Faktorisierung natürlicher Zahlen

Thorsten Reinecke

Juni 1999 (überarbeitet 11/2007)

1 Einführung und Motivation

Den Vorgang des Aufspaltens natürlicher Zahlen in ihre Primfaktoren (Primfaktorzerlegung) bezeichnet man als Faktorisierung. Jede natürliche Zahl (die größer ist als 1) lässt sich (bis auf die Reihenfolge der Primfaktoren) eindeutig in ihre Primfaktoren zerlegen. Das ist eine einfache Erkenntnis und Grundlage vieler zahlentheoretischer Betrachtungen.

Die Multiplikation zweier natürlicher Zahlen ist nicht schwer. Sie kann aus algorithmischer Sicht in zufriedenstellender Komplexität bewerkstelligt werden, auch wenn die betroffenen Zahlen sehr groß sind. Prinzipiell geeignete Verfahren hierzu lernt bereits jedes Kind in der Schule. So ist es beispielsweise kein großes Problem, das Produkt 7*3=21 zu berechnen. Die Umkehrung des Problems, also die Zahl 21 in seine Faktoren 3 und 7 zu zerlegen, erscheint daher ebenfalls trivial. Für dieses spezielle Problem ist es das auch, da die Faktorisierung der Zahl 21 bereits bekannt ist und man im Zweifelsfall durch einfaches Probieren (Probedivision mit verschiedenen erratenen Faktoren) schnell zum Ziel gelangt.

Bereits das Beispiel 37*49999=1849963 stellt eine größere Herausforderung dar. Üblicherweise wird man hier für das Ausmultiplizieren zum Taschenrechner greifen oder sich anderer Hilfsmittel (Stift und Papier) bedienen; zur Not erzielt man das Ergebnis auch noch mit Kopfrechnung. Die Bewerkstelligung solcher Multiplikationsaufgaben stellt allerdings keine besonderen intellektuellen Fähigkeiten an den Rechner – und zwar unabhängig davon, ob dieser menschlich ist oder nicht. Es kommt allein darauf an, ein entsprechendes Multiplikationsverfahren verinnerlicht zu haben. Die herausragenden Fähigkeiten mancher Menschen im Kopfrechnen belegen also weniger die Tatsache, dass solche Menschen gute Mathematiker sind, als vielmehr die Tatsache, dass die angewendeten Verfahren zur Multiplikation so effizient und einfach sind, dass sie sogar im Kopf beherrscht werden können.

Wenn man nun umgekehrt fragt, welche Primfaktorzerlegung die Zahl 1849963 besitzt, so wird die Antwort vermutlich etwas länger dauern. Dies liegt daran, dass man die Faktorisierung in den meisten Fällen nicht auswendig kennt, sondern erst irgendwie rekonstruieren muss. Und wirklich effiziente Verfahren zur Primfaktorzerlegung beliebiger Zahlen sind derzeit leider nicht bekannt. Die Diskrepanz zwischen der Lösung der Multiplikationsaufgabe und der dazu inversen Faktorisierung nimmt also mit der Größe der betroffenen Zahlen zu.

Nun muss zunächst einmal darauf hingewiesen werden, dass die Ermittlung der Primfaktorzerlegung einer einzelnen Zahl zwar durchaus interessant sein kann, aber aus wissenschaftlicher Sicht keinen Meilenstein darstellt. Das ist deshalb so, weil die pure Angabe der Zerlegung einer konkreten Zahl nicht dazu beitragen kann, auch beliebige andere Zahlen zu faktorisieren.

Vielmehr ist das Ziel darin zu suchen, die Primfaktorzerlegung einer konkret vorgegebenen Zahl zu einer mathematischen Trivialität verkommen zu lassen. Das Finden einer Zerlegung sollte eine möglichst langweilige und stupide Sache sein; so langweilig und stupide, dass man sie nicht nur einem Computer überlassen kann, sondern dass man es darüberhinaus nicht einmal mehr nötig hat, sich eine konkrete Faktorisierung zu merken. So langweilig, dass man nicht mehr gespannt darauf wartet, wie die Faktorisierung wohl aussehen könnte. So langweilig, dass sich eigentlich niemand mehr für die Primfaktorzerlegung einer konkreten Zahlen interessiert.

Im Augenblick ist das leider nicht so! Und allein das macht die Sache interessant. So interessant, dass man mit der Primfaktorzerlegung von Zahlen Geld verdienen kann. So interessant, dass selbst Geheimdienste, wenn sie ein solches Verfahren gefunden hätten, dieses vermutlich nicht verraten würden. So interessant, dass manche Staaten es ihren Bürgern im Prinzip verbieten, zwei große Primzahlen miteinander zu multiplizieren und das Ergebnis der Multiplikation zu veröffentlichen, ohne dem Staat die Primfaktoren vorher zu verraten.[1] So interessant, dass man damit Daten verschlüsseln kann, die nur diejenigen Personen unberechtigt entschlüsseln können, denen es gelingt, die Primfaktorzerlegung einer solchermaßen verwendeten Zahl zu rekonstruieren.

