1)$ besitzt die Schnittmultiplizität 2. Die Gerade $X = xZ$ ist also eine Tangente im Punkt P.
   Angenommen es gibt noch eine weitere Tangente $\tilde{t}$ im Punkt P. Diese muß dann von der Form $\tilde{t} : Y = \alpha(X - x)$ sein. Nach dem 2. Fall des Beweises zu Satz 5.1 folgt dann, daß $(\alpha(X - x))^2 - X^3 - aX - b$ und damit auch $-X^3 - aX - b$, eine doppelte Nullstelle bei $X = x$ besitzt. Dies ist aber ein Widerspruch zu $4a^3 + 27b^2 \neq 0$, denn nach [Kunz94, S. 19] besagt diese Bedingung, daß $X^3 + aX + b$ keine mehrfachen Nullstellen besitzt.

2. Fall: $y \neq 0$:
   Nach dem 1. Fall des Beweises zu Satz 5.1 kann die Gerade $X = xZ$ nicht die Tangente im Punkt P sein. Eine Tangente muß also von der Form $Y = \alpha(X - x) + y$ sein. Dabei ist an α die Bedingung gestellt, daß $(\alpha(X - x) + y)^2 - X^3 - aX - b$ eine doppelte Nullstelle bei $X = x$ haben muß. Es ist

$$\begin{aligned}((\alpha(X - x) + y)^2 - X^3 - aX - b) : (X - x)^2 = \\ = X + \alpha^2 + 2x + \frac{X(a - 2\alpha y + 3x^2) + 2\alpha yx - 3x^3 - ax}{(X - x)^2}\end{aligned}$$

   Man erhält die Bedingungen

$$\begin{aligned} a - 2\alpha y + 3x^2 &= 0 \\ 2\alpha yx - 3x^3 - ax &= 0 \end{aligned}$$

   Es folgt $\alpha = \frac{3x^2+a}{2y}$. Die Gleichung der einzigen Tangenten im Punkt $P = (x, y)$ lautet also $Y = \frac{3x^2+a}{2y}(X - x) + y$.

Die in diesem Beweis gegebene Beschreibung der Tangente ist, wie man leicht zeigen kann, äquivalent mit der Beschreibung im Satz. □

Die Verknüpfung „ $+$ " kann nun folgendermaßen in Formeln gefaßt werden:

Satz 5.3
Es sei $E\left(\overline{K}\right)$ eine elliptischen Kurve. Die Verknüpfung

$$\begin{array}{rcl} + : E\left(\overline{K}\right) \times E\left(\overline{K}\right) & \to & E\left(\overline{K}\right) \\ (P, Q) & \mapsto & P + Q \end{array}$$

kann wie folgt beschrieben werden:

1. Ist $P = O$, dann ist $P + Q = Q$.
2. Ist $Q = O$, dann ist $P + Q = P$.

Seien im folgenden $P, Q \neq O$, und $P =: (x_P, y_P)$ sowie $Q =: (x_Q, y_Q)$.

(3) Ist $(x_P, y_P) = (x_Q, -y_Q)$, dann ist $P + Q = O$.

(4) Ist $(x_P, y_P) \neq (x_Q, -y_Q)$ und $(x_P, y_P) \neq (x_Q, y_Q)$, dann gilt mit

$$\begin{array}{rcl} x_{PQ} & := & \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q \\ y_{PQ} & := & -y_P + \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_{PQ}) \end{array} \tag{3}$$

daß $P + Q = (x_{PQ}, y_{PQ})$.

(5) Ist $(x_P, y_P) \neq (x_Q, -y_Q)$ und $(x_P, y_P) = (x_Q, y_Q)$, dann gilt mit

$$\begin{array}{rcl} x_{PQ} & := & \left(\dfrac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \\ y_{PQ} & := & -y_P + \left(\dfrac{3x_P^2 + a}{2y_P}\right)(x_P - x_{PQ}) \end{array} \tag{4}$$

daß $P + P = (x_{PQ}, y_{PQ})$.

Beweis:
Es bezeichnen R, S dieselben Punkte wie in der Definition der Verknüpfung. Ist $R = (x_R, y_R) \neq O$, so ist die Gerade $g(R, O)$ durch die Geradengleichung $X = x_R Z$ gegeben, und der 3. Schnittpunkt ist $(x_R, -y_R) = S$.
Der Fall $P = Q = O$ folgt aus der Tatsache, daß die Tangente an O die unendlich ferne Gerade ist, welche im Punkt O Schnittmultiplizität 3 besitzt.

Sei $P = O$ und $Q = (x_Q, y_Q) \neq O$. Dann ist $R = (x_Q, -y_Q)$ und daher $S = (x_R, -x_R) = (x_Q, y_Q) = Q$. Der 2. Fall folgt analog. Für den 3. Fall gilt $S = O$, und $R = O$. Es gilt also nur noch die Fälle 4 und 5 zu beweisen.
Wie man sich leicht vergewissern kann, ist die Gerade $g(P, Q)$ nicht von der Form $X = xZ$ für ein $x \in \overline{K}$.[1] Die beiden Fälle können also im Affinen betrachtet werden. Es sei $Y = \alpha X + \beta$ die Gleichung der Geraden $g(P, Q)$, dabei ist $\beta = y_P - \alpha x_P$. Ein Punkt, der sowohl auf der Geraden als auch auf der elliptischen Kurve liegt, muß die Gleichung $(\alpha X + \beta)^2 = X^3 + aX + b$ erfüllen. Man hat also ein Polynom vom Grad 3, von dem man schon zwei Nullstellen, nämlich x_P und x_Q, kennt. Allgemein gilt für normierte Polynome, daß der zweithöchste Koeffizient die Summe der Nullstellen ist. In diesem Fall gilt also $\alpha^2 = x_P + x_Q + x_R$, d.h. $x_R = \alpha^2 - x_P - x_Q$ und $y_R = \alpha x_R + \beta = \alpha x_R + y_P - \alpha x_P = y_P - \alpha(x_P - x_R)$. Damit können die Fälle 4 und 5 bewiesen werden:

(4) Fall $P \neq Q$:
Es ist dann $\alpha = \frac{y_Q - y_P}{x_Q - x_P}$. Damit gilt für $P+Q = (x_{PQ}, y_{PQ}) = (x_R, -y_R)$, daß

$$\begin{aligned} x_{PQ} &= \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q \\ y_{PQ} &= -y_P + \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_{PQ}) \end{aligned}$$

(5) Fall $P = Q$:
Es ist α die Steigung der Tangenten der elliptischen Kurve im Punkt P. Aus der Tangentengleichung (Formel 2) folgt, daß

$$\alpha = \frac{3x_P^2 + a}{2y_P}$$

Damit erhält man für $P + P = (x_{PQ}, y_{PQ}) = (x_S, y_S) = (x_R, -y_R)$, daß gilt

$$\begin{aligned} x_{PQ} &= \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \\ y_{PQ} &= -y_P + \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_{PQ}) \end{aligned}$$

□

[1] Für den 4. Fall folgt dies aus $x_P \neq x_Q$, und für den 5. Fall folgt dies aus $y_P \neq 0$ (vgl. Gleichung 1).

Satz 5.4
$(E\left(\overline{K}\right), +)$ ist eine abelsche Gruppe mit dem unendlich fernen Punkt O als neutralem Element.

Beweis:
Das neutrale Element ist nach den Fällen 1 und 2 des Satzes 5.3 der unendlich ferne Punkt O. Ist $P = (x, y) \in E\left(\overline{K}\right) \setminus \{O\}$, so ist das Negative durch $-P := (x, -y) \in E\left(\overline{K}\right)$ gegeben. Die Kommutativität der Verknüpfung folgt aus der Symmetrie der Definition der Addition.
Es bleibt also nur noch die Assoziativität der Verknüpfung zu zeigen, dies ist allerdings ein nichttriviales Unterfangen. Das Assoziativgesetz kann mit verschiedenen Methode bewiesen werden, nämlich

1. mittels geometrischer und algebraischer Argumente (vgl. [Kunz95, S. 87]),
2. mit Hilfe des Satzes von Riemann–Roch (vgl. [Silverman, S. 66]),
3. oder direkt aus der Beschreibung der Addition in Satz 5.3.

In dieser Arbeit wird die 3. Methode zum Beweis des Assoziativgesetzes verwendet. Der Beweis ist allerdings sehr langwierig, ihm ist deshalb das ganze Kapitel 8 gewidmet. □

Konvention:
Ist $p > 3$ eine Primzahl und sind $\overline{a}, \overline{b} \in \mathfrak{p}$ mit $4\overline{a}^3 + 27\overline{b}^2 \neq 0$, so soll $E_{\overline{a},\overline{b}}(\mathfrak{p})$ als die Menge der $\mathfrak{p}$–rationalen Punkte der elliptischen Kurve $E_{\overline{a},\overline{b}}(\overline{\mathfrak{p}})$ verstanden werden, wobei $\overline{\mathfrak{p}}$ ein beliebiger algebraischer Abschluß von $\mathfrak{p}$ ist. Ein solcher algebraischer Abschluß existiert nach dem Satz von Steinitz [Kunz94, S. 98]. Wie man sich leicht vergewissern kann, ist die Definition von $E_{\overline{a},\overline{b}}(\mathfrak{p})$ unabhängig von der Wahl des algebraischen Abschluß.

Die K-rationalen Punkte einer elliptischen Kurve $E\left(\overline{K}\right)$ bilden eine Untergruppe, denn wegen der multiplikativen und additiven Abgeschlossenheit eines Teilkörpers gilt nach Satz 5.3, daß

$$+(E(K) \times E(K)) \subset E(K)$$

Insbesondere ist also für jede Primzahl $p > 3$ und alle $\overline{a}, \overline{b} \in \mathfrak{p}$ mit $4\overline{a}^3 + 27\overline{b}^2 \neq 0$ auch $E_{\overline{a},\overline{b}}(\mathfrak{p})$ eine abelsche Gruppe. Für diese Gruppen gilt der folgende wichtige Satz:

Satz 5.5 (Theorem von Hasse)
Sei p eine Primzahl und $E(\mathfrak{p})$ die Menge der $\mathfrak{p}$-rationalen Punkte einer elliptische Kurve, dann gilt:

$$| \sharp E(\mathfrak{p}) - (p+1) | \leq 2\sqrt{p}$$

Der Satz gilt etwas allgemeiner auch für beliebige endliche Körper; der Beweis des allgemeinen Satzes kann in [Silverman, S. 131] nachgelesen werden. Zudem gilt der folgende Satz (vgl. [Deuring]):

Satz 5.6
Sei $p > 3$ eine Primzahl. Dann gibt es zu jedem $m \in \mathbb{N}$ mit

$$\mid m - (p+1) \mid \leq 2\sqrt{p}$$

Elemente $\overline{a}, \overline{b} \in \mathfrak{p}$ mit $4\overline{a}^3 + 27\overline{b}^2 \neq 0$, so daß $\sharp E_{\overline{a},\overline{b}}(\mathfrak{p}) = m$.

Nach heutigen Erkenntnissen kann man davon ausgehen, daß für eine Primzahl $p > 3$ die Werte $\sharp E_{\overline{a},\overline{b}}(\mathfrak{p})$ im Intervall $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$ einigermaßen gleich verteilt sind. Eine genauere Schilderung des heutigen Erkenntnisstandes wird in [Pomerance90, S. 40] gegeben.

5.2 Elliptische Pseudokurven

Die bisherige Herleitung diente dazu, eine weitere n–Pseudogruppe zu finden. Es bietet sich an „$E(\mathbb{Z}/n\mathbb{Z})$“ als n–Pseudogruppe zu wählen. Allerdings ist n höchstwahrscheinlich zusammengesetzt, und $E(\mathbb{Z}/n\mathbb{Z})$ ist bis jetzt gar nicht definiert. Aber man kann $E(\mathbb{Z}/n\mathbb{Z})$ ähnlich wie elliptische Kurven definieren:

Für den Rest des Kapitels seien $n \in \mathbb{N}$ mit $\mathrm{ggT}(n, 6) = 1$ und $\overline{a}, \overline{b} \in \mathbb{Z}/n\mathbb{Z}$ mit $4\overline{a}^3 + 27\overline{b}^2 \in (\mathbb{Z}/n\mathbb{Z})^*$ fest gewählt.

Definition:
Die Menge

$$E(\mathbb{Z}/n\mathbb{Z}) := E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z}) := \{(\overline{x}, \overline{y}) \in (\mathbb{Z}/n\mathbb{Z})^2 \mid \overline{y}^2 = \overline{x}^3 + \overline{a}\,\overline{x} + \overline{b}\} \cup \{O\}$$

wird als elliptische Pseudokurve bezeichnet.

Die elliptischen Pseudokurven benötigen natürlich noch eine (teilweise) Verknüpfung. Auch diese übernimmt man von den elliptischen Kurven, allerdings in der Formulierung von Satz 5.3:

Definition:
Es seien $P, Q \in E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$.

1. Ist $P = O$, dann setze $P + Q := Q$.
2. Ist $Q = O$, dann setze $P + Q := P$.

Seien im folgenden $P, Q \neq O$, und $P =: (\overline{x}_P, \overline{y}_P)$ sowie $Q =: (\overline{x}_Q, \overline{y}_Q)$.

(3) Ist $\overline{x}_P = \overline{x}_Q$ und $(\overline{y}_P - \overline{y}_Q) \in (\mathbb{Z}/n\mathbb{Z})^*$ bzw. $\overline{y}_P = -\overline{y}_Q$, dann setze $P + Q := O$.

(4) Ist $\overline{x}_P \neq \overline{x}_Q$ und $(\overline{x}_P - \overline{x}_Q) \in (\mathbb{Z}/n\mathbb{Z})^*$, dann setze

$$\begin{aligned} \overline{x}_{PQ} &:= \left(\frac{\overline{y}_Q - \overline{y}_P}{\overline{x}_Q - \overline{x}_P}\right)^2 - \overline{x}_P - \overline{x}_Q \\ \overline{y}_{PQ} &:= -\overline{y}_P + \left(\frac{\overline{y}_Q - \overline{y}_P}{\overline{x}_Q - \overline{x}_P}\right)(\overline{x}_P - \overline{x}_{PQ}) \end{aligned} \tag{5}$$

und $P + Q := (\overline{x}_{PQ}, \overline{y}_{PQ})$.

(5) Ist $\overline{x}_P = \overline{x}_Q$, $\overline{y}_P = \overline{y}_Q$, $\overline{y}_P \neq 0$ und $2\overline{y}_P \in (\mathbb{Z}/n\mathbb{Z})^*$, dann setze

$$\begin{aligned} \overline{x}_{PQ} &:= \left(\frac{3\overline{x}_P^2 + a}{2\overline{y}_P}\right)^2 - 2\overline{x}_P \\ \overline{y}_{PQ} &:= -\overline{y}_P + \left(\frac{3\overline{x}_P^2 + a}{2\overline{y}_P}\right)(\overline{x}_P - \overline{x}_{PQ}) \end{aligned} \tag{6}$$

und $P + P := (\overline{x}_{PQ}, \overline{y}_{PQ})$.

Tritt keiner der 5 Fälle auf, so ist $P + Q$ nicht definiert.

Bemerkung 5.7
Ist die Verknüpfung zweier Punkte $P = (x_P, y_P)$ und $Q = (x_Q, y_Q)$ nicht definiert, so hängt das immer damit zusammen, daß ein $\overline{a} \notin (\mathbb{Z}/n\mathbb{Z})^* \cup \{0\}$ gefunden wurde.[1] Insbesondere hat man also, wenn die Addition von P und Q nicht definiert ist, mit $d := \mathrm{ggT}(a, n)$ einen echten Teiler von n gefunden.

Man kann sich leicht vergewissern, daß es für $E(\mathbb{Z}/n\mathbb{Z})$ T und θ im Sinne der Definition der n–Pseudogruppen gibt.

Satz 5.8 (Eindeutigkeit des Inversen)
Sei $P \in E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$. Dann gibt es genau einen Punkt Q, so daß $P + Q = O$.

