RSA & Co. in der Schule

Moderne Kryptologie, alte Mathematik, raffinierte Protokolle

Neue Folge – Teil 5: Der Miller-Rabin-Primzahltest oder:
Falltüren für RSA mit Primzahlen aus Monte Carlo

von Helmut Witten und Ralph-Hardo Schulz

Der Fürst der Mathematiker

Johann Carl Friedrich Gauß starb am 23. Februar 1855 in Göttingen. Bereits 1856 ließ Georg V., der zweite Herzog von Cumberland und Teviotdale und letzte König von Hannover, Gedenkmünzen mit dem Bildnis von Gauß und der Aufschrift *Mathematicorum Principi* prägen („dem Fürsten der Mathematiker", in der englischsprachigen Literatur wird diese Ehrenbezeichnung häufig mit „the Prince of Mathematicians" übersetzt).

Das hätte Gauß vermutlich gefallen, da er Zeit seines Lebens überzeugter Monarchist war. Im Gegensatz zu seinem Freund, dem Physiker Wilhelm Eduard Weber, sowie den Brüdern Grimm gehörte Gauß auch nicht zu den „Göttinger Sieben", die 1837 gegen die Aufhebung der relativ liberalen Verfassung durch den neuen, erzreaktionären hannöverschen König Ernst August I. (dem Vater Georgs V.) protestierten und die daraufhin ihres Amtes enthoben wurden.

Ernst August I. (1771–1851) kann heute noch auf dem Bahnhofsvorplatz von Hannover bewundert werden, dort steht sein Reiterdenkmal mit der schönen Inschrift „Dem Landesvater sein treues Volk" (siehe Bild 2). Nur während des Baus der U-Bahn musste er auf den „Platz der Göttinger Sieben" ausweichen, was ihm vermutlich gegen den Strich gegangen wäre.

Sein Nachfahre Ernst August (geb. 1954), Prinz von Hannover, das aktuelle Oberhaupt der Welfen, machte in den letzten Jahren übrigens in einer Weise von sich reden, die die Zahl der Monarchisten in Deutschland sicherlich nicht erhöht hat (vgl. Wikipedia, Stichwort „Ernst August von Hannover …").

Gauß gilt als Begründer der modernen Zahlentheorie, ihm wird – passend zu seinen monarchistischen Überzeugungen – der Ausspruch zugeschrieben, dass die Mathematik die Königin der Wissenschaften, die Zahlentheorie aber die Königin der Mathematik sei. In seiner Promotionsschrift *Disquisitiones Arithmeticae* (auf Deutsch „Zahlentheoretische Untersuchungen", gedruckt in Leipzig 1801 und seinem Förderer Karl Wilhelm Ferdinand, dem Fürsten von Braunschweig-Wolfenbüttel, gewidmet) stellte er sinngemäß fest, dass die Untersuchung des Problems, ob eine Zahl prim oder zusammengesetzt sei, sowie die Methoden zur Zerlegung zusammengesetzter Zahlen in ihre Primfaktoren zu den wichtigsten und nützlichsten in der Zahlentheorie gehörten.

Wir sind Gauß bereits in der letzten Folge dieser Artikelserie begegnet, als wir der Frage nachgegangen sind, ob es überhaupt genügend viele Primzahlen für die massenhafte Anwendung des RSA-Verfahrens im Internet gibt: Die Antwort „Ja" auf diese Frage ergibt sich aus dem Gauß'schen Prim-

http://de.wikipedia.org/wiki/Deutsche_Mark

Bild 1: Nicht nur Münzen waren Johann Carl Friedrich Gauß gewidmet, auch Geldscheine – hier die Vorderseite eines 10-DM-Scheins, der am 16. April 1991 erstmals herausgegeben und am 1. Januar 2002 durch den Euro ersetzt wurde.

http://de.wikipedia.org/wiki/Ernst-August-Denkmal

Bild 2: Ernst August I. – Landesvater des Fürsten der Mathematiker J. C. F. Gauß.

zahlsatz (vgl. Witten/Schulz, 2010, S. 98 f.). Offen blieb die Frage, wie man effizient Primzahlen mit mehreren hundert Stellen finden kann.

Wie lang müssen RSA-Schlüssel sein?

Wenn man von einigen wenigen praktischen Errungenschaften wie z. B. der Gauß'schen Osterformel absieht, war die Zahlentheorie immer das Musterbeispiel der „reinen" Mathematik, die nicht durch Anwendungen „befleckt" war – ein sehr exklusives intellektuelles Abenteuer. Das hat sich mit dem Vordringen der Computer seit der Mitte des letzten Jahrhunderts gründlich gewandelt. Neben der reinen Zahlentheorie gibt es inzwischen einen kräftigen angewandten Zweig dieser Wissenschaft: die algorithmische Zahlentheorie, die hauptsächlich für die Kryptologie wichtig wurde. Aber auch für Fragen der klassischen Zahlentheorie – wie die Suche nach Primzahlrekorden oder der Bestimmung von Nullstellen der Riemann'schen Zeta-Funktion – haben sich mit der Kraft der elektronischen Rechenknechte neue Dimensionen aufgetan (siehe Kasten „Vollkommene Zahlen, Marin Mersenne und die Primzahlrekorde", Seite 95 f.).

Die Wurzeln der algorithmischen Zahlentheorie gehen bis ins Altertum zurück. Der euklidische Algorithmus gilt als das älteste nichttriviale Rechenverfahren, das z. B. für die Schlüsselerzeugung beim RSA-Verfahren unverzichtbar ist (vgl. Witten/Schulz, 2006b, S. 52 f.). In den letzten Jahrzehnten wurden aber auch völlig neue Algorithmen entwickelt, die in Sekundenbruchteilen feststellen können, ob eine Zahl mit mehreren hundert Stellen zusammengesetzt oder prim ist. Ein solcher revolutionär neuer Algorithmus soll im Zentrum dieses Artikels stehen: der *Miller-Rabin-Primzahltest*.

In diesem Zusammenhang kommt ein Zweig der theoretische Informatik ins Spiel, der sich mit der Komplexität von Algorithmen beschäftigt. Beim RSA-Verfahren werden bekanntlich Primzahlen benötigt, deren Produkt sich nur mit riesigem Aufwand wieder in die Primfaktoren zerlegen lässt. Das bedeutet, dass die Faktorisierung ein algorithmisch schweres Problem sein muss. Wie schwer dieses Problem ist, hängt offensichtlich von der Größe der Zahl ab. 55 kann jedes Grundschulkind in seine Primfaktoren 5 und 11 zerlegen, aber wie sieht es z. B. mit 8616460799 aus? (*Anmerkung:* Wenn Sie 8616460799 bei Google eingeben, erhalten Sie Hinweise zur Geschichte, Bedeutung und Faktorisierung dieser Zahl.) Im Artikel *Zeit-Experimente zur Faktorisierung* werden Anregungen gegeben, wie man sich diesem Problem durch eigene Versuche mit freier Software wie *SAGE* oder *CrypTool* nähern kann (vgl. Schulz/Witten, 2010, S. 107 ff. in diesem Heft).

Empfehlungen zur Schlüssellänge bei RSA hängen sehr stark vom Zeitpunkt ihrer Veröffentlichung ab. Im Buch *Datensicherheit mit PGP* des renommierten Experten William Stallings aus dem Jahr 1995 werden Schlüssel der Länge 384 Bit für den Normalgebrauch, 512 Bit für kommerzielle Zwecke und 1024 Bit für militärische Sicherheit vorgeschlagen. In der letzten Folge unserer Artikelserie haben wir berichtet, dass nach dem aktuellen Stand der Forschung nur Schlüssel, die eine Länge von 2048 Bit haben, auch in den Jahren ab 2014 bei der Anwendung im RSA-Verfahren ausreichende Sicherheit bieten (vgl. Witten/Schulz, 2010, S. 98).

RSA-Schlüssel enthalten als wesentlichen Bestandteil ein Produkt, das aus zwei verschiedenen Primzahlen in der gleichen Größenordnung gebildet wird (vgl. z. B. Witten/Schulz, 2006b). Um sichere 2048-Bit-Schlüssel zu erhalten, braucht man also zwei 1024-Bit-Primzahlen, die im Dezimalsystem jeweils ca. 310 Stellen haben.

Wie findet man solche riesigen Primzahlen mit über 300 Stellen? Die Zahlen aus der Liste der Primzahlrekorde, die großen Mersenne-Primzahlen, scheiden dafür aus. Für das RSA-System sind nämlich Mersenne-Primzahlen schon deshalb uninteressant, weil es so wenige davon gibt (zurzeit weniger als 50, siehe Kasten „Vollkommene Zahlen, Marin Mersenne und die Primzahlrekorde", Seite 95 f.). Bei dieser sehr übersichtlichen Anzahl könnte man einen RSA-Modul aus Mersenne-Primzahlen, der ja als Semiprimzahl ein Produkt von zwei Primzahlen sein muss, durch einfache Probedivision zerlegen.

Zum Auffinden von Primzahlen in allgemeiner Form stehen zahlreiche Verfahren zur Verfügung. Das im Kasten „Primzahlrekorde" erwähnte Überblickskapitel aus dem Buch von Ribenboim (2006) umfasst über hundert Seiten. Wir wollen uns im Folgenden daher auf diejenigen Verfahren beschränken, die für die asymmetrische Kryptografie im Allgemeinen und für RSA im Besonderen relevant sind.

Probedivision als Primzahltest

Die einfachste Methode, um Primzahlen zu finden, ist die Probedivision, auch *brute force* (auf Deutsch „rohe Gewalt") genannt. Man teilt die zu untersuchende natürliche Zahl n der Reihe nach durch die Zahlen von 2 bis n–1; wenn keine der Divisionen „aufgeht" (d. h. den Rest 0 lässt), ist die Zahl offenbar eine Primzahl. Man sieht leicht ein, dass sich dieses Verfahren stark vereinfachen lässt. Es reicht nämlich, dass man die Probedivision nur mit den Primzahlen $p \leq \sqrt{n}$ durchführt.

Wir interessieren uns für die Frage, ob sich dieses elementare Verfahren prinzipiell zum Nachweis der Primalität von 1024-Bit-Zahlen eignet, d. h. von Zahlen n mit $2^{1023} \leq n < 2^{1024}$. Dazu berechnen wir mit dem Gauß'schen Primzahlsatz (vgl. Witten/Schulz, 2010, S. 98 ff.) die ungefähre Anzahl der Primzahlen, mit denen die Probedivision durchgeführt werden müsste, falls die Zahl wirklich prim sein sollte (sonst ist man schon früher fertig – es sei denn, dass die fragliche Zahl das Quadrat einer Primzahl ist). Zur Erinnerung: Mit dem Gauß'schen Primzahlsatz lässt sich die Anzahl der Primzahlen $p \leq x$ ziemlich gut durch $\pi(x) \approx x/\ln x$ abschätzen.

Die Anzahl der Primzahlen kleiner oder gleich $\sqrt{n}$ – d.h. die Primzahlen kleiner als $(2^{1024})^{1/2} = 2^{512} \approx 1{,}34 \cdot 10^{154}$ – berechnen wir nach dieser Formel zu $2^{512}/\ln 2^{512} = 2^{512}/(512 \ln 2) \approx 3{,}8 \cdot 10^{151}$.
Wir erinnern daran, dass die Zahl aller Elementarteilchen im gesamten Universum auf weniger als 10^{80} geschätzt wird, sodass man diese Zahlen schon aus physikalischen Gründen nicht speichern kann. Es kann also keine „Liste" geben, in der die Primzahlen für die Probedivisionen aufgeführt wären. Und selbst wenn dies möglich wäre, ist die Zahl der benötigten Divisionen so hoch, dass auch mit den schnellsten Computern der Welt so viel Rechenzeit benötigt würde, dass unsere Sonne inzwischen zum roten Riesen geworden und das Wasser in den Weltmeeren verdampft wäre – und das nur, um eine einzige Primzahl mit ca. 310 Dezimalstellen zu berechnen.

