# Komplexität und Evaluation des AKS-Primzahltests

**Salman Salman**

**3753924**


UNIVERSITÄT LEIPZIG

Bachelorarbeit

Lehrstuhl für Informatik
und Mathematik
Universität Leipzig

Betreuer:

Leipzig, den 11. 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 geprüft und bewiesen.

# 1 Einleitung

Die Frage, ob eine Zahl eine Primzahl oder nicht, war schon im antiken Griechenland interessant, Euklid hat sich mit dieser Frage beschäftigt und hat beweisen, dass es unendlich viele Primzahlen gibt. Ein anderer griechischer Mathematiker Eratosthenes hat einen Algorithmus zur Bestimmung von Primzahlen vorgestellt, aber sein Algorithmus war trotz seiner Einfachheit ineffizient. Im Laufe der Jahre wurden auch mehrere Algorithmen zur Lösung des Primalitätsproblems entwickelt, die entweder von unbewiesenen Hypothesen abhängig waren oder probabilistisch waren. Das heißt randomisierte Algorithmen, die auch ein falsches Ergebnis liefern können, aber der beste nicht probabilistische Algorithmus (vor AKS) konnte das Primalitätsproblem in $\Omega(\sqrt{n})$ Schritten lösen, wobei n die Eingabegröße ist, solcher Algorithmus braucht exponentielle Zeit, um das Problem zu lösen. In 2002 haben drei Informatiker Agrawal, Kayal, und Saxena den ersten deterministischen unbedingten Algorithmus, der das Primalitätsproblem in Polynomialzeit lösen kann.

In dieser Arbeit wird der Algorithmus aus dem Orginalpaper studiert. Im ersten Abschnitt werden die nötigen Theoreme und Definitionen vorgestellt, welche im Folgenden verwendet werden. Im zweiten Abschnitt wird zuerst die Idee hinter dem Algorithmus erläutert. Danach wird der Algorithmus in seiner ursprünglichen Form angegeben, anschließend wird der Korrektheitsbeweis des Algorithmus geführt. Im dritten Abschnitt wird die Laufzeit des Algorithmus formal analysiert, dabei werden die einzelnen Schritte (Steps 1-5 im Orginalpaper) ausführlich analysiert. Darüber hinaus werden die Resultate der Laufzeitexperimente angegeben.

## 1.1 Definitionen und Vorbereitungen

In diesem Abschnitt werden die essenziellen Begriffe und Theoreme aus der Zahlentheorie und aus der Algebra, die für den AKS-Algorithmus relevant sind, definiert. Das Ziel hinter diesem Abschnitt, ist einen formalen Literaturhinweis zu haben, um später die hier definierten Begriffe und Theoreme zu referenzieren.

### 1.1.1 Zahlentheorie

**Definition 1.1.** Seien $a, b \in \mathbb{N}$. Der **größte gemeinsame Teiler** von a und b wird mit gcd(a,b) bezeichnet, ist die größte positive Zahl n, sodass $n \mid a$ und n $\mid b$

**Definition 1.2.** Zwei Zahlen $a, b \in \mathbb{N}$ heißen genau dann **Teilerfremd**, wenn $gcd(a, b) = 1$.

**Definition 1.3.** 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 $\equiv$ b (mod n) bezeichnet.

**Definition 1.4.** Seien $r, n \in \mathbb{N}$ mit gcd(n,r) = 1, dann ist die **Ordnung** von n modulo r das kleinste k, sodass $n^k \equiv 1 (mod$ r). Die Ordnung wird mit $ord_r(n)$ bezeichnet.

**Definition 1.5.** Sei $n \in \mathbb{N}$, die Primzahlzerlegung von n ist die Darstellung der Zahl als Produkt ihrer Primfaktoren
$n = p_1^{e_1} p_2^{e_2} ... p_M^{e_M} = \prod_{k=1}^{M} p_k$. Wobei $e_k$ die Vielfachheit der Primzahl $p_k$ ist.