Das ist ein unbefriedigender Zustand, denn er erhebt eine Sache in den Mittelpunkt, die dort eigentlich nicht stehen sollte. Es wäre daher wünschenswert, Verfahren zu entwickeln, die es schaffen, Zahlen effizient zu faktorisieren. Das Problem ist aber so schwer, dass man derzeit noch nicht einmal genau sagen kann, wie schwer ist eigentlich ist. Das bedeutet: man hat bisher weder einen zufriedenstellenden Algorithmus zur Faktorisierung von Zahlen gefunden, noch weiß man, ob es einen solchen überhaupt gibt. Der einzige Lichtblick am Horizont: Es wurde ein Verfahren entdeckt, das mit Hilfe einer bisher noch nicht entwickelten Computertechnologie (Quantenrechner) jede Primfaktorzerlegung effizient finden würde.

Das Faktorisierungsproblem stellt also eine ungeahnte Schnittstelle zwischen Mathematik, Informatik und Technik dar, welches nicht nur theoretisch interessant ist, sondern darüberhinaus auch im kryptologischen Bereich Anwendung findet. Die gesellschaftlichen Auswirkungen sind so gravierend, dass sie sich sowohl wirtschaftlich als auch kriminell verwerten lassen und deshalb sogar staatliche Interessen tangieren.

2 Grundlegendes

2.1 Primzahlen

Eine natürliche Zahl, die genau zwei verschiedene Teiler (nämlich sich selbst und die Eins) besitzt, bezeichnet man als Primzahl. Jede natürliche Zahl, die größer ist als Eins, ist entweder eine solche Primzahl oder lässt sich auf eindeutige Weise (bis auf die Reihenfolge der Faktoren) in ein Produkt von (nicht notwendigerweise verschiedenen) Primzahlen zerlegen.

Die Zahl 1 ist deshalb keine Primzahl, weil sie das neutrale Element der Multiplikation darstellt und man sie deshalb beliebig oft mit einer Zahl multiplizieren könnte, ohne dass sich das Ergebnis jemals ändern würde. Die Eindeutigkeit der Primfaktorzerlegung würde unnötig verletzt werden, ließe man die Zahl 1 als Primzahl zu.

Ein bereits seit der Antike bekanntes Verfahren zur Ermittlung von Primzahlen ist das sogenannte Sieb des Eratosthenes: Man schreibt alle natürlichen Zahlen bis zu einer vorher gewählten Obergrenze beginnend mit der Zwei auf. Nun unterstreicht man die Zwei, denn sie ist eine Primzahl. In Zweierschritten streicht man nun alle aufgeschriebenen Vielfachen der Zwei durch, denn sie sind keine Primzahlen. Nun sucht man diejenige Zahl, die nach der Zwei kommt und noch nicht durchgestrichen wurde: es ist die Drei. Man unterstreicht sie, denn sie ist eine Primzahl. Nun streicht man alle Vielfachen der Drei, in dem man in Dreierschritten durch die Zahlenliste geht. Das Verfahren des Suchens der nächsten Primzahl und des Streichens der Vielfachen der Zahl wiederholt man solange, bis das Quadrat der gefundenen Primzahl größer ist als die vorher festgesetzte Obergrenze, bis zu der man die Primzahlen ermitteln wollte. Alle nicht durchgestrichenen Zahlen in dieser Liste sind prim (sofern man beim Ausstreichen der Vielfachen keinen Fehler begangen hat).

Eine ebenfalls seit dem Altertum bekannte Tatsache ist, dass unendlich viele Primzahlen existieren. Der Beweis lässt sich durch Widerspruch führen: Wenn es nur endlich viele Primzahlen gäbe, so wäre das Produkt dieser Zahlen eine natürliche Zahl. Zu dieser Zahl addieren wir die Zahl Eins. Die so entstandene Zahl ist aber nicht durch eine der vorher verwendeten Primzahlen teilbar. Also muss es weitere Primzahlen geben. Dies steht eindeutig im Widerspruch zu der Annahme, dass wir bereits das Produkt sämtlicher Primzahlen gebildet hätten.

[1] Hiermit ist das RSA-Verfahren gemeint. Das Produkt der Primzahlen dient bei diesem Public-Key-Verschlüsselungsverfahren als öffentlicher Schlüssel. Wenn die Primfaktorzerlegung der Zahl bekannt ist, bricht das Verfahren für die gewählte Zahl zusammen. Das erklärt, dass übergeordnete Instanzen gern im Besitz der zur Entschlüsselung erforderlichen Informationen kommen wollen: Sie können dann auch die verschlüsselten Daten ohne großen Aufwand abhören und entschlüsseln. – Man sollte daraus jedoch keineswegs den Schluss ziehen, staatliche Instanzen wären ohne diese Informationen hilflos: Es ist keineswegs klar, ob staatliche Instanzen derzeit nicht in der Lage sind, kryptographische Probleme wesentlich effizienter zu lösen, als man vermuten würde. Man kann im Gegenteil davon ausgehen, dass Geheimdiensten diesbezüglich ein Vorsprung von mehreren Jahren Forschungs- und Entwicklungsarbeit auf diesem Sektor gegenüber der öffentlichen Forschung und Entwicklung zugestanden werden muss.

