Seriennummern werden von den Kunden zufällig erzeugt. Für jede solche Seriennummer m erzeugt der Kunde eine Zufallszahl r, schickt mr^e mod N an die Bank und erhält (nach Belastung seines Kontos) eine Unterschrift u für diese Nachricht zurück. Wie oben berechnet er daraus durch Multiplikation mit r^{-1} die Unterschrift $v = m^d$ mod N für die Seriennummer N, und mit diesem Block kann er bezahlen.

Der Zahlungsempfänger berechnet v^e mod N; falls dies die Form einer gültigen Seriennummer hat, kann er sicher sein, einen von der Bank unterschriebenen Geldschein vor sich zu haben. Er kann allerdings noch nicht sicher sein, daß dieser Geldschein nicht schon einmal ausgegeben wurde.

Deshalb muß er die Seriennummer an die Bank melden, die mit ihrer Datenbank bereits ausbezahlter Seriennummern vergleicht. Falls sie darin noch nicht vorkommt, wird sie eingetragen und der Händler bekommt sein Geld; andernfalls verweigert sie die Zahlung.

Bei 10^{75} möglichen Nummern liegt die Wahrscheinlichkeit dafür, daß zwei Kunden, die eine (wirklich) zufällige Zahl wählen, dieselbe Nummer erzeugen, bei etwa $10^{-37,5}$. Die Wahrscheinlichkeit, mit jeweils einem Spielschein fünf Wochen lang hintereinander sechs Richtige im Lotto zu haben, liegt dagegen bei $\binom{49}{6}^{-5} \approx 5 \cdot 10^{-35}$, also etwa um den Faktor sechzig höher. Zwei gleiche Seriennummern sind also praktisch auszuschließen, wenn auch theoretisch möglich.

Falls wirklich einmal zufälligerweise zwei gleiche Seriennummern erzeugt worden sein sollten, kann das System nur funktionieren, wenn der zweite Geldschein mit derselben Seriennummer nicht anerkannt wird, so daß der zweite Kunde sein Geld verliert. Dies muß als eine zusätzliche Gebühr gesehen werden, die mit an Sicherheit grenzender Wahrscheinlichkeit nie fällig wird, aber trotzdem nicht ausgeschlossen werden kann.

Da digitales Bargeld allerdings nur in kleinen Stückelungen sinnvoll ist (Geldscheinen im Millionenwert wären auf Grund ihrer Seltenheit nicht wirklich anonym und würden, wegen der damit verbundenen Möglichkeiten zur Geldwäsche, auch in keinem seriösen Wirtschaftssystem angeboten), wäre der theoretisch mögliche Verlust ohnehin nicht sehr groß.

e) Bankkarten mit Chip

In Deutschland und den meisten anderen Ländern hat eine Bankkarte einen Magnetstreifen, auf dem die wichtigsten Informationen wie Kontenname und -nummer, Bankleitzahl, Gültigkeitsdauer *usw.* gespeichert sind; dazu kommt, mit Triple-DES verschlüsselte Information, die unter anderem die Geheimzahl enthält, aber auch von den obengenannten Daten abhängt.

Der Schlüssel, mit dem Triple-DES hier arbeitet, muß natürlich streng geheimgehalten werden: Wer ihn kennt, kann problemlos die Geheimzahlen fremder Karten ermitteln und eigene Karten zu beliebigen Konten erzeugen.

Um eine Karte zu überprüfen, muß daher eine Verbindung zu einem Zentralrechner aufgebaut werden, an den sowohl der Inhalt des Magnetstreifens als auch die vom Kunden eingetippte Zahl übertragen werden; dieser wendet Triple-DES mit dem Systemschlüssel an und meldet dann, wie die Prüfung ausgefallen ist.

In Frankreich haben die entsprechenden Karten zusätzlich zum Magnetstreifen noch einen Chip, in dem ebenfalls die Kontendaten gespeichert sind sowie, in einem auslesesicheren Register, Informationen über die Geheimzahl. Dort wird die ins Lesegerät eingetippte Geheimzahl nicht an den Zentralrechner übertragen, sondern an den Chip, der sie überprüft und akzeptiert oder auch nicht.

Da frei programmierbare Chipkarten relativ billig sind, muß dafür Sorge getragen werden, daß ein solches System nicht durch einen *Yes-Chip* unterlaufen werden kann, der ebenfalls die Konteninformationen enthält, ansonsten aber ein Programm, das ihn *jede* Geheimzahl akzeptieren läßt. Das Terminal muß also, bevor es überhaupt eine Geheimzahl anfordert, zunächst einmal den Chip authentizieren, d.h. sich davon überzeugen, daß es sich um einen vom Bankenkonsortium ausgegebenen Chip handelt.

Aus diesem Grund sind die Kontendaten auf dem Chip mit dem privaten RSA-Schlüssel des Konsortiums unterschrieben. Die Terminals

kennen den öffentlichen Schlüssel dazu und können so die Unterschrift überprüfen.

Diese Einzelheiten und speziell deren technische Implementierung wurden vom Bankenkonsortium zunächst streng geheimgehalten. Trotzdem machte sich 1997 ein Ingenieur namens SERGE HUMPICH daran, den Chip genauer zu untersuchen. Er verschaffte sich dazu ein (im freien Verkauf erhältliches) Terminal und untersuchte sowohl die Kommunikation zwischen Chip und Terminal als auch die Vorgänge innerhalb des Terminals mit Hilfe eines Logikanalysators. Damit gelang es ihm nach und nach, die Funktionsweise des Terminals zu entschlüsseln und in ein äquivalentes PC-Programm zu übersetzen. Durch dessen Analyse konnte er die Authentizierungsprozedur und die Prüflogik entschlüsseln und insbesondere auch feststellen, daß hier mit RSA gearbeitet wurde.

Blieb noch das Problem, den Modul zu faktorisieren. Dazu besorgte er sich ein japanisches Programm aus dem Internet, das zwar eigentlich für kleinere Zahlen gedacht war, aber eine Anpassung der Wortlänge ist natürlich auch für jemanden, der den Algorithmus hinter dem Programm nicht versteht, kein Problem. Nach sechs Wochen Laufzeit hatte sein PC damit den Modul faktorisiert:

$$\begin{aligned} &213598703592091008239502270499962879705109534182\backslash \\ &641740644252416500858395774644508840500943086599 9 \\ = {} &111395432514882798792549017547702484407092284484 3 \\ \times {} &191748170252450443937578626823086218069693418929 3 \end{aligned}$$

Als er seine Ergebnisse über einen Anwalt dem Bankenkonsortium mitteilte, zeigte sich, was dieses sich unter Sicherheitsstandards vorstellt: Es erreichte, daß HUMPICH wegen des Eindringens in ein DV-System zu zehn Monaten Haft auf Bewährung sowie einem Franc Schadenersatz plus Zinsen verurteilt wurde; dazu kamen 12 000 F Geldstrafe.

Seit November 1999 haben neu ausgegebene Bankkarten nun noch ein zusätzliches Feld mit einer Unterschrift, die im Gegensatz zum obigen 320-Bit-Modul einen 768-Bit-Modul verwendet. Natürlich kann es nur von neueren Terminals überprüft werden, so daß viele Transaktionen weiterhin nur über den 320-Bit-Modul mit inzwischen wohlbekannter Faktorisierung „geschützt“ sind.

§4: Wie findet man große Primzahlen?

a) Wie groß sollten die Primzahlen sein?

Das Beispiel der französischen Bankkarten zeigt, daß RSA höchstens dann sicher ist, wenn die Primzahlen p und q hinreichend groß gewählt werden. Als erstes müssen wir uns daher die Frage stellen, wie groß eine „hinreichend große“ Zahl heute sein muß.

Ein treu sorgender Staat läßt seine Bürgern bei einer derart wichtigen Frage natürlich nicht allein: Zwar gibt es noch keine oberste Bundesbehörde für Primzahlen, aber das Bundesamt für Sicherheit in der Informationstechnik (BSI) und die Regulierungsbehörde für Telekommunikation und Post erarbeiten jedes Jahr ein gemeinsames Dokument mit dem Titel *Geeignete Kryptoalgorithmen gemäß §17 (2) SigV.*

SigV steht für die aufgrund des Signaturgesetztes SigG erlassene Signaturverordnung; beide gemeinsam legen fest, daß elektronische Unterschriften in Deutschland grundsätzlich zulässig 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. Für Dokumente, die länger gütlig sein sollen, sind elektronische Unterschriften also nicht vorgesehen.

Offiziell geht bei den Empfehlungen allgemein um geeignete Algorithmen für elektronische Unterschriften sowie deren Schlüssellängen, aber wie die Entwicklung der letzten Jahre zeigte, drehen sich die Diskussionen, die zu den jeweiligen Empfehlungen führen, tatsächlich fast ausschließlich um die jeweils notwendige Schlüssellänge für RSA.

Natürlich hat in einer Demokratie bei so einer wichtigen Frage auch die Bevölkerung ein Mitspracherecht; deshalb beginnt das BSI jeweils

zunächst einen Entwurf, zu dem es um Kommentare bittet; erst einige Monate später wird die endgültige Empfehlung verkündet und im Bundesanzeiger veröffentlicht.

Die interessierte Öffentlichkeit, von der die Kommentare zu den Entwürfen kommen, besteht naturgemäß in erster Linie aus Anbietern von Kryptographie-Software, und als erfahrene Experten für Datensicherheit wissen diese, daß ein Verfahren nur dann wirklich geeignet sein kann, wenn es die eigene Firma im Angebot hat. (Am geeignetsten sind natürlich die Verfahren, die keines der Konkurrenzunternehmen anbietet.)

Derzeit erhältliche Hardware-Implementierungen von RSA unterstützen typischerweise Schlüssellängen von bis zu 1024 Bit; größere Schlüssel sind vor allem in *public domain* Software wie PGP zu finden. Dies erklärt, warum es in den letzten Jahren recht lebhafte Diskussionen gab:

Bis Ende 2000 galten 768 Bit als ausreichende Größe für das Produkt N der beiden Primzahlen, jener Wert also, den die *neueren* französischen Bankkarten verwenden und den ebenfalls nur die neueren Terminals lesen können. Schon in den Richtlinien für 1998 wurden 768 Bit jedoch ausdrücklich nur übergangsweise zugelassen; längerfristig, d.h. bei Gültigkeit über 2000 hinaus, waren mindestens 1024 Bit vorgeschrieben.

