# 1 Einleitung

Seit der Erfindung des RSA, hat die Faktorisierung von großen Zahlen eine ganz neue Bedeutung gewonnen, mit ihr steht und fällt die Sicherheit des RSA. Es ist nämlich recht leicht, zwei sehr große Zahlen zu multiplizieren, aber außerordentlich schwer, eine sehr große Zahl in ihre Primfaktoren zu zerlegen, solange die Faktoren nicht zu klein sind oder bei nur zwei Faktoren diese nicht zu dicht beisammen liegen. Dann hat man nämlich sehr schnell durch probieren eine Lösung gefunden. Wie das bei kleinen Primzahlen geht, ist leicht nachzuvollziehen, wie das bei zwei dicht zusammenliegenden Faktoren geht, sehen wir im nächsten Kapitel. Es gibt natürlich auch noch andere Bedingungen, bei denen man sehr schnell zu einer Faktorisierung gelangt. Diese haben hier aber keine so große Bedeutung.

Bevor wir uns mit der ersten Methode beschäftigen, möchte ich erst noch ein par Notationen angeben.

1. $\lfloor r \rfloor$ ist die größte ganze Zahl, die $\leq r$ ist (nach unten runden).
2. $\lceil r \rceil$ ist die kleinste ganze Zahl, die $\geq r$ ist (nach oben runden).
3. $\pi(x)$ ist die Anzahl der Primzahlen $\leq x$.
4. $f(n) = \mathrm{O}(g(n))$, genau dann wenn ein $m \in \mathbb{N}$ und $c \in \mathbb{R}$ existiert, so daß für alle $n \geq m$ gilt: $f(n) \leq c \cdot g(n)$.
5. $f(n) = \mathrm{o}(g(n))$, genau dann wenn $\frac{f(n)}{g(n)} \xrightarrow[n\to\infty]{} 0$ ist.
6. $$\left(\frac{a}{b}\right) = \begin{cases} 0 & : \quad b|a \\ 1 & : \quad a \equiv z^2 \pmod{b} \quad \text{für ein } z \in \mathbb{Z} \\ -1 & : \quad \text{sonst} \end{cases}$$

# 2 Fermat-Faktorisierung

Die Fermat-Faktorisierung macht sich zu nutze, daß man das Produkt zweier Zahlen auch als Differenz zweier Quadrate darstellen kann, welche manchmal einfacher zu bestimmen ist, und daß es zwischen den beiden Darstellungen eine einfache Beziehung gibt.

**2.1 Satz**
*Für eine positive Zahl $n$ sind die zwei folgenden Darstellungen äquivalent:*

1. $n = ab$ *mit* $a \geq b > 0$
2. $n = m^2 - d^2$ *mit* $m > d \geq 0$

*Die Beziehung zwischen den beiden Darstellungen beschreiben die Gleichungen*

$$m = \frac{a+b}{2}, \quad d = \frac{a-b}{2}; \qquad a = m + d, \quad b = m - d.$$

**Beweis:**
Es gilt $n = ab = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$ und $n = m^2 - d^2 = (m+d)(m-d)$. Hieraus kann man direkt die Beziehungen zwischen $m$, $d$, $a$ und $b$ ablesen. □

Wenn für $n = ab$ $a$ und $b$ nahe beieinander liegen, dann ist $d$ ziemlich klein und $m$ ist leicht größer als $\sqrt{n}$. In diesem Fall probiert man einfach für $m$ alle ganzen Zahlen durch beginnend mit $\lceil\sqrt{n}\rceil$ bis $m^2 - n$ eine Quadratzahl $d^2$ ist.

Im folgenden nehmen wir an, daß $n$ nie eine Quadratzahl ist, so daß wir uns über triviale Faktorisierungen keine Gedanken machen müssen.

Falls $a$ und $b$ weit auseinander liegen, braucht man auch mit der Fermat-Faktorisierung sehr viele Schritte bis man eine Faktorisierung erhält. Es gibt eine Verallgemeinerung der Fermat-Faktorisierung, die im allgemeinen in solchen Fällen besser funktioniert. Wir wählen ein kleines $k$ und testen $m = \lceil\sqrt{kn}\rceil, \lceil\sqrt{kn}\rceil + 1$, usw. bis $m^2 - kn$ eine Quadratzahl $d^2$ ist. Dann ist $(m+d)(m-d) = kn$ und somit haben $(m+d)$ und $kn$ einen nichttrivialen gemeinsamen Teiler, falls $n \neq m+d$, also haben auch $(m+d)$ und $n$ einen nichttrivialen gemeinsamen Teiler, da ansonsten $n = m - d < m + d = k$ gelten würde, was der Wahl einer kleinen Zahl für $k$ widersprechen würde. Dann muß man nur noch den $\mathrm{ggt}(m+d, n)$ berechnen und erhält eine Faktorisierung.

**2.2 Beispiel**
Wir suchen die Faktorisierung von $n = 141467$. Wir fangen an mit $m = \lceil\sqrt{n}\rceil = 377, 378, \ldots$, nach ein paar Versuchen ohne Erfolg versuchen wir $m = \lceil\sqrt{3n}\rceil = 652, 653, \ldots$. Nach kurzer Zeit finden wir $655^2 - 3 \cdot 141467 = 68^2$ und $\mathrm{ggt}(655 + 68, 141467) = 241$, woraus wir $141467 = 241 \cdot 587$ folgern. Der Grund dieses schnellen Erfolges liegt in der Tatsache, daß es eine Faktorisierung $n = ab$ gibt, wobei $b$ in der Nähe von $3a$ liegt. Man erhält dann nämlich $3n = (3a)(ca)$, mit $c \approx 3$, d.h. für $3n$ ist $d$ ziemlich klein und daher liegt $m$ in der Nähe von $\sqrt{3n}$, was dann auch schnell gefunden ist. Mit der normalen Fermat-Faktorisierung hätten wir 38 Durchgänge gebraucht.

# 3 Faktorbasis

Eine weitere Verbesserung der Fermat-Faktorisierung liefert die Verwendung einer Faktorbasis. Dazu benutzt man folgendes Resultat.

