Komplexität und Evaluation des AKS-Primzahltests

Salman Salman

3753924

UNIVERSITÄT
LEIPZIG

Bachelorarbeit

Lehrstuhl für Mathematik
und Informatik
Universität Leipzig

Betreuer: Prof. Dr. Andreas Maletti

Leipzig, den 25. März 2020

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abstract

Primzahlen haben in der Informatik, speziell im Anwendungsgebiet der Kryptographie, eine sehr hohe Relevanz für moderne kryptographische Systeme ist es von Wichtigkeit selbige schnell bestimmen zu können. Hierzu werden effiziente Algorithmen zur Lösung des mathematischen Primalitätsproblem benötigt. Das Primalitätsproblem umfasst die Frage um die Entscheidung, ob eine gegebene Zahl eine Primzahl ist oder nicht. Der erste deterministische Primzahltest in Polynomialzeit wurde von den indischen Informatikern Agrawal, Kayal, und Saxena vorgestellt. Der nach ihnen benannte AKS-Algorithmus wird im Rahmen dieser Arbeit repräsentiert, implementiert und evaluiert. Außerdem wird die Korrektheit des Algorithmus experimentell geprüft und mathematisch bewiesen.

1 Einleitung

1.1 Gliederung der Arbeit

Die Arbeit gliedert sich in vier Bestandteile, wobei der erste Teil die Einleitung darstellt. Dort wird ein kurzer Überblick in das Thema gegeben und die Zielsetzung der Arbeit definiert.

Das zweite Kapitel enthält die Grundlagen und die Definitionen. Hier werden Terminologie und Konzepte erläutert, die für das weitere Verständnis der Arbeit eine Rolle spielen.

Der Algorithmus und seine Korrektheit sind die Bestandteile des dritten Kapitels. Zuerst wird die Grundidee des Algorithmus erläutert, danach wird der Algorithmus in seiner ursprünglichen Form(wie im AKS-Paper) angegeben. Anschließend wird der Korrektheitsbeweis des Algorithmus geführt.

Im vierten Kapitel wird die Laufzeitanalyse des Algorithmus vorgestellt, dabei werden die Laufzeit der einzelnen Schritte (STEPS 1-6) des Algorithmus ausführlich analysiert.

1.2 Überblick

Eine Primzahl ist eine von 1 verschiedene natürliche Zahl, die keine Teiler außer 1 und sich selbst hat. Die Frage, ob eine Zahl eine Primzahl ist oder nicht, war schon im antiken Griechenland interessant. Euklid hat sich mit dieser Frage beschäftigt und beweisen, dass es unendlich viele Primzahlen gibt. Im 3. Jahrhundert v. Chr. hat der griechische Mathematiker Eratosthenes einen Algorithmus zur Bestimmung einer Liste aller Primzahlen kleiner oder gleich einer vorgegebenen Zahl vorgestellt, dieser Algorithmus ist jedoch zur Lösung des Primalitätsproblem nicht effizient, da die Generierung solcher Listen bei sehr großen Zahl zu aufwendig wird. Im 17. und 18. Jahrhundert haben Wissenschaftler versucht eine Formel für Primzahlen zu erfinden. Der französische Wissenschaftler Marin Mersenne hat eine Formel für die nach ihm sogenannten Mersenne-Zahlen entwickelt, 51 der bekannten Mersenne-Zahlen sind Primzahlen. Solche Zahlen heißen Mersenne-Primzahlen, die größte Primzahl heute $2^{82,589,933}1$ ist eine Mersenne-Primzahl. Ein anderer französischer Mathematiker Pierre de Fermat hat einen Satz zur Beschreibung der Eigenschaften von Primzahlen, aufgestellt. Auf seinen Satz bauen viele Primzahltests(einschließlich des AKS-Primzahltests) auf.

Im Laufe der Zeit wurden auch mehrere Algorithmen zur Lösung des Primalitätsproblems entwickelt, die entweder von unbewiesenen Hypothesen, wie zum Beispiel die verallgemeinerte Riemannsche Vermutung abhängig waren oder probabilistisch waren. Probabilistische Algorithmen sind randomisierte Algorithmen, die auch ein falsches Ergebnis liefern können. Ein bekannter probabilistischer Primzahltest ist der Miller-Rabin-Primzahltest, er kann das Primalitätsproblem in $O(klog^3n)$ lösen. Dennoch kann der beste nicht probabilistische Algorithmus (vor AKS) das Primalitätsproblem in $\Omega(\sqrt{n})$ Schritten lösen, wobei n die Eingabegröße ist(Anzahl der nötigen Bits, um die Zahl n zu repräsentieren). Sol-

cher Algorithmus braucht exponentielle Zeit, um das Problem zu lösen. In 2002 haben drei Informatiker Agrawal, Kayal, und Saxena den ersten deterministischen unbedingten Algorithmus entwickelt, der das Primalitätsproblem in Polynomialzeit lösen kann. Das heißt sie haben gezeigt, dass das Primalitätsproblem(PRIMES) zur Komplexitätsklasse P gehört.

1.3 Zielsetzung

In dieser Arbeit wird der AKS-Primzahltest Algorithmus aus dem Orginalpaper behandelt. Ziel dieser Arbeit ist die Korrektheit des AKS-Algorithmus zu testen und evaluieren. Dabei wird die Idee Algorithmus dargestellt und erläutert, danach kommt die Analyse der Korrektheit, da wird die Korrektheit zuerst mathematisch bewiesen, dafür wird ein bidirektionaler Beweis geführt, wo gezeigt wird, dass wenn der Algorithmus PRIME liefert, ist die Zahl auch eine Primzahl. Aber auch wenn die Eingabe eine Primzahl ist, liefert der Algorithmus PRIME. Dazu wird seine Korrektheit auf einem Rechner implementiert und getestet. Der Algorithmus ist dafür bekannt, das Primalitätsproblem in Polynomialzeit zu lösen, diese Eigenschaft wird auch zuerst mathematisch bewiesen. Schließlich soll die Polynomialzeit Eigenschaft durch eine Grafik experimentell bewiesen.

Um die Korrektheit zu demonstrieren, soll der Algorithmus in Python implementiert werden, dabei wird die Ausgabe des Algorithmus bei mehreren Zahlen(meistens große Zahlen) geprüft. Zudem soll auch die Laufzeit durch eine Grafik illustriert werden.

2 Definitionen und Grundlagen

In diesem Kapitel werden die essenziellen Begriffe und Theoreme aus der Zahlentheorie, der Algebra und der Komplexitätstheorie, die für den AKS-Algorithmus relevant sind, definiert. In diesem Kapitel handelt es sich darum, einen formalen Literaturhinweis zu haben, um später die hier definierten Begriffe und Theoreme zu referenzieren.

2.1 Zahlentheorie

Definition 2.1. Seien $a, b \in \mathbb{N}$. Der **größte gemeinsame Teiler** von a und b wird mit (a, b) bezeichnet, ist die größte positive Zahl n, sodass $n \mid a$ und $n \mid b$

Definition 2.2. Zwei Zahlen $a, b \in \mathbb{N}$ heißen genau dann **Teilerfremd**, wenn $(a, b) = 1$.

Definition 2.3. Seien n, m zwei natürliche Zahlen, dann heißt die kleinste positive natürliche Zahl, die sowohl ein Vielfaches von n, als auch von m **kleinstes gemeinsames Vielfaches** beider zahlen. Hier wird das kleinste gemeinsame Vielfaches mit **kgV** bezeichnet.

Definition 2.4. Seien $a, b, n \in \mathbb{N}$. a ist genau dann **kongruent** zu b modulo n, wenn $n \mid a - b$ gilt, dies wird mit $a = b \pmod{n}$ bezeichnet.

Definition 2.5. Seien $r, n \in \mathbb{N}$ mit $(n, r) = 1$, dann ist die **Ordnung** von n modulo r das kleinste k, sodass $n^k = 1(mod\ \mathrm{r})$. Die Ordnung wird mit $o_r(n)$ bezeichnet.

