ist. Für $p > 2$ hat $f(x) = 0$ in $\mathbb{F}_p$ dann die beiden Nullstellen

$$x = \left[\sqrt{N}\right] \pm w\ ;$$

andernfalls gibt es keine Lösung.

Insbesondere kann also $f(x)$ nur dann durch p teilbar sein, wenn N modulo p ein Quadrat ist; dies ist für etwa die Hälfte aller Primzahlen der Fall. Offensichtlich sind alle anderen Primzahlen nutzlos, die Faktorbasis sollte also nur Primzahlen enthalten, für die N modulo p ein Quadrat ist.

Für solche p kann man dann die beiden Lösungen der Gleichung $f(x) = 0$ in $\mathbb{F}_p$ berechnen: Im Vergleich zum sonstigen Aufwand der Faktorisierung ist die Nullstellensuche durch Probieren durchaus vertretbar, allerdings kann man Quadratwurzeln modulo p mit etwas besseren Zahlentheoriekenntnissen auch sehr viel schneller berechnen.

Das eigentliche Sieben zum Auffinden der komplett über der Faktorbasis zerlegbaren Funktionswerte $f(x)$ geht nun folgendermaßen vor sich: Man legt ein Siebintervall $x = 0, 1, \ldots, M$ fest und speichert in einem Feld der Länge $M + 1$ für jedes x eine ganzzahlige Approximation von $\log_2 f(x)$.

Für jede Primzahl p aus der Faktorbasis berechnet man dann die beiden Nullstellen $x_{1/2}$ von f modulo p im Intervall von 0 bis $p - 1$ und subtrahiert von jedem Feldelement mit Index der Form $x_1 + kp$ oder $x_2 + kp$ eine ganzzahlige Approximation von $\log_2 p$.

Falls $f(x)$ über der Faktorbasis komplett faktorisierbar ist, sollte dann am Ende der entsprechende Feldeintrag bis auf Rundungsfehler gleich null sein; um keine Fehler zu machen, untersucht man daher für alle Feldelemente, die unterhalb einer gewissen Grenze liegen, durch Abdividieren, ob sie wirklich komplett faktorisieren, und man bestimmt auf diese Weise auch *wie* sie faktorisieren. Damit läßt sich dann das oben erwähnte Gleichungssystem über $\mathbb{F}_2$ aufstellen und, falls genügend viele Relationen gefunden sind, nichttrivial so lösen, daß eine der daraus resultierenden Gleichungen $x^2 \equiv y^2 \bmod p$ zu einer nichttrivialen Faktorisierung von N führt.

c) Varianten des quadratischen Siebs

Der Rechenaufwand beim quadratischen Sieb entfällt größtenteils auf das Sieben: Nur ein verschwindend kleiner Teil aller Zahlen zerfällt über der gewählten Faktorbasis, und je größer x wird, desto weniger dicht liegen diese Zahlen. Bei einer Zahl um 10^{100} und einer Faktorbasis aus 10 000 Primzahlen etwa kann man für $x \leq 10^{10}$ etwa fünf vollständig faktorisierbare Werte von $f(x)$ erwarten, im neunmal so großen Intervall $[10^{10}, 10^{11}]$ nur noch etwa 23 und so weiter.

Zumindest qualitativ ist dies klar, denn je größer die Zahlen werden, desto größer wird die Wahrscheinlichkeit großer Primfaktoren. Eine Abschätzung mit (teils nur heuristischen) Formeln über die Verteilung von Primfaktoren zeigt, daß man in einem solchen Fall ein Intervall sieben muß, das bis über 10^{14} hinausreicht. Verbesserungen des quadratischen Siebs konzentrieren sich daher darauf, die Siebphase zu optimieren um so in kürzerer Zeit mehr Relationen zu finden.

1.) Die Multipolynomialversion: Die Multipolynomialversion des quadratischen Siebs optimiert dieses an zwei Stellen: Einmal betrachtet sie außer dem Polynom $\left(x - [\sqrt{N}]\right)^2 - N$ noch weitere Polynome, um Relationen zu bekommen, so daß man jedes dieser Polynome nur über ein kürzeres Intervall sieben muß; zum andern betrachtet sie auch negative Werte von x, so daß – bei geschickt gewählten Polynomen f – der Betrag von $f(x)$ für ein längeres Intervall klein bleibt.

Dabei muß natürlich berücksichtigt werden, daß nun auch negative Werte $f(x)$ auftreten können; dies geschieht einfach dadurch, daß man die Zahl -1 wie eine Primzahl behandelt, die genau dann in der Faktorisierung von $f(x)$ auftritt, wenn $f(x)$ negativ ist.