**3.1 Satz**
*Für $n, t, s \in \mathbb{N}$, mit $t^2 \equiv s^2 \pmod{n}$ und $t \not\equiv \pm s \pmod{n}$, ist $a := \mathrm{ggt}(t+s, n)$ ein nichttrivialer Teiler von $n$ und für $b := n/a$ gilt $b \mid \mathrm{ggt}(t-s, n)$.*

**Beweis:**
Sei also $n, t, s \in \mathbb{N}$, mit $t^2 \equiv s^2 \pmod{n}$ und $t \not\equiv \pm s \pmod{n}$. Dann gilt $n \mid t^2 - s^2 = (t+s)(t-s)$, und da $n$ weder $t+s$ noch $t-s$ teilt ($t \not\equiv \pm s \pmod{n}$), ist $\text{ggt}(t+s, n)$ ein nichttrivialer Teiler von $n$. Daß $b$ den $\text{ggt}(t-s, n)$ teilt ist klar, da in $b$ die gemeinsamen Primfaktoren von $t+s$ und $n$ eliminiert worden sind. Also sind die restlichen Primfaktoren von $b$ auch in $t-s$ enthalten. Und da $b$ gemäß Konstruktion $n$ teilt, teilt $b$ auch den $\text{ggt}(t-s, n)$. □

Das Problem hierbei ist natürlich, daß man zuerst passende $t$ und $s$ finden muß, was für große $n$ ziemlich lange dauern kann, wenn man einfach nur probiert. Also muß man sich überlegen, wie man die Suche beschleunigen kann. Die Idee ist, daß man sich eine Menge $M := \{b_1, b_2, \ldots, b_k\}$ wählt, so daß $a_i := b_i^2 \bmod n$ ein Produkt aus Potenzen von kleinen Primzahlen ist und es eine Untermenge von $M$ gibt, so daß das Produkt der entsprechenden $a_i$ ein Quadrat ist.

**3.2 Definition**
*Der* kleinste absolute Rest *einer Zahl $a$ modulo $n$ ist die Zahl $z \in \mathbb{Z} \cap \left[\lfloor -n/2 \rfloor + 1, \lfloor n/2 \rfloor\right]$, für die $z \equiv a \pmod{n}$ gilt.*
*Dafür schreiben wir* $a \bmod n$.

**3.3 Definition**
*Eine* Faktorbasis *ist eine Menge $B = \{p_1, p_2, \ldots, p_h\}$ von verschiedenen Primzahlen, wobei $p_1$ auch $-1$ sein darf.*
*Eine Zahl $z$ ist eine* $B$-Zahl *(für gegebenes $n$), wenn sich $z \bmod n$ als Produkt von Zahlen aus $B$ schreiben läßt.*

**3.4 Beispiel**
Für $n = 21$ und $B = \{-1, 2, 3\}$ ist 33 eine $B$-Zahl, da $33 \bmod 21 = -9 = -1 \cdot 3 \cdot 3$ ist. Außerdem sind die Quadrate von 5 und 6 $B$-Zahlen, da $5^2 \bmod 21 = 25 \bmod 21 = 4 = 2 \cdot 2$ und $6^2 \bmod 21 = 36 \bmod 21 = -6 = -1 \cdot 2 \cdot 3$. Aber 16 ist keine $B$-Zahl, da $16 \bmod 21 = -5 = -1 \cdot 5$ ist.

Sei $\mathbb{F}_2^h$ der $h$-dimensionale Vektorraum über dem Körper mit zwei Elementen. Für gegebenes $n$ und Faktorbasis $B$ mit $h$ Elementen, kann man jeder $B$-Zahl $z$ einen eindeutigen Vektor $\epsilon$, wie folgt zuordnen. Wir schreiben $z$ als $z = \prod_{j=1}^{h} p_j^{\alpha_j}$ und setzen dann die $j$-te Komponente $\epsilon_j := \alpha_j \bmod 2$. D.h. wir unterscheiden nur zwischen geradem und ungeradem Exponent.

**3.5 Beispiel**
Für $n = 35$, $B = \{-1, 2, 3, 5\}$ und $z = 23$ ist $\epsilon = (1, 0, 1, 0)$, weil $23 \bmod 35 = -12 = -1^1 \cdot 2^2 \cdot 3^1 \cdot 5^0$ ist.

Angenommen wir haben eine Menge $M = \{b_1, b_2, \ldots, b_k\}$, so daß $a_i := b_i^2 \bmod n$ eine $B$-Zahl ist, und die Summe der korrespondierenden $\epsilon_i = (\epsilon_{i,1}, \ldots, \epsilon_{i,h})$ in

$\mathbb{F}_2^h$ gleich Null ist. Dann ist das Produkt der $a_i$ ein Produkt aus Quadraten der $p_j$.

$$\prod_{i=1}^{k} a_i = \prod_{j=1}^{h} p_j^{\sum_{i=1}^{k} \alpha_{i,j}} = \prod_{j=1}^{h} p_j^{2\frac{1}{2}\sum_{i=1}^{k} \alpha_{i,j}} = \left( \prod_{j=1}^{h} p_j^{\frac{1}{2}\sum_{i=1}^{k} \alpha_{i,j}} \right)^2$$

Sei daher $c := \prod_{j=1}^{h} p_j^{\gamma_j} \bmod n$ mit $\gamma_j := \frac{1}{2}\sum_{i=1}^{k} \alpha_{i,j}$ und $b = \prod_{i=1}^{k} b_i \bmod n$. Dann erhalten wir zwei Zahlen, die auf verschiedene Weise konstruiert wurden und deren Quadrate kongruent modulo n sind.

Es kann natürlich passieren, daß $b \equiv \pm c \pmod{n}$ ist, dann muß man einfach mit einer anderen Menge $M$ nochmal anfangen. Um das zu vermeiden, sollte man zumindest darauf achten, daß für alle $i$ gilt $b_i > \sqrt{n/2}$, da sonst $a_i = b_i^2$ ist, was dazu führt daß der korrespondierende Vektor Null ist und somit auf eine Linearkombination keinen Einfluß hat. Außerdem wäre dann auch $b = c$.

**3.6 Beispiel**
Sei $n = 4633$. Wir wählen $M = \{67, 68\}$ und erhalten damit $a_1 = 67^2 \bmod 4633 = -144 = -1^1 \cdot 2^4 \cdot 3^2$ und $a_2 = 68^2 \bmod 4633 = -9 = -1^1 \cdot 3^2$. Wir wählen also $B = \{-1, 2, 3\}$, womit die korrespondierenden Vektoren $(1, 0, 0)$ und $(1, 0, 0)$ sind, und deren Summe ist über $\mathbb{F}_2^3$ gleich Null. Wir erhalten also $b = 67 \cdot 68 \bmod 4633 = -77$ und $c = 2^2 \cdot 3^2 = 36$ (man kann sich die Potenz von $-1$ sparen, da sowohl $+c$ als auch $-c$ eine Wurzel von $c^2 \equiv b^2 \pmod{n}$ ist). Da $-77 \not\equiv \pm 36 \pmod{4633}$ ist, erhalten wir mit $\mathrm{ggt}(-77 + 36, 4633) = 41$ einen Faktor von 4633. Also ist $4633 = 41 \cdot 113$.