Definition 2.6. Sei $n \in \mathbb{N}$, die **Primzahlzerlegung** von n ist die Darstellung der Zahl als Produkt ihrer Primfaktoren

$$n = p_1^{e_1} p_2^{e_2} ... p_M^{e_M} \tag{1}$$

Wobei e_k die Vielfachheit der Primzahl p_k ist.

Definition 2.7. Sei $n \in \mathbb{N}$ mit $n > 1$. Die **eulersche Phi-Funktion** wird mit $\phi(n)$ bezeichnet, ist die Anzahl der Zahlen k, $1 \leq k \leq n$, sodass $(k, n) = 1$

Theorem 2.1 (**Satz von Euler**). *Seien $a, n \in \mathbb{N}$ teilerfremd,*
dann gilt $a^{\phi(n)} = 1(mod\ n)$.

Beweis. Seien $R = \{a \in \mathbb{N} \mid (a, n) = 1\} = \{a_1, a_2, ..., a_{\phi(n)}\}$ und $c \in R, c \leq n$.

Da $(c, n) = (a_i, n) = 1$, gilt auch $(c \cdot a_i, n) = 1$

und damit ist auch $a_i = c \cdot a_j (\mathrm{mod}\ n)$ wahr für ein $a_j \in R \Rightarrow a_i = a_j$. Das heißt es gilt:

$$\prod_{k=0}^{\phi(n)} c \cdot a_k = c^{\phi(n)} \prod_{k=0}^{\phi(n)} a_k = \prod_{k=0}^{\phi(n)} a_k (mod\, n) \tag{2}$$

$\Rightarrow c^{\phi(n)} = 1\ (\mathrm{mod}\ n)$.

□

Theorem 2.2 (**Binomischer Lehrsatz**). *Seien R ein kommutativer Ring und n eine natürliche Zahl, dann gilt für $a, b \in R$:*

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}. \tag{3}$$

Theorem 2.3. *Sei n eine Primzahl, dann gilt $\binom{n}{i} = 0(mod\ n)$.*

Beweis.

$$\binom{n}{i} = \frac{(n-i-1)\cdots(n-1)\cdot n}{i!} \tag{4}$$

Da n im Zähler steht und n eine Primzahl ist(nicht durch Zahlen im Nenner teilbar), muss die obere Gleichung (4) durch n teilbar sein. □

Theorem 2.4 (**Kleiner Satz von Fermat**). *Für eine Primzahl p und ein beliebiges $a \in \mathbb{Z}$ gilt:*

$$a^p = a \, mod \, p$$

Beweis. Die Aussage kann per Induktion über $a \in \mathbb{Z}_+$ gezeigt werden.

IA: $a = 0 \Rightarrow 0^p = 0 \, mod \, p$.

IH: Nun sei die Aussage für ein beliebiges $a \in \mathbb{Z}_+$ wahr.

IS: für a + 1 gilt nach dem binomischen Lehrsatz:

$$(a+1)^p - (a+1) = \Big[a^p + \binom{p}{1}a^{p-1} + ... + \binom{p}{p-1}a + 1\Big] - (a+1).$$

Für die Koeffizienten gilt:

$$\binom{p}{k} = \frac{p \cdot (p-1)\cdots(p-k+1)}{1 \cdot 2 \cdots k}$$

$\forall k$, $1 \leq k \leq p-1$, somit sind alle Koeffizienten(bis auf das erste und das letzte) durch p teilbar.

Also es gilt:

$$(a+1)^p - (a+1) = \Big[a^p + 1\Big] - (a+1) = a^p - a \, mod \, p.$$

und das ist nach der IH durch p teilbar.

□

Theorem 2.5 (**Gaußsche Summeformel**). *Sei n eine natürlich Zahl, dann gilt:*

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \tag{5}$$

Der Beweis ist durch Induktion trivial.

2.2 Algebra

Bemerkung: In diesem Kapitel wird davon ausgegangen, dass die grundlegenden Definitionen von Gruppen, Ringen, Körpern und ihre Eigenschaften dem Leser bekannt sind. Hier werden Theoreme und Begriffe der Algebra und Polynomentheorie, die später für den Korrektheitsbeweis von Bedeutung sind, vorgestellt und bewiesen.

Definition 2.8. Sei R ein Ring, dann ist der **Polynomring** $R[X]$ die Menge aller Polynome der Form $a_0 + a_1X + a_2X^2 + ... + a_nX^n$, wobei $a_0, a_1..., a_n \in R$.

Definition 2.9. Sei $f = (a_0, a_1, ...a_n) \in R[X]$, $f \neq 0$ ein Polynom, dann heißt

$$deg\, f = max\{i \mid a_i \neq 0\}$$

Definition 2.10. Sei I eine Teilmenge des Rings $(R, +, \cdot)$. I heißt **Ideal**, wenn gilt:

1. $0 \in I$.
2. $\forall a, b \in I$ ist $a + b \in I$.
3. $\forall a \in I$ und $r \in R$ ist $r \cdot a \in I$.
4. $\forall a \in I$ und $r \in R$ ist $a \cdot r \in I$.

Definition 2.11. ist $(R, +, \cdot)$ ein Ring und I ein Ideal von R, dann bildet die Menge $R/I = \{a + I \mid a \in R\}$ der Äquivalenzklassen modulo I mit folgenden Verknüpfungen einen Ring:

- $(a + I) + (b + I) = (a + b) + I$.
- $(a + I) \cdot (b + I) = (a \cdot b) + I$.

Diesen Ring nennt man den **Quotientenring**.

Theorem 2.6. *Sei R ein kommutativer Ring und $f \in R[X]$ ein Polynom, sodass $f \neq 0$. Ferner seien $a_1, a_2, ..., a_n \in R$ die Nullstellen/Wurzeln von f mit $a_i - a_j \in R^*$, $\forall i, j$, $1 \leq i, j, \leq n$. Dann ist $deg\, f \geq n$.*

Beweis. Beweis durch Widerspruch, angenommen $deg\, f = n - 1$ für ein beliebiges n. Sei $f(X) = b_{n-1}X^{n-1} + b_{n-2}X^{n-2} + ... + b_1X + b_0$ mit $b_{n-1} \neq 0$. Weiterhin seien $a_1, a_2, ..., a_n$ die Wurzeln des Polynoms $f(X)$, dann gilt $f(a_i) = 0$, $\forall 1 \leq i \leq n$, das heißt $b_{n-1}a_i^{n-1} + ... + b_1a_i + b_0 = 0$, $\forall 1 \leq i \leq n$.

Zudem sei $A = [a_{i,j}]_{1 \leq i,j \leq n}$ die Matrix der Wurzeln und $b = (b_1b_2...b_n) \neq 0$ der Vektor der Koeffizienten von $f(X)$. Dann gilt $b \cdot A = 0$, aber $det\, A \neq 0 \Rightarrow b = 0$ und dies ist ein Widerspruch(Da $b \neq 0$), daraus folgt es existiert $f(X) : deg\, f \geq n$ □

Theorem 2.7. $\binom{2n+1}{n} > 2^{n+1}, \forall n \geq 2$

Beweis. Induktion: Sei n = 2, offensichtlich gilt $\binom{5}{2} > 2^3$. Sei nun das obere Theorem für ein beliebiges $k > 2$ wahr.

für $n = k + 1$ ist das folgende zu zeigen:

$$\frac{(2k+3)!}{(k+2)! \cdot (k+1)!} > 2^{k+2}. \tag{6}$$

Die obere Ungleichung (6) lässt sich wie folgt umschreiben:

$$\underbrace{\frac{(2k+1)!}{(k+1)! \cdot k!}}_{nachIH > 2^{k+1}} \cdot \frac{(2k+2) \cdot (2k+3)}{(k+2) \cdot (k+1)} > 2^{k+1} \cdot 2. \tag{7}$$

Der erste Teil der . Multiplikation auf der linken ist nach der Induktionshypothese größer als 2^{k+1}. Es bleibt nur zu zeigen, dass der rechte Teil der Multiplikation größer als 2 ist, das heißt:

$$\frac{(2k+2) \cdot (2k+3)}{(k+2) \cdot (k+1)} = 2 \cdot \underbrace{\frac{(2k+3)}{k+2}}_{>1} > 2 \tag{8}$$

Somit ist die Ungleichung für $n = k + 1$ wahr. □

Theorem 2.8. *Seien $a, b, n \in \mathbb{N}$ Dann gilt:*

$$a^n - b^n = (a-b) \cdot \sum_{k=0}^{n-1} a^{n-1-k} b^k. \tag{9}$$

2.3 Zyklotomische Polynome

Definition 2.12. Sei n eine natürliche Zahl, die n-te **Einheitswurzel** wird mit ζ bezeichnet ist eine komplexe Zahl, sodass

$$\zeta^n = 1. \tag{10}$$

Z.B. 1 und -1 sind die quadratischen Einheitswurzeln, und 1, -1, i, -i, sind die Einheitswurzeln für $n = 4$.