Primzahlen sind ein Forschungsgegenstand der Zahlentheorie. Trotz ausgiebigen Studiums sind nach wie vor noch viele Probleme, die im Zusammenhang mit Primzahlen stehen, ungelöst.

2.2 Zusammengesetzte Zahlen

Eine natürliche Zahl, die das Produkt mehrerer Primzahlen ist, bezeichnen wir als zusammengesetzt. Für eine natürliche Zahl, die größer ist als 1, gibt es prinzipiell zwei verschiedene Möglichkeiten, diese als zusammengesetzt zu erkennen:

- Man findet einen nichttrivialen Teiler der Zahl. Ein nichttrivialer Teiler teilt die gegebene Zahl so, dass weder Eins noch die Zahl selbst übrig bleiben.
- Man zeigt, dass die gegebene Zahl keine Primzahl ist. Beispielsweise könnte man das vorgangs genannte Sieb hernehmen und nachschauen, ob die Zahl dort durchgestrichen wurde.

Interessant ist dabei, dass es in der Mathematik tatsächlich schnelle Verfahren gibt, die auch ohne die Faktorisierung der Zahl (oft[2]) zeigen können, dass diese zusammengesetzt ist. Jedoch wurde erst im Jahr 2002 ein Verfahren bekannt, das bewiesenermaßen für jede Zahl schnell entscheiden kann, ob diese zusammengesetzt oder prim ist. Aber auch mit älteren (und einfachen) Verfahren kann man die meisten zusammengesetzten Zahlen schnell als zusammengesetzt entlarven, auch wenn damit oftmals keine Faktorisierung gefunden ist.

Als Beispiel ein Primzahltest (oder besser: ein Test für zusammengesetzte Zahlen), der auf dem kleinen Satz von Fermat basiert und für verschiedene Basen ausprobiert werden kann[3]:

function fermat(N,basis)
if $basis^{N-1} \equiv 1$ (mod N)
then return “N könnte Primzahl sein”
else return “N ist zusammengesetzt”
fi

2.3 Der größte gemeinsame Teiler

Manche haben schon einmal von ihm gehört: dem größten gemeinsamen Teiler. Das weitläufig bekannte Verfahren zur Ermittlung desselben ist gleichzeitig ein ziemlich dummes: Um den ggT(a,b) zu berechnen, ermittelt man zunächst die Primfaktorzerlegungen von a und b. Anschließend multipliziert man sämtliche Primzahlen aus der Zerlegung von a zusammen, die auch in b auftreten (taucht eine Primzahl mehrfach auf, so ist die minimale Anzahl des Auftretens in den beiden Zahlen a und b relevant!).

Mit den sogenannten euklidischen Algorithmus geht es einfacher. Wenn nämlich zwei Zahlen a und b einen gemeinsamen Teiler besitzen, dann besitzen sowohl die Summe als auch die Differenz dieser Zahlen den gemeinsamen Teiler ebenfalls.

Zur Berechnung des ggT(a,b) ermitteln wir zunächst die größere Zahl der beiden: max(a,b). Analog hierzu ist min(a,b) das Minimum. Für den Fall a=b sind wir übrigens bereits fertig! – Ansonsten setzen wir x:=max(a,b) und ziehen nun solange min(a,b) von x ab, bis x<min(a,b). Falls nun der Fall eintreten sollte, dass x=0, so ist min(a,b) der gesuchte ggT. Andernfalls setzen wir a:=min(a,b) und b:=x; und wiederholen den Vorgang. Da die Zahlen immer kleiner, aber nie negativ werden, terminiert das Verfahren. - Zur Beschleunigung kann man statt wiederholter Subtraktion von min(a,b) auch den gleich den Rest der Division durch min(a,b) verwenden. Am Ergebnis ändert sich dadurch nichts!

[2] Die üblicherweise eingesetzten probabilistischen Algorithmen geben als Antwort entweder zurück, dass die Zahl beweisbar zusammengesetzt ist oder sie geben zurück, dass sie keinen Beleg für die Zusammengesetztheit gefunden haben. Für die letztgenannten Zahlen ist damit also nicht bewiesen, dass sie tatsächlich prim sind.

Erst wenn wir also einen deterministischen Algorithmus hätten, der bewiesenermaßen für jede Zahl schnell nachweisen könnte, ob sie zusammengesetzt ist, dann hätten wir damit auch das komplementäre Problem entschieden. (Denn solange der Nachweis nicht gelungen ist, dass eine Zahl prim ist, könnte sie ja noch zusammengesetzt sein.) Im Jahr 2002 schließlich wurde ein solcher Algorithmus gefunden.

[3] Leider gibt es die sogenannten Carmichel-Zahlen, bei denen dieser Test auf allen Basen bis auf die Teiler von N die Zusammengesetztheit von N nicht entdeckt. Aber in der Praxis kann man mit dieser Methode oft schnell herausfinden, ob man eine Faktorisierung versuchen oder besser einen Primzahlbeweis führen sollte.