Schauen wir uns nun einmal an, wie wahrscheinlich es ist, daß man $b$ und $c$ erhält für die $b \equiv \pm c \pmod{n}$ ist. Für ein zusammengesetztes $n$ aus $r$ Primzahlen gilt, daß jede Quadratzahl $2^r$ Wurzeln modulo $n$ hat. D.h. falls $n$ keine Primzahl ist, gibt es mindestens vier Wurzeln, also nur zwei von mindestens vier Möglichkeiten für einen Fehlschlag. Die Wahrscheinlichkeit für einen Erfolg liegt also mindestens bei 1/2. Verallgemeinert haben wir also, daß wir mit einer Wahrscheinlichkeit von höchstens $2^{-k}$ mehr als $k$ Versuche brauchen bis wir einen nichttrivialen Faktor von $n$ bestimmt haben.

Wie bekommt man aber in der Praxis die Menge $M = \{b_1, \ldots, b_k\}$ und die passende Faktorbasis? Eine Methode ist, daß man als Faktorbasis $B$ die ersten $h$ Primzahlen (oder $h - 1$ Primzahlen und $p_1 = -1$) wählt und sich dann zufällig Zahlen für $M$ aussucht deren Quadrate $B$-Zahlen sind. Eine andere Methode ist, daß man sich $M$ so wählt, daß $b_i^2 \bmod n$ für alle $i$ einen kleinen Betrag hat. Hier kann man z.B. Zahlen wählen, die nahe an $\sqrt{kn}$ liegen, wobei allerdings die Größe von $kn$ eine entscheidende Rolle spielt; Je größer $kn$ ist, desto näher müssen die Zahlen an $\sqrt{kn}$ liegen. Dann wählt man $B$ so, daß genügend der $a_i = b_i^2 \bmod n$ $B$-Zahlen sind. Dieses Verfahren wurde auch in Beipiel 3.6 benutzt.

Wie groß muß man aber nun $M$ wählen, um sicher zu sein, daß man eine Untermenge findet, deren korrespondierende Vektoren sich zu Null addieren, d.h.

linear abhängig sind. Die Antwort liefert uns die Lineare Algebra: Man braucht höchstens $h+1$ Vektoren in einem $h$-dimensionalen Vektorraum, hier $\mathbb{F}_2^h$, falls $|B| = h$. Man kann natürlich auch schon mit weniger als $h+1$ Vektoren erfolgreich sein, wie wir in Beipiel 3.6 gesehen haben. Wenn $h$ sehr groß ist, wird die Suche nach einer solchen Linearkombination von Vektoren natürlich etwas schwieriger. Aber dafür kann man auch wieder die Lineare Algebra oder die Numerik zu Rate ziehen und das Problem z.B. mit dem Gaußeliminationsverfahren lösen.

**3.7 Beispiel**
Wir wollen nun $n = 1829$ faktorisieren und verwenden dazu Werte für $b_i$, die nahe bei $\sqrt{kn}$ liegen und für die $b_i^2 \bmod n$ ein Produkt aus Primzahlen $< 20$ und evtl. $-1$ ist. Im folgenden tragen wir in eine Tabelle in jede Zeile jeweils in der ersten Spalte den Wert von $b_i$ ein, in der zweiten $a_i := b_i^2 \bmod n$ und in den folgenden Spalten die zugehörigen Exponenten der $p_j$, wobei $B = \{p_1, p_2, \ldots, p_h\}$ die Faktorbasis bestehend aus $-1$ und den Primzahlen $< 20$ ist. Wir wollen dies einmal für $k = 1, 2, 3, 4$ tun.

| $b_i$ | $a_i$ | $-1$ | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|
| 42 | $-65$ | 1 | - | - | 1 | - | - | 1 | - | - |
| 43 | 20 | - | 2 | - | 1 | - | - | - | - | - |
| 61 | 63 | - | - | 2 | - | 1 | - | - | - | - |
| 74 | $-11$ | 1 | - | - | - | - | 1 | - | - | - |
| 85 | $-91$ | 1 | - | - | - | 1 | - | 1 | - | - |
| 86 | 80 | - | 4 | - | 1 | - | - | - | - | - |

60 und 75 werden hier nicht ausgewählt, da $60^2 \bmod 1829 = -58 = -1 \cdot 2 \cdot 29$ und $75^2 \bmod 1829 = 138 = 2 \cdot 3 \cdot 23$. Unsere Aufgabe ist es nun, eine Untermenge dieser Zeilen zu finden, so daß die Summe nur noch gerade Zahlen bei den Exponenten stehen hat. Man findet recht leicht, daß die zweite und sechste Zeile diese Bedingung erfüllen. Es ergibt sich als Summe die Zeile - 6 - 2 - - - - -. Also bekommen wir $b = 43 \cdot 86 \bmod 1829 = 40$ und $c = 2^{6/2} \cdot 5^{2/2} \bmod 1829 = 40$, womit $b \equiv \pm c \pmod{n}$ ist. Da uns dies aber keine neuen Erkentnisse liefert müssen wir es noch einmal mit einer anderen Kombination probieren. Nehmen wir diesmal die ersten drei und die fünfte Zeile, dann erhalten wir als Summe die Zeile 2 2 2 2 2 - 2 - -. Dann ist also $b = 42 \cdot 43 \cdot 61 \cdot 85 \bmod 1829 = -370$ und $c = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 13 \bmod 1829 = 901$ (auch hier kann man die $-1$ wieder vernachlässigen). Somit ist ein Faktor von $n$ gleich $\mathrm{ggt}(-370 + 901, 1829) = 59$ und damit $n = 31 \cdot 59$.

Wir wollen nun noch einmal zusammenfassend, den Faktorbasis-Algorithmus beschreiben, um sehr große $n$ zu faktorisieren, wobei $M$ zufällig gewählt wird. Man wähle sich ein $y$ mittlerer Größe, z.B. wenn $n$ eine 50-stellige Dezimalzahl ist, nimmt man für $y$ eine 5- oder 6-stellige Dezimalzahl. Anschließend wählt man als Faktorbasis $B$ die Menge aller Primzahlen $\leq y$ und fügt noch die $-1$ hinzu.

Wähle zufällig Zahlen $b_i$ aus, so daß $b_i \bmod n$ $B$-Zahlen sind. Wenn man $\pi(y)+2$ solcher Zahlen gefunden hat, kann man aufhören und eine Linearkombination der korrespondierenden Vektoren suchen, die über $\mathbb{F}_2^h$, $h = \pi(y+1)$, Null ist, wozu man am einfachsten das Eliminationsverfahren von Gauß verwendet. Dann berechnet man $b$ und $c$ und überprüft, ob $b \not\equiv \pm c$ ist. Falls dies der Fall ist, sucht man neue Zahlen $b_i$ oder versucht, um Zeit zu sparen, nur eine neue Linearkombination zu finden, die Null ergibt. Wenn man dann $b$ und $c$ gefunden hat mit $b^2 \equiv c^2 \pmod{n}$ und $b \not\equiv \pm c$, ist $\mathrm{ggt}(b+c, n)$ ein nichttrivialer Teiler von $n$.