Theorem 2.9. *Für jede natürliche Zahl n, gibt es n viele Einheitswurzeln, die durch:*

$$e^{2k\pi i/n} = cos(\frac{2k\pi}{n}) + isin(\frac{2k\pi}{n}) \tag{11}$$

definiert sind, mit $1 \leq k \leq n$.

Definition 2.13. Die **n-te primitive Einheitswurzel** ist jede Einheitswurzel ζ, für die $ord_{\mathbb{C}}(\zeta) = n$ gilt. Die Menge aller primitiven n-ten Einheitswurzeln wird mit **P(n)** bezeichnet.

Definition 2.14. Zyklotomisches Polynom: das n-te zyklotomische Polynom Φ_n ist durch:

$$\begin{aligned} \Phi_n(x) &= \prod_{\zeta \in P(n)} (x - \zeta) \\ &= \prod_{\substack{1 \leq k \leq n \\ (k,n)=1}} \left(x - \zeta^k\right). \end{aligned} \tag{12}$$

definiert. Dabei ist $P(n)$ die Menge aller primitiven n-ten Einheitswurzeln aus der Definition (2.13).

Theorem 2.10. *Für jede natürliche Zahl n . gilt:*

$$x^n - 1 = \prod_{d|n} \Phi_d(n). \tag{13}$$

Für den Beweis, siehe (Fields and Cyclotomic Polynomials)

Theorem 2.11. *Für alle* $n \geq 1$ *existieren genau* $\phi(n)$ *n-te primitive Einheitswurzeln.*

Definition 2.15. Sei $\mathbb{K}$ ein Körper, und $a \in \mathbb{K}^x$. Dann ist **Ordnung** $ord_{\mathbb{K}}(a)$, die kleinste natürliche Zahl k, für die $a^k = 1$ gilt, wenn ein solches k nicht existiert, dann hat a eine unendliche Ordnung.

Theorem 2.12. *Sei $\mathbb{K}$ ein Körper mit m Elementen, dann gilt:*

$$a^{m-1} = 1 \tag{14}$$

für alle $a \in \mathbb{K}^x$
Beweis: siehe (Fields and Cyclotomic Polynomials)

Theorem 2.13. *Sei $\mathbb{K}$ ein Körper und $a \in K^x$. Weiterhin sei $n \in \mathbb{N}$, mit $n \geq 1$. Dann ist $a^n = 1 \Leftrightarrow ord_{\mathbb{K}}(a) \mid n$.*

Beweis. " $\Rightarrow$ "
Sei $ord_{\mathbb{K}}(a) = k$, wenn $k \mid n$, dann $n = mk$, für ein $m \geq 1$.

$$a^n = a^{mk} = (a^k)^m. \tag{15}$$

a^k ist definitionsmäßig(2.15) gleich 1. Daher gilt folgendes:

$$a^n = a^{mk} = (a^k)^m = (1)^m = 1. \tag{16}$$

" $\Leftarrow$ "
Sei nun $a^m = 1$, Außerdem seien $i, j \in \mathbb{N}$, sodass:
$im + jk = (m, k)$. Dabei ist k die Ordnung des Körpers(2.15).

Weiterhin gilt:

$$a^{(m,k)} = a^{im+jk} = (a^m)^i \cdot (a^k)^j = (1)^i(1)^k = 1. \tag{17}$$

Aus (17) und Der Definition(2.15) folgt, dass $(m, k) = k$ und somit $k \mid m$. □

2.4 Komplexitätstheorie

Hier handelt es sich darum, die relevanten Begriffe aus dem Gebiet der Komplexitätstheorie, deutlich zu definieren. Grundlagen der Komplexitätstheorie, wie Turingmaschinen, Polynomiallaufzeit, und Entscheidungsprobleme werden in dieser Arbeit nicht definiert, jedoch sind sie eine Voraussetzung, um die hier behandelten Themen zu verstehen.

Definition 2.16. In der Komplexitätstheorie ist **P** die Komplexitätsklasse, die alle Entscheidungsprobleme enthält, die in Polynomialzeit für deterministische Turingmaschinen lösbar sind.

Definition 2.17. Eine Funktion $f(n)$ ist $O^{\sim}(t(n))$, wenn

$$f(n) = O(t(n) \cdot poly(logt(n))) \tag{18}$$

Beobachtung:

$$O^{\sim}(log^k n) = O(log^k n) \cdot poly(loglog^k n) = O(log^k n) \cdot poly(log(k \cdot log n)) = O(log^{k+\epsilon} n) \quad (19)$$

Für alle $\epsilon > 0$, wenn der Algorithmus eine Laufzeit von $O^{\sim}(log^k n)$ hat. Dann läuft er in Polynomialzeit von $log\, n$.

Definition 2.18. Für eine natürliche Zahl $n \geq 1$, ist $\|n\| = \lceil log(n+1) \rceil$, wobei $log\, n = log_2 n$

Fakt 2.14. *Seien $n, m \in \mathbb{N}$:*

1. *Addition und Subtraktion von n und m können in $O(\|n\| + \|m\|) = O(log\, n + log\, m)$ Bit-Operationen berechnet werden.*

2. *Multiplikation $n \cdot m$ kann in $O(\|n\| \cdot \|m\|) = O(log\, n \cdot log\, m)$ Bit-Operationen berechnet werden.*

3. *Modulo-Berechnung von n mod m kann in $O(\|n\| - \|m\| + 1)$ Bit-Operationen berechnet werden.*

Fakt 2.15. *Seien $n, m \in \mathbb{N}$, $\|n\| = \|m\| = k$*

1. *Multiplikation kann in $O(k(log\, k)(log\, log\, k)) = O^{\sim}(k)$ Bit-Operationen berechnet werden.*

2. *n mod m kann in $O(k(log\, k)(log\, log\, k)) = O^{\sim}(k)$ Bit-Operationen berechnet*

3 Der AKS-Primzahltest

3.1 Grundidee des Algorithmus

Die Idee des Algorithmus ist basiert auf Verallgemeinerung des kleineren fermatschen Satz (2.4) für Polynome.

Lemma 3.1. *seien $a, n \in \mathbb{N}$ mit $a < n$ und teilerfremd, dann ist n genau dann eine Primzahl, wenn*

$$(X + a)^n = X^n + a (mod\ n). \quad (20)$$

Dabei ist X ein Polynom über dem Ring $\mathbb{Z}_n[X]$

Beweis. Aus dem binomischen Lehrsatz(2.2) folgt, dass der Koeffizient von X^i in dem Polynom $\binom{n}{i}a^{n-i}$ ist.

