Universiteit Utrecht

Faculty of Science
Department of Mathematics

Bachelor Thesis

Mathematics

Algorithms for Testing Primality

Lammert Westerdijk

Supervisor:

Dr. S. Yamagishi
Universiteit Utrecht

June 17, 2021

Abstract

Large prime numbers are very important in modern-day information security. To find such large prime numbers, algorithms which can efficiently determine primality for large numbers are needed. Beginning with one of the oldest and most well-known primality testing algorithms, we discuss several ways in which more efficient algorithms can be derived. Some of the algorithms we will look at are probabilistic, in the sense that they will determine a number as “probably prime”. However, assuming a very deep hypothesis in mathematics, we can construct efficient deterministic variants of these probabilistic algorithms. We conclude our theoretical discussion of primality testing algorithms with an algorithm which was discovered at the beginning of this century in one of the biggest breakthroughs in primality testing, since it provides an efficient deterministic method to test for primality without relying on any unproven assumptions. We complement the theoretical discussion with a short practical analysis of the different algorithms presented, where we will see that several of the algorithms perform much more efficient in practice than what has been theoretically proven.

Contents

1 Introduction

At first glance, prime numbers seem like nothing more than an interesting mathematical construct with no practical application; a problem which unfortunately is widespread throughout all areas of mathematics. However, while this might have been true when the ancient mathematicians first described them, the ever-growing need for secure data transfer and storage in our modern society has created a practical field with an immense demand for large prime numbers. To secure data, so-called *cryptograhic algorithms* are used to encrypt the data with a specific key. Only the person with access to this key can decrypt the encrypted data. Several widely used cryptographic algorithms, such as the RSA public key cryptosystem and cryptosystems based on the discrete log problem in finite fields, such as Diffie-Hellman and ECC, use large random prime numbers to guarantee their safety [1].

To generate these large prime numbers, we need an efficient algorithm to test and verify which large numbers are prime. The aim of this thesis is to introduce the reader to the topic of primality testing, by presenting several primality testing algorithms, proving their correctness, and comparing them against each other. We begin by giving a short recap on prime numbers and present the Trial Division algorithm, a primality testing algorithm that has been known since the ancient mathematicians first described the primes. The remainder of Chapter 2 will serve to introduce the Solovay-Strassen and Miller-Rabin primality tests. These two algorithms are *probabilistic*: if an integer satisfies the test, it is very likely to be prime, but there is a small chance that a non-prime integer is incorrectly identified as prime. This compromise in accuracy is contrasted by extreme efficiency; we will see that both probabilistic tests can test an integer with a thousand digits for primality in mere seconds.

To contrast the probabilistic primality testing algorithms presented in Chapter 2, deterministic variants of these algorithms are presented in Chapter 3. To prove that these deterministic variants are also somewhat efficient, we use a consequence of the Extended Riemann Hypothesis. This is a deep hypothesis in analytic number theory which is widely believed to be true. The final algorithm we discuss is the AKS primality testing algorithm. What sets the AKS algorithm apart from the other primality testing algorithms discussed in this thesis is the fact that it was the first efficient deterministic primality test which did not depend on any unproven hypotheses, such as the Extended Riemann Hypothesis.

Finally, we present a theoretical and practical analysis of all the presented algorithms in Chapter 4 and 5 respectively. From this analysis, it will become clear that the probabilistic algorithms are by far the most efficient primality testing algorithms. While the AKS primality testing algorithm is arguably the most impressive algorithm from a theoretical standpoint, we will see that it is many times slower than the other efficient algorithms presented and thus does not have any practical use.

Sometimes, we will somewhat irresponsibly view the elements of $\mathbb{Z}_n$ and $\mathbb{Z}_n^*$ as integers. This will always mean that we make use the unique integer representative $0 \leq a < n$ of the equivalence class in question.

An important final note is that most of the primality testing algorithms we present in this thesis assume that the number being tested for primality is odd. It is however easy to test if an even number is prime, since the only even prime number is 2. Testing if a number is even is also trivial by looking at the last bit of the number in its binary representation, which is the representation in which the number is stored on a modern computer. Therefore, any primality test that tests odd numbers can easily be adapted to also facilitate even numbers.

2 Testing for Primality

We've seen why primes are so important in our daily lives, but to ensure the integrity of the systems and applications that use them, there needs to be a way to confirm that the (often very large) numbers used are indeed prime. The most intuitive primality test is based on the original definition of primes, which has been known for centuries.

2.1 Trial Division

In his *Elements*, Euclid defines prime numbers as follows:

"A prime number is that which is measured by a unit alone."[2]

Keeping in mind that, in ancient mathematics, an integer being *measured* by another meant the latter exactly divides the former, we can transform Euclid's definition into the well-known modern definition of a prime number.

Definition 2.1. A *prime number* is an integer $n \geq 2$ which has no positive divisors other than 1 and n. An integer $n \geq 2$ is called *composite* if it is not prime; it has at least one nontrivial divisor.

This definition leads us to the first primality testing algorithm; the *Trial Division* algorithm.

Algorithm 2.2 (Trial Division)**.** Let $n \geq 2$ be an integer, whose primality we want to test. Then, for all integers $1 < a \leq \sqrt{n}$, test if $a|n$ ("a divides n"). If none do, then n is prime. Otherwise, n is composite.

Proof. Firstly, suppose that n is prime. Then, by Definition 2.1, the only positive divisors of n are 1 and n, so no integer $1 < a \leq \sqrt{n}$ will divide n, since $n \geq 2$ and thus $n > \sqrt{n}$. In this case, the algorithm correctly identifies n as prime.

Now suppose that n is composite. Then there exists at least one non-trivial positive divisor k of n. By the definition of divisibility, this implies that $n = k\ell$, where ℓ is also a positive divisor of n. Now, since $1 < k < n$, we also have $1 < \ell < n$ and we claim that at least one of these is less than or equal to $\sqrt{n}$. Indeed, suppose that $k, \ell > \sqrt{n}$, then it follows that $n = k\ell > n$, a clear contradiction. The algorithm will thus test n for divisibility against at least one of the divisors k and ℓ, correctly concluding that n is composite. $\square$

As an example, we will test the integer $n = 9$ (which we know to be composite) for primality. Since $\sqrt{n} = 3$, we only test for divisibility against the integers $a = 2$ and $a = 3$. Since n is odd, $a = 2$ is not a divisor of n, but $a = 3$ *is*, since $n = 9 = 3 \cdot 3$. We thus see that $n = 9$ is indeed composite by the Trial Division primality test.

Remark 2.3. Note that Algorithm 2.2 immediately provides a factor of n if n is found to be composite, namely the integer a against which n fails the divisibility test. Furthermore, if we test for divisibility against the integers $1 < a \leq \sqrt{n}$ in increasing order, we are guaranteed that this factor is prime, since for any composite number a that divides n, all the prime factors of a (which are strictly smaller than a) also divide n and thus would be tested earlier. This property to immediately yield a prime factor of a composite number is very useful for converting the primality test into a factoring algorithm. Unfortunately, as we will see, most faster primality testing algorithms do not provide us with such luxury[1].

While the idea of trial division might have suited our primality testing needs for centuries, the question has always been whether it can be done faster. Several improvements have been made to the algorithm, the biggest ones being only testing for divibility against *prime* integers a and testing multiple integers for primality at once; eventually resulting in the Sieve of Eratosthenes. However, to test single integers for primality efficiently, we need to consider alternative ways to test integers for primality, not based on divisibility, but on more abstract properties of primes.