Eine weitere Verbesserung, besonders bei maschinennaher Programmierung lässt sich mit dem binären ggT-Algorithmus erreichen:

```
function bingcd(a,b)
a2=0; b2=0;
while (2 teilt a) do a:=a/2; a2:=a2+1; od;
while (2 teilt b) do b:=b/2; b2:=b2+1; od;
while (a≠b) do
if (b>a) then swap(a,b); fi;
a:=a-b;
while (2 teilt a) do a:=a/2; od;
od;
return a * 2^min{a2,b2)
```

Mit den beschriebenen Verfahren lässt sich der größte gemeinsame Teiler zweier Zahlen also auch ohne vorherige Primzahlzerlegung schnell ermitteln. Dies ist äußerst bedeutsam, wie wir bald sehen werden...

2.4 Schnelle Exponentiation

Wenn wir a^b berechnen wollen, so können wir dies bei einem kleinen b mit einer Schleife machen, die folgenden Sachverhalt ausnutzt: $a^0 = 1$ und $a^{b+1} = a * a^b$.

Für große b ergibt dies aber unnötig viele Schleifendurchläufe und ist nicht mehr praktikabel. – Besser ist es, den Exponenten b in Binärdarstellung zu bringen. Die erste Binärstelle repräsentiert den Wert $2^0 = 1$. Von einer Binärstelle zur nächsten gelangt man durch Verdoppelung, so dass die k-te Binärstelle den Wert 2^{k-1} repräsentiert. Diesen Binärstellen sind nun die Binärziffern 0 und 1 als Koeffizienten zugeordnet, die angeben, ob die k-te Binärstelle summiert wird. Ziel ist es also, den Wert von b als Summe von Zweierpotenzen darzustellen. Was bringt uns diese Darstellung? – Nun, wir können jetzt die Potenzgesetze ausnutzen. Anstatt den Exponenten zu verdoppeln, können wir die Basis quadrieren. Von einer Binärstelle zur nächsten kommen wir also auch durch Ersetzen der Basis mit dem Quadrat der bisherigen Basis. – Wenn wir also den Exponenten b als Summe von Zweierpotenzen darstellen, so können wir den Ausdruck a^b als ein Produkt der zugehörigen Quadrate darstellen.

So verringert sich die Anzahl der erforderlichen Multiplikationen zur Auswertung des Terms a^{17} von 16 auf 5 Multiplikationen, wenn wir 17 als $2^0 + 2^4$ in Binärdarstellung bringen und dann den Term $a * (((a^2)^2)^2)^2$ auswerten.

Die Umsetzung des Verfahrens geschieht folgendermaßen: Die Ergebnisvariable wird mit 1 initialisiert. Solange der Exponent ungleich Null ist, werden die Schritte (a), (b) und (c) wiederholt. (a) Der Exponent wird (ganzzahlig) halbiert. (b) Tritt ein Divisionsrest auf, so wird die Ergebnisvariable mit der Basis multipliziert. (c) Anschließend wird die Basis quadriert.

2.5 Restklassen

Bei fortgesetzter Multiplikation "explodieren" die Zahlen sehr schnell. Oftmals benötigt man aber keine natürliche Zahl als Ergebnis, sondern nur einen Restklassenwert. Die Uhrzeit ist hierfür ein Beispiel: Drei Stunden nach 22 Uhr ist es 1 Uhr und nicht 25 Uhr.

Addition, Subtraktion, Multiplikation und sogar – in gewissem Sinne – Division bleiben gültig, wenn man in Restklassen rechnet. Die Zahlen, die als Reste einer ganzzahligen Division durch N verbleiben, also die Zahlen 0 bis N-1, bilden ein Repräsentantensystem der Restklassen von N. Alle Zahlen, die den selben Divisionsrest besitzen, gehören der selben Restklasse an. Restklassen sind Äquivalenzklassen. Um der Tatsache Rechnung zu tragen, dass zwei Ausdrücke der selben Restklasse angehören, verwenden wir das Kongruenzzeichen $\equiv$, das analog zum Gleichheitszeichen eingesetzt wird. Durch ein nachgestelltes (mod N) oder manchmal auch nur (N) geben wir zusätzlich an, worauf sich die Kongruenz bezieht. Der Ausdruck $a \equiv b$ (mod N) bedeutet also, dass die Divisionsreste der Werte a und b bezüglich der Division durch N identisch sind. Das Rechnen mit Kongruenzen ist einfach: Bei Addition, Subtraktion und Multiplikation rechnet man zunächst den "richtigen" Wert aus und dividiert ihn anschließend (ganzzahlig) durch N. Der verbleibende Rest ist das Ergebnis der Operation.

Auch die Exponentiation können wir auf mehrere (Restklassen-)Multiplikationen zurückführen. Lediglich die Division ist etwas problematischer und braucht uns hier nicht zu interessieren.[4]

Wenn eine Zahl N zusammengesetzt ist, so besitzt die Gleichung $a * b \equiv 0$ (mod N) neben den trivialen Lösungen a=0 bzw. b=0 auch nichttriviale Lösungen, für die sowohl a als auch b Vielfache von Teilern von N sind. Im letzteren Fall liefern uns dann ggT(a,N) und ggT(b,N) nichttriviale Teiler von N. Diese sind nicht notwendigerweise Primzahlen, sondern können ebenfalls noch zusammengesetzt sein.

