

Ausarbeitung zum Seminar
Kryptographie
am
Lehrstuhl für Informatik I
Algorithmen und Komplexität
Prof. Dr. Berthold Vöcking
Rheinisch-Westfälische Technische Hochschule Aachen

PRIMES is in P

Der AKS-Primzahltest

Zusammenfassung

Diese Ausarbeitung behandelt den, im Jahr 2002 von Agrawal, Kayal und Saxena vorgestellten, ersten (deterministischen) Algorithmus, der in Polynomialzeit entscheidet, ob eine natürliche Zahl zusammengesetzt oder prim ist.

Florian Weingarten

Betreuer:
Dipl.-Inform.
Alexander Skopalik

Aachen, 4. August 2009

Inhaltsverzeichnis

1	Einleitung	3
2	Komplexitätstheorie und bisherige Ergebnisse	3
3	Notation und Voraussetzungen	4
4	Der AKS-Algorithmus	5
4.1	Idee	5
4.2	Algorithmus	6
4.3	Korrektheit	7
4.4	Terminierung	10
4.5	Laufzeit	11
5	Zusammenfassung	12
	Literatur	13

1 Einleitung

The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. (...) Further, the dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated.

Carl Friedrich Gauss

Eine natürliche Zahl, die größer als 1 ist, nennt man *prim*, wenn sie keine nichttrivialen Teiler hat, d.h. durch genau zwei natürliche Zahlen teilbar ist, nämlich nur durch 1 und sich selbst. Primzahlen spielen eine wichtige Rolle in vielen Teilgebieten der Mathematik, insbesondere in der Zahlentheorie, und haben in den letzten Jahren in der Kryptographie eine äußerst praktische und aus dem täglichen Leben nicht mehr wegzudenkende Anwendung gefunden.

In vielen kryptographischen Algorithmen werden große Primzahlen benötigt. Die übliche Vorgehensweise dabei ist, dass eine Zahl zufällig gewählt wird um anschliessend zu prüfen ob sie prim ist. Lange Zeit war nicht bekannt, ob es effizient möglich ist diesen Test durchzuführen, bis im Jahre 2002 die drei indischen Mathematiker Manindra Agrawal, Neeraj Kayal und Nitin Saxena ihre Arbeit „PRIMES is in P“ publizierten und damit für Aufsehen sorgten.

Die von Agrawal, Kayal und Saxena benutzten Methoden sind verblüffend einfach, wenn man bedenkt, dass die Frage nach einem effizienten Test extrem lange ungelöst war und von sehr vielen Mathematikern versucht wurde zu beantworten. Der Beweis bedient sich ausschliesslich elementarer Methoden aus der Algebra und der Zahlentheorie, die von vermutlich jedem Mathematik Studenten nachvollzogen werden können. Im folgenden soll der Algorithmus, sein Korrektheitsbeweis sowie eine Laufzeitabschätzung vorgestellt werden.

2 Komplexitätstheorie und bisherige Ergebnisse

Unmittelbar aus der Definition von *Primzahl* folgt bereits ein einfacher Primzahltest, die sogenannte *Probe Division*: Teste, ob n durch irgendeine der Zahlen $1 < m \leq \sqrt{n}$ teilbar ist. Falls ja, ist n zusammengesetzt, anderenfalls ist n prim. Dieser Test war bereits vor über 2000 Jahren dem griechischen Mathematiker Eratosthenes bekannt.

Leider ist dieser Test nicht *effizient*, da man $O(\sqrt{n})$ Divisionen durchführen muss. Wir nennen einen Primzahltest *effizient*, wenn sich seine Laufzeit durch ein Polynom in der Eingabelänge¹ abschätzen lässt, d.h. $O(\log^d(n))$ für ein $d \in \mathbb{N}$. Die Probedivision (oder auch *Sieb des Eratosthenes*) hat daher eine Laufzeit, die exponentiell in der Eingabelänge ist, denn $\sqrt{n} = n^{\frac{1}{2}} = 2^{\frac{1}{2} \cdot \log(n)}$.

Eine interessante (und effiziente) Möglichkeit erhält man aus dem kleinen Satz von Fermat: Gilt $a^n \not\equiv a \pmod{n}$, so kann n nicht prim sein. Das Problem ist, dass, falls $a^n \equiv a \pmod{n}$ doch gilt, n nicht zwangsweise prim sein muss. Solche Zahlen n (die nicht prim sind) heißen Carmichael-Zahlen und unglücklicherweise gibt es davon unendlich viele.

¹Um eine Zahl n binär zu kodieren benötigt man mindestens $\lfloor \log_2(n) \rfloor + 1$ Bits.

Aus der Komplexitätstheorie ist die folgende Klassenhierarchie bekannt:

$$P \subseteq ZPP = RP \cap \text{Co-RP} \subseteq RP, \text{Co-RP} \subseteq BPP \subseteq NP \cap \text{Co-NP} \subseteq NP, \text{Co-NP}$$

Mit PRIMES bezeichnet man das folgende Entscheidungsproblem: „Gegeben eine natürliche Zahl $n > 1$. Ist n prim oder zusammengesetzt?“. Es stellt sich die Frage, zu welcher Komplexitätsklasse PRIMES gehört. Offensichtlich ist $\text{PRIMES} \in \text{Co-NP}$ (denn falls n zusammengesetzt ist, gibt es dafür ein kurzes Zertifikat, nämlich einen Faktor von n). Vaughan Pratt hat 1975 gezeigt, dass $\text{PRIMES} \in \text{NP}$ ebenfalls gilt (d.h. falls n prim ist, dann lässt sich dies effizient verifizieren).