" $\Rightarrow$ "
Angenommen n ist eine Primzahl, dann ist es nach (2.3) klar, dass $\forall i$, $0 < \mathrm{i} < \mathrm{n}$, $\binom{n}{i} = \frac{n!}{(n-i)!i!} = 0$ (mod n). Das heißt alle Koeffizienten (bis auf den ersten und den letzten) sind Null.
Für $i = 0$ erhält man $\binom{n}{0}a^n X^0 = a^n$, analog für $i = n$, $\binom{n}{n}a^0 X^n = X^n$. Daraus folgt:
$(X + a)^n = a^n + 0 + 0 + ... + 0 + X^n = a^n + X^n (\mathrm{mod\ n})$.

" $\Leftarrow$ "
Sei n nun eine zusammengesetzte Zahl(COMPOSITE). Betrachte einen Faktor q von n, mit der Vielfachheit k(Dabei ist zu beachten, dass Für $1 < q < n$, $q^k | n$, aber $q^{k+1} \nmid n$). Der Koeffizient von X^q sieht wie folgt aus:

$$\binom{n}{q} \cdot a^{n-q} = \frac{n!}{(n-q)!n!} \cdot a^{n-q} = \frac{n(n-1)\cdots(n-q+1)}{q!} \cdot a^{n-q}. \tag{21}$$

Im Nenner lässt sich $q!$ als $q \cdot (q-1)!$ schreiben und im Zähler lässt sich n als $q^k \cdot m$ schreiben, $m \in \mathbb{Z}_+$. Das q im Nenner hebt sich mit einem der qs im Zähler auf. Der resultierende Term ist daher nicht durch q^k teilbar, außerdem sind q^k und a^{n-k} teilerfremd. Daraus folgt $(X + a)^n \neq X^n + a (\mathrm{mod}\ n)$. □

Es wäre nun möglich anhand dieser Identität einen Primzahltest für eine Zahl n zu realisieren. Dies wäre jedoch sehr ineffizient, da im Polynom die Auswertung von n Koeffizienten nötig ist. Mit anderen Worten der Algorithmus braucht $\Omega(n)$ um zu entscheiden, ob die Zahl n eine Primzahl ist oder nicht, und das ist nicht in Polynomialzeit realisierbar.

Die Idee von AKS ist nicht nur modulo n, sondern auch modulo ein Polynom $(X^r - 1)$ zu nehmen. Das bedeutet, dass wir nicht nur alle Koeffizienten c_k durch c_k modulo n ersetzen, sondern auch jedes X^k durch $X^k mod\, r$ ersetzen, das heißt wir arbeiten immer mit Polynomen vom Grad weniger als r. Dabei wird r so gewählt, dass die Anzahl an Berechnungen kleiner als bei (20) ist. Außerdem hat der Rest von $(X + a)^n (\mathrm{mod}\ n, X^r - 1)$ nur $r + 1$ Koeffizienten.
Dies kann für ein entsprechend kleines r in Polynomialzeit berechnet werden. Daher das Hauptziel jetzt ist ein entsprechend kleines r zu wählen und zu testen, ob die Gleichung:

$$(X + a)^n = X^n + a (\mathrm{mod}\ X^r - 1, n) \tag{22}$$

erfüllt ist.

Nach Lemma 3.1 ist die Gleichung (22) für alle Primzahlen erfüllt. Aber ein Problem bei diesem Ansatz wäre, dass es auch zusammengesetzte Zahlen gibt, für die die Gleichung für manche Werte von a und r erfüllt ist. Jedoch ist das geeignete r von oben durch ein Polynom in $logn$ beschränkt, das heißt der Algorithmus muss nur $logn$ a testen, um eine Entscheidung über die Primalität der Zahl n zu treffen.

3.2 Der Algorithmus

Notation:

- $o_r(n)$ bezeichnet die Ordnung der Zahl(Def. 2.5).
- (a,b) ist der größte gemeinsame Teiler (Def. 2.1).
- $\phi(n)$ ist die eulersche Phi-Funktion(Def. 2.7)

Algorithm 1 AKS-Primzahltest

Input: $n \in \mathbb{N}, n \geq 2$.

1. **if** $n = a^b, a \in \mathbb{N}, b \geq 1$, **return** COMPOSITE.
2. finde das kleinste r, sodass $o_r(n) > log^2 n$.
3. **if** $1 < (a, n) < n, a \geq n$, **return** COMPOSITE.
4. **if** $n \leq r$, **return** PRIME.
5. **for** a = 1 to $\lfloor\sqrt{\phi(r)}log(n)\rfloor$:
   if$(X + a)^n \neq X^n + a(\text{mod } X^r - 1, n)$, **return** COMPOSITE.
6. **return** PRIME.

Agrawal, Kayal und Saxena haben drei Kriterien gefunden, wenn sie für eine Zahl erfüllt sind, dann ist diese Zahl eine Primzahl.

Proposition 3.1. *Seien $n, r \geq 2$ natürliche Zahlen, mit $r \leq n$ und $o_r(n) > log^2 n$. n ist genau dann prim, wenn:*

1. *$n \neq a^b$, $f\ddot{u}r\, b, a \in \mathbb{N}$.*
2. *n keinen Primfaktor $p \leq r$ hat.*
3. *$(X + a)^n = X^n + a(mod\, n,\, X^r - 1)$ für alle a, $1 \leq a \leq \sqrt{r}logn$.*

Der Algorithmus überprüft alle drei Bedingungen für eine Zahl n.

Schritt 1: Im ersten Schritt sucht der Algorithmus zwei Zahlen $a, b \in \mathbb{N}$, $b > 1$, sodass $n = a^b$. Wenn solche Zahlen existieren, gibt er COMPOSITE zurück.

Schritt 2: Hier probiert der Algorithmus sukzessive Werte von r aus, bis $n^k \neq 1(mod\, r)$, $\forall k \leq log^2 n$.

Schritt 3: In diesem Schritt wird überprüft, ob n und a, $a \leq r$ gemeinsame Faktoren haben.

Schritt 4: Wenn keine Primfaktoren bei Schritt 3 gefunden worden sind, gibt der Algorithmus bei diesem Schritt PRIME zurück. Da $n \leq r$ und $(k, n) = 1$, $\forall k \leq r$(sonst hätte er beim 3. Schritt schon COMPOSITE zurückgegeben).

Schritt 5: Hier wird die dritte Bedingung aus 3.1 überprüft.

Schritt 6: Wenn kein a in Schritt 5 gefunden wurde, dann muss die Zahl prim sein.

3.3 Korrektheitsbeweis

Theorem 3.2. ***Hauptsatz der Korrektheit***
Der Algorithmus gibt genau dann PRIME zurück, wenn n eine Primzahl ist.

Dieses Ergebnis wird durch eine Reihe von Lemmata festgestellt. Der Korrektheitsbeweis lässt sich in zwei Teilen zerlegen, der erste Teil befasst sich mit der Hinrichtung des Beweises. Also, dass der Algorithmus PRIME liefert, wenn die Eingabe eine Primzahl ist. Dies ist aber trivial und braucht keine Lemmata. Für den zweiten Teil(Rückrichtung) sind jedoch mehrere Lemmata und Sätze nötig. Diese werden in diesem Kapitel in der Reihenfolge, wie sie im Originalartikel vorkamen, dargestellt und bewiesen. Zudem wird vor jedem Lemma kurz erläutert, warum dieses Lemma für den Algorithmus von Bedeutung ist. **Erster Teil des Beweises($\Rightarrow$)**

Theorem 3.3. *Wenn n eine Primzahl ist, dann gibt der Algorithmus PRIME zurück.*

Beweis. Wenn n eine Primzahl ist, gibt der erste Schritt in (3.2) niemals COMPOSITE zurück, da keine Primzahl sich als a^b schreiben lässt, sonst wäre n durch a teilbar. Der dritte Schritt kann auch niemals COMPOSITE zurückgeben, da $(a, n) = 1, \forall a$. Nach Lemma (3.1) kann Schritt 5 auch nie COMPOSITE zurückgeben. Daher muss der Algorithmus entweder bei viertem oder bei sechstem Schritt PRIME zurückgeben. □

Zweiter Teil des Beweises($\Leftarrow$)

Hier wird die Rückrichtung behandelt; also wenn die Ausgabe PRIME ist, dann ist n eine Primzahl. Wir werden uns nur auf die Schritte konzentrieren, die PRIME zurückgeben. Dies sind die schritte 4 und 6.