Als Polynome betrachtet man quadratische Polynome der Form

$$f(x) = ax^2 + 2bx + c \quad \text{mit} \quad 0 \leq b < a \quad \text{und} \quad b^2 - ac = N\ .$$

Für diese ist

$$a f(x) = (ax + b)^2 - b^2 + ac = (ax + b)^2 - N\ ,$$

so daß $a f(x)$ zwar kongruent $(ax + b)^2$ ist, aber nicht gleich. Auch diese Polynome liefern also die Art von Relationen, die wir brauchen, und sie können genauso gesiebt werden wie das spezielle Polynom aus dem letzten Abschnitt.

Ein gewisser Nachteil dabei ist, daß man vor dem Sieben für jedes neue Polynom f und jede Primzahl p neu die Nullstellen von f modulo p ausrechnen muß. Bei geschickter Vorgehensweise fällt diese Zeit jedoch kaum ins Gewicht:

Zur Konstruktion von Polynomen f wählt man zunächst eine Zahl a so, daß das Polynom in einem Intervall $[-M, M]$ möglichst beschränkt bleibt. Falls a, b, c deutlich kleiner sind als M, liegt der Maximalwert von $f(x)$ an den Intervallenden, liegt das Maximum bei

$$f(-M) \approx \frac{1}{a}\left(a^2M^2 - N\right);$$

das Minimum wird bei $x = -b/a$ angenommen und ist

$$f\left(-\frac{b}{a}\right) = \frac{b^2}{a} - \frac{2b^2}{a} + c = \frac{-b^2 + ac}{a} = -\frac{N}{a}\,.$$

Für $a \approx \sqrt{2N}/M$ haben beide Zahlen ungefähr denselben Betrag, aber entgegengesetzte Vorzeichen; also wählen wir a in dieser Größenordnung.

Da $b^2 - 4ac = N$ werden muß, kommen für b nur Werte in Frage, für die $b^2 \equiv N \mod a$ ist, uns sobald ein solches b gewählt ist, liegt auch c eindeutig fest.

Mit Hilfe der Theorie quadratischer Reste lassen sich die Nullstellen modulo p aller möglicher Polynome zu einem gegebenen Wert von a simultan berechnen, wobei der Aufwand pro Polynom ziemlich gering ist. Man verwendet daher bei der Multipolynomialversion des quadratischen Siebs sehr viele Polynome und relativ kleine Siebintervalle $[-M, M]$.

Die Anzahl der Polynome, die Polynome selbst und die Zahl M sollten idealerweise so gewählt werden, daß die Rechenzeit minimal wird. Deren Abschätzung hängt ab von einer ganzen Reihe von Größen, die teils nur mit großem Aufwand berechnet werden können, teils nur modulo unbewiesener Vermutungen wie etwa der RIEMANN-Vermutung bekannt sind und für die teils sogar nur rein heuristische Formeln existieren. Ich

möchte auf die damit verbundenen Probleme nicht eingehen, sondern nur das Ergebnis angeben, wonach bei optimaler Wahl aller Größen der Rechenaufwand zum Faktorisieren einer Zahl N mit der Multipolynomialvariante des quadratischen Siebs proportional ist zu $e^{\sqrt{\ln N \ln\ln N}}$.

2.) Das Zahlkörpersieb: Die derzeit schnellste Verbesserung des quadratischen Siebs ist das *Zahlkörpersieb,* das nicht mehr mit quadratischen Polynomen arbeitet, sondern mit Polynomen beliebigen Grades.

Der Grad d dieser Polynome wird in Abhängigkeit der zu faktorisierenden Zahl N festgelegt, sodann wählt man führende Koeffizienten a_d und eine natürliche Zahl

$$m \approx \sqrt[d]{\frac{N}{a_d}}\,.$$

Alle Polynome sind homogen, haben also die Form

$$F(x, y) = a_d x^d + a_{d-1}x^{d-1}y + \cdots + a_1 xy^{d-1} + a_0 y^d\,,$$

wobei die a_i mit $i < d$ höchstens Betrag $m/2$ haben.

Hinzu kommt das homogene lineare Polynom

$$G(x, y) = x - my\,;$$

das Sieb sucht nach Zahlenpaaren (x, y), für die sowohl $F(x, y)$ als auch $G(x, y)$ über der Faktorbasis zerfallen. Da die Polynome von zwei Variablen abhängen, muß man entweder jeweils eine Variable festhalten und über die andere sieben, oder aber ein zweidimensionales Siebverfahren anwenden; üblich ist eine Kombination beider Methoden.