Wir müssen also andere Wege zum Testen der Primalität von sehr großen Zahlen suchen. Fast alle modernen Primzahltests beruhen auf dem *kleinen Satz von Fermat*, den wir in der vorletzten Folge dieser Artikelserie genauer kennen gelernt haben (vgl. Witten/Schulz, 2008, S.62). Bei all diesen Primzahltests wird allerdings zunächst eine Probedivision mit kleinen Primzahlen durchgeführt, bevor der eigentliche Test gestartet wird. Auf diese Weise können ohne großen Rechenaufwand schon sehr viele Zahlen als zusammengesetzt erkannt werden. Bei den Primzahltests von *CrypTool* werden z.B. die zu testenden Zahlen zunächst mit allen Primzahlen $p \leq 37$ auf Zerlegbarkeit geprüft.

Der kleine Satz von Fermat als Basis von Primzahltests

Wir erinnern uns an den kleinen Satz von Fermat (siehe Kasten „Der Fermat-Test, Pseudoprimzahlen und Carmichael-Zahlen", Seite 98):

> Wenn p eine Primzahl ist,
> dann ist $a^p - a$ für alle a durch p teilbar.

Bekannter ist folgende Formulierung mit einer Kongruenz:

> Wenn p eine Primzahl ist, gilt für alle
> $a \in \{2, \ldots, p-1\}$ die Kongruenz $a^{p-1} \equiv 1 \pmod{p}$.

Aus der Verneinung (Kontraposition) dieses Satzes erhält man ein wichtiges Kriterium für die Zerlegbarkeit:

> Gilt $a^{n-1} \not\equiv 1 \pmod{n}$ für mindestens *ein* $a \in \{2, \ldots, n-1\}$, so ist n *garantiert zusammengesetzt*.

Findet man ein solches a, hat man einen *Fermat-Zeugen* für die Zerlegbarkeit von n gefunden. Man kann sich leicht überlegen, dass aus der Gültigkeit der Kongruenz $a^{n-1} \equiv 1 \pmod{n}$ für *alle* $a \in \{2, \ldots, n-1\}$ folgt, dass n prim ist (vgl. z.B. Karpfinger/Kiechle, 2010, S.145). Das bedeutet, dass der kleine Satz von Fermat umkehrbar ist; man könnte ihn als Äquivalenz formulieren. Für praktische Zwecke nutzt das leider überhaupt nichts, weil dieser „Test" noch aufwendiger wäre als die Probedivision. Das Kriterium für die Zerlegbarkeit, das wir aus der Kontraposition des Satzes von Fermat gewonnen haben, wird sich dagegen als sehr nützlich erweisen.

Bevor wir überlegen, wie wir einen Primzahltest auf der Basis des kleinen Satzes von Fermat gewinnen können, wollen wir einige Computerexperimente durchführen, um Erfahrungen mit dem modularen Potenzieren zu sammeln und um auf diese Weise zu prüfen, ob die untersuchten Zahlen zusammengesetzt oder (wahrscheinlich) prim sind. Für diese Übungen können wir auch kleinere Primzahlen verwenden, sodass zur Kontrolle zahlreiche Primzahltabellen aus dem Internet zur Verfügung stehen, die man sich mit einem Programm nach dem Siebverfahren von Eratosthenes auch leicht selber erzeugen kann (vgl. Witten/Schulz, 2006a, S.52).

Computer-Experimente zum kleinen Satz von Fermat

Zur Berechnung der modularen Potenzen steht den Lernenden z.B. der PYTHON-Interpreter mit eingebauter Langzahlarithmetik zur Verfügung, man kann aber auch für JAVA die Klasse BigInteger verwenden. Alle CAS-Systeme bieten ebenfalls die Möglichkeit, modulare Potenzen einfach zu berechnen; wir werden dies im Folgenden am Beispiel von PYTHON demonstrieren. Diese Beispiele lassen sich auch sehr einfach mit dem Web-Interface des Open-Source-CAS-Systems *SAGE* nachvollziehen, da dort die Syntax von PYTHON verwendet wird. (*Anmerkung:* Damit die Syntax von *SAGE* und PYTHON weitgehend identisch bleibt, verwenden wir weiterhin die aktuellste Version 2.x von PYTHON und warten mit dem Umstieg auf die Version 3.x, bis auch *SAGE* diesen Schritt vollzogen haben wird.)

Wir haben oben die Zahl 8616460799 erwähnt. Sie haben sich wahrscheinlich inzwischen informiert, dass es sich dabei um eine zusammengesetzte Zahl handelt, die *Jevons Zahl* genannt wird (vgl. z.B. http://www.numericana.com/answer/factoring.htm#jevons oder Schulz/Witten, 2010, Seite 107ff. in diesem Heft). Bei einem Primzahltest geht es aber nicht um die Frage nach den Faktoren einer zusammengesetzten Zahl, man will nur wissen, ob eine Zahl prim ist oder nicht. Nach dem o.a. Kriterium für die Zerlegbarkeit reicht es daher, wenn wir eine Zahl a im Bereich $1 < a < 8616460799$ (bzw. einen Fermat-Zeugen) finden, sodass der Rest bei der Division von $a^{8616460799-1}$ durch 8616460799 ungleich 1 ist. Am einfachsten ist es offenbar, wenn wir unsere Versuche mit $a = 2$ starten.

Nun wäre es nicht sehr geschickt, wenn man zunächst $2^{8616460799-1}$ berechnen und dann den Rest bezüglich 8616460799 bilden würde. Besser ist es, das Po-

Vollkommene Zahlen, Marin Mersenne und die Primzahlrekorde

Am 12. Oktober 2009 meldete die US-amerikanische *Electronic Frontier Foundation* (EFF), die sich um die Bürgerrechte im Cyberspace kümmert, dass *mersenne .org* einen Preis von 100000 US-$ durch das Aufspüren einer Primzahl mit über 12 Millionen Dezimalstellen gewonnen habe:

http://www.eff.org/press/archives/2009/10/14-0

Was hat es mit dieser unvorstellbar großen Rekord-Primzahl auf sich? Warum werden solche riesigen Zahlen gesucht? Was verbirgt sich hinter *mersenne.org*? Was sind vollkommene Zahlen? Wer war Marin Mersenne? Was hat dies mit den Primzahlen für RSA zu tun?

Um auf die letzte Frage zuerst zu antworten: Diese riesigen Primzahlen sind für das RSA-Kryptosystem nicht relevant. Aber die anderen Fragen scheinen uns interessant genug, um in diesem Kasten eine kurze Antwort darauf zu geben.

Da es unendlich viele Primzahlen gibt, gibt es keine größte unter ihnen. Mit den Primzahlrekorden meint man daher die zu bestimmten Zeitpunkten größten *bekannten* Primzahlen – eine aktuelle Liste findet man z. B. unter

http://www.primzahlen.de/primzahlen/top-listen.htm

In dem bei Primzahlfreunden sehr geschätzten Buch von Paulo Ribenboim (2006) werden im Kapitel 2 mit der Überschrift *Wie kann man Primzahlen erkennen?* auf Seite 122 ff. „titanische Primzahlen" (mindestens 1000 Dezimalstellen), „gigantische Primzahlen" (mindestens 10000 Dezimalstellen) sowie „Megaprimzahlen" (über 1000000 Dezimalstellen) erwähnt.

Wer sich für solche riesigen Primzahlen und weitere Informationen dazu interessiert, dem empfiehlt Ribenboim einen Besuch von „The Prime Page" (http://primes.utm.edu/), dem „Guiness Buch der Primzahlrekorde".

Bei den Primzahlrekorden sucht man meist unter den Zahlen in der speziellen Form 2^n-1, den sogenannten *Mersenne-Zahlen*. Primzahlen, die diese Form haben, werden *Mersenne-Primzahlen* genannt. Diese Zahlen haben ihren Namen zu Ehren des Mönchs, Theologen, Mathematikers und Musiktheoretikers Pater Marin Mersenne (1588–1648, siehe Bild) bekommen.

LOG-IN-Archiv

Pater Marin Mersenne (1588–1648), Vorkämpfer der modernen Naturwissenschaften und Namensgeber der Mersenne-Primzahlen.

So große Verdienste sich Mersenne als Vorreiter der Aufklärung und Vorkämpfer der modernen Naturwissenschaften erworben hat, bei den später sogenannten Mersenne-Primzahlen ist er nur durch eine Liste mit Primzahlen der Form 2^p-1 bekannt geworden, die sich als fehlerhaft und – unterhalb der von Mersenne gewählten Grenze von 2^{257} – auch als unvollständig herausgestellt hat (vgl. auch *Neuer Primzahlrekord* in LOG IN 128/129 (2004), S. 116 f.).

Man kann zeigen, dass bei Mersenne-Primzahlen der Exponent selbst eine Primzahl sein muss: Wenn der Exponent zusammengesetzt ist, gilt dies auch für die Mersenne-Zahl. Dass die Umkehrung dieser Aussage nicht gilt, wusste man bereits zu Beginn der Neuzeit: $2^{11}-1 = 2047 = 23 \cdot 89$ ist die kleinste Mersenne-Zahl mit einem Primzahl-Exponenten, die selbst nicht prim ist.

Zu jeder Mersenne-Primzahl 2^p-1 lässt sich eine vollkommene Zahl nach der Formel $2^{p-1}(2^p-1)$ berechnen. Dieser Zusammenhang war schon Euklid bekannt. *Vollkommene Zahlen* (auch *perfekte Zahlen* genannt) stimmen nach ihrer – vermutlich auf die Pythagoräer zurückgehenden – Definition mit der Summe ihrer echten Teiler überein. So erhält man mit den o. a. Formeln für $p = 2$ die erste Mersenne-Primzahl 3 und die kleinste vollkommene Zahl 6, für die in der Tat $6 = 1+2+3$ gilt.

Vollkommene Zahlen hatten im Altertum eine mythische Bedeutung, so wurde die Welt nach der Schöpfungsgeschichte im Alten Testament in 6 Tagen erschaffen. Die nächste Mersenne-Primzahl 7 ergibt sich für $p = 3$, die nach der o. a. Formel zugehörige Zahl 28 ist ebenfalls vollkommen: $28 = 1+2+4+7+14$. Die nächsten beiden vollkommenen Zahlen sind 496 (für $p = 5$) und 8128 (für $p = 7$); mehr waren im Altertum nicht bekannt. Im Moment kennt man 47 Mersenne-Primzahlen (und damit auch 47 vollkommene Zahlen); durch das GIMPS-Projekt (s. u.) kommt pro Jahr etwa eine neue dazu.

Leonhard Euler hat bewiesen, dass sich auch jede gerade vollkommene Zahl in der Form $2^{p-1}(2^p-1)$ – mit 2^p-1 prim – darstellen lässt. Insofern gibt es zu jeder Mersenne-Primzahl eine gerade vollkommene Zahl und umgekehrt (vgl. z. B. Ribenboim, 2006, S. 84 ff.). Man weiß bis zum heutigen Tag nicht, ob es unendlich viele gerade vollkommene Zahlen (bzw. Mersenne-Primzahlen) gibt. Ein weiteres ungelöstes Problem der Zahlentheorie ist die Frage, ob es ungerade vollkommene Zahlen gibt – man kennt bislang keine einzige (vgl. Ribenboim, 2006). Auf elementarem Niveau werden solche Fragen auch im Artikel „Hardys Taxi" in dem Buch *Alles ist Zahl* (Baptist, 2008) behandelt.

Eine Besonderheit der Mersenne-Zahlen ist, dass ihre Darstellung im Dualsystem nur die Ziffer 1 enthält; Nullen kommen nicht vor. Die Zahl der Einsen entspricht dabei der Zahl im Exponenten. Für Mersenne-Zahlen gibt es einfache und effiziente Primzahltests, die die Besonderheiten dieses speziellen Zahlentyps ausnutzen. Bei fast allen Primzahlrekorden seit 1876 handelt es sich um Mersenne-Primzahlen, seit 1951 wurden alle mithilfe von Computern gefunden.