Der 1977 von Robert M. Solovay und Volker Strassen vorgestellte *Solovay-Strassen-Test* zeigt, dass PRIMES zur Komplexitätsklasse $\text{Co-RP} \subseteq \text{BPP}^2$ gehört, d.h. seine Laufzeit ist polynomiell, er arbeitet deterministisch und falls n prim ist, liefert der Test immer die korrekte Antwort. Falls n zusammengesetzt ist, liefert er in mehr als der Hälfte der Fälle das korrekte Ergebnis. Durch wiederholte Anwendung solcher probabilistischer Primzahltests lässt sich die Wahrscheinlichkeit, eine falsche Antwort zu erhalten, beliebig klein machen (aber eben nicht auf 0 bringen). Der Miller-Rabin-Test funktioniert ähnlich.

Adleman und Huang zeigten 1992, dass PRIMES auch in RP ist (und damit in $\text{Co-RP} \cap \text{RP} = \text{ZPP}^3$). Mit ZPP bezeichnet man die Klasse der Probleme, für die es eine nicht-deterministische Turingmaschine gibt, die immer die korrekte Antwort liefert (daher der Name), deren Laufzeit aber nur im Durchschnitt polynomiell ist. Die Klasse P ist in ZPP enthalten. Ob sie echt enthalten ist, ist nicht bekannt und PRIMES war lange Zeit eines der wenigen Probleme, von denen man wusste, dass sie in ZPP sind, aber nicht ob sie in P sind.

Lange Zeit kannte man keinen Test, der deterministisch ist, für alle Primzahlen funktioniert, polynomielle Laufzeit hat und dessen Korrektheit ohne Anwendung unbewiesener Vermutungen (wie z.B. der Riemannschen Hypothese) bewiesen werden kann. Algorithmen, die jeweils drei dieser vier Punkte erfüllen, gab es allerdings schon. Das bekannteste Beispiel ist der *Adleman-Pomerance-Rumely primality test*, der die ersten beiden und den letzten Punkt erfüllt und eine Laufzeit von $\log(n)^{O(\log \log \log n)}$ hat (was in der Praxis „fast polynomiell“ ist, da der Exponent z.B. nicht größer als 4 wird für $n \leq 10^{10^{23}}$).

Der AKS-Primzahltest löst dieses offene Problem und zeigt, dass PRIMES zur Komplexitätsklasse P gehört.

3 Notation und Voraussetzungen

In diesem kurzen Abschnitt wollen wir einige Notationen einführen sowie Fakten aus der Algebra und der elementaren Zahlentheorie wiederholen. Für eine ausführlichere Erklärung siehe [8], [13] und [7].

- Eine natürliche Zahl n heißt *echte Potenz*, falls $n = a^b$ für $a \in \mathbb{N}$, $b \in \mathbb{N} \setminus \{1\}$.
- Ein Element eines kommutativen Ringes R , dessen r -te Potenz ($r > 0$) gleich 1 ist, wird *r -te Einheitswurzel* genannt. Die r -ten Einheitswurzeln sind genau die Nullstellen des Polynoms $x^r - 1$ aus $R[x]$. Gilt $d|r$, so ist jede d -te Einheitswurzel auch eine r -te Einheitswurzel, insbesondere teilt $x^d - 1$ also $x^r - 1$ in $R[x]$. Das ganzzahlige normierte Polynom größten Grades, dass $x^r - 1$ teilt, aber zu allen $x^d - 1$ mit $d < r$ teilerfremd ist,

²RP steht für *Randomized Polynomial Time*, BPP für *Bounded Error Probability Polynomial Time*.

³Zero-error Probabilistic Polynomial Time.

nennt man das r -te *Kreisteilungspolynom*. Seine Nullstellen sind genau die primitiven r -ten Einheitswurzeln.

- Ein Polynom f heißt *irreduzibel*, wenn es sich nicht in ein Produkt zweier nicht-konstanter Polynome von kleinerem Grad faktorisieren lässt.
- Für jede Primzahl p ist \mathbb{Z}_p ein endlicher Körper mit p Elementen. Weiterhin ist für jedes irreduzible Polynom $h(x) \in \mathbb{Z}_p[x]$ vom Grad d der Restklassenring $\mathbb{Z}_p[x]/\langle h(x) \rangle$ ebenfalls ein endlicher Körper mit p^d Elementen.
- Sei $a \in \mathbb{Z}$ und $n \in \mathbb{N}$. Die *Ordnung von a modulo n* (kurz $\text{ord}_n(a)$) ist das kleinste $k \in \mathbb{N}$ mit $a^k \equiv 1 \pmod{n}$.
- Das r -te Kreisteilungspolynom teilt $x^r - 1$ und zerfällt über $\mathbb{Z}_p[x]$ in irreduzible Faktoren vom Grad $\text{ord}_r(p)$ (siehe [8]).
- Mit $\log(n) := \log_2(n)$ bezeichnen wir den Logarithmus zur Basis 2. Für $x \in \mathbb{R}$ bezeichnen wir mit $\lfloor x \rfloor$ die größte ganze Zahl, die kleiner oder gleich x ist. Analog ist $\lceil x \rceil$ die kleinste ganze Zahl, die größer oder gleich x ist. Es gilt die Abschätzung $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$.
- Sei $a \in \mathbb{Z}$ und p prim, dann gilt $a^p \equiv a \pmod{p}$ (Kleiner Satz von Fermat).
- Sei R ein kommutativer Ring mit 1 (etwa \mathbb{Z} oder $\mathbb{Z}[x]$). Dann gilt für alle $a, b \in R$ und alle $n \in \mathbb{N}$ der Binomische Lehrsatz:

$$(a + b)^n = \sum_{i=0}^n \binom{n}{i} a^{n-i} b^i.$$

- Wir schreiben $f(x) \equiv g(x) \pmod{h(x), n}$, falls $f(x) = g(x)$ im Ring $\mathbb{Z}_n[x]/\langle h(x) \rangle$ gilt.
- Gilt $a \equiv b \pmod{n}$, so gilt auch $a \equiv b \pmod{d}$ für jeden Teiler d von n . Die analoge Aussage für Polynome gilt ebenfalls: Ist $f(x) \equiv g(x) \pmod{h(x), n}$, so ist auch $f(x) \equiv g(x) \pmod{q(x), n}$ für alle Teiler $q(x)$ von $h(x)$.
- Wir schreiben $\tilde{O}(\log^k(n))$ als Abkürzung für $O(\log^k(n) \cdot f(\log(\log(n)))) = O(\log^{k+\epsilon}(n))$ für ein Polynom $f(n)$. Eine polynomielle Schranke in der \tilde{O} -Notation ergibt natürlich auch eine polynomielle Schranke im klassischen Sinne (denn $\log(\log(n))$ lässt sich durch $\log(n)$ abschätzen).
- Teilt eine Primzahl p ein Produkt $a \cdot b$, so teilt p bereits a oder b . (Beweisskizze: $p|ab$, d.h. $kp = ab$ für ein $k \in \mathbb{Z}$. $p \nmid a$, dann ist $\text{ggT}(a, p) = 1$, also $1 = sa + tp$ für $s, t \in \mathbb{Z}$ (nach Euklidischem Algorithmus); Also $b = bsa + btp = skp + btp = p \cdot (sk + bt)$, also $p|b$).

4 Der AKS-Algorithmus

4.1 Idee

Die Kernidee ist die folgende Charakterisierung von Primzahlen.

Lemma 4.1. *Seien $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ wobei a und n teilerfremd sind. Dann ist n genau dann eine Primzahl, wenn $(x + a)^n = x^n + a$ in \mathbb{Z}_n gilt.*

Beweis. Nach dem Binomischen Lehrsatz ist für $0 < i < n$ der Koeffizient von x^i in $(x+a)^n$ gleich $\binom{n}{i}a^{n-i}$.

Sei $n = p$ prim. Es gilt

$$\binom{p}{i} = \frac{p \cdot (p-1) \cdot \dots \cdot (p-i+1)}{i \cdot (i-1) \cdot \dots \cdot 1}.$$

Also teilt p das Produkt $\binom{p}{i} \cdot i \cdot (i-1) \cdot \dots \cdot 1$. Da p prim ist, teilt p bereits einen der Faktoren. Keiner der Faktoren $1, \dots, i \leq p-1$ ist durch p teilbar, also ist p ein Teiler von $\binom{p}{i}$ und damit ist $\binom{p}{i}$ gleich 0 in $\mathbb{Z}_p[x]$. Der Koeffizient von x^p ist 1 und der von x^0 ist a^p . Nach dem kleinen Satz von Fermat ist $a^p = a$ in $\mathbb{Z}_p[x]$. (Diese Richtung des Beweises ist ein Spezialfall eines bekannten Satzes aus der Algebra: In endlichen kommutativen Ringen der Charakteristik p ist die Abbildung $x \mapsto x^p$ ein Ringendomorphismus, der sogenannte *Frobeniushomomorphismus*).

Sei n zusammengesetzt, dann existiert ein Primfaktor $p < n$ von n . Dieses p teilt im Koeffizienten

$$\binom{n}{p} = \frac{n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-(p-1))}{p \cdot (p-1) \cdot \dots \cdot 1}$$

das n im Zähler sowie das p im Nenner und sonst keinen weiteren Term. Sei p^k die größte p -Potenz, die n teilt, dann ist p^{k-1} die größte Potenz von p , die $\binom{n}{p}$ teilt. Also ist $\binom{n}{p}$ nicht durch n teilbar (und damit ungleich 0 modulo n), denn sonst wäre wegen $p^k | n$ und $n | \binom{n}{p}$ auch $p^k | \binom{n}{p}$. Ausserdem gilt wegen $1 = \text{ggT}(a, n) = \text{ggT}(a, p)$, dass auch $\text{ggT}(a^{n-p}, p^k) = 1$ ist. Insgesamt folgt, dass $\binom{n}{p}a^{n-p}$ ungleich 0 ist in \mathbb{Z}_n .

□

Lemma 4.1 stellt selbst bereits einen einfachen Primzahltest dar, der aber nicht effizient ist, da auf der linken Seite der Gleichung alle n Koeffizienten ausgewertet werden müssen.

Die Anzahl der Koeffizienten lässt sich reduzieren, wenn man die Gleichung modulo einem Polynom der Form $x^r - 1$ betrachtet (denn es gilt $x^r = 1 \pmod{x^r - 1}$, d.h. alle Exponenten von x lassen sich modulo r reduzieren). Man testet also ob die Gleichung

$$(x+a)^n \equiv x^n + a \pmod{x^r - 1, n}$$

gilt. Jede Primzahl, die die ursprüngliche Gleichung erfüllt, erfüllt nun natürlich auch die neue (für jedes r). Das Problem ist, ähnlich wie beim Fermat-Test, dass es eventuell Zahlen gibt, die nicht prim sind, die Gleichung aber trotzdem erfüllen. Es stellt sich aber heraus, dass sich dieser Ansatz trotzdem zu einem korrekten Primzahltest erweitern lässt.