Beweis:
Für $P = O$ ist dies trivial. Sei daher $(\overline{x}_P, \overline{y}_P) \in E(\mathbb{Z}/n\mathbb{Z}) \setminus \{O\}$. Setze $M_{x_P} := \{P = (\overline{x}, \overline{y}) \in E(\mathbb{Z}/n\mathbb{Z}) \mid \overline{x} = \overline{x}_P\}$ und $\overline{d} := \overline{y}_P^2$. Es ist $M_{x_P} = \{(\overline{x}_P, \overline{h}) \mid \overline{h}^2 = \overline{d}\}$. Ist $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ mit paarweise verschiedenen Primzahlen $p_1, \ldots, p_l$, so gibt es nach Satz 4.1 genau 2^l verschiedene Wurzeln von $\overline{d}$; diese sind alle das direkte Produkt von Wurzeln von $\overline{d}_{p_i^{e_i}}$ in den Ringen $\mathbb{Z}/p_i^{e_i}\mathbb{Z}$. Es ist also $\sharp(M_{x_P}) = 2^l$. Sind $\overline{v} = (\overline{v}_1, \ldots, \overline{v}_l)$ und

[1] Dabei ist entweder $\overline{a} = \overline{y}_P - \overline{y}_Q$, oder $\overline{a} = \overline{x}_P - \overline{x}_Q$ oder $\overline{a} = 2\overline{y}_P$.

$\overline{w} = (\overline{w}_1, \ldots, \overline{w}_l) \in \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \ldots \times \mathbb{Z}/p_1^{e_1}\mathbb{Z}$ zwei Wurzeln von $\overline{d}$, so ist $\overline{v}_i = \pm\overline{w}_i$. Es ist $\overline{v} - \overline{w} \in (\mathbb{Z}/n\mathbb{Z})^*$ genau dann, wenn $\overline{v}_i - \overline{w}_i \neq 0$ für alle i. Es gibt also genau ein $(\overline{x}_P, \overline{y}_Q) \in M_{x_P}$ mit $\overline{y}_P - \overline{y}_Q \in (\mathbb{Z}/n\mathbb{Z})^*$ und $\overline{y}_Q^2 = \overline{y}_P^2$, nämlich $(\overline{x}_P, \overline{y}_Q) = (\overline{x}_P, -\overline{y}_P)$. □

Damit macht es also Sinn, für einen Punkt $P = (\overline{x}, \overline{y}) \in E(\mathbb{Z}/n\mathbb{Z}) \setminus \{O\}$ $-P := (\overline{x}, -\overline{y})$ zu definieren. Ferner ist $-O := O$.

Damit $E(\mathbb{Z}/n\mathbb{Z})$ die Eigenschaften einer n–Pseudogruppe erfüllt, muß es insbesondere für jeden Primteiler p von n eine verknüpfungserhaltende Abbildung in eine Gruppe geben. Daß dies so ist, versichert uns der folgende Satz:

Satz 5.9
Sei p ein Primteiler von n. Dann ist die Abbildung

$$\begin{array}{rccl} \pi_{np}: & E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z}) & \to & E_{\overline{a}_p,\overline{b}_p}(\mathfrak{p}) \\ & P = (\overline{x}, \overline{y}) & \mapsto & (\overline{x}_p, \overline{y}_p) =: P_p \\ & O & \mapsto & O \end{array}$$

wohldefiniert und hat die Eigenschaft, daß für alle $P, Q \in E(\mathbb{Z}/n\mathbb{Z})$ gilt:

1. $P = O \Leftrightarrow P_p = O$.
2. $-P_p = (-P)_p$.
3. Ist $P + Q$ definiert, so gilt: $P_p + Q_p = (P + Q)_p$.

Beweis:
Es ist $E_{\overline{a}_p,\overline{b}_p}(\mathfrak{p})$ die Menge der $\mathfrak{p}$-rationalen Punkte einer elliptischen Kurve, da aus $4\overline{a}^3 + 27\overline{b}^2 \in (\mathbb{Z}/n\mathbb{Z})^*$ folgt, daß $4\overline{a}_p^3 + 27\overline{b}_p^2 \neq 0$. Die Abbildung π_{np} ist wohldefiniert, denn ist $P = (\overline{x}, \overline{y}) \in E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z}) \setminus \{O\}$, so ist $\overline{y}^2 = \overline{x}^3 + \overline{a}\,\overline{x} + \overline{b}$, und da $\pi_{np} : \mathbb{Z}/n\mathbb{Z} \to \mathfrak{p}$ ein Ringhomomorphismus ist, folgt $\overline{y}_p^2 = \overline{x}_p^3 + \overline{a}_p\overline{x}_p + \overline{b}_p$, d.h. $(\overline{x}_p, \overline{y}_p) \in E_{\overline{a}_p,\overline{b}_p}(\mathfrak{p})$. Es sind also nur noch die Behauptungen 1 bis 3 zu zeigen:

1. Behauptung 1 folgt direkt aus der Definition der Abbildung.
2. Für $P = O$ ist Behauptung 2 trivial, ist hingegen $P = (\overline{x}, \overline{y})$, so folgt diese daraus, daß $(\overline{x}, -\overline{y})$ auf $(\overline{x}_p, -\overline{y}_p)$ abgebildet wird.
3. Für $P = O$ bzw. $Q = O$ folgt Behauptung 3 direkt aus der Definition der Verknüpfungen. Ist $P + Q = O$, so gilt nach Satz 5.8, daß $Q = -P$,

die Behauptung folgt also aus 2. Seien im folgenden $P = (\overline{x}_1, \overline{y}_1)$ und $Q = (\overline{x}_2, \overline{y}_2) \in E(\mathbb{Z}/n\mathbb{Z}) \setminus \{O\}$ mit $P + Q \neq O$. Ist $P + Q$ definiert, dann erfolgt folgende Fallunterscheidung:

(a) Fall: $\overline{x}_1 \neq \overline{x}_2$:
Wegen $\overline{x}_1 - \overline{x}_2 \in (\mathbb{Z}/n\mathbb{Z})^*$ folgt $\overline{x}_{1p} \neq \overline{x}_{2p}$. Dann erfolgen die Verknüpfungen $P+Q$ und P_p+Q_p durch die äquivalenten Formeln 3 und 5.

(b) Fall: $\overline{x}_1 = \overline{x}_2$:
Wegen $2\overline{y}_1 \in (\mathbb{Z}/n\mathbb{Z})^*$ folgt $\overline{y}_{1p} \neq 0$. Dann erfolgen die Verknüpfungen $P + Q$ und $P_p + Q_p$ durch die äquivalenten Formeln 4 und 6.

Die Gleichheit $P_p + Q_p = (P + Q)_p$ folgt dann aus der Tatsache, daß $\pi_{np} : \mathbb{Z}/n\mathbb{Z} \to \mathfrak{p}$ ein Ringhomomorphismus ist.

□

Wäre $E(\mathbb{Z}/n\mathbb{Z}) \subset (\mathbb{Z}/n\mathbb{Z})^3$, so würde $E(\mathbb{Z}/n\mathbb{Z})$ alle Eigenschaften einer n–Pseudogruppe erfüllen. Dieses letzte Hindernis überwindet man, indem man $E(\mathbb{Z}/n\mathbb{Z})$ mittels der Abbildung

$$\begin{array}{rccc} \iota: & E(\mathbb{Z}/n\mathbb{Z}) & \hookrightarrow & (\mathbb{Z}/n\mathbb{Z})^3 \\ & (\overline{x}, \overline{y}) & \mapsto & (\overline{x}, \overline{y}, 0) \\ & O & \mapsto & (0, 0, 1) \end{array}$$

in $(\mathbb{Z}/n\mathbb{Z})^3$ einbettet. Im folgenden wird $E(\mathbb{Z}/n\mathbb{Z})$ mit $\iota(\mathbb{Z}/n\mathbb{Z})$ identifiziert. Analog kann man für jeden Primteiler p von n auch $E(\mathbb{Z}/p\mathbb{Z})$ in $(\mathbb{Z}/n\mathbb{Z})^3$ einbetten. $E(\mathbb{Z}/n\mathbb{Z})$ ist somit eine n–Pseudogruppe.

6 Algorithmen unter Verwendung von elliptischen Kurven

6.1 Der Faktorisierungsalgorithmus von Lenstra

In den Kapiteln 6.1 und 6.2 sei n eine festgewählte Zahl mit $\mathrm{ggT}(n,6)=1$, und es seien $\overline{a},\overline{b}\in\mathbb{Z}/n\mathbb{Z}$ mit $4\overline{a}^3+27\overline{b}^2\in(\mathbb{Z}/n\mathbb{Z})^*$, und $P\in E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$.

Mit $E(\mathbb{Z}/n\mathbb{Z})$ als n–Pseudokurve erhält man folgendes Korollar aus Satz 4.5:

Satz 6.1
Besitzt n einen Primteiler p, und ist $k\in\mathbb{N}$ so gewählt, daß $\sharp E_{\overline{a}_p,\overline{b}_p}(\mathfrak{p})\mid k$, so ist entweder $kP=O$, oder kP ist nicht definiert.[1]

Beweis:
Nach Satz 4.5 ist $(kP)_p=O_p$, aus Satz 5.9 folgt dann $kP=O$. □

Wie beim $p-1$–Algorithmus wird man wieder

$$k:=k(B):=\prod_{p\le B,p\text{ prim}}p^{\left[\frac{\ln B}{\ln p}\right]}=\mathrm{kgV}((\,;1),2,\ldots,[B])$$

mit einem geeigneten B wählen. Um die Gesamtzeit zu ermitteln, bestimmt man zunächst die nötige Zeit für die Faktorisierung von n in Abhängigkeit von B und sucht dann das B, für welches die Zeit minimal wird.

Satz 6.2
Ist $B\in\mathbb{N}$, so werden $O(B(\ln n)^3)$ Bit–Operationen für die Berechnung von $k(B)P$ benötigt.

Beweis:
Für die Addition zweier Punkte auf $E(\mathbb{Z}/n\mathbb{Z})$ mittels der Additionsformeln 5 bzw. 6 benötigt man $O((\ln n)^3)$ Bit–Operationen, denn bei der Addition erfolgt einmal die Bestimmung des multiplikativ Inversen wofür man $O((\ln n)^3)$ Bit–Operationen braucht. Mit der Methode des sukzessiven Verdoppelns (vgl. Algorithmus 3.4) benötigt man $O(\ln k(B))$ Additionen für die Berechnung von $k(B)P$. Nach Satz 4.4 ist $\ln k(B)\approx B$. Die Gesamtzeit beläuft sich also auf $O((\ln n)^3)\cdot B=O(B(\ln n)^3)$ Bit–Operationen. □

[1] Dabei ist $kP:=k\odot P$ definiert, kP wird also mittels der Methode des sukzessiven Quadrierens berechnet.

Im folgenden sei p ein Primteiler von n. Nach dem Theorem von Hasse ist $m_{\overline{a}_p,\overline{b}_p} := \sharp E_{\overline{a}_p,\overline{b}_p}(\mathfrak{p}) \approx p$, für $\overline{a}_p, \overline{b}_p \in \mathbb{Z}/p\mathbb{Z}$ mit $4\overline{a}_p^3 + 27\overline{b}_p^2 \in (\mathbb{Z}/p\mathbb{Z})^*$. Nach Satz 4.2 ist die Wahrscheinlichkeit, daß $m_{\overline{a}_p,\overline{b}_p}$ $L(p)^a$–glatt ist, ungefähr $L(p)^{-\frac{1}{2a}+o(1)}$. Geht man davon aus, daß dieses Ergebnis ebenso für $L(p)^a$–potenzglatt gilt, und daß die Ordnungen $m_{\overline{a}_p,\overline{b}_p}$ gleichmäßig über das Intervall $[p+1-\sqrt{p}, p+1+\sqrt{p}]$ verteilt sind, so benötigt man durchschnittlich $L(p)^{\frac{1}{2a}+o(1)}$ Kurven, bis man eine Kurve mit $L(p)^a$–potenzglatter Ordnung gefunden hat. Setzt man $B := L(p)^a$, so ergibt sich eine Gesamtzeit von[1] $L(p)^{\frac{1}{2a}+o(1)}O(L(p)^a(\ln n)^3) = O(L(p)^{\frac{1}{2a}+\epsilon+a}(\ln n)^3)$ Bit–Operationen. Diese wird minimal für $a = \sqrt{\frac{1}{2}}$. Für das Auffinden des Teilers p benötigt man also $O(L(p)^{\sqrt{2}+\epsilon})$ Bit–Operationen. Mit $p \leq \sqrt{n}$ bemißt sich die Gesamtzeit für den allgemeinen Fall auf

$$\begin{aligned} O\left(L(p)^{\sqrt{2}+\epsilon}(\ln n)^3\right) &\leq O\left(L(\sqrt{p})^{\sqrt{2}+\epsilon}(\ln n)^3\right) = O\left(e^{(\sqrt{2}+\epsilon)\sqrt{\frac{1}{2}\ln n \ln\ln n}}\right) \\ &= O\left(e^{(1+\epsilon)\sqrt{\ln n \ln\ln n}}\right) = O\left(L(n)^{1+\epsilon}\right) \end{aligned}$$

Bit–Operationen. Dabei ist $B = L(\sqrt{n})^{\sqrt{\frac{1}{2}}} = \sqrt{L(n)}$, und man benötigt durchschnittlich $\sqrt{L(n)}$ verschiedene elliptische Kurven.

Damit erhält man die gleiche Zeitabschätzung für das Auffinden eines echten Teilers wie beim quadratischen Sieb. Allerdings ist der bei der O–Notation unterschlagene konstante Vorfaktor beim Lenstra–Algorithmus deutlich größer. Deswegen wird bei der Faktorisierung von RSA–Zahlen das quadratische Sieb bzw. das Zahlkörpersieb bevorzugt. Der Lenstra–Algorithmus hat aber zwei große Vorteile, welche ihn anwendungsrelevant machen:

1. Die Faktorisierungszeit hängt hauptsächlich von der Größe des zu findenden Primteilers ab. Der Lenstra–Algorithmus benötigt beispielsweise für das Auffinden eines 30–stelligen Primfaktors in einer 200–stelligen Zahl nur etwas länger als für das Auffinden in einer 100–stelligen Zahl.[2] Im Gegensatz dazu, ist es momentan nicht möglich, mit Hilfe des quadratischen Siebs einen Faktor in einer 200–stelligen Zahl zu finden.

[1] Die Gleichheit folgt aus $O(\epsilon) = o(1)$, dies wiederum folgt direkt aus den Definitionen.

[2] Der Geschwindigkeitsunterschied beim Lenstra–Algorithmus rührt daher, daß man in unterschiedlichen Ringen, nämlich $\mathbb{Z}/10^{200}\mathbb{Z}$ bzw. $\mathbb{Z}/10^{100}\mathbb{Z}$, rechnet. Der Algorithmus ist im Ring $\mathbb{Z}/10^{200}\mathbb{Z}$ um etwa den Faktor $\frac{\ln(10^{200})^3}{\ln(10^{100})^3} = 8$ langsamer. Beim quadratischen Sieb ist der Faktor etwa $\frac{L(10^{200})}{L(10^{100})} \approx 5.1 \cdot 10^7$.

2. Man benötigt für den Lenstra–Algorithmus nur sehr wenig Speicher, im Gegensatz zu dem großen Speicherbedarf, der sich durch die z.T. riesigen Matrizen ergibt, mit denen beim Faktor–Basis–Algorithmus bzw. beim quadratischen Sieb gearbeitet wird.[1]

Der von H. W. Lenstra 1987 entwickelte Algorithmus, sieht folgendermaßen aus:

Algorithmus 6.1 (Lenstra)
Sei n eine zu faktorisierende Zahl mit $\mathrm{ggT}(n, 6) = 1$.[2]

1. Setze $B := \sqrt{L(n)} = e^{\frac{1}{2}\sqrt{\ln n \ln\ln n}}$.
2. Wähle $\overline{x}, \overline{y}, \overline{a} \in \mathbb{Z}/n\mathbb{Z}$ und setze $\overline{b} := \overline{y}^{\,2} - \overline{x}^{\,3} - \overline{a}\,\overline{x}$.
3. Ist $4\overline{a}^{\,3} + 27\overline{b}^{\,2} = 0$, so gehe erneut zu 2. Tritt hingegen der Fall ein, daß $4\overline{a}^{\,3} + 27\overline{b}^{\,2} \notin (\mathbb{Z}/n\mathbb{Z})^* \cup \{0\}$, so hat man einen echten Teiler von n gefunden. Breche den Algorithmus ab.
4. Setze $P := (\overline{x}, \overline{y})$. Berechne kP mit $k := \prod_{p \le B, p \text{ prim}} p^{\left\lfloor \frac{\ln B}{\ln p} \right\rfloor}$.[3]
5. Ist die Addition jedesmal definiert, so gehe erneut zu 2. Andernfalls hat man nach Bemerkung 5.7 einen nichttrivialen Teiler von n gefunden. Breche den Algorithmus ab.