Beispiel: $15 * 12 \equiv 0$ *(mod 20), da* $15 * 12 = 180$ *und* $180/20 = 9$ *mit Divisionsrest 0. Es ergibt sich somit, dass ggT(15,20)=5 und ggT(12,20)=4 jeweils nichttriviale Teiler von 20 sind. Triviale Teiler von 20 hingegen wären 20 und 1, wobei* $20 \equiv 0$ *(mod 20).*

3 Faktorisierungsverfahren

3.1 Probedivision

Als Primteiler einer zusammengesetzten Zahl kommen alle Primzahlen in Frage, die kleiner sind als die gegebene Zahl. Wenn man also einfach alle Primzahlen ausprobiert, dann wird man einen Teiler finden: Für alle Teiler der Zahl geht die Division ohne Rest auf.

Die Probedivision ist allerdings nur dann schnell, wenn man schnell die richtigen Zahlen geraten hat. Da jede zweite Zahl durch zwei teilbar ist, jede dritte durch drei, jede fünfte durch fünf, usw., empfiehlt es sich, die Probedivision beginnend mit der 2 in aufsteigender Reihenfolge vorzunehmen.

Es ist also vorteilhaft, mit der Probedivision jeweils den kleinsten Primfaktor einer Zahl zu ermitteln, die Zahl durch diesen Primfaktor zu dividieren und das Verfahren dann fortzusetzen, bis das Quadrat des betrachteten Primfaktors die Zahl überschreitet (und die Zahl daher prim ist). Man sollte dabei allerdings beachten, nach Abdividierung einer Primzahl den Test erneut mit dieser Primzahl fortzusetzen, damit man keine Primzahlpotenzen als Teiler übersieht.

Da die Ermittlung von Primzahlen allerdings ebenfalls Zeit benötigt, die u.U. länger dauert als Probedivisionen mit zusammengesetzten Zahlen, kann es günstiger sein, eine Zahlenfolge zu verwenden, die neben den Primzahlen auch zusammengesetzte Zahlen enthält, die sich aber schneller berechnen lässt. Bewährt hat es sich z.B., zunächst auf Teilbarkeit von 2 und 3 zu testen und anschließend die Probedivision sukzessive für alle Zahlen der Form $6 * n \pm 1$ fortzuführen.

Für das Finden größerer Primteiler braucht man allerdings sehr viel Geduld. Kleinere Primteiler findet man damit aber so schnell, dass man in der Praxis – also vor Anwendung schnellerer Faktorisierungsverfahren oder eines Primzahltests – immer eine Probedivision mit kleinen Primzahlen bis zu einer festgelegten Obergrenze durchführen sollte.

```
function trialdivision(N, Obergrenze)
while (2 teilt N) do N:=n/2; print(2 teilt); od;
while (3 teilt N) do N:=N/3; print (3 teilt); od;
```

[4] Die Restklassen-Division (modulo m) ist die Multiplikation mit dem Inversen des Divisors. Wenn das Inverse des Divisors nicht existiert (wenn der Divisor mit m einen gemeinsamen Teiler hat), dann ist die Division nicht definiert. Das Inverse kann z.B. folgendermaßen gefunden werden:

```
function invmod(x,m) // gibt 1/x (mod m) zurück
x:=x mod m;
if x=0 then error; fi; // Inverses existiert nicht
if x=1 then return 1; fi; // Inverses von 1 ist immer 1
v=x-invmod(m,x); // rekursiver Aufruf
return (v*m+1)/x;
```

Die Korrektheit der Funktion sieht man folgendermaßen: Sei $y \equiv \frac{1}{x}$ (mod m) unser gesuchtes Inverses. Offenbar genügt es, ein geeignetes v zu finden, so dass $y \equiv \frac{v*m+1}{x} \equiv \frac{1}{x}$ (mod m) erfüllt ist und $v * m + 1$ Vielfaches von x ist. Damit ist $v * m + 1 \equiv 0$ (mod x), woraus $v \equiv -\frac{1}{m} \equiv x - \frac{1}{m}$(mod x) folgt.

Hiermit lässt sich nun die Rekursion aufbauen. Da m mit jedem Rekursionsschritt kleiner wird und x zwischen 0 und $m - 1$ liegt (durch die modulo-Normierung), muss die Rekursion mit Erreichen des Falls x=0 (Fehlerausgabe, weil Inverses nicht existiert) oder x=1 (dem Erreichen des neutralen Elements der Multiplikation) enden.

```
t=5; d=2;
while (t≤Obergrenze) do
while (t teilt N) do N:=N/t; print(t teilt); od;
t=t+d; d=6-d;
od;
return N; // verbleibende Zahl
```

3.2 Fermat-Methode

Fermat benutzte einen Sachverhalt, der uns aus der Schulzeit geläufig ist, um große Faktoren zusammengesetzter Zahlen zu finden: eine binomische Formel.

