Faktorisierung

9

Wie wir gezeigt haben, hängt die Sicherheit des RSA-Verfahrens und die Sicherheit des Rabin-Verfahrens eng mit der Schwierigkeit zusammen, natürliche Zahlen in ihre Primfaktoren zu zerlegen. Es ist nicht bekannt, ob das Faktorisierungsproblem für natürliche Zahlen leicht oder schwer ist. In den letzten Jahrzehnten wurden immer effizientere Faktorisierungsmethoden entwickelt. Trotzdem ist RSA heute immer noch sicher, wenn man die Parameter richtig wählt. Es könnte aber sein, dass schon bald ein so effizienter Faktorisierungsalgorithmus gefunden wird und RSA nicht mehr sicher ist. Daher ist es wichtig, kryptographische Systeme so zu implementieren, dass die grundlegenden Verfahren leicht ersetzt werden können.

In diesem Kapitel beschreiben wir einige Faktorisierungsalgorithmen. Dabei ist n immer eine natürliche Zahl, von der schon bekannt ist, dass sie zusammengesetzt ist. Das kann man z. B. mit dem Fermat-Test oder mit dem Miller-Rabin-Test feststellen (siehe Abschn. 7.2 und Abschn. 7.4). Diese Tests bestimmen aber keinen Teiler von n. Wir skizzieren die Faktorisierungsverfahren nur. Für weitere Details sei auf [42] und [18] verwiesen. Die beschriebenen Algorithmen sind in der Bibliothek *LiDIA*[47] implementiert.

9.1 Probedivison

Um die kleinen Primfaktoren von n zu finden, berechnet man die Liste aller Primzahlen unter einer festen Schranke B. Dafür kann man das Sieb des Erathostenes verwenden (siehe [4]). Dann bestimmt man für jede Primzahl p in dieser Liste den maximalen Exponenten $e(p)$, für den p die Zahl n teilt. Eine typische Schranke ist $B = 10^6$.

Beispiel 9.1 Wir wollen die Zahl $n = 3^{21} + 1 = 10460353204$ faktorisieren. Probedivision aller Primzahlen bis 50 ergibt die Faktoren 2^2, 7^2 und 43. Dividiert man die Zahl n durch diese Faktoren, so erhält man $m = 1241143$. Es ist $2^{m-1} \equiv 793958 \bmod m$. Nach dem kleinen Satz von Fermat ist m also zusammengesetzt.

J. Buchmann, *Einführung in die Kryptographie*, Springer-Lehrbuch,
DOI 10.1007/978-3-642-39775-2_9

9.2 Die $p-1$-Methode

Es gibt Faktorisierungsmethoden, die Zahlen mit bestimmten Eigenschaften besonders gut zerlegen können. Solche Zahlen müssen als RSA- oder Rabin-Moduln vermieden werden. Als Beispiel für einen solchen Faktorisierungsalgorithmus beschreiben wir die $(p-1)$-Methode von John Pollard.

Das $(p-1)$-Verfahren ist für zusammengesetzte Zahlen n geeignet, die einen Primfaktor p haben, für den $p-1$ nur kleine Primfaktoren hat. Man kann dann nämlich ohne p zu kennen ein Vielfaches k von $p-1$ bestimmen. Wie das geht, beschreiben wir unten. Für dieses Vielfache k gilt nach dem kleinen Satz von Fermat

$$a^k \equiv 1 \mod p$$

für alle ganzen Zahlen a, die nicht durch p teilbar sind. Das bedeutet, dass p ein Teiler von a^k-1 ist. Ist a^k-1 nicht durch n teilbar, so ist $\gcd(a^k-1,n)$ ein echter Teiler von n. Damit ist n faktorisiert.

Der Algorithmus von Pollard verwendet als Kandidaten für k die Produkte aller Primzahlpotenzen, die nicht größer als eine Schranke B sind, also

$$k = \prod_{q\in\mathbb{P}, q^e\le B} q^e.$$

Wenn die Primzahlpotenzen, die $p-1$ teilen, alle kleiner als B sind, dann ist k ein Vielfaches von $p-1$. Der Algorithmus berechnet $g=\gcd(a^k-1,n)$ für eine geeignete Basis a. Wird dabei kein Teiler von n gefunden, so wird ein neues B verwendet.