2.2 Properties of Primes

The trial division primality test follows almost immediately from the definition of prime numbers as given in Definition 2.1. However, we saw that the resulting algorithm is quite slow. One approach to creating a more efficient (faster) primality testing algorithm is by considering different, more abstract properties of primes. For this, let us first look at one of the most well-known properties of primes: Fermat's Little Theorem.

Theorem 2.4 (Fermat's Little Theorem)**.** *Let p be a prime number and a any integer. Then $a^p \equiv a \mod p$ and, if $(a,p) = 1$, $a^{p-1} \equiv 1 \mod p$. Here (a,p) denotes the greatest common divisor of a and p.*

Proof. We present a more formal version of the combinatorical proof in [3]. The case $p = 2$ is trivial, so suppose that $p > 2$; then p is odd and thus $p-1$ is even. This means that $a^{p-1} = (-a)^{p-1}$, so we can assume that a is non-negative. Now consider *colorings* of $\mathbb{Z}_p$; functions of the form $c : \mathbb{Z}_p \to A$, where A is a set of a distinct "colors". Since there are a possible colors for each of the p elements of $\mathbb{Z}_p$, the total number of such colorings is a^p. Now, let $C \subset A^{\mathbb{Z}_p}$ be the set of colorings c for which there are $i, j \in \mathbb{Z}_p$ with $c(i) \neq c(j)$; the colorings with at least two different colors. Since the total number of colorings is a^p and there are exactly a colorings containing only one color (one for each of the a colors), we know that the number of colorings with at least two different colors is given by

$$|C| = a^p - a.$$

We now claim that the number of elements of C is divisible by p. View the group action $\phi : \mathbb{Z}_p \to S_C$ of $\mathbb{Z}_p$ on C, given by $(\phi(g)(c))(k) = c(k-g)$; shifting a coloring g places to the right. Here, S_C denotes the set of all possible permutations of a coloring. The idea now is to show that the orbit of any $c \in C$ under ϕ contains p distinct elements; every shift yields a different coloring. We prove this by contradiction.

Suppose that for a certain $c \in C$ there are two shifts that produce the same coloring, implying that there are $g, h \in \mathbb{Z}_p$ with $g \neq h$ and $\phi(g)(c) = \phi(h)(c)$. This implies that $c(k-g) = c(k-h)$ for any $k \in \mathbb{Z}_p$. Writing $d = g - h \neq 0$ and $k' = k - g$, we see that $c(k') = c(k'+d)$ for any $k' \in \mathbb{Z}_p$. In particular, this implies that $c(id) = c(0)$ for all $i \in \mathbb{Z}_p$.

Now, on one hand it follows from the definition of C that there is an integer $j \in \mathbb{Z}_p$ with $c(j) \neq c(0)$. On the other hand, since p is prime and $d \neq 0$, there is a $d^{-1} \in \mathbb{Z}_p$, implying that $jd^{-1} \in \mathbb{Z}_p$ and thus $c(j) = c((jd^{-1})d) = c(0)$. This is a clear contradiction. Therefore, $\phi(g)(c) \neq \phi(h)(c)$ for all $g, h \in \mathbb{Z}_p$ with $g \neq h$: every shift will produce a different coloring. Since there are $|\mathbb{Z}_p| = p$ different shifts, it follows that $\{\phi(g)(c) \mid g \in \mathbb{Z}_p\}$, the orbit of $c \in C$ under ϕ, will always contain exactly p distinct colorings.

Since the orbits of a group action on C partition C, and every orbit contains exactly p elements, p must divide the total number of elements in C, which is $a^p - a$. This means that $p \mid a^p - a$ and therefore $a^p \equiv a \mod p$ for any integer a. If $(a,p) = 1$, $a^{-1} \in \mathbb{Z}^p$ exists, implying that

$$a^{p-1} \equiv a^{-1}a^p \equiv a^{-1}a \equiv 1 \mod p.$$

□

Now, if the above statement was to *only* hold for prime numbers p, Fermat's Little Theorem would lead to an alternative definition of primes. Unfortunately, this is not the case and there exist composite numbers n for which there is an integer a with $(a,n) = 1$ and $a^{n-1} \equiv 1 \mod n$. Such composite numbers are called pseudoprimes.

Definition 2.5 (Pseudoprime)**.** A composite number n is called a *pseudoprime to the base b* if $b^{n-1} \equiv 1 \mod n$ for an integer b with $(b,n) = 1$ [1].

For example, the composite number $n = 15$ is a pseudoprime to the base $a = 4$, since $(4,15) = 1$ and $4^{14} = 16^7 \equiv 1^7 \equiv 1 \mod 15$. However, since $2^{14} = 4 \cdot 16^3 \equiv 4 \cdot 1^3 \equiv 4 \mod 15$, we see that 15 is not a pseudoprime to the base 2.

By Fermat's Little Theorem we know that, for a prime number p, the equation $a^{p-1} \equiv 1 \mod p$ holds for any integer with $(a,p) = 1$. This might lead one to think that testing every $a \in \mathbb{Z}_p$ until a counterexample is found (as we did with $n = 15$) would be a good way to test for primality. However, it turns out that this does not work since there are also composite integers that will pass this test for any integer a. Such integers are called Carmichael numbers.

Definition 2.6. A *Carmichael number* is a composite number n such that for any integer b with $(b,n) = 1$, the equation $b^{n-1} \equiv 1 \mod n$ holds; a Carmichael number is a pseudoprime to any base b [1].

Theorem 2.7. *An odd composite number n is a Carmichael number if and only if n is square-free (that is, there is no integer $m > 1$ such that $m^2|n$) and $p-1|n-1$ for every prime divisor p of n.*

Proof. We write $\mathbb{Z}_k^*$ to represent the multiplicative group of all invertible elements modulo k. For any group G, the *order* of an element $g \in G$ is defined as the least integer $i \geq 1$ for which $g^i = 1$ in G. Additionally, a *generator* of a cyclic group G is an element $g \in G$ such that $\{g^i\}_{i \in \mathbb{Z}} = G$.

Suppose that n is not square free, then we see that there is a prime number q such that $q^2|n$. If we take b as a specific generator of $\mathbb{Z}_{q^2}^*$ (which we know to be cyclic) which we will construct below, it follows that the order of b is $|\mathbb{Z}_{q^2}^*| = q(q-1)$. However, since $q|n$ and $n > 1$, we have that $q \nmid n-1$, so $q(q-1) \nmid n-1$ and therefore $b^{n-1} \not\equiv 1 \mod q^2$. In particular, since $q^2|n$, we also have that $b^{n-1} \not\equiv 1 \mod n$, so n is indeed not a Carmichael number. One might argue that b does not need to satisfy $(b,n) = 1$, but by the Chinese Remainder Theorem there is always an integer b' that satisfies $b' \equiv b \mod q^2$ and $b' \equiv 1 \mod n'$ (where n' is the product of all prime factors of n other than p), implying $(b', n) = 1$. This is the specific generator b we choose.