# 4 Laufzeit des Faktorbasis-Algorithmus

Wenn man einen Algorithmus entwirft, ist man natürlich auch immer an der Laufzeit interessiert. Im Folgenden werden wir durch Annäherungen und Abschätzungen eine ungefähre wahrscheinliche Laufzeit erhalten. Dazu werden wir die zwei folgenden Sätze benutzen:

**4.1 Satz**
$\ln(n!) = n\ln(n) - n + \mathrm{o}(n)$

**Beweis:**
Es ist $\ln(n!) = \sum_{i=1}^{n} \ln(i) = \int_1^n \ln(x)\,dx + \mathrm{O}(\ln(n)) = n\ln(n) - n + 1 + \mathrm{O}(\ln(n)) = n\ln(n) - n + \mathrm{o}(n)$. Der Summand $\mathrm{O}(\ln(n))$ ergibt sich aus dem Rechenfehler, den man bei der Annäherung der Summe durch das Integral macht. Zum einen entspricht das Integral nur der Summe bis $n-1$, wodurch also der Summand $\ln(n)$ in dem Integral fehlt. Zum anderen machen wir auf jedem Intervall $[a, a+1]$ einen Fehler von maximal $(\ln(a+1) - \ln(a)) \cdot 1$ (da $\ln(x)$ monoton wachsend ist), was sich über das ganze Intervall $[1, n]$ zu $\ln(n)$ aufsummiert. □

**4.2 Satz**
*Sei $n \in \mathbb{N}$ und $u \in \mathbb{R}^+$, dann existieren genau $\binom{\lfloor u \rfloor + N}{N}$ $N$-Tupel $(\alpha_1, \alpha_2, \ldots, \alpha_N)$ mit $\sum_{j=1}^{N} \alpha_j \leq u$.*

**Beweis:**
Sei $\beta_0 = 0$ und $\beta_{j+1} = \beta_j + 1 + \alpha_{j+1}$, d.h. zwischen $\beta_j$ und $\beta_{j+1}$ liegen genau $\alpha_{j+1}$ ganze Zahlen und $\beta_N \leq \lfloor u \rfloor + N$, da $\beta_N = 0 + 1 + \alpha_1 + 1 + \alpha_2 + \cdots + 1 + \alpha_N = N + \sum_{j=1}^{N} \alpha_j$. Zwischen den $\alpha_j$ und $\beta_j$ existiert also eine eindeutige Beziehung. Und für die $\beta_j$ haben wir genau $\binom{\lfloor u \rfloor + N}{N}$ Möglichkeiten auszuwählen, da wir $N$ Zahlen aus $\lfloor u \rfloor + N$ auswählen. □

Um nun die Laufzeit abzuschätzen, müssen wir zuerst einmal wissen, wie wahrscheinlich es ist, daß eine zufällige Zahl, die kleiner als $x$ ist, ein Produkt aus Primzahlen ist, die kleiner als $y$ sind, wobei hier $y$ viel kleiner als $x$ sein soll. Dazu sei $u := \ln(x)/\ln(y)$. D.h. wenn $x$ eine $r$-Bit Zahl ist und $y$ eine $s$-Bit Zahl

ist, dann ist $u \approx r/s$. Für spätere Abschätzungen ist es sinnvoll anzunehmen, daß $u$ sehr viel kleiner als $y$ ist. Da $\pi(y) \approx y/\ln(y)$ ist, können wir auch annehmen, daß $u$ viel kleiner als $\pi(y)$ ist. Typische Größen für $x$, $y$ und $u$ sind:

$x \approx 10^{48}$;
$y \approx 10^6$ (dann ist $\pi(y) \approx 7 \cdot 10^4$ und $\ln(y) \approx 14$);
$u \approx 8$.

Es ist üblich mit $\Psi(x, y)$ die Anzahl der Zahlen $\leq x$ zu bezeichnen, die nicht durch Primzahlen $> y$ teilbar sind. Also die Anzahl der Zahlen, die man als Produkt $\prod p_j^{\alpha_j} \leq x$ darstellen kann, wobei die $p_j$ alle Primzahlen $\leq y$ sind und $\alpha_j \in \mathbb{N}_0$. Es besteht offensichtlich eine eindeutige Beziehung zwischen $\pi(y)$-Tupeln aus nichtnegativen ganzen Zahlen $\alpha_j$ für die $\prod p_j^{\alpha_j} \leq x$ und den natürlichen Zahlen $\leq x$, die nicht durch eine Primzahl $> y$ teilbar sind. Also gibt es $\Psi(x, y)$ Lösungen zu der Ungleichung

$$\sum_{j=1}^{\pi(y)} \alpha_j \ln(p_j) \leq \ln(x)$$

mit Unbekannten $\alpha_j$. Die meisten Primzahlen, die kleiner als $y$ sind, haben etwa genauso viele Binärstellen wie $y$. Daher gilt $\ln(p_j) \approx \ln(y)$ für die meisten $p_j$. Wir erlauben uns deshalb in der Ungleichung $\ln(p_j)$ durch $\ln(y)$ und $\ln(x)/\ln(y)$ durch $u$ zu ersetzen. Dann hat die Ungleichung

$$\sum_{j=1}^{\pi(y)} \alpha_j \leq u$$

ungefähr $\Psi(x, y)$ Lösungen.

Wir machen nun eine weitere wichtige Vereinfachung, indem wir $\pi(y)$ durch $y$ ersetzen. Das ist natürlich eine sehr grobe Abschätzung, die aber zum selben Ergebnis der Laufzeitabschätzung führt, wie eine genauere Abschätzung von $\Psi(x, y)$. Deswegen nehmen wir nun an, daß $\Psi(x, y)$ in etwa genauso groß ist wie die Anzahl der Lösungen von

$$\sum_{j=1}^{y} \alpha_j \leq u.$$

Mit Satz 4.2 und $N = y$ erhalten wir, daß $\Psi(x, y)$ ungefähr $\binom{\lfloor u \rfloor + y}{y}$ ist. Wir berechnen im folgenden den Logarithmus der Wahrscheinlichkeit, daß eine zufällige natürliche Zahl ein Produkt aus Primzahlen $\leq y$ ist, also $\ln\left(\frac{\Psi(x,y)}{x}\right)$. Man beachte, daß per Definition $\ln(x) = u \ln(y)$ ist. Wenn wir nun noch Satz 4.1 benutzen,

erhalten wir

$$\begin{aligned}
\ln\left(\frac{\Psi(x,y)}{x}\right) &\approx \ln\left(\frac{(\lfloor u\rfloor+y)!}{\lfloor u\rfloor! y!}\right) - u\ln(y)\\
&\approx (\lfloor u\rfloor+y)\ln(\lfloor u\rfloor+y) - (\lfloor u\rfloor+y) - (\lfloor u\rfloor\ln(\lfloor u\rfloor) - \lfloor u\rfloor)\\
&\quad - (y\ln(y)-y) - u\ln(y)\\
&\overset{\lfloor u\rfloor\approx u\ll y}{\approx} (u+y)\ln(y) - (u+y) - (u\ln(u)-u)\\
&\quad - (y\ln(y)-y) - u\ln(y)\\
&\approx -u\ln(u),
\end{aligned}$$

also

$$\frac{\Psi(x,y)}{x} \approx u^{-u}.$$

Z.B. für $x \approx 10^{48}$ und $y \approx 10^6$ heißt das, wenn wir zufällig eine natürliche Zahl $\leq x$ auswählen, ist die Chance, daß diese Zahl ein Produkt nur aus Primzahlen $\leq y$ ist, 1 zu $8^8 = 16777216 \approx 10^7$.