Die Richtlinien für 2000 erlaubten die 768 Bit ebenfalls noch bis zum Ende des Jahres; für Dokumente mit einer längeren Gültigkeit verlangten sie bis Mitte 2005 eine Mindestgröße von 1024 Bit, danach bis Ende 2005 sogar 2048 Bit.

Anbieterproteste führten dazu, daß nach den Richtlinien von 2001 eine Schlüssellänge von 1024 dann doch noch bis Ende 2006 sicher war; die Schlüssellänge 2048 war nur noch „empfohlen“, also nicht mehr verbindlich.

Im April 2002 erschien der erste Entwurf für die 2002er Richtlinien; darin war für 2006 und 2007 nur eine Mindestlänge von 2048 Bit wirklich sicher. Einsprüche führten im September 2002 zu einem revidierten

Entwurf, wonach 2006 doch noch 1024 Bit reichen, 2007 aber mindestens 1536 notwendig werden. Die Mindestlänge von 2048 Bit wurde wieder zur „Empfehlung“ zurückgestuft.

Am 2. Januar 2003 erschienen endlich die offiziellen Richtlinien des Jahres 2002; veröffentlich wurden sie am 11. März 2003 im Bundesanzeiger Nr. 48, S. 4202–4203. Danach reichen 1024 Bit auch noch bis Ende 2007, erst 2008 werden 1280 Bit erforderlich. Die 2048 Bit blieben dringend empohlen.

Nach diesem großen Kraftakt erschienen 2003 keine neuen Richtlinien mehr; erst für 2004 gab es am 2. Januar 2004 neue Empfehlungen (Bundesanzeiger Nr. 30 vom 13. Februar 2004, S. 2537–2538). Für den Zeitraum bis Ende 2008 wurden die alten Empfehlungen beibehalten, bis Ende 2009 aber 1536 Bit gefordert. Die nächsten Richtlinien für 2005 werden wohl gegen Ende dieser Vorlesung erscheinen; die offiziellen Versionen sind bei `www.regtp.de` zu finden.

Die beiden Primfaktoren p, q sollen 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

$$\varepsilon_1 = 0{,}1 \quad \text{und} \quad \varepsilon_2 = 30$$

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

$$1{,}071773463\, p \approx \sqrt[10]{2}\, p < q < 2^{30} p \approx 10^9 p\,.$$

Wenn wir der „dringenden Empfehlung“ der Regulierungsbehörde folgen, müssen wir somit Primzahlen mit ungefähr 1024 Bit oder 309 Dezimalstellen finden.

Im Land der unbegrenzten Möglichkeiten ist das kein großes Problem: Dort kann man Primzahlen, wie alles andere auch, einfach kaufen. Unter Sicherheitsgesichtspunkten ist das allerdings nicht unbedingt die beste Strategie, denn erstens kann dann der Verkäufer der Primzahlen die gesamte verschlüsselte Korrespondenz des Käufers lesen und auch dessen elektronische Unterschrift nachmachen, und zweitens wird man nie

ganz sicher sein können, ob Billiganbieter wie *Thrifty Primes* oder *Primes for a Buck* nicht gelegentlich dieselbe Primzahl an mehrere Kunden verkaufen, so daß ein Kunde die öffentlichen Schlüssel seiner Konkurrenz durch die gerade gekauften Primzahlen teilen kann und damit teilweise Erfolg hat.

Sowohl aus Sicherheitsgründen als auch weil wir Mathematiker sind, sollten wir also unsere Primzahlen selbst erzeugen. Die größte derzeit bekannte Primzahl wurde am 15. Mai 2004 gefunden; wie alle anderen Rekord-Primzahlen der letzten Jahre auch ist sie eine sogenannte MERSENNEsche Primzahl, d.h. eine Primzahl der Form $2^p - 1$, wobei auch p eine Primzahl ist. (Die Zelle des französischen Mönchs MARIN MERSENNE (1588–1648) war ein Treffpunkt für Mathematiker wie FERMAT, PASCAL und anderen. MERSENNE selbst beschäftigte sich außer mit seinen Primzahlen auch mit Mechanik, wo er als erster GALILEIs Ideen außerhalb Italiens bekannt machte. Außerdem schrieb er ein Buch über Musik, Musikinstrumente und Akustik.)

Das Projekt GIMPS (Great Internet Mersenne Prime Search, im Internet unter `www.mersenne.org/prime.htm`) sucht systematisch auf rund 240 000 Computern von Freiwilligen nach solchen Zahlen. Der bisherige Rekord ist die oben erwähnte Zahl mit $p = 24\,036\,583$; die Primzahl $2^p - 1$ hat 7 235 733 Dezimalstellen.

Da 309 Dezimalstellen (entsprechend 1024 Bit) verglichen damit recht wenig ist, sollte die Suche nach solchen Primzahlen eigentlich nicht sonderlich schwer sein. Zu bedenken ist allerdings, daß MERSENNEsche Primzahlen relativ selten sind: Die oben erwähnte Zahl ist erst die 41. bekannte, und schon daraus folgt, daß diese Zahlen für die Kryptographie völlig nutzlos sind: 41 Möglichkeiten kann schließlich jeder durchprobieren. (Es ist übrigens nicht bekannt, ob es unendlich viele MERSENNEsche Primzahlen gibt; auch wenn das niemand wirklich glaubt, könntedie 41. auch die letzte sein.)

b) Wie dicht liegen die Primzahlen?

Bevor man daran geht, Primzahlen zu suchen, 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 Lizenz) *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 folgende Verschärfung einer etwa vierzig Jahre älteren Ungleichung von TSCHEBYSCHEFF:

$$0{,}95695\frac{x}{\ln x} < \pi(x) < 1{,}04423\frac{x}{\ln x}\,,$$

$\pi(x)$ verhält sich also asymptotisch ungefähr wie $x/\ln x$. Nach dem Primzahlsatz, den GAUSS vermutete und den HADAMARD und DE LA VALLÉE POUSSIN 1896 bewiesen, ist die Asymptotik sogar exakt, d.h.

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1\,.$$

Numerisch besser ist die Approximation durch den Integrallogarithmus:

$$\pi(x) = \int_2^x \frac{d\xi}{\ln \xi} + O(xe^{-c\sqrt{\ln x}})$$

mit einer (im Prinzip berechenbaren) Konstanten $c > 0$.

Für uns wichtig ist die Folgerung: Unter den Zahlen der Größenordnung N haben der Primzahlen die Dichte $1/\ln N$, d.h. der Abstand zwischen zwei Primzahlen liegt im Durchschnitt bei etwa $\ln N$. Für $N = 10^n$ ist dies

$$\ln 10^n = n \ln 10 \approx 2{,}3n\,;$$

bei hundertstelligen Zahlen ist also etwa jede 230ste prim und bei 1024-Bit-Zahlen ungefähr jede 710. Allerdings sind dies natürlich nur grobe Anhaltspunkte, denn die tatsächliche Verteilung der Primzahlen zeigt enorme Schwankungen: So hat man bislang in allen untersuchten Größenordnungen sowohl Primzahlzwillinge gefunden, d.h. Paare $(p, p+2)$ von Primzahlen, als auch primzahlenfreie Intervalle, die deutlich länger sind als $\ln N$.

c) Das Sieb des Eratosthenes

Das wohl älteste Verfahren, das Primzahlen effizienter als durch Probedivisionen liefert, ist das von ERATOSTHENES (275–194 v.Chr.) angegebene Siebverfahren. In seiner klassischen Form dient es dazu, sich alle Primzahlen unterhalb einer Schranke Nzu verschaffen. Dazu schreibt man die Zahlen von eins bis N (oder auch nur die ungeraden darunter) in eine Reihe, streicht die Nicht-Primzahl Eins und sodann, solange die kleinste noch nicht gestrichene Zahl nicht größer als $\sqrt{N}$ ist, deren sämtliche Vielfache. Was am Ende übrigbleibt, sind genau die Primzahlen aus der Liste.

In dieser Form ist das Sieb für uns nutzlos: Wenn wir eine dreihundertstellige Primzahl brauchen, können wir nicht eine Liste aller Primzahlen unterhalb von 10^{150} erzeugen.

Trotzdem kann es uns auch da eine Hilfe sein: Wenn die Liste anstelle der ersten N natürlichen Zahlen die Zahlen aus einem Suchintervall $[N, N+\ell]$ enthält, können wir immer noch die Vielfachen kleiner Primzahlen aussieben; wir müssen nur für jede Primzahl p, mit der wir sieben, ihr erstes Vielfaches im Suchintervall bestimmen. Dieses ist offensichtlich $N + p - (N \bmod p)$, und ab dort wird jeder p-te Eintrag in der Liste gestrichen.

Natürlich müssen hier die Primzahlen p aus einer separaten Liste kommen, und p wird um Größenordnungen kleiner sein als $\sqrt{N+\ell}$. Somit können wir nicht erwarten, daß alle nicht ausgestrichenen Listenelemente Primzahlen sind. Somit müssen wir diese Zahlen weiteren Tests unterziehen; da diese, wie wir sehen werden, erheblich aufwendiger sind

als das Sieb des ERATOSTHENES, lohnt es sich aber trotzdem, mit dem Sieb zu beginnen.

Da die Feinverteilung der Primzahlen sehr inhomogen ist, sollte man bei der Wahl der Intervallänge ℓ eine gewisse Sicherheitsreserve einplanen und ℓ so wählen, daß im Durchschnitt etwa drei bis zehn Primzahlen in $[N, N+\ell]$ zu erwarten sind, d.h. $\ell \approx a \ln \ell$ mit einem a zwischen drei und zehn. Der Intervallanfang N muß natürlich zufällig gewählt werden.

Die Primzahlen, die wir bei einem solchen Verfahren erwarten können, sind allerdings nicht gleichverteilt: Ist p eine Primzahl und q die größte Primzahl echt kleiner p, so gibt es offenbar genau $p - q$ Anfangswerte N, die zu p als erster Primzahl in $[N, N+\ell]$ führen. Um jeder Primzahl dieselbe Chance zu geben, müßte man Zufallszahlen auf Primalität testen und, sofern eine Zahl den Test nicht besteht, zur nächsten Zufallszahl übergehen. Der Aufwand hierfür ist erheblich größer als der für das Sieb, da wir im Mittel $\ln p$ Zahlen testen müssen, bevor wir eine Primzahl p finden. Da keine Verfahren bekannt sind, wie ein Gegner die bei Verwendung des Siebs resultierenden Abweichungen von einer Gleichverteilung ausnutzen kann, nimmt man diesen Nachteil daher oft in Kauf.

d) Der Fermat-Test

Unabhängig davon, ob wir das Sieb des ERATOSTHENES benutzen oder nicht, brauchen wir auf jeden Fall noch weitere Primzahltest. 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* $a^{p-1} \neq 1 \bmod p$, *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

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

Nachrechnen zeigt, daß

$$3^{(F_{20}-1)/2} \neq \pm 1 \bmod F_{20}\,,$$

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 anscheinend etwas weltabgewandt lebenden Autoren meinten, das sei die teuerste bislang produzierte 1-Bit-Information.)