Now suppose that $p - 1 \nmid n - 1$ for some prime divisor p of n. The argument is almost exactly the same as the one given above. This time, take b as a generator of $\mathbb{Z}_p^*$, which we again construct to satisfy $(b,n) = 1$ using the Chinese Remainder Theorem. The order of b is given by $|\mathbb{Z}_p^*| = p-1$. However, since $p-1 \nmid n-1$, we have that $b^{n-1} \not\equiv 1 \mod p$, implying that $b^{n-1} \not\equiv 1 \mod n$, since $p|n$. This means that n is indeed not a Carmichael number, as it is not a pseudoprime to b.

Finally, suppose that n is square-free and satisfies $p-1|n-1$ for any prime divisor p of n. Choose an integer b with $(b,n) = 1$. Since n is square-free, we can write $n = p_1p_2\cdots p_k$, where the p_i are distinct primes and where, by Fermat's Little Theorem, we know that $b^{p_i-1} \equiv 1 \mod p_i$ for all $1 \leq i \leq k$. Since $p_i - 1|n-1$, we have $b^{n-1} \equiv b^{\ell_i(p_i-1)} \equiv 1^{\ell_i} \equiv 1 \mod p_i$ for any integer $1 \leq i \leq k$. We know that $n = p_1p_2\cdots p_k$, from which the Chinese Remainder Theorem now lets us conclude that $b^{n-1} \equiv 1 \mod p_1p_2\cdots p_k (= n)$. Since this holds for any integer b with $(b,n) = 1$, we conclude that n is indeed a Carmichael number. □

As an example, $n = 561 = 3 \cdot 11 \cdot 17$ is a Carmichael number, since it is square-free and $2|560, 10|560$ and $16|560$. This turns out to be the smallest Carmichael number, as shown by Carmichael himself in 1910[4].

The criteria given in Definition 2.6 and Theorem 2.7 give us ways to test if an integer is a Carmichael number, but they give us little insight into how we might find these Carmichael numbers in general. One explicit construction of a subset of Carmichael numbers is given by the following theorem.

Theorem 2.8. *For any positive integer k, the number $n = (6k+1)(12k+1)(18k+1)$ is a Carmichael number if all three of its factors are prime.*

Proof. We will simply check all the criteria from Theorem 2.7. Let k be an integer such that $(6k+1), (12k+1)$ and $(18k+1)$ are all prime. This immediately implies that n is square-free, since it is a product of three distinct primes. Furthermore, by writing out the product for n, we see that

$$n - 1 = 1296k^3 + 396k^2 + 36k.$$

It is now easily checked that $6k|n-1$, $12k|n-1$ and $18k|n-1$, so by Theorem 2.7 we conclude that n is indeed a Carmichael number. □

Remark 2.9. It is not known whether the above construction yields infinitely many Carmichael numbers but this would be implied by Dickson's conjecture, widely believed to be true. However, the existence of infinitely many Carmichael numbers has also unconditionally been proved; it has been shown that for sufficiently large n, there are at least $n^{2/7}$ Carmichael numbers between 1 and n [5].

The existence of Carmichael numbers, these infinite "false positives" to Fermat's Little Theorem, poses a big problem when trying to use Fermat's Little Theorem to test for primality; we cannot distinguish Carmichael numbers from actual primes without doing additional tests (like the criteria in Theorem 2.7). To successfully overcome this problem, stronger versions of Fermat's Little Theorem are needed.

2.3 The Solovay-Strassen Primality Test

Before we can introduce the main idea behind the first primality testing algorithm we will discuss that is faster than the Trial Division algorithm, we first need to introduce the Jacobi symbol.

Definition 2.10 (Legendre and Jacobi symbol)**.** For a prime number p and integer a, we define the *Legendre symbol* $(\frac{a}{p})$ as 0 if $p \mid a$, as 1 if a is a quadratic residue modulo p (there is an integer b such that $b^2 \equiv a \mod p$) and -1 otherwise. We can extend the definition of the Legendre symbol to composite numbers, which results in the *Jacobi symbol.* For any positive odd integer n with prime factorization $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_i^{\alpha_i}$ and integer a, the Jacobi symbol $(\frac{a}{n})$ is defined as the product of Legendre symbols [1]:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_i}\right)^{\alpha_i}.$$

Having introduced the Jacobi symbol, we can now derive the main idea of the first efficient primality testing we present: the following stronger version of Fermat's Little Theorem.

Theorem 2.11. *Let p be an odd prime number and let $(\frac{a}{p})$ denote the Jacobi symbol. Then for any integer a we have*

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \mod p.$$

The proof of this theorem will mainly be based on the proof given in Chapter 5 of [1]. We first prove the following two lemmata.

Lemma 2.12. *For any prime p, an integer a which satisfies $a^2 \equiv 1 \mod p$ must have $a \equiv \pm 1 \mod p$. In other words, the only square roots of 1 in $\mathbb{Z}_p$ are ± 1.*

Proof. Since $a^2 \equiv 1 \mod p$, we know that $p | a^2 - 1 = (a-1)(a+1)$ and since p is prime, this implies that $p | a - 1$ or $p | a + 1$. In other words, $a \equiv \pm 1 \mod p$. □

Lemma 2.13. *Let p be an odd prime number and g a generator of $\mathbb{Z}_p^*$. An integer $a \in \mathbb{Z}_p^*$ is a quadratic residue mod p, if and only if $a \equiv g^k \mod p$ with k even.*

Proof. If $a \equiv g^k \mod p$ with k even, then clearly $a \equiv \left(g^{k/2}\right)^2 \mod p$ is a quadratic residue modulo p. On the other hand, if $a \equiv x^2 \mod p$, then write $x = g^j$ and we find $a \equiv g^{2j} \mod p$. □

Using these two lemmata, we can now prove Theorem 2.11.

Proof (Theorem 2.11). Firstly, if $a \equiv 0 \mod p$, we have $(\frac{a}{p}) = 0$ and $a^{(p-1)/2} \equiv 0^{(p-1)/2} = 0 \mod p$, so the theorem holds. Now suppose that $(a, p) = 1$, so that $(\frac{a}{p}) = \pm 1$. Take g to be a generator of $\mathbb{Z}_p^*$ and write $a \equiv g^k \mod p$. Then, by Lemma 2.13, we know that $(\frac{a}{p}) = 1$ if k is even and $(\frac{a}{p}) = -1$ if k is odd.

If k is even, it follows that

$$a^{(p-1)/2} \equiv \left(g^k\right)^{(p-1)/2} = g^{k(p-1)/2} = \left(g^{p-1}\right)^{k/2} \equiv 1^{k/2} = 1 = \left(\frac{a}{p}\right) \mod p.$$

On the other hand, if k is odd, we can write $k = 2\ell + 1$, so that

$$a^{(p-1)/2} \equiv \left(g^k\right)^{(p-1)/2} = \left(g^{2\ell+1}\right)^{(p-1)/2} = g^{\ell(p-1)}g^{(p-1)/2} \equiv g^{(p-1)/2} \not\equiv 1 \mod p,$$

since the order of g in $\mathbb{Z}_p^*$ is $p-1$. However, by Fermat's Little Theorem, we know that $(a^{(p-1)/2})^2 = a^{p-1} \equiv 1 \mod p$ and, since p is prime, by Lemma 2.12 it follows that $a^{(p-1)/2} \equiv \pm 1 \mod p$. Since $a^{(p-1)/2} \not\equiv 1 \mod p$ it again follows that

$$a^{(p-1)/2} \equiv -1 = \left(\frac{a}{p}\right) \mod p.$$

□

Remark 2.14. Note that this theorem is indeed a stronger version of Fermat's Little Theorem; squaring both sides of the equation implies Fermat's Little Theorem, since $\left(\frac{a}{p}\right) = \pm 1$ for integers a satisfying $(a, p) = 1$.