Wie der Faktor–Basis–Algorithmus hat auch der Lenstra–Algorithmus, im Gegensatz zum Rho–Algorithmus, den unschätzbaren Vorteil, daß das Faktorisierungsproblem auf viele Computer aufgeteilt werden kann, denn jedem Computer können elliptische Kurven zugewiesen werden, auf denen die Berechnung erfolgt. Ein psychologischer Nachteil des Lenstra–Algorithmus ist vielleicht, daß man während der Berechnung keine Fortschritte sieht; hat man z.B. nach 1000 Kurven noch keinen Teiler gefunden, ist die Wahrscheinlichkeit, demnächst einen Teiler zu finden, nicht größer als ganz am Anfang. Im Gegensatz dazu kann man beim Faktor–Basis–Algorithmus anhand der Zahl der schon gefundenen F–Zahlen abschätzen, wie viel Zeit der Algorithmus noch benötigt.

[1] Für die Faktorisierung einer 129–stelligen RSA–Zahl mittels des quadratischen Siebs wurde mit einer 188346×188346–Matrix gearbeitet.

[2] Vgl. Anmerkung zum Rho–Algorithmus.

[3] Vergleiche Anmerkung zu Algorithmus 4.3.

Vergleicht man Satz, Beweis und Algorithmus des $p-1$–Algorithmus mit denen des Lenstra–Algorithmus, so kann man folgende Analogien feststellen:

	$p-1$–Algorithmus	Lenstra–Algorithmus
Vorgegebene Menge mit teilweise erklärter Verknüpfung	$((\mathbb{Z}/n\mathbb{Z})^*, \cdot)$	$(E(\mathbb{Z}/n\mathbb{Z}), „ + “)$
Verknüpfungserhaltende Abbildung in eine Gruppe	$\pi_{np}:(\mathbb{Z}/n\mathbb{Z})^* \to (\mathfrak{p})^*$	$\pi_{np}:E(\mathbb{Z}/n\mathbb{Z}) \to E(\mathbb{Z}/p\mathbb{Z})$
Gruppenordnung, welche glatt sein soll	$p-1 = \sharp(\mathbb{Z}/p\mathbb{Z})^*$	$\sharp E(\mathfrak{p})$
Was passiert, wenn die Verknüpfung in der Gruppe das neutrale Element ergibt?	$\overline{a}_p \overline{b}_p = 1$ in $\mathbb{Z}/p\mathbb{Z}$ $\Rightarrow p \mid (ab-1)$ $\Rightarrow p \mid \mathrm{ggT}(ab-1, n)$	$P_p + Q_p = O$ $\Rightarrow P + Q = O$ oder $P + Q$ ist nicht definiert

6.2 Ein deterministischer Primzahltest

Betrachtet man $E(\mathbb{Z}/n\mathbb{Z})$ als n–Pseudogruppe wie in Kapitel 6.1, so erhält man als Korollar aus Satz 2.20 den folgenden Satz:

Satz 6.3
Sei $m := \sharp E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$. Gibt es eine Primzahl $p > (n^{\frac{1}{4}}+1)^2$ mit $p \mid m$ und einen Punkt P auf $E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$ mit den Eigenschaften

1. $mP = O$, und
2. $\left(\frac{m}{p}\right) P$ ist definiert und $\left(\frac{m}{p}\right) P \neq O$,

dann ist n eine Primzahl.

Beweis:
Angenommen n ist zusammengesetzt, dann gibt es einen Primteiler q von n mit $q \leq \sqrt{n}$. Aus dem Theorem von Hasse folgt

$$\sharp E_{\overline{a}_q,\overline{b}_q}(\mathbb{Z}/q\mathbb{Z}) \leq q + 1 + 2\sqrt{q} = (\sqrt{q}+1)^2 \leq (n^{\frac{1}{4}}+1)^2 < p$$

Der Satz folgt damit als Korollar aus Satz 2.20. □

Um aus Satz 6.3 einen Primzahltest–Algorithmus zu formulieren, benötigt man einen Algorithmus zur Bestimmung der Gruppenordnung $\sharp E(\mathbb{Z}/n\mathbb{Z})$. Alle bisher entwickelten Algorithmen sind jedoch sehr komplex und noch ziemlich langsam. Ausgehend von einem Algorithmus von Schoof [Schoof] benötigt der derzeit schnellste Algorithmus $O((\ln n)^{5+\epsilon})$ Bit–Operationen, um $\sharp E(\mathbb{Z}/n\mathbb{Z})$ zu bestimmen. Unter Annahme einer Hypothese von Goldwasser und Kilian, erhält man für den Primzahltest eine Laufzeit von $O((\ln n)^{10+l})$ Bit–Operationen (vgl. dazu [Gerhard] S. 88), wobei $l \in \mathbb{N}$ eine feste Konstante ist.

Der von Goldwasser und Kilian 1986 entwickelte Algorithmus sieht folgendermaßen aus:

Algorithmus 6.2 (Goldwasser–Kilian)
Sei n eine Zahl mit $\mathrm{ggT}(n, 6) = 1$, die auf Primalität überprüft werden soll.[1]

1. Ist n klein (etwa $n < 10^9$), so überprüfe die Primalität mittels des Probedivisionen–Algorithmus (Algorithmus 2.1). Breche den Goldwasser–Kilian–Algorithmus ab.

2. Wähle $\overline{x}, \overline{y}, \overline{a} \in \mathbb{Z}/n\mathbb{Z}$ und setze $\overline{b} := \overline{y}^2 - \overline{x}^3 - \overline{a}\,\overline{x}$.

3. Ist $4\overline{a}^3 + 27\overline{b}^2 = 0$, so gehe erneut zu 2. Tritt hingegen der Fall ein, daß $4\overline{a}^3 + 27\overline{b}^2 \notin (\mathbb{Z}/n\mathbb{Z})^* \cup \{0\}$, so breche den Algorithmus ab, n ist keine Primzahl.

4. Bestimme $m := \sharp E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$, z.B. mittels des Algorithmus von Schoof (vgl. [Schoof]).

5. Wähle eine Schranke $A \leq \sqrt{n}$.

6. Finde alle Primfaktoren $q < A$ von m. Sei p die Zahl, die man erhält, wenn man m durch die maximalen Potenzen der gefundenen Primfaktoren teilt. Ist $p \leq (n^{\frac{1}{4}} + 1)^2$, oder besteht p den Fermat–Test zu einer beliebigen Basis nicht, so gehe erneut zu 2.

7. Setze $P := (\overline{x}, \overline{y})$. Berechne $(\frac{m}{p})P$. Ist $(\frac{m}{p})P$ nicht definiert, so kann es sich bei n um keine Primzahl handeln. Breche den Algorithmus ab. Ist $(\frac{m}{p})P = O$, so gehe erneut zu 2. Ansonsten sind bis auf den Nachweis der Primalität von p alle Voraussetzungen von Satz 6.3 erfüllt. Um die Primalität von p zu beweisen, starte den Test neu mit $n := p$.

[1] Da der Goldwasser–Kilian–Algorithmus deutlich langsamer als der Fermat–Test ist, wird man auf n zuerst den Fermat–Test anwenden. Damit werden fast alle zusammengesetzten Zahlen aussortiert.

Eine Verbesserung dieses Algorithmus stammt von Atkin. Er sucht zuerst nach einer passenden Gruppenordnung und konstruiert dazu eine elliptische Kurve mit eben dieser Ordnung. Die Laufzeit kann so auf $O((\ln n)^{6+\epsilon})$ gedrückt werden. Dadurch wird der Algorithmus so leistungsfähig, daß die Primalität von mehr als 1000–stelligen Zahlen bewiesen werden konnte. Ausführlicher wird dieser Algorithmus in [LL87] geschildert.

Vergleicht man Satz 6.3 und seinen Beweis mit Satz 2.18 und dessen Beweis, so kann man folgende Analogien erkennen:

	Pocklington–Algorithmus	Goldwasser–Kilian–Algorithmus
Vorgegebene Menge mit teilweise erklärter Verknüpfung	$((\mathbb{Z}/n\mathbb{Z})^*, \cdot)$	$(E(\mathbb{Z}/n\mathbb{Z}), „, +_1)$
Anzahl der Elemente	$n-1$	$\sharp E(\mathbb{Z}/n\mathbb{Z}) = m$
Hat n einen Primteiler q so gibt es die verknüpfungserhaltende Abbildung	$\pi_{nq}: (\mathbb{Z}/n\mathbb{Z})^* \to (\mathfrak{q})^*$	$\pi_{nq}: E(\mathbb{Z}/n\mathbb{Z}) \to E(\mathbb{Z}/q\mathbb{Z})$
maximale Ordnung einer solchen Gruppe	$q-1 < \sqrt{n}-1 < p$	$m_q < (n^{\frac{1}{4}}+1)^2 < p$
1.Folgerung	Es ist $\mathrm{ggT}(p, q-1) = 1$, es existiert daher $up \equiv 1 \bmod (q-1)$	Es ist $\mathrm{ggT}(p, m_q) = 1$, es existiert daher $up \equiv 1 \bmod m_q$
2.Folgerung	$a^{\frac{n-1}{p}} \equiv a^{up\frac{n-1}{p}} \equiv (a^{n-1})^u \equiv 1 \bmod q$	$(\frac{m}{p})P_q = up(\frac{m}{p})P_q = u(mP)_q = O$
Wenn die Verknüpfung in der Gruppe das neutrale Element ergibt, folgt	aus $a^{\frac{n-1}{p}} \equiv 1 \bmod q$ folgt $\mathrm{ggT}(a^{\frac{n-1}{p}} - 1, n) \neq 1$	aus $(\frac{m}{p})P_q = O$ folgt $(\frac{m}{p})P = O$

6.3 Ein probabilistischer Primzahltest

Es sei $n \in \mathbb{N}$ eine festgewählte, auf Primalität zu überprüfende Zahl, mit $\mathrm{ggT}(n, 6) = 1$.

Im Kapitel 2 wurden zwei Primzahltests eingeführt die auf dem kleinen Fermatschen Satz basieren. Es bietet sich also an, dies auch für elliptische Kurven zu versuchen. Für die Anwendung des kleinen Fermat benötigt man die Kenntnis der Gruppenordnung, diese ist jedoch bei den elliptischen Kurven

bis auf wenige Ausnahmen nur sehr schwer zu bestimmen ist. Diese Ausnahmen sollen jetzt genauer untersucht werden:

Sei $E(\overline{K})$ eine elliptische Kurve. Dann bildet die Menge $\mathrm{End}(E(\overline{K}))$ der Endomorphismen $E(\overline{K}) \to E(\overline{K})$ mit den Verknüpfungen

$$\begin{aligned} \oplus : \mathrm{End}(E(\overline{K})) \times \mathrm{End}(E(\overline{K})) &\to \mathrm{End}(E(\overline{K})) \\ (\varphi, \psi) &\mapsto (P \mapsto \varphi(P) + \psi(P)) \\ \odot : \mathrm{End}(E(\overline{K})) \times \mathrm{End}(E(\overline{K})) &\to \mathrm{End}(E(\overline{K})) \\ (\varphi, \psi) &\mapsto (P \mapsto \varphi(\psi(P))) \end{aligned}$$

einen (evtl. nicht kommutativen) Ring mit eins.
Im folgenden bezeichne $\overline{\mathbb{Q}}$ einen algebraischen Abschluß von $\mathbb{Q}$. Für elliptische Kurven $E(\overline{\mathbb{Q}})$ gilt normalerweise, daß $\mathrm{End}(E(\overline{\mathbb{Q}})) = \mathbb{Z} \odot \mathrm{Id}$. Allerdings gibt es Ausnahmen. Beispielsweise gilt für die elliptische Kurve $E(\overline{\mathbb{Q}})$, welche durch die Weierstrass–Gleichung $Y^2 = X^3 + aX$ gegeben ist, daß $\mathrm{End}(E(\overline{\mathbb{Q}})) \otimes \overline{\mathbb{Q}} \cong \mathbb{Q}(\sqrt{-1})$, wobei

$$\begin{aligned} i : E(\overline{\mathbb{Q}}) &\to E(\overline{\mathbb{Q}}) \\ (x, y) &\mapsto (-x, iy) \\ O &\mapsto O \end{aligned}$$

Denn für $P = (x, y) \in E(\overline{\mathbb{Q}}) \setminus \{O\}$ ist

$$i^2(P) = i^2(x, y) = i(-x, iy) = (x, -y) = -P$$

Man sagt dann, die elliptische Kurve verfügt über eine komplexe Multiplikation.

Für diese Ausnahmen gilt der folgende Satz (vgl. [Silverman, S. 340] sowie [Cohen, S. 374 und 378]):

Satz 6.4
Sei $\overline{\mathbb{Q}}$ ein algebraischer Abschluß von $\mathbb{Q}$. Ist $E_{a,b}(\overline{\mathbb{Q}})$ eine elliptische Kurve mit $a, b \in \mathbb{Z}$ und

$$\mathrm{End}(E_{a,b}(\overline{\mathbb{Q}})) \otimes \mathbb{Q} \ncong \mathbb{Q}$$

so ist

$$\mathrm{End}(E_{a,b}(\overline{\mathbb{Q}})) \otimes \mathbb{Q} \cong \mathbb{Q}(\sqrt{-d})$$

mit einem quadratfreiem $d \in \mathbb{N}$. Ferner gilt:

1. Es ist $d \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$[1].

[1] Dies sind gerade die Zahlen, für welche die Klassenkörperzahl von $\mathbb{Q}(\sqrt{-d})$ gleich 1 ist.

2. Für jede Primzahl p mit $(-d \mid p) = -1$ [2] gilt mit $\overline{a} = \pi_p(a)$ und $\overline{b} = \pi_p(b)$, daß
$$\sharp(E_{\overline{a},\overline{b}}(\mathbb{Z}/p\mathbb{Z})) = p + 1$$

[2]Das Legendresymbol besagt in diesem Fall, daß es kein $\overline{a} \in \mathbb{Z}/p\mathbb{Z}$ mit $\overline{a}^2 = -\overline{d}$ gibt.

Definition:
Für $n \in \mathbb{N}$ bezeichne M_n die Menge aller Paare $(E_{a,b}(\overline{\mathbb{Q}}), P)$ mit $a, b \in \mathbb{Z}$, $\mathrm{End}(E_{a,b}(\overline{\mathbb{Q}})) \otimes \mathbb{Q} \cong \mathbb{Q}(\sqrt{-d})$ und $(-d \mid n) = 1$, sowie $P \in E_{a,b}(\overline{\mathbb{Q}})$.

Will man also eine Zahl n auf Primalität überprüfen, so wählt man zuerst $(E_{a,b}(\overline{\mathbb{Q}}), P) \in M_n$. Dann berechnet man $(n+1)P_n$ in $E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$, wobei $\overline{a} := \pi_n(a)$ und $\overline{b} := \pi_n(b)$. Ist n eine Primzahl, so muß das Ergebnis O sein.

Mit folgender Tabelle, welche Tabelle 1 in [Gordon89] entspricht, kann man für fast alle Zahlen n eine geeignete elliptische Kurve finden:

d	Elliptische Kurve mit $\mathrm{End}(E_{a,b}(\overline{\mathbb{Q}})) \otimes \mathbb{Q}) \cong \mathbb{Q}(\sqrt{-d})$	Punkt P auf der Kurve	geeignet für folgende n
1	$Y^2 = X^3 - 5X$	$P = (5, 10)$	$n \equiv 3 \bmod 4$ [1]
2	$Y^2 = X^3 - 120X - 448$	$P = (64, 504)$	$n \equiv 5, 7 \bmod 8$ [1]
3	$Y^2 = X^3 + 3$	$P = (1, 2)$	$n \equiv 2 \bmod 3$ [1]
7	$Y^2 = X^3 - 3500X - 98000$	$P = (84, 448)$	$n \equiv 3, 5, 6 \bmod 7$ [1]
11	$Y^2 = X^3 - 1056X + 13552$	$P = (33, 121)$	$(-11 \mid n) = -1$
19	$Y^2 = X^3 - 2432X - 46208$	$P = (57, 19)$	$(-19 \mid n) = -1$
43	$Y^2 = X^3 - 495360X - 134193024$	$P = (817, 2537)$	$(-43 \mid n) = -1$
67	$Y^2 = X^3 - 117920X + 15585808$	$P = (201, 67)$	$(-67 \mid n) = -1$
163	$Y^2 = X^3 - 34790720X + 78984748304$	$P = (3400, 548)$	$(-143 \mid n) = -1$

Tabelle 2: Elliptische Kurven mit verschiedenen Endomorphismenringen

Nur für jede 512. Zahl läßt sich in der Tabelle keine geeignete Kurve finden. In [Gordon89, S. 233] wird ein Weg beschrieben, wie auch dieses Problem umgangen werden kann.