Wir sind nun in der Lage, die Laufzeit für den oben beschriebenen Faktorbasis Algorithmus zu bestimmen. Zur Vereinfachung nehmen wir an, daß $B$ aus den ersten $h = \pi(y)$ Primzahlen besteht und die $-1$ nicht darin enthalten ist. Außerdem betrachten wir mit $b_i^2 \bmod n$ nur den kleinsten positiven Rest anstatt wie bisher den kleinsten absoluten Rest.

Wir müssen also für folgende Schritte die Anzahl der Bit-Operationen berechnen:

1. Wähle $\pi(y)+1$ zufällige (verschiedene) $b_i$ zwischen 1 und $n$, so daß der kleinste positive Rest $b_i^2 \bmod n$ eine $B$-Zahl ist.

2. Finde eine Linearkombination der korrespondierenden Vektoren, die über $\mathbb{F}_2^h$ Null ergibt, so daß man eine Kongruenz der Form $b^2 \equiv c^2 \pmod n$ erhält.

3. Wenn $b \equiv \pm c \pmod n$ ist, dann führe die Schritte (1) und (2) solange aus bis $b \not\equiv \pm c \pmod n$ ist, aber trotzdem $b^2 \equiv c^2 \pmod n$ ist. Berechne mit $\mathrm{ggt}(b+c, n)$ einen nichttrivialen Faktor von $n$.

Angenommen daß die $b_i^2 \bmod n$ zufällig zwischen 1 und $n$ verteilt sind, erwarten wir, daß wir etwa $u^u$ Versuche brauchen, bis wir ein $b_i$ finden, so daß $b_i^2 \bmod n$ ein Produkt aus Primzahlen $\leq y$ ist und somit eine $B$-Zahl ist. Hierbei ist $u = \ln(n)/\ln(y)$. Wir werden später bestimmen wie groß $y$ sein muß, um möglichst schnell ein Ergebnis zu bekommen. Für große $y$ wird $u$ zwar ziemlich klein, aber dafür muß man bei der Faktorisierung der $b_i^2 \bmod n$ sehr viele Primzahlen beachten und bei der Suche nach der Linearkombination bekommen wir entsprechend auch mehr zu tun. Für kleine $y$ ist zwar die Faktorisierung der

$b_i^2 \bmod n$ und die Suche nach der Linearkombination sehr leicht, dafür findet man aber nur sehr selten $b_i$, die sich als Produkt aus Primzahlen $\leq y$ darstellen lassen. Man muß also $y$ aus einem mittleren Bereich wählen.

Um zu entscheiden, wie $y$ gewählt werden muß, werden wir zunächst eine grobe Laufzeitabschätzung in Abhängigkeit von $y$ (und $n$) machen und diesen Ausdruck dann in $y$ optimieren.

Angenommen $n$ ist eine $r$-Bit-Zahl und $y$ eine $s$-Bit-Zahl, dann ist $u$ ziemlich genau $r/s$. Zunächst einmal wollen wir uns fragen wieviele Schritte man braucht um ein $b_i$ als Produkt aus Primzahlen $\leq y$ darzustellen? Wir behaupten, daß die Anzahl der Schritte polynomial in $r$ und $y$ ist, also $\mathrm{O}(r^l e^{ks})$ für sehr kleine natürliche Zahlen $k$ und $l$. Man braucht für ein zufälliges Bit eine feste Zeit, also braucht man für die Auswahl eines zufälligen $b_i$ zwischen 1 und $n$ $\mathrm{O}(r)$ Schritte. Um $b_i^2 \bmod n$ auszurechnen braucht man $\mathrm{O}(r^2)$ Schritte. Wir müssen dann $b_i^2 \bmod n$ nacheinander durch alle Primzahlen $\leq y$ und ihre Potenzen teilen, die beim Teilen keinen Rest lassen, und schauen, ob am Ende 1 übrig bleibt. Ein einfaches Verfahren dazu ist, mit 2 anzufangen und dann durch alle ungeraden Zahlen $p \leq y$ zu teilen. Wenn $p$ keine Primzahl ist, wird beim Teilen ein Rest bleiben, da wir schon durch alle Faktoren von $p$ geteilt haben. Eine Division einer $r$-Bit-Zahl durch eine $s$-Bit-Zahl braucht $\mathrm{O}(rs)$ Operationen und damit braucht der ganze Test $\mathrm{O}(r^2+rsy) = \mathrm{O}(rsy)$ Operationen ($r \approx su < sy$) für einen zufällig gewählten Wert von $b_i$.

Um Schritt (1) zu vervollständigen, muß man etwa $u^u(\pi(y)+1)$ Werte testen. Da $\pi(y) \approx y/\ln(y) = \mathrm{O}(y/s)$ ist, brauchen wir für Schritt (1) $\mathrm{O}(u^u r y^2)$ Bit-Operationen.

Schritt (2) ist polynomial in $y$ und $r$ (Zeilenreduktion und Berechnung von $b$ und $c$ modulo $n$). Also braucht Schritt (2) $\mathrm{O}(y^j r^h)$ Bit-Operationen mit kleinen natürlichen Zahlen $j$ und $h$.

Jeder Durchlauf der ersten beiden Schritte hat eine Chance von mindestens 50% auf Erfolg. Wenn wir also mit einer Erfolgswahrscheinlichkeit von $1 - 2^{-50}$ zufrieden sind, reicht es 50 mal durch die ersten beiden Schritte zu laufen.

Zusammenfassend haben wir also eine Laufzeit von

$$\mathrm{O}(50(u^u r^2 y^2 + y^j r^h)) = \mathrm{O}(r^h u^u y^j) = \mathrm{O}(r^h u^u e^{ks}) = \mathrm{O}(r^h (r/s)^{r/s} e^{ks}),$$

für geeignete natürliche Zahlen $h$ und $k$.