Bei dem Wettlauf nach den Primzahlrekorden ist der oben erwähnte Wettbewerb der EFF ein wesentlicher Antrieb. Das Preisgeld stammt von einem anonymen

Fortsetzung nächste Seite

Spender, der es für diesen speziellen Zweck zur Verfügung gestellt hat; die Preisvergabe wird von der EFF organisiert (vgl. http://www.eff.org/awards/coop). Der erste Preis mit 50000 US-$ wurde vor mehr als zehn Jahren für die erste Megaprimzahl mit mehr als 1 Million Stellen ebenfalls an einen Teilnehmer des GIMPS-Projekts (***G**reat **I**nternet **M**ersenne **P**rime **S**earch*, http://www.mersenne.org/) vergeben. Die Preisgelder für Primzahlen mit mehr als 100 Millionen Stellen (150000 US-$) und für solche mit mehr als 1 Milliarde Stellen (250000 US-$) sind noch offen und warten auf die Vergabe.

Über den Sinn der Preise führte John Gilmore, Mitgründer der EFF und Projektleiter dieses Wettbewerbs, aus: „The EFF awards are about cooperation. Prime numbers are important in mathematics and encryption, but the real message is that many other problems can be solved by similar methods."

Der aktuelle Rekord, und zwar die Primzahl $2^{43112609}-1$, hat 12978189 Stellen im Dezimalsystem. Würde man die Zahl ausdrucken, benötigte man bei 80 Ziffern pro Zeile und 60 Zeilen pro Seite über 2700 Seiten, die vermutlich nicht in ein einziges Buch passen würden. Allerdings werden unter

http://www.perfsci.com/wall-art.asp

Poster mit einigen der letzten Primzahlrekorde angeboten, die aber nur mit speziellen Lupen gelesen werden können.

Wer sich genauer über die erfolgreiche Suche nach Mersenne-Primzahlen informieren und sich vielleicht auch selbst daran beteiligen will, dem sei die bereits erwähnte Seite von GIMPS empfohlen. Auf dieser Seite wird u.a. Software für alle gängigen Betriebssysteme angeboten, mit der man im GIMPS-Projekt aktiv werden und eventuell einen Preis der EFF gewinnen kann.

Aufgrund der große Zahl von Freiwilligen, die sich dem GIMPS-Projekt angeschlossen haben, ist durch den Zusammenschluss unzähliger Computer über das Internet der weltweit größte Supercomputer als Graswurzelprojekt realisiert worden – und das nur, um neue Primzahlrekorde aufzustellen.

tenzieren in eine Folge von Quadrieren und Multiplizieren aufzuteilen und bei jedem Schritt modulo 8616460799 zu reduzieren, d.h. jedes Mal den Rest zu bilden, bevor man erneut quadriert oder multipliziert. Das ist nach den Regeln des modularen Rechnens erlaubt. Die Methode zur Aufspaltung des Potenzierens heißt *square and multiply* und ist schon vor über 2000 Jahren von indischen Mathematikern entdeckt worden (vgl. Witten/Schulz, 2006a, und den Nachtrag in Witten/Schulz, 2006b).

Die eingebaute PYTHON-Funktion pow (für *power*) liefert sehr schnell und effizient mit dieser Methode das gewünschte Ergebnis (der erste Parameter von pow ist die Basis, der zweite der Exponent, der dritte der Modul):

```
Python 2.7.1 [...]
>>> pow(2,8616460798,8616460799)
3816009237L
```

Hiermit wurde also $2^{8616460798}$ mod 8616460799 berechnet und auf diese Weise gezeigt, dass die Kongruenz $2^{8616460798} \equiv 3816009237 \pmod{8616460799}$ gilt, d.h.: 2 ist ein Fermat-Zeuge dafür, dass 8616460799 sicher zusammengesetzt ist. (*Anmerkung:* Das L am Ende der Zahl 3816009237L deutet an, dass PYTHON auf die (eingebaute) Langzahlarithmetik umgeschaltet hat.) Mit diesem Ergebnis erhält man im Übrigen keinerlei Hinweise auf die Faktoren $8616460799 = 89681 \cdot 96079$. (*Anmerkung:* In der Gauß'schen Schreibweise wird der Modul (hier: 8616460799) in Klammern hinter der Kongruenz geschrieben. Für die Modulo-Funktion, die den Rest bei der Division liefert, wird wie bei den Sprachen mit PASCAL-ähnlicher Syntax mod in der Operator-Schreibweise (ohne Klammern) gewählt. In den Sprachen mit C-ähnlicher Syntax wie JAVA oder PYTHON wird für den Modulo-Operator das Prozent-Zeichen % verwendet.)

Experiment 1: Der „chinesische Primzahltest"

Zunächst erhalten die Schülerinnen und Schüler die Aufgabe, den sogenannten *chinesischen Primzahltest* zu untersuchen, der sich leider als nicht zuverlässig herausstellen wird. Angeblich wurde vor zweieinhalbtausend Jahren in China ein einfaches Verfahren entdeckt, um für eine ungerade Zahl n zu überprüfen, ob sie prim ist oder nicht. Wenn nämlich $2^{n-1} \equiv 1 \pmod{n}$ gilt, sei die Zahl prim, sonst sei sie zusammengesetzt. Dieser „Forschungsauftrag" erfordert einige Mühe und Durchhaltevermögen, da es nur wenige Zahlen gibt, die die Kongruenz erfüllen, obwohl sie zusammengesetzt sind. Solche Zahlen heißen *Pseudoprimzahlen zur Basis 2* (auch kurz nur *Pseudoprimzahlen* genannt; siehe Kasten „Der Fermat-Test …", Seite 98). Zwischen 2 und 2000 gibt es davon nur sieben, die kleinste ist $341 = 11 \cdot 31$. Also ist die Aussage des chinesischen Primzahltests in Bezug auf die Primzahlen leider falsch. Nur wenn die Kongruenz *nicht* gilt, weiß man sicher, dass die untersuchte Zahl *zusammengesetzt* ist. Die Geschichte von dem angeblichen Primzahltest, der sich als Gerücht erweist, wird sehr nett von Paulo Ribenboim (2006, S. 91 ff.) richtiggestellt.

Nach dem Satz von Fermat muss die eben erwähnte Kongruenz gelten, wenn n eine Primzahl ist. Problematisch für Primzahltests sind die Pseudoprimzahlen, die mit dem Erfüllen der Kongruenz nur scheinbar prim, aber in Wirklichkeit zusammengesetzt sind. Wie wir noch sehen werden, ist es mit den heutigen Computern sehr einfach, Tabellen von Pseudoprimzahlen zu erzeugen.

Die Programmieraufgabe besteht darin, eine Liste von Pseudoprimzahlen unterhalb einer gegebenen Grenze zu erzeugen. Ein naheliegendes Vorgehen wäre, sich zunächst mit dem Programm zum Sieb des Eratosthenes (vgl. Witten/Schulz, 2006a) eine Liste von Prim-

zahlen zu erzeugen. Dann wird für *alle* Zahlen unterhalb der gegebenen Grenze geprüft, ob $2^{n-1} \equiv 1 \pmod{n}$ gilt. Falls eine dieser Zahlen auch in der Primzahlliste steht, wird sie gestrichen. Die übrigen Zahlen sind dann die Pseudoprimzahlen (zur Basis 2). Hier folgt das entsprechende PYTHON-Fragment:

```
N = 20
for n in range(2,N):
    if pow(2,n-1,n) == 1:
        print n," ist moeglicherweise pseudoprim!"
```

Die Ausgabe besteht in diesem Fall aus den Primzahlen kleiner 20. Bei größeren n-Werten erhält man zusätzlich zu den Primzahlen auch die Pseudoprimzahlen (zur Basis 2), zuerst bei 341, da $341 = 11 \cdot 31$ zusammengesetzt ist:

```
Python 2.7.1 ...
>>> pow(2,340,341)
1
```

Wenn man also aus der Liste der Zahlen, die mit dem obigen Programmfragment erzeugt wurde, die Primzahlen entfernt, erhält man die gesuchte Liste mit den Pseudoprimzahlen kleiner n. Im CAS-Paket *SAGE*, das ebenfalls PYTHON als Interface und Skriptsprache verwendet, gibt es eine eingebaute Funktion is_prime(), sodass man die sieben Pseudoprimzahlen zwischen 2 und 2000 sehr einfach durch das folgende *SAGE*/PYTHON-Skript erhält:

```
for n in range(2,2000):
    if not(is_prime(n)):
        if pow(2,n-1,n) == 1: print n
```

Experiment 2: Dase und die Zahl R_{11}

Weitere Aufgaben zum modularen Potenzieren nach dem kleinen Satz von Fermat erhält man z. B. durch die Anekdote, dass der Rechenkünstler Zacharias Dase (siehe Bild 3) nach sechs Stunden intensiven Kopfrechnens 1846 feststellte, dass R_{11} = 11111111111 (elf Einsen) keine Primzahl, sondern eine zusammengesetzte Zahl ist. Dieses Ergebnis veranlasste den Mathematiker Carl Gustav Jacob Jacobi (1804–1851) zu einer Veröffentlichung mit dem schönen Titel *Untersuchung, ob die Zahl 11111111111 eine Primzahl ist oder nicht – Ein Kuriosum, veranlasst durch Dase* (vgl. auch Baumann, 2009, S. 118). Nach Jacobi benutzte Dase den kleinen Satz von Fermat, um die Zerlegbarkeit der Zahl mit den elf Einsen nachzuweisen.

Das Ergebnis lässt sich heute mithilfe eines Computers in Sekunden nachvollziehen: pow(2,11111111110, 11111111111) liefert 1496324899 und bestätigt damit, dass die Zahl keine Primzahl sein kann. Solche Zahlen aus lauter Einsen heißen *Repunit* (ein Kunstwort aus dem Englischen *repeated units*, daher die Abkürzung R_n, wobei n die Zahl der Einsen angibt), im Deutschen manchmal auch *Einserkolonne* oder *Einserschlange* genannt. Wie schon erwähnt ist es ein Nachteil des Verfahrens nach Fermat, dass man aus dem Ergebnis keinerlei Aufschlüsse über die Faktoren erhält, es gilt nämlich R_{11} = 11111111111 = 21649 · 513239. Aber für die Lernenden wird damit die Untersuchung angeregt, ob es Repunits gibt, die Primzahlen (oder zumindest Pseudoprimzahlen) sind. Es lässt sich zeigen, dass n eine Primzahl sein muss, wenn R_n eine Repunit-Primzahl sein soll. Die Umkehrung gilt nicht, wie man an den Beispielen R_{11} oder $R_3 = 111 = 3 \cdot 37$ sieht.

Bild 3: Daguerreotypie des Rechenkünstlers Johann Martin Zacharias Dase (1824–1861).

Daguerreotypie von C. F. Stelzner, um 1845 (Museum für Kunst und Gewerbe, Hamburg)

Experiment 3: Pseudoprimzahlen und Carmichael-Zahlen

Wegen der Existenz von Pseudoprimzahlen mit der Basis 2 ist es naheliegend, auch andere Basen zu untersuchen. Man sieht sofort, dass pow(3,340,341) das Ergebnis 56 liefert, womit auch auf diesem Weg bewiesen ist, dass 341 zwar Pseudoprimzahl (zur Basis 2, s. o.), aber keine Primzahl ist, da hier 3 ein Fermat-Zeuge ist.

Damit drängt sich die Frage auf, ob es auch Pseudoprimzahlen zur Basis 3, 5, 7, … gibt. Man spricht in diesem Fall von 3-Pseudoprimzahlen, 5-Pseudoprimzahlen usw. Wenn keine Basis angegeben wird, ist stets die Basis 2 gemeint. Durch Vergleich mit einer Primzahltabelle finden die Schülerinnen und Schüler solche Beispiele. Bei der Basis 3 werden sie schon bei $91 = 7 \cdot 13$ fündig, da pow(3,90,91) das Ergebnis 1 liefert.