Diese Primalitätsbedingung ist leider nicht hinreichend. Es gibt zusammengesetzte Zahlen n und $(E_{a,b}(\overline{\mathbb{Q}}), P) \in M_n$ mit $(n+1)P_n = O$ in $E_{\overline{a}_n,\overline{b}_n}(\mathbb{Z}/n\mathbb{Z})$. Solche Zahlen n werden als elliptische Pseudoprimzahlen (zur elliptischen Kurve $E_{a,b}(\overline{\mathbb{Q}})$ und zum Punkt P) bezeichnet.

Die bisher beste Abschätzung für

$$P_{E,P}(n) := \{l \in \{1, \ldots, n\} \mid l \text{ ist elliptische Pseudoprimzahl (zur elliptischen Kurve } E \text{ und zum Punkt } P)\}$$

stammt von Gordon und Pomerance aus dem Jahre 1991 (vgl. [GP]); demnach ist

$$P_{E,P}(n) \leq n e^{\frac{-\ln n \ln\ln\ln n}{3 \ln\ln n}}$$

[1] Diese Bedingung ist gleichbedeutend mit $(-d \mid n) = -1$.

Diese Abschätzung ist deutlich schwächer als die Abschätzung für Pseudoprimzahlen (vgl. Kapitel 2.2). Allerdings wird vermutet, daß dies noch nicht die schärfste Abschätzung ist, sondern daß Pseudoprimzahlen und elliptische Pseudoprimzahlen ähnlich selten sind.

Das dazugehörige Primzahltest–Verfahren, welches 1987 von Gordon entwickelt wurde (vgl. [Gordon87]), sieht folgendermaßen aus:

Verfahren 6.3 (Gordon)
Sei $n \in \mathbb{N}$ eine Zahl mit $\mathrm{ggT}(n, 6) = 1$, die auf Primalität überprüft werden soll.

1. Wähle eine elliptische Kurve E mit passendem Endomorphismenring und einen Punkt $P \in E$ nach Tabelle 2. Gibt es keine solche Kurve, dann breche das Verfahren ab, es kann nicht durchgeführt werden.

2. Berechne $(n+1)P_n$ in $E_{\overline{a},\overline{b}}(\mathbb{Z}/n\mathbb{Z})$. Ist $(n+1)P_n \neq O$, oder ist $(n+1)P_n$ nicht definiert, so ist n sicher eine zusammengesetzte Zahl. Ansonsten handelt es sich bei n höchstwahrscheinlich um eine Primzahl.

Das Verfahren benötigt, wie man analog zum Fermat–Test zeigen kann, $O((\ln n)^4)$ Bit–Operationen. Damit ist der Algorithmus langsamer als der Fermat–Test, welcher $O((\ln n)^3)$ Bit–Operationen benötigt. Allerdings siebt das Gordon–Verfahren „offensichtlich“ zusammengesetzte Zahlen schnell aus. Beispielsweise werden im Durchschnitt nach zwei Additionen auf $E(\mathbb{Z}/n\mathbb{Z})$ (d.h. nach zweimaligen Bestimmen des multiplikativ Inversen) ungerade Zahlen aussortiert. Dadurch relativiert sich der Geschwindigkeitsunterschied. Um die Zahlen 10^{100} bis $10^{100} + 230$ [1] auf Primalität zu überprüfen, benötigt das Programm „Enigma“ mit dem Fermat–Test etwa 22 Minuten und mit dem Gordon–Test etwa 44 Minuten.

Analog zur Definition der Miller–Rabin–Bedingung erfolgt folgende Definition:

Definition:
Sei $n \in \mathbb{N}$ mit $n = 2^s t$ und $2 \nmid t$. Ist $(E, P) \in M_n$, dann sagt man, n erfüllt die starke Primzahlbedingung zu (E, P), falls

1. $tP_n = O$, oder

2. es ein $l \in \{0, \ldots, s-1\}$ gibt, so daß $2^l tP_n = (\overline{x}, 0)$ mit einem $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$.

[1] Das ist also gerade der Bereich, in dem nach dem Primzahlsatz die erste Primzahl größer 10^{100} liegen sollte, denn $230 \approx \ln 10^{100}$.

Wie man sich leicht vergewissern kann, erfüllt jede Primzahl p die starke Primzahlbedingung zu allen $(E, P) \in M_p$. Es gibt aber auch zusammengesetzte Zahlen, welche die Bedingung erfüllen, sie werden starke elliptische Pseudoprimzahlen genannt. Setze

$$SP_{E,P}(n) := \{l \in \{1, \ldots, n\} \mid l \text{ ist starke elliptische Pseudoprimzahl (zur elliptischen Kurve } E \text{ und zum Punkt } P)\}$$

Die folgende Tabelle gibt einen Überblick über die Anzahl der starken Pseudoprimzahlen (vgl. [Gordon89, S. 235]):

n	$SP_{Y^2=X^3-13,(17,70)}(n)$ $d=3$	$SP_{Y^2=X^3-2,(3,5)}(n)$ $d=3$	$SP_{Y^2=X^3+3,(1,2)}(n)$ $d=3$
10^3	0	0	0
10^4	1	5	1
10^5	5	12	3
10^6	25	33	14
10^7	59	81	48
10^8	160	211	138

Abbildung 4: Anzahl von elliptischen Pseudoprimzahlen zu verschiedenen Kurven

Berücksichtigt man, daß nur jede 2. Zahl mit einer Kurve auf Primalität überprüft werden kann, und vergleicht dann diese Tabelle mit Tabelle 1, so sieht man, daß starke Pseudoprimzahlen und starke elliptische Pseudoprimzahlen etwa gleich häufig sind. Trotz allem sind die starken elliptischen Pseudoprimzahlen von Interesse, es gibt z.B. keine Zahl n mit $(-3 \mid n) = -1$ die starke elliptische Pseudoprimzahl zu den drei Kurven aus Tabelle 4 ist. Es verbleibt also noch viel Forschungsarbeit um die genauen Eigenschaften des Gordon–Verfahrens auszuloten. Ein interessanter Ansatzpunkt ist die Kombination von Fermat–Test und Gordon–Verfahren; ein solcher Test verspricht außergewöhnlich sicher zu sein.

7 Vergleich der Faktorisierungsalgorithmen

7.1 Einführung in das Programm „Enigma“

Das Programm „Enigma“ wurde von mir für diese Zulassungsarbeit in der Computersprache Turbo Pascal 5.5 geschrieben. Das Programm ist so angelegt, daß die Bedienung keinerlei Probleme aufwirft, zumal immer die jeweils wichtigsten Tastenbefehle angegeben sind. Es sollen deshalb im folgenden nur kurz die Möglichkeiten des Programms aufgeführt werden.
Das Programm ist in 3 Teile aufgegliedert:

1. Teil: „Term berechnen“:
   Dieser Teil des Programms ermöglicht die Eingabe und Berechnung von Termen im Ring $\mathbb{Z}$ [1], bzw. in den Restklassenringen $\mathbb{Z}/n\mathbb{Z}$ mit $n < 10^{120}$. Bei der Eingabe eines Terms können die vier Grundrechenarten[2] „$+1$, „-1, „$\cdot 1$, „$/1$, das Modulozeichen „mod“ sowie das Potenzzeichen „$\wedge$“ verwendet werden. Zudem arbeitet das Programm mit den folgenden Funktionen:

   (a) Fakultät einer Zahl: fak(...)[3].
   (b) Größter gemeinsamer Teiler zweier Zahlen: ggT(...,...)[3].
   (c) Das Legendresymbol: lgd(...,...)[3].
   (d) Berechnung des Inversen in Restklassenringen: inv(...).

2. Teil: „Primzahltest & Faktorisierungsmethoden“:
   In diesem Programmteil sind alle in dieser Arbeit beschriebenen Primzahltests und Faktorisierungsalgorithmen aufgeführt, mit Ausnahme des Goldwasser–Kilian–Algorithmus aus Kapitel 6.2. Stattdessen ist noch der Solovay–Strassen–Primzahltest einprogrammiert; die Theorie dazu kann z.B. in [Koblitz, S.128] nachgelesen werden. Angewandt werden können die Verfahren auf Zahlen die kleiner 10^{120} sind.
   Bei den verschiedenen Verfahren ist es jeweils möglich, die dazugehörigen Parameter selber festzulegen, z.B. welches Polynom beim Rho–Algorithmus verwendet werden soll. Die Faktorisierungsalgorithmen

[1] Genauer müßte man sagen: Das Programm ermöglicht Berechnungen im Zahlenbereich -10^{120} bis 10^{120}.

[2] Für $a, b \in \mathbb{Z}$ ist $a/b := [a/b]$ definiert. Wird hingegen im Ring $\mathbb{Z}/n\mathbb{Z}$ gerechnet, so ist $\overline{a}/\overline{b} := \overline{a}\,\overline{b}^{-1}$, sofern das Inverse von $\overline{b}$ bestimmt werden kann.

[3] Die Funktion kann auch in Restklassenringen verwendet werden. Anstatt mit der Restklasse $\overline{a}$ wird mit der Zahl $a = \eta_n(\overline{a})$ gerechnet. Beispielsweise ist $\mathrm{fak}(\overline{a}) := \overline{\mathrm{fak}(\eta_n(a))}$.

versuchen die eingegebenen Zahlen vollständig zu faktorisieren. Es kann zudem festgelegt werden, ob ein gefundener Teiler mittels eines Primzahltests auf Primalität überprüft werden soll (was die Voreinstellung ist), oder ob er immer einem weiteren Faktorisierungsverfahren unterworfen wird.
Die Geschwindigkeiten der verschiedenen Faktorisierungsalgorithmen können in dem Unterpunkt „Zeitbedarf der verschiedenen Tests“ verglichen werden. Dabei werden den verschiedenen Algorithmen Zahlen aufsteigender Größenordnung vorgelegt und die Faktorisierungszeiten gemessen. Das Ergebnis kann auch graphisch aufgetragen werden. Mit dieser Option wurden die Tabellen und Graphiken der folgenden Kapitel 7.2 und 7.3 erstellt.

3. Teil: „RSA–Verschlüsselungsalgorithmus“:
Als Motivation und Anwendung der verschiedenen Algorithmen habe ich noch den RSA–Algorithmus einprogrammiert. Das Programm erstellt auf Wunsch einen Code und ver- bzw. entschlüsselt damit Texte. Mit Hilfe des Faktor–Basis–Algorithmus kann man auch versuchen, einen Code zu knacken.

7.2 Faktorisierung von RSA–Zahlen

Um die Geschwindigkeit der verschiedenen Faktorisierungsalgorithmen zu vergleichen, bieten sich zwei verschiedene Möglichkeiten an. Entweder man vergleicht die Geschwindigkeiten bei der Faktorisierung von RSA–Zahlen, dies erfolgt in diesem Kapitel. Oder man vergleicht die Geschwindigkeiten bei der Faktorisierung von beliebigen Zahlen, was in Kapitel 7.3 erfolgt.

Für den Geschwindigkeitsvergleich bei der Faktorisierung von RSA–Zahlen wurden den Algorithmen 20 bis 30 Zahlen pro Größenordnung vorgelegt. Die Faktorisierungszeiten wurden mit dem Computerprogramm „enigma“ auf einem Pentium I mit 150 MHz gemessen. In den Tabellen 3 und 6 werden die Zeiten (in Hundertstel Sekunden) von drei repräsentativen Werten pro Größenordnung sowie der Durchschnittswert angegeben. Da die Zeiten für den Probedivisionen–Algorithmus für große Zahlen zu lang sind, wurden diese nicht mehr bestimmt.

Folgende Funktionen geben die Faktorisierungszeiten einer Zahl n für die jeweiligen Algorithmen an:

1. Probedivisionen–Algorithmus: $P(n)$,

2. Rho–Algorithmus: $R(n)$,
3. Faktor–Basis–Algorithmus: $F(n)$,
4. Lenstra–Algorithmus: $L(n)$.

zu faktorisierende Zahl n	$P(n)$	$R(n)$	$F(n)$	$L(n)$
$1033 \cdot 1009$	22	5	11	17
$1153 \cdot 1103$	22	11	11	110
$1069 \cdot 1031$	22	11	11	60
Durchschnittswert für $n \approx 10^6$	21	13	13	74
$10211 \cdot 10037$	154	22	38	110
$10477 \cdot 10159$	154	16	33	302
$10181 \cdot 10111$	154	49	27	204
Durchschnittswert für $n \approx 10^8$	157	21	25	172
$100391 \cdot 101533$	1670	55	60	593
$100501 \cdot 100907$	1670	44	44	247
$100769 \cdot 100483$	1670	66	66	132
Durchschnittswert für $n \approx 10^{10}$	1675	67	65	610
$1001153 \cdot 1004963$	18449	137	110	88
$1003463 \cdot 1000777$	18449	61	137	2713
$1004903 \cdot 1004287$	18449	132	176	1675
Durchschnittswert für $n \approx 10^{12}$	18449	170	153	1711
$10015127 \cdot 10009093$	182820	742	297	23749
$10004063 \cdot 10000189$	182820	571	318	1143
$10004921 \cdot 10008263$	182820	374	242	2335
Durchschnittswert für $n \approx 10^{14}$	182820	478	333	5154
$100001581 \cdot 100040407$		3949	802	3834
$100023367 \cdot 100033469$		2949	632	429
$100042409 \cdot 100003151$		1989	912	34377
Durchschnittswert für $n \approx 10^{16}$		2437	888	19674
$1000039643 \cdot 1000003853$		7926	1572	45604
$1000014397 \cdot 1000042991$		9200	1824	52081
$1000022533 \cdot 1000030879$		6761	1724	36525
Durchschnittswert für $n \approx 10^{18}$		6817	1812	35978
$10000089851 \cdot 10000379371$		17318	5179	65070
$10000110181 \cdot 10000034557$		42034	5701	13720
$10000277153 \cdot 10000326353$		33120	3340	51295
Durchschnittswert für $n \approx 10^{20}$		25989	3938	89330
$100001293211 \cdot 100000232609$		90567	8118	13473
$100000317581 \cdot 100000656751$		47197	9931	349107
$100000592309 \cdot 100000256047$		17301	7486	317996
Durchschnittswert für $n \approx 10^{22}$		73727	9168	176761
$1000003817011 \cdot 1000000203053$		70392	18949	55662
$1000003411313 \cdot 1000001009801$		283147	18076	800802
$1000003674661 \cdot 1000001380181$		159833	17121	336830
Durchschnittswert für $n \approx 10^{24}$		253825	19863	370831
$10000004920579 \cdot 10000011064391$		311433	46296	403252
$10000014595591 \cdot 10000003354069$		468701	44165	9519
$10000009841687 \cdot 10000015713283$		252201	54272	153962
$10000010627249 \cdot 10000006573769$		891853	53866	3682243
$10000013451499 \cdot 10000004348627$		574241	52509	69673
$10000014184481 \cdot 10000012274569$		1125264	52454	2239499
$10000011456937 \cdot 10000014104819$		1419237	43138	324357
$10000013384567 \cdot 10000013368391$		893874	44017	456436
Durchschnittswert für $n \approx 10^{26}$		766239	47436	825710

Tabelle 3: Vergleich der Faktorisierungszeiten für RSA–Zahlen

Schon ein flüchtiger Blick auf die Tabelle zeigt ein Unterscheidungsmerkmal zwischen den verschiedenen Faktorisierungsalgorithmen:
Während beim Probedivisionen–Algorithmus die Zeiten für Zahlen der gleichen Größenordnung so gut wie gleich sind, und beim Faktor–Basis–Algorithmus die Unterschiede nur gering sind, gibt es bei den anderen Algorithmen größere Abweichungen. Am größten sind die Unterschiede beim Lenstra–Algorithmus; dies ist auch nicht weiter verwunderlich, denn die Wahrscheinlichkeit, daß ein Teiler gefunden wird, ist für jede elliptische Kurve gleich. Mit etwas Glück kann also ein Teiler schon bei der ersten Kurve gefunden werden; wenn man Pech hat, dauert es hingegen ein Vielfaches der veranschlagten Zeit. Ein gutes Beispiel hierfür kann man bei den Faktorisierungszeiten für Zahlen der Größenordnung 10^{26} sehen, wo sich die Zeiten beim Lenstra–Algorithmus um mehr als einen Faktor 300 unterscheiden.