Wir können diesen Ausdruck in $y$ bzw. $s$ für festes $n$ bzw. $r$ minimieren oder auch einfacher den Logarithmus, also $\frac{r}{s}\ln\left(\frac{r}{s}\right) + ks$. Wir setzen also

$$0 = \frac{d}{ds}\left(\frac{r}{s}\ln\left(\frac{r}{s}\right) + ks\right) = -\frac{r}{s^2}\left(\ln\left(\frac{r}{s}\right) + 1\right) + k \approx -\frac{r}{s^2}\ln\left(\frac{r}{s}\right) + k,$$

d.h. wir müssen $s$ so wählen, daß $ks$ ungefähr so groß ist wie $\frac{r}{s}\ln\left(\frac{r}{s}\right)$, mit anderen Worten so, daß die beiden Faktoren in $(r/s)^{r/s}e^{ks}$ etwa gleich groß sind. Aus der obigen ungefähren Gleichung ergibt sich daß $s^2$ und $r\ln(r/s)$ die gleiche

Größenordnung haben, da $k$ konstant ist. Das bedeutet, daß $s$ zwischen $\sqrt{r}$ und $\sqrt{r\ln(r)}$ liegt, woraus wiederum folgt, daß $\ln(s) \approx \frac{1}{2}\ln(r)$ ist. Damit folgt aus obiger Relation, daß $0 \approx -\left(\frac{r}{s^2}\ln(r) - \frac{r}{s^2}\ln(s)\right) + k \approx -\frac{r}{2s^2}\ln(r) + k$, d.h.

$$s \approx \sqrt{\frac{r}{2k}\ln(r)}$$

Mit diesem Wert für $s$ sind die beiden Faktoren $(r/s)^{r/s}$ und $e^{ks}$ ungefähr gleich groß, und es ergibt sich für die Gesamtlaufzeit $\mathrm{O}(e^{2ks}) = \mathrm{O}\left(e^{\sqrt{2k}\sqrt{r\ln(r)}}\right)$. Wenn man die Konstante $\sqrt{2k}$ durch $C$ ersetzt, erhält man für die Anzahl der benötigten Bit-Operationen, um eine $r$-Bit-Zahl zu faktorisieren:

$$\mathrm{O}\left(e^{C\sqrt{r\ln(r)}}\right).$$

Wir haben hier sehr grob abgeschätzt und haben auch keine genaue Vorstellung von dem Fehler, den wir gemacht haben. Außerdem ist sowohl das Verfahren, als auch die Laufzeitabschätzung probabilistischer Natur.

Bis zur Entwicklung des Zahlkörpersiebs, hatten die schnellsten Faktorisierungsalgorithmen eine Laufzeit von $\mathrm{O}\left(e^{C\sqrt{r\ln(r)}}\right)$. Der Hauptunterschied zwischen den verschiedenen Verfahren war die Konstante $C$.

**4.3 Bemerkung**
Da $r = \mathrm{O}(\ln(n))$ kann man die Laufzeitabschätzung auch in folgender Form schreiben:

$$\mathrm{O}\left(e^{C\sqrt{\ln(n)\ln(\ln(n))}}\right)$$

Mit Ausnahme des Zahlkörpersiebs haben alle schnellen Faktorisierungsalgorithmen eine Laufzeit dieser Größenordnung mit $C = 1 + \epsilon$ für beliebig kleines $\epsilon$.

# 5 Das quadratische Sieb

Das quadratische Sieb wurde anfang der 80er von Pomerance entwickelt und war lange Zeit das schnellste Verfahren, um große Zahlen $n$ zu faktorisieren deren Primfaktoren etwa in der Größenordnung von $\sqrt{n}$ liegen. Für spezielle Fälle gibt es natürlich auch noch bessere Verfahren, z.B. für Primfaktoren die viel kleiner sind als $\sqrt{n}$ ist das ellipitsche Kurven Verfahren besser.

Bevor wir uns dem quadratischen Sieb zuwenden, müssen wir uns noch damit beschäftigen, wie man quadratische Äquivalenzen löst. Schauen wir uns zunächst den einfachen Fall an, bei dem man eine Äquivalenz der Form $x^2 \equiv n \pmod{p}$ mit $\left(\frac{n}{p}\right) = 1$ lösen will. Dazu zunächst der Algorithmus:

1. Suche $m$ mit $\left(\frac{m}{p}\right) = -1$.
2. Schreibe $p-1$ als $2^\alpha s$, mit ungeradem $s$.
3. Setze $b := m^s \bmod p$.
4. Setze $r := n^{(s+1)/2} \bmod p$.
5. Wenn $\left(\frac{\left(rb^j\right)^2}{n}\right)^{2^{\alpha-2-k}} = \left\{\begin{smallmatrix}1\\-1\end{smallmatrix}\right\}$, mit $j = \sum_{i=0}^{k-1} j_i 2^i$, setze $j_k := \left\{\begin{smallmatrix}0\\1\end{smallmatrix}\right\}$, für $k = 0, 1, 2, \ldots, \alpha-2$.
6. $x := rb^j$ ist eine Lösung.

Und nun folgt die Erklärung, warum das ganze funktioniert: Im Schritt 3 generiert man zunächst mal eine primitive $2^\alpha$-te Einheitswurzel, da $b^{2^\alpha} = m^{2^\alpha s} = m^{p-1} = 1$. Wäre $b$ nicht primitiv, dann wäre $b$ eine gerade Potenz einer primitiven $2^\alpha$-ten Einheitswurzel und somit ein Quadrat, was aber wegen $\left(\frac{b}{p}\right) = \left(\frac{m}{p}\right)^s = -1$ (Schritt 1) nicht sein kann. Als nächstes stellen wir fest, daß

$$(n^{-1}r^2)^{2^{\alpha-1}} = n^{s2^{\alpha-1}} = n^{\frac{p-1}{2}} = \left(\frac{n}{p}\right) = 1$$

ist. Wir müssen also nur noch $r$ so mit einer geeigneten Potenz einer $2^\alpha$-ten Einheitswurzel (d.h. $b$) modifizieren, daß $x = rb^j$ eine Quadratwurzel von $n$ wird. Dies geschieht in Schritt 5, in dem schrittweise das entsprechende $j$ gesucht wird.

Als nächstes suchen wir eine Lösung der Äquivalenz $x^2 \equiv n \pmod{p^\beta}$ mit $\left(\frac{n}{p}\right) = 1$. Auch dazu erst der Algorithmus:

1. Löse $\tilde{x}^2 \equiv n \pmod{p^{\beta-1}}$
2. Schreibe $\tilde{x}$ als $\sum_{i=0}^{\beta-2} x_i p^i$ mit $0 \le x_i < p$.
3. Schreibe $\tilde{x}^2$ als $n + bp^{\beta-1}$
4. Setze $x_{\beta-1} := -(2x_0)^{-1} b \bmod p$.
5. $x := \sum_{i=0}^{\beta-1} x_i p^i$ ist eine Lösung.