4.2 Algorithmus

Eingabe: $n \in \mathbb{N}$ mit $n > 1$.

1. Falls $n = a^b$ für $a, b \in \mathbb{N}$ und $a, b > 1$, gebe ZUSAMMENGESETZT aus.
2. Finde das kleinste r , so dass $\text{ord}_r(n) > \log^2 n$.
3. Falls $1 < \text{ggT}(a, n) < n$ für ein $a \leq r$, gebe ZUSAMMENGESETZT aus.
4. Falls $n \leq r$, gebe PRIM aus.
5. Für $a = 1$ bis $\lfloor \sqrt{\varphi(r)} \cdot \log(n) \rfloor$

Falls $(x+a)^n \not\equiv x^n + a \pmod{x^r - 1, n}$, gebe ZUSAMMENGESETZT aus.

6. Gebe PRIM aus.

4.3 Korrektheit

Wir beweisen die Korrektheit in zwei Schritten. Wenn PRIM ausgegeben wird, dann ist n tatsächlich prim und falls n prim ist, so wird auch PRIM ausgegeben.

Satz 4.2. *Falls n prim ist, so gibt der Algorithmus PRIM aus.*

Beweis. Schritt 1 und 3 können niemals ZUSAMMENGESETZT ausgeben, da die Bedingungen dafür für Primzahlen niemals erfüllt sind. Nach Lemma 4.1 kann auch in Schritt 5 niemals ZUSAMMENGESETZT ausgegeben werden, denn die Gleichung ist für alle Primzahlen immer erfüllt. Also wird entweder im 4. oder im 6. Schritt PRIM ausgegeben. \square

Falls in Schritt 4 PRIM ausgegeben wird, so muss n tatsächlich prim sein, denn anderenfalls wäre in Schritt 3 ein Faktor gefunden worden. Es bleibt zu zeigen, dass, falls in Schritt 6 PRIM ausgegeben wird, n prim sein muss. Für den restlichen Beweis gehen wir also davon aus, dass in Schritt 6 PRIM ausgegeben wird, insbesondere heisst das, dass in Schritt 5 *nicht* ZUSAMMENGESETZT ausgegeben wird, dass $r < n$ ist (wegen Schritt 4), dass n und r teilerfremd sind (wegen Schritt 3) und dass n keine echte Potenz ist (wegen Schritt 1).

Da $\text{ord}_r(n) > 1$ ist (nach Wahl von r), muss es einen Primteiler p von n geben, für den bereits $\text{ord}_r(p) > 1$ gilt (denn hätten alle Primteiler Ordnung 1, so natürlich auch n). Weiterhin muss dieses p größer als r sein, denn sonst hätte der Algorithmus in Schritt 3 bereits ZUSAMMENGESETZT ausgegeben. Da n und r teilerfremd sind (s.o.), sind auch p und r teilerfremd und damit sind n und p in der multiplikativen Gruppe \mathbb{Z}_r^* enthalten.

Im folgenden bezeichnen wir die obere Grenze der Schleife aus Schritt 5 zur Vereinfachung der Notation mit $\ell := \lfloor \sqrt{\varphi(r)} \cdot \log(n) \rfloor$.

Definition 4.3. *Sei $f(x)$ ein Polynom und $m \in \mathbb{N}$ eine natürliche Zahl. Man nennt m introspektiv für $f(x)$, falls $f(x)^m \equiv f(x^m) \pmod{x^r - 1, n}$ gilt.*

Da der Algorithmus in Schritt 5 nicht ZUSAMMENGESETZT ausgibt, gilt

$$(x + a)^n \equiv x^n + a \pmod{x^r - 1, n}$$

und da n durch p teilbar ist folgt

$$(x + a)^n \equiv x^n + a \pmod{x^r - 1, p}.$$

Ausserdem wissen wir nach Lemma 4.1, dass, da p prim ist, auch

$$(x + a)^p \equiv x^p + a \pmod{x^r - 1, p}$$

gilt.

Betrachte das Polynom $(x + a)^{\frac{n}{p}}$. Dieses lässt sich ausmultiplizieren, d.h. es gibt $b_i \in \mathbb{Z}_p$, so dass $(x + a)^{\frac{n}{p}} = \sum b_i x^i$ gilt. Da $x \mapsto x^p$ ein Homomorphismus ist (vgl. Satz 4.1) und n durch p teilbar ist, gilt

$$x^n + a \equiv (x + a)^n \equiv ((x + a)^{\frac{n}{p}})^p \equiv \left(\sum b_i x^i\right)^p \equiv \sum b_i^p (x^i)^p \equiv \sum b_i x^{ip} \pmod{x^r - 1, p}.$$

Durch Koeffizientenvergleich der beiden Polynome erhält man $b_0 = a$, $b_{\frac{n}{p}} = 1$ und $b_i = 0$ für alle anderen Koeffizienten. Zusammen folgt also

$$(x + a)^{\frac{n}{p}} \equiv x^{\frac{n}{p}} + a \pmod{x^r - 1, p}$$

Die natürlichen Zahlen n , p und $\frac{n}{p}$ sind also introspektiv für $f(x) = x + a$ (für alle $0 \leq a \leq \ell$).

Wir benötigen zunächst zwei wichtige strukturelle Abschlusseigenschaften introspektiver Zahlen.

Lemma 4.4.

1. Die Menge der introspektiven Zahlen für ein Polynom ist multiplikativ abgeschlossen.
2. Die Menge der Polynome, für die m introspektiv ist, ist multiplikativ abgeschlossen.