Gesucht sind Paare (x, y) teilerfremder ganzer Zahlen, für die

$$F(x, y) \qquad \text{und} \qquad G(x, y)$$

beide über der gewählten Faktorbasis zerfallen, und das Ziel ist, wie beim quadratischen Sieb, eine Relation der Form

$$\prod_{(x,y)\in\mathcal{M}} F(x, y) \equiv \prod_{(x,y)\in\mathcal{M}} G(x, y) \mod N\,,$$

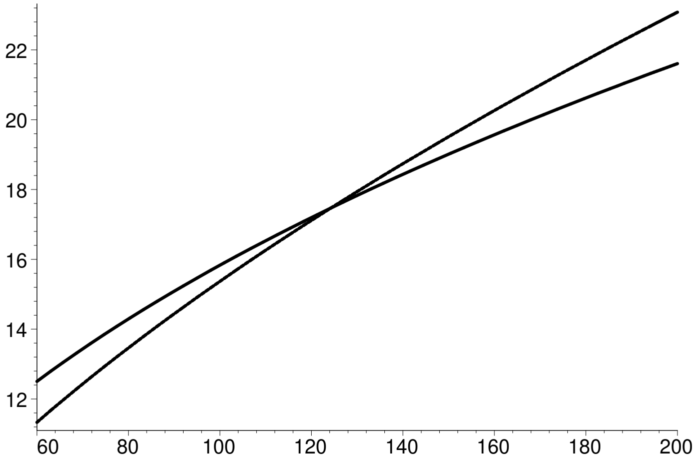
in der rechts und links Quadrate stehen.

Die besten derzeit bekannten Strategien zur Polynomauswahl *usw.* führen auf eine Laufzeitabschätzung proportional

$$e^{c(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}} \quad c = \sqrt[3]{\frac{64}{9}} \approx 1{,}923$$

für die Faktorisierung einer Zahl N nach dieser Methode.

Für hinreichend große N ist diese Methode offensichtlich schneller als das quadratische Sieb, bei dem $\ln N$ als Quadratwurzel im Exponenten steht; in der Abbildung, wo der Aufwand für die Faktorisierung einer Zahl 10^x aufgetragen ist, sieht man, daß das Zahlkörpersieb (durgezogene Linie) ab etwa 125-stelligen Zahlen dem quadratischen Sieb (gestrichelte Linie) überlegen ist.



Zeitaufwand für quadratisches Sieb und Zahlkörpersieb

d) Faktorisierungsrekorde

Neue Faktorisierungsverfahren werden meist vorgestellt mit einem Faktorisierungsbeispiel, das den bisherigen Verfahren getrotzt hat. Berühmt sind dabei die sogenannten *ten most wanted factorisations* des *Cunningham-Projekts,* wo es vor allem um Zahlen der Form $b^n \pm a$ für

kleine Werte von a und b. Mit diesen Problemen befaßten sich die algorithmischen Zahlentheoretiker schon lange vor der praktischen Bedeutung von Faktorisierungen im Zusammenhang mit dem RSA-Verfahren.

Im Zusammenhang mit der Kryptographie interessanter ist ein Liste von *challenges,* die *RSA Computer Security Incorperated* regelmäßig zusammenstellt, denn hier geht es um Zahlen, die sorgfältig unter kryptographischen Gesichtspunkten ausgewählt wurden.

Die größte bislang faktorisierte Zahl aus dieser Liste ist RSA-155, eine, wie der Name sagt, 155-stellige natürliche Zahl. Diese Ziffernlänge ist vor allem deshalb interessant, weil sie 512 Binärziffern entspricht, also jener Schlüssellänge, die Web-Browser typischerweise für die sogenannten „sicheren“ Verbindungen wählen und die auch im SSL-Protokoll *(Secure Socket Layer)* verwendet wird.

RSA-155 wurde 1999 mit dem Zahlkörpersieb faktorisiert; beteiligt waren 16 Wissenschaftler von 14 Institutionen aus drei Kontinenten, 160 *workstations,* acht R10000-Prozessoren, 120 Pentium II PCs, vier Digital/Compaq boxes, die in 3,7 Monaten mit einer Rechenleistung von 35,7 CPU-Jahren das Sieben durchführten. Sodann brauchte ein SGI Origin 2000 Computer einen Monat, um eine Matrix mit 6 699 191 Zeilen und 6 711 336 Spalten aufzustellen, aus der schließlich eine CRAY C916 in 214 CPU-Stunden mit zwei Gigabyte RAM 64 Relationen fand. Für vier dieser Relationen berechnete die SGI Origin 2000 die Quadratwurzeln links und rechts mit einem Rechenaufwand von zwischen etwa vierzig und etwa sechzig CPU-Stunden pro Relation. Die anschließenden Faktorisierungsversuch evia EUKLIDischen Algorithmus benötigten natürlich kaum eine Sekunde und lieferten die Zerlegung in zwei 78-stellige Primzahlen.