Für die Suche nach weiteren Pseudoprimzahlen bietet es sich an, wiederum ein kleines Programm zu schreiben (s. o.). Im Internet findet man z. B. unter

http://www.mathe-schule.de/download/pdf/Primzahl/PSP.pdf

eine umfängliche Sammlung von Pseudoprimzahlen zu den Basen 2, 3, 5 und 7 sowie die Wahrscheinlichkeiten, beim Testen mit diesen Basen, eine zusammengesetzte Zahl irrtümlich als Primzahl zu erhalten – dies allerdings nur bis zur Obergrenze $25 \cdot 10^9$, was für unsere Zwecke (Testen von Zahlen mit 1024 Bit oder ca. 310 Dezimalstellen) leider viel zu klein ist.

Aus der erwähnten Tabelle kann man entnehmen, dass es auch zusammengesetzte Zahlen gibt, die Pseudoprimzahlen zu mehreren verschiedenen Basen sind. So ist $29341 = 13 \cdot 37 \cdot 61$ Pseudoprimzahl zu den Basen 2, 3, 5 und 7.

Es gibt sogar zusammengesetzte Zahlen n, die bezüglich aller zu n teilerfremden Zahlen a pseudoprim sind. Solche Zahlen werden zu Ehren des amerikanischen Mathematikers Robert Daniel Carmichael (1897–1967), der uns bereits in der vorletzten Folge dieser Artikelserie im Zusammenhang mit der Carmichael-Funktion und dem Satz von Carmichael begegnet ist,

Der Fermat-Test, Pseudoprimzahlen und Carmichael-Zahlen

Der *Fermat-Test* dient der Untersuchung, ob eine Zahl n eine Primzahl ist oder nicht. In jeder Runde dieses Tests wird eine *Basis a* (für eine natürliche Zahl a mit $1 < a < n$) ausgewählt und $a^{n-1} \pmod n$ berechnet. Gilt

$a^{n-1} \equiv 1 \pmod n$,

so sagt man: „n hat den Fermat-Test zur Basis a bestanden." n heißt dann *Pseudoprimzahl zur Basis a* (oder Primzahl, wenn n prim sein sollte).
In jeder folgenden Runde des Tests wird eine weitere Zahl als Basis gewählt. Nach dem Satz von Fermat (vgl. Witten/Schulz, 2008, S. 65) besteht eine Primzahl garantiert jede dieser Runden.
Umgekehrt liegt nur mit einer gewissen Wahrscheinlichkeit in n eine Primzahl vor, wenn n die Tests besteht. Denn es gibt Zahlen, die den Fermat-Test bestehen, ohne dass sie Primzahlen sind. 341 ist so ein *Beispiel* zur Basis 2.
Findet man hingegen ein a_1 mit $a_1^{n-1} \not\equiv 1 \pmod n$, so hat man einen sogenannten *Fermat-Zeugen* a_1 identifiziert; aus der Kontraposition des Satzes von Fermat folgt in diesem Fall, dass n mit Sicherheit zusammengesetzt ist.
Falls eine natürlich Zahl n den Fermat-Test für jede zu ihr teilerfremden Basis a besteht, ohne Primzahl zu sein, so heißt sie *Carmichael-Zahl*; für eine solche Zahl n gilt also $a^{n-1} \equiv 1 \pmod n$ für alle zu n teilerfremden ganzen Zahlen a. Carmichael-Zahlen sind also spezielle Pseudoprimzahlen. In einem zunächst nicht beachteten Artikel wurden 1899 die jetzt Carmichael-Zahlen genannten Zahlen von Alwin Korselt untersucht, erst 1912 unabhängig davon von Carmichael (vgl. Ribenboim, ^{3}1986, S. 118).

Ein *Beispiel* für eine Carmichael-Zahl ist $561 = 3 \cdot 11 \cdot 17$. Für diese Zahl hat nämlich die Carmichael-Funktion (vgl. Witten/Schulz, 2008) folgenden Wert: $\lambda(561) = \text{kgV}(\lambda(3), \lambda(11), \lambda(17)) = \text{kgV}(2, 10, 16) = 80$; und 80 teilt (561–1). Daraus folgt, dass 561 Carmichael-Zahl ist; denn es gilt allgemein:

Hilfssatz 1: *Für jede zusammengesetzte ungerade natürliche Zahl n sind folgende drei Aussagen äquivalent:*
(i) n ist Carmichael-Zahl.
(ii) $\lambda(n)$ *teilt* $(n-1)$.
(iii) n hat keine mehrfachen Primteiler (d.h. $n = \prod_{i=1}^{t} p_i$ *mit* $p_i \neq p_j$ *für* $i \neq j$*), und es gilt* $(p-1) \mid (n-1)$ *für jeden Primteiler p von n.*

Der Beweis dieses Satzes ist u.a. im LOG-IN-Service (siehe Seite 143) erhältlich.
Eine Folgerung aus Hilfssatz 1 ist:

Hilfssatz 2: *Jede Carmichael-Zahl n hat mindestens drei verschiedene Primteiler.*

Beweisskizze: Gemäß Definition ist n nicht prim. Nach Hilfssatz 1 ist nur auszuschließen, dass n Produkt zweier Primzahlen ist. Wäre $n = pq$ mit $p < q$, so ist einerseits $n-1 \equiv p-1 \pmod{q-1}$ wegen $pq-1 = p(q-1)+(p-1)$, andererseits $(q-1) \mid (n-1)$ nach Hilfssatz 1; es folgte $(q-1) \mid (p-1)$ im Widerspruch zu $p < q$.

In Anbetracht der Existenz von unendlich vielen Carmichael-Zahlen (nach Alford, Granville und Pomerance, vgl. Ribenboim, ^{3}1986) und somit der Zerschlagung der Hoffnung auf eine (endliche) Tabelle dieser Zahlen, ist ein Verfeinerung des Fermat-Tests gesucht; eine solche liefert z.B. der Primzahltest von Miller-Rabin.

Carmichael-Zahlen genannt (siehe Kasten „Der Fermat-Test, Pseudoprimzahlen und Carmichael-Zahlen").

Carmichael-Zahlen n bestehen den Fermat-Test mit *jeder* zu n teilerfremden Basis a, obwohl sie zusammengesetzt sind. Die kleinste Carmichael-Zahl ist $561 = 3 \cdot 11 \cdot 17$. Die Teiler dieser Zahl sind daher 3, 11, 17, 33, 51 und 187. So liefert z.B. `pow(50,560,561)` den Wert `1` (pseudoprim zur Basis 50), aber `pow(51,560,561)` liefert den Wert `408`, also ist die Zahl zusammengesetzt. Die Lernenden erhalten die Aufgabe, dies auch mit den anderen nicht-teilerfremden Basen zu überprüfen.

Man kann durch modulares Potenzieren nach dem kleinen Satz von Fermat also auch Pseudoprimzahlen und sogar Carmichael-Zahlen als zusammengesetzt entlarven, wenn man zufällig eine Zahl erwischt, die *nicht teilerfremd* zu der untersuchten Zahl ist (das ist auch der tiefere Grund, warum man mit dem kleinen Satz von Fermat letztlich alle Zahlen aussortieren kann, die nicht prim sind). Dieses Aussortieren ist allerdings bei den Carmichael-Zahlen am schwierigsten. Auch ein Programm zum automatischen Auffinden von Carmichael-Zahlen ist aufwendiger als die oben erwähnten Programmbeispiele für die Suche nach Pseudoprimzahlen bei einer vorgegebenen Basis.

Der probabilistische Fermat-Test

Wir kommen damit zu einem probabilistischen Primzahltest, der nach Fermat benannt wurde (vgl Rempe/Waldecker, 2009, S. 83 f.) und z.B. in der E-Learning-Software *CrypTool* implementiert ist (siehe Bild 4, nächste Seite, und Kasten „Der Fermat-Test …"). Wir haben gesehen, dass eine vollständige Überprüfung auf Primalität einer Zahl nach dem kleinen Satz von Fermat zwar im Prinzip möglich, in der Praxis bei sehr großen Zahlen nicht durchführbar ist.

Wenn man immer mehr $a \in \{2, \ldots, n-1\}$ überprüft, so trifft man entweder auf eine Zahl a mit $a^{n-1} \not\equiv 1 \pmod n$ und weiß dann sicher, dass die Zahl zusammengesetzt ist, oder man erhält jedes Mal das Ergebnis $a^{n-1} \equiv 1 \pmod n$ und gewinnt so die Überzeugung, dass die Zahl mit einer gewissen (bei zunehmender Testrundenzahl wachsenden) Wahrscheinlichkeit prim ist.

Bild 4: Aufruf des Fermat-Tests bei *CrypTool* – Einzelverfahren → RSA-Kryptosystem → Primzahltest → Fermat-Test.

Der Fermat-Test ist nach Auskunft von Bernhard Esslinger, dem Projektleiter von *CrypTool*, folgendermaßen implementiert:

- ▷ Zunächst werden Probedivisionen mit allen Primzahlen von 2 bis 37 durchgeführt; damit wird die Primalität oder Zerlegbarkeit aller Zahlen bis $37^2 = 1369$ sicher erkannt. Auch für größere Zahlen gilt, dass viele weitere zusammengesetzte auf diese Weise gefunden werden.
- ▷ Pro Aufruf wird bis zu 100-mal mit einer zufälligen Basis a mit $2 \le a \le n-1$ getestet, ob $a^{n-1} \equiv 1 \pmod{n}$ gilt: Falls ja, wird die Zahl als (wahrscheinlich) prim erkannt. Ergibt sich bei diesen Rechnungen einmal das Ergebnis ungleich 1, hat man einen Fermat-Zeugen gefunden, und es wird zurückgemeldet, dass die Zahl sicher zusammengesetzt ist.

Wir wollen die Leistungsfähigkeit dieses Tests mit großen Carmichael-Zahlen überprüfen, weil wir ja wissen, dass diese für den Fermat-Test die härtesten Nüsse sind. Im Internet finden wir z. B. die Carmichael-Zahl $252601 = 41 \cdot 61 \cdot 101$, deren Primfaktoren durch die vorgeschaltete Probedivision nicht erwischt werden sollten. Diese Zahl wird aber bei 10 Fermat-Tests immer als zusammengesetzt erkannt. (Vielleicht führt *CrypTool* die Probedivision auch noch mit 41 durch? Da *CrypTool* Open-Source-Software ist, könnte man in den Programmquellen nachschauen.)

Aber wir können auch anders – versuchen wir es mit der Carmichael-Zahl $1299963601 = 601 \cdot 1201 \cdot 1801$. Hiermit können wir tatsächlich den Fermat-Test von *CrypTool* foppen: Bei zehn Versuchen geht diese Zahl beim Fermat-Test etwa fünfmal als Primzahl durch, während sich der Miller-Rabin-Test (s. u.) kein einziges Mal hinters Licht führen lässt. Wenn man schließlich mit der Carmichael-Zahl $173032371289 = 3067 \cdot 6133 \cdot 9199$ antritt, hat der Fermat-Test von *CrypTool* keine echte Chance mehr – es ist einfach zu unwahrscheinlich, zufällig auf eine nicht teilerfremde Zahl zu treffen.

Der Fermat-Test ist also nicht besonders zuverlässig, auch gibt es bislang keine Aussage darüber, mit welcher Wahrscheinlichkeit er versagt. Allerdings ist inzwischen bewiesen, dass es zu jeder Basis unendlich viele Pseudoprimzahlen gibt (vgl. Ribenboim, 2006, S. 93 ff.); seit 1994 wissen wir auch, dass es sogar unendlich viele Carmichael-Zahlen gibt (vgl. Ribenboim, 2006, S. 101 ff.). Trotzdem wurde der Fermat-Test in den ersten Versionen der bekannten Software PGP von Phil Zimmermann zur Konstruktion der RSA-Schlüssel verwendet (vgl. Wobst, 1997, S. 283). Aber immerhin funktioniert RSA auch mit Carmichael-Zahlen (RSA ist kein Primzahltest; vgl. Wikipedia, Stichwort „RSA-Kryptosystem“).