Beweis. 1. Seien m und m' introspektiv für $f(x)$. Dann gilt

$$f(x)^{mm'} \equiv f(x^m)^{m'} \pmod{x^r - 1, p},$$

da m introspektiv für f ist. Ersetzt man x durch x^m , so erhält man

$$f(x^m)^{m'} \equiv f((x^m)^{m'}) \equiv f(x^{mm'}) \pmod{x^{mr} - 1, p}.$$

Da $x^r - 1$ ein Teiler von $x^{mr} - 1$ ist, folgt (siehe Kapitel 3):

$$f(x)^{mm'} \equiv f(x^{mm'}) \pmod{x^r - 1, p}.$$

2. Sei m introspektiv für $f(x)$ und $g(x)$, dann gilt:

$$(f(x) \cdot g(x))^m \equiv f(x)^m \cdot g(x)^m \equiv f(x^m) \cdot g(x^m) \pmod{x^r - 1, p}$$

□

Da, wie wir oben gesehen haben, $\frac{n}{p}$ und p introspektiv sind für alle $x + a$, $0 \leq a \leq \ell$, folgt mit Lemma 4.4, dass *jede* Zahl aus $I := \{(\frac{n}{p})^i \cdot p^j \mid i, j \geq 0\}$ introspektiv ist für *jedes* Polynom aus $P := \{\prod_{a=0}^{\ell} (x + a)^{e_a} \mid e_a \geq 0\}$.

Wir wollen nun zwei Gruppen auf Grundlage der Mengen I und P definieren.

Mit G bezeichnen wir die Menge der Restklassen der Elemente von I modulo r . Diese Menge bildet eine Gruppe (mit der Multiplikation modulo r) und wird, nach Definition von I , von n und p erzeugt (denn wegen $\text{ggT}(n, r) = \text{ggT}(p, r) = 1$ folgt $\text{ggT}(a, r) = 1$ für alle $a \in I$) und ist eine Untergruppe von \mathbb{Z}_r^* . Die Anzahl der Elemente von G bezeichnen wir mit $t := |G|$. Da die von n erzeugte Untergruppe von G genau $\text{ord}_r(n)$ Elemente hat gilt $\text{ord}_r(n) \leq t$. Da (per Wahl von r) ausserdem $\log^2(n) < \text{ord}_r(n)$ gilt, folgt $t > \log^2(n)$.

Sei $h(x)$ ein irreduzibler Faktor des r -ten Kreisteilungspolynoms $\Phi_r(x)$, dann hat $h(x)$ Grad $\text{ord}_r(p) > 1$ (siehe Kapitel 3). Der Restklassenring $\mathbb{Z}_p[x]/\langle h(x) \rangle =: \mathbb{F}$ ist also ein Körper. Mit \mathcal{G} bezeichnen wir die Untergruppe von \mathbb{F}^* , die von den Elementen $x, x+1, x+2, \dots, x+\ell$ erzeugt wird (diese sind alle ungleich 0, da der Grad von $h(x)$ größer als 1 ist). Die Gruppe \mathcal{G} ist also die „Reduktion von P modulo $h(x)$ und p “ und G ist die „Reduktion von I modulo r “.

Als nächstes werden wir die Anzahl der Elemente von \mathcal{G} nach oben und nach unten abschätzen und somit zu einem Widerspruch gelangen, der den Beweis abschliesst.

Lemma 4.5. \mathcal{G} enthält mindestens $\binom{t+\ell}{t-1}$ Elemente.

Beweis. Wir zeigen zunächst per Widerspruch, dass zwei verschiedene Elemente $f(x) \neq g(x)$ von Grad $< t$ aus P verschiedene Restklassen in \mathbb{F}^* bilden. Angenommen es sei also

$f(x) = g(x)$ in \mathbb{F} . Dann gilt natürlich auch $f(x)^m = g(x)^m$ in \mathbb{F} für ein beliebiges $m \in I$. Nach Definition von I und P ist m introspektiv für $f(x)$ und $g(x)$, d.h.

$$f(x)^m \equiv f(x^m) \text{ und } g(x)^m \equiv g(x^m) \pmod{x^r - 1, p}.$$

Da $h(x)$ ein Teiler von $x^r - 1$ ist, folgt per Definition von \mathbb{F} als Restklassenring modulo $h(x)$, dass auch $f(x^m) = g(x^m)$ in \mathbb{F} gilt (vgl. Kapitel 3). x^m ist also Nullstelle des Polynoms $Q(y) := f(y) - g(y)$ aus $\mathbb{F}[y]$ ⁴ (für jedes $m \in G$).

Da $m \in G$ beliebig war, hat $Q(y)$ also $|G| = t$ verschiedene Nullstellen, der Grad von $Q(y)$ muss also mindestens t sein (da \mathbb{F} ein Körper ist). Per Definition von $Q(y)$ als Differenz zweier Polynome vom Grad $< t$ hat aber auch Q einen kleineren Grad als t . Die Annahme, dass $f(x) = g(x)$ in \mathbb{F} gilt, muss also falsch gewesen sein.

Weiterhin sind die Restklassen der Elemente $1 \leq i \leq \ell$ in \mathbb{Z}_p wegen

$$\ell = \lfloor \sqrt{\phi(r)} \log(n) \rfloor < \lfloor \sqrt{r} \log(n) \rfloor < \sqrt{r} \log(n) < r < p$$

alle paarweise verschieden und damit auch die Elemente $x, x+1, x+2, \dots, x+\ell$ in \mathbb{F} . Da der Grad von $h(x)$ größer als 1 ist, sind diese Restklassen ausserdem alle ungleich 0, d.h. es gibt mindestens $\ell+1$ verschiedene Polynome vom Grad 1 in \mathcal{G} .