Wie man sieht, ist diese Faktorisierung noch deutlich jenseits der Möglichkeiten eines Gelegenheitshackers; Zugriff auf ein Netz von etwa dreihundert *workstations* und PCs ist zwar kein großes Problem, aber 214 CPU-Stunden auf einer CRAY sind außer für Regierungsstellen und Forschungszentren höchstens noch für Großunternehmen realistisch, wobei selbst bei denen einiges an kreativer Buchführung notwendig wäre, falls der Faktorisierungsversuch geheimbleiben soll.

Dieser aus Sicht eines Anwenders von 512-Bit RSA-Verschlüsselung einzig positive Aspekt der Faktorisierung von RSA-155 galt aber leider nur wenige Monate; dann wurde eine weitere Faktorisierung einer 512-Bit-Zahl bekannt, die unter völlig anderen Rahmenbedingungen zustandekam.

Der durch sein populärwissenschaftliches Buch zum Beweis der FERMAT-Vermutung bekannte Autor SIMON SINGH veröffentlichte 1999 unter dem Titel *The Code Book* ein Buch über Kryptographie und verband dies mit einem zehnstufigen Preisrätsel, in dem Kryptogramme verschiedener Schwierigkeitsstufe geknackt werden mußten. Die zehnte Stufe enthielt unter anderem eine RSA-Verschlüsselung mit 512 Bit.

Die Gruppe schwedischer Studenten, die das Gesamträtsel im Oktober 2000 als erste löste, bestand nicht, wie die RSA-155-Gruppe, aus führenden Experten für algorithmische Zahlentheorie; sie enthielt kein einziges Mitglied, das die Mathematik des Zahlkörpersiebs verstanden hätte. Da der Programmcode der RSA-155-Faktorisierung allerdings frei erhältlich ist, konnten sie damit arbeiten und ihn an ihre Bedürfnisse anpassen.

Wegen der fehlenden Erfahrung der Anwender war die Polynomauswahl hier deutlich schlechter als bei RSA-155, was zu einer fast doppelt so langen Siebzeit führte.

Das nächste Problem war die Lösung des linearen Gleichungssystems, für die bis dahin nur CRAY-Code zur Verfügung stand. Die Gruppe entwickelte daher ein neues Programm, das auch auf kleineren Rechnern läuft, und löste das Gleichungssystem in dreizehn Tagen auf einem Compaq quad processor ES40 System mit vier Alpha-Prozessoren mit 667 MHz und 8 GB Hauptspeicher, der von der Herstellerfirma zu diesem Zweck zur Verfügung gestellt wurde. Damit war gezeigt, daß auch relative Laien mit relativ beschränkten Mitteln eine 512-Bit-Faktorisierung durchführen können; einschließlich aller theoretischer Vorbereitungen mußten sie dazu ungefähr elf Monate aufwenden.

Es steht zu erwarten, daß entsprechende Faktorisierungen in wenigen Jahren routinemäßig durchgeführt werden können; die Tage der 512-Bit-Moduln sollten also hoffentlich bald gezählt sein.

e) Faktorisierung durch Spezialhardware

Faktorisierungen mit dem quadratischen oder Zahlkörpersieb benötigen zwar zumindest für einige Schritte wie die Lösung des linearen Gleichungssystems leistungsfähige Rechner mit viel Speicher, aber im wesentlichen handelt es sich dabei doch um Standardcomputer. Da die Grundoperation des Siebens so einfach ist, dürfte es sich wohl auch kaum lohnen, dafür spezielle Chips zu entwerfen und einzusetzen.

Es gibt allerdings zumindest einen Vorschlag, wie man das Sieben mit einem optoelektronischen Gerät etwa um den Faktor Tausend beschleunigen kann: ADI SHAMIR, einer der Erfinder des RSA-Verfahrens, entwarf 1999 TWINKLE, *The Weitzman INstitut Key Locating Engine.*

Das Gerät sitzt in einer schwarzen Röhre mit etwa 15 cm Durchmesser und 25 cm Länge, deren wesentlicher Chip im Innern etwa eine Million LEDs enthält. Jede dieser LEDs steht für eine Primzahl p aus der Faktorbasis und hat eine Leuchtkraft proportional $\log_2 p$, was etwa über ihre Größe oder (einfacher) mittels einer Abdeckfolie mit kontinuierlichem Grauschleier realisiert werden kann.

Hierin liegt der wesentliche Unterschied zu Software-Implementierungen der Siebe: Dort werden die Primzahlen nacheinander behandelt, während den x-Werten Speicherzellen entsprechen. Bei TWINKLE werden die x-Werte auf die Zeitachse abgebildet; da die x-Werte, für die $f(x)$ durch p teilbar ist, von der Form $x_{1/2} + kp$ sind, muß also jede LED periodisch aufleuchten, was nicht schwer zu realisieren ist. Die Taktrate, mit der das Gerät arbeitet, soll bei 10 GHz liegen, ein Wert, der in optischen Hochgeschwindigkeitsnetzen heute durchaus normal ist.