Für große Zahlen ist das asymptotische Verhalten der verschiedenen Algorithmen entscheidend. In den jeweiligen Kapiteln wurden die folgenden Geschwindigkeitsabschätzungen ermittelt:

Algorithmus	benötigte Bit–Operationen für die Faktorisierung einer Zahl n
Probedivisionen–Algorithmus	$O\left(n^{\frac{1}{2}+\epsilon}\right) = O\left(e^{\left(\frac{1}{2}+\epsilon\right)\ln n}\right)$
Rho–Algorithmus	$O\left(n^{\frac{1}{4}+\epsilon}\right) = O\left(e^{\left(\frac{1}{4}+\epsilon\right)\ln n}\right)$
Faktor–Basis–Algorithmus	$O\left(e^{(\sqrt{2}+\epsilon)\sqrt{\ln n \ln\ln n}}\right)$
Lenstra–Algorithmus	$O\left(e^{(1+\epsilon)\sqrt{\ln n \ln\ln n}}\right)$

Tabelle 4: Faktorisierungszeiten für Produkte zweier großer Primzahlen

Nach dieser Tabelle sollte der Lenstra–Algorithmus asymptotisch deutlich schneller sein als der Faktor–Basis–Algorithmus und dieser wiederum schneller als der Rho–Algorithmus, welcher schneller ist als der Probedivisionen–Algorithmus. In der Praxis bietet sich aber ein anderes Bild. Während der Faktor–Basis–Algorithmus die in ihn gesetzten Erwartungen erfüllt, ist der Lenstra–Algorithmus über den ganzen Meßbereich langsamer als der Rho–Algorithmus und der Faktor–Basis–Algorithmus. Dies bedeutet jedoch nicht, daß der Lenstra–Algorithmus für immer langsamer ist. Die leicht lesbare O–Notation verschweigt nämlich, daß es bei der Geschwindigkeitsabschätzung noch einen Vorfaktor gibt, der beim Lenstra–Algorithmus sehr hoch ist;

schließlich verschlingt die Addition auf elliptischen Kurven deutlich mehr Rechenoperationen als die Verknüpfung im Ring $\mathbb{Z}/n\mathbb{Z}$. Der Lenstra–Algorithmus wird also erst für große Zahlen seinen Geschwindigkeitsvorteil ausspielen können.

Betrachtet man die folgende doppeltlogarithmische Auftragung der Faktorisierungszeiten, so erkennt man, daß der Lenstra–Algorithmus gegenüber dem Rho–Algorithmus aufholt und ihn „demnächst“ überholen wird:

Abbildung 5: Doppeltlogarithmische Auftragung der verschiedenen Faktorisierungszeiten. Die punktierten Linien entsprechem dem theoretisch zu erwartendem Verlauf.

Kleinere Abweichungen der gemessenen von der theoretisch zu erwartenden Kurve lassen sich durch die Nichtberücksichtigung des „ϵ's“ bei der punktierten Linie und durch die (abgesehen vom Probedivisionen–Algorithmus) immer vorhandenen Zufallselemente erklären. Da der Faktor–Basis–Algorithmus viel Freiraum für Abänderungen bietet, und die Implementierung als Programm deutlich aufwendiger ist als bei den anderen Algorithmen, kommen noch weitere Faktoren hinzu, welche die deutliche Abweichung von der eigentlich zu erwartenden Kurve erklären. Beispielsweise konnten durch geschickte Implementierung des Algorithmus viele Berechnungen an den Co–Prozessor abgeleitet werden. In dem ausgewerteten Zahlenbereich muß die Geschwindigkeit des Faktor–Basis–Algorithmus daher eher mit $e^{(1+\epsilon)\sqrt{\ln n \ln\ln n}}$ angegeben werden.

Die Ergebnisse zeigen aber auch, warum das RSA–Verfahren so sicher ist. Rechnet man die die Zeiten des Lenstra–Algorithmus auf Zahlen der Größenordnung 10^{100} hoch, so erhält man eine voraussichtliche Faktorisierungszeit von 94142 Jahren! Im Gegensatz dazu kann man mittels des Fermat–Verfahrens in wenigen Sekunden mit minimaler Fehlerwahrscheinlichkeit sagen, ob es sich bei einer 100–stelligen Zahl um eine Primzahl handelt. Selbst wenn es in der Zukunft gelingen sollte, mittels besserer Computer und besserer Algorithmen, 100–stellige Zahlen in einem Bruchteil der heutigen Zeit zu faktorisieren, so kann das RSA–Verfahren weiter benützt werden; schließlich genügt es, auf größere Zahlen umzusteigen, um den Geschwindigkeitsvorsprung wieder herzustellen. Das RSA–Verfahren ist erst dann gefährdet, wenn die Faktorisierungsalgorithmen auch asymptotisch ähnlich schnell sind wie die Primzahltests. Die meisten Forscher gehen allerdings davon aus, daß es unmöglich ist, solche Faktorisierungsalgorithmen zu finden. Dies ist jedoch nicht bewiesen, und in [Riesel, S. 218] wird z.B. die These vertreten, daß es einen Faktorisierungsalgorithmus geben müßte, der $O((\ln n)^k)$ Bit–Operationen mit einem festen $k \in \mathbb{R}^+$ benötigt. Nach dem heutigen Stand der Forschung ist man aber davon weit entfernt.

7.3 Faktorisierung von beliebigen Zahlen

Sieht man sich die verschiedenen Faktorisierungsalgorithmen etwas genauer an, dann bemerkt man, daß Tabelle 4 nur die Zeit angibt, die man braucht, um eine RSA–Zahl zu faktorisieren. Bis auf den Faktor–Basis–Algorithmus hängt bei den Faktorisierungsalgorithmen die Zeit für das Auffinden eines Teilers jedoch von der Größe des zu findenden Primteilers ab:

Algorithmus	benötigte Bit–Operationen für das Auffinden eines Primfaktors p einer Zahl n
Probedivisionen–Algorithmus	$O\left(n^{\epsilon}p\right)$
Rho–Algorithmus	$O\left(n^{\epsilon}p^{\frac{1}{2}}\right)$
Faktor–Basis–Algorithmus	$O\left(n^{\epsilon}e^{\sqrt{2\ln n \ln\ln n}}\right)$
Lenstra–Algorithmus	$O\left(n^{\epsilon}e^{\sqrt{\ln p \ln\ln p}}\right)$

Tabelle 5: Zeiten für das Auffinden eines Primteilers p in einer Zahl n

Der Faktor n^ϵ steht dabei für die Zeit, die für die Berechnungen im Ring $\mathbb{Z}/n\mathbb{Z}$ benötigt wird, während der Faktor in p für die Anzahl der Berechnungen in diesem Ring steht. Die „worst case“ Zeitabschätzungen erhält man aus der Tabelle, indem man p durch $\sqrt{n}$ ersetzt.

Für eine beliebig gewählte Zahl $n \in \mathbb{N}$ kann man folgende beide Vorhersagen treffen:

1. Die Zahl n hat im Durchschnitt etwa $\ln \ln n$ verschiedene Primfaktoren.

2. Der Logarithmus des größten Primteiler von n ist im Durchschnitt von der Größenordnung $0.624 \ln n$ und der Logarithmus des zweitgrößten von der Größenordnung $0.210 \ln n$.

Genauer formuliert und bewiesen werden diese Aussagen in [HW68, S. 355] bzw. [KTP]. Die erste Aussage hat keine Auswirkungen auf die oben genannten Geschwindigkeitsangaben; sie ist wegen $\ln \ln n = O(n^\epsilon)$ schon in dem „ϵ“ enthalten. Die zweite Aussage hat aber sehr wohl Auswirkungen, denn wie man sich leicht überlegen kann, hängt, außer beim Faktor–Basis–Algorithmus, die Geschwindigkeit für die Faktorisierung einer Zahl n von der Größe des zweitgrößten Primteilers ab. Die 2. Aussage soll daher im folgenden etwas genauer diskutiert werden:

Definition:
Für gegebenes $n \in \mathbb{N}$ und $x \in [0, 1]$ setze

$$F_{nk}(x) := \frac{\sharp\{l \in \{1, \ldots, n\} \mid \text{ der } k. \text{ größte Primteiler von } l \text{ ist kleiner } n^x\}}{n}$$

Für $F_k(x) := \lim_{n\to\infty} F_{nk}(x)$ ergibt sich folgender Graph (vgl. [KTP]):

Abbildung 6: Graph von $F_k(x)$ für $k = 1, 2, 3$

Als zu erwartende Zeit $T(n)$ für die Faktorisierung einer beliebig gewählten Zahl der Größenordnung n mit Hilfe des Probedivisionen–Algorithmus ergibt sich dann

$$T(n) = \int_{x=0}^{x=1} n^x F_2'(x) \mathrm{d}x =: n^{f(n)}$$

Man kann sich leicht vergewissern, daß zwar $f(n) < \frac{1}{2}$, aber $\lim_{n\to\infty} f(n) = \frac{1}{2}$. Als einzige Abschätzung mit einem konstanten Exponenten erhält man also wieder $O\left(n^{\frac{1}{2}+\epsilon}\right)$. Obwohl die zu erwartenden Zeiten für „kleine“ n deutlich geringer sind als bei der Faktorisierung der RSA–Zahlen, erhält man doch asymptotisch die gleiche Geschwindigkeitsabschätzung. Analog erhält man für den Rho–Algorithmus, daß er wiederum $O\left(n^{\frac{1}{4}+\epsilon}\right)$ Bit–Operationen benötigt. Asymptotisch ist damit der Faktor–Basis–Algorithmus zwar immer noch schneller als der Rho–Algorithmus, aber das wird sich erst sehr viel später bemerkbar machen.

Die folgende Tabelle gibt die Faktorisierungszeiten für beliebig gewählte Zahlen aufsteigender Größenordnung an. Es wurden den Algorithmen jeweils 50 Zahlen pro Größenordnung vorgelegt, und daraus dann die durchschnittliche Zeit ermittelt. Die Faktorisierungszeiten sind wieder in hundertstel Sekunden angegeben.[1]

[1]Beim Rho–, beim Faktor–Basis– und beim Lenstra–Algorithmus wird ein gefundener Faktor mittels des kleinen Fermat auf Zusammengesetztheit überprüft. Ist ein Faktor

zusammengesetzt, so wird er mit Hilfe des jeweiligen Faktorisierungsalgorithmus weiterzerlegt. Ausgenommen davon sind nur kleine Faktoren (d.h. Faktoren$<$1000) welche von dem Programm mit dem Rho–Algorithmus faktorisiert werden.

Zahl n	Faktorisierung	$P(n)$	$R(n)$	$F(n)$	$L(n)$
1001183	$37 \cdot 27059$	5	11	11	17
1001623	$7 \cdot 17 \cdot 19 \cdot 443$	11	28	44	43
1000622	$2 \cdot 7 \cdot 71473$	11	11	17	22
Durchschnittswert für Zahlen der Größenordnung 10^6		9	15	25	33
100043440	$4^2 \cdot 5 \cdot 7 \cdot 227 \cdot 787$	17	33	43	33
100041329	$23 \cdot 29 \cdot 127 \cdot 1181$	11	22	33	38
100014116	$4 \cdot 67 \cdot 373187$	11	16	39	55
Durchschnittswert für Zahlen der Größenordnung 10^8		15	21	41	45
10000288152	$2 \cdot 3^2 \cdot 4 \cdot 373 \cdot 372367$	17	38	50	33
10000122836	$4 \cdot 11 \cdot 227275519$	11	17	61	32
10000134481	$43 \cdot 443 \cdot 524969$	16	22	230	138
Durchschnittswert für Zahlen der Größenordnung 10^{10}		60	28	93	93
1000004323275	$3^2 \cdot 5^2 \cdot 461 \cdot 9640919$	22	44	93	50
1000004571681	$3 \cdot 38543 \cdot 8648389$	643	115	214	159
1000000033612	$2^2 \cdot 67 \cdot 12671 \cdot 294479$	209	50	248	164
Durchschnittswert für Zahlen der Größenordnung 10^{12}		270	41	179	261
100000020256507	$7 \cdot 1297 \cdot 44531 \cdot 247343$	741	121	456	1455
100000004037552	$3^2 \cdot 4^2 \cdot 694444472483$	11	27	143	27
100000037898292	$4 \cdot 7 \cdot 887 \cdot 6673 \cdot 603389$	121	71	396	203
Durchschnittswert für Zahlen der Größenordnung 10^{14}		1979	70	412	313
10000000202202128	$4^2 \cdot 587749 \cdot 1063379117$	11375	362	977	259
10000000177957109	$59 \cdot 8609 \cdot 81331 \cdot 242069$	1367	116	1489	472
10000000204210732	$4 \cdot 109 \cdot 22935780284887$	22	27	818	55
Durchschnittswert für Zahlen der Größenordnung 10^{16}		20567	170	1033	1616
1000000001843328864	$2 \cdot 3 \cdot 4^2 \cdot 7 \cdot 1488095240838287$	33	66	1170	44
1000000004288348480	$4^3 \cdot 5 \cdot 4663 \cdot 195809 \cdot 3422567$	3400	165	1269	1071
1000000003053913745	$5 \cdot 13^2 \cdot 37963 \cdot 31173299167$	687	132	1444	154
Durchschnittswert für Zahlen der Größenordnung 10^{18}		188257	322	2118	2128
10^{20}+42480415034	$2 \cdot 3^2 \cdot 43 \cdot 573343 \cdot 225343235137$		544	2960	549
10^{20}+17648570818	$2 \cdot 11 \cdot 2164843 \cdot 2099669373833$		1719	5295	27056
10^{20}+12514367103	$13 \cdot 6131 \cdot 1254657917675801$		61	6025	192
Durchschnittswert für Zahlen der Größenordnung 10^{20}			645	4735	3688
10^{22}+249364805009	$3 \cdot 972225211 \cdot 3428560888673$		10183	18422	49685
10^{22}+191597369837	$3 \cdot 19 \cdot 4521607 \cdot 38800054160963$		2494	11095	1450
10^{22}+309051026423	$3^2 \cdot 79 \cdot 619 \cdot 27762739 \cdot 818422273$		1659	9128	22311
Durchschnittswert für Zahlen der Größenordnung 10^{22}			1737	11775	6462
10^{24}+3725703927766	$2 \cdot 379 \cdot 2027 \cdot 650844210027248051$		88	52305	440
10^{24}+3218591483819	$3^2 \cdot 1091 \cdot 1334314297 \cdot 76326368633$		27304	43413	38689
10^{24}+4641075764622	$2 \cdot 19 \cdot 23 \cdot 42261629 \cdot 27073370970407$		2082	24722	17109
Durchschnittswert für Zahlen der Größenordnung 10^{24}			3959	25347	14031
10^{26}+14504192406775	$5^2 \cdot 54088068943 \cdot 73953462901697$		173263	212990	367484
10^{26}+26873819890042	$2 \cdot 19 \cdot 67 \cdot 79 \cdot 218079649 \cdot 2279813756587$		5982	118518	12248
10^{26}+46783924958530	$2 \cdot 5 \cdot 17 \cdot 588235294117922258382109$		60	53239	264
10^{26}+14340646344927	$129459249071 \cdot 772443844048337$		100749	49307	85728
10^{26}+42207385477433	$3 \cdot 79 \cdot 80809 \cdot 5221459593241102901$		143	41716	1741
10^{26}+23324802288390	$2 \cdot 5 \cdot 5779 \cdot 547909 \cdot 3158194488395449$		538	45500	3098
Durchschnittswert für Zahlen der Größenordnung 10^{26}			23176	55836	77711

Tabelle 6: Vergleich der Faktorisierungszeiten von beliebigen Zahlen

Beim Probedivisionen–Algorithmus, beim Rho–Algorithmus sowie beim Lenstra–Algorithmus kann man leicht den direkten Zusammenhang zwischen der Größe des zweitgrößten Primteilers und den Faktorisierungszeiten sehen. Vergleicht man Tabelle 6 mit Tabelle 3, erkennt man auch, daß die Zeiten für die drei primzahlengrößenabhängigen Algorithmen bei der Faktorisierung beliebiger Zahlen deutlich geringer sind als die Zeiten bei der Faktorisierung von RSA–Zahlen. Im Gegensatz dazu sind in Tabelle 6 die Zeiten des Faktor–Basis–Algorithmus sogar etwas höher. Dies erklärt sich daraus, daß bei beliebigen Zahlen nicht nur eine, sondern meistens mehrere Faktorisierungen erfolgen müssen.