Using Theorem 2.11, we can define an analog of the pseudoprimes from Definition 2.5; the so-called Euler pseudoprimes.

Definition 2.15 (Euler pseudoprime)**.** An odd composite number n is called an *Euler pseudoprime to the base b* if $(b, n) = 1$ and $b^{(n-1)/2} \equiv \left(\frac{b}{n}\right) \mod n$ for an integer b [1].

The reason that Theorem 2.11 is better suited for a use in primality testing than Fermat's Little Theorem is that there are no analogs to Carmichael numbers for Euler pseudoprimes, as is shown by the following theorem.

Theorem 2.16. *Let n be an odd composite number. Then for at least half of all $a \in \mathbb{Z}_n^*$ we have*

$$a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \mod n.$$

In other words, n is an Euler pseudoprime to at most half of all possible bases $a \in \mathbb{Z}_n^*$.

Proof. The main idea of this proof is based on the proof given in Chapter 5 of [1]. First, we will prove that there is at least one base $b \in \mathbb{Z}_n^*$ to which n is not an Euler pseudoprime. We distinguish two cases, based on whether n is square-free or not.

Firstly, suppose that n is not square-free. Then, by Theorem 2.7, we know that n is not a Carmichael number. This means that there is a $b \in \mathbb{Z}_n^*$ for which $b^{n-1} \not\equiv 1 \mod n$. However, since $b \in \mathbb{Z}_n^*$, we know that $\left(\frac{b}{n}\right) = \pm 1$, which means that $\left(\frac{b}{n}\right)^2 = (\pm 1)^2 = 1 \not\equiv b^{n-1} = \left(b^{(n-1)/2}\right)^2 \mod n$, and therefore $\left(\frac{b}{n}\right) \not\equiv b^{(n-1)/2} \mod n$.

Now suppose that n is square-free and let p be a prime divisor of n. Let $\alpha \in \mathbb{Z}_p$ be a quadratic non-residue modulo p. Since n is square-free we have $(p, n/p) = 1$, by the Chinese Remainder Theorem, there exists an integer b satisfying $b \equiv \alpha \mod p$ and $b \equiv 1 \mod n/p$. We know that p is prime and $\alpha \neq 0$, so $(b, p) = (\alpha, p) = 1$. Furthermore, since $b \equiv 1 \mod n/p$, we have $(b, n/p) = 1$. This means that $(b, n) = 1$. By definition of b, we see that

$$\left(\frac{b}{n}\right) = \left(\frac{b}{p}\right)\left(\frac{b}{n/p}\right) = \left(\frac{\alpha}{p}\right)\left(\frac{1}{n/p}\right) = -1.$$

However, we also have $b^{(n-1)/2} \equiv 1^{(n-1)/2} = 1 \mod n/p$. Since n is odd and composite, we know that $n/p > 2$, which means that $\left(\frac{b}{p}\right) \not\equiv 1 \mod n/p$. Since n/p is a divisor of n, this implies that $b^{(n-1)/2} \equiv 1 \not\equiv \left(\frac{b}{n}\right) \mod n$.

We will now prove that n is an Euler pseudoprime to at most half of the bases $a \in \mathbb{Z}_n^*$. Let $E \subset \mathbb{Z}_n^*$ be the set of Euler pseudoprime bases of n and define the function $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ as $f(k) = bk$, where $b \in \mathbb{Z}_n^*$ satisfies $b^{(n-1)/2} \not\equiv \left(\frac{b}{n}\right) \mod n$, as constructed above. Since the function $g : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$, defined as $g(k) = b^{-1}k$ is both a left- and right-inverse of f, f is bijective.

For $\ell \in E$, by definition we have $\ell^{(n-1)/2} \equiv \left(\frac{\ell}{n}\right) \mod n$ and therefore

$$(f(\ell))^{(n-1)/2} = (b\ell)^{(n-1)/2} = b^{(n-1)/2}\ell^{(n-1)/2} \equiv b^{(n-1)/2}\left(\frac{\ell}{n}\right) \not\equiv \left(\frac{b}{n}\right)\left(\frac{\ell}{n}\right) = \left(\frac{b\ell}{n}\right) = \left(\frac{f(\ell)}{n}\right) \mod n.$$

Here we used the fact that $\ell \in E \subset \mathbb{Z}_n^*$, implying that $\left(\frac{\ell}{n}\right) = \pm 1 \not\equiv 0 \mod n$. By the above incongruence, we see that n is not an Euler pseudoprime base to $f(\ell)$ for any $\ell \in E$, which implies that $\mathrm{Im}_f(E) \subseteq \mathbb{Z}_n^* \backslash E$. Since f is bijective, we see that

$$|E| = |\mathrm{Im}_f(E)| \leq |\mathbb{Z}_n^* \backslash E| = |\mathbb{Z}_n^*| - |E|$$

and thus $|E| \leq \frac{1}{2}|\mathbb{Z}_n^*|$. □

Remark 2.17. Note that Theorem 2.16 indeed implies that there are no Euler pseudoprime analogs to Carmichael numbers. Any odd composite n is guaranteed to not be an Euler pseudoprime to at least half of all possible bases and can therefore certainly not be an Euler pseudoprime to all possible bases.

Remark 2.18. The upper bound given in Theorem 2.16 is in fact the best possible bound if n is a so-called *special Carmichael number*, that is $a^{(n-1)/2} \equiv 1 \mod n$ for all $a \in \mathbb{Z}_n^*$; a special Carmichael number is an Euler pseudoprime to exactly half of all possible bases $a \in \mathbb{Z}_n^*$. An example of a special Carmichael number is 1729; the famous taxicab number. It turns out that the number of special Carmichael numbers is infinite [6]. Therefore, the upper bound given in Theorem 2.16 is asymptotically optimal.

The fact that there are no analogs to Carmichael numbers for Euler pseudoprimes allows us to derive a property that only holds for prime numbers; a direct corollary of Theorem 2.11 and Theorem 2.16 is the following result.

Theorem 2.19. *An odd positive integer $n > 1$ satisfies*

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$$

for all integers $0 < a < n$ if and only if n is prime.

Proof. The result follows immediately from Theorem 2.11 and Theorem 2.16. If n is prime, then we know by Theorem 2.11 that $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$ for all integers $0 < a < n$. If n is composite, then we know by Theorem 2.16 that there is at least one $a \in \mathbb{Z}_n^*$ (which we can represent by an integer $0 < a < n$) that does not satisfy $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$. This completes the proof. □

Remark 2.20. Using Theorem 2.19, we cannot immediately derive a faster primality testing algorithm. In order to be sure that a number is prime, we would need to test at least half of all $a \in \mathbb{Z}_n^*$; this is because only if at least half are tested, we know by Theorem 2.16 that a composite number cannot be an Euler pseudoprime to all the tested bases a. Since trial division only tests for divisibility against $\sqrt{n}$ integers, testing half of all $a \in \mathbb{Z}_n^*$ in this way would require more tests and thus be slower. For a more detailed study of the efficiency of the presented algorithms, we refer the reader to Chapter 4.