In jedem Takt mißt ein den LEDs gegenüberliegender Sensor die Gesamtlichtstärke. Da ein Takt nur eine Länge von 10^{-10} Sekunden hat, läßt sich die Ausbreitungsgeschwindigkeit des Lichts hier nicht vernachlässigen; bei einer Geschwindikeit von etwa 300 000 km/sec legt es pro Takt etwa drei Zentimeter zurück. Die Laufwegunterschiede der Lichtstahlen zwischen den verschiedenen Dioden und der Meßzelle müssen also deutlich kleiner als drei Zentimeter sein. Dies wird dadurch erreicht, daß alle LEDs auf einem einzigen Wafer sitzen.

Die Meßgenauigkeit der Zelle muß nicht übermäßig hoch sein: Beim klassischen Sieb arbeitet man schließlich auch nur mit ganzzahligen Approximationen der $\log_2 p$. Eine Zahl x ist uninteressant, wenn zum entsprechenden Zeitpunkt eine Lichtstärke gemessen wird, die kleiner ist als $\log_2 f(x)$ minus einem Sicherheitsabstand; ansonsten wird sie an konventionelle Elektronik weitergereicht und dort bearbeitet. Wie wir oben gesehen haben, ist dies ein sehr seltenes Ereignis, das im Schnitt höchstens einmal pro einer Milliarde x-Werte eintritt, also etwa zehnmal pro Sekunde. Mit diesen Datenraten können auch einfache Computer leicht fertigwerden.

Das Hauptproblem ist der Bau das Chips mit den Dioden und deren Steuerelektronik; SHAMIR meint, daß dies mit der heute existierende GaAs-Technologie gerade möglich sein sollte, und möglicherweise stehen inzwischen schon solche oder ähnliche Maschinen in Labors von NSA und ähnlichen Organisationen. In der offenen Literatur ist nicht über die Existenz solcher Maschinen bekannt und auch nichts über Pläne welche zu bauen. Der Entwicklungsaufwand dürfte sicherlich in die Hundertausende oder gar Millionen gehen, aber SHAMIR schätzt, daß das Gerät dann mit Stückkosten von etwa 5 000 $ hergestellt werden kann.

2000 stellte SHAMIR zusammen mit A. LENSTRA, einem der Erfinder des Zahlkörpersiebs, eine verbesserte Version vor; die beiden Autoren schätzen, daß diese Version zusammen mit 15 PCs eine 512-Bit-Zahl in einem halben Jahr faktorisieren kann. Wegen der sehr guten Parallelisierbarkeit des Zahlkörpersiebs könnte man mit mehr TWINKLEs und PCs natürlich auf deutlich kürzere Zeiten kommen.

Für 768-Bit-Faktorisierungen schätzen sie den Aufwand auf fünf Tausend TWINKLEs, unterstützt von achtzig Tausend PCs, bei einem Zeitbedarf von insgesamt neun Monaten.

f) Folgerungen für die Wahl der Schlüssellänge

Eigentlich braucht man sich in Deutschland keine großen Gedanken über die Wahl einer Schlüssellänge zu machen: Ein treu sorgender Staat, in Gestalt der Regulierungsbehörde für Telekommunikation und Post,

veröffentlich jedes Jahr ein Dokument mit dem Titel *Geeignete Kryptoalgorithmen gemäß §17 (2) SigV;* das neueste erschien am 7. Dezember 2000 im Bundesanzeiger Nr. 230 auf Seite 22.946.

SigV steht für die aufgrund des Signaturgesetztes SigG erlassene Signaturverordnung; beide gemeinsam legen fest, daß elektronische Unterschriften in Deutschland grundsätzlich zusätzlich und rechtsgültig sind, sofern gewisse Bedingungen erfüllt sind. Zu diesen Bedingungen gehört unter anderem, daß das Verfahren und die Schlüssellänge gemeinsam einen „geeigneten Kryptoalgorihmus“ im Sinne der jeweils gültigen Veröffentlichung der Regulierungsbehörde ist.