Beispiel 9.2 Die Zahl $n = 1241143$ ist in Beispiel 9.1 übriggeblieben. Sie muss noch faktorisiert werden. Wir setzen $B = 13$. Dann ist $k = 8*9*5*7*11*13$ und

$$\gcd(2^k-1,n) = 547.$$

Also ist $p = 547$ ein Teiler von n. Der Kofaktor ist $q = 2269$. Sowohl 547 als auch 2269 sind Primzahlen.

Eine Weiterentwicklung der $(p-1)$-Methode ist die Faktorisierungsmethode mit elliptischen Kurven (ECM). Sie funktioniert für beliebige zusammengesetzte Zahlen n.

9.3 Das Quadratische Sieb

Einer der effizientesten Faktorisierungsalgorithmen ist das Quadratische Sieb (QS), das in diesem Abschnitt beschrieben wird.

9.3.1 Das Prinzip

Wieder soll die zusammengesetzte Zahl n faktorisiert werden. Wir beschreiben, wie man einen echten Teiler von n findet. Wenn n wie im RSA-Verfahren Produkt zweier Primzahlen ist, dann ist damit die Primfaktorzerlegung von n gefunden. Andernfalls müssen die gefundenen Faktoren ihrerseits faktorisiert werden.

Im Quadratischen Sieb werden ganze Zahlen x und y bestimmt, für die

$$x^2 \equiv y^2 \mod n \tag{9.1}$$

und

$$x \not\equiv \pm y \mod n \tag{9.2}$$

gilt. Dann ist n nämlich ein Teiler von $x^2-y^2 = (x-y)(x+y)$, aber weder von $x-y$ noch von $x + y$. Also ist $g = \gcd(x - y, n)$ ein echter Teiler von n und das Berechnungsziel ist erreicht.

Beispiel 9.3 Sei $n = 7429$, $x = 227$, $y = 210$. Dann ist $x^2 - y^2 = n$, $x - y = 17$, $x + y = 437$. Daher ist $\gcd(x - y, n) = 17$. Das ist ein echter Teiler von n.

9.3.2 Bestimmung von x und y

Das beschriebene Prinzip wird auch in anderen Faktorisierungsalgorithmen, z. B. im Zahlkörpersieb (Number Field Sieve, siehe [44]), angewendet. Die Verfahren unterscheiden sich aber in der Art und Weise, wie x und y berechnet werden. Wir beschreiben, wie das im Quadratischen Sieb gemacht wird.

Sei

$$m = \lfloor \sqrt{n} \rfloor$$

und

$$f(X) = (X + m)^2 - n.$$

Wir erläutern das Verfahren zuerst an einem Beispiel.

Beispiel 9.4 Wie in Beispiel 9.3 sei $n = 7429$. Dann ist $m = 86$ und $f(X) = (X + 86)^2 - 7429$. Es gilt

$$\begin{aligned} f(-3) &= 83^2 - 7429 = -540 = -1 * 2^2 * 3^3 * 5 \\ f(1) &= 87^2 - 7429 = 140 = 2^2 * 5 * 7 \\ f(2) &= 88^2 - 7429 = 315 = 3^2 * 5 * 7. \end{aligned}$$

Hieraus folgt

$$83^2 \equiv -1 * 2^2 * 3^3 * 5 \bmod 7429$$
$$87^2 \equiv 2^2 * 5 * 7 \bmod 7429$$
$$88^2 \equiv 3^2 * 5 * 7 \bmod 7429$$

Multipliziert man die letzten beiden Kongruenzen, so erhält man

$$(87 * 88)^2 \equiv (2 * 3 * 5 * 7)^2 \bmod n.$$

Man kann also

$$x = 87 * 88 \bmod n = 227, \quad y = 2 * 3 * 5 * 7 \bmod n = 210$$

setzen. Das sind die Werte für x und y aus Beispiel 9.3.

In Beispiel 9.4 werden Zahlen s angegeben, für die $f(s)$ nur kleine Primfaktoren hat. Es wird ausgenutzt, dass

$$(s + m)^2 \equiv f(s) \bmod n \tag{9.3}$$

ist, und es werden Kongruenzen der Form (9.3) ausgewählt, deren Produkt auf der linken und rechten Seite ein Quadrat ergibt. Auf der linken Seite einer Kongruenz (9.3) steht ohnehin ein Quadrat. Das Produkt beliebiger linker Seiten ist also immer ein Quadrat. Auf der rechten Seite ist die Primfaktorzerlegung bekannt. Man erhält also ein Quadrat, wenn die Exponenten aller Primfaktoren und der Exponent von -1 gerade sind. Wir erklären als nächstes, wie geeignete Kongruenzen ausgewählt werden und anschließend, wie die Kongruenzen bestimmt werden.