To solve this issue, we define a *probabilistic* primality test, instead of the deterministic test described above. This results in the Solovay-Strassen primality testing algorithm [1].

Algorithm 2.21 (Solovay-Strassen)**.** Let $n \geq 2$ be an odd integer, whose primality we want to test. Choose k random integers $0 < a < n$ and for every a test if

(i) $(\frac{a}{n}) \neq 0$ and

(ii) $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$.

If this test fails for any a, n is composite and the test stops. If n passes all of the tests, n is *probably* prime.

Proof. Firstly, suppose that n is prime. Then by the definition of the Jacobi symbol we have $(\frac{a}{n}) \neq 0$ for all integers $0 < a < n$ and, by Theorem 2.19, every integer $0 < a < n$ will also satisfy (ii), implying that the algorithm correctly identifies n as prime.

Now suppose that n is composite. Take a random integer $0 < a < n$ which n is being tested against and distinguish between the cases $(a, n) > 1$ and $(a, n) = 1$. In the case that $(a, n) > 1$, we have $(\frac{a}{n}) = 0$, so that n will fail to satisfy test (i). In the case that $(a, n) = 1$, by Theorem 2.16, we know that n fails to satisfy test (ii) with a probability of at least $\frac{1}{2}$. In both cases, the probability that n satisfies both (i) and (ii) for a random $0 < a < n$ is at most $\frac{1}{2}$ and so, since we choose the k integers $0 < a < n$ at random, the probability that all k integers satisfy both (i) and (ii) for a composite n is at most $\frac{1}{2^k}$. This means that a composite n will be correctly identified as such with a probability of at least $\frac{2^k-1}{2^k}$. □

Remark 2.22. As a consequence of Remark 2.18, note that we can only improve the bound of $\frac{2^k-1}{2^k}$ on the probability that a composite n is correctly identified as such, by using a better upper bound for $|\mathbb{Z}_n^*|$ than n. This is because there are infinitely many special Carmichael numbers which satisfy test (ii) for exactly half of all $a \in \mathbb{Z}_n^*$. Alternatively, we can perform an additional test to determine if n is a special Carmichael number.

2.4 The Miller-Rabin Primality Test

One of the most widely used primality tests, the Miller-Rabin primality test, is based on an even stronger version of Fermat's Little Theorem.

Theorem 2.23. *Let p be an odd prime number and write $p - 1 = 2^s t$ with t odd. Then for any integer a with $(a, p) = 1$, either $a^t \equiv 1 \mod p$ or there exists an integer $0 \leq r < s$ such that*

$$a^{2^r t} \equiv -1 \mod p.$$

Proof. We present our own proof of this theorem. This result is almost a direct corollary of Fermat's Little Theorem and Lemma 2.12. We will proceed by contraction; assume that $a^t \not\equiv 1 \mod p$ and that $a^{2^r t} \not\equiv -1 \mod p$ for any integer $0 \leq r < s$. Define R as the set of integers $0 \leq r \leq s$ for which $a^{2^r t} \not\equiv \pm 1 \mod p$, then $0 \in R$ so $R \neq \emptyset$ and R has a maximal element, say r_{max}. By Fermat's Little Theorem, $a^{2^s t} = a^{p-1} \equiv 1 \mod p$, so $s \notin R$ and $r_{max} < s$. Since r_{max} is the greatest element of R, it follows that $r_{max} + 1 \notin R$ and thus $a^{2^{r_{max}+1} t} \equiv \pm 1 \mod p$. However, since $a^{2^r t} \not\equiv -1 \mod p$ for any integer $0 \leq r < s$ and $a^{2^s t} = a^{p-1} \equiv 1 \mod p$ by Fermat's Little Theorem, we conclude that $a^{2^{r_{max}+1} t} \equiv 1 \mod p$ and thus that

$$\left(a^{2^{r_{max}} t}\right)^2 = a^{2(2^{r_{max}} t)} = a^{2^{r_{max}+1} t} \equiv 1 \mod p.$$

By Lemma 2.12 this implies that $a^{2^{r_{max}} t} \equiv \pm 1 \mod p$ and thus $r_{max} \notin R$, a clear contradiction. □

Remark 2.24. Again, note that this theorem is indeed a stronger version of Fermat's Little Theorem by squaring both sides $(s - r)$ times to derive the congruence $a^{p-1} = a^{2^s t} \equiv 1 \mod p$.

Based on Theorem 2.23, we can define another analog of the pseudoprimes from Definition 2.5 as follows.

Definition 2.25 (Strong pseudoprime)**.** Let n be an odd composite number, and write $n-1=2^s t$ with t odd. Then for an integer b, n is called a *strong pseudoprime to the base b* if $(b,n)=1$ and either of the following hold [1]:

(i) $b^t \equiv 1 \mod n$ or

(ii) $b^{2^r t} \equiv -1 \mod n$ for an integer $0 \le r < s$.

Theorem 2.26. *Let n be an odd composite number. If an integer $b \in \mathbb{Z}_n^*$ is a strong pseudoprime base of n, then it is also an Euler pseudoprime base of n.*

Proof. This proof is mainly based on the proof given in Chapter 5 of [1]. Write $n-1=2^s t$ with t odd and suppose $b \in \mathbb{Z}_n^*$ is a strong pseudoprime base of n. To prove that b is also an Euler pseudoprime base of n, we need to show that $b^{(n-1)/2} \equiv \left(\frac{b}{n}\right) \mod n$.

First, if $b^t \equiv 1 \mod n$, then we know that $(\frac{b}{n})^t = (\frac{b^t}{n}) = (\frac{1}{n}) = 1$. Now, if $(\frac{b}{n}) = 0$ or $(\frac{b}{n}) = -1$, we would have $(\frac{b}{n})^t = 0$ or $(\frac{b}{n})^t = -1$, since t is odd. Therefore we see that $(\frac{b}{n}) = 1$ and it follows that

$$b^{(n-1)/2} = b^{2^{s-1}t} = \left(b^t\right)^{2^{s-1}} \equiv 1^{2^{s-1}} = 1 = \left(\frac{b}{n}\right) \mod n.$$

This proves that b is an Euler pseudoprime base of n if $b^t \equiv 1 \mod n$. Now suppose that there exists an integer $0 \le r < s$ such that $b^{2^r t} \equiv -1 \mod n$. Let p be any prime factor of n and write $p-1 = 2^{s'}t'$ with t' odd. We claim that $s' > r$ and that $(\frac{b}{p}) = -1$ if $s' = r+1$ and $(\frac{b}{p}) = 1$ if $s' > r+1$.