Das Produkt zweier Zahlen $p * q$ lässt sich als $(a + b) * (a - b)$ notieren. Die Zahl a ist dabei die Mitte zwischen p und q; b ist die halbierte Differenz zwischen p und q. Wenn die zu faktorisierende Zahl ungerade ist, ist sichergestellt, dass a und b ganzzahlig sind: Alle Faktoren müssen dann ebenfalls ungerade sein, und da die Differenz zweier ungerader Zahlen immer gerade ist, lässt sie sich problemlos halbieren. Solange die zu faktorisierende Zahl gerade ist, dividieren wir die 2 als Primteiler heraus; übrig bleibt dann immer eine ungerade Zahl.

Multiplizieren wir den obigen Term aus, so erhalten wir $(a + b) * (a - b) = a^2 - b^2$. Alle Zahlen, die sich so darstellen lassen, können also auch als Differenz zweier Quadrate dargestellt werden. Für eine gegebene Zahl N setzen wir also $N = (a + b) * (a - b) = a^2 - b^2$ an und formen dies zu $a^2 - N = b^2$ um.

Nun kann man für verschiedene a's (beginnend mit dem aufgerundeten Wert von $\sqrt{N}$) testen, ob der Ausdruck $a^2 - N$ eine Quadratzahl ergibt. Falls dies der Fall sein sollte, ziehen wir daraus die Wurzel. Diese ist unser gesuchtes b. Eine Zerlegung von N (nicht notwendigerweise in Primfaktoren!) ergibt sich dann durch die Teiler (a+b) sowie (a-b).

Leider ist das Verfahren in dieser einfachen Form nicht effizienter als die Probedivision. Annähernd gleichgroße Faktoren findet man damit allerdings recht schnell.

Ein zeitraubendes Wurzelziehen kann man häufig dadurch vermeiden, in dem man vorher testet, ob eine Zahl überhaupt eine Quadratzahl sein kann. Beispielsweise können nur solche Zahlen Quadratzahlen sein, deren Rest bei der Division durch 8 entweder 0,1 oder 4 ergibt. (Dies weiß man entweder durch die Betrachtung der Moduloklasse - oder wenn man mit weniger Theorie auskommen will, klappt auch ein Induktionsbeweis.)

3.3 (p-1)-Methode

Die (p-1)-Methode ist schon deutlich kniffeliger und kann vollständig nur mit einem mathematischen Hintergrund vermittelt werden, der hier in Kürze leider nicht darstellbar ist.

Sie nutzt die unter dem Namen "Kleiner Fermatscher Satz" bekannte Tatsache aus, dass für jede Primzahl p der Ausdruck $a^{p-1} - 1$ durch p teilbar ist, unabhängig davon, welchen Wert wir für a einsetzen (sofern a und p teilerfremd sind).

Wenn nun p ein Teiler von unserer zu faktorisierenden Zahl N ist und (p-1) seinerseits ausschließlich in kleine Teiler zerlegbar ist, so lässt sich ein Potenzgesetz ausnutzen: $a^{t_1 * t_2 * \cdots * t_k} = (...((a^{t_1})^{t_2})^{\cdots})^{t_k}$

Zunächst wählen wir einen Startwert für die Basis a (z.B. a=2), dann potenzieren wir den Term mit einer kleinen Zahl b. Wir setzen $a := a^b$. Nun testen wir, ob das um 1 verminderte Ergebnis einen gemeinsamen Teiler mit N besitzt, ob also $ggT(a - 1, N) \neq 1$ ist. Falls ja, so ist ein Faktor gefunden. Falls nein, so raten[5] wir ein neues b und wiederholen den Vorgang mit der neuen Basis. Damit die Zahlen durch die Exponentiation nicht explosionsartig anwachsen, rechnen wir im Restklassensystem von N.

Falsch geratene b's, die keine Teiler von unserem p-1 sind, schaden bei der Methode nicht (allenfalls kosten sie unnötige Zeit), da der "Kleine Fermatsche Satz" unabhängig von der gewählten Basis gilt. Fast alle Primteiler beliebiger Zahlen lassen sich so finden, sofern sie um eins vermindert in nicht allzugroße Teiler (unsere b's, bzw. Teiler unserer b's) zerfallen.

[5] Am schönsten wäre es tatsächlich, wenn man die richtigen Teiler von (p-1) einfach raten könnte. Hilfsweise beschränkt man sich auf das Durchprobieren kleinerer Primzahlen (und Primzahlpotenzen) in aufsteigender Reihenfolge. Kleine Zahlen teilen häufiger!

3.4 Pollard-Rho-Methode

Hier wird ein besonderer Trick benutzt, um schnell zu großen Zahlen zu gelangen, die in viele Teiler zerfallen. Durch die Bildung des größten gemeinsamen Teilers dieser so erzeugten Zahlen mit der zu faktorisierenden Zahl testet man dann viele potentielle Teiler der Zahl in einem Abwasch.

Wie kann man nun große Zahlen erzeugen, die dann in viele kleine Teiler zerfallen? Auch hier hilft wieder die binomische Formel. Starten wir mit einem Wert a_0 und verwenden wir die Rekursionsformel $a_{k+1} := a_k^2 + c$, wobei c eine vorher festgelegte Konstante ist (z.B. c=1). Betrachten wir nun den Ausdruck $a_{2n} - a_n$. Dieser lässt sich folgendermaßen aufsplitten:

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

Alles, was nun noch zu tun bleibt, ist sowohl die Werte von a_{2n} als auch die Werte von a_n sukzessive auszurechnen. Beide Berechnungen führt man parallel durch, wobei der Ausdruck a_{2n} durch eine doppelte Ausführung der Berechnungsformel erzeugt werden kann. Die Berechnung der beiden Folgen packt man in eine Schleife, die man abbricht, wenn $ggT(a_{2n} - a_n, N) \neq 1$ wird. Dies ergibt dann den gefundenen Teiler, welcher allerdings nicht notwendigerweise prim sein muss.