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

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

und

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

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

(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. (Dieses Argument gilt strenggenommen nur unter der Voraussetzung, daß es sich bei fehlgeschlagenen FERMAT-Tests mit verschiedenen Primzahlen um unabhängige Ereignisse handelt, was wahrscheinlich nicht der Fall ist. Die allgemeine experimentelle Erfahrung mit Primzahlen zeigt jedoch, daß sie sich zumindest in den überprüften Bereichen oft wie zufallsverteilt verhalten.)

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. Die Regulierungsbehörde empfiehlt allerdings, bei probabilistischen Primzahltests eine Irrtumswahrscheinlichkeit von höchsten $2^{-80} \approx 8{,}27 \cdot 10^{-25}$ zuzulassen; diese Schranke wird erst bei knapp zweihundertstelligen Primzahlen mit einem einzigen FERMAT-Test erreicht. Für die jenseits 2007 bis 2008 erforderliche Mindestlänge von 1280 Bit braucht man etwas mehr als 190-stellige Primzahlen, und damit dürfte man in diesem Bereich sein.

Einige Leute reden bei Zahlen, die einen FERMAT-Test bestanden haben, von „wahrscheinlichen Primzahlen“. Das ist natürlich Unsinn: Eine Zahl ist entweder *sicher* prim oder *sicher* zusammengesetzt; für Wahrscheinlichkeiten gibt es hier keinen Spielraum. Besser ist der Ausdruck

„industrial grade primes“, also „Industrieprimzahlen“, der ausdrücken soll, daß wir als Mathematiker zwar nicht entscheiden können, ob die Zahl wirklich prim ist, daß sie aber für industrielle Anwendungen gut genug ist.

Man kann das FERMAT-Verfahren ohne großen Aufwand noch etwas verbessern zu einem Test, den erstmalig ARTJUHOV 1966/67 vorschlug. Die Grundidee ist folgende: Falls p eine Primzahl ist, ist $\mathbb{Z}/p$ ein Körper. Ist dort $a^n = 1$ für eine gerade Zahl $n = 2m$, so erfüllt $x = a^m$ die Gleichung $x^2 = 1$. Da ein quadratisches Polynom über einem Körper höchstens zwei Nullstellen haben kann, muß also $x = \pm 1$ sein. (Falls $\mathbb{Z}/p$ kein Körper ist, p also keine Primzahl, gilt dies nicht: Wäre etwa p das Produkt zweier ungerader Primzahlen, so gäbe es vier Lösungen der Gleichung $x^2 = 1$ in $\mathbb{Z}/p$.)

Dies läßt sich folgendermaßen ausnutzen: Für eine zu testende ungerade Zahl p schreiben wir $p - 1 = 2^r u$ mit einer ungeraden Zahl u. Sodann wählen wir eine Basis a zwischen 2 und $p-2$ und berechnen a^u mod p. Ist diese Zahl gleich eins, so ist erst recht $a^{p-1} \equiv 1 \bmod p$, und der Test ist bestanden. Dasselbe gilt für das Ergebnis $p - 1 \equiv -1 \bmod p$, denn $r \geq 1$ für eine ungerade Zahl p.

Andernfalls quadriere man das Ergebnis höchstens $r - 1$ mal modulo p; dies liefert die Potenzen $a^{2^s u}$ mod p für $s = 1, \ldots, r-1$. Sobald ein Ergebnis gleich $p - 1$ wird, bricht der Test ab mit dem Ergebnis *bestanden;* offensichtlich ist dann $a^{p-1} \equiv 1 \bmod p$. Falls keines der Ergebnisse gleich $p - 1$ ist, kann p keine Primzahl sein, denn dann ist entweder $a^{p-1} \not\equiv 1 \bmod p$, oder aber wir haben eine Zahl gefunden, die von ± 1 verschieden ist, aber trotzdem Quadrat Eins hat, was in einem Körper nicht möglich ist.

Wie MONIER und RABIN 1980 gezeigt haben, ist die Anzahl der Basen a, die ein *falsches* Ergebnis liefern, für die eine zusammengesetzte Zahl p also den Test besteht, kleiner als $p/4$; so etwas wie CARMICHAEL-Zahlen kann es für diesen verschärften Test daher nicht geben.

Natürlich kennt die Mathematik auch Verfahren, um exakt festzustellen, ob eine Zahl prim ist oder nicht; das einfachste besteht darin, alle potentiellen Primteiler einfach auszuprobieren. Wir wollen uns im folgenden überlegen, wie man den FERMAT-Test zu einem solchen Verfahren ausbauen kann. Dazu brauchen wir allerdings zunächst einige weitere algebraische Vorbereitungen:

e) Die multiplikative Gruppe eines endlichen Körpers

Im Körper mit q Elementen erfüllt jedes Element $x \neq 0$ die Gleichung $x^{q-1} = 1$, ist also eine $(q-1)$-te Einheitswurzel. Wenn wir Einheitswurzeln in den *komplexen Zahlen* statt in einem endlichen Körper betrachten, können wir die Lösungen dieser Gleichung leicht angeben: Es sind genau die Zahlen

$$z_\nu = e^{\frac{2\pi i \nu}{q-1}} \quad \text{mit} \quad k = 1, 2, \ldots, q-1 \,.$$

Jede dieser Zahlen läßt sich beispielsweise als Potenz der ersten schreiben, denn

$$e^{\frac{2\pi i \nu}{q-1}} = \left(e^{\frac{2\pi i}{q-1}}\right)^\nu \,.$$

In so einer Situation bezeichnen wir die Menge aller dieser Zahlen als eine *zyklische Gruppe* im Sinne der folgenden

Definition: *a)* Eine Gruppe ist eine Menge G zusammen mit einer Verknüpfung $\cdot : G \times G \to G$ derart, daß gilt:

1. $a(bc) = (ab)c$ für alle $a, b, c \in G$
2. Es gibt ein Element $e \in G$, so daß $ea = ae = a$ für alle $a \in G$.
3. Zu jedem $a \in G$ gibt es ein $b \in G$ mit $ab = ba = e$.
   Die Gruppe heißt *abelsch* oder *kommutativ*, wenn zusätzlich gilt:
4. $ab = ba$ für alle $a, b \in G$.

b) Die Gruppe G heißt *zyklisch*, wenn es ein Element $a \in \mathbb{Z}$ gibt, so daß sich jedes andere Element als Potenz von a schreiben läßt.
c) Die *Ordnung* eines Elements $g \in G$ einer Gruppe G ist die kleinste natürliche Zahl n, für die g^n gleich dem Neutralelement von G ist. Falls es keine solche Zahl n gibt, sagen wir, g habe unendliche Ordnung. *d)* Die Ordnung einer *endlichen* Gruppe G ist die Elementanzahl von G.

Offensichtlich ist eine endliche Gruppe der Ordnung n genau dann zyklisch, wenn sie (mindestens) ein Element der Ordnung n enthält.

So ist beispielsweise die additiven Gruppen Z/n zyklisch, da das Element 1 die Ordnung n hat. (Man beachte, daß wir in dieser additiv geschriebenen Gruppe an Stelle von Potenzen Vielfache betrachten müssen.) Es ist allerdings im allgemeinen nicht das einzige: In $\mathbb{Z}/3$ beispielsweise hat auch 2 diese Eigenschaft, den $1 \cdot 2 = 2, 2 \cdot 2 = 1$ und $3 \cdot 2 = 0$ sind die sämtlichen Elemente von $\mathbb{Z}/3$. Dagegen hat das Element 2 von $\mathbb{Z}/6$ nur die Ordnung drei; als Erzeugende können wir hier nur die Elemente 1 und 5 nehmen. Allgemein überlegt man sich leicht, daß das Element $r \in \mathbb{Z}/n$ die Ordnung $n/\,\mathrm{ggT}(n,r) = \mathrm{kgV}(n,r)$ hat; r ist also genau dann ein erzeugendes Element, wenn r als ganze Zahl betrachtet teilerfremd zu n ist.

Satz von Lagrange: Die Ordnung r eines Elements a einer endlichen Gruppe ist stets ein Teiler der Gruppenordnung n.

Beweis: Wir führen auf G eine Äquivalenzrelation ein, indem wir schreiben $g \sim h$, wenn es eine natürliche Zahl s gibt, so daß $h = a^s g$ ist. Dann besteht die Äquivalenzklasse eines Elements g genau aus den r Elementen $g, ga, ga^2, \ldots ga^{r-1}$, und da jedes Element von G in genau einer Äquivalenzklasse liegt, muß r ein Teiler der Gruppenordnung sein. ∎

Das (bis auf Isomorphie) einzige Beispiel einer unendlichen zyklischen Gruppe ist die Gruppe $\mathbb{Z}$ der ganzen Zahlen; als erzeugendes Element können wir +1 oder -1 wählen. Andere Elemente kommen nicht in Frage, obwohl außer der Null jedes Element unendliche Ordnung hat.

Für gewisse Primzahltests und später auch für Kryptosysteme auf der Basis diskreter Logarithmen wichtig ist der folgende

Satz: Die multiplikative Gruppe eines endlichen Körpers ist zyklisch.

Beweis: Wir wissen bereits, daß jedes Element außer Null eines Körpers mit q Elementen der Gleichung $x^{q-1} = 1$ genügt; die Ordnung eines jeden Elements ist also ein Teiler von $q-1$. Nach obiger Bemerkung folgt die Behauptung des Satzes, wenn es ein Element gibt, dessen Ordnung *genau* $q-1$ ist.