Wenn der Algorithmus bei Schritt 4 PRIME zurückgibt, dann muss n eine Primzahl sein, da sonst der Algorithmus einen nicht trivialen Faktor von n in Schritt 3 gefunden hätte. Das heißt, es bleibt nur der Fall bei Schritt 6. Der Algorithmus hat Zwei Hauptschritte, das sind Schritt 2 und Schritt 5. Bei Schritt 2 wird ein geeignetes r gefunden und bei Schritt 5 wird verifiziert, ob die Gleichung (22) für mehrere a Werte gilt.

Für diesen Teil ist eine Vorarbeit nötig. Wir brauchen zunächst eine untere Schranke für das kleinste gemeinsame Vielfaches, diese Abschätzung ist bei dem Existienzbeweis für das geeignete r von Bedeutung.

Lemma 3.4. *Für $m \geq 7$:*

$$kgV(m) \geq 2^m. \tag{23}$$

Beweis. Sei $d_n = kgV1 \leq m \leq n\{m\}$, zuerst wird gezeigt, dass $m\binom{n}{m} \mid d_n$.
Wir betrachten das folgende Integral für $n \geq 1$:

$$\begin{aligned} I_{n,m} &= \int_0^1 x^{m-1}(1-x)^{n-m}dx \underbrace{=}_{(2.2)} \int_0^1 x^{m-1} \sum_{r=0}^{n-m} (-1)^r \binom{n-m}{r} \cdot x^r dx \\ &= \sum_{r=0}^{n-m} (-1)^r \binom{n-m}{r} \cdot \int_0^1 x^{m+r-1} = \sum_{r=0}^{n-m} (-1)^r \binom{n-m}{r} \cdot \frac{x^{m+r}}{m+r}\Big|_0^1 \\ &= \sum_{r=0}^{n-m} (-1)^r \binom{n-m}{r} \cdot \frac{1}{m+r} \end{aligned} \tag{24}$$

Aus (24) ist es leicht zu sehen, dass $r \leq n-m$ und folglich $r+m \leq n$. Das heißt $m+r \mid d_n$. Dabei ist es offensichtlich, dass $d_n \cdot I_{n,m} \in \mathbb{N}$.

Wir wollen nun durch Induktion nach $n-m$ und partielle Integration zeigen, dass für $n, m \in \mathbb{N}$, $1 \leq m \leq n$

$$I_{n,m} = \int_0^1 x^{m-1}(1-x)^{n-m}dx = \frac{1}{m \cdot \binom{n}{m}} \tag{25}$$

immer gilt.

IA: $n-m=0 \iff n=m$:

$$I_{n,m} = \int_0^1 x^{m-1} \cdot (1-x)^{n-m=0}dx = \int_0^1 x^{m-1}dx = \left[\frac{x^m}{m}\right]_0^1 = \frac{1}{m} = \frac{1}{m \cdot \binom{m}{m}}$$

IH: Nun wird Angenommen, dass (25) für $1 \leq m < n$ wahr ist.

IS:

Partielle Integration:

$$I_{n,m} = \int_0^1 \underbrace{x^{m-1}}_{f'} \cdot \underbrace{(1-x)^{n-m}}_{g} dx = \Big[\underbrace{\frac{x^m}{m}}_{f} \cdot \underbrace{(1-x)^{n-m+1}}_{g} \Big]_0^1 - \int_0^1 \underbrace{\frac{x^m}{m}}_{f} \cdot \underbrace{(-1) \cdot (n-m) \cdot (1-x)^{m-n-1}}_{g'} dx$$

$$= 0 + \int_0^1 \frac{(n-m) \cdot x^m}{m} \cdot (1-x)^{n-m-1} dx$$

$$= \frac{(n-m)}{m} \cdot \int_0^1 x^{(m+1)-1} \cdot (1-x)^{n-(m+1)} dx$$

$$\underbrace{=}_{IH} \frac{(n-m)}{m} \cdot \frac{1}{(m+1) \cdot \binom{n}{m+1}} = \frac{(n-m)}{m} \cdot \frac{(n-(m+1))! \cdot (m+1)!}{(m+1) \cdot n!}$$

$$= \frac{(n-m)! \cdot m!}{m \cdot n!} = \frac{1}{m \cdot \binom{n}{m}}$$

Somit ist die Aussage(25) wahr.

Demzufolge gilt:

$$\frac{d_n}{m \cdot \binom{n}{m}} \in \mathbb{N} \Rightarrow m \cdot \binom{n}{m} \mid d_n \tag{26}$$

$\forall m$ mit $1 \leq m \leq n$.

Somit gilt auch:

$$n\binom{2n}{n} \mid d_{2n} \tag{27}$$

Beziehungsweise

$$(2n+1)\binom{2n}{n} = (n+1)\binom{2n+1}{n+1} \mid d_{2n+1}. \tag{28}$$

$$\Rightarrow n\binom{2n}{n} \mid d_{2n} \mid d_{2n+1}$$

Da n und $2n+1$ teilerfremd sind gilt auch:

$$n(2n+1)\binom{2n}{n} \mid d_{2n+1}$$

Daraus folgt:

$$n(2n+1)\binom{2n}{n} \leq d_{2n+1} \leq d_{2n+2}$$

Nun zeigen wir nun durch vollständige Induktion, dass

$$n \cdot (2n+1)\binom{2n}{n} \geq 2^{2n+2}, \forall n \geq 3 \tag{29}$$

gilt.

IA: $n = 3$

$$3 \cdot (2 \cdot 3 + 1)\binom{6}{3} = 420 > 256 = 2^{2 \cdot 3+2}.$$

IH: Für ein beliebiges $n \geq 3$ sei (29) wahr.

IS:

$$(n+1) \cdot (2n+3) \cdot \binom{2n+2}{n+1} = (n+1) \cdot (2n+3) \cdot \binom{2n}{n} \cdot \frac{2 \cdot (2n+1)}{(n+1)}$$

$$\geq 2 \cdot (n+1) \cdot (2n+3)\binom{2n}{n} \cdot \frac{2n}{(n+1)} = 2 \cdot 2n \cdot (2n+3) \cdot \binom{2n}{n}$$

$$= 4 \cdot n \cdot (2n+3) \cdot \binom{2n}{n} \underbrace{\geq}_{IH} 4 \cdot 2^{2n+2} = 2^2 \cdot 2^{2n+2}$$

$$= 2^{2n+4} = 2^{2 \cdot (n+1)+2}$$

Somit gilt die Aussage für alle $n \geq 3$.

Daraus folgt $d_{2n+2} \geq d_{2n+1} \geq n \cdot (2n+1)\binom{2n}{n} \geq 2^{2n+2}$, $\forall n \geq 3$.

Folglich gilt auch $\forall N \in \mathbb{N}$, $N \geq 7$:

$$d_N \geq 2^N$$

□

Abschätzung von r.

Das zweite Lemma befasst sich mit der Abschätzung von r. Zuerst wird gezeigt, dass r in Schritt 2 des Algorithmus polynomial von $log\, n$ abhängig ist und das ermöglicht die Polynomiallaufzeit des Algorithmus.

Lemma 3.5. *Es existiert ein* $r \leq max\{3, \lceil log^5 n \rceil\}$*, sodass* $o_r(n) > log^2 n$.

Beweis. Der Beweis lässt sich in 3 Schritten zerlegen; beim ersten Schritt handelt es sich darum, zu zeigen, dass ein $r \leq B$ existiert, danach wird die Existenz von $o_r(n)$ gezeigt. Als letztes wird die Eigenschaft $o_r(n) \geq log^2 n$ für dieses r bewiesen. Sei $n \geq 1$.