Das ist eine Lösung, da

$$\begin{aligned} x^2 &\equiv \left(\tilde{x} - (2x_0)^{-1} b p^{\beta-1}\right)^2 \\ &\equiv \tilde{x}^2 - 2\tilde{x}(2x_0)^{-1} b p^{\beta-1} + \left((2x_0)^{-1} b p^{\beta-1}\right)^2 \\ &\equiv n + b p^{\beta-1} - x_0 x_0^{-1} b p^{\beta-1} \\ &\equiv n \pmod{p^\beta} \end{aligned}$$

Kommen wir damit zum eigentlichen Inhalt dieses Kapitels.

Das quadratische Sieb ist eine Variation des Faktorbasis-Algorithmus. Als Faktorbasis $B$ nehmen wir die Menge aller Primzahlen $p \leq P$ für die $n$ ein quadratischer Rest modulo p ist, d.h. $\left(\frac{n}{p}\right) = 1$ ist (es ex. ein $a$, so daß $a^2 \equiv n \pmod{p}$). Hierbei muß $P$ in irgendeiner noch zu klärenden Weise optimal gewählt werden. Außerdem ist die 2 immer in $B$ enthalten. Die Menge $S$, in der wir nach $B$-Zahlen suchen, ist die gleiche wie bei der Fermat-Faktorisierung, also

$$S = \left\{ t^2 - n \;\middle|\; \lfloor \sqrt{n} \rfloor + 1 \leq t \leq \lfloor \sqrt{n} \rfloor + A \right\}$$

für ein passend gewähltes $A$.

Die eigentliche Idee bei dem Verfahren ist, daß man nicht jedes einzelne $s \in S$ versucht durch jedes $p \in B$ zu teilen, um zu sehen ob es eine $B$-Zahl ist, sondern daß man für jedes $p \in B$ die Teilbarkeit für alle $s \in S$ gleichzeitig testet. Daher kommt auch der Begriff „Sieb", ähnlich wie beim Sieb des Eratosthenes, mit dem man sich z.B. auch die Primzahlen $\leq P$ erzeugen kann.

Wir werden nun zuerst einmal den Algorithmus betrachten und uns anschließend ein Beispiel ansehen. Der hier verwendete Algorithmus ist nur eine mögliche Variante und nicht unbedingt die schnellste. Für eine zusammengesetzte ungerade Zahl $n$ geht man folgendermaßen vor:

1. Wähle $P$ und $A$ in der Größenordnung von $e^{\sqrt{\ln(n)\ln(\ln(n))}}$. $A$ sollte größer sein als $P$, aber nicht größer als eine sehr kleine Potenz von $P$, z.B. $P < A < P^2$.

2. Für $t = \lfloor \sqrt{n} \rfloor + 1, \lfloor \sqrt{n} \rfloor + 2, \ldots, \lfloor \sqrt{n} \rfloor + A$ trage $t^2 - n$ in eine Spalte ein.

3. Für jede ungerade Primzahl $p \leq P$ überprüfe, ob $\left(\frac{n}{p}\right) = 1$. Falls nicht, streiche $p$ aus der Faktorbasis, ansonsten fahre mit dem nächsten Schritt fort.

4. Sei also $p \in B$ eine ungerade Primzahl (den Fall $p = 2$ behandeln wir später). Man wähle $\beta$ maximal, so daß die Kongruenz $t^2 \equiv n \pmod{p^\beta}$ noch eine Lösung $t$ hat, mit $t \equiv s \pmod{p^\beta}$ für ein $s \in [\lfloor \sqrt{n} \rfloor + 1, \lfloor \sqrt{n} \rfloor + A]$. Seien im folgenden $t_1$ und $t_2$ Lösungen mit $t_2 \equiv -t_1 \pmod{p^\beta}$ ($t_1$ und $t_2$ müssen nicht im Intervall von $\lfloor \sqrt{n} \rfloor + 1$ bis $\lfloor \sqrt{n} \rfloor + A$ liegen).

5. Für den selben Wert von $p$, gehe die Liste aus Schritt 2 ($t^2 - n$) durch und schreibe eine 1 unter $p$ in jede Zeile, in der $t$ von $t_1$ um ein Vielfaches von $p$ abweicht. Ersetze in jeder Zeile unter $p$ die 1 durch eine 2, in der $t$ und $t_1$ um ein Vielfaches von $p^2$ abweichen, usw. bis $p^\beta$. Mit $t_2$ geht man anschließend genauso vor.

6. Jedesmal wenn man im vorherigen Schritt eine 1 einträgt oder einen Eintrag erhöht, teile man den Eintrag in der Spalte $t^2 - n$ durch den entsprechenden Wert von $p$, und schreibe das Ergebnis wieder in die gleiche Spalte.

7. Für $p = 2$ ermittelt man die maximale Potenz von $p$, die $t^2 - n$ teilt, und teilt dann den Eintrag unter $t^2 - n$ durch diese Potenz von $p$ (was auf Computern sehr schnell zu implementieren ist). Anschließend trägt man noch den entsprechenden Exponent in der Spalte für $p = 2$ ein.

8. Wenn man alle Primzahlen $\leq P$ behandelt hat, streiche man alle Zeilen, in denen der Eintrag der Spalte $t^2 - n$ ungleich 1 ist. Man hat nun für jede Zahl $t$ zwischen $\lfloor \sqrt{n} \rfloor + 1$ und $\lfloor \sqrt{n} \rfloor + A$, für die $t^2 - n$ eine $B$-Zahl ist die Exponenten der Primzahlzerlegung von $t^2 - n$.

9. Nun kann man fortfahren wie bei dem Faktorbasis-Algorithmus.

**5.1 Bemerkung**
Warum funktionieren eigentlich die Schritte 4 - 6? Angenommen man hat also in Schritt 4 ein maximales $\beta$ und die Lösungen $t_1$ und $t_2 \equiv -t_1 \pmod{p^\beta}$ gefunden. Es gelte $p^\alpha \mid t^2 - n$, mit $\alpha \leq \beta$. Dann gilt auch $t^2 \equiv n \pmod{p^\alpha}$ und $t_1^2 \equiv t_2^2 \equiv n \pmod{p^\alpha}$ und somit ist $t^2 \equiv t_{1,2}^2 \pmod{p^\alpha}$. Für $p > 2$ und festes $t$ hat die Kongruenz $t^2 \equiv s^2 \pmod{p^\alpha}$ genau die 2 Lösungen $s_{1,2} \equiv \pm t \equiv t_{1,2} \pmod{p^\alpha}$. Man findet also alle Primfaktoren. Man findet mit diesem Verfahren aber auch nicht mehr, da die obigen Aussagen alle äquivalent sind.