Für jeden Primteiler p_i von $q-1$ hat die Polynomgleichung

$$x^{\frac{q-1}{p_i}} = 1$$

höchstens $(q-1)/p_i$ Lösungen im Körper; es gibt also zu jedem p_i ein Körperelement a_i mit

$$a_i^{\frac{q-1}{p_i}} \neq 1\,.$$

Sei q_i die größte Potenz von p_i, die $q-1$ teilt, und

$$g_i = a_i^{\frac{q}{q_i}}$$

die $(q-1)/q_i$-te Potenz von a_i. Dann ist

$$g_i^{q_i} = a_i^{q-1} = 1 \quad \text{und} \quad g_i^{\frac{q_i}{p_i}} = a_i^{\frac{q-1}{p_i}} \neq 1\,;$$

g_i hat also die Ordnung q_i. Da die verschiedenen q_i Potenzen verschiedener Primzahlen p_i sind, hat daher das Produkt g aller g_i somit das Produkt aller q_i als Ordnung, also $q-1$. Damit ist die multiplikative Gruppe des Körpers zyklisch. ∎

Der obige Beweis ist nicht konstruktiv, und in der Tat gibt es keinen guten Algorithmus zur deterministischen Konstruktion eines Elements g der Ordnung $q-1$. Üblicherweise verwendet man daher in der Computeralgebra bei der Suche nach einem solchen sogenannten primitiven Erzeugers probabilistische Algorithmen, die zufällige Elemente g erzeugen und testen.

f) Anwendung auf Primzahltests

Aus obigem Satz läßt sich leicht ein Test machen, der *garantiert*, daß eine vorgegebene Zahl p prim ist:

Satz: Die natürliche Zahl p ist genau dann prim, wenn es eine ganze Zahl a gibt, so daß

$$a^{p-1} \equiv 1 \bmod p \text{ und } a^{\frac{p-1}{q}} \not\equiv 1 \bmod p \text{ für alle Primteiler } q \text{ von } p-1\,.$$

Beweis: Falls p prim ist, ist $\mathbb{Z}/p$ ein Körper, dessen multiplikative Gruppe die Ordnung $p-1$ hat und zyklisch ist. Ein erzeugendes Element a dieser

Gruppe hat daher die Ordnung $p-1$ und erfüllt damit die Bedingung des Satzes.

Umgekehrt folgt für eine Zahl p, die obige Bedingung erfüllt, daß die multipliktive Gruppe von $\mathbb{Z}/p$ die Ordnung $p-1$ hat, d.h. jedes Element außer der Null ist invertierbar. Damit ist $\mathbb{Z}/p$ ein Körper, also p eine Primzahl. ∎

Die praktische Nützlichkeit dieses Satzes ist leider eher klein, denn er ist nur anwendbar, wenn man alle Primteiler von $p-1$ kennt, d.h. man braucht eine komplette Faktorisierung von $p-1$. Eine Modifikation dieses Satzes führte aber immerhin im August 2002 zum ersten Algorithmus, der beweisbar in polynomialer Zeit feststellt, ob eine gegebene Zahl prim ist oder nicht; er ist unter

`http://www.cse.iitk.ac.in/news/primality.X`

zu finden mit `X` = `ps` oder `pdf` für die Arbeit von MANINDRA AGRAWAL, NEERAJ KAYAL, NITIN SAXENA und `X` = `html` für eine Übersicht.

Seine Laufzeit wächst allerdings mit mindestens der sechsten Potenz der Ziffernlänge von p, so daß auch er bei den heute interessierenden Zahlen weder für Kryptologen noch für algorithmische Zahlentheoretiker interessant ist: Diese haben Algorithmen, die zwar asymptotisch schlechter, dafür aber in realistischen Anwendungen deutlich besser sind.

Diese Algorithmen benutzen anspruchsvollere Mathematik als FERMAT; die meisten arbeiten mit Charaktersummen und/oder elliptischen Kurven. Da sich FERMAT, wie wir gesehen haben, nur selten irrt, sei hier auf ihre Behandlung verzichtet.

§5: Faktorisierungsverfahren

Der offensichtliche Angriff auf RSA ist die Faktorisierung der öffentlich bekannten Zahl N; sobald man diese in ihre beiden Primfaktoren p und q zerlegt hat, ist das Verfahren gebrochen. Wir wollen daher in diesem Paragraphen sehen, welche Möglichkeiten es gibt, N in seine Primfaktoren zu zerlegen.

a) Mögliche Ansätze zur Faktorisierung

Grundsätzlich gibt es zwei Klassen von Verfahren, mit denen man einen Teiler einer natürlichen Zahl N finden kann: Einmal Verfahren, deren erwartete Laufzeit von der Länge des Faktors abhängt, zum anderen solche, deren Laufzeit nur von N abhängt.

Bei der Anwendung von RSA wird man, um Verfahren der ersten Kategorie auszubremsen, p und q ungefähr gleich groß wählen, so daß diese Verfahren im schlechtestmöglichen Fall arbeiten müssen.

Die einfachste Art der Faktorisierung ist das Abdividieren von Primzahlen; zumindest für kleine Primfaktoren; mindestens bis etwa 2^{16}) ist dies auch die schnellste und effizienteste Methode, da die anderen Verfahren Schwierigkeiten haben, Produkte kleiner Primzahlen zu trennen.

Für etwas größere Faktoren bis zu etwa acht Dezimalstellen ist die POLLARDsche Monte-Carlo-Methode oder ρ-Methode sehr gut geeignet: Man erzeugt mit einem quadratischen Generator (populär ist z.B. $x_{i+1} = x_i^2 + c \bmod N$) Zufallszahlen und berechnet deren ggT mit der zu faktorisierenden Zahl. Da der EUKLIDische Algorithmus im Vergleich zur Erzeugung der Zufallszahlen relativ teuer ist, empfiehlt es sich, die erzeugten Zufallszahlen zunächst modulo N miteinander zu multiplizieren und dann erst in etwa jedem hundertsten Schritt den ggT von N mit diesem Produkt zu berechnen. (Dies setzt voraus, daß alle sehr kleinen Faktoren bereits abdividert sind; sonst ist die Gefahr zu groß, daß im Produkt von hundert Zufallszahlen mehr als ein Primfaktor steckt.) Bei Faktoren mit mehr als acht Dezimalstellen wird die Methode schnell langsamer, so daß man dann zu alternativen Verfahren übergehen sollte.

Die nächste Klasse von Verfahren beruht auf gruppentheoretischen Überlegungen, im wesentlichen dem kleinen Satz von FERMAT im Falle zyklischer Gruppen und Verallgemeinerungen auf weitere Gruppen wie die multiplikative Gruppe eines Körpers $\mathbb{F}_{p^2}$ oder einer elliptischen Kurve. Diese Verfahren sind sehr effizient, wenn die Gruppenordnung nur relativ kleine Primteiler hat. Aus diesem Grund wurde früher häufig empfohlen, daß für die Primteiler p eines RSA-Modulus N sowohl $p-1$ als auch $p+1$ jeweils mindestens einen „großen“ Primfaktor haben sollen; auch heute ist diese Empfehlung noch in einigen Büchern

zu finden. Da die genannten Verfahren ihre Stärke jedoch bei Faktoren mit einer Länge von bis etwa 30 oder 35 Dezimalstellen haben und ein N mit 70 Dezimalstellen für RSA heute natürlich völlig unsicher ist, hat die Empfehlung heute ihre Berechtigung verloren und wir müssen uns insbesondere auch nicht näher mit den Faktorisierungsverfahren beschäftigen, vor denen sie schützen sollte.

Umso interessanter ist dagegen ein Verfahren, dessen Stärke bei nahe beieinander liegenden Primfaktoren liegt. Es wurde von PIERRE DE FERMAT vorgeschlagen und beruht auf der Formel

$$x^2 - y^2 = (x + y)(x - y)\,.$$

Ist $N = pq$ Produkt zweier ungerader Primzahlen, so ist

$$N = (x + y)(x - y) \quad \text{mit} \quad x = \frac{p+q}{2} \quad \text{und} \quad y = \frac{p-q}{2}\,;$$

zusammen mit obiger Formel folgt, daß dann

$$N + y^2 = x^2$$

ist.

FERMAT berechnet nun für $y = 0, 1, 2, \ldots$ die Zahlen $N + y^2$; falls er auf ein Quadrat x^2 stößt, berechnet er

$$\text{ggT}(N, x + y) \quad \text{und} \quad \text{ggT}(N, x - y)\,.$$

Wenn er Pech hat, sind dies die beiden Zahlen eins und N, wenn er Glück hat, sind es p und q. Für zufällig gewählte Paare (x, y) kommt beides jeweils mit fünfzigprozentiger Wahrscheinlichkeit vor.

Falls p und q nahe beieinander liegen, führt schon ein kleines y zur korrekten Faktorisierung; bei der Wahl der Primzahlen muß also darauf geachtet werden, daß sie zwar die gleiche *Größenordnung* haben, aber nicht zu weit beieinander liegen. Liegt etwa p in der Größenordnung von $2q$, hat die Differenz die gleiche Größenordnung wie p, und FERMATS Verfahren bräuchte etwa p Rechenschritte, wird also vom Aufwand her vergleichbar mit der Faktoriserung durch Abdividieren. Somit kann man sich auch gegen diese Attacke recht gut schützen.

Wirklich gefährlich sind eine Klasse von Siebverfahren, die auf FERMATS Methode aufbauen. Diese Verfahren sind die schnellsten derzeit

bekannten zur Faktorisierung von RSA-Moduln; die Wahl einer sicheren Ziffernlänge hängt also davon ab, welche Zahlen diese Verfahren faktorisieren können.

b) Das quadratische Sieb

Das quadratische Sieb ist der Grundalgorithmus der ganzen Klasse; es ist logisch einfacher, allerdings vor allem für große Zahlen auch deutlich langsamer als die Variationen, mit denen wir uns im nächsten Abschnitt beschäftigen werden.

Bei allen diesen Verfahren geht es darum, Zahlenpaare (x, y) zu finden, für die

$$x^2 \equiv y^2 \bmod N$$

ist. Für diese erwarten wir, daß in etwa der Hälfte aller Fälle

$$\text{ggT}(x + y, N) \quad \text{und} \quad \text{ggT}(x - y, N)$$

nichttriviale Teiler von N sind.

Bei quadratischen Sieb betrachten wir dazu das Polynom

$$f(x) = \left(x + \left\lceil\sqrt{N}\right\rceil\right)^2 - N\,.$$

Offensichtlich ist für jedes x

$$f(x) \equiv \left(x + \left\lceil\sqrt{N}\right\rceil\right)^2 \bmod N\,,$$

wobei links und rechts verschiedene Zahlen stehen. Insbesondere steht links im allgemeinen keine Quadratzahl.