Da Rechner immer schneller und leistungsfähiger werden und auch auf der theoretisch-algorithmischen Seite fast jedes Jahr kleinere oder größere Fortschritte zu verzeichnen sind, gelten die jeweiligen Empfehlungen nur für etwa sechs Jahre; die derzeit neueste Version etwa betrachtet nur den Zeitraum bis Ende 2005 und betrachtet für RSA folgende Schlüssellängen als sicher:

bis Ende 2000:	768 Bit
bis Mitte 2005:	1024 Bit
bis Ende 2005:	2048 Bit

Außerdem sollen die beiden Primfaktoren p, q zufällig und unabhängig voneinander erzeugt werden und aus einem Bereich stammen, in dem

$$\varepsilon_1 < |\log_2 p - \log_2 q| < \varepsilon_2$$

gilt. Als *Anhaltspunkte* werden dabei die Werte

$$\varepsilon_1 = 0{,}5 \qquad \text{und} \qquad \varepsilon_2 = 30$$

vorgeschlagen; ist p die kleinere der beiden Primzahlen, soll also

$$\sqrt{2}p < q < 2^{30}p \approx 10^9 p$$

sein.

Falls man rechtsgültige elektronische Unterschriften braucht, muß man also darauf achten, daß während der Vertragslaufzeit diese Mindestanforderungen erfüllt sind.

Für die Übermittlung verschlüsselter privater oder auch geschäftlicher Informationen darf natürlich jeder seine eigenen Standards festlegen; wir wollen sehen, was sich auf Grund der vorherigen Abschnitte für die Sicherheit in Abhängigkeit von der Schlüssellänge folgern läßt.

Die Faktorisierung von RSA-155 zeigt, daß ein öffentlicher Schlüssel N mit 512 Bit faktorisiert werden kann, wenn auch derzeit noch mit beträchtlichem Aufwand sowohl an Rechenzeit als auch an Kalenderzeit. Ein solcher Schlüssel bietet also keine Sicherheit, gegen einen Opponenten, der ein größeres Netzwerk an Rechnern einsetzen kann und für den die Information auch noch in etwa sechs Monaten diesen Einsatz wert ist. (In den nächsten Jahren wird natürlich sowohl die Gesamtrechenzeit als auch die verflossene absolute Zeit für eine solche Faktorisierung recht schnell sinken.) Zur Übertragung einer Kreditkartennummer mit einem Kreditlimit von wenigen Tausend Mark oder eines Paßworts, das man (hoffentlich) ohnehin in wenigen Monaten ändert, sollten 512 Bit allerdings zumindest gegen Gelegenheitshacker im Augenblick noch genügend Schutz bieten.

Jedes weitere Bit Schlüssellänge erhöht die Sicherheit, macht aber wegen der magischen Grenze von 512 Bit bei vieler Software einen Upgrade erforderlich, der in vielen Fällen nicht erhältlich sein dürfte: Für amerikanische Hersteller etwa gelten 512 Bit immer noch als Grenze, jenseits derer für RSA zumindest keine Exportgenehmigungen im vereinfachten Verfahren mehr gegeben werden.

Niemand kann vorhersagen, welche neuen Faktorisierungsverfahren in den nächsten Jahren möglicherweise entdeckt werden oder welche völlig neuen Rechnerarchitekturen möglichweise für einen Durchbruch bei der Faktorisierung sorgen werden: Falls beispielsweise die „Quantencomputer", mit denen wir uns am Ende der Vorlesung beschäftigen werden, je Realität werden mit Quantenregistern, deren Länge in der Größenordnung der Bitzahl der RSA-Moduln liegt, wird die Faktorisierung der Moduln nicht mehr schwieriger sein als die Verschlüsselung eines kurzen Texts.

Bislang ist es allerdings noch ein ungelöstes Problem, die Zahl 15 mit einem Quantencomputer zu faktorisieren; daher wollen wir diese im

Augenblick ignorieren und einfach ausrechnen, wie stark der Aufwand für eine Faktorisierung mit dem Zahlkörpersieb als dem schnellsten derzeit bekannten Algorithmus ansteigt, wenn wir die Bitzahl auf mehr als 512 erhöhen.

Die folgende Tabelle zeigt dies anhand der oben angegebenen Formel für den Zeitbedarf des Zahlkörpersiebs:

Dezimalziffern	*Bits*	*Zeitfaktor*
155	512	1,00
160	529	1,89
165	545	3,53
170	562	6,52
175	579	11,93
180	595	21,56
185	612	38,63
190	628	68,51
195	645	120,29
200	662	209,75
210	695	621,91
220	728	1788,09
230	761	4996,63
240	794	13596,95
250	828	36094,23
260	861	93615,09
270	894	237562,59
280	927	590593,30
290	961	1440060,02
300	994	3447569,30
310	1027	8111574,17

Falls man abschätzen möchte, welchen Aufwand ein Gegner *heute* treiben muß, um einen RSA-Schlüssel einer gegebenen Länge zu faktorisieren, gibt diese Tabelle wohl einen recht guten Überblick; falls man allerdings abschätzen möchte, wie lange eine Schlüssellänge noch sicher ist, dürfte sie auf deutlich zu optimistische Schlußfolgerungen führen.