Um eine explosionsartige Vergrößerung der Werte zu vermeiden, ist es auch hier erforderlich, modulo N zu rechnen, was die Korrektheit der Rechnung in keinerlei Weise beeinträchtigt. Das Verfahren ist somit recht einfach implementierbar und findet kleinere Teiler schnell. Für ein fest gewähltes c kann man sogar genau bestimmen, wie viele Schleifendurchläufe man benötigt, um den Teilbarkeitstest für vorgegebene Primzahlen abzudecken.

```
function pollard_rho(N,c)
a:=1; // a nimmt die Werte für a_i mod N an, Startwert a:=a_0 = 1
b:=1; // b nimmt die Werte für a_2i mod N an, Startwert b:=a_0 = 1
repeat
a:=a^2 + c (mod N);
b:=b^2 + c (mod N); b:=b^2 + c (mod N);
until gcd(b-a,N)≠ 1;
return gcd(b-a,N);
```

3.5 Quadratisches Sieb

Jetzt wird's richtig wild. Das Quadratische Sieb vereinigt die Idee des Siebs des Eratosthenes mit der Anwendung der binomischen Formeln und der Probedivision mit kleinen Zahlen. Das ganze Spiel wird auf den Restklassen der zu faktorisierenden Zahl N betrieben. Und da das noch zu wenig Mathematik ist, kommt noch ein Schuss linearer Algebra hinzu.

Die Grundidee ist es, eine Quadratische Kongruenz $a^2 \equiv b^2 (modN)$ aufzutreiben. Denn wenn wir das umformen, erhalten wir $a^2 - b^2 \equiv 0(modN)$, und das ist gleichbedeutend damit, dass $(a + b) * (a - b)$ ein Vielfaches von N sein muss.

Allerdings begibt man sich nicht direkt auf die Suche nach einer solchen Kongruenz, denn das würde unnötig lange dauern. Vielmehr konstruiert man sie aus sogenannten Relationen, die man stattdessen ermittelt.

Diese Relationen bleiben im Quadratischen Sieb hängen, mit denen man gleichsam nach ihnen fischt. In dem Quadratischen Sieb werden alle Zahlen der Form $(R + \Delta)^2 - N$ (wobei R der abgerundete Wert von $\sqrt{N}$ ist, damit für die relativ kleinen Werte von Δ der Term möglichst klein bleibt) innerhalb eines festgelegten Suchraums daraufhin geprüft, ob sie ausschließlich in kleine Faktoren zerlegbar sind. Aus diesen kleinen Faktoren ist nämlich unser Sieb gestrickt!

Wir müssen daher zunächst eine Faktorbasis festlegen, die diejenigen Primzahlen enthält, mit denen wir den Siebvorgang durchführen. Der Siebvorgang ergibt dann sich allerdings nicht durch Probedivision, sondern vielmehr dadurch, dass man wie beim Sieb des Eratosthenes die Vielfachen der betrachteten Primzahlen durchstreicht.

Beispiel: *Wir haben ermittelt, dass $(R+c)^2 - N$ durch p teilbar ist. Dann ist auch $(R+c+p)^2 - N$ durch p teilbar: $(R+c+p)^2 - N = ((R+c)+p)^2 - N = (R+c)^2 + 2*(R+c)*p + p^2 - N = (R+c)^2 - N + 2*(R+c)*p + p^2$. Da nun aber sowohl $(R+c)^2 - N$ (nach Annahme) also auch $2*(R+c)*p$ und p^2 durch p teilbar sind, ist auch deren Summe durch p teilbar. Wir können also beginnend mit $\Delta = c$ sukzessive alle Terme von $(R+\Delta)^2 - N$ ermitteln, die durch p teilbar sind, in dem wir schrittweise Δ um p erhöhen.*

Die Anzahl der Durchstreichungen an jeder Stelle kann man nun auf geschickte Weise zählen. Sind genügend Striche vorhanden, so ist die Zahl hochgradig in solche Primfaktoren zerlegbar, mit denen gesiebt wurde. Nur diese Zahlen faktorisiert man anschließend nochmals mittels Probedivision (da man sich die Teiler beim Durchstreichen nicht gemerkt hat).

Nun notiert man jede so erhaltene Relation in einer Tabelle. Auf die linke Seite der Tabelle wandern alle Teiler, die in der Faktorisierung aufgetreten sind. Auf der rechten Seite der Tabelle merkt man sich den Wert $R+\Delta$, denn dessen Quadrat ist kongruent zu dem Wert von $(R+\Delta)^2 - N$, an dem diese Faktorisierung gefunden wurde.