**5.2 Beispiel**
Wir wollen nun einmal $n = 1042387$ faktorisieren. Wir wählen dazu die Grenzen $P = 50$ und $A = 500$. Dann ist $\lceil \sqrt{n} \rceil = 1020$. Die möglichen Werte für $t$ sind somit $\{1021, \ldots, 1520\}$. Unsere Faktorbasis besteht aus den 8 Primzahlen $\{2, 3, 11, 17, 19, 23, 43, 47\}$, da für diese $n$ ein quadratischer Rest ist.

Fangen wir mit der Spalte für $p = 3$ an: Wir suchen also zuerst eine Lösung $t_1 = t_{1,0} + t_{1,1} \cdot 3 + t_{1,2} \cdot 3^2 + \cdots t_{1,\beta-1} \cdot 3^{\beta-1}$ für die Äquivalenz $t_1^2 = 1042387 \pmod{3^\beta}$, wobei $t_{1,j} \in \{0, 1, 2\}$ (die andere Lösung $t_2$ ist dann $t_2 = 3^\beta - t_1$). Für $t_{1,0}$ finden wir durch einfaches Ausprobieren den Wert 1 (für unsere 8 Primzahlen kann man die Lösung für $t_{1,0}$ noch durch einfaches Ausprobieren herausfinden, wenn man später größere Primzahlen behandelt, kann man obiges Verfahren benutzen). Als nächstes rechnen wir modulo 9 und suchen eine Lösung $t_{1,1}$ für $(1 + 3t_{1,1})^2 \equiv 1042387 \equiv 7 \pmod{9}$, also $6t_{1,1} \equiv 6 \pmod{9}$ und somit $2t_{1,1} \equiv 2 \pmod{3}$, woraus man $t_{1,1} = 1$ erhält. Dies kann man so weiter führen bis zu $3^7$, dann

findet man die Lösungen $t_1 \equiv (210211)_3 \pmod{3^7}$ und $t_2 \equiv (2012012)_3 \pmod{3^7}$, allerdings gibt es dazu kein $t$ zwischen 1021 und 1520 mit $t \equiv t_1 \pmod{3^7}$ oder $t \equiv t_2 \pmod{3^7}$. Somit erhalten wir $\beta = 6$, $t_1 = (210211)_3 = 589 \equiv 1318 \pmod{3^6}$ und $t_2 = 3^6 - t_1 = 140 \equiv 1112 \pmod{3^5}$ (wobei zu bemerken ist, daß es keine Zahl zwischen 1021 und 1520 gibt, die $\equiv t_2 \pmod{3^6}$ ist).

Wir können nun unseren Sieb für $p = 3$ starten. Wir beginnen bei $t = 1318$ und gehen die Liste in 3-er-Schritten nach unten und oben ab. Dabei tragen wir jeweils eine 1 in die entsprechende Zeile ein und teilen den Eintrag unter $t^2 - n$ durch 3. Als nächstes durchlaufen wir die Liste wieder beginnend bei 1318, jetzt aber mit Schrittweite 9, ändern den Eintrag unter $p = 3$ von einer 1 zu einer 2 und teilen den Eintrag unter $t^2 - n$ wieder durch 3. Damit fährt man fort bis zu einer Schrittweite von $3^\beta = 729$, wobei dann nur noch der Eintrag bei 1318 betroffen ist. Jetzt muß man die Liste beginnend mit $t_2 = 1112$ anstatt mit $t_2 = 1318$ durchlaufen und auch hierbei wieder die Schrittweite beginnend mit 3 jedesmal mit 3 multiplizieren. Nachdem man diesen Vorgang für die restlichen 6 ungeraden Primzahlen wiederholt hat, muß man noch die Zweierpotenzen eliminieren, was auf einem Computer sehr schnell und leicht erledigt werden kann.

Nachdem man all diese Schritte durchgeführt hat erhält man folgende Tabelle:

| $t$ | $t^2 - n$ | 2 | 3 | 11 | 17 | 19 | 23 | 43 | 47 |
|---|---|---|---|---|---|---|---|---|---|
| 1021 | 54 | 1 | 3 | - | - | - | - | - | - |
| 1027 | 12342 | 1 | 1 | 2 | 1 | - | - | - | - |
| 1030 | 18513 | - | 2 | 2 | 1 | - | - | - | - |
| 1061 | 83334 | 1 | 1 | - | 1 | 1 | - | 1 | - |
| 1112 | 194157 | - | 5 | - | 1 | - | - | - | 1 |
| 1129 | 232245 | 1 | 3 | 1 | 1 | - | 1 | - | - |
| 1148 | 275517 | - | 2 | 3 | - | - | 1 | - | - |
| 1175 | 338238 | 1 | 2 | - | - | 1 | 1 | 1 | - |
| 1217 | 438702 | 1 | 1 | 1 | 2 | - | 1 | - | - |
| 1390 | 889713 | - | 2 | 2 | - | 1 | - | 1 | - |
| 1520 | 1268013 | - | 1 | - | 1 | - | 2 | - | 1 |

Nun muß man wie beim Faktorbasis-Algorithmus linear abhängige Zeilen modulo 2 finden. Mit etwas Glück (oder systematischer Suche) findet man, daß die Zeilen für $t = 1112$ und $t = 1520$ linear abhängig sind. Man erhält damit $(1112 \cdot 1520)^2 \equiv (3^3 \cdot 17 \cdot 23 \cdot 47)^2 \pmod{1042387}$, also $647853^2 \equiv 496179^2 \pmod{1042387}$ und erhalten somit einen nichttrivialen Faktor $\mathrm{ggt}(647853 - 496179, 1042387) = 1487$.

**5.3 Bemerkung**

Auch das quadratische Sieb hat eine Laufzeit von

$$\mathrm{O}\left(e^{(1+\epsilon)\sqrt{\log(n)\log(\log(n))}}\right)$$

mit $\epsilon > 0$ und hat einen Platzbedarf von der Größenordnung

$$e^{C\sqrt{\log(n)\log(\log(n))}}.$$

# 6 Das Zahlkörpersieb

Zum Schluß wollen wir noch einen kurzen Blick auf das Zahlkörpersieb werfen. Bis zur Entwicklung dieses neuen Algorithmus, war die beste Laufzeit, die man erreichte

$$\exp\left(\mathrm{O}\left(\sqrt{\ln(n)\ln(\ln(n))}\right)\right).$$

Man dachte sogar, daß diese Schranke vielleicht eine natürliche untere Schranke für die Laufzeit der Faktorisierung von großen Zahlen sei. Das Zahlkörpersieb hat jedoch eine Laufzeit von

$$\exp\left(\mathrm{O}\left(\sqrt{\ln(n)^{1/3}\ln(\ln(n))^{2/3}}\right)\right).$$

Auch bei dem Zahlkörpersieb wird nach einer Lösung der Kongruenz $x^2 \equiv y^2 \pmod{n}$ gesucht. Dies geschieht allerdings mittels algebraischer Zahlentheorie, insbesondere mit normierten irreduziblen Polynomen.