Für $n = 2$ und $r = 3$ gilt trivialerweise :

$ord_3(2) = 2^2 = 4 = 1(\text{mod } 3) > 1 = log^2 2$.

Ab jetzt wird angenommen, dass $n > 2$.

Bemerkung: $\lceil log^5 3 \rceil = 11 \Rightarrow log^5 n > 10, \forall n > 2$.

Sei $B = \lceil log^5 n \rceil$, nach Lemma 3.4 gilt $kgV(B) \geq 2^B$. Zunächst wird gezeigt, dass ein $r \leq B$ existiert, sodass

$$N_B = n^{\lfloor logB \rfloor} \cdot \prod_{i=1}^{\lfloor log^2 n \rfloor} (n^i - 1) \tag{30}$$

r dieses Produkt nicht teilt. Dies kann durch Widerspruch bewiesen werden.

Angenommen, $\forall 1 \leq r \leq n$, r teilt N_B. Das Produkt ist ein gemeinsames Vielfaches aller Zahlen kleiner gleich B, daher gilt $N_B \geq kgV(B)$.

Wir betrachten N_B:

$$\begin{aligned} N_B &= n^{\lfloor logB \rfloor} \cdot \prod_{i=1}^{\lfloor log^2 n \rfloor} (n^i - 1) < n^{\lfloor logB \rfloor} \cdot \prod_{i=1}^{\lfloor log^2 n \rfloor} n^i \\ &= n^{\lfloor logB \rfloor + \sum_{i=1}^{\lfloor log^2 n \rfloor} i} \\ &\underbrace{=}_{(2.5)} n^{\lfloor logB \rfloor + \frac{log^2 n \cdot (log^2 n + 1)}{2}} \\ &= n^{\lfloor logB \rfloor + \frac{log^4 n + log^2 n}{2}} \\ &\leq n^{\lfloor logB \rfloor + \frac{log^4 n + \frac{log^4 n}{2}}{2}} \\ &\leq n^{\lfloor \frac{log^4 n}{4} \rfloor + \frac{log^4 n + \frac{log^4 n}{2}}{2}} \\ &\leq n^{log^4 n} = 2^{log n^{log^4 n}} = 2^{log^5 n} = 2^B. \end{aligned} \tag{31}$$

Aus (31) folgt: $kgV(B) \leq N_B \leq 2^B$. Das ist aber ein Widerspruch zu Lemma 3.4. Das heißt es existiert eine Menge von Zahlen $R = \{r_1, r_2, ..., r_t\}$, $1 \leq r_i \leq B, i = 1, 2, .., t$, sodass r_i das Produkt nicht teilt. Ferner sei r_m das kleinste Element dieser Menge.

Jetzt bleibt zu zeigen, dass $o_{r_m}(n)$ für dieses r_m existiert, und $o_{r_m}(n) \geq log^2 n$.

Sei $r_m = ab$, wobei a aus den Primfaktoren besteht, die n teilen und b aus den restlichen Primfaktoren. Offensichtlich $(b, n) = 1$, außerdem ist die höchste Potenz, die ein Primfaktor(in der Primfaktorzerlegung) haben kann, kleiner als $\lfloor logB \rfloor$. Da sonst a größer als B wäre. Das heißt, jeder Primfaktor in a hat einen Exponent kleiner als der Exponent beim gleichen Primfaktor von n und jeder vorkommende Faktor in a, kommt auch in n vor(alle Primfaktoren von a teilen n).

$\Rightarrow a \mid n^{logB}$.

Das heißt b teilt $\prod_{i=1}^{\lfloor log^2 n \rfloor}(n^i - 1)$ nicht, da sonst r_m das Produkt teilen würde. b teilt n^{logB} auch nicht, da n und b keine gemeinsamen Primfaktoren haben$((b, n) = 1)$. Zudem war r_m das kleinste Element aus R, das N_B nicht teilt, daher ist $r_m = b$ und $(b, n) = 1$ beziehungsweise $(r_m, n) = 1$. Somit existiert $o_{r_m}(n)$.

Zuletzt bleibt nur der Beweis der Ungleichung $o_{r_m}(n) > log^2 n$. Das lässt sich auch wie oben durch einen Widerspruch zeigen. Sei $o_{r_m}(n) = k \leq log^2 n$, per Definition ist k die kleinste Zahl, für die $n^k = 1 \pmod{r_m}$ wahr ist. daher gilt auch:

$$n^k - 1 = 0 (\text{mod } r_m) \Rightarrow r_m \mid n^k - 1.$$

Aber wenn das gelten würde, würde r_m das Produkt N_B teilen und das führt zum Widerspruch, da $r_m \nmid N_B$. □

Lemma 3.6. *wenn $o_r(n) > 1$, dann existiert ein Primfaktor p von n, sodass $o_r(p) > 1$ gilt.*

Beweis. Sei $o_r(n) > 1$, mit der Primfaktorzerlegung $n = \prod_{i=1}^{M} p_i^{e_i}$.

Weiterhin sei $o_r(p_i) = 1, \forall 1 \leq i \leq M$ (es existiert kein $p_i : o_r(p_i) > 1$).

$\Rightarrow p_i = 1 (\text{mod } r)$, $\forall i$. Folglich ist $n = \prod_{i=1}^{M} p_i^{e_i} = \prod_{i=1}^{M} (1)^{e_i} = 1 (\text{mod } r)$.

Daraus wird ersichtlich, dass $o_r(n) = 1$ und das ist ein Widerspruch zur Voraussetzung, dass $o_r(n) > 1$. Daher muss mindestens ein Primfaktor $p : o_r(p) > 1$. □

Nach Lemma (3.6) muss für $o_r(n) > 1$ ein Primteiler p von n(Aus der Primfaktorzerlegung) existieren, sodass $o_r(p) > 1$. Außerdem ist es klar, dass $p > r$, da sonst der Algorithmus eine Entscheidung über die Primalität schon im 3. beziehungsweise im 4. Schritt getroffen hätte(also, für den Fall $a \leq r$ oder $n \leq r$). Seien $p, n \in \mathbb{Z}_r^*$ fest. Ferner sei $l = \lfloor \sqrt{\phi(r)} log n \rfloor$.

Der 5. Schritt des Algorithmus überprüft l Gleichungen. Wir wollen nun die Ausgabe PRIME untersuchen, also nach Voraussetzung ist die Ausgabe PRIME . Das heißt, für alle Gleichungen l gilt:

$$(X + a)^n = X^n + a \ (\text{mod } X^r - 1, n) \quad .$$

$\forall a$, $0 \leq a \leq l$.

Bemerkung: Für a $= 0 \Rightarrow (X + 0)^n = X^n + 0$ und damit ist die Gleichung trivialerweise wahr.

Aus der oberen Gleichung kann man drei andere Gleichungen folgern:

1. $(X + a)^n = X^n + a (\text{mod } X^r - 1, p)$. $\forall\ 0 \leq a \leq l$.

   Da p ein Primfaktor von n, gilt auch nach Lemma 3.1

2. $(X + a)^p = X^p + a (\text{mod } X^r - 1, p)$. $\forall\ 0 \leq a \leq l$.

   Aus 1 und 2 gilt auch

3. $(X+a)^{\frac{n}{p}} = X^{\frac{n}{p}} + a (\text{mod } X^r - 1, p)$. $\forall\ 0 \leq a \leq l$.

Das heißt,$\forall\ 0 \leq a \leq l$, n, $\frac{n}{p}$ und p haben ein identisches Verhalten bezüglich der obigen Gleichung. Diese Eigenschaft heißt Introspektivität.

Introspektive Zahlen und Polynome: Introspektive Zahlen und Polynome spielen eine große Rolle für den Korrektheitsbeweis des AKS-Primzahltests. Zunächst ist eine formale Definition von Introspektiven Zahlen beziehungsweise Polynomen nötig.