9.3.3 Auswahl geeigneter Kongruenzen

In Beispiel 9.4 kann man direkt sehen, welche Kongruenzen multipliziert werden müssen, damit das Produkt der rechten Seiten der Kongruenzen ein Quadrat ergibt. Bei großen Zahlen n muss man die geeigneten Kongruenzen aus mehr als 100 000 möglichen Kongruenzen auswählen. Dann wendet man lineare Algebra an. Dies wird im nächsten Beispiel illustriert.

Beispiel 9.5 Wir zeigen, wie die Auswahl der geeigneten Kongruenzen in Beispiel 9.4 durch Lösung eines linearen Gleichungssystems erfolgt. Drei Kongruenzen stehen zur Wahl. Daraus sollen die Kongruenzen so ausgewählt werden, dass das Produkt der rechten Seiten ein Quadrat ergibt. Man sucht also Zahlen $\lambda_i \in \{0, 1\}$, $1 \leq i \leq 3$, für die

$$(-1 * 2^2 * 3^3 * 5)^{\lambda_1} * (2^2 * 5 * 7)^{\lambda_2} * (3^2 * 5 * 7)^{\lambda_3}$$
$$= (-1)^{\lambda_1} * 2^{2\lambda_1+2\lambda_2} * 3^{3\lambda_1+2\lambda_3} * 5^{\lambda_1+\lambda_2+\lambda_3} * 7^{\lambda_2+\lambda_3}$$

ein Quadrat ist. Diese Zahl ist genau dann ein Quadrat, wenn der Exponent von -1 und die Exponenten aller Primzahlen gerade sind. Man erhält also das folgende Kongruenzensystem:

$$\begin{aligned}
\lambda_1 &\equiv 0 \bmod 2\\
2\lambda_1 + 2\lambda_2 &\equiv 0 \bmod 2\\
3\lambda_1 + 2\lambda_3 &\equiv 0 \bmod 2\\
\lambda_1 + \lambda_2 + \lambda_3 &\equiv 0 \bmod 2\\
\lambda_2 + \lambda_3 &\equiv 0 \bmod 2.
\end{aligned}$$

Die Koeffizienten der Unbekannten λ_i kann man modulo 2 reduzieren. Man erhält dann das vereinfachte System

$$\begin{aligned}
\lambda_1 &\equiv 0 \bmod 2\\
\lambda_1 + \lambda_2 + \lambda_3 &\equiv 0 \bmod 2\\
\lambda_2 + \lambda_3 &\equiv 0 \bmod 2.
\end{aligned}$$

Daraus erhält man die Lösung

$$\lambda_1 = 0, \quad \lambda_2 = \lambda_3 = 1.$$

Wir skizzieren kurz, wie das Quadratische Sieb geeignete Kongruenzen im allgemeinen findet.

Man wählt eine natürliche Zahl B. Gesucht werden kleine ganze Zahlen s, für die $f(s)$ nur Primfaktoren in der *Faktorbasis*

$$F(B) = \{p \in \mathbb{P} : p \leq B\} \cup \{-1\}$$

hat. Solche Werte $f(s)$ heißen B-glatt. Tab. 9.1 gibt einen Eindruck von den Faktorbasisgrößen. Hat man soviele Zahlen s gefunden, wie die Faktorbasis Elemente hat, so stellt man das entsprechende lineare Kongruenzensystem auf und löst es. Da das Kongruenzensystem tatsächlich ein lineares Gleichungssystem über dem Körper $\mathbb{Z}/2\mathbb{Z}$ ist, kann man zu seiner Lösung den Gauß-Algorithmus verwenden. Da aber die Anzahl der Gleichungen und die Anzahl der Variablen sehr groß ist, verwendet man statt dessen spezialisierte Verfahren, auf die wir hier aber nicht näher eingehen.