Miller-Rabin-Primzahltest

Mit dem von Gary Miller 1976 vorgeschlagenen und von Michael O. Rabin auf eine solide mathematische Grundlage gestellten Test prüft man ebenfalls, ob eine natürliche Zahl n eine Primzahl ist oder nicht.
Dabei benutzt man folgende *Sprechweisen*:

1.) Sei $n = 2^s \cdot d+1$ und d ungerade. Man sagt, dass n die *Miller-Rabin-Primzahltest-Runde mit Testbasis a* (für $1 < a < n$) besteht (und damit kein Anhaltspunkt für die Zerlegbarkeit von n vorliegt), wenn gilt:
$a^d \equiv 1 \pmod n$ oder $a^{2^j d} \equiv -1 \pmod n$
für ein $j \in \{0, 1, \ldots, s-1\}$.
Eine Zahl n, die den Test nicht besteht, ist mit Sicherheit zusammengesetzt (s. u.).

2.) Eine natürliche Zahl n mit $n = 2^s d+1$ heißt *starke Pseudoprimzahl zur Basis a* (für $1 < a < n$ und $\mathrm{ggT}(a, n) = 1$), falls sie den Miller-Rabin-Primzahltest zur Basis a besteht, obwohl sie zusammengesetzt ist.

Um die Wirkungsweise des Tests zu verstehen, benötigen wir einige Vorbetrachtungen:

Hilfssatz 1: *Ist p eine Primzahl, so hat die Kongruenz* $x^2 \equiv 1 \pmod p$ *außer* $x \equiv 1 \pmod p$ *und* $x \equiv -1 \pmod p$ *keine weiteren Lösungen.*

Beweis:

$$x^2 \equiv 1 \pmod p \;\Leftrightarrow\; p \mid (x^2-1) \;\Leftrightarrow\; p \mid (x+1)\,(x-1)$$
$$\underset{p \text{ prim}}{\Leftrightarrow} \; p \text{ teilt } (x-1) \text{ oder } p \text{ teilt } (x+1)$$
$$\Leftrightarrow \; x \equiv 1 \pmod p \text{ oder } x \equiv -1 \pmod p.$$

Anmerkung 1: Hilfssatz 1 besagt mit anderen Worten, dass das Polynom x^2-1 im Körper $K = \mathbb{Z}_p$ höchstens 2 Nullstellen hat. Dies gilt auch für beliebige Körper K wegen der Darstellung $x^2-1 = (x-a)(x-b)$ für mögliche Nullstellen a, b von x^2-1 und der Eindeutigkeit der Zerlegung in lineare Faktoren.

Anmerkung 2: Ist n keine Primzahl, so kann die Kongruenz $x^2 \equiv 1 \pmod n$ mehr als 2 Lösungen haben.
Beispiel: $n = 8, x \in \{1, 3, 5, 7\}$.

Satz 1: Grundlage des Miller-Rabin-Primzahltests

Ist p eine Primzahl, $p \neq 2$, *und* $p-1 = 2^s \cdot d$ *mit ungerader Zahl d, so ist die für* $a \in \{2, \ldots, p-1\}$ *u. a. durch fortgesetztes Quadrieren gewonnene Folge*
$F_a = (a^d \bmod p, (a^d)^2 \bmod p, (a^{2d})^2 \bmod p, \ldots, a^{2^s d} \bmod p)$
entweder gleich $(1, 1, \ldots, 1)$ *oder von der Form* $(*, *, \ldots, *, -1, 1, \ldots, 1)$*; hierbei stehen die Sterne, sofern überhaupt benötigt, für Zahlen ungleich* ± 1.

Beweis: Nach dem kleinen Satz von Fermat (siehe z. B. Witten/Schulz, 2008, S. 65) gilt für jede Primzahl p die Aussage $a^{2^s d} = a^{p-1} \equiv 1 \pmod p$; also endet $F := F_a$ mit 1. In F sei nun u der kleinste Index mit $a^{2^u d} \equiv 1 \pmod p$. Ist $u = 0$, so gilt $F = (1, \ldots, 1)$. Sei u ungleich 0 und x der Wert an der Stelle $u-1$ von F; an der Stelle u steht nach Definition von F dann $x^2 \bmod p$; nach Definition von u gilt aber auch $x^2 \equiv 1 \pmod p$, ferner $x \not\equiv 1 \pmod p$; es folgt $x = -1$ nach Hilfssatz 1.

Funktionsweise des Miller-Rabin-Primzahltests

Es sei zu testen, ob n eine Primzahl ist. Dazu werden zunächst s und d mit $n-1 = 2^s \cdot d$ bestimmt. Dann wird in jeder Testrunde ein zufälliges $a \in \{2, \ldots, n-1\}$ gewählt und begonnen, die Folge F_a solange zu berechnen, bis einer der im Folgenden aufgeführten Fälle zu erkennen ist:

▷ Wenn $a^d \equiv \pm 1 \pmod n$ gilt, dann ist $F_a = (\pm 1, 1, \ldots, 1)$ und a somit kein Zeuge für die Zerlegbarkeit von n, sondern ein *Entlastungszeuge*. Die Testrunde mit a kann beendet werden.

▷ Ein *Entlastungszeuge a* liegt auch vor, falls man (nach fortgesetztem Quadrieren) zu $a^{2^j d} \equiv -1 \pmod p$ für ein j mit $1 \leq j \leq s-1$ gelangt, also $F_a = (*, *, \ldots, *, -1, 1, \ldots, 1)$ ist.

Hilfssatz 2: *Ist a Entlastungszeuge für n, so ist n (definitionsgemäß) Primzahl oder starke Pseudoprimzahl zur Basis a. (Eine solche Zahl n kann also trotzdem noch zusammengesetzt sein.)*

▷ Beginnt die Folge F_a nicht mit 1 oder enthält keine -1 oder ist ein einer 1 vorangehendes Folgenglied ungleich ± 1, d. h. nicht-triviale Quadratwurzel von 1, dann kann nach Hilfssatz 1 die Zahl n keine Primzahl sein: die Zahl a ist dann Zeuge für die Zerlegbarkeit (*Belastungszeuge*) von n, und der Test insgesamt kann beendet werden. Denn es gilt ja:

Hilfssatz 3: *Gibt es einen Belastungszeugen (Zeugen für die Zerlegbarkeit) von n, so ist n keine Primzahl.*

Ist nun a kein Belastungszeuge, so wird in der nächsten Testrunde eine (weitere) zufällige Zahl a gewählt und mit ihr wie in der ersten Runde verfahren. (Oft wird für a jedes Element einer vorgegebenen Menge gewählt, z. B. der Menge aller Primzahlen unter einer bestimmten Schranke.) Mit jeder Runde ohne Zeugen für die Zerlegbarkeit erhöht sich die Wahrscheinlichkeit, dass n Primzahl ist.
Als Ergebnis des Miller-Rabin-Primzahltests wird entweder (mit Sicherheit) festgestellt, dass n keine Primzahl ist, oder mit einer kleinen Irrtums-Wahrscheinlichkeit (siehe Kasten „Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Primzahltest“, Seite 103) auf die Primalität geschlossen. Es handelt sich insofern um einen probabilistischen Primzahltest.

Anmerkung 3: Der Miller-Rabin-Primzahltest ist stärker als der Fermat-Test.
Aus $a^d \equiv 1 \pmod m$ folgt $a^{n-1} = (a^d)^{2^s} \equiv 1 \pmod n$;
aus $a^{2^j d} \equiv -1 \pmod n$ für $j < s$ ergibt sich
$a^{n-1} = (a^{2^j d})^{2^{s-j}} \equiv (-1)^{2^{s-j}} \equiv 1 \pmod n$.

Jede Zahl, die den Miller-Rabin-Primzahltest besteht, passiert also auch den Fermat-Test erfolgreich. Die Umkehrung ist falsch (s. Kasten „Beispiele zum Miller-Rabin-Primzahltest“, Beispiel c, Seite 102).

Michael O. Rabin und der Miller-Rabin-Primzahltest

Michael Oser Rabin (siehe Bild 5) wurde am 1. September 1931 in Breslau als Sohn eines Rabbiners geboren. Die Familie emigrierte 1935 nach Palästina, dort studierte Rabin nach der Beteiligung am israelischen Unabhängigkeitskrieg Mathematik an der Hebräischen Universität Jerusalem, schloss 1953 dort mit einem Diplom ab und promovierte 1956 in den USA an der Princeton University. Ende der Fünfzigerjahre entwickelte er zusammen mit Dana Scott die Idee einer nichtdeterministischen Turingmaschine und schuf damit die Grundlagen für die Formulierung des berühmten P-NP-Problems. Rabin und Scott erhielten 1976 für das Papier *Finite Automata and Their Decision Problem* (1959), in dem sie diese Theorie darlegten, den Turing-Preis für Informatik. Rabin gehört damit zu den bedeutendsten lebenden Informatikern.

Der Nichtdeterminismus spielt auch eine große Rolle im Miller-Rabin-Primzahltest (s.u. sowie Kasten „Miller-Rabin-Primzahltest“, vorige Seite). Daneben entwickelte Rabin das nach ihm benannte Kryptosystem sowie – zusammen mit seinem Freund Richard Karp – einen wichtigen Algorithmus zur Stringsuche. Aktuell beschäftigt er sich mit Computersicherheit und hält dazu Vorlesungen an der Harvard University sowie in Jerusalem und New York. Den Hörerinnen und Hörern gefallen besonders die witzigen Aussprüche, mit denen er seine Vorlesungen würzt und die von ihnen als „Rabinismus“ gesammelt und im Internet unter

http://www.digitas.harvard.edu/cgi-bin/wiki/ken/RabinismCollection

veröffentlicht werden. Hier zwei kleine Beispiele für den Rabinismus: „One plus one equals zero. We have to get used to this fact of life.“ Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Test führte er aus: „The probability of an error is smaller than the probability that none of us is awake and we are all dreaming this.“

Der Miller-Rabin-Test ist *das* Beispiel für ein sogenanntes *Monte-Carlo-Verfahren*. Monte-Carlo-Algorithmen sind dadurch gekennzeichnet, dass sie immer sehr schnell arbeiten, aber manchmal nicht korrekt (allerdings mit beschränkter Fehlerwahrscheinlichkeit; siehe Kasten „Miller-Rabin-Primzahltest“, vorige Seite, und Kasten „Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Primzahltest“, Seite 103). Natürlich will man möglichst keine Algorithmen, bei denen man überhaupt nicht weiß, in welchem Maße man sich auf die Ausgabe verlassen kann – diese Situation haben wir beim Fermat-Test. Aber interessant sind solche Algorithmen, bei denen die Wahrscheinlichkeit für ein falsches Ergebnis vernachlässigbar klein ist – und genau das trifft, wie wir sehen werden, auf den Miller-Rabin-Test zu. Das Besondere an den Monte-Carlo-Algorithmen ist also, dass sie erheblich schneller als deterministische Verfahren sind, wenn man eine (vernachlässigbar kleine) Fehlerwahrscheinlichkeit in Kauf nimmt (vgl. z.B. Rempe/Waldecker, 2009, S.116ff.).

Bild 5:
Michael O. Rabin, Turing-Preisträger und Miterfinder des Miller-Rabin-Primzahltests, am 13. Oktober 2009.

http://commons.wikimedia.org/wiki/File:Michael_O._Rabin.jpg

Gary L. Miller erlangte 1975 an der University of California in Berkeley seinen Doktortitel mit der Arbeit *Riemann's Hypothesis and tests for primality*. Dort entwickelte er einen (deterministischen) Primzahltest, der unter der Voraussetzung, dass die erweiterte Riemann'sche Vermutung (ERH = ***E**xtended **R**iemann **H**ypothesis*) gültig ist, in Polynomialzeit läuft. (*Anmerkung:* Die Erweiterung der Vermutung betrifft die betrachtete Funktion: Statt der Riemann'schen Zeta-Funktion werden die Nullstellen von Dirichlets L-Funktionen betrachtet – zur Riemann'schen Vermutung vgl. z.B. Witten/Schulz, 2010, S.100).