Definition 3.1. Seien $f(X)$ ein Polynom in $\mathbb{Z}[X]$ und $m \in \mathbb{N}$, m ist **introspektiv** bezüglich $f(X)$, wenn

$$[f(X)]^m = f(X^m) \text{ (mod } X^r - 1, p)$$

gilt.

Für den Korrektheitsbeweis ist eine Eigenschaft introspektiver Zahlen relevant. Dies ist die Multiplikativität von solchen Zahlen.

Lemma 3.7. *Seien $m, m^{'}$ introspektive Zahlen bezüglich eines Polynoms $f(X)$ in $\mathbb{Z}[X]$. Dann ist $m \cdot m^{'}$ auch introspektiv bezüglich $f(X)$.*

Beweis. Da m bezüglich $f(X)$ introspektiv ist, gilt:

$$[f(X)]^{m \cdot m^{'}} = [f(X)^m]^{m^{'}} (\text{ mod } X^r - 1, p).$$

$m^{'}$ ist auch introspektiv bezüglich $f(X^m)$, das heißt es gilt auch:

$$[f(X^m)]^{m^{'}} = f(X^{m \cdot m^{'}}) (\text{ mod } X^{m \cdot r} - 1, p).$$

Es gilt auch $X^{m \cdot r} - 1 = (X^r)^m - (1)^m \underbrace{=}_{(2.8)} (X^r - 1) \sum_{k=0}^{m-1} X^{r \cdot k}$.

$\Rightarrow X^r - 1 \mid X^{mr} - 1$. Somit gilt

$$[f(X^m)]^{m^{'}} = f(X^{m \cdot m^{'}}) (\text{ mod } X^r - 1, p).$$

Daraus folgt:

$$[f(X)]^{m \cdot m'} = f(X^{m \cdot m'})(\text{ mod } X^r - 1, p).$$

Das heißt wenn eine Zahl m bezüglich eines Polynoms $f(X)$ introspektiv ist, ist auch $m \cdot m'$ bezüglich des selben Polynoms $f(X)$ introspektiv. □

Lemma 3.8. *wenn m bezüglich der Polynome $f(X)$ und $g(X)$ introspektiv ist. Dann ist m auch bezüglich $f(X) \cdot g(X)$ introspektiv.*

Beweis.

$$[f(X) \cdot g(X)]^m = [f(X)]^m \cdot [g(X)]^m \underbrace{=}_{(3.7)} f(X^m) \cdot g(X^m)(mod\, X^r - 1, p).$$

□

Wir definieren nun die folgenden Mengen:

$$I := \{\frac{n^i}{p^i} \cdot p^j \mid i, j \geq 0\}. \tag{32}$$

$$P := \{\prod_{a=0}^{l} (X + a)^{e_a} \mid e_a \geq 0\}. \tag{33}$$

Aus Lemma 3.7 und Lemma 3.8 folgt, dass jedes Element $i \in I$ bezüglich jedes Polynoms $p \in P$ introspektiv ist.

Wir definieren jetzt zwei Gruppen basierend auf diese zwei Mengen, die für den Beweis von Bedeutung sind.

Die erste Gruppe G sei die Gruppe aller Restklassen in I modulo r, das heißt $G = \{(\frac{n}{p})^i \cdot p^j (mod\, r) \mid i, j \geq 0\}$. Da $(n, r) = (p, r) = 1$ ist G eine multiplikative Untergruppe von $\mathbb{Z}_r^*$. Sei die Kardinalität im weiteren $|G| = t$. G wird von n und p modulo r erzeugt, und da $o_r(n) > log^2 n$, gilt auch $t > log^2 n$.

Um die zweite Gruppe zu definieren, ist die folgende Eigenschaft von zyklotomischen Polynomen über endlichen Körpern erforderlich.

Lemma 3.9. ***Irreduzible Teiler zyklotomischer Polynome*** *Sei $\mathbb{K}_p$ ein Körper und $\Phi_r(X)$ das r-te zyklotomische Polynom über $\mathbb{K}_p$. Dann existiert ein irreduzibler Teiler $h(X)$ von $X^r - 1$ über $\mathbb{K}_p$ vom Grad $o_r(p)$.*

Beweis. Sei p eine Primzahl und $\mathbb{K}_p$ der dazugehörige Restklassenkörper. Zudem seien r und p teilerfremd. Nach Def. 2.14 gilt:

$$\Phi_r(X) = \prod_{\substack{1 \le k \le n \\ (k,n)=1}} \left(X - \zeta^k\right). \tag{34}$$

Nach Lemma 2.10 gilt $\Phi_r(X) \mid X^r - 1$. Sei ζ eine beliebige Einheitswurzel von $\Phi_r(X)$ über $\mathbb{K}_p$. $h(X) = \prod_{k=1}^{q}(X - \zeta^{p^k})$ ist genau dann das minimale Polynom von ζ über $\mathbb{K}_p$, wenn q die kleinste positive natürliche Zahl, sodass $\zeta^{p^q} = 1$ beziehungsweise $\zeta^{p^q \bmod r} = 1$(Da ζ eine primitive Einheitswurzel von r ist), das ist äquivalent zu $q = o_r(p)$, es ist außerdem offensichtlich, dass diese Relation nur von r, p abhängig ist, also unabhängig von der Wahl von ζ. Das Minimalpolynom $h(X)$ teilt jedes Polynom mit den Nullstellen ζ. Daher ist $h(X)$ irreduzibl über $\mathbb{K}_p$ und teilt $\Phi_r(X) \Rightarrow h(X) \mid X^r - 1$, folglich zerfällt $\Phi_r(X)$ in Polynome über $\mathbb{K}_p$ jeweils vom Grad $o_r(p)$. Das bedeutet $\Phi_r(X)$ zerfällt in $\frac{deg(\Phi_r(X))}{o_r(p)} = \frac{\phi(p)}{o_r(p)}$ irreduzible Polynome vom Grad $o_r(p)$ über $\mathbb{K}_p$. Das heißt es existiert ein über $\mathbb{K}_p$ irreduzibler Teiler von $X^r - 1$ vom Grad $o_r(p)$. □

Zur Definition der zweiten Gruppe sei nun $h(X)$ einer dieser Teiler aus Lemma 3.9. Die zweite Gruppe $\mathcal{G}$ sei die multiplikative Gruppe aller Restklassen von Polynomen in P modulo $h(X)$ und p, also $\mathcal{G} = \{\prod_{a=0}^{l}(X + a)^{e_a}(mod\, h(X), p) \mid e_a \ge 0\}$. $\mathcal{G}$ ist eine Untergruppe von $\mathbb{K} = \mathbb{K}_p[X]/(h(X))$ und wird von $X, X+1, X+2, ..., X+l$ im Körper $\mathbb{K}$ erzeugt.

Wie benötigen nun eine untere Schranke für $\mathcal{G}$.

Lemma 3.10. $|\mathcal{G}| > \binom{t+l}{t-1}$

Beweis. Zunächst sei $h(X)$ ein Faktor des r-ten zyklotomischen Polynoms $\Phi_r(X)$ und somit auch von $X^r - 1$. Das heißt X ist eine primitive r-te Einheitswurzel in $\mathbb{K}$. Wir zeigen zuerst, dass alle $f(X), g(X)$, $f(X) \neq g(X)$ aus P vom Grad $< t$ auf zwei verschiedene Elemente in $\mathcal{G}$ abgebildet werden.