Insgesamt gilt also: Je zwei verschiedene Polynome aus P vom Grad kleiner t bilden verschiedene Restklassen in \mathcal{G} . Die Anzahl solcher Polynome vom Grad kleiner t entspricht (nach Definition von P) der Anzahl der Möglichkeiten $t-1$ Elemente aus der $\ell+2$ elementigen Menge $\{1, x, x+1, \dots, x+\ell\}$ auszuwählen, mit Wiederholung, ohne Berücksichtigung der Reihenfolge. Hierfür gilt (siehe [15]) die Formel

$$\binom{(\ell+2) + (t-1) - 1}{t-1} = \binom{\ell+t}{t-1}.$$

□

Nun zur Abschätzung nach oben.

Lemma 4.6. *Ist n keine Potenz von p , so enthält \mathcal{G} höchstens $n^{\sqrt{t}}$ Elemente.*

Beweis. Mit \hat{I} bezeichnen wir die folgende Einschränkung von I :

$$\hat{I} := \left\{ \left(\frac{n}{p} \right)^i \cdot p^j \mid 0 \leq i, j \leq \lfloor \sqrt{t} \rfloor \right\}$$

Falls n keine Potenz von p ist, so gilt $|\hat{I}| = (\lfloor \sqrt{t} \rfloor + 1)^2 > (\sqrt{t})^2 = t = |G|$. Also müssen mindestens zwei Elemente aus \hat{I} gleich sein modulo r , d.h. die gleiche Restklassen in G bilden, etwa m_1 und m_2 , wobei wir ohne Einschränkung annehmen, dass $m_1 > m_2$. Dann gilt auch

$$x^{m_1} = x^{m_2} \pmod{x^r - 1}$$

(denn aus $m_1 = m_2 + k \cdot r$ für ein k , folgt $x^{m_1} \equiv x^{m_2} \cdot x^{rk} \equiv x^{m_2} \cdot 1 \pmod{x^r - 1}$). Sei nun $f(x)$ ein beliebiges Polynom aus P , dann sind m_1 und m_2 (nach Definition von I) introspektiv für $f(x)$ und es gilt

$$f(x)^{m_1} \equiv f(x^{m_1}) \equiv f(x^{m_2}) \equiv f(x)^{m_2} \pmod{x^r - 1, p}$$

und damit auch

$$f(x)^{m_1} = f(x)^{m_2} \pmod{h(x), p}.$$

⁴Hierbei handelt es sich also um Polynome in y , deren Koeffizienten Elemente aus \mathbb{F} , d.h. Polynome in x sind.

Also sind $f(x)^{m_1}$ und $f(x)^{m_2}$ in \mathbb{F} in der gleichen Restklasse. Analog zum Beweis für die untere Schranke von \mathcal{G} gilt nun: $f(x)$ ist eine Nullstelle von $Q(y) := y^{m_1} - y^{m_2} \in \mathbb{F}[y]$. Da $f(x)$ beliebig gewählt war, hat Q mindestens $|\mathcal{G}|$ verschiedene Nullstellen in \mathbb{F} . Der Grad von Q ist aber gleich m_1 und per Definition von \hat{I} gilt

$$|\mathcal{G}| \leq \text{Anzahl Nullstellen von } Q \leq m_1 \leq \left(\frac{n}{p}\right)^{\lfloor \sqrt{t} \rfloor} \cdot p^{\lfloor \sqrt{t} \rfloor} = n^{\lfloor \sqrt{t} \rfloor} \leq n^{\sqrt{t}}.$$

□

Wir können nun den zweiten Schritt im Korrektheitsbeweis des AKS-Algorithmus zeigen.

Satz 4.7. *Falls der Algorithmus PRIM ausgibt, so ist n prim.*

Beweis. Angenommen der Algorithmus gibt PRIM aus.

Mit der gerade gemachten Abschätzungen erhält man:

$$|\mathcal{G}| \stackrel{4.5}{\geq} \binom{t+\ell}{t-1} \stackrel{(1)}{\geq} \binom{\ell+1+\lfloor \sqrt{t} \log(n) \rfloor}{\lfloor \sqrt{t} \log(n) \rfloor} \stackrel{(2)}{\geq} \binom{2\lfloor \sqrt{t} \log(n) \rfloor + 1}{\lfloor \sqrt{t} \log(n) \rfloor} \stackrel{(3),(4)}{>} 2^{\lfloor \sqrt{t} \log(n) \rfloor + 1} \stackrel{(5)}{\geq} n^{\sqrt{t}}$$

- (1) Wie in der Definition von G bereits erwähnt, gilt $t > \log^2(n)$, also auch $\sqrt{t} > \log(n)$ und damit $t = \sqrt{t} \cdot \sqrt{t} > \sqrt{t} \log(n)$.
- (2) G ist eine Untergruppe von \mathbb{Z}_r^* , also ist $t = |G| \leq \varphi(r)$, deshalb gilt $\ell = \lfloor \sqrt{\varphi(r)} \log(n) \rfloor \geq \lfloor \sqrt{t} \log(n) \rfloor$.
- (3) Wegen $\sqrt{t} > \log(n)$ und $n \geq 2$ folgt $\lfloor \sqrt{t} \log(n) \rfloor > \lfloor \log^2(n) \rfloor \geq 1$.
- (4) $\binom{2k+1}{k} = \frac{(2k+1) \cdot 2k \cdot \dots \cdot k+1}{(k+1) \cdot k \cdot \dots \cdot 2 \cdot 1} = \frac{2k+1}{k+1} \cdot \frac{2k}{k} \cdot \dots \cdot \frac{k+2}{2} \cdot \frac{k+1}{1} \geq 2^{k+1}$
- (5) $\lfloor \sqrt{t} \log(n) \rfloor + 1 \geq \sqrt{t} \log(n)$ und $2^{\log(n)} = n$, also gilt $2^{\lfloor \sqrt{t} \log(n) \rfloor + 1} \leq 2^{\log(n) \sqrt{t}} \leq n^{\sqrt{t}}$.