Wenn ein Primzahltest in Polynomialzeit läuft, bedeutet dies, dass eine deterministische Turingmaschine existiert, die das Entscheidungsproblem, ob eine Zahl prim ist oder nicht, in Polynomialzeit lösen kann, d.h. die Laufzeit kann durch ein Polynom nach oben abgeschätzt werden. Solche Algorithmen werden in der Komplexitätstheorie zu der Klasse *P* zusammengefasst (zur Polynomialzeit und zum P-NP-Problem vgl. auch Niedermeier u.a., 2007). Obwohl es sehr wünschenswert war, einen solchen Test zu finden, wurde zum Beweis von Gary L. Miller ein sehr starkes mathematisches Geschütz aufgefahren – der Beweis der ERH (oder ihre Widerlegung) liegt nach wie vor im Nebel der Zukunft.

Ebenfalls im Jahr 1975 trafen sich Rabin und Miller am MIT. Die Arbeit von Miller inspirierte Rabin dazu, den Primzahltest zu konstruieren, der heute nach Miller und Rabin benannt wird. Rabin präsentierte den Test 1976 bei einer Vorlesung an der Carnegie-Mellon-Universität. Später stellte sich heraus, dass John L. Selfridge diesen Test schon 1974 verwendet hatte, bevor Miller und Rabin ihn veröffentlichten. Daher rührt der alternative Name *Miller-Selfridge-Rabin-Test*. Wir bleiben aber im Folgenden bei der allgemein üblichen Bezeichnung *Miller-Rabin-Test*.

Parallel dazu entwickelten Robert M. Solovay und Volker Strassen einen anderen Monte-Carlo-Algorithmus zur Bestimmung von Primzahlen (dieser Test ist ebenfalls im Programm *CrypTool* implementiert) und publizierten ihn 1977 (vgl. Solovay/Strassen, 1977). Volker Strassen ist ein Wegbereiter der Komplexitätstheorie und hat neben dem erwähnten probabilistischen

Primzahltest für die Matrizenmultiplikation den neuen, schnelleren mittlerweile nach ihm benannten Strassen-Algorithmus entwickelt. Die Matrizen müssen allerdings groß sein, wenn der *Strassen-Algorithmus* schneller als das klassische Verfahren sein soll.

Außerdem entwickelte Strassen 1971 zusammen mit Arnold Schönhage den *Schönhage-Strassen-Algorithmus* zur Multiplikation großer ganzer Zahlen (vgl. Schönhage/Strassen, 1971), der bis 2007 in punkto Schnelligkeit den Weltrekord gehalten hat. Für seine bahnbrechenden Erfindungen wurde Volker Strassen u.a. 2008 mit dem Knuth-Preis der Association for Computing Machinery (ACM, US-amerikanisches Pendant zur Gesellschaft für Informatik) ausgezeichnet; der Preis wurde ihm von Gary L. Miller überreicht (siehe Bild 6).

http://en.wikipedia.org/wiki/File:Strassen_Knuth_Prize_presentation.jpg

Bild 6: Volker Strassen (rechts) wird 2008 mit dem Knuth-Preis der ACM als Anerkennung seiner bahnbrechenden Beiträge zur Theorie und Anwendung der Algorithmen-Entwicklung ausgezeichnet. Die Auszeichnung wird von Gary L. Miller (links), dem Miterfinder des Miller-Rabin-Tests, überreicht.

Michael O. Rabin veröffentlichte seine Arbeit *Probalistic Algorithm for Testing Primality* 1980 in der Zeitschrift *Jounal of Number Theory*; eingereicht hatte er die Arbeit bereits am 10. Dezember 1977 – wissenschaftliche Zeitschriften arbeiteten seinerzeit gründlich und langsam. In der Arbeit schreibt er nicht ohne Stolz: „Our test does *not* assume ERH and is considerably faster." Der Miller-Rabin-Test ist aber auch dem Solovay-Strassen-Test im Hinblick auf Geschwindigkeit und Zuverlässigkeit überlegen und daher inzwischen Standard. Im Jahr 2003 erhielten Miller, Rabin, Solovay und Strassen den *Paris Kanellakis Award* der ACM für ihre Arbeiten zu den probabilistischen Primzahltests.

Wie funktioniert der Miller-Rabin-Test?

Wenn man den Fermat-Test verbessern will, muss man neben dem kleinen Satz von Fermat eine zusätzliche Aussage über die Primalität der zu untersuchenden Zahl n verwenden. Diese Aussage lautet im Fall des Miller-Rabin-Tests: Falls p eine Primzahl ist, so hat die Kongruenz $x^2 \equiv 1 \pmod{p}$ *nur* die Lösungen $x \equiv 1 \pmod{p}$ und $x \equiv -1 \pmod{p}$. Der Beweis dieser Aussage ist relativ einfach zu führen (siehe Kasten „Miller-Rabin-Primzahltest", Seite 100). Diejenigen, die mit dem modularen Rechnen nicht vertraut sind (also z.B.

Beispiele zum Miller-Rabin-Primzahltest

Beispiel (a)

Für $n = 73$ ist $p-1 = 72 = 2^3 \cdot 9$, also $s = 3$ und $d = 1$. Für

- ▷ $a = 20$ ist $F_a = (20^9, 20^{18}, 20^{36}, 20^{72})$ modulo 73 zu berechnen; wegen $20^9 \equiv 10 \pmod{73}$, $10^2 \equiv 27 \pmod{73}$, $27^2 \equiv -1 \pmod{73}$ und $(-1)^2 \equiv 1 \pmod{73}$ gilt $F_{20} = (10, 27, -1, 1)$.
- ▷ $a = 37$ ist $(37^9, 37^{18}, 37^{36}, 37^{72})$ modulo 73 kongruent zu $(1, 1, 1, 1)$.

Also ist 73 Primzahl oder Pseudoprimzahl zu den Basen 20 und 37, ferner z.B. auch zu den Basen
55 wegen $55^9 \equiv 1 \pmod{73}$,
29 wegen $29^{36} \equiv -1 \pmod{73}$,
27 wegen $27^{18} \equiv -1 \pmod{73}$ und
60 wegen $60^{36} \equiv -1 \pmod{73}$.

Anmerkung: Für kleine Zahlen n ist die Primalität natürlich schneller durch Probedivisionen mit Teilern kleiner gleich $\sqrt{n}$ festzustellen. Für $n = 73$ sind lediglich 2, 3, 5, 7 als Teiler auszuschließen.

Beispiel (b)

Für n = 22564845703 gilt z.B.
$13^{11282422851} \equiv 1 \pmod{22564845703}$ und
$19^{11282422851} \equiv -1 \pmod{22564845703}$.
Weitere Entlastungszeugen sind: 23, 31, 41, 49 und 5701.
Hingegen ist 3 wegen
$3^{11282422851} \equiv 424875 \pmod{22564845703}$ und
$424875^2 \equiv 1 \pmod{22564845703}$ ein Belastungszeuge, also dieses n nicht prim.

Anmerkung: $n = 22564845703 = 106219 \cdot 212437$ wurde von DERIVE 4.11 noch für eine Primzahl gehalten (vgl. Wiesenbauer, 2001, S.15); bei DERIVE vor Version 5 wurden, abgesehen von einer Prüfung auf kleine Primteiler, standardmäßig nur sechs Miller-Rabin-Testrunden durchgeführt. (Die angegebene Zerlegung erhielten wir mit DERIVE 5.)

Beispiel (c)

$n = 561$: Die Carmichael-Zahl $n = 561$ besteht (mit $a = 2$, $s = 4$ und $d = 35$) wegen
$2^{35} \equiv 263 \pmod{561}$, $2^{70} \equiv 263^2 \equiv 166 \pmod{561}$ und $2^{140} \equiv 67 \pmod{561}$ sowie $2^{280} \equiv 1 \pmod{561}$,
also $F_2 = (263, 166, 67, 1)$, nicht den Miller-Rabin-Test.

unsere Schülerinnen und Schüler), werden diesen Satz nicht besonders erstaunlich finden, hat doch die quadratische Gleichung $x^2 = 1$ bei geeigneten Grundmengen sowieso nur die Lösungen 1 und –1.

Es liegt also nahe, die quadratische Kongruenz $x^2 \equiv 1 \pmod{n}$ für Module n zu untersuchen, die keine Primzahlen sind. Für $n = 8$ findet man tatsächlich heraus, dass neben 1 und –1 auch 3 und 5 die Kongruenz lösen, da $3^2 = 9$ und $5^2 = 25$ bei Division durch 8 jeweils den Rest 1 lassen. Man spricht davon, dass 3 und 5 *nicht-triviale Quadratwurzeln* von 1 sind; diese dürfen nicht auftreten, falls n eine Primzahl sein soll. Weitere Übungen mit zerlegbaren Moduln machen die Aussage für die Lernenden plausibel.

Der geniale Gedanke von Miller (und Selfridge) war es, aus diesen Überlegungen einen einfachen Test zu konstruieren. Da alle Primzahlen (außer 2) ungerade sind, gehen sie davon aus, dass die zu testende Zahl ungerade ist. Daher lässt sie sich in der Form $n = 2k + 1$ darstellen. Nun kann k gerade oder ungerade sein. Falls k gerade ist, kann erneut der Faktor 2 abgespalten werden. Dies wird so lange fortgesetzt, bis eine ungerade Zahl übrig bleibt, im Extremfall die 1. Man erhält damit die Zerlegung $n = 2^s d+1$ mit d ungerade, wobei $s \geq 1$ sein muss, da sich die 2 mindestens einmal abspalten lässt.

Falls n eine Primzahl ist, gilt (wie beim Fermat-Test, s. o.) für jede beliebige Testbasis a mit $2 \leq a \leq n–1$ Folgendes: $a^{n-1} \equiv 1 \pmod{n}$. Mit der obigen Zerlegung kann man für den Exponenten $n–1$ aber auch $2^s d$ schreiben. Die Berechnung der Potenz wird dann schrittweise erfolgen: Zunächst berechnen wir a^d und bilden den Rest bezüglich der Division durch n. Sollte dieser Rest 1 sein, sind wir fertig, da das weitere Potenzieren mit 2^s ja einem s-maligen Quadrieren entspricht, und da wird sich das Ergebnis 1 sicherlich nicht mehr ändern. Wir sprechen in diesem Fall davon, dass n die Miller-Rabin-Primzahltest-Runde mit der Basis a bestanden hat.

Soweit haben wir gegenüber dem Fermat-Test außer einem kleineren Exponenten noch nichts Neues erreicht. Wir gehen daher im Folgenden davon aus, dass $a^d \not\equiv 1 \pmod{n}$ ist. Dann wird geprüft, ob das Ergebnis eventuell –1 ist. Auch in diesem Fall sind wir ebenfalls mit einem positiven Ergebnis fertig, da mindestens einmal quadriert werden muss, sodass wir wieder bei 1 landen. Darüber hinaus stellt die –1 sicher, dass wir in der beim Quadrieren entstehenden Folge nur die trivialen Wurzeln von 1 haben, denn nur solche sind bei Primzahlen erlaubt (siehe Kasten „Miller-Rabin-Primzahltest“, Seite 100). Ist das Ergebnis weder 1 noch –1, wird (höchstens (s–1)-mal) quadriert. Wenn wir bei dieser Folge auf eine –1 stoßen, sind wir wiederum mit einem positiven Ergebnis fertig, falls nicht, endet die Folge nicht bei 1 oder es tritt eine nicht-triviale Quadratwurzel von 1 auf. Dann hat n die Miller-Rabin-Primzahltest-Runde mit der Basis a nicht bestanden und ist sicher zusammengesetzt. Für Zahlenbeispiele zum Miller-Rabin-Test siehe Kasten „Beispiele zum Miller-Rabin-Primzahltest“, vorige Seite.