Angenommen $f(X) = g(X)$ im Körper $\mathbb{K}$. Weiterhin sei $m \in I$ beliebig und $[f(X)]^m = [g(X)]^m$ in $\mathbb{K}$. m ist introspektiv bezüglich f, g daher gilt:

$$f(X^m) = g(X^m)(mod\, X^r - 1, p) \quad .$$

Da $h(X) \mid X^r - 1$ gilt auch

$$f(X^m) = g(X^m)(mod\, h(X), p).$$

folglich gilt $f(X^m) = g(X^m)$ in $\mathbb{K}$. Daraus folgt, dass X^m eine Wurzel des Polynoms $Q(Y) = f(Y) - g(Y)$, $\forall m \in I$ ist. Da X eine primitive r-te Einheitswurzel von $\Phi_r(X)$ in $\mathbb{K}$ und $(m, r) = 1$, $\forall m \in G$, folgt auch, dass X^m eine r-te primitive Einheitswurzel in $\mathbb{K}$ ist.

Es existieren also $|G| = t$ verschiedene Wurzeln von Q in $\mathbb{K}$. Nach der Wahl von f, g ist $deg\, Q < t$. Das ist ein Widerspruch zu Theorem 2.6 und folglich gilt $f(X) \neq g(X)$ in $\mathbb{K}$.

Es gilt auch, dass $i \neq j \in \mathbb{K}_p, \forall\, 1 \leq i \neq j \leq l = \lfloor\sqrt{\phi(r)}logn\rfloor$. Da $o_r(n) > log^2 n$, gilt $r > log^2 n$ und folglich gilt auch:

$$l = \lfloor\sqrt{\phi(r)}logn\rfloor < \sqrt{r}logn < r < p$$

Daher sind $X, X+1, ..., X+l$ verschieden in $\mathbb{K}$. Außerdem gilt $deg\, h > 1$ und $X + a \neq 0\, in \mathbb{K}$, $\forall a$, $0 \leq a \leq l$. Das bedeutet es existieren $l + 1$ verschiedene Polynome vom Grad 1 (Da $deg\, X + a = 1$). Gesucht ist gerade die Anzahl von Polynomen mit Grad $< t$ in $\mathcal{G}$. Das kann man durch Multiplikation bis $t - 1$ nicht unbedingt eindeutige Elemente in $\{X, X+1, ..., X+l\} \subseteq P$ erreichen, also es existieren insgesamt $\binom{t-1+l+1}{t-1} = \binom{t+l}{t-1}$ eindeutige Polynome $f \in P$ vom Grad $< t$ □

Wenn n keine Potenz von p ist, erhält man auch eine obere Grenze für $|\mathcal{G}|$.

Lemma 3.11. ***Obere Schranke für*** $|\mathcal{G}|$.
Wenn n keine Potenz von p, dann gilt:

$$|\mathcal{G}| \leq n^{\sqrt{t}} \tag{35}$$

Beweis. Angenommen n ist keine Potenz von p. Dann ist $\frac{n}{p}$ durch $q \neq p$ teilbar. Wir betrachten folgende Teilmenge von I:

$$\widetilde{I} = \{(\frac{n}{p})^i p^j \mid 0 \leq i, j \leq \sqrt{t}\} \subseteq I. \tag{36}$$

Es gilt $|\widetilde{I}| = (\lfloor\sqrt{t}\rfloor + 1)^2 > t = |G|$, da es $\lfloor\sqrt{t}\rfloor + 1$ für jeweils i und j gibt. Deshalb müssen mindestens zwei Elemente $m_1, m_2 \in \widetilde{I}$ existieren, sodass $m_1 = m_2\, mod\, r$. O.B.d.A gilt $m_1 > m_2$ Folglich gilt auch:

$$X^{m_1} = X^{m_2}\, (mod\, X^r - 1) \quad .$$

für alle Polynome $f \in P$. Das heißt es gilt:

$$\begin{aligned} f(X)^{m_1} &= f(X^{m_1})\, (mod\, X^r - 1, p) \\ &= f(X^{m_2})\, (mod\, X^r - 1, p) \\ &= f(X)^{m_2}\, (mod\, X^r - 1, p) \end{aligned} \tag{37}$$

Aus (37) folgt $[f(X)]^{m_1} = [f(X)]^{m_2}$ in $\mathbb{K}$. Demnach ist $f(X) \in \mathcal{G}$ eine Wurzel des Polynoms $Q^{'}(Y) = Y^{m_1} - Y^{m_2}$ im Körper $\mathbb{K}$. Da $f(X)$ beliebig ist, gibt es $|\mathcal{G}|$ verschiedene Wurzeln von $Q^{'}(Y)$ im Körper $\mathbb{K}$. Außerdem gilt auch:

$$deg\, Q^{'}(Y) = m_1 \leq (\frac{n}{p} \cdot p)^{\lfloor\sqrt{t}\rfloor} \leq n^{\sqrt{t}}. \tag{38}$$

Nach (2.6) gilt $|\mathcal{G}| \leq deg\, Q^{'}(Y)$, und folglich gilt $|\mathcal{G}| \leq n^{\sqrt{t}}$.

□

Nun haben wir alle nötigen Lemmata für die Rückrichtung des Beweises

Theorem 3.12. *Ist die Ausgabe PRIME, dann ist n eine Primzahl*

Beweis. Nach Voraussetzung ist die Ausgabe PRIME. Nach Lemma 3.10 gilt für $t = |G|$, und $l = \lfloor\sqrt{\phi(r)}logn\rfloor$ das folgende:

$$\begin{aligned}
|\mathcal{G}| &\geq \binom{t+l}{t-1} \\
&\geq \binom{l+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}log\, n\rfloor} && (\, t = \sqrt{t}\sqrt{t} \geq \sqrt{t}\log n) \\
&\geq \binom{2\lfloor\sqrt{t}\log n\rfloor + 1}{\lfloor\sqrt{t}log\, n\rfloor} && (\, l = \lfloor\sqrt{\phi(r)}log\, n\rfloor \geq \lfloor\sqrt{t}\log n\rfloor) \\
&\geq 2^{\lfloor\sqrt{t}\log n\rfloor+1} && (\, Theorem\, 2.7\,) \\
&\geq n^{\sqrt{t}}.
\end{aligned}$$

Gemäß Lemma 3.11 gilt $|\mathcal{G}| \leq n^{\sqrt{t}}$, falls n keine Potenz von p ist, daraus kann man folgern, dass $n = p^k$, für ein $k \geq 1$. Aber die Ausgabe ist nach Voraussetzung PRIME, folglich gilt $k = 1 \Rightarrow n = p$. Somit ist n eine Primzahl. □

Aus Theorem 3.3 und Theorem 3.12 folgt der Hauptsatz der Korrektheit.

4 Laufzeitanalyse

Zur Berechnung der Zeitkomplexität werden die Fakten(2.14) und (2.15) aus dem Zweiten Kapitel verwendet. Das heißt, Multiplikation, und Modulorechnung von zwei Zahlen jeweils mit $\|m\|$ Bits können in $O^{\sim}(\|m\|)$ Schritten durchgeführt werden. Nach dem gleichen

Prinzip können diese Operationen auf zwei Polynome übertragen werden, für Polynome vom Grad d und eine maximale Bitlänge von $\|m\|$ können Multiplikationen und Modulorechnungen in $O^{\sim}(d \cdot m)$ Schritten durchgeführt werden.

Theorem 4.1. *Die asymptotische Laufzeit des Algorithmus ist* $O^{\sim}(log^{21/2}n)$.

5 Zusammenfassung

A Anhang A

Algorithm 2 Potenz-Prüfung

Input: $n \in \mathbb{N}, n \geq 2$.

1. a, b, c, m
2. $b = 2$.
3. **while** $2^b \leq n$ **repeat**
4. $\quad a = 1$, $c = n$
5. $\quad$ **while** $c - a \geq 2$ **repeat**
6. $\quad\quad m = \lfloor (a + c)/2 \rfloor$
7. $\quad\quad p = min\{m^b, n + 1\}$
8. $\quad\quad$ **if** $p = n$ **return** True.
9. $\quad\quad$ **if** $p < n$
10. $\quad\quad\quad a = m$
11. $\quad\quad$ **else**
12. $\quad\quad\quad c = m$
13. $\quad b = b + 1$
14. **return** False

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.
Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Leipzig, den 25. März 2020 Salman Salman