Because t' is odd, raising both sides of the congruence $b^{2^r t} \equiv -1 \mod n$ to the power of t' yields

$$b^{2^r tt'} \equiv (-1)^{t'} = -1 \mod n.$$

However, since $p|n$, this also implies that $b^{2^r tt'} \equiv -1 \mod p$. Now, if $s' \le r$, we see that

$$-1 \equiv b^{2^r tt'} = \left(b^{2^{s'}t'}\right)^{2^{r-s'}t} = \left(b^{p-1}\right)^{2^{r-s'}t} \equiv 1^{2^{r-s'}t} = 1 \mod p,$$

which cannot be true since $p > 2$ because n is odd. This means that $s' > r$. Since $(b^{2^r t'})^t = b^{2^r tt'} \equiv -1 \mod p$, it follows from Theorem 2.11 that

$$\left(\frac{b}{p}\right)^t \equiv \left(b^{(p-1)/2}\right)^t = b^{2^{s'-1}tt'} = \left(b^{2^r tt'}\right)^{2^{s'-r-1}} \equiv (-1)^{2^{s'-r-1}} \mod p.$$

Now if $s' = r+1$, we see that $(\frac{b}{p})^t \equiv (-1)^1 = -1 \mod p$ and thus $(\frac{b}{p}) = -1$ since $p > 2$ and t is odd. If $s' > r+1$, we have $2 \mid 2^{s'-r-1}$ and so $(\frac{b}{p}) \equiv 1 \mod p$, implying that $(\frac{b}{p}) = 1$, again since $p > 2$ and t is odd. This proves the claim stated above.

Let k denote the number of prime factors $p = 2^{s'}t' + 1$ (with t' odd) of n for which $s' = r+1$, counted with multiplicity; if n has $\nu_p(n)$ factors p, we count each of the $\nu_p(n)$ factors individually. From the above, we know that $(\frac{b}{p}) = -1$ for those k prime factors p of n and $(\frac{b}{p}) = 1$ for all other prime factors p of n. Now, by the definition of the Jacobi symbol, we have $(\frac{b}{n}) = \prod_{p|n,p\ prime}(\frac{b}{p})^{\nu_p(n)} = (-1)^k$. We will now calculate the parity of k by calculating $n \mod 2^{r+2}$ in two different ways. Since $p = 2^{s'}t'+1$, we see that $p \equiv 1 \mod 2^{r+2}$ if $s' > r+1$, while $p \equiv 1 + 2^{r+1} \mod 2^{r+2}$ if $s' = r+1$. This implies that

$$n = \prod_{p|n,p\ prime} p^{\nu_p(n)} \equiv \left(1+2^{r+1}\right)^k \equiv 1 + k2^{(r+1)} \mod 2^{r+2},$$

where the last step follows from the Binomial Theorem. Therefore $n \equiv 1 \mod 2^{r+2}$ if and only if k is even and $n \equiv 1 + 2^{r+1} \mod 2^{r+2}$ if and only if k is odd. We now distinguish two cases; $r = s-1$ and $r < s-1$.

Suppose that $r = s-1$, then we know that $n = 2^s t + 1 = 2^{r+1}t + 1 \equiv 1 + 2^{r+1} \mod 2^{r+2}$ and thus k must be odd. This implies that $(\frac{b}{n}) = (-1)^k = -1$ and, since $r = s-1$, we also have

$$b^{(n-1)/2} = b^{2^{s-1}t} = b^{2^r t} \equiv -1 = \left(\frac{b}{n}\right) \mod n.$$

Here $b^{2^r t} \equiv -1 \mod n$ follows from the definition of r. Now suppose that $r < s-1$, then we know that $n = 2^s t + 1 = 2^{s-r-2}2^{r+2} + 1 \equiv 1 \mod 2^{r+2}$ and thus k must be even. This implies that $(\frac{b}{n}) = (-1)^k = 1$ and, since $r < s-1$, we also have

$$b^{(n-1)/2} = b^{2^{s-1}t} = \left(b^{2^r t}\right)^{2^{s-r-1}} \equiv (-1)^{2^{s-r-1}} = 1 = \left(\frac{b}{n}\right) \mod n.$$

This concludes the proof of Theorem 2.26. □

Because every strong pseudoprime is also an Euler pseudoprime, it follows that there are also no analogs of Carmichael numbers when considering strong pseudoprimes. While a composite number can only be an Euler pseudoprime to at most half of all possible bases, it turns out that there are at most a quarter of all possible bases to which n is a strong pseudoprime.

Theorem 2.27. *Let n be an odd composite number and write $n-1 = 2^s t$ with t odd. Then for at least three quarters of all $0 < a < n$ neither of the following hold:*

(i) $a^t \equiv 1 \mod n$ *or*

(ii) $a^{2^r t} \equiv -1 \mod n$ *for an integer* $0 \le r < s$.

In other words, n is a strong pseudoprime to at most a quarter of all possible bases $0 < a < n$.

The proof of this theorem and the required lemmata will again be based on the proofs given in [1]. To prove this theorem, we first prove two lemmata about the number of solutions to (i) and (ii) above.

Lemma 2.28. *Let G be a cyclic group of order m with a generator g and let k be an integer. Then the number of unique elements $x \in G$ which satisfy $x^k = 1$ is exactly $d = (k, m)$.*

Proof. Since g is a generator of G, we know that $G = \{g^i\}_{i=1}^m$. We will now prove that $x = g^i$ satisfies $x^k = 1$ for exactly d of the indices $0 < i \le m$. Note that $x = g^i$ satisfies $x^k = 1$ if and only if $g^{ik} = 1$, and thus if and only if $m|ik$ since g has order m. This holds if and only if $\frac{m}{d}|i\frac{k}{d}$, where $(\frac{m}{d}, \frac{k}{d}) = 1$, since $d = (m, k)$. This is equivalent to $\frac{m}{d}|i$, so that i is a multiple of $\frac{m}{d}$. Since $0 < i \le m$, we know there are exactly d such i, proving the lemma. □

Lemma 2.29. *Let p be an odd prime number and write $p-1 = 2^{s'}t'$ with t' odd. Also, let t and r be integers, where t is odd. Then the number of unique elements $x \in \mathbb{Z}_p^*$ which satisfy $x^{2^r t} \equiv -1 \mod p$ is exactly $2^r(t, t')$ if $r < s'$ and 0 if $r \ge s'$.*

Proof. Since p is prime, we know that $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$, so let g be a generator for $\mathbb{Z}_p^*$. Then $x = g^i$ satisfies $x^{2^r t} \equiv -1 \mod p$ if and only if $g^{2^r it} \equiv -1 \mod p$. Squaring both sides yields $g^{2^{r+1}it} \equiv 1 \mod p$, so $p-1|2^{r+1}it$, but $p-1 \nmid 2^r it$, since g has order $p-1$. On the other hand, if $p-1|2^{r+1}it$ and $p-1 \nmid 2^r it$, then Fermat's Little Theorem and Lemma 2.12 imply that $g^{2^r it} \equiv -1 \mod p$. This implies that $x = g^i$ is a solution to $x^{2^r t} \equiv -1 \mod p$ if and only if

$$2^r it \equiv \frac{p-1}{2} = 2^{s'-1}t' \mod p-1.$$

Because $p-1 = 2^{s'}t'$, it follows that $x = g^i$ is a solution to $x^{2^r t} \equiv -1 \mod p$ if and only if $2^{s'}t' \mid (2^r it - 2^{s'-1}t')$. If $r \ge s'$, then clearly $2^{s'} \mid 2^r$, but $2^{s'} \nmid 2^{s'-1}$ and since t is odd, this implies that $2^{s'} \nmid 2^{s'-1}t$, from which we see that $2^{s'}t' \nmid (2^r it - 2^{s'-1}t')$. Therefore there are no solutions to $x^{2^r t} \equiv -1 \mod p$ in the case that $r \ge s'$.