Wie in der Vergangenheit werden wohl auch in der Zukunft immer neue

Verfahren gefunden und bei den alten die Konstanten verbessert: So war beispielsweise die Faktorisierung von RSA-155 etwa viermal so schnell, wie anhand der (ebenfalls mit dem Zahlkörpersieb durchgeführten) Faktorisierung von RSA-150 zu erwarten gewesen wäre. Hinzu kommt, daß die Harware immer schneller wird und vielleicht auch neue Spezialhardware hinzukommt.

Die Autoren der RSA-155-Faktorisierung haben Rekordfaktorisierungen der letzten dreißig Jahre untersucht (erster Rekord war 1970 die 39-stellige FERMAT-Zahl $2^{2^7}+1$, die Maple heute auf einem Pentium II PC mit 266 MHz und 128 kB RAM in gut einer Viertelstunde faktorisiert) und kamen zu der heuristischen Formel, daß eine d-stellige allgemeine Dezimalzahl wohl erstmals im Jahr

$$13{,}24\sqrt[3]{d}+1928{,}6$$

faktorisiert werden dürfte. Für gängige Bitlängen von RSA-Moduln sind also folgende Jahre zu erwarten, in denen entsprechende Moduln erstmalig faktorisiert werden:

Bit:	768	1024	1280	1536	2048	2560	3072	4096
Jahr:	2010	2018	2025	2031	2041	2050	2057	2070

Im Jahr 2100 würden demnach Zahlen mit etwa 7200 Bit faktorisiert; für das Jahr 3000, bis zu dem die Formel ganz sicher nicht gilt, ergäben sich etwa 1,7 Millionen Bit.

Dies bietet zumindest einen gewissen Anhaltspunkt für Verschlüsselungen, die längerfristig sicher sein müssen – auch wenn niemand garantieren kann, daß die nächsten Jahren keinen so dramatischen Durchbruch bei den Faktorisierungsverfahren bringen, daß dadurch all diese Extrapolationen über den Haufen geworfen werden.

§5: Wie findet man große Primzahlen?

Nachdem wir gewisse Vorstellungen davon haben, wie man in Abhängigkeit von der gewünschten Sicherheit die Ziffernzahl von N festlegen sollte und damit auch die von p und q, müssen wir uns als nächstes überlegen, wie wir geeignete Primzahlen finden.

p und q sollten nicht zu speziell sein, da sonst ein Gegner einfach alle Primzahlen einer speziellen Form durchprobieren muß. Wir sollten also möglichst zufällige Primzahlen erzeugen, d.h. jede Primzahl einer bestimmten Größenordnung sollte die gleiche Chance haben, daß sie gewählt wird.

Dies läßt sich am sichersten dadurch erreichen, daß wir Zufallszahlen erzeugen und diese Testen, ob sie Primzahlen sind; wir brauchen daher einen Test, der uns sagt, ob eine gegebene Zahl prim ist.

a) Der Fermat-Test

Der kleine Satz von FERMAT gibt uns sofort eine Aussage darüber, wann eine Zahl p *nicht* prim ist:

Falls für eine natürliche Zahl $1 \le a \le p-1$ gilt $a^{p-1} \not\equiv 1 \bmod p$, kann p keine Primzahl sein.

Beispiel: Ist $F_{20} = 2^{2^{20}}+1$ eine Primzahl? Falls ja, ist nach dem kleinen Satz von FERMAT insbesondere

$$3^{F_{20}-1} \equiv 1 \bmod F_{20}\,.$$

Nachrechnen zeigt, daß

$$3^{(F_{20}-1)/2} \not\equiv \pm 1 \bmod F_{20}\,,$$

die Zahl ist also nicht prim. (Das *Nachrechnen* ist bei dieser 315653-stelligen Zahl natürlich keine Übungsaufgabe für Taschenrechner: 1988 brauchte eine Cray X-MP dazu 82 Stunden, eine Cray-2 immerhin noch zehn; siehe *Math. Comp.* **50** (1988), 261–263.)

Die Umkehrung der obigen Aussage gilt leider nicht: Ist beispielsweise

$$p = (6t+1)(12t+1)(18t+1)$$

Produkt dreier *Primzahlen* dieser speziellen Form, so ist nach FERMAT

$$a^{6t} \equiv 1 \bmod (6t+1), \qquad a^{12t} \equiv 1 \bmod (12t+1)$$

und

$$a^{18t} \equiv 1 \bmod (18t+1)\,,$$

also

$$a^{36t} = 1 \bmod p\,.$$

Wegen

$$p - 1 = 36t(36t^2 + 11t + 1)$$

ist dann auch

$$a^{p-1} = 1 \bmod p$$

für alle zu p teilerfremden ganzen Zahlen a.