Falls wir allerdings Werte $x_1, x_2, \ldots, x_r$ finden können, für die das Produkt der $f(x_i)$ eine Quadratzahl ist, dann ist

$$\prod_{i=1}^{r} f(x_i) \equiv \prod_{i=1}^{r} \left(x + \left\lceil\sqrt{N}\right\rceil\right)^2 \bmod N$$

eine Relation der gesuchten Art.

Um die x_i zu finden, betrachten wir eine Menge $\mathcal{B}$ von Primzahlen, die sogenannte Faktorbasis. Typischerweise enthält $\mathcal{B}$ für die Faktorisierung einer etwa hundertstelligen Zahl etwa 100–120 Tausend Primzahlen, deren größte somit, wie die folgende Tabelle zeigt, im einstelligen Millionenbereich liegt.

n	n-te Primzahl	n	n-te Primzahl
100 000	1 299 709	600 000	8 960 453
200 000	2 750 159	700 000	10 570 841
300 000	4 256 233	800 000	12 195 257
400 000	5 800 079	900 000	13 834 103
500 000	7 368 787	1 000 000	15 485 863

Beim quadratischen Sieb interessieren nur x-Werte, für die $f(x)$ als Produkt von Primzahlen aus $\mathcal{B}$ (und eventuell auch Potenzen davon) darstellbar ist. Ist

$$f(x_i) = \prod_{p \in \mathcal{B}} p^{e_{ip}} ,$$

so ist

$$\prod_{i=1}^{r} f(x_i)^{\varepsilon_i} = \prod_{p \in \mathcal{B}} p^{\sum_{i=1}^{r} \varepsilon_i e_{ip}}$$

genau dann ein Quadrat, wenn

$$\sum_{i=1}^{r} \varepsilon_i e_{ip}$$

für alle $p \in \mathcal{B}$ gerade ist. Dies hängt natürlich nur ab von den ε_i mod 2 und den e_{ip} mod 2; wir können ε_i und e_{ip} daher als Elemente des Körpers mit zwei Elementen auffassen und bekommen dann über $\mathbb{F}_2$ das Gleichungssystem

$$\sum_{i=1}^{r} \varepsilon_i e_{ip} = 0 \quad \text{für alle } p \in \mathcal{B} .$$

Betrachten wir die ε_i als Variablen, ist dies ein homogenes lineares Gleichungssystem in r Variablen mit soviel Gleichungen, wie es Primzahlen in der Faktorbasis gibt. Dieses Gleichungssystem hat nichttriviale Lösungen, falls die Anzahl der Variablen die der Gleichungen übersteigt,

falls es also mehr Zahlen x_i gibt, für die $f(x_i)$ über der Faktorbasis faktorisiert werden kann, als Primzahlen in der Faktorbasis.

Für jede nichttriviale Lösung ist

$$\prod_{i=1}^{r} f(x_i)^{\varepsilon_i} = \prod_{i=1}^{r} \left(x + \left\lfloor \sqrt{N} \right\rfloor \right)^{2\varepsilon_i} \bmod N$$

eine Relation der Form $x^2 \equiv y^2 \bmod N$, die mit einer Wahrscheinlichkeit von etwa ein halb zu einer Faktorisierung von N führt. Falls wir zehn linear unabhängige Lösungen des Gleichungssystems betrachten, führt also mit einer Wahrscheinlichkeit von etwa 99,9% mindestens eine davon zu einer Faktorisierung.

Da ε_i nur die Werte 0 und 1 annimmt, stehen in obigem Produkt natürlich keine echten Potenzen: Man multipliziert einfach nur die Faktoren miteinander, für die $\varepsilon_i = 1$ ist. Außerdem interessieren nicht die links- und rechtsstehenden Quadrate, sondern deren Quadratwurzeln; tatsächlich also berechnet

$$x = \prod_{p \in \mathcal{B}} p^{\frac{1}{2} \sum_{i=1}^{r} \varepsilon_i e_{ip}} \quad \text{und} \quad y = \prod_{i=1}^{r} \left(x + \left\lfloor \sqrt{N} \right\rfloor \right)^{\varepsilon_i} .$$

wobei beide Produkte nur modulo N berechnet werden müssen.

Zum besseren Verständnis des Verfahrens wollen wir versuchen, damit die Zahl 15 zu faktorisieren. Dies ist zwar eine sehr untypische Anwendung, da das quadratische Sieb üblicherweise erst für mindestens etwa vierzigstellige Zahlen angewandt wird, aber zumindestens das Prinzip sollte auch damit klarwerden.

Als Faktorbasis verwenden wir die Menge

$$\mathcal{B} = \{2, 3, 7, 11\} ;$$

die Primzahl fünf fehlt, da $3 \cdot 5 = 15$ ist und daher bei einer Faktorbasis, die sowohl drei als auch fünf enthält, die Gefahr zu groß ist, daß die linke wie auch die rechte Seite der Kongruenz durch fünfzehn teilbar ist. Bei realistischen Anwendungen muß man auf solche Überlegungen keine Rücksicht nehmen, denn dann sind die Elemente der Faktorbasis

höchstens siebenstellig und somit erheblich kleiner als die gesuchten Faktoren.

Wir berechnen $f(x)$ für $x = 1, 2, \ldots$, bis wir einige Funktionswerte haben, die über der Faktorbasis faktorisiert werden können. Die faktorisierbaren Werte sind in folgender Tabelle zusammengestellt:

x	$x + \lfloor\sqrt{N}\rfloor$	$f(x)$	*Faktorisierung*
1	4	1	
3	6	21	$3 \cdot 7$
5	8	49	7^2
6	9	66	$2 \cdot 3 \cdot 11$
10	13	154	$2 \cdot 7 \cdot 11$
54	57	3234	$2 \cdot 3 \cdot 7^2 \cdot 11$

Die erste und die dritte Zeile sind selbst schon Relationen der gesuchten Art, nämlich

$$4^2 \equiv 1 \bmod 15 \quad \text{und} \quad 8^2 \equiv 7^2 \bmod 15\,.$$

Die zweite Relation ist nutzlos, denn $8 - 7 = 1$ und $8 + 7 = 15$. Die erste dagegen führt zur Faktorisierung, denn

$$\mathrm{ggT}(4 + 1, 15) = 5 \quad \text{und} \quad \mathrm{ggT}(4 - 1, 15) = 3\,.$$

Da dies aber ein Zufall ist, der bei großen Werten von N so gut wie nie vorkommt, wollen wir das ignorieren und mit den Relationen zu $x = 3, 6, 10$ und 51 arbeiten:

$$\begin{aligned} 6^2 &\equiv 3 \cdot 7 && \bmod 15 \\ 9^2 &\equiv 2 \cdot 3 \cdot 11 && \bmod 15 \\ 13^2 &\equiv 2 \cdot 7 \cdot 11 && \bmod 15 \\ 54^2 &\equiv 2 \cdot 3 \cdot 7^2 \cdot 11 && \bmod 15 \end{aligned}$$

Multipliziert man die ersten drei dieser Relationen miteinander, folgt

$$(6 \cdot 9 \cdot 13)^2 \equiv (2 \cdot 3 \cdot 7 \cdot 11)^2 \bmod 15$$

oder $702^2 \equiv 462^2 \bmod 15$. Da

$$\mathrm{ggT}(702 - 462, 15) = \mathrm{ggT}(240, 15) = 15$$

ist, bringt das leider nichts.

Wir erhalten auch dann rechts ein Quadrat, wenn wir das Produkt der ersten, dritten und vierten Relation bilden; dies führt auf

$$(6 \cdot 13 \cdot 57)^2 \equiv (2 \cdot 3 \cdot 7^2 \cdot 11)^2 \bmod 15$$

oder $4446^2 \equiv 3234^2 \bmod 15$. Hier ist

$$\mathrm{ggT}(4446 - 3234, 15) = \mathrm{ggT}(1212, 15) = 3\,,$$

womit wir die Zahl 15 faktorisiert haben – wenn auch nicht unbedingt auf die einfachstmögliche Weise.

Bei realistischen Beispielen sind die Funktionswerte $f(x)$ deutlich größer als die Primzahlen aus der Faktorbasis; außerdem liegen die vollständig faktorisierbaren Zahlen viel dünner als hier: Bei der Faktorisierung einer hundertstelligen Zahl etwa muß man davon ausgehen, daß nur etwa jeder 10^9-te Funktionswert über der Faktorbasis zerfällt.

Daher ist es wichtig, ein Verfahren zu finden, mit dem diese wenigen Funktionswerte schnell und einfach bestimmt werden können. Das ist zum Glück möglich:

Der Funktionswert $f(x)$ ist genau dann durch p teilbar, wenn

$$f(x) \equiv 0 \bmod p$$

ist. Da für $x, y, a, b \in \mathbb{Z}$ und mit $x \equiv y \bmod p$ und $a \equiv b \bmod p$ gilt

$$a + x \equiv b + y \bmod p \quad \text{und} \quad a \cdot x \equiv b \cdot y \bmod p$$

und für $n \in \mathbb{N}$ auch

$$x^n \equiv y^n \bmod p\,,$$

ist für jedes Polynom f mit ganzzahligen Koeffizienten

$$f(x) \equiv f(y) \bmod p\,.$$

Ist also insbesondere $f(x) \equiv 0 \bmod p$, so ist auch

$$f(x + kp) \equiv 0 \bmod p \quad \text{für alle } k \in \mathbb{Z}\,.$$

Es genügt daher, im Bereich $0 \le x < p - 1$ nach Werten zu suchen, für die $f(x)$ durch p teilbar ist.