9.3.4 Das Sieb

Es bleibt noch zu klären, wie die Zahlen s gefunden werden, für die $f(s)$ B-glatt ist. Man könnte für $s = 0, \pm 1, \pm 2, \pm 3, \ldots$ den Wert $f(s)$ berechnen und dann durch Probedivision ausprobieren, ob $f(s)$ B-glatt ist. Das ist aber sehr aufwendig. Um herauszufinden,

Tab. 9.1 Siebintervall- und Faktorbasisgrößen

# Dezimalstellen von n	50	60	70	80	90	100	110	120
# Faktorbasis in Tausend	3	4	7	15	30	51	120	245
# Siebintervall in Millionen	0,2	2	5	6	8	14	16	26

dass $f(s)$ nicht B-glatt ist, muss man nämlich durch alle Primzahlen $p \leq B$ dividieren. Wie man in Tab. 9.1 sieht, sind die Faktorbasen sehr groß, und daher kann die Probedivison sehr lange dauern. Schneller geht ein Siebverfahren.

Wir beschreiben eine vereinfachte Form des Siebverfahrens. Man fixiert ein *Siebintervall*

$$S = \{-C, -C+1, \ldots, 0, 1, \ldots, C\}.$$

Gesucht werden alle $s \in S$,für die $f(s)$ B-glatt ist. Man berechnet zuerst alle Werte $f(s)$, $s \in S$. Für jede Primzahl p der Faktorbasis dividiert man alle Werte $f(s)$ durch die höchstmögliche p-Potenz. Die B-glatten Werte $f(s)$ sind genau diejenigen, bei denen eine 1 oder -1 übrigbleibt.

Um herauszufinden, welche Werte $f(s) = (s+m)^2 - n$ durch eine Primzahl p der Faktorbasis teilbar sind, bestimmt man zuerst die Zahlen $s \in \{0, 1, \ldots, p-1\}$, für die $f(s)$ durch p teilbar ist. Da das Polynom $f(X)$ höchstens zwei Nullstellen modulo p hat, sind das entweder zwei, eine oder keine Zahl s. Für kleine Primzahlen kann man die Nullstellen durch ausprobieren finden. Ist p groß, muss man andere Methoden anwenden (siehe [4]). Geht man von diesen Nullstellen in Schritten der Länge p nach rechts und links durch das Siebintervall, so findet man alle s-Werte, für die $f(s)$ durch p teilbar ist. Diesen Vorgang nennt man *Sieb* mit p. Man dividiert dabei nur die teilbaren Werte $f(s)$. Es gibt keine erfolglosen Probedivisionen mehr.

Beispiel 9.6 Wie in Beispiel 9.3 und Beispiel 9.4 sei $n = 7429$, $m = 86$ und $f(X) = (X+86)^2 - 7429$. Als Faktorbasis wähle die Menge $\{2, 3, 5, 7\} \cup \{-1\}$ und als Siebintervall die Menge $\{-3, -2, \ldots, 3\}$. Das Sieb ist in Tab. 9.2 dargestellt.

Das Sieb kann noch sehr viel effizienter gestaltet werden. Das wird hier aber nicht weiter beschrieben. Wir verweisen statt dessen auf [58].

Tab. 9.2 Das Sieb

s	-3	-2	-1	0	1	2	3
$(s+m)^2 - n$	-540	-373	-204	-33	140	315	492
Sieb mit 2	-135		-51		35		123
Sieb mit 3	-5		-17	-11		35	41
Sieb mit 5	-1				7	7	
Sieb mit 7					1	1	

9.4 Analyse des Quadratischen Siebs

In diesem Abschnitt skizzieren wir die Analyse des Quadratischen Siebs, damit der Leser einen Eindruck erhält, warum das Quadratische Sieb effizienter ist als Probedivision. Die in der Analyse verwendeten Techniken gehen über den Rahmen dieses Buches hinaus. Darum werden sie nur angedeutet. Interessierten Lesern wird als Einstieg in eine vertiefte Beschäftigung mit dem Gegenstand [43] empfohlen.

Seien n, u, v reelle Zahlen und sei n größer als die Eulersche Konstante e. Dann schreibt man

$$L_n[u,v] = e^{v(\log n)^u(\log\log n)^{1-u}}. \tag{9.4}$$

Diese Funktion wird zur Beschreibung der Laufzeit von Faktorisierungsalgorithmen verwendet. Wir erläutern zuerst ihre Bedeutung.