Auch diesesmal ist die graphische Auftragung sehr aussagekräftig:

Abbildung 7: Doppeltlogarithmische Auftragung der verschiedenen Zeiten

Die Ausschläge der Graphen für die primfaktorgrößen–abhängigen Algorithmen rühren daher, daß die Zeiten für die Faktorisierung beliebiger Zahlen sehr unterschiedlich sind und 50 Meßwerte noch nicht genug sind, um die Schwankung in Grenzen zu halten. Der Faktor–Basis–Algorithmus hingegen zeigt sich nahezu unbeeindruckt von der Gestalt der Zahlen und hat keine nennenswerte Auschläge.

Der Graph für den Probedivisionen–Algorithmus bestätigt die These, daß der Algorithmus auch für beliebige Zahlen $O(n^{\frac{1}{2}+\epsilon})$ Bit–Operationen benötigt. Der Graph steigt zwar zu Beginn etwas langsamer an, aber ab $n \approx 10^{12}$ ist die Steigung ähnlich wie für RSA–Zahlen.

8 Ein Beweis des Assoziativgesetzes für elliptische Kurven

Für den Rest des Kapitels sei $E_{a,b}(\overline{K})$ eine fest gewählte elliptische Kurve über einem algebraisch abgeschlossenen Körper $\overline{K}$ mit Char $\overline{K} > 3$. In diesem Kapitel wird bewiesen, daß

$$(A + B) + C = A + (B + C)$$

für alle $A, B, C \in E_{a,b}(\overline{K})$.

Dabei werden folgende Tatsachen verwendet welche sich direkt aus der Definition der Verknüpfung bzw. deren Beschreibung in Satz 5.3 ergeben:

1. Die Verknüpfung „+“ ist kommutativ.
2. Für $A = (x, y) \in E_{a,b}(\overline{K}) \setminus \{O\}$ ist $A + A = O$ äquivalent mit $y = 0$.
3. Für ein $x \in \overline{K}$ sei $M_x := \{A = (x_A, y_A) \in E_{a,b}(\overline{K}) \setminus \{O\} \mid x_A = x\}$. Ist $A = (x_A, y_A) \in M_x$, dann ist $M_x = \{A = (x_A, y_A), -A = (x_A, -y_A)\}$. M_x enthält also höchstens zwei Elemente.

Nach Satz 5.3 kann die Verknüpfung „+“ bis auf drei Spezialfälle durch die Formeln 3 und 4 beschrieben werden. In Kapitel 8.1 wird die Assoziativität für 3 der 4 Fälle bewiesen, bei denen die Verknüpfungen jeweils mit einer der beiden Formeln ausgedrückt werden kann; dies erfolgt durch direktes Nachrechnen. Im anschließenden Kapitel 8.2 werden verschiedene Lemmas bewiesen, die dann zum allgemeinen Beweis des Assoziativgesetzes in Kapitel 8.3 herangezogen werden.

8.1 Beweis mehrerer Spezialfälle mit Hilfe des Computers

Sei $p := \text{Char}(\overline{K})$. Setze $R_0 := \mathbb{Z}$ und, falls $p > 0$, $R_p := \mathfrak{p}$. Zudem bezeichne für $q = 0, p$ im folgenden

$$\begin{aligned} P_q &:= R_q[x_A, x_B, x_C, y_A, y_B, y_C, a, b] \\ I_q &:= (y_A^2 - x_A^3 - ax_A - b, y_B^2 - x_B^3 - ax_B - b, y_C^2 - x_C^3 - ax_C - b) \subset P_q \end{aligned}$$

In den Beweisen zu den Sätzen 8.1 bis 8.3 muß gezeigt werden, daß verschiedene Terme f im Ring P_p/I_p verschwinden. Aufgrund des kommutativen Diagramms[1]

$$\begin{array}{ccc} P_0 & \xrightarrow{\pi_p} & P_p \\ \pi_I \downarrow & & \downarrow \pi_I \\ P_0/I_0 & \xrightarrow{\pi_p} & P_p/I_p \end{array}$$

genügt es zu zeigen, daß $(\pi_I)^{-1}(\pi_p^{-1}(f))$ in I_0 liegt. Da die Polynome in P_0 zum Teil mehrere zigtausend Summanden umfassen, ist die Berechnung von Hand nicht mehr möglich. Die Berechnungen müssen deshalb mit einem geeigneten Computerprogramm erfolgen.

Lemma 8.1
Seien $A, B, C \in E_{a,b}(\overline{K}) \setminus \{O\}$. Wird bei der Berechnung von $(A+B)+C$ und $A+(B+C)$ nur die Additionsformel 3 aus Satz 5.3 verwendet, d.h. ist $A \neq \pm B$, $B \neq \pm C$, $A+B \neq \pm C$ und $B+C \neq \pm A$, so gilt:

$$(A+B)+C = A+(B+C)$$

Beweis:
Setze $(x_A, y_A) := A, (x_B, y_B) := B, (x_C, y_C) := C$ und $(x_1, y_1) := (A+B)+C$ und $(x_2, y_2) := A+(B+C)$. Mit[2]

$$\begin{aligned} \alpha &:= \frac{y_B - y_A}{x_B - x_A} \\ \beta &:= \frac{y_A + y_C - \alpha(2x_A + x_B - \alpha^2)}{x_A + x_B + x_C - \alpha^2} \\ \gamma &:= \frac{y_B - y_C}{x_B - x_C} \\ \tau &:= \frac{y_A + y_B - \gamma(2x_B + x_C - \gamma^2)}{x_A + x_B + x_C - \gamma^2} \end{aligned}$$

erhält man durch zweimaliges Verwenden der Formel 3, daß

$$x_1 = \beta^2 + x_A + x_B - x_C - \alpha^2$$

[1] Dabei bezeichnen π_p, π_I die kanonischen Projektionen.

[2] Die Nenner in den Termen verschwinden wegen den Einschränkungen bei der Wahl von A, B, C nicht.

$$\begin{aligned} y_1 &= -y_C + \beta(2x_C - x_A - x_B - \beta^2 + \alpha^2) \\ x_2 &= \tau^2 + x_B + x_C - x_A - \gamma^2 \\ y_2 &= -y_A + \tau(2x_A - x_B - x_C - \tau^2 + \gamma^2) \end{aligned}$$

Setzt man

$$\begin{aligned} \tilde{\alpha} &:= y_B - x_A \\ \tilde{\beta} &:= (y_A + y_C)(x_B - x_A)^3 - \tilde{\alpha}((2x_A + x_B)(x_B - x_A)^2 - \tilde{\alpha}^2) \\ \tilde{\gamma} &:= y_B - y_C \\ \tilde{\tau} &:= (y_A + y_B)(x_B - x_C)^3 - \tilde{\gamma}((2x_B + x_C)(x_B - x_C)^2 - \tilde{\gamma}^2) \\ \tilde{\eta} &:= x_B - x_A \\ \tilde{\mu} &:= x_B - x_C \end{aligned}$$

so kann man zeigen, daß $x_1 = x_2$ äquivalent ist zu

$$\begin{aligned} &(\tilde{\beta}^2(x_B - x_C)^2 + (((2x_A - 2x_C)(x_B - x_C)^2 + \tilde{\gamma}^2)(x_B - x_A)^2 - \tilde{\alpha}^2(x_B - x_C)^2) \\ &((x_A + x_B + x_C)(x_B - x_A)^2 - \tilde{\alpha}^2)^2)((x_A + x_B + x_C)(x_B - x_A)^2 - \tilde{\gamma}^2)^2 \\ -&\tilde{\tau}^2((x_A + x_B + x_C)(x_B - x_A)^2 - \tilde{\alpha}^2)^2(x_B - x_A)^2 = 0 \end{aligned}$$

und $y_1 = y_2$ äquivalent ist zu

$$\begin{aligned} &(y_A - y_C)((x_A + x_B + x_C)\tilde{\eta}^2 - \tilde{\alpha}^2)^3((x_A + x_B + x_C)\tilde{\mu}^2 - \tilde{\gamma}^2)^3\tilde{\eta}^3\tilde{\mu}^3 \\ +&\tilde{\beta}(((2x_C - x_A - x_B)\tilde{\eta}^2 + \tilde{\alpha}^2)((x_A + x_B + x_C)\tilde{\eta}^2 - \tilde{\eta}^2)^2 - \tilde{\beta}^2) \\ &\quad ((x_A + x_B + x_C)\tilde{\mu}^2 - \tilde{\gamma}^2)^3\tilde{\mu}^3 \\ -&\tilde{\tau}(((2x_A - x_B - x_C)\tilde{\mu}^2 + \tilde{\gamma}^2)((x_A + x_B + x_C)\tilde{\eta}^2 - \tilde{\gamma}^2)^2 - \tilde{\tau}^2) \\ &\quad ((x_A + x_B + x_C)\tilde{\eta}^2 - \tilde{\alpha}^2)^3\tilde{\eta}^3 = 0 \end{aligned}$$

Diese beide Gleichheiten sind im Ring R_p/I_p zu zeigen. Nach der Vorbemerkung genügt es zu zeigen, daß beide linke Seiten im Ideal I_0 liegen. Daß dies so ist, wurde mittels des Computerprogramms „cocoa“ nachgewiesen. □

Lemma 8.2
Seien $A, B \in E_{a,b}(\overline{K}) \setminus \{O\}$. Ist $A \neq -A, A \neq \pm B, A + A \neq \pm B$ und $A + B \neq \pm A^1$, so gilt

$$(A + A) + B = A + (A + B)$$

Beweis:
Setze $(x_A, y_A) := A$, $(x_B, y_B) := B$ und $(x_1, y_1) := (A + B) + C$ sowie

$(x_2, y_2) := A + (B + C)$. Mit[1]

$$\begin{aligned}
\alpha &:= \frac{3x_A^2 + a}{2y_A} \\
\beta &:= \frac{y_A + y_B - \alpha(3x_A - \alpha^2)}{2x_A + x_B - \alpha^2} \\
\gamma &:= \frac{y_B - y_A}{x_B - x_A} \\
\tau &:= \frac{2y_A - \gamma(2x_A + x_B - \gamma^2)}{2x_A + x_B - \gamma^2}
\end{aligned}$$

erhält man durch Verwenden der Additionsformeln 3 und 3, daß

$$\begin{aligned}
x_1 &= \beta^2 + 2x_A - x_B - \alpha^2 \\
y_1 &= -y_B + \beta(2x_B - 2x_A - \beta^2 + \alpha^2) \\
x_2 &= \tau^2 + x_B - \gamma^2 \\
y_2 &= -y_A + \tau(x_A - x_B - \tau^2 + \gamma^2)
\end{aligned}$$

Setzt man

$$\begin{aligned}
\tilde{\alpha} &:= 3x_A^2 + a \\
\tilde{\beta} &:= (y_A + y_B)8y_A^3 - \tilde{\alpha}(12x_A y_A^2 - \tilde{\alpha}^2) \\
\tilde{\gamma} &:= y_B - y_A \\
\tilde{\tau} &:= 2y(x_B - x_A)^3 - \tilde{\gamma}((2x_A + x_B)(x_B - x_A)^2 - \tilde{\gamma}^2)
\end{aligned}$$

so kann man zeigen, daß $x_1 = x_2$ äquivalent ist zu

$$\begin{aligned}
&(\tilde{\beta}^2(x_B - x_A)^2 + (((2x_A - 2x_B)(x_B - x_A)^2 + \tilde{\gamma}^2)4y_A^2 - \tilde{\alpha}^2(x_B - x_A)^2) \\
&\quad ((2x_A + x_B)4y_A^2 - \tilde{\alpha}^2)^2)((2x_A + x_B)(x_B - x_A)^2 - \tilde{\gamma}^2)^2 \\
&-4\tilde{\tau}^2 y_A^2((2x_A + x_B)4y_A^2 - \tilde{\alpha}^2)^2 = 0
\end{aligned}$$

und $y_1 = y_2$ äquivalent ist zu

$$\begin{aligned}
&(y_A - y_B)((x_B + 2x_A)4y_A^2 - \tilde{\alpha}^2)^3((2x_A + x_B)(x_B - x_A)^2 - \tilde{\gamma}^2)^3 8y_A^3(x_B - x_A)^3 \\
&+\tilde{\beta}(((2x_B - 2x_A)4y_A^2 + \tilde{\alpha}^2)((x_B + 2x_A)4y_A^2 - \tilde{\alpha}^2)^2 - \tilde{\beta}^2) \\
&\quad ((2x_A + x_B)(x_B - x_A)^2 - \tilde{\gamma}^2)^3(x_B - x_A)^3 \\
&-\tilde{\tau}(((x_A - x_B)(x_B - x_A)^2 + \tilde{\gamma}^2)((2x_A + x_B)(x_B - x_A)^2 - \tilde{\gamma}^2)^2 - \tilde{\tau}^2) \\
&\quad ((x_B + 2x_A)4y_A^2 - \tilde{\alpha}^2)^3 8y_A^3 = 0
\end{aligned}$$

[1] Die Nenner in den Termen verschwinden wegen den Einschränkungen bei der Wahl von A und B nicht.