Dazu kann man f auch als Polynom über dem Körper mit p Elementen betrachten und nach Nullstellen in diesem Körper suchen. Für Polynome großen Grades und große Werte von p kann dies recht aufwendig sein; hier, bei einem quadratischen Polynom, müssen wir natürlich einfach eine quadratische Gleichung lösen: In $\mathbb{F}_p$ wie in jedem anderen Körper auch gilt

$$f(x) = \left(x - \left[\sqrt{N}\right]\right)^2 - N = 0 \Longleftrightarrow \left(x - \left[\sqrt{N}\right]\right)^2 = N\,,$$

und diese Gleichung ist genau dann lösbar, wenn es ein Element $w \in \mathbb{F}_p$ gibt mit Quadrat N, wenn also in $\mathbb{Z}$

$$w^2 \equiv N \bmod p$$

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

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

Für diese ist

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

so daß $af(x)$ 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

$$f(-M) \approx \frac{1}{a}\bigl(a^2M^2 - N\bigr)\,;$$

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 $b^2 \equiv N \bmod a$ 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

$$F(x,y) = a_dx^d + a_{d-1}x^{d-1}y + \cdots + a_1xy^{d-1} + a_0y^d\,,$$

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

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

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

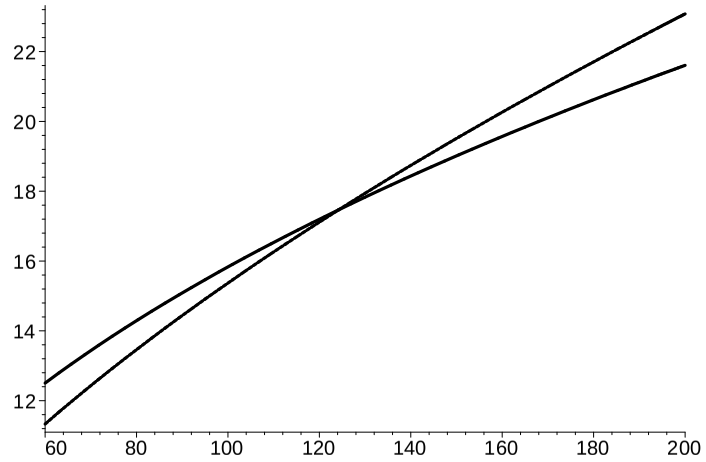
$$e^{c(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}} \quad c = \sqrt[3]{\frac{64}{9}} \approx 1{,}923$$

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.

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.



Zeitaufwand für quadratisches Sieb und Zahlkörpersieb

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 aktuellen *Factoring Challenges* von RSA Security sind derzeit unter

`http://www.rsasecurity.com/rsalabs/node.asp?id=2093`

zu finden; sie beginnen mit einer 576-Bit-Zahl RSA-576 (174 Dezimalstellen), für deren Faktorisierung ein Preis von 10 000 $ ausgesetzt ist, und endet mit einer Zahl RSA-2048 mit 617 Dezimalstellen, für die es immerhin schon 200 000$ gibt.

Am 3. Dezember 2003 gab der Bonner Mathematiker JENS FRANKE die Faktorisierung von RSA-576 bekannt, publizierte aber bislang abgesehen vom Ergebnis und der Liste der Beteiligten nur wenige Einzelheiten. Bekannt ist nur, daß zum Sieben Desktop-PCs des Max-Planck-Instituts für Mathematik und verschiedener anderer Institute verwendet wurden und das lineare Gleichungssystem auf dem Supercomputer parnass2 des Instituts für numerische Simulation der Universität Bonnm gelöst wurde.

Mehr Einzelheiten sind über den vorangegangenen Rekord bekannt: Dabei handelt es sich um die Vorgängerliste der bisherigen *Factoring Challenges*, in der die Zahlen nach der Anzahl ihrer Dezimalziffern benannt wurden. Die größte faktorisierte Zahl aus dieser Liste ist RSA-155, eine 155-stellige natürliche Zahl also. 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, lag 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 letzteren noch einiges an kreativer Buchführung notwendig wäre, falls der Faktorisierungsversuch geheim bleiben sollte.

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. Im April 2004 erschien ein *preprint* von vier japanischen Mathematikern, die sich die Zahlen RSA-100 bis RSA-150 der alten Liste noch einmal vornahmen und alle mit derselben Hardware faktorisierten:

KAZAMARO AOKI, YUJI KIDA, TAKESHI SHIMOYAMA, HIROKI UEADA: GNFS Factoring Statistics of RSA-100, 110, . . . , 150

Sie benutzten Pentium 4 Computer mit 2,53 GHz und 1024 MB RDRAM unter FreeBSD und berechneten die Siebzeit *hochgerechnet auf einen solchen PC*. Zur Lösung des linearen Gleichungssystems verwendeten sie ein Cluster aus 16 über 100 Leitungen vernetzten PCs. Die Zeiten waren:

	Siebzeit dd:hh:mm:ss	*LGS-Zeit hh:mm:ss*
RSA-100	0 : 15 : 04 : 31	00:07:19
RSA-110	2 : 05 : 47 : 56	00:28:49
RSA-120	9 : 03 : 28 : 58	02:24:15
RSA-130	26 : 19 : 09 : 07	08:47:28
RSA-140	77 : 20 : 24 : 18	15:07:39
RSA-150	238 : 09 : 27 : 40	101:31:17

Die für RSA-150 angegebenen gut 238 Tage Siebzeit würden bei parallelem Rechnen auf 16 PCs zu knapp 15 Tagen, dazu kommen gut vier Tage für das LGS. Mit knapp drei Wochen Rechenzeit kann eine solche Zahl also bereits heute von jedem faktorisiert werden, der Zugang zu einem PC-Pool hat.

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 beschlëunigen 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

Wie wir bereits gesehen haben, veröffentlichen das Bundesamt für Sicherheit in der Informationstechnik und die Regulierungsbehörde für Telekommunikation und Post jedes Jahr Empfehlungen für die Länge von RSA-Schlüsseln; falls man rechtsgültige elektronische Unterschriften braucht, muß man auf jeden Fall darauf achten, daß während der Vertragslaufzeit diese Mindestanforderungen erfüllt sind.

Für die Übermittlung verschlüsselter privater oder auch geschäftlicher Informationen darf allerdings 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.

Schlüssel mit 512 Bit sind auch heute noch recht verbreitet, da amerikanische Firmen bis vor wenigen Jahren weder Hardware noch Software zur RSA-Verschlüsselung exportieren durften, die mit längeren Schlüsseln arbeitet. Wie wir oben gesehen haben, bieten diese Schlüssel heute noch einen einigermaßen akzeptablen Schutz gegen Gelegenheitshacker, aber definitiv nicht mehr. In wenigen Jahren dürften sie wohl völlig unsicher sein.

Eine weitere magische Grenze sind 1024 Bit, da viele Hersteller von kryptographischer Hardware keine längeren Schlüssel unterstützen oder

gerade erst anfangen, sie zu unterstützen. Dies war schließlich auch der Hauptgrund dafür, daß die Regulierungsbehörde für Telekommunikation und Post so lange brauchte, Empfehlungen für Schlüssel jenseits dieser Grenze durchzusetzen.

Es ist unwahrscheinlich, daß heute irgendjemand in der Lage ist, ohne Zusatzinformationen einen 1024 Bit-Modul zu faktorisieren, allerdings kann niemand vorhersagen, welche neuen Faktorisierungsverfahren in den nächsten Jahren möglicherweise entdeckt werden oder welche neuen Rechnerarchitekturen möglichweise für einen Durchbruch auf dem Gebiet der Faktorisierung sorgen werden. Daher muß man für die Verschlüsselung von Information, die auch in einigen Jahren noch geheim sein muß, mit einer recht großzügig bemessenen Sicherheitsreserve arbeiten, und vor diesem Hintergrund erscheinen 1024 Bit-Moduln bedenklich.

Selbst bei 2048 Bit-Moduln können wir nicht völlig sicher sein, daß sie noch in mehreren Jahren sind: 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 von RSA-Moduln liegt, wird die Faktorisierung dieser Moduln nicht mehr aufwendiger sein als die Verschlüsselung eines kurzen Texts. Dann wird also RSA *als Verfahren* unsicher.

Bislang gibt es allerdings gerade erst einem Quantencomputer, der leistungsfähig genug war, um die Zahl 15 in ihre Primfaktoren zu zerlegen; er benutzte einen Ansatz, bei dem es unwahrscheinlich erscheint, daß man damit wesentlich mehr als die sieben dort realisierten Quantenbits erreichen kann. Bei anderen Technologien, die erheblich besser skalierbar sein sollten, kam man bislang noch nicht über drei Quantenbits hinaus. Wir wollen Quantencomputer daher 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 Hardware 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 alle diese Extrapolationen über den Haufen geworfen werden.

§ 6: Weitere Aspekte zum RSA-Verfahren

Wir haben die Sicherheit des RSA-Verfahrens bislang nur unter dem Aspekt der Faktorisierung des Moduls N betrachtet. Tatsächlich muß man ein Kryptoverfahren aber immer als ganzes betrachten; viele im Prinzip gute Verfahren wurden in der Praxis schon durch unsachgemäße Anwendung angreifbar und konnten geknackt werden. In diesem Paragraphen sollen einige im Zusammenhang mit RSA wesentliche Aspekte dieses Problems diskutiert werden. Als ein roter Faden wird sich dabei immer wieder zeigen, daß leider fast alles, was die Anwendung von RSA effizienter macht, gleichzeitig zu Sicherheitsproblemen führt und oft die Faktorisierung selbst eines 2048 Bit-Moduls in wenigen Sekunden ermöglicht.

a) Das Problem der Schlüsselübergabe

Der große Vorteil asymmetrischer Kryptoverfahren besteht darin, daß die beiden Teilnehmer keinen gemeinsamen Schlüssel zu vereinbaren brauchen; jeder verwendet einfach den öffentlichen Schlüssel des Empfängers. Wie bekommt er den aber?

Falls es einem Gegner gelingt, dem Absender einen selbst konstruierten Schlüssel als Schlüssel des Empfängers zu unterschieben, kann nur der *Gegner* die so verschlüsselte Nachricht lesen, nicht aber der intendierte Empfänger.

Für die Sicherheit von RSA ist daher von entscheidender Wichtigkeit, daß die benutzten Schlüssel die korrekten sind. Da E-Mails und Webseiten gefälscht werden können, empfiehlt sich bei unbekannten Empfängern, den Schlüssel von einer Zentrale zu holen, die Name und Schlüssel des Empfängers digital unterschreibt und dann übermittelt. Auf diese Weise muß nur der öffentliche Schlüssel der Zentrale allgemein bekannt sein, und der kann leicht über die verschiedensten elektronischen und klassischen Medien häufig publiziert werden. In Deutschland zertifiziert die Regulierungsbehörde für Telekommunikation und Post solche Zentralen, sofern sie die Gewähr für eine sichere und zuverlässige Schlüsselübermittlung bieten.

b) Primzahlen sind Wegwerfartikel

Die Suche nach einer Primzahl mit 1024 Bit kann auf einem nicht sonderlich leistungsfähigen PC rund eine Minute dauern; wenn man viele Schlüssel erzeugen muß, bietet sich also an, mit den teuren Primzahlen sparsam umzugehen.