Es ist

$$L_n[0,v] = e^{v(\log n)^0(\log\log n)^1} = (\log n)^v \tag{9.5}$$

und

$$L_n[1,v] = e^{v(\log n)^1(\log\log n)^0} = e^{v\log n}. \tag{9.6}$$

Ein Algorithmus, der die Zahl n faktorisieren soll, erhält als Eingabe n. Die binäre Länge von n ist $\lfloor \log_2 n \rfloor + 1 = O(\log n)$. Hat der Algorithmus die Laufzeit $L_n[0,v]$, so ist seine Laufzeit polynomiell, wie man aus (9.5) sieht. Dabei ist v der Grad des Polynoms. Der Algorithmus gilt dann als effizient. Die praktische Effizienz hängt natürlich vom Polynomgrad ab. Hat der Algorithmus die Laufzeit $L_n[1,v]$, so ist seine Laufzeit exponentiell, wie man aus (9.6) sieht. Der Algorithmus gilt als ineffizient. Hat der Algorithmus die Laufzeit $L_n[u,v]$ mit $0 < u < 1$, so ist heißt seine Laufzeit *subexponentiell*. Sie ist schlechter als polynomiell und besser als exponentiell. Die schnellsten Faktorisierungsalgorithmen haben subexponentielle Laufzeit.

Die Laufzeit der Probedivision zur Faktorisierung ist exponentiell.

Die Laufzeit des Quadratischen Siebs konnte bis jetzt nicht völlig analysiert werden. Wenn man aber einige plausible Annahmen macht, dann ist die Laufzeit des Quadratischen Siebs $L_n[1/2, 1+o(1)]$. Hierin steht $o(1)$ für eine Funktion, die gegen 0 konvergiert, wenn n gegen Unendlich strebt. Die Laufzeit des Quadratischen Siebs liegt also genau in der Mitte zwischen polynomiell und exponentiell.

Wir begründen die Laufzeit des Quadratischen Siebs. Im Quadratischen Sieb werden Schranken B und C festgelegt und dann werden diejenigen Zahlen s im Siebintervall $S = \{-C, -C+1, \ldots, C\}$ bestimmt, für die

$$f(s) = (s+m)^2 - n = s^2 + 2ms + m^2 - n \tag{9.7}$$

B-glatt ist. Die Schranken B und C müssen so gewählt sein, dass die Anzahl der gefundenen Werte s genauso groß ist wie die Anzahl der Elemente der Faktorbasis.

Da $m = \lfloor \sqrt{n} \rfloor$, ist $m^2 - n$ sehr klein. Für kleines s ist $f(s)$ nach (9.7) daher in derselben Größenordnung wie $\sqrt{n}$. Wir nehmen an, dass der Anteil der B-glatten Werte $f(s)$, $s \in S$,

genauso groß ist wie der Anteil der B-glatten Werte aller natürlichen Zahlen $\leq \sqrt{n}$. Diese Annahme wurde nie bewiesen, und es ist auch unklar, wie sie bewiesen werden kann. Sie ist aber experimentell verifizierbar und ermöglicht die Analyse des Quadratischen Siebs.

Die Anzahl der B-glatten natürlichen Zahlen unter einer Schranke x wird mit $\psi(x, B)$ bezeichnet. Sie wird im folgenden Satz abgeschätzt, der in [24] bewiesen wurde.

Theorem 9.1 *Sei ε eine positive reelle Zahl. Dann gilt für alle reellen Zahlen $x \geq 10$ und $w \leq (\log x)^{1-\varepsilon}$*

$$\psi(x, x^{1/w}) = x w^{-w+f(x,w)}$$

für eine Funktion f, die $f(x, w)/w \to 0$ für $w \to \infty$ und gleichmäßig für alle x erfüllt.

Theorem 9.1 bedeutet, dass der Anteil der $x^{1/w}$-glatten Zahlen, die kleiner gleich x sind, ungefähr w^{-w} ist.