Zahlen dieser Art gibt es tatsächlich, z.B.

$$1729 = 7 \times 13 \times 19, \quad 294409 = 37 \times 73 \times 109 \quad \textit{usw.};$$

es gibt auch entsprechende Zahlen, die nicht von dieser speziellen Form sind: Wir benötigen offenbar nur, daß für jeden Primteiler q von p auch $q - 1$ ein Teiler von $p - 1$ ist. Es ist nicht bekannt, ob es unendlich viele Zahlen mit dieser Eigenschaft gibt, es gilt aber als wahrscheinlich.

Trotzdem wird es für große Zahlen zunehmend unwahrscheinlich, daß eine Zahl p für auch nur ein a den obigen Test besteht, ohne Primzahl zu sein. Rechnungen von

SU HEE KIM, CARL POMERANCE: The probability that a Random Probable Prime is Composite, *Math. Comp.* **53** (1989), 721–741

geben folgende obere Schranke für die Fehlerwahrscheinlichkeit ε:

$p \approx 10^{60}$	10^{70}	10^{80}	10^{90}	10^{100}
$\varepsilon \leq 7{,}16 \cdot 10^{-2}$	$2{,}87 \cdot 10^{-3}$	$8{,}46 \cdot 10^{-5}$	$1{,}70 \cdot 10^{-6}$	$2{,}77 \cdot 10^{-8}$
$p \approx 10^{120}$	10^{140}	10^{160}	10^{180}	10^{200}
$\varepsilon \leq 5{,}28 \cdot 10^{-12}$	$1{,}08 \cdot 10^{-15}$	$1{,}81 \cdot 10^{-19}$	$2{,}76 \cdot 10^{-23}$	$3{,}85 \cdot 10^{-27}$

(Sie geben natürlich auch eine allgemeine Formel an, jedoch ist diese zu grausam zum Abtippen.)

Selbst wenn wir noch mit 512-Bit-Moduln arbeiten und somit knapp achtzigstellige Primzahlen brauchen, liegt also die Fehlerwahrscheinlichkeit bei nur etwa 10^{-5}; falls man sie erniedrigen möchte, testet man einfach mit mehreren zufällig gewählten Basen und hat dann etwa bei

drei verschiedenen Basen eine Irrtumswahrscheinlichkeit von höchsten etwa 10^{-15}, daß alle drei Tests das falsche Ergebnis liefern.

Bei den etwa 155-stelligen Zahlen, die wir für 1024-Bit-Moduln brauchen, hat schon ein einziger Test eine geringere Fehlerwahrscheinlichkeit als 10^{-15}, so daß es sich nur selten lohnt, einen größeren Aufwand zu treiben. Wir wollen uns daher nicht weiter mit Primzahltests beschäftigen.

Zur Beruhigung der Leser sei lediglich darauf hingewiesen, daß es mit realistischem Aufwand durchführbare Tests gibt, mit denen man exakt entscheiden kann, ob eine gegebene Zahl prim ist oder nicht; die Mathematik hinter diesen Tests ist allerdings etwas aufwendiger. Für theoretisch interessierte sei noch hinzugefügt, daß das Problem, Primzahlen zu erkennen, in der Komplexitätsklasse P liegt – obwohl alle *realistischen* Algorithmen höhere asymptotische Komplexität haben.

b) Wie dicht liegen die Primzahlen?

Bevor man daran geht, Zufallszahlen zu testen, sollte man zunächst wissen, wie wahrscheinlich es ist, daß eine zufällig gewählte Zahl prim ist. Dies ist ein sehr altes Problem, das immer noch nicht ganz vollständig gelöst ist; die bekannte Antwort ist aber für praktische Zwecke gut genug. Die Mathematik, die im Beweis der folgenden Aussagen steckt, ist allerdings leider jenseits des zeitlichen Rahmens dieser Vorlesung – obwohl einige der Beweise inzwischen vergleichsweise sehr elementar geworden sind. Wer einigermaßen mit Funktionentheorie vertraut ist, sollte in der Lage sein, die Darstellung in

HELMUT KOCH: Einführung in die klassische Mathematik I, *Akademie-Verlag* und (in Lizens) *Springer-Verlag,* 1986, §27

zu lesen; alle notwendigen Voraussetzungen aus Funktionentheorie, Zahlentheorie usw. sind im Buch selbst zu finden.

Wir bezeichnen die Anzahl der Primzahlen, die kleiner oder gleich einer Zahl x sind, mit $\pi(x)$. Demnach ist also $\pi(2) = 1$ und $\pi(3) = \pi(4) = 2$ und so weiter. Der englische Mathematiker SYLVESTER bewies 1892