Genau das darf man aber aus Sicherheitsgründen auf keinen Fall tun: Wenn jemals zwei unterschiedliche Moduln M, N dieselbe Primzahl p enthalten, kann p leicht als ggT von M und N berechnet werden, so daß beide Moduln faktorisiert sind. Der EUKLIDische Algorithmus erfordert auch für 2048-Bit-Zahlen auf einem Standard-PC einen Aufwand, der höchstens im unteren Sekundenbereich liegt; es ist also durchaus möglich, auch bei einer Liste von mehreren Tausend Moduln für jedes Paar den ggT zu berechnen.

Der sichere Umgang mit Primzahlen besteht darin, die Primzahlen sofort nach Berechnung des öffentlichen und des privaten Schlüssels zu vernichten. Die Wahrscheinlichkeit, daß später wieder einmal eine dieser Primzahlen als Zufallsprimzahl auftaucht, liegt bei vernünftiger Implementierung des „Zufalls“ deutlich unter 10^{-100} und kann daher für alle praktischen Zwecke ignoriert werden.

c) Kenntnis des privaten Exponenten führt zur Faktorisierung

Zu zwei RSA-Moduln, die eine Primzahl gemeinsam haben, liefert der EUKLIDische Algorithmus diese Primzahl und damit die Faktorisierung der beiden Moduln. Falls die beiden Moduln allerdings gleich sind, läßt sich mit dem EUKLIDischen Algorithmus natürlich nichts anfangen-.

Innerhalb einer Organisation, in der RSA-Schlüssel zentral vergeben werden, könnte es sich daher anbieten, den Modul N für alle Teilnehmer gleich zu wählen und lediglich für jeden Teilnehmer einen eigenen öffentlichen Exponenten e und zugehörigen geheimen Exponenten d zu erzeugen.

Bei einem solchen System könnte dann natürlich die Zentrale alle Nachrichten aller Teilnehmer entschlüsseln, aber daß kann in gewissen Fällen durchaus erwünscht sein: Falls es sich etwa im Bankenbereich um Nachrichten handelt, die Geld transferieren, muß die Zentrale alles kontrollieren können.

Schlimm wäre allerdings, wenn auch einzelne Teilnehmer alles entschlüsseln könnten, denn dann wäre es möglich, Überweisungsaufträge mit falschem Namen zu unterschreiben, womit jede Art von Betrug möglich würde.

Wir wollen uns überlegen, wie ein Teilnehmer mit seinem öffentlichen und privaten Exponenten die Zahl N faktorisieren kann und damit die privaten Exponenten aller anderer Teilnehmer aus den öffentlichen Exponenten berechnen kann.

Dazu sei $N = pq$ Produkt zweier Primzahlen, e sei der öffentliche Exponent und d der private. Dann ist für alle $a \in \mathbb{Z}$

$$(a^e)^d = a^{ed} \equiv a \bmod N\,;$$

für ein zu N teilerfremdes a ist also

$$a^{de-1} \equiv 1 \bmod N\,.$$

Wir schreiben

$$de - 1 = 2^r \cdot u \quad \text{mit einer ungeraden Zahl } u$$

und betrachten die Folge der Zahlen

$$a^u \bmod N, \quad a^{2u} \bmod N, \quad \ldots, \quad a^{2^r u} \bmod N\,.$$

Die letzte dieser Zahlen ist stets gleich eins; wenn wir Pech haben, ist für das gerade betrachtete a auch schon die erste gleich eins. Wenn nicht, gibt es einen kleinsten Exponenten s, so daß

$$a^{2^s u} \not\equiv 1 \bmod N \quad \text{und} \quad a^{2^{s+1} u} \equiv 1 \bmod N\,.$$

Es könnte sein, daß dann $a^{2^s u} \equiv -1 \bmod N$ ist, aber da N eine zusammengesetzte Zahl ist, muß das nicht der Fall sein: Modulo 15 ist beispielsweise auch vier eine Zahl mit Quadrat eins.

Modulo einer Primzahl hat eins natürlich nur die beiden Quadratwurzeln ± 1, denn die natürlichen Zahlen modulo einer Primzahl bilden einen Körper, und in einem Körper kann das quadratische Polynom $x^2 - 1$ höchstens zwei Nullstellen haben.

Ist aber $x^2 \equiv 1 \bmod N = pq$, so ist auch $x^2 \equiv 1 \bmod p$ und mod$q$, also $x \equiv \pm 1 \bmod p$ und $x \equiv \pm 1 \bmod q$, wobei nicht in beiden Fällen das gleiche Vorzeichen stehen muß. In der Hälfte aller Fälle werden beide Vorzeichen gleich sein; dann ist $x \equiv \pm 1 \bmod N$ mit demselben Vorzeichen.

Ist aber etwa $x \equiv 1 \bmod p$ und $x \equiv -1 \bmod q$, so ist

$$p = \mathrm{ggT}(x - 1, N) \quad \text{und} \quad q = \mathrm{ggT}(x + 1, N)\,,$$

sobald wir eine solche Zahl kennen, haben wir N also faktorisiert.

Wenn wir mit einem zufällig gewählten a so wie oben verfahren, werden wir in etwa der Hälfte aller Fälle eine von ± 1 verschiedene Quadratwurzel der Eins erhalten. Mit nur wenigen Werten für a bekommen wir fast sicher eine Faktorisierung von N.

d) Der chinesische Restesatz

Das Argument im vorigen Abschnitt kann auch zu einer schnelleren Durchführung der RSA-Entschlüsselung und damit zu schnelleren elektronischen Unterschriften führen: Die Berechnung von $m^d \bmod N$ besteht aus Quadrierungen und Multiplikationen modulo N, und deren Aufwand steigt quadratisch mit der Ziffernlänge von N. Kennt man also die Faktorisierung $N = pq$, kann man die beiden Werte $a^d \bmod p$ und $m^d \bmod q$ in jeweils einem Viertel der Zeit für $m^d \bmod N$ berechnen, beide zusammen also in der halben Zeit. Das Gesamtergebnis ist dadurch eindeutig bestimmt, denn da p und q teilerfremd sind, ist jede Zahl die modulo p und modulo q bekannt ist, auch modulo N eindeutig bestimmt.

Dies läßt sich leicht konstruktiv durchführen: Mit dem erweiterten Euklidischen Algorithmus findet man Zahlen α, β, so daß $\alpha p + \beta q = 1$ ist; dann ist

$$\beta q \equiv 1 \bmod p \quad \text{und} \quad \alpha p \equiv 1 \bmod q\,.$$

Für zwei beliebig vorgegebene Zahlen a, b ist entsprechend

$$a\beta q \equiv a \bmod p \quad \text{und} \quad b\alpha p \equiv b \bmod p$$

eine Lösung der Kongruenz

$$x \equiv a \bmod p \quad \text{und} \quad x \equiv b \pmod q\,.$$

Da α und β nur einmal für p und q berechnet werden müssen, ist der Aufwand für das Zusammensetzen der Restklassen modulo p und modulo q zu einer Restklasse modulo N sehr gering.

Mit diesem Verfahren läßt sich der Aufwand für eine elektronische Unterschrift also praktisch halbieren, was vor allem dann von Bedeutung ist, wenn die Unterschrift mit einer Smartcard geleistet wird.

Leider ist das Verfahren aber mit einem potentiell katastrophalen Risiko verbunden: Fall bei einer der beiden Rechnungen

$$a \leftarrow m^d \bmod p \quad \text{und} \quad b \leftarrow m^d \bmod q$$

ein Fehler auftritt, ist das Ergebnis für $m^d \bmod N$ korrekt modulo der einen, nicht aber modulo der anderen Primzahl. Nehmen wir etwa an,

das Ergebnis modulo q sei falsch; dann wird also eine Unterschrift u berechnet, die modulo p kongruent ist zu m^d, nicht aber modulo q.

Zum Überprüfen der Unterschrift berechnet der Empfänger u^e mod N und vergleicht dies mit m; im vorliegenden Beispiel stimmen die beiden Zahlen nur modulo p überein, nicht aber modulo q, sie sind also insbesondere verschieden, so daß die Unterschrift nicht akzeptiert wird.

Da $u^e - m$ aber durch p teilbar ist und nicht durch q, kann der Prüfer nun

$$p = \text{ggT}(u^e - m, N)$$

berechnen und dadurch N faktorisieren; er kann also künftig die Unterschrift des anderen nachmachen.

Hardwarefehler, die bei der Berechnung von einem der beiden Teilergebnisse zu einem falschen Ergebnis führen sind sicherlich sehr seltene Ereignisse, allerdings kann man dem nachhelfen: Durch Magnetfelder, Mikrowelle, Hitze, mechanische Belastung und ähnliches kann man die Karte durchaus so beeinflussen, daß Fehler wahrscheinlich, aber nicht zu wahrscheinlich werden, so daß man eine gute Chance hat, daß genau eines der beiden Zwischenergebnisse falsch berechnet wird.

Sicherer ist es also auf jeden Fall, wenn man trotz des rund doppelt so hohen Aufwands direkt modulo N rechnet; die oben aufgestellte Regel, wonach man die Primzahlen so schnell wie möglich vergessen sollte, hat durchaus ihren Sinn.

Die Methode, eine Zahl modulo N aus ihren Restklassen modulo von Teilern von N zu bestimmen, hat aber noch viele andere Anwendungen, so daß wir den Satz hier allgemein formulieren wollen. Er wurde angeblich früher von chinesischen Generälen benutzt, um Truppenstärken zu berechnen, indem sie die Soldaten in Reihen verschiedener Breite antreten ließen und dann die Anzahl der Soldaten in der letzten Reihe zählten.

Chinesischer Restesatz: Die natürlichen Zahlen $d_1, \ldots, d_r$ seien paarweise teilerfremd und $N = d_1 \cdots d_r$ sei ihr Produkt. Dann hat das Gleichungssystem

$$x \equiv a_1 \bmod d_1, \quad \ldots, \quad x \equiv a_r \bmod d_r$$

für jede Wahl der a_i eine modulo N eindeutig bestimmte Lösung; diese kann mit Hilfe des erweiterten EUKLIDischen Algorithmus berechnet werden.