Die Verschärfung des Miller-Rabin-Tests gegenüber dem Fermat-Test besteht also kurz gesagt darin, dass nicht nur der kleine Satz von Fermat überprüft wird (Auftreten der 1), sondern dass zusätzlich das Auftreten nichttrivialer Wurzeln aus 1 untersucht wird. Der Rechenaufwand ist sogar geringer als beim Fermat-Test, da das Potenzieren nach der Methode „teile und herrsche“ zerlegt wird. Die dabei entstehende Zahlenfolge der Zwischenergebnisse liefert die erwünschten Aussagen in einigen Fällen schon bevor das Potenzie-

Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Primzahltest

Man kann zeigen (s. u. Satz 1), dass eine zusammengesetzte Zahl bei mindestens $\frac{3}{4}$ der Zahlen zwischen 1 und n–1 als Basen die Miller-Rabin-Primzahltest-Runde nicht besteht. Durch k Tests und zufällige Wahl der Basen lässt sich für die Aussage „n ist Primzahl“ (d. h. für das Ereignis, dass ausschließlich Entlastungszeugen, jeder mit einer Wahrscheinlichkeit kleiner gleich $\frac{1}{4}$, ausgewählt wurden) eine Irrtums-Wahrscheinlichkeit kleiner gleich $\left(\frac{1}{4}\right)^k$ erreichen und damit durch eine geeignete Anzahl k von Testrunden unter eine gegebene Schranke drücken. Dies wollen wir mit folgendem Satz von Rabin weiter verdeutlichen:

Satz 1: Zur Anzahl der falschen Entlastungszeugen

Sei n eine ungerade zusammengesetzte natürliche Zahl. Dann gibt es in der Menge $\{1, \ldots, n–1\}$ höchstens $\frac{n-1}{4}$ Zahlen a, die zu n teilerfremd und keine Belastungszeugen (Zeugen für die Zerlegbarkeit) von n sind.

Die ausführliche Beweisskizze finden Sie als PDF-Datei im LOG-IN-Service, siehe Seite 143.

Anmerkung 1: Ist $a < n$ nicht teilerfremd zu n, also $\mathrm{ggT}(a, n) = g \neq 1$, so tritt der Fall $a^{2^j d} \equiv \pm 1 \pmod{n}$ nicht ein; andernfalls wäre auch $a^{2^j d} \equiv \pm 1 \pmod{g}$ im Widerspruch zu $a \equiv 0 \pmod{g}$. Daher ist dann a Zeuge für die Zerlegbarkeit von n.

Anmerkung 2: Wie man dem Beweis von Satz 1 (im LOG-IN-Service, siehe Seite 143) entnehmen kann, besteht ein zusammengesetztes ungerades n mit $n \neq 9$ höchstens für $\frac{\varphi(n)}{4}$ aller Basen a mit $0 < a < n$ den Rabin-Miller-Primzahltest, ohne die Zerlegbarkeit von n anzuzeigen. (Hierbei bezeichnet φ die Euler'sche Phi-Funktion, d. h. $\varphi(n)$ ist die Anzahl der zu n teilerfremden natürlichen Zahlen kleiner n). Bei $n = 9$ gibt es die Entlastungszeugen +1, –1, wogegen $\frac{\varphi(9)}{4} = \frac{6}{4} < 2$ ist.

Korollar: Fehlerwahrscheinlichkeit des Miller-Rabin-Primzahltests

Ist $n > 9$ und stellt sich in k Runden das jeweils zufällig und gleichverteilt gewählte $a \in \{1, \ldots, n–1\}$ als Zeuge für die Primalität von n heraus, so ist die Fehlerwahrscheinlichkeit des Tests (bei Ausgabe „n ist prim“) kleiner gleich $\left(\frac{1}{4}\right)^k$. Gibt es hingegen einen Zeugen für die Zerlegbarkeit von n, so ist n mit Sicherheit zerlegbar.

ren zu Ende geführt wurde (siehe Kasten „Beispiele zum Miller-Rabin-Primzahltest“, Seite 102).

Wir geben im Folgenden eine kompakte Implementierung des Miller-Rabin-Tests als PYTHON-Funktion rab_test(a,N) an. Im Parameter a wird die Testbasis, bei N die zu testende Zahl übergeben. Wir haben uns dabei an einem Programm aus dem Buch von Otto Forster zur algorithmischen Zahlentheorie orientiert, in dem die MODULA-2-ähnliche Programmiersprache ARIBAS verwendet wird (Forster, 1996, S. 102 f.). Wenn man in diesem Programm geeignete print-Anweisungen für die Variable u einbaut, kann man die Zahlenbeispiele aus Kasten „Beispiele zum Miller-Rabin-Primzahltest“ sowie eigene Beispiele leicht nachvollziehen.

```
def rab_test(a,N):
    if N%2 == 0: return False # N ist gerade
    d = N/2 # N ist ungerade, Teilen ohne Rest!
    s = 1   # mindestens einmal wird quadriert
    while d%2 == 0:
        d = d/2 # Abspalten des Faktors 2
        s = s+1
    # Nach dem Schleifendurchlauf
    # gilt N-1 = (2^s)*d
    u = pow(a,d,N)    # modulares Potenzieren mod N
    if (u == 1) or (N-u == 1):
        return True    # u ist kongurent +1 oder -1
    for k in range(s-1):
        u = (u*u)%N
               # Quadrieren und modulo N reduzieren
        if N-u == 1:
            return True # u ist kongruent -1
    return False # Miller-Rabin-Testrunde nicht
                 # bestanden!
```

Bei den Experimenten mit den Carmichael-Zahlen hatten wir beim Vergleich der Ergebnisse des Fermat-Tests und des Miller-Rabin-Tests aus *CrypTool* bereits gesehen, dass der Miller-Rabin-Test dem Fermat-Test überlegen ist. Dabei handelt es sich aber um eine „Black Box“ mit vorgeschalteter Probedivision und hundertfacher Ausführung der Tests mit Zufallszahlen; die Zwischenergebnisse werden nicht angezeigt.

Mit dem PYTHON-Programm können wir jetzt feinere Untersuchungen vornehmen, indem wir die Testbasis vorgeben und jeweils nur eine Testrunde durchführen. Dabei zeigt sich, dass auch der Miller-Rabin-Test nicht in jedem Fall das richtige Ergebnis liefert:

```
>>> rab_test(50,561)
True
>>> rab_test(2,252601)
True
```

In diesem Fall wurden die Carmichael-Zahlen 561 und 252601 (s. o.) fälschlicherweise für Primzahlen gehalten. 561 wird eine *starke Pseudoprimzahl* zur Basis 50 genannt, ebenso ist 252601 eine starke Pseudoprimzahl zur Basis 2. Auf der anderen Seite liefert der Miller-Rabin-Test sehr schnell ein sicheres Ergebnis, wenn die Zahl sehr groß und zusammengesetzt ist. Man kann sich davon z. B. mit der Zahl RSA-768 und der Testbasis 2 überzeugen, bei der in Sekundenbruchteilen False zurückgegeben wird (zu den Zahlen aus der RSA-Challenge vgl. Witten/Schulz, 2010, sowie Schulz/Witten, 2010, S. 107 ff. in diesem Heft).

Um die Zuverlässigkeit des Miller-Rabin-Tests abschätzen zu können, möchte man wissen, wie der Anteil der starken Pseudoprimzahlen ist. Zunächst einmal weiß man seit 1980, dass es zu jeder Basis unendlich viele starke Pseudoprimzahlen gibt (vgl. Ribenboim, 2006, S. 98). Immerhin gibt es kein Analogon zu den Carmichael-Zahlen, d. h. es gibt keine starken Pseudoprimzahlen, die diese Eigenschaft zu *allen* zu *n* teilerfremden Basen *a* haben. Rabin bewies sogar, dass jede Zahl *n* zu höchstens ein Viertel der Basen starke Pseudoprimzahl sein kann.

Damit ist es möglich, den Fehler beim Miller-Rabin-Test abzuschätzen: Bei *k* Runden mit jeweils unabhängig voneinander und zufällig ausgewählten Basen *a* ist die Fehlerwahrscheinlichkeit kleiner gleich $(\frac{1}{4})^k$ (siehe Kasten „Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Primzahltest“ und LOG-IN-Service). Nehmen wir an, dass wir 100 Testrunden durchführen. Dann ist die Fehlerwahrscheinlichkeit kleiner gleich $(\frac{1}{4})^{100} \approx 6{,}3 \cdot 10^{-61}$, d. h. wir müssten im Mittel erst bei $1/(6{,}3 \cdot 10^{-61}) \approx 1{,}6 \cdot 10^{60}$ Versuchen mit einem Fehlschlag rechnen. Ribenboim schreibt dazu (2006, S. 120):

> Vielleicht möchten Sie ja Primzahlen für kryptographische Zwecke verkaufen. [...] Und Sie möchten sicher sein, oder zumindest mit einer vernachlässigbaren Fehlerquote davon überzeugt sein, dass Sie tatsächlich Primzahlen verkaufen, so dass Sie werben könnten: „Garantierte Zufriedenheit oder Geld zurück.“ Auf der Grundlage von Rabins Test können Sie ein solides Unternehmen aufbauen und das Produkt guten Gewissens verkaufen.

Deterministische Primzahltests – Geht es auch ohne Glücksspiel?

Es bleibt bei manchem ein unbefriedigendes Gefühl – warum muss man mit dem Zufall arbeiten? Eine Möglichkeit ist, dass jemand die erweiterte Riemann’sche Vermutung beweist, dann hat uns Gary L. Miller gezeigt, dass es unter dieser Voraussetzung einen deterministischen Algorithmus gibt, der in polynomialer Zeit entscheiden kann, ob eine Zahl prim oder zusammengesetzt ist – und zwar wirklich prim und nicht nur (höchst-)wahrscheinlich prim. Aber wann wird das sein? Wird es überhaupt sein?

Im Jahr 1979 hatten Leonard M. Adleman (das „A“ von RSA), Carl Pomerance und Robert S. Rumely mit der Arbeit *On distinguishing prime numbers from composite numbers* das Zeitalter der modernen Primzahltests eröffnet (veröffentlicht 1983). Nach Adleman war das die erste Arbeit über Informatik, die in der angesehenen führenden US-Mathematik-Zeitschrift *Annals of Mathematics* veröffentlicht wurde. Ihr Test, der ohne Zufallszahlen auskommt und deshalb deterministisch arbeitet, wurde wenig später von Henri Cohen und Hendrik W. Lenstra, Jr., noch verbessert (vgl. Cohen/Lenstra, 1984). Die Laufzeit ist für eine zu testende Zahl *n* aber leider nur fast polynomial.

Der Miller-Rabin-Test benötigt ebenso wie der Solovay-Strassen-Test polynomiale Laufzeit, beide Tests sind aber „nur“ probabilistisch. Für die theoretischen Informatiker war es daher lange Zeit ein offenes Pro-

blem, ob die Primzahltests zur Klasse P der Algorithmen mit Polynomialzeit gehört.

Im Jahr 2002 wurde von drei indischen Informatikern (siehe Bild 7) ein Artikel veröffentlicht, der eine Sensation auslöste (Bornemann, 2002, S. 14):

„New Method Said to Solve Key Problem in Math" titelte die New York Times am 8. August 2002 und meinte den Nachweis von PRIMES ∈ P, ein bislang großes offenes Problem der algorithmischen Zahlentheorie und theoretischen Informatik. Manindra Agrawal, Neeraj Kayal und Nitin Saxena vom Indian Institute of Technology war es durch einen überraschend eleganten und brillant einfachen Algorithmus gelungen, die binnen weniger Tag von der Korrektheit überzeugte Fachwelt ins Schwärmen zu versetzen: „This algorithm is beautiful" (Carl Pomerance), „It's the best result I've heard in over 10 years" (Shafi Goldwasser).