Diese beide Gleichheiten sind im Ring R_p/I_p zu zeigen. Nach der Vorbemerkung genügt es zu zeigen, daß beide linke Seiten im Ideal I_0 liegen. Daß dies so ist, wurde mittels des Computerprogramms „cocoa“ nachgewiesen. □

Lemma 8.3
Sei $A \in E_{a,b}(\overline{K}) \setminus \{O\}$. Ist $A \neq -A$, $A + A \neq -(A + A)$, $(A + A) + A \neq \pm A$ und $A + A \neq \pm A$, so gilt

$$(A + A) + (A + A) = A + (A + (A + A))$$

Beweis:
Setze $(x_A, y_A) := A$, $(x_1, y_1) := (A+A)+(A+A)$, $(x_2, y_2) := A+(A+(A+A))$. Im folgenden sei[1]

$$\begin{aligned}
\alpha &:= \frac{3x_A^2 + a}{2y_A} \\
\beta &:= \frac{3(\alpha^2 - 2x_A)^2 + a}{-2y_A + 2\alpha(3x_A - \alpha^2)} \\
\gamma &:= \frac{-2y_A + \alpha(3x_A - \alpha^2)}{\alpha^2 - 3x_A} \\
\tau &:= \frac{-2y_A + \gamma(\alpha^2 - \gamma^2)}{\gamma^2 - \alpha^2}
\end{aligned}$$

Dann ist

$$\begin{aligned}
x_1 &= \beta^2 - 2\alpha^2 + 4x_A \\
y_1 &= y_A - \alpha(3x_A - \alpha^2) + \beta(3\alpha^2 - 6x_A - \beta^2) \\
x_2 &= \tau^2 - \gamma^2 + \alpha^2 - 2x_A \\
y_2 &= -y_A + \tau(\gamma^2 - \tau^2 - \alpha^2 + 3x_A)
\end{aligned}$$

Setzt man

$$\begin{aligned}
\tilde{\alpha} &:= 3x_A^2 + a \\
\tilde{\beta} &:= 3(\tilde{\alpha}^2 - 8x_A y_A^2)^2 + 16a y_A^4 \\
\tilde{\gamma} &:= -16y_A^4 + \tilde{\alpha}(12x_A y_A^2 - \tilde{\alpha}^2) \\
\tilde{\tau} &:= -16y_A^4(\tilde{\alpha}^2 - 12x_A y_A^2)^3 + \tilde{\gamma}(\tilde{\alpha}^2(\tilde{\alpha}^2 - 12x_A y_A^2)^2 - \tilde{\gamma}^2) \\
\tilde{\eta} &:= -16y_A^4 + 2\tilde{\alpha}(12x_A y_A^2 - \tilde{\alpha}^2)
\end{aligned}$$

[1]Die Nenner in den Termen verschwinden wegen den Einschränkungen bei der Wahl von A nicht.

so ist $x_1 = x_2$ äquivalent zu

$$((\tilde{\beta}^2 + (24x_A y_A^2 - 3\tilde{\alpha}^2)\tilde{\eta}^2)(\tilde{\alpha}^2 - 12x_A y_A^2)^2 + \tilde{\gamma}^2\tilde{\eta}^2)(\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - 12x_A y_A^2)^2)^2 - \tilde{\tau}^2\tilde{\eta}^2 = 0$$

und $y_1 = y_2$ äquivalent zu

$$\begin{aligned}&(\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - 12x_A y_A^2)^2)^3(\tilde{\alpha}^2 - 12x_A y_A^2)^3\\ &((16y_A^4 - \tilde{\alpha}(12x_A y_A^2 - \tilde{\alpha}^2))\tilde{\eta}^3 + \tilde{\beta}((3\tilde{\alpha}^2 - 24x_A y_A^2)\tilde{\eta}^2 - \tilde{\beta}^2))\\ -\tilde{\tau}\tilde{\eta}^3((\tilde{\gamma}^2 - (\tilde{\alpha}^2 - 12x_A y_A^2)^3)(\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - 12x_A y_A^2)^2)^2 - \tilde{\tau}^2) &= 0\end{aligned}$$

Diese beide Gleichheiten sind im Ring R_p/I_p zu zeigen. Nach der Vorbemerkung genügt es zu zeigen, daß beide linke Seiten im Ideal I_0 liegen. Daß dies so ist, wurde mittels des Computerprogramms „cocoa“ nachgewiesen. □

An dieser Stelle würde man nun das ein Lemma erwarten, welches besagt, daß, unter bestimmten Voraussetzungen,

$$(A + B) + (A + B) = A + (B + (A + B))$$

den auch diesen Fall kann man auf eine Rechnung zurückführen, welche man mit dem Computer berechnen könnte:

Sind nämlich $A, B \in E_{a,b}(\overline{K}) \setminus \{O\}$ mit $A \neq \pm B$, $A + B \neq -(A + B)$, $B+(A+B) \neq \pm A$ und $A+B \neq \pm B$. Dann setze $(x_A, y_A) := A, (x_B, y_B) := B$ und $(x_1, y_1) := (A + B) + (A + B)$ sowie $(x_2, y_2) := A + (B + (A + B))$. Mit[1]

$$\begin{aligned}\alpha &:= \frac{y_B - y_A}{x_B - x_A}\\ \beta &:= \frac{3(\alpha^2 - x_A - x_B)^2 + a}{-2y_A + 2\alpha(2x_A + x_B - \alpha^2)}\\ \gamma &:= \frac{-y_A - y_B + \alpha\,(2x_A + x_B - \alpha^2)}{\alpha^2 - x_A - 2x_B}\\ \tau &:= \frac{-y_B - y_A + \gamma(x_B - x_A + \alpha^2 - \gamma^2)}{\gamma^2 - \alpha^2}\end{aligned}$$

erhält man durch Verwenden der Additionsformeln 3 und 4, daß

$$\begin{aligned}x_1 &= \beta^2 - 2\alpha^2 + 2x_A + 2x_B\\ y_1 &= y_A - \alpha(2x_A + x_B - \alpha^2) + \beta(3\alpha^2 - 3x_A - 3x_B - \beta^2)\\ x_2 &= \tau^2 - \gamma^2 + \alpha^2 - 2x_A\\ y_2 &= -y_A + \tau(3x_A + \gamma^2 - \alpha^2 - \tau^2)\end{aligned}$$

[1]Die Nenner in den Termen verschwinden wegen den Einschränkungen bei der Wahl von A und B nicht.

Setzt man

$$\begin{aligned}
\tilde{\alpha} &:= y_B - y_A \\
\tilde{\beta} &:= 3(\tilde{\alpha}^2 - (x_A + x_B)(x_B - x_A)^2)^2 + a(x_B - x_A)^4 \\
\tilde{\gamma} &:= (-y_A - y_B)(x_B - x_A)^3 + \tilde{\alpha}((2x_A + x_B)(x_B - x_A)^2 - \tilde{\alpha}^2) \\
\tilde{\tau} &:= (-y_B - y_A)(x_B - x_A)^3(\tilde{\alpha}^2 - (x_A + 2x_B)(x_B - x_A)^2)^3 \\
&\quad +\tilde{\gamma}(((x_B - x_A)^3 + \tilde{\alpha}^2)(\tilde{\alpha}^2 - (x_A + 2x_B)(x_B - x_A)^2)^2 - \tilde{\gamma}^2) \\
\tilde{\eta} &:= -2y_A(x_B - x_A)^3 + 2\tilde{\alpha}((2x_A + x_B)(x_B - x_A)^2 - \tilde{\alpha}^2)
\end{aligned}$$

so kann man zeigen, daß $x_1 = x_2$ äquivalent ist zu

$$\begin{aligned}
&((\tilde{\beta}^2 + ((4x_A + 2x_B)(x_B - x_A)^2 - 3\tilde{\alpha}^2)\tilde{\eta}^2)(\tilde{\alpha}^2 - (x_A + 2x_B)(x_B - x_A)^2)^2 + \tilde{\gamma}^2\tilde{\eta}^2) \\
&(\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - (x_A + 2x_B)(x_B - x_A)^2)^2)^2 - \tilde{\tau}^2\tilde{\eta}^2 = 0
\end{aligned}$$

und $y_1 = y_2$ äquivalent ist zu

$$\begin{aligned}
&(\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - (2x_B + x_A)(y_B - y_A)^2)^2)^3(\tilde{\alpha}^2 - (2x_B + x_A)(y_B - y_A)^2)^3 \\
&((2y_A(y_B - y_A)^3 - \tilde{\alpha}((2x_A + x_B)(y_B - y_A)^2 - \tilde{\alpha}^2))\tilde{\eta}^3 \\
&\quad +\tilde{\beta}((3\tilde{\alpha}^2 - (3x_A + 3x_B)(y_B - y_A)^2)\tilde{\eta}^2 - \tilde{\beta}^2)) \\
&-\tilde{\eta}^3\tilde{\tau}(((3x_A(y_B - y_A)^2 - \tilde{\alpha}^2)(\tilde{\alpha}^2 - (2x_B + x_A)(y_B - y_A)^2)^2 + \tilde{\gamma}^2) \\
&\quad (\tilde{\gamma}^2 - \tilde{\alpha}^2(\tilde{\alpha}^2 - (2x_B + x_A)(y_B - y_A)^2)^2 - \tilde{\tau}^2) = 0
\end{aligned}$$

Tatsächlich konnte die erste Gleichheit mit Hilfe des Computerprogramms „Cocoa“ gezeigt werden. Die linke Seite in der 2. Gleichung konnte, ebenfalls mit „Cocoa“ berechnet werden, es handelt sich dabei um ein Polyonom mit etwa 170.000 Monomen. Der Versuch mit Hilfe des Programms zu Zeigen, daß der Term in I_0 liegt, wurde nach 43 Stunden Laufzeit auf einem Pentium II mit 266 Mhz abgebrochen.

Der Beweis, daß $(A + B) + (A + B) = A + (B + (A + B))$, erfolgt, ohne Rückgriff auf den Computer, im Beweis zu Satz 8.13.

8.2 Vorbereitende Lemmas

Lemma 8.4
Seien $A, B \in E_{a,b}(\overline{K})$, dann gilt

$$-A - B = -(A + B)$$

Beweis:
Die Fälle $A = O$ bzw. $B = O$ sind trivial. Es seien daher $A =: (x_A, y_A)$ und $B =: (x_B, y_B) \in E_{a,b}(\overline{K}) \setminus \{O\}$. Ist $A = -B$ so folgt

$$-A - B = -A + A = O = -(B - B) = -(B + A)$$

Sei im folgenden $A \neq -B$. Für $C, D \in E(\overline{K}) \setminus \{O\}$ mit $C \neq -D$ bezeichne $\alpha(C, D)$ die Steigung der (affinen) Geraden $g(C, D)$. Wie man sich leicht vergewissern kann, ist $\alpha(A, B) = -\alpha(-A, -B)$. Es folgt

$$\begin{aligned}-(A + B) &= (\alpha(A, B)^2 - x_A - x_B, -(y_A + \alpha(A, B)(2x_A + x_B - \alpha(A, B)^2) = \\ &= (\alpha(-A, -B)^2 - x_A - x_B, -y_A + \alpha(-A, -B)(2x_A + x_B - \alpha(-A, -B)^2) = \\ &= -A - B\end{aligned}$$

□

Lemma 8.5
Seien $A, B \in E_{a,b}(\overline{K})$. Ist $A + B = A - B$ und $A \neq -A$, so folgt $B = -B$.

Beweis:
Für $A = O$ bzw. $B = O$ ist die Aussage trivialerweise richtig. Es seien daher $A =: (x_A, y_A), B =: (x_B, y_B) \in E_{a,b}(\overline{K}) \setminus \{O\}$ mit $A + B = A - B$. Man erhält folgende Fallunterscheidung:

1. Fall: $A = B$.
   Dann ist $A + A = A + B = A - B = O$, d.h. $A = -A$. Es ist also $B = A = -A = -B$.

2. Fall: $A = -B$.
   Dann ist $A + A = A - B = A + B = A - A = O$, d.h. $A = -A$. Es ist also $B = -A = A = -B$.

3. Fall: $A \neq \pm B$.
   Dann folgt

$$\begin{aligned} & \left(\tfrac{y_B - y_A}{x_B - x_A}\right)^2 - x_A - x_B &&= \left(\tfrac{-y_B - y_A}{x_B - x_A}\right)^2 - x_A - x_B \\ \Rightarrow \quad & y_B^2 + y_A^2 - 2y_A y_B &&= y_B^2 + y_A^2 + 2y_A y_B \\ \Rightarrow \quad & -2y_A y_B &&= 2y_A y_B \end{aligned}$$

   Da $A \neq -A$ ist $y_A \neq 0$. Aus $\operatorname{Char} \overline{K} > 3$ folgt $y_B = 0$.

□

Lemma 8.6 (Eindeutigkeit des neutralen Elements)
Seien $A, B \in E_{a,b}(\overline{K})$. Ist $A + B = A$, so folgt $B = O$.

Beweis:
Für $A = O$ ist die Aussage trivialerweise richtig. Ist $A = -B$, so folgt $B = -A = -A - B = B - B = O$. Angenommen es gibt $A, B \neq O$ mit $A \neq -B$ und $A + B = A$. Setze $(x_A, y_A) := A$, $(x_B, y_B) := B$ und $(x_C, y_C) := A + B = A = (x_A, y_A)$. Je nachdem, ob $A \neq B$ oder $A = B$ setze $\alpha := \left(\frac{y_B - y_A}{x_B - x_A}\right)$ bzw. $\alpha := \left(\frac{3x_A^2 + a}{2y_A}\right)$. Dann gilt nach Formel 3 bzw. 4, daß

$$y_A = y_C = -y_A + \alpha \underbrace{(x_A - x_C)}_{=0} = -y_A$$

d.h. $y_A = 0$, also $A = -A$. Es folgt

$$A + B = A = -A = -A - B = A - B$$

Nach Lemma bedeutet dies aber, daß $B = -B$, d.h. $y_B = O$. Es ist insbesondere $A \neq B$, denn sonst wäre $B = A = A + B = A + A = A - A = O$. Nach Formel 3 gilt dann wegen $y_A = y_B = 0$, daß

$$x_A = x_C = \left(\frac{y_B - y_A}{x_B - x_A}\right)^2 - x_A - x_B = -x_A - x_B$$

Das Polynom $X^3 + aX + b$ besitzt also eine Nullstellen bei x_A und eine bei $x_B = -x_A - x_A$. Bekanntlich ist der Koeffizient des 2. höchsten Gliedes eines normierten Polynoms gleich der Summe der Nullstellen; in diesem Fall gilt also für die 3. Nullstelle x_0, daß

$$0 = x_0 + x_A + x_B = x_0 - x_A$$

Es ist also $x_0 = x_A$, d.h. das Polynom $X^3 + aX + b$ hat eine doppelte Nullstelle, was aber wegen $4a^3 + 27b^2 \neq 0$ ausgeschlossen ist. Die Annahme $B \neq O$ ist damit zum Widerspruch geführt. □

Lemma 8.6 wird im folgenden ohne Verweis zitiert.

Lemma 8.7
Sei $A \in E_{a,b}(\overline{K})$ mit $A \neq -A$ und $A + A \neq -A$, dann gilt

$$(A + A) - A = A$$

Beweis:
Für $A = O$ ist die Aussage klar. Sei im folgenden $A \neq O$, setze $(x_A, y_A) := A$ und $(x_{AA}, y_{AA}) := A + A$, es ist

$$\begin{aligned} x_{AA} &= \left(\frac{3x_A^2 + a}{2y_A}\right)^2 - 2x_A \\ y_{AA} &= -y_A + \left(\frac{3x_A^2 + a}{2y_A}\right)(x_A - x_{AA}) \end{aligned}$$

für $(x_E, y_E) := (A + A) - A$ gilt dann, wegen $A + A \neq \pm A$, daß

$$x_E = \left(\frac{y_{AA} + y_A}{x_{AA} - x_A}\right)^2 - x_{AA} - x_A = \underbrace{\left(\frac{3x_A^2 + a}{2y_A}\right)^2 - 2x_A}_{=x_{AA}} - x_{AA} + x_A = x_A$$

$$y_E = y_A + \left(\frac{y_{AA} + y_A}{x_{AA} - x_A}\right)\underbrace{(x_A - x_E)}_{=0} = y_A$$

Es ist also $(A + A) - A = A$. □

Lemma 8.8
Seien $A, B \in E_{a,b}(\overline{K})$, dann gilt

$$A + B = -A \Rightarrow B = -A - A$$

Beweis:
Für $A = O$ bzw. $B = O$ ist die Aussage trivialerweise richtig. Im folgenden seien $A =: (x_A, y_A), B =: (x_B, y_B) \in E_{a,b}(\overline{K}) \setminus \{O\}$. Es werden vier Fälle unterschieden:

1. Fall: $A = B$.
   Aus $A + B = -A$ folgt $B = A = -A - B = -A - A$.

2. Fall: $A = -B$.
   Aus $-A = A + B$ folgt $-A = A - A = O$, d.h. $A = O, B = O$. Daraus folgt $B = -A - A$.

3. Fall: $A = -A$.
   Dann ist $-A + B = -A$, aus Lemma 8.6 folgt $B = O$. Daraus ergibt sich $B = O = A - A = -A - A$.

4. Fall: $A \neq \pm B$ und $A \neq -A$.
Aus $-A = A + B$ folgt insbesondere:

$$\begin{aligned} & x_A = \left(\frac{y_A - y_B}{x_A - x_B}\right)^2 - x_A - x_B \\ \Rightarrow \quad & 2x_A + x_B = \frac{y_A^2 + x_B^3 + ax_B + b - 2y_A y_B}{x_A^2 + x_B^2 - 2x_A x_B} \\ \Rightarrow \quad & 2y_A y_B = y_A^2 + ax_B + b - 2x_A^3 + 3x_A^2 x_B \end{aligned}$$

Auf beiden Seiten quadriert ergibt sich

$$4x_B^3 y_A^2 - x_B^2(3x_A^2+a)^2 + x_B(2a^2 x_A + 6x_A^5 - 12bx_A^2) - (y_A^2 - b)^2 + 4ax_A^4 + 8bx_A^3 = 0$$

Dies ist, wie man nachrechnen kann, äquivalent zu

$$\left(x_B - \left(\left(\frac{3x_A^2 + a}{2y_A}\right)^2 - 2x_A\right)\right)(x_B - x_A)^2 = 0$$

Der Fall $x_A = x_B$ ist aber wegen $A \neq \pm B$ ausgeschlossen. Es folgt

$$x_B = \left(\frac{3x_A^2 + a}{2y_A}\right)^2 - 2x_A$$

d.h. es ist $B = A+A$ oder $B = -(A+A) = -A-A$. Wäre $A+A = -A$, so folgte $B = \pm(A + A) = \pm A$. Es ist also $A + A \neq -A$. Nach Lemma 8.7 ist dann $B = -A - A$ eine Lösung der Gleichung $A + B = -A$. Ist $B = A + A$ auch eine Lösung der Gleichung, so hat man den Fall, daß $A + B = A + (A + A) = -A = A - (A + A) = A - B$. Wegen $A \neq -A$ folgt aus Lemma 8.5, daß $B = -B$. Es ist also $B = -B = -A - A$.