Now assume that $r < s'$. Then we see that $2^{s'}t' \mid (2^r it - 2^{s'-1}t')$ holds if and only if $2^{s'-r}t' \mid (it - 2^{s'-r-1}t')$, by dividing both sides by 2^r. If we now write $d = (t, t')$, dividing both sides by d gives

$$2^{s'-r}\frac{t'}{d} \;\Big|\; \left(i\frac{t}{d} - 2^{s'-r-1}\frac{t'}{d}\right).$$

This implies that $i\frac{t}{d}$ is an odd multiple of $2^{s'-r-1}\frac{t'}{d}$. Here, we say that a is an odd multiple of b if $a = kb$ with k odd. Now, since $d = (t, t')$ and t is odd, we know that $\frac{t}{d}$ and $2^{s'-r-1}\frac{t'}{d}$ are relatively prime, from which it follows that i must be an odd multiple of $2^{s'-r-1}\frac{t'}{d}$. Since we can write every $x \in \mathbb{Z}_p^*$ as $x = g^i$ with $0 < i \leq p-1 = 2^{s'}t'$, there are $2^{r+1}d$ such i which are multiples of $2^{s'-r-1}\frac{t'}{d}$, half of which are odd multiples, so that, if $r < s'$, there are exactly $2^r d = 2^r(t, t')$ unique elements $x \in \mathbb{Z}_p^*$ which satisfy $x^{2^r t} \equiv -1 \mod p$. □

Proof (Theorem 2.27). We will once again distinguish two cases, based on whether n is square-free or not.

Firstly, suppose that n is not square-free and let p be a prime number such that $p^2|n$. We will show that, in this case, n is not even a pseudoprime, let alone a strong pseudoprime, to more than a quarter of all possible bases $0 < a < n$. If an integer $0 < a < n$ is a base to which n is a pseudoprime, by definition we must have $a^{n-1} \equiv 1 \mod n$ and since $p^2|n$, it follows that we must also have

$$a^{n-1} \equiv 1 \mod p^2.$$

We know that $\mathbb{Z}_{p^2}^*$ is a cyclic group of order $p(p-1)$, so that by Lemma 2.28 there are exactly $(p(p-1), n-1)$ elements $a \in \mathbb{Z}_{p^2}^*$ which satisfy the congruence above. However, since $p|n$, we know that $(p, n-1) = 1$ so that the total number of integers $a \in \mathbb{Z}_{p^2}^*$ which satisfy the above congruence is $(p(p-1), n-1) = (p-1, n-1)$, and is thus at most $p-1$. Since any $a \in \mathbb{Z}_{p^2}$ with $(a, p^2) > 1$ clearly cannot satisfy this congruence, we conclude that at most $p-1$ of all $a \in \mathbb{Z}_{p^2}$ can satisfy $a^{n-1} \equiv 1 \mod p^2$. Now note that, since $p^2|n$, every element $a \in \mathbb{Z}_{p^2}$ has exactly $\frac{n}{p^2}$ elements in $\mathbb{Z}_n$ which reduce to a. Therefore, at most $\frac{n(p-1)}{p^2}$ integers $0 \leq a < n$ can satisfy $a^{n-1} \equiv 1 \mod p^2$, let alone satisfy $a^{n-1} \equiv 1 \mod n$. Now, since this congruence does not hold for $a = 0$, we see that at most $\frac{n(p-1)}{p^2}$ integers $0 < a < n$ can be a pseudoprime base for n. Since $p^2|n$, we know that $p^2 \leq n$ and thus $\frac{n}{p^2} \leq \frac{n-1}{p^2-1}$, which allows us to bound the number of integers $0 < a < n$ which are pseudoprime bases of n by

$$\frac{n(p-1)}{p^2} \leq \frac{(n-1)(p-1)}{p^2-1} = \frac{1}{p+1}(n-1) \leq \frac{1}{4}(n-1).$$

Now suppose that n is square-free and write $n = \prod_{i=1}^k p_i$, where p_i are distinct primes for $1 \leq i \leq k$. In the same way that we wrote $n-1 = 2^s t$, we also write $p_i - 1 = 2^{s_i} t_i$, where all t_i are odd. Now, an equivalence class $a \in \mathbb{Z}_n^*$ is a base to which n is a strong pseudoprime if either (i) or (ii) holds. We will count the number of bases for which (i) and (ii) hold separately.

First we count the number of integers $0 < a < n$ which satisfy (i). If $a^t \equiv 1 \mod n$, we must have $(a, n) = 1$ and $a^t \equiv 1 \mod p_i$ for all $1 \leq i \leq k$. By Lemma 2.28, for any $1 \leq i \leq k$ there are $(t, p_i - 1) = (t, t_i)$ elements $a \in \mathbb{Z}_{p_i}^*$ for which $a^t \equiv 1 \mod p_i$. Now, by the Chinese Remainder Theorem, any combination of these choices of $a \in \mathbb{Z}_{p_i}^*$ leads to a single solution to $a^t \equiv 1 \mod n$ in $\mathbb{Z}_n^*$, so that there are a total of $\prod_{i=1}^k (t, t_i)$ integers $0 < a < n$ which satisfy (i).

Next, we will count the number of integers $0 < a < n$ which satisfy (ii). Let $0 \leq r < s$ be fixed. Then, if $a^{2^r t} \equiv -1 \mod n$, we again must have $(a, n) = 1$ and $a^{2^r t} \equiv -1 \mod p_i$ for all $1 \leq i \leq k$. By Lemma 2.29, for any $1 \leq i \leq k$ there are $2^r(t, t_i)$ elements $a \in \mathbb{Z}_{p_i}^*$ for which $a^{2^r t} \equiv -1 \mod p_i$ if $r < s_i$ and 0 if $r \geq s_i$. Again, by the Chinese Remainder Theorem, any combination of these choices of $a \in \mathbb{Z}_{p_i}^*$ leads to a single solution to $a^{2^r t} \equiv -1 \mod n$ in $\mathbb{Z}_n^*$.

Assuming, without loss of generality, that s_1 is the smallest of the s_i, we see that there are a total of $\prod_{i=1}^k (2^r(t,t_i)) = 2^{kr}\prod_{i=1}^k (t,t_i)$ integers $0 < a < n$ for which $a^{2^r t} \equiv -1 \mod n$ if $r < s_1$ and 0 if $r \geq s_1$. Summing over all $0 \leq r < s$, we see that the total number of integers $0 < a < n$ which satisfy (ii) is equal to

$$\sum_{r=0}^{s_1-1}\left(2^{kr}\prod_{i=1}^k (t,t_i)\right) = \left(\sum_{r=0}^{s_1-1} 2^{kr}\right)\left(\prod_{i=1}^k (t,t_i)\right) = \frac{2^{ks_1}-1}{2^k-1}\prod_{i=1}^k (t,t_i),$$

since no integer $0 < a < n$ can satisfy (ii) for multiple r. We also see that no integer $0 < a < n$ can satisfy both (i) and (ii), so the total number of bases $0 < a < n$ to which n is a strong pseudoprime is exactly

$$\prod_{i=1}^k (t,t_i) + \frac{2^{ks_1}-1}{2^k-1}\prod_{i=1}^k (t,t_i) = \left(1+\frac{2^{ks_1}-1}{2^k-1}\right)\prod_{i=1}^k (t,t_i).$$