Aus diesem Satz lässt sich folgendes Resultat ableiten:

Korollar 9.1 *Seien a, u, v positive reelle Zahlen. Dann gilt für $n \in \mathbb{N}, n \to \infty$*

$$\psi(n^a, L_n[u, v]) = n^a L_n[1-u, -(a/v)(1-u) + o(1)].$$

Beweis Es ist

$$L_n[u, v] = (e^{(\log n)^u (\log\log n)^{1-u}})^v = n^{v((\log\log n)/\log n)^{1-u}}.$$

Setzt man also

$$w = (a/v)((\log n)/(\log\log n))^{1-u}$$

und wendet Theorem 9.1 an, so erhält man

$$\psi(n^a, L_n[u, v]) = n^a w^{-w(1+o(1))}.$$

Nun ist

$$w^{-w(1+o(1))} = (e^{(1-u)(\log(a/v)+\log\log n-\log\log\log n)(-(a/v)((\log n)/(\log\log n))^{1-u}(1+o(1)))})^{1-u}.$$

Hierin ist

$$\log(a/v) + \log\log n - \log\log\log n = \log\log n(1 + o(1)).$$

Daher ist

$$\begin{aligned} & w^{-w(1+o(1))} \\ &= e^{(\log n)^{1-u}(\log\log n)^u(-(a/v)(1-u)+o(1))} \\ &= L_n[1-u, -(a/v)(1-u) + o(1)]. \end{aligned}$$

Damit ist die Behauptung bewiesen. □

Im Quadratischen Sieb werden Zahlen $f(s)$ erzeugt, die von der Größenordnung $n^{1/2}$ sind. In Korollar 9.1 ist also $a = 1/2$. Um ein s zu finden, für das $f(s)$ $L_n[u, v]$-glatt ist, braucht man nach Korollar 9.1 $L_n[1-u, (1/(2v))(1-u)+o(1)]$ Elemente im Siebintervall. Die Anzahl der Elemente der Faktorbasis ist höchstens $L_n[u, v]$. Insgesamt braucht man also $L_n[u, v]$ solche Werte s, damit man das Gleichungssystem lösen kann. Die Zeit zur Berechnung der passenden Werte für s ist also ein Vielfaches von $L_n[u, v]L_n[1-u, (1/(2v))(1-u)+o(1)]$. Dieser Wert wird minimal für $u = 1/2$. Wir wählen also $u = 1/2$.

Die Faktorbasis enthält also alle Primzahlen $p \leq B = L_n[1/2, v]$. Für jede erfolgreiche Zahl s braucht das Siebintervall $L_n[1/2, 1/(4v)]$ Elemente. Da insgesamt $L_n[1/2, v]$ erfolgreiche Werte s berechnet werden müssen, ist die Größe des Siebintervalls $L_n[1/2, v]L_n[1/2, 1/(4v)] = L_n[1/2, v + 1/(4v)]$.

Ein geeigneter Wert für v wird in der Analyse noch gefunden. Wir tragen zuerst die Laufzeiten für die einzelnen Schritte zusammen.

Die Berechnung der Quadratwurzeln von $f(X)$ modulo p für eine Primzahl p in der Faktorbasis ist in erwarteter Polynomzeit möglich. Daraus leitet man ab, dass die Zeit zur Berechnung der Wurzeln für alle Faktorbasiselemente $L_n[1/2, v + o(1)]$ ist.

Die Siebzeit pro Primzahl p ist $O(L_n[1/2, v+1/(4v)+o(1)]/p)$, weil man in Schritten der Länge p durch ein Intervall der Länge $L[1/2, v]$ geht. Daraus kann man ableiten, dass die gesamte Siebzeit einschließlich der Vorberechnung $L_n[1/2, v + 1/(4v) + o(1)]$ ist.

Mit dem Algorithmus von Wiedemann, der ein spezialisierter Gleichungslöser für dünn besetzte Systeme ist, benötigt die Lösung des Gleichungssystems Zeit $L_n[1/2, 2v+o(1)]$. Der Wert $v = 1/2$ minimiert die Siebzeit und macht Siebzeit und Zeit zum Gleichungslösen gleich. Wir erhalten also insgesamt die Laufzeit $L_n[1/2, 1 + o(1)]$.

9.5 Effizienz anderer Faktorisierungsverfahren

Nach der Analyse des Quadratischen Siebs im letzten Abschnitt stellen sich zwei Fragen: Gibt es effizientere Faktorisierungsalgorithmen und gibt es Faktorisierungsverfahren, deren Laufzeit man wirklich beweisen kann?

Der effizienteste Faktorisierungsalgorithmus, dessen Laufzeit bewiesen werden kann, benutzt quadratische Formen. Es handelt sich um einen probabilistischen Algorithmus mit erwarteter Laufzeit $L_n[1/2, 1+o(1)]$. Seine Laufzeit entspricht also der des Quadratischen Siebs. Die Laufzeit wurde in [45] bewiesen.