**Definition 1.6.** Sei $n \in \mathbb{N}$ mit n > 1. Die Eulersche Phi-Funktion wird mit $\phi(n)$ bezeichnet, ist die Anzahl an Zahlen zwischen n und 0, die Teilerfremd sind. Außerdem ist $|\mathbb{U}_n| = \phi(n)$. Sei nun $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$ die Primzahlfaktorisierung von n, dann ist die Eulersche Phi-Funktion folgendermaßen definiert:

$$\phi(n) = n \prod_{i=1}^{k} (1 - \frac{1}{p_i})$$

**Theorem 1.1** (**Satz von Euler**). *Seien $a, n \in \mathbb{N}$ und Teilerfremd, dann gilt $a^{\phi(n)} \equiv 1$(mod n).*

**Theorem 1.2** (**Kleiner fermatscher Satz**). *Sei $a \in \mathbb{N}$ und p eine Primzahl dann gilt:*

1. *a und p sind genau dann Teilerfremd, wenn $a^{p-1} \equiv 1$ (mod p) gilt.*
2. *$\forall a$ gilt $a^p \equiv a$ (mod p)*

### 1.1.2 Algebra

Es wird davon ausgegangen, dass die grundlegenden Definitionen von Gruppen, Ringen, Körpern und ihre Eigenschaften dem Leser bekannt sind. Resultate die später vorkommen, werden hier zitiert. Außerdem werden Theoreme und Begriffe der Polynomentheorie, die später für den Korrektheitsbeweis von Bedeutung sind, vorgestellt und bewiesen.

**Definition 1.7.** Sei R ein Ring, dann ist ein **Polynomring R[X]** die Menge aller Polynome der Form $a_0 + a_1 X + a_2 X^2 + ... + a_n X^n$, wobei $a_0, a_1 ..., a_n \in R$.

**Theorem 1.3.** $\binom{2n+1}{n} > 2^{n+1}, \forall n \geq 2$

*Beweis.* Induktion: Für n = 2 ist es leicht zu sehen, dass $\binom{5}{2} > 2^3$. Sei nun das obere Theorem für ein beliebiges $k > 2$, mit $k \in \mathbb{N}$ erfüllt.

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

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

Die obere Ungleichung (1) lässt sich wie folgt umschreiben:

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

Der erste Teil 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:

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

□

**Theorem 1.4** (**Binomischer Lehrsatz**). *Seien $R$ ein kommutativer Ring und $n$ eine natürliche Zahl, dann gilt für $a, b \in R$:*

$$(a+b)^n = \Sigma_{k=1}^{n} \binom{n}{k} a^k b^{n-k}$$

# 2 Der AKS-Primzahltest

## 2.1 Grundidee des Algorithmus

Die Idee des Algorithmus ist basiert auf Verallgemeinerung des kleineren fermatschen Satz.

**Lemma 2.1.** *seien $a, n \in \mathbb{N}$ mit $a < n$ und teilerfremd, dann ist $n$ genau dann eine Primzahl, wenn*

$$(X+a)^n = X^n + a (mod\ n). \tag{4}$$

.

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

*Beweis.* Aus dem binomischen Lehrsatz(1.4) 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 (1.4) offensichtlich, dass $\forall i$, 0 < i < n, $\binom{n}{i} = \frac{n!}{(n-i)!i!} = 0$ (mod n). Das heißt alle Koeffizienten sind Null.
Für i = 0 erhält man $\binom{n}{0}a^nX^0 = a^n$, analog für i = n, $\binom{n}{n}a^0X^n = X^n$. Daraus folgt:
$(X + a)^n = a^n + 0 + 0 + ... + 0 + X^n = a^n + X^n$(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:

$$\binom{n}{q} \cdot a^{n-q} = \frac{n!}{(n-q)!n!} \cdot a^{n-q} = \frac{n(n-1)\cdots(n-q+1)}{q!} \cdot a^{n-q}. \tag{5}$$

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. Das heißt $(X + a)^n \neq X^n + a$(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, um die Anzahl an Koeffizienten zu reduzieren, dabei wird r so gewählt, dass die Anzahl an Berechnungen kleiner ist als bei (4). Daher das Hauptziel Jetzt ist ein entsprechend kleines r zu wählen und zu testen, ob die Gleichung:

$$(X + a)^n = X^n + a(\text{mod } X^r - 1, n) \tag{6}$$

erfüllt ist.

Nach Lemma 2.14 ist die Gleichung (6) 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 log n

beschränkt, das heißt der Algorithmus muss nur $logn$ a's testen, um eine Entscheidung über die Primalität der Zahl n zu treffen.

## 2.2 Der Algorithmus

**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** $1 < gcd(a, n) < n, a \geq n$, **return** COMPOSITE.
4. **if** $n \leq r$, **return** PRIME.
5. **for** a = 1 to $\lfloor \sqrt{\phi(r)} logn$:
6. **if** $(X + a)^n \neq X^n + a (\text{mod }$ )

# A Anhang A

# 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 11. März 2020 Salman Salman