Beweis: Für nur zwei Zahlen $d_1 = p$ und $d_2 = q$ haben wir das oben nachgerechnet, wobei offensichtlich nur eine Rolle spielte, daß p und q teilerfremd sind, nicht aber, daß sie Primzahlen sind. Der allgemeine Fall folgt durch vollständige Induktion, denn auch $d_1 \cdots d_s$ und $d_1 \cdots d_s d_{s+1}$ sind teilerfremd. ∎

e) Kleine öffentliche Exponenten

Da der Verschlüsselungsaufwand bei RSA linear in der Ziffernzahl des Exponenten steigt, wird gelegentlich vorgeschlagen, einen kleinen Verschlüsselungsexponenten e zu verwenden; populär sind etwa $e = 3$ und $e = 2^{16} + 1$.

Im allgemeinen ist das unproblematisch, insbesondere wenn es um den öffentlichen Exponenten zur Überprüfung einer digitalen Unterschrift geht, in einem Fall kann es allerdings Probleme geben: Wenn nämlich dieselbe Nachricht (oder derselbe Nachrichtenteil, wie z.B. eine Anlage oder ein Block ASCII-Kunst am Ende) an mehrere Empfänger geht.

Nehmen wir etwa an, die Nachricht m solle an drei Empfänger geschickt werden, deren öffentliche Schlüssel $(N_1, 3)$, $(N_2, 3)$ und $(N_3, 3)$ seien. Verschickt werden also die drei Blöcke

$$m^3 \bmod N_1, \quad m^3 \bmod N_2 \quad \text{und} \quad m^3 \bmod N_3\,.$$

Ein Gegner, der alle drei abfängt, kann dann nach dem chinesischen Restesatz

$$m^3 \bmod N_1 N_2 N_3$$

berechnen, und da m kleiner als jedes N_i sein muß, kennt er somit m^3. Die Berechnung der Kubikwurzel auch einer sehr großen Zahl ist vom Aufwand her mit einer Division vergleichbar, liegt also höchstens im unteren Sekundenbereich.

(Falls N_1, N_2, N_3 nicht paarweise teilerfremd sein sollten, merkt man das bei der Anwendung des erweiterten EUKLIDischen Algorithmus

und hat dann sogar eine Faktorisierung von mindestens zwei Moduln, was dann über die privaten Exponenten insbesondere auf die Nachricht führt.)

f) Kleine private Exponenten

Da elektronische Unterschriften häufig mit Smartcards oder in Zukunft vielleicht auch Mobiltelephonen und ähnlichen Geräte mit vergleichsweise schwacher Rechenleistung erzeugt werden, bietet sich an, nicht den öffentlichen, sondern den privaten Exponenten möglichst klein zu wählen. Ein privater Exponent drei wäre natürlich unmöglich, denn der private Exponent ist schließlich geheim und darf nicht durch systematisches Durchprobieren kleiner Zahlen gefunden werden.

Systematisches Durchprobieren ist aber, wie wir bei der Diskussion symmetrischer Kryptoverfahren gesehen haben, mit heutiger Technologie nur bis zu etwa 2^{64} Fällen möglich; 2^{128} Möglichkeiten, wie sie etwa der AES bietet, gelten heute als sicher. Ein privater Exponent mit 128 statt 1024 Bit führt zu einer Reduktion des Rechenaufwands um den Faktor acht, was gerade bei Smartcards spürbar sein sollte.

Leider gilt aber auch hier, wie in so vielen Fällen bei RSA, das Rechenerleichterungen zu Sicherheitsmängeln führen; ein privater Exponent mit 128 Bit würde bei einem Modul von 1024 oder 2048 Bit innerhalb von Sekunden zu dessen Primfaktorzerlegung führen.

Der Grund ist folgender: Der öffentliche Exponent e und der private Exponent d erfüllen die Gleichung

$$de - k(p-1)(q-1) = 1$$

mit einer natürlichen Zahl k. Division durch $d(p-1)(q-1)$ macht daraus

$$\frac{e}{(p-1)(q-1)} - \frac{k}{d} = \frac{1}{d(p-1)(q-1)} .$$

Der linke Bruch hat einen Nenner, der ungefähr die gleiche Größenprdnung hat wie der Moduk $N = pq$; davon subtrahiert wird ein Bruch mit Nenner d, und die rechte Seite der Gleichung sagt uns, daß die Differenz sehr klein ist.

Ist also der private Exponent d klein, so kann der linksstehende Bruch durch einen Bruch mit sehr viel kleinerem Nenner sehr gut approximiert werden. Damit kann ein Gegner noch nichts anfangen, denn er kennt den Nenner $(p-1)(q-1)$ nicht; andererseits kennt er $N = pq$, und die Differenz ändert sich nicht sehr, falls man den Nenner durch N ersetzt:

$$\begin{aligned}\left|\frac{e}{N} - \frac{k}{d}\right| &= \left|\frac{e}{N} - \frac{e}{(p-1)(q-1)} + \frac{e}{(p-1)(q-1)} - \frac{k}{d}\right| \\ &\le \left|\frac{e(p-1)(q-1) - epq}{N(p-1)(q-1)}\right| + \frac{1}{d(p-1)(q-1)} \\ &= \frac{e(p+q-1)}{N(p-1)(q-1)} + \frac{1}{d(p-1)(q-1)} .\end{aligned}$$

Da p und q in der Größenordnung von $\sqrt{N}$ liegen, ist auch das noch eine recht kleine Zahl.

Bei kleinem d kann sich das ein Gegner mittels des folgenden Satzes zunutze machen:

Satz: Für die reelle Zahl $x > 0$ gebe es teilerfremde natürliche Zahlen a, b derart, daß

$$\left|x - \frac{a}{b}\right| < \frac{1}{2b^2} .$$

Dann ist a/b eine Konvergente der Kettenbruchentwicklung von x.

Um damit etwas anfangen zu können, müssen wir zunächst wissen, was die Kettenbruchentwicklung einer reellen Zahl ist. Diese berechnet sich nach folgendem Algorithmus:

1. Schritt: Setze $x_0 = x$ und $a_0 = [x]$.

n-ter Schritt, $n \ge 1$: Falls $x_{n-1} = a_{n-1}$ bricht der Algorithmus an dieser Stelle ab; andernfalls setze

$$x_n = \frac{1}{x_{n-1} - a_{n-1}} \quad \text{und} \quad a_n = [x_n] .$$

Die n-te Konvergente dieser Kettenbruchentwicklung ist die rationale Zahl

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{\cdots}{a_{n-1} + \cfrac{1}{a_n}}}}}.$$

Der Beweis des obigen Satzes ist elementar, aber langwierig; wir wollen daher darauf verzichten.

Falls man diesen Satz anwenden kann, läßt sich d also bestimmen, indem man die Nenner der Konvergenten der Kettenbruchentwicklung von e/N bestimmt und jeweils durch Ausprobieren nachprüft, ob für einen Zufallsblock a die gewünschte Beziehung $a^{de} \equiv a \bmod N$ gilt.

Eine einfache Abschätzung zeigt, daß er für p und q von etwa gleicher Größe anwendbar ist, sofern d höchstens die Größenordnung von etwa $\sqrt[4]{N}$ hat; neuere, etwas aufwendigere Untersuchungen zeigen, daß auch man d auch noch für $d < N^{0{,}289}$ rekonstruieren kann; Fachleute erwarten, daß möglicherweise sogar alle $d < \sqrt{N}$ unsicher sind.

Private Exponenten müssen also immer groß sein. Falls man von einem vorgegebenen öffentlichen Exponenten ausgeht, ist das für realistische N mit an Sicherheit grenzender Wahrscheinlichkeit erfüllt; Vorsicht ist nur geboten, wenn man mit dem privaten Exponenten startet.

§ 7: RSA in der Praxis

Die Multiplikation zweier Langzahlen mit 1024 oder gar 2048 Bit ist zwar für einen Computer kein großes Problem, verglichen mit symmetrischen Kryptoverfahren sind die erreichbaren Datenraten aber um mehr als einen Faktor Tausend geringer. RSA wird daher nie zur Übertragung längerer Nachrichten eingesetzt: Man verwendet das Verfahren gelegentlich, wenn kurze Nachrichten zu übertragen sind, die in einen oder einige wenige Blöcke passen, ansonsten benutzt man, falls man

auf die Vorteile asymmetrischer Kryptographie nicht verzichten kann oder will, hybride Verfahren. Dabei wird RSA nur benutzt zur Übertragung eines Schlüssels sowie, falls möglich, zur Authentizierung der Kommunikationspartner; die eigentliche Nachricht wird dann mit einem symmetrischen Verfahren übertragen. Wir wollen uns in diesem Paragraphen einige gebräuchliche Methoden anschauen und insbesondere auch zusätzliche Sicherheitsprobleme diskutieren, die eventuell durch Standards für die Anwendung von RSA in einer solchen Situation auftreten können.

a) SSL & Co

Der Prototyp für die Anwendung hybrider Verfahren im Internet ist der Standard SSL *(Secure Sockets Layer)* sowie sein Nachfolger TLS *(Transport Layer Security)*. Er wurde zwar in erster Linie für `http` konzipiert, kann jedoch auch mit `ftp`, `telnet` und ähnlichen Diensten angewandt werden.

Allen Verbindungen gemeinsam ist, daß sich die Partner zunächst auf Verschlüsselungsverfahren einigen müssen. Hier machen sich immer noch alte amerikanische Exportbeschränkungen bemerkbar: Bis September 1998 galt alle Kryptographie als Munition die nur mit Einzelgenehmigung ins Ausland verkauft werden durfte. Diese Genehmigung wurde in der Praxis nur erteilt, wenn bei symmetrischer Kryptographie die Schlüssellänge höchstens gleich vierzig war und bei asymmetrischer Kryptographie zumindest für die Verschlüsselung mit ähnlich schwachen Algorithmen gearbeitet wurde. (Für reine Authentisierung durften auch starke Algorithmen exportiert werden.)

Da die beiden damals am meisten verbreiteten Browser, Netscape und Windows Explorer beide aus USA kamen, unterstützten diese zumindest in ihren Exportversionen daher nur schwache Kryptographie. Auch bei den Servern von Netscape waren Versionen mit starker Kryptographie (die natürlich nur innerhalb der USA verkauft werden durften) deutlich teurer als die Exportversionen, so daß viele Unternehmen noch heute nur Server mit schwacher Kryptographie unterhalten.

Nimmt ein Client Kontakt auf zu einem Server, teilt er ihm daher zunächst einmal mit, welche Kryptoverfahren er kennt. Dazu gibt es