□

Lemma 8.9 (Kürzungsregel)
Seien $A, B, \tilde{B} \in E_{a,b}(\overline{K})$, dann gilt

$$A + B = A + \tilde{B} \Rightarrow B = \tilde{B}$$

Beweis:
Seien also $A, B, \tilde{B} \in E_{a,b}(\overline{K})$ mit $A + B = A + \tilde{B}$. Sofern $A, B, \tilde{B}$ jeweils nicht der unendlich ferne Punkt O sind, seien $(x_A, y_A), (x_B, y_B), (\tilde{x}_B, \tilde{y}_B)$ die zugehörigen Koordinaten. Folgende Fälle werden unterschieden:

1. Fall: $A = O$.
   Es folgt sofort $B = \tilde{B}$.

Sei im folgenden $A \neq O$.

(2) Fall: $B = O$ oder $\tilde{B} = O$, o.B.d.A. sei $\tilde{B} = O$.
Dann gilt $A + B = A$, aus Lemma 8.6 folgt $B = O$.

Seien im folgenden $B, \tilde{B} \neq O$.

(3) Fall: $A + B = A + \tilde{B} = O$.
Nach der Beschreibung der Addition in Satz 5.3 besitzt ein Punkt A genau ein Inverses, d.h. $\tilde{B} = -A = B$.

Seien im folgenden $B, \tilde{B} \neq -A$.

(4) Fall: $A + B = A + \tilde{B} = -A$.
Aus Lemma 8.8 folgt dann $B = -A - A$ und $\tilde{B} = -A - A$.

(5) Fall: $A + B \neq -A$.
Sei α bzw. $\tilde{\alpha}$ die Steigung der Geraden $g(A, B)$ bzw. $g(A, \tilde{B})$. Es folgt aus $A + B = A + \tilde{B} =: (x_C, y_C)$, daß

$$\begin{array}{rcccl} x_C & = & \alpha^2 - x_A - x_B & = & \tilde{\alpha}^2 - x_A - \tilde{x}_B \\ y_C & = & -y_A + \alpha(x_A - x_C) & = & -y_A + \tilde{\alpha}(x_A - x_C) \end{array}$$

Wegen $A + B \neq \pm A$ ist $x_A \neq x_C$, es folgt also aus der 2. Gleichung, daß

$$\alpha = \tilde{\alpha}$$

Mittels der 1. Gleichung erhält man dann $x_B = \tilde{x}_B$, d.h. $B = -\tilde{B}$, oder $B = \tilde{B}$. Es können zwei Fälle auftreten:

(a) Fall $A = -A$:
Es ist $B, \tilde{B} \neq -A = A$, daraus folgt, daß die Addition $A + B$ und $A + \tilde{B}$ durch die Additionsformel 3 beschrieben werden können. Es ist also

$$\frac{y_B - y_A}{x_B - x_A} = \alpha = \tilde{\alpha} = \frac{\tilde{y}_B - y_A}{\tilde{x}_B - x_A}$$

Aus $x_B = \tilde{x}_B$ ergibt sich $y_B = \tilde{y}_B$, also $B = \tilde{B}$.

(b) $A \neq -A$:
Angenommen es ist $B = -\tilde{B}$, dann folgt, daß $A + B = A + \tilde{B} = A - B$. Lemma 8.5 besagt dann, wegen $A \neq -A$, daß $B = -B$, d.h. $B = \tilde{B}$.

□

Lemma 8.10
Seien $A, B \in E_{a,b}(\overline{K})$.

$$(A + B) - B = A$$

Beweis:
Für $A = O$ oder $B = O$ ist die Aussage trivialerweise richtig. Seien daher $A =: (x_A, y_A), B =: (x_B, y_B) \in E_{a,b}(\overline{K}) \setminus \{O\}$. Folgende Fälle werden unterschieden:

1. Fall: $A = -B$.
   Es folgt $(A + B) - B = -B = A$.

Sei im folgenden $A \neq -B$.

(2) Fall: $A + B = -B$.
Nach Lemma 8.8 gilt $A = -B - B$, es folgt $(A+B)-B = -B-B = A$.

Sei im folgenden $A + B \neq -B$.

(3) Fall: $A = B$.
Die Aussage folgt aus Lemma 8.7.

(4) Fall: $A \neq B$.
Es sei $A + B =: (x_{AB}, y_{AB})$, dann ist

$$\begin{aligned} x_{AB} &= \left(\frac{y_A - y_B}{x_A - x_B}\right)^2 - x_A - x_B \\ y_{AB} &= -y_B + \left(\frac{y_A - y_B}{x_A - x_B}\right)(x_B - x_{AB}) \end{aligned}$$

für $(x_E, y_E) := (A + B) - B$ gilt dann

$$x_E = \left(\frac{y_{AB} + y_B}{x_{AB} - x_B}\right)^2 - x_{AB} - x_B = \underbrace{\left(\frac{y_A - y_B}{x_A - x_B}\right)^2 - x_A - x_B}_{=x_{AB}} - x_{AB} + x_A = x_A$$

$$y_E = y_B + \left(\frac{y_{AB} + y_B}{x_{AB} - x_B}\right)(x_B - x_E) = y_B + \left(\frac{y_A - y_B}{-x_A + x_B}\right)(x_B - x_A) = y_A$$

Es ist also $(A + B) - B = A$.

□

Korollar 8.11
Seien $A, B, C \in E_{a,b}(\overline{K})$, dann gilt

$$A + B = C \Rightarrow A = C - B$$

8.3 Beweis der Assoziativität der Verknüpfung

Lemma 8.12
Seien $A, B, C \in E_{a,b}(\overline{K})$. Ist

1. $(A + B) \neq C$ und $A \neq (B + C)$, oder
2. $A = B$, oder $B = C$, oder $A = C$, oder
3. $O \in \{A, B, C, A + B, B + C, (A + B) + C, A + (B + C)\}$

so gilt

$$(A + B) + C = A + (B + C)$$

Beweis:
Die Fälle $A = O$, $B = O$, $C = O$ und $A = C$ sind trivial. Seien daher im folgenden $A =: (x_A, y_A), B =: (x_B, y_B), C =: (x_C, y_C) \in E_{a,b}(\overline{K}) \setminus \{O\}$ mit $A \neq C$. Folgende Fälle werden unterschieden:

1. Fall: $A = -B$ oder $C = -B$, o.B.d.A. sei $A = -B$.
   Nach Lemma 8.10 ist $C = -B + (B + C)$. Es folgt, daß

$$(A + B) + C = C = -B + (B + C) = A + (B + C)$$

Seien im folgenden $A \neq -B$ und $B \neq -C$.

(2) Fall: $A + B = -C$ oder $B + C = -A$, o.B.d.A. sei $A + B = -C$.
Es gilt nach Lemma 8.10, daß $-A = B + (-B - A)$, es ist also

$$(A+B)+C = -C+C = O = A-A = A+(B+(-B-A)) = A+(B+C)$$

Damit ist der 3. Fall der Behauptung bewiesen.
Es seien im folgenden $A + B \neq -C$ und $B + C \neq -A$.

(3) Fall: $A = B$ oder $B = C$, o.B.d.A. sei $A = B$.
Es ist zu zeigen, daß $(A + A) + C = A + (A + C)$. Ist $C \neq A + A$, so folgt dies aus Lemma 8.2 [1], ist $C = A + A$, so folgt dies aus Lemma 8.3 [2].

Damit ist der 2. Fall der Behauptung bewiesen.

(4) Fall: $A \neq B$ und $B \neq C$.
Dieser Fall folgt aus Lemma 8.1.

□

Satz 8.13
Die Verknüpfung „+“ ist assoziativ, d.h. für alle $A, B, C \in E_{a,b}(\overline{K})$ gilt

$$(A + B) + C = A + (B + C)$$

Beweis:
Es ist nur noch der Fall für die $A, B, C \in E_{a,b}(\overline{K})$ zu beweisen, welche nicht die Bedingungen von Lemma 8.12 erfüllen. Es sei also $A + B = C$ oder $B + C = A$, o.B.d.A. sei $A + B = C$. Es ist also zu zeigen, daß

$$(A + B) + (A + B) = A + (B + (A + B))$$

wobei $A, B, C, A + B, B + C, (A + B) + C, A + (B + C) \neq O$ und A, B, C paarweise verschieden sind. Es können folgende Fälle auftreten:

1. Fall: $(A + B) + (A + B) = -A$.
Nach Korollar 8.11 ist $A + B = (-B - A) - A$, aus dem 2. Fall von Lemma 8.12 folgt, daß $(-B - A) - A = -B + (-A - A)$. Es ist also $A + B = -B + (-A - A)$. Damit ergibt sich:

$$A + (B + (A + B)) = A + (B + (-B + (-A - A))) = A + (-A - A) = \\ = -A = (A + B) + (A + B)$$

[1] Der Rückgriff auf das mit dem Computer bewiesene Lemma 8.2 ist vermeidbar. Man kann nämlich mit Lemma 8.1 und dem separaten abhandeln der Spezialfälle zeigen, daß $(C + (A + A)) - A = C + ((A + A) - A)$. Daraus folgt

$$(C + (A + A)) - A = C + ((A + A) - A) = C + A = (C + A) + A) - A$$

Dies ist nach Lemma 8.9 gleichbedeutend mit $C + (A + A) = (C + A) + A$.

[2] Wie beim Fall $C \neq A + A$ kann man auch beim Fall $C = A + A$ den Rückgriff auf ein mit dem Computer bewiesenes Lemma vermeiden. Hat man nämlich gezeigt, daß $((A + A) + (A + A)) - A = (A + A) + ((A + A) - A)$, so folgt daraus

$$((A + A) + (A + A)) - A = (A + A) + A = ((A + A) + A) + A) - A$$

2. Fall: $(A+B)+(A+B) \neq -A$.
   Aus dem 2. Fall von Lemma 8.12 folgt, daß

$$((A+B)+(A+B))-A = (A+B)+((A+B)-A)$$

   Damit erhält man, daß

$$\begin{aligned}((A+B)+(A+B))-A = (A+B)+((A+B)-A) = (A+B)+B = \\ = (A+(B+(A+B)))-A\end{aligned}$$

   Aus Lemma 8.9 folgt $(A+B)+(A+B) = A+(B+(A+B))$.

□

Literaturverzeichnis

[AGP] W. R. Alford, A. Granville, C. Pomerance, *There are infinitely many Carmichael Numbers*, Annals of Mathematics **139** (1994), S. 703-722

[BRK] A. Bartholomé, J. Rung, H. Kern, *Zahlentheorie für Einsteiger*, Vieweg Verlag, Braunschweig–Wiesbaden (1995)

[Bressoud] D. M. Bressoud, *Factorization and Primality Testing*, Springer Verlag, Berlin–Heidelberg–New York (1989)

[CEP] E. R. Canfield, P. Erdös, C. Pomerance, *On a problem of Oppenheim concerning „Factorisatio Numerorum“*, Journal of Number Theory **17** (1983), S. 1-28

[Cohen] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer Verlag, Berlin–Heidelberg–New York (1993)

[Coppersmith] D. Coppersmith, *Modification to the number field sieve*, Journal of Cryptology **6** (1993), S. 169-180

[Davenport] H. Davenport, *Multiplicative Number Theory*, Second Edition, Springer Verlag, Berlin–Heidelberg–New York (1980)

[Deuring] M. Deuring, *Die Typen der Multiplikatorenringe elliptischer Funktionenkörper*, Abh. Math. Seminar der Hansischen Universität **14** (1941), S. 197-272

[DH] W. Diffie, M. E. Hellmann, *New directions in cryptography*, IEEE Transactions on Information Theory IT **22** (1976), S. 644-654

[Forster] O. Forster, *Algorithmische Zahlentheorie*, Vieweg Verlag, Braunschweig–Wiesbaden (1996)

[Gerhard] J. Gerhard, *Elliptische Kurven – Theorie und Anwendungen*, Studienarbeit im Fach Informatik, Lehrstuhl für Automatentheorie und formale Sprachen (Informatik I), Universität Erlangen–Nürnberg (1993)

[Gordon87] D. Gordon, *Pseudoprimes on Elliptic Curves*, Proc. Internat. Number Theory Conference, Laval (1987)

[Gordon89] D. Gordon, *On the Number of Elliptic Pseudoprimes*, Mathematics of Computation **52** (1989), S. 231-245

[GP] D. Gordon, C. Pomerance, *The distribution of Lucas and elliptic pseudoprimes*, Mathematics of Computation **57** (1991), S. 825-838

[HS] F. H. Hinsley, A. Stripp, *Codebreakers, the inside story of Bletchley Park*, Oxford, Oxford (1993)

[Hardy] G. H. Hardy, *A mathematician's apology*, Cambridge University Press, Cambridge (1940)

[HW68] G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers*, 4. Auflage, Oxford, Oxford (1968)

[KO] A. Karatsuba, Y. Ofman, *Multiplication of multidigit numbers on automata*, Dokl. Akad. Nauk USSR 145 (1962) S. 293-294 (Englische Übersetzung in Soviet Physics Doklady 7 (1963) S. 595-596)

[KTP] D. E. Knuth, L. Trabb-Pardo, *Analysis of a Simple Factorization Algorithm*, Theoretical Computer Science **3** (1976), S. 321-348

[Koblitz] N. Koblitz, *A course in Number Theory and Cryptography*, 2. Auflage, Springer Verlag, Berlin–Heidelberg–New York (1994)

[Kunz94] E. Kunz, *Algebra*, 2. Auflage Vieweg Verlag, Braunschweig–Wiesbaden (1994)

[Kunz95] E. Kunz, *Ebene algebraische Kurven*, Der Regensburger Trichter, Band 23, Regensburg (1995)

[Lenstra87] H. W. Lenstra, *Factoring integers with elliptic curves*, Annals of Mathematics **126** (1987), S. 649-673

[LL87] A. Lenstra, H. W. Lenstra, *Algorithms in number theory*, Technical Report 87-008, University of Chicago (1987)

[LL90] A. Lenstra, H. W. Lenstra, *Algorithms in number theory*, Handbook of Theoretical Computer Science, Elsevier Science Publishers B.V. (1990), S. 673-715

[LL93] A. Lenstra, H.W. Lenstra, *The development of the number field sieve*, Springer Verlag, Berlin–Heidelberg–New York (1993)

[MB] M. A. Morrison, J. Brillhart, *A method of factorization and the factorization of* $\mathbb{F}_7$, Mathematics of Computation **29** (1975), S. 183-205

[Miller] G. L. Miller, *Riemann's hypothesis and tests for primality*, Proc. 7th Annual ACM Symposium on the Theory of Computing, S. 234-239

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

[Pomerance81] C. Pomerance, *On the distribution of pseudo primes*, Mathematics of Computation **37** (1981), S. 587-593

[Pomerance90] C. Pomerance, Editor, *Cryptology and Computational Number Theory*, American Mathematical Society (1990)

[Pomerance96] C. Pomerance, *A tale of two sieves*, Notices of the American Mathematical Society (1996), S. 1473-1485

[Ribenboim] P. Ribenboim, *The New Book of Prime Number Records*, Springer Verlag, Berlin–Heidelberg–New York (1995)

[Riesel] H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, Boston–Basel–Stuttgart (1985)

[RSA] R. L. Rivest, L. M. Adleman, A. Shamir, *A method for obtaining digital signatures and public–key cryptosystems*, Communications of the ACM **21** (1978), S. 120-126

[Schoof] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod p*, Mathematics of Computation **44** (1985), S. 483-494

[Silverman] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer Verlag, Berlin–Heidelberg–New York (1986)

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel benützt habe.

Regensburg, den 4. November

Stefan Friedl