Nach Lemma 4.6 muss n also eine Potenz von p sein, denn anderenfalls wäre $n^{\sqrt{t}} < |\mathcal{G}| \leq n^{\sqrt{t}}$, d.h. $n = p^k$ für ein $k \in \mathbb{N}$. Wäre $k \geq 2$, so hätte der Algorithmus n bereits im ersten Schritt als echte Potenz identifiziert und ZUSAMMENGESETZT ausgegeben, also muss $k = 1$ gelten, d.h. $n = p$.

□

4.4 Terminierung

Bisher wurde gezeigt, dass, falls der Algorithmus PRIM ausgibt, n tatsächlich prim ist und falls er ZUSAMMENGESETZT ausgibt, n nicht prim sein kann. Es bleibt zu zeigen, dass der Algorithmus tatsächlich stets etwas ausgibt, d.h. für jede mögliche Eingabe n terminiert. Der einzige kritische Schritt ist natürlich der zweite, in dem das kleinste r gefunden wird, dessen Ordnung größer als $\log^2(n)$ sein soll.

Dass man so ein r immer finden kann (und dass man es „schnell“ finden kann) zeigt der folgende Satz.

Satz 4.8. *Für jedes $n \in \mathbb{N}$ existiert ein $r \leq \max\{3, \lceil \log^5(n) \rceil\}$ mit $\text{ord}_r(n) > \log^2(n)$.*

Zunächst ein kurzes Lemma, das wir aber nicht beweisen wollen. Ein einfacher Beweis findet sich in [8] und der Originalbeweis in [5].

Lemma 4.9. Für $m \geq 7$ ist das kleinste gemeinsame Vielfache der ersten m natürlichen Zahlen mindestens 2^m .

Beweis von Satz 4.8. Für $n = 2$ erfüllt $r = 3$ die Bedingungen, denn $\text{ord}_3(2) = 2 > 1 = \log^2(2)$. Sei also $n > 2$, dann ist $\lceil \log^5(n) \rceil > 10 \geq 7$ und wir können Lemma 4.9 anwenden. Sei $\{r_1, \dots, r_t\}$ die Menge der Zahlen r_i , die n teilen oder für die $\text{ord}_{r_i}(n) \leq \log^2(n)$ gilt. Jedes r_i teilt also entweder n oder $n^k - 1$ für ein $1 \leq k \leq \lfloor \log^2(n) \rfloor$ (denn $n^k \equiv 1 \pmod{r_i}$), d.h. jedes r_i teilt das folgende Produkt:

$$\begin{aligned}
& n \cdot \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1) \\
& < n \cdot \prod_{i=1}^{\lfloor \log^2(n) \rfloor} n^i \\
& = n \cdot n^{\sum_{i=1}^{\lfloor \log^2(n) \rfloor} i} \\
& = n^{1 + \frac{1}{2}(\lfloor \log^4(n) \rfloor + \lfloor \log^2(n) \rfloor)} \\
& < n^{1 + \lfloor \log^4(n) \rfloor} \\
& \leq n^{\log^4(n)} \\
& = (2^{\log(n)})^{\log^4(n)} \\
& = 2^{\log^5(n)} \\
& \stackrel{4.9}{\leq} \text{kgV}(1, \dots, \lceil \log^5(n) \rceil)
\end{aligned}$$

Angenommen jedes $1 \leq s \leq \lceil \log^5(n) \rceil$ ist in $\{r_1, \dots, r_t\}$. Alle r_i teilen das obige Produkt und dieses ist kleiner als $2^{\lceil \log^5(n) \rceil}$. Nach Lemma 4.9 ist die kleinste Zahl, die von allen solchen s geteilt wird aber größer oder gleich $2^{\lceil \log^5(n) \rceil}$. Also existiert ein $s \leq \lceil \log^5(n) \rceil$, welches nicht in $\{r_1, \dots, r_t\}$ enthalten ist.

Falls $\text{ggT}(s, n) = 1$ so muss $\text{ord}_s(n) > \log^2(n)$ gelten (anderenfalls wäre $s \in \{r_1, \dots, r_t\}$) und wir sind fertig: Wähle $r := s$. Falls s und n einen gemeinsamen Teiler haben, so ist dieser gleich einem der r_i . Da n selbst aber nicht durch s teilbar ist ($s \notin \{r_1, \dots, r_t\}$) folgt $r := \frac{s}{\text{ggT}(s, n)} \notin \{r_1, \dots, r_t\}$, also $\text{ord}_r(n) > \log^2(n)$.

□

4.5 Laufzeit

Addition, Multiplikation und Division zweier m Bit Zahlen lässt sich in $\tilde{O}(m)$ Operationen durchführen. Diese Operationen lassen sich auf Polynome übertragen: Addition, Multiplikation und Division zweier Polynome vom Grad höchstens d und Koeffizienten der Länge m , lassen sich in $\tilde{O}(d \cdot m)$ durchführen. Für Details dazu siehe [4] und [8].

Wir werden nun Schritt für Schritt den Algorithmus analysieren.