Die Mathematik hinter diesem Test, für den sich inzwischen die Bezeichnung *AKS-Primzahltest* eingebürgert hat, ist etwas anspruchsvoller als beim Miller-Rabin-Test, aber immer noch auf ziemlich elementarem Niveau. Folkmar Bornemann schildert auch, welche Blüten diese Nachricht bei den Zeitungsredaktionen getrieben hat. So stand z. B. in den Osnabrücker Nachrichten vom 11. August 2002 zu lesen, dass es den indischen Computerwissenschaftlern zum ersten Mal seit 2200 Jahren gelungen sei, Primzahlen fehlerfrei zu berechnen (Bornemann, 2002, S. 14).

Inzwischen ist das Buch *Primzahltests für Einsteiger – Zahlentheorie, Algorithmik, Kryptografie* von Lasse Rempe und Rebecca Waldecker erschienen (2009). Dieses Buch, in dem die hier vorgestellten Themen ausführlich behandelt werden, ist aus einem Kurs für hochbegabte Schülerinnen und Schüler an der Deutschen SchülerAkademie entstanden. Das Niveau entspricht dem Grundstudium Mathematik und Informatik. Das Buch hat als weiteren Schwerpunkt eine Einführung in mathematisches Arbeiten anhand eines aktuellen Forschungsthemas: des AKS-Primzahltests.

Der AKS-Primzahltest ist gewiss ein großer, in dieser Form unerwarteter Fortschritt für die Komplexitätstheorie; für das RSA-Kryptosystem ist er allerdings weniger revolutionär, dort wird weiterhin der Miller-Rabin-Test verwendet.

Von dieser Tatsache kann man sich mithilfe von *CrypTool* überzeugen, da neben den probabilistischen Tests von Miller-Rabin, Solovay-Strassen und dem nach Fermat auch der deterministische AKS-Test implementiert wurde. Wenn wir z. B. den Primfaktor q der Zahl RSA-768 auf Primalität testen, liefern alle drei probabilistischen Tests innerhalb von Sekunden das richtige Ergebnis. Beim AKS-Test erscheint eine Anzeige, dass der 384-Bit-AKS-Test zu 0 % erledigt wurde. Danach wird die benötigte Rest-Zeit geschätzt, und wenn diese nach einiger Zeit mehrere Jahre oder schließlich Jahrhunderte beträgt, kann man den AKS-Test getrost abbrechen.

In der Tat ist der originale AKS-Algorithmus, der in *CrypTool* implementiert wurde, enttäuschend langsam: So wird für den Nachweis der Primalität der 32-Bit Zahl 3571589027 mehr als eine Stunde benötigt. Bei den folgenden Experimenten geben wir die von *CrypTool* geschätzte Restzeit an, die zwar nur grobe Werte liefert, aber schon einen Einblick in das Wachstum der benötigten Zeit bei der Verdopplung der Bit-Längen gibt:

Bild 7: Der AKS-Primzahltest wurde nach Professor Manindra Agrawal (links) und seinen beiden Studenten Neeraj Kayal (unten links) und Nitin Saxena (unten rechts) benannt.

Fotos: IIT Kanpur (Indien)

64 Bit: 14 Tage,
128 Bit: 1,7 Jahre,
256 Bit: > 100 Jahre.

Und selbst wenn man die 100 Jahre investieren würde, erhielte man doch nur einen RSA-Schlüssel der Länge von $2 \cdot 256 = 512$ Bit, der schon seit zehn Jahren als unsicher gilt. Für die Gewinnung von RSA-Schlüsseln ist der AKS-Test in seiner ursprünglichen Form leider noch völlig unbrauchbar. Zu dieser Frage hat uns Bernhard Esslinger folgende Informationen übermittelt (Esslinger, 2011):

In der Literatur findet man einige starke Verbesserungen zu dem ursprünglichen AKS-Algorithmus. Einer der zurzeit schnellsten Algorithmen ist der Algorithmus von Daniel J. Bernstein (vgl. Bernstein, 2007). Man findet bei Bernstein auch eine sehr detaillierte Aufstellung der Primzahltests (vgl. Bernstein, 2004 ff.). Der Algorithmus hat die Laufzeit $\mathcal{O}(\log(p)^{(4+o(1))})$. Den Ausführungen Bernsteins ist zu entnehmen, dass für eine 279-Bit-Zahl eine Laufzeit von $1{,}18 \cdot 10^{11}$ CPU-Clock-Zyklen gemessen wurde (bei einer 1.3 GHz-Maschine ergibt das ca. 90 Sekunden). Das würde ca. 4,5 Stunden für eine 1024-Bit-Zahl ergeben. Selbst diese starke Verbesserung ist in der kommerziellen Anwendung unbrauchbar. Jedoch wäre eine solche Laufzeit z. B. für eine Implementierung in *CrypTool 2* wohl noch akzeptabel.

Es wurden also schon einige Verbesserungen erzielt, aber es bedarf weiterer Optimierungen, damit der AKS-Test nicht nur in der Theorie, sondern auch für die Praxis interessant wird. Miller, Rabin und Selfridge sind nach wie vor die unangefochtenen Champions zur Erzeugung von *industrial grade primes* (eine von H. Cohen geprägte Bezeichnung) für RSA.

Ausblick

In der nächsten Folge werden wir uns, wie angekündigt, mit der Sicherheit des RSA-Kryptosystems beschäftigen. Einen ersten Einblick zu dieser Frage erhält man mit den *Zeit-Experimenten zur Faktorisierung* (vgl. Schulz/Witten, 2010, S. 107 ff., in diesem Heft).

Helmut Witten
Brandenburgische Straße 23
10707 Berlin

E-Mail: helmut@witten-berlin.de

Prof. Dr. Ralph-Hardo Schulz
Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Mathematik
Arnimallee 3
14195 Berlin

E-Mail: schulz@math.fu-berlin.de

Wir danken Bernhard Esslinger, Malte Hornung und Astrid Witten für Verbesserungsvorschläge zu den ersten Entwürfen dieses Beitrags.

Im **LOG-IN-Service** (siehe Seite 143) stehen ein Text mit den Beweisen von Hilfssatz 1 aus dem Kasten „Der Fermat-Test …“ sowie der Rabin'schen Fehlerschranke für den Miller-Rabin-Test aus dem Kasten „Zur Fehlerwahrscheinlichkeit beim Miller-Rabin-Primzahltest“, Musterlösungen zu den o. a. Computerexperimenten und eine umfangreiche Liste weiterer themenbezogener Internetquellen zum Herunterladen bereit.

Literatur und Internetquellen

Adleman, L. M.; Pomerance, C.; Rumely, R. S.: On distinguishing prime numbers from composite numbers. In: Annals of Mathematics, Band 117 (1983), Nr. 1, S. 173–206.

Baptist, P. (Hrsg.): Alles ist Zahl. Motive von Eugen Jost. Köln: Kölner Universitäts-Verlag, ^{2}2009.

Baumann, R.: Das Schmidt'sche Postulat. In: LOG IN, 29. Jg. (2009), Heft 160/161, S. 118–119.

Bernstein, D. J.: Proving primality in essentially quartic random time. In: Mathematics of Computation, Band 76 (2007), Nr. 257, S. 389–403.
http://cr.yp.to/primetests/quartic-20060914-ams.pdf

Bernstein, D. J.: Distinguishing prime numbers from composite numbers, 2004 ff.
http://cr.yp.to/primetests.html

Bornemann, F.: Ein Durchbruch für „Jedermann“. In: Mitteilungen der DMV, 10. Jg. (2002), Heft 4, S. 14–21.
http://www-m3.ma.tum.de/foswiki/pub/M3/Allgemeines/FolkmarBornemannPublications/aks.pdf

Cohen, H.; Lenstra Jr., H. W.: Primality Testing and Jacobi Sums. In: Mathematics of Computation, Band 42 (1984), Nr. 165, S. 297–330.

Esslinger, B.: Persönliche Mitteilung an die Verfasser, 2011.

Forster, O.: Algorithmische Zahlentheorie. Braunschweig; Wiesbaden: Vieweg, 1996.

Karpfinger, C.; Kiechle, H.: Kryptologie – Algebraische Methoden und Algorithmen. Wiesbaden: Vieweg + Teubner, 2010.

Niedermeier, R.; Vogel, J.; Fothe, M.; König, M.: Das Knotenüberdeckungsproblem – Eine Fallstudie zur Didaktik NP-schwerer Probleme. In: LOG IN, 27. Jg. (2007), Heft 146/147, S. 53–59 (Teil 1), Heft 148/149, S. 81–89 (Teil 2).

Rabin, M. O.: Probabilistic Algorithm for Testing Primality. In: Journal of Number Theory, 12. Jg. (1980), Heft 1, S. 128–138.
http://tinyurl.com/5ssyrbx (ggf. kostenpflichtig)

Rabin, M. O.; Scott, D.: Finite Automata and Their Decision Problem. In: IBM Journal of Research and Development, 3. Jg. (1959), Heft 2, S. 114–125.
http://www.cse.chalmers.se/~coquand/AUTOMATA/rs.pdf

Rempe, L.; Waldecker, R.: Primzahltests für Einsteiger – Zahlentheorie, Algorithmik, Kryptographie. Wiesbaden: Vieweg + Teubner, 2009.

Ribenboim, P.: The New Book of Prime Number Records. Berlin; Heidelberg; New York: Springer, ^{3}1996.

Ribenboim, P.: Die Welt der Primzahlen – Geheimnisse und Rekorde. Berlin; Heidelberg; New York: Springer, 2006.

Schönhage, A.; Strassen, V.: Schnelle Multiplikation großer Zahlen, In: Computing, 7. Jg. (1971), Heft 3, S. 281–292.

Schulz, R.-H.: Codierungstheorie – Eine Einführung. Wiesbaden: Vieweg, ^{2}2003.

Schulz, R.-H.; Witten, H.: Zeit-Experimente zur Faktorisierung – Ein Beitrag zur Didaktik der Kryptologie. In: LOG IN, 30. Jg. (2010), Heft 166/167, S. 107–114 *(in diesem Heft)*.

Solovay, R. M.; Strassen, V.: A Fast Monte-Carlo Test for Primality. In: SIAM Journal on Computing, 6. Jg. (1977), Heft 1, S. 84–85.

Stallings, W.: Datensicherheit mit PGP. München; New York: Prentice Hall, 1995.

Wiesenbauer, J.: Primzahltests und Faktorisierungsalgorithmen I. In: IMN – Internationale Mathematische Nachrichten, 55. Jg. (April 2001), Nr. 186, S. 9–23.
http://www.oemg.ac.at./IMN/imn186.pdf

Wikipedia – Stichwort „Ernst August von Hannover (1954) – Öffentliches Leben“:
http://de.wikipedia.org/wiki/Ernst_August_von_Hannover_(1954)#.C3.96ffentliches_Leben

Wikipedia – Stichwort „RSA-Kryptosystem“
http://de.wikipedia.org/wiki/RSA-Kryptosystem

Witten, H.; Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge – Teil 1: RSA für Einsteiger. In: LOG IN, 26. Jg. (2006a), Heft 140, S. 45–54.

Witten, H.; Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge – Teil 2: RSA für große Zahlen. In: LOG IN, 26. Jg. (2006b), Heft 143, S. 50–58.

Witten, H.; Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge – Teil 3: RSA und die elementare Zahlentheorie. In: LOG IN, 28. Jg. (2008), Heft 152, S. 60–70.

Witten, H.; Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge – Teil 4: Gibt es genügend Primzahlen für RSA? In: LOG IN, 30. Jg. (2010), Heft 163/164, S. 97–103.

Wobst, R.: Abenteuer Kryptologie – Methoden, Risiken und Nutzen der Datenverschlüsselung. Bonn; Reading (MA, USA): Addison-Wesley, 1997.

Ziegler, G. M.: Primzahltests und Primzahlrekorde. In: Informatik Spektrum, 32. Jg. (2009), Heft 1, S. 33–39.

Alle Internetquellen wurden zuletzt am 30. Dezember 2010 geprüft.