Man kann damit ein lineares Gleichungssystem aufstellen, welches diejenigen Relationen als Lösung hat, die miteinander multipliziert eine quadratische Kongruenz modulo N ergeben. Für eine Faktorbasis mit 1000 Primzahlen werden hierzu auch etwa 1000 Relationen benötigt; die Größe der "idealen" Faktorbasis und die darin enthaltenen Primzahlen hängen dabei stark von der zu faktorisierenden Zahl ab. Das mit den gefundenen Relationen erstellte Gleichungssystem gilt es dann zu lösen. Wenn das Gleichungssystem nicht so viele Unbekannte hätte und man nicht vorher so viele Relationen sieben müsste, wäre das alles eigentlich recht einfach...

einfaches Beispiel: *Seien (unter anderem) die Relationen $p_1 * p_2 = (R+\Delta_1)^2 - N$, $p_1 * p_3 = (R+\Delta_2)^2 - N$ sowie $p_2 * p_3 = (R+\Delta_3)^2 - N$ ermittelt worden. Dann können wir diese Relationen miteinander multiplizieren: $(p_1 * p_2) * (p_1 * p_3) * (p_2 * p_3) \equiv (R+\Delta_1)^2 * (R+\Delta_2)^2 * (R+\Delta_3)^2$ modulo N. Dies ist aber gleichbedeutend mit $(p_1 * p_2 * p_3)^2 \equiv ((R+\Delta_1) * (R+\Delta_2) * (R+\Delta_3))^2$ modulo N.*

Was bringt uns nun diese errechnete quadratische Kongruenz? – Nun, zunächst einmal besteht sie aus dem Produkt derjenigen Relationen, die auf der linken Seite ausmultipliziert nur gerade-oft auftretende Faktoren besitzen. Auf der linken Seite steht also ein Quadrat, nennen wir es mal a^2. Niemand hindert uns daran, nun auch dessen Wurzel a zu ermitteln. – Und auf der rechten Seite steht ein Produkt von Zahlen, die wir zum Aussieben quadriert haben. Nennen wir dieses Produkt b.

Einmal kurz scharf nachgedacht: Offenbar gilt $a^2 \equiv b^2 (modN)$! Außerdem kennen wir auch die Wurzeln dieser Quadrate, nämlich a und b. Nun gibt es zwei Möglichkeiten:

- Wenn wir Pech haben, dann gilt entweder $a \equiv b$ oder $a \equiv N - b$. Dann sind (a+b) oder (a-b) selbst Vielfache von N und tragen daher nicht zu einer Faktorisierung bei. In diesem Falle müssen wir halt noch etwas weitersieben und weitere Relationen auftreiben.

- Wenn wir Glück haben, dann sind (a+b) und (a-b) selbst keine Vielfachen von N, sondern lediglich Vielfache von den Teilern dieser Zahl! Dann ist aber beispielsweise $ggT(a+b, N)$ ein Teiler von N. Ein glücklicher Zufall? – Nein, vielmehr das Ziel einer raffinierten Konstruktion!

Das quadratische Sieb gehört – allerdings mit leichten Abwandlungen – zum schnellsten, was die die Mathematiker an Faktorisierungsmethoden bisher ausknobeln konnten. Obwohl sich damit derzeit Zahlen bis um 130 Dezimalstellen faktorisieren lassen, enthält auch dieses Verfahren noch zu viele Stellen, an denen man raten und suchen muss (Wo sind die richtigen Relationen zu finden? Eigentlich benötige ich ja nur die Relationen, die ich nachher zusammenmultipliziere! Deren Anzahl sollte zudem möglichst gering sein. Doch woher weiß ich, ob gerade diese Relationen zusammenmultipliziert einen nichttrivialen Nullteiler ergeben?). Und dieses Raten und Suchen (bzw. der Verzicht auf das Raten zu Lasten des Suchens) dauern einfach zu lange, um ein wirklich effizientes Faktorisierungsverfahren zu ergeben...

Literatur

[1] Prime Numbers and Computer Methods for Factorization, Hans Riesel, 2.Auflage, Basel, 1994

[2] Faktorisierung großer Zahlen, Johannes Buchmann, in: Spektrum der Wissenschaft, September 1996

[3] Finding Primes and Proving Primality, Chris K. Caldwell, http://www.utm.edu/research/primes

[4] The Multiple Polynomial Quadratic Sieve, Robert D. Silverman, in: Mathematics Of Computation, Vol. 48, 1987, p.329-339

[5] Public-Key Cryptography, Arto Salomaa, Berlin, 1990

[6] Blue Ribbon Campaign, http://www.eff.org/blueribbon.html

[7] Krypto-Debatte: Bonn diskutiert hinter verschlossenen Türen, Peter Panther, in: c't magazin für computertechnik, 14/1997, S. 22 ff.

[8] Quantencomputer: ungeahnte Rechenleistung, Jürgen Rink, in: c't magazin für computertechnik, 3/1997, S. 110 ff.

[9] Der Mann, der seine Frau mit einem Hut verwechselte, Oliver Sacks, Hamburg, 1987, Kap. 23: Die Zwillinge

[10] Grünes Licht für Kryptographie, Christiane Schulzki-Haddouti, in: c't magazin für computertechnik, 13/1999, S. 46