Die Elliptische-Kurven-Methode (ECM) ist ebenfalls ein probabilistischer Algorithmus mit erwarteter Laufzeit $L_p[1/2, \sqrt{1/2}]$ wobei p der kleinste Primfaktor von n ist. Während das Quadratische Sieb für Zahlen n gleicher Größe gleich lang braucht, wird ECM schneller, wenn n einen kleinen Primfaktor hat. Ist der kleinste Primfaktor aber von der Größenordnung $\sqrt{n}$, dann hat ECM die erwartete Laufzeit $L_n[1/2, 1]$ genau wie das Quadratische Sieb. In der Praxis ist das Quadratische Sieb in solchen Fällen sogar schneller.

Bis 1988 hatten die schnellsten Faktorisierungsalgorithmen die Laufzeit $L_n[1/2, 1]$. Es gab sogar die Meinung, dass es keine schnelleren Faktorisierungsalgorithmen geben könne. Wie man aus 9.1 sieht, ist das auch richtig, solange man versucht, natürliche Zahlen n mit glatten Zahlen der Größenordnung n^a für eine feste positive reelle Zahl a zu faktorisieren. Im Jahre 1988 zeigte aber John Pollard, dass es mit Hilfe der algebraischen Zahlentheorie möglich ist, zur Erzeugung der Kongruenzen (9.1) und (9.2) systematisch kleinere Zahlen zu verwenden. Aus der Idee von Pollard wurde das Zahlkörpersieb (Number Field Sieve, NFS). Unter geeigneten Annahmen kann man zeigen, dass es die Laufzeit $L_n[1/3, (64/9)^{1/3}]$ hat. Es ist damit einem Polynomzeitalgorithmus wesentlich näher als das Quadratische Sieb. Eine Sammlung von Arbeiten zum Zahlkörpersieb findet man in [44].

In den letzten zwanzig Jahren hat es dramatische Fortschritte bei der Lösung des Faktorisierungsproblems gegeben. Shor [69] hat bewiesen, dass auf Quantencomputern natürliche Zahlen in Polynomzeit faktorisiert werden können. Es ist nur nicht klar, ob und wann es entsprechende Quantencomputer geben wird. Aber es ist auch möglich, dass ein klassischer polynomieller Faktorisierungsalgorithmus gefunden wird. Mathematische Fortschritte sind eben nicht voraussagbar. Dann sind das RSA-Verfahren, das Rabin-Verfahren und all die anderen Verfahren, die ihre Sicherheit aus der Schwierigkeit des Faktorisierungsproblems beziehen, unsicher.

Aktuelle Faktorisierungsrekorde findet man zum Beispiel in [27]. So konnte im Dezember 2009 die von den RSA-Laboratories veröffentlichte 232-stellige Challenge-Zahl RSA-768 mit dem Zahlkörpersieb faktorisiert werden.

9.6 Übungen

Übung 9.1 (Fermat-Faktorisierungsmethode) Fermat faktorisierte eine Zahl n, indem er eine Darstellung $n = x^2 - y^2 = (x - y)(x + y)$ berechnete. Faktorisieren Sie auf diese Weise $n = 13199$ möglichst effizient. Funktioniert diese Methode immer? Wie lange braucht die Methode höchstens?

Übung 9.2 Faktorisieren Sie 831802500 mit Probedivision.

Übung 9.3 Faktorisieren Sie $n = 138277151$ mit der $p - 1$-Methode.

Übung 9.4 Faktorisieren Sie $n = 18533588383$ mit der $p - 1$-Methode.

Übung 9.5 Schätzen Sie die Laufzeit der $(p - 1)$-Methode ab.

Übung 9.6 Die *Random-Square-Methode* von Dixon ist der Quadratisches-Sieb-Methode ähnlich. Der Hauptunterschied besteht darin, dass die Relationen gefunden werden, indem $x^2 \bmod n$ faktorisiert wird, wobei x eine Zufallszahl in $\{1, \ldots, n - 1\}$ ist. Verwenden

Sie die Random-Square-Methode, um 11111 mit einer möglichst kleinen Faktorbasis zu faktorisieren.

Übung 9.7 Finden Sie mit dem quadratischen Sieb einen echten Teiler von 11111.

Übung 9.8 Zeichnen Sie die Funktion $f(k) = L_{2^k}[1/2, 1]$ für $k \in \{1, 2, \ldots, 2048\}$.