Satz 4.10. Der AKS-Algorithmus kann mit einer Laufzeit von $\tilde{O}(\log^{\frac{21}{2}}(n))$ implementiert werden.

Beweis. 1. Ob eine Zahl $n > 1$ eine echte Potenz ist, lässt sich in $\tilde{O}(\log^3(n))$ testen [4].

2. Wir suchen ein r mit $\text{ord}_r(n) > \log^2(n)$. Um für ein konkretes r diese Bedingung zu testen, können einfach die Werte n^i für $1 \leq i \leq \log^2(n)$ ausprobiert werden. Dafür sind also jeweils $O(\log^2(n))$ Multiplikationen modulo r nötig, wovon jede einzelne in $\tilde{O}(\log(r))$

durchführbar ist. Nach Lemma 4.8 werden wir spätestens für $r = \lceil \log^5(n) \rceil$ fündig, die Gesamtkomplexität von Schritt 2 beträgt also $\tilde{O}(\log^7(n))$.

3. Im dritten Schritt wird r mal der größte gemeinsame Teiler ausgerechnet. Dies ist effizient möglich (z.B. mit dem Euklidischen Algorithmus) in $O(\log(n))$ für jedes r . Insgesamt also $O(r \log(n)) = O(\log^6(n))$.
4. Dieser Schritt ist offensichtlich in $O(\log(n))$ durchführbar (durch Bitvergleich).
5. Wir testen $\ell = \lfloor \sqrt{\varphi(r)} \log(n) \rfloor$ Gleichungen. Jede einzelne benötigt $\tilde{O}(\log(n))$ Multiplikationen von Polynomen des Grades r mit Koeffizienten der Größe $O(\log(n))$. Jede einzelne Gleichung lässt sich also in $\tilde{O}(r \log^2(n))$ Schritten testen. Insgesamt erhält man

$$\tilde{O}(r \sqrt{\varphi(r)} \log^3(n)) \stackrel{\sqrt{\varphi(r)} \leq \sqrt{r}}{=} \tilde{O}(r^{\frac{3}{2}} \log^3(n)) \stackrel{4.8}{=} \tilde{O}(\log^{5 \cdot \frac{3}{2}}(n) \log^3(n)) = \tilde{O}(\log^{\frac{21}{2}}(n)).$$

□

Die Gesamtlaufzeit wird also durch den letzten Schritt bestimmt und beträgt damit im schlechtesten Fall $\tilde{O}(\log^{\frac{21}{2}}(n))$.

5 Zusammenfassung

Wir haben in dieser Ausarbeitung den *AKS-Algorithmus* kennen gelernt sowie seine Korrektheit bewiesen und seine Laufzeit abgeschätzt. Das Hauptresultat ist der folgende Satz:

Satz 5.1 (AKS). *Es existiert ein deterministischer Algorithmus, der in $O(\log^{10.5+\epsilon}(n))$ Operationen entscheidet ob eine natürliche Zahl eine Primzahl oder eine zusammengesetzte Zahl ist. Der Algorithmus funktioniert für jede Primzahl, ist nicht probabilistisch und seine Korrektheit lässt sich ohne unbewiesene Vermutungen beweisen. Kurz: Das Primzahltestproblem PRIMES ist in der Komplexitätsklasse P.*

Es stellt sich die Frage, ob das Faktorisierungsproblem für natürliche Zahlen ebenfalls zur Klasse P gehört. Diese Frage ist bis heute (2009) offen und eine positive Antwort hätte gravierende Folgen für die Kryptographie. Man beachte, dass aus einem effizienten Primzahltest keineswegs ein effizientes Faktorisierungsverfahren folgt!

Literatur

- [1] Andrew Granville. It is easy to determine whether a given integer is prime. *Bulletin of the American Mathematical Society*, vol. 42, no. 1, pages 3–38, 2004.
- [2] Daniel J. Bernstein. Detecting perfect powers in essentially linear time. *Mathematics of Computation*, vol. 67, no. 223, pages 1253–1283, 1998.
- [3] Daniel J. Bernstein. Proving primality after Agrawal-Kayal-Saxena. 2003.
- [4] Joachim von zur Gathen,, Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.
- [5] M. Nair. On Chebyshev-Type Inequalities for Primes. *Mathematical Association of America*, 1982.
- [6] Manindra Agrawal, Neeraj Kayal, Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, vol. 160, pages 781–793, 2004.
- [7] Michael Artin. *Algebra*. Birkhäuser Verlag, 1998.
- [8] Michiel Smid. Primality testing in polynomial time. 2003.
- [9] Nitin Saxena. Morphisms of Rings and Applications to Complexity. *Ph.D. thesis*, 2006.
- [10] Otto Forster. *Algorithmische Zahlentheorie*. vieweg, 1996.
- [11] Prof. Dr. Aloys Krieg. *Elementare Zahlentheorie (Skript zur Vorlesung)*. Lehrstuhl A für Mathematik: Analysis und Zahlentheorie, RWTH Aachen, 2006.
- [12] Prof. Dr. Aloys Krieg. *Algebraische Zahlentheorie (Skript zur Vorlesung)*. Lehrstuhl A für Mathematik: Analysis und Zahlentheorie, RWTH Aachen, 2008.
- [13] Prof. Dr. Eva Zerz. *Computeralgebra (Skript zur Vorlesung)*. Lehrstuhl D für Mathematik, RWTH Aachen, 2008.
- [14] Song Y. Yan. *Number Theory for Computing*. Springer Verlag, 2000.
- [15] Angelika Steger. *Diskrete Strukturen 1: Kombinatorik, Graphentheorie, Algebra*. Springer, 2002.