We will now bound $\prod_{i=1}^k (t,t_i)$. We first assume that $k \geq 3$, the case $k = 2$ will be handled separately. Since $n = \prod_{i=1}^k p_i$ with $k \geq 2$ because n is composite, we know that

$$n-1 > \prod_{i=1}^k (p_i - 1) = \prod_{i=1}^k (2^{s_i}t_i) = 2^{s_1+\ldots+s_k}\prod_{i=1}^k t_i \geq 2^{ks_1}\prod_{i=1}^k t_i,$$

because s_1 is the smallest of all s_i. Since $0 < (t,t_i) \leq t_i$, we can bound the number of bases $0 < a < n$ to which n is a strong pseudoprime by

$$\left(1+\frac{2^{ks_1}-1}{2^k-1}\right)\prod_{i=1}^k (t,t_i) \leq \left(1+\frac{2^{ks_1}-1}{2^k-1}\right)\prod_{i=1}^k t_i < \left(1+\frac{2^{ks_1}-1}{2^k-1}\right)2^{-ks_1}(n-1).$$

Furthermore, $s_1 \geq 1$, since $p_1 - 1$ is even, so we see that the number of bases $0 < a < n$ to which n is a strong pseudoprime is bounded by

$$\begin{aligned}\left(1+\frac{2^{ks_1}-1}{2^k-1}\right)2^{-ks_1}(n-1) &= \left(2^{-ks_1}\frac{2^k-2}{2^k-1}+\frac{1}{2^k-1}\right)(n-1) \leq \left(2^{-k}\frac{2^k-2}{2^k-1}+\frac{1}{2^k-1}\right)(n-1)\\ &= \left(\frac{1-2^{1-k}}{2^k-1}+\frac{1}{2^k-1}\right)(n-1) = \left(\frac{2-2^{1-k}}{2^k-1}\right)(n-1) = 2^{1-k}(n-1) \leq \frac{1}{4}(n-1).\end{aligned}$$

If we were to repeat the reasoning above for $k = 2$, we would find an upper bound of $\frac{1}{2}(n-1)$, so we need to derive an additional factor of $\frac{1}{2}$. To do this, we first assume that $s_1 < s_2$. Now we can follow the same procedure as above, but replace $2^{-s_1-s_2} \leq 2^{-2s_1}$ by $2^{-s_1-s_2} \leq 2^{-2s_1-1}$ to gain the extra factor of $\frac{1}{2}$. If however $s_1 = s_2$, then we claim that either $(t,t_1) \leq \frac{1}{2}t_1$ or $(t,t_2) \leq \frac{1}{2}t_2$, again yielding the extra factor of $\frac{1}{2}$ we need to complete the proof for $k = 2$. We prove this by contradiction; suppose that $(t,t_1) > \frac{1}{2}t_1$ and $(t,t_2) > \frac{1}{2}t_2$. Then, since $(t,t_1)|t_1$ and $(t,t_2)|t_2$, we have $(t,t_1) = t_1$ and $(t,t_2) = t_2$, which means that $t_1|t$ and $t_2|t$. Since $n = p_1p_2$, we can write

$$2^s t = n-1 = p_1p_2 - 1 = (2^{s_1}t_1+1)(2^{s_2}t_2+1) - 1 = 2^{s_1+s_2}t_1t_2 + 2^{s_1}t_1 + 2^{s_2}t_2.$$

Since t_1 and t_2 divide t, we see that they must divide the right hand side of the above equation as well. Therefore $t_1|2^{s_2}t_2$ and $t_2|2^{s_1}t_1$. However, since t_1 and t_2 are odd, this implies that $t_1|t_2$ and $t_2|t_1$ and since $t_1, t_2 > 0$, we see that $t_1 = t_2$. Now note that we assumed that $s_1 = s_2$, which implies that $p_1 = 2^{s_1}t_1 = 2^{s_2}t_2 = p_2$; a clear contradiction with the fact that n is square-free. Following the same proof as for $k \geq 3$ and including this additional factor of $\frac{1}{2}$, we also complete the proof for $k = 2$. This concludes the proof of Theorem 2.27.

□

Remark 2.30. The derivation of the upper bound of $\frac{1}{4}$ in the proof of Theorem 2.27 seems quite crude. For example, in most cases the bound can be lowered to at least $\frac{1}{8}$: if $(t, t_i) \neq t_i$ for some $1 \leq i \leq k$, if the s_i are not all equal, or if n has more than three distinct prime factors, to name some examples. However, for Carmichael numbers of the form $n = p_1p_2p_3$, where $p_i - 1 = 2t_i$ with t_i odd, we have that $k = 3$ and since $p_i - 1 | n - 1$ also that $t_i | n - 1 = 2^s t$, which implies that $t_i | t$ since t_i is odd. We also see that $s_i = 1$ for all $1 \leq i \leq 3$. In this case, the upper bound of $\frac{1}{4}$ turns out to be a very good bound. We know that infinitely many of such Carmichael numbers n exist with an arbitrarily large smallest prime factor, implying that the bound is even asymptotically optimal [6][7].

As in Theorem 2.19, the fact that there are no analogs to Carmichael numbers for strong pseudoprimes allows us to derive another property that only holds for prime numbers, this time based on Theorem 2.23 and Theorem 2.27.

Theorem 2.31. *An odd positive integer $n > 1$ satisfies either one of*

(i) $a^t \equiv 1 \mod p$ *or*

(ii) $a^{2^r t} \equiv -1 \mod p$ *for some integer* $0 \leq r < s$

for all integers $0 < a < n$ *if and only if* n *is prime.*

Proof. This follows immediately from Theorem 2.23 and Theorem 2.27. If n is prime, then we know by Theorem 2.23 that for every integer a with $(a, n) = 1$, either (i) or (ii) holds. Since n is prime, we see that $(a, n) = 1$ for all $0 < a < n$, so for every integer $0 < a < n$, either (i) or (ii) is satisfied. If n is composite, then we know by Theorem 2.27 that there is at least one $a \in \mathbb{Z}_n^*$ (which we can represent by an integer $0 < a < n$) that does not satisfy either (i) or (ii). This completes the proof. □

In a similar way to how we derived the Solovay-Strassen primality test, we can derive another probabilistic primality test from Theorem 2.27 and Theorem 2.31: the *Miller-Rabin* primality test [1].

Algorithm 2.32 (Miller-Rabin)**.** Let $n \geq 2$ be an odd integer, whose primality we want to test. Firstly, we write $n - 1 = 2^s t$, where t is odd. Next, choose k random integers $0 < a < n$ and for every a test if:

(i) If $a^t \equiv 1 \mod n$, the test passes.

(ii) Compute $a^{2^r t} \mod n$ for every $0 \leq r < s$ by repeatedly squaring $a^t \mod n$. If $a^{2^r t} \equiv -1 \mod n$ for any $0 \leq r < s$, the test passes.

(iii) If we never obtain $a^{2^r t} \equiv -1 \mod n$, meaning that we find $a^{2^r t} \equiv 1 \mod n$ but $a^{2^{r-1} t} \not\equiv -1 \mod n$, the test fails.

If the test fails for any a, then n is composite and the test stops. If n passes all of the tests, n is *probably* prime.

Proof. The proof is very similar to that of the Solovay-Strassen primality test. Firstly, suppose that n is prime. Then, by Theorem 2.23, any $0 < a < n$ will pass the test and thus the algorithm will correctly identify n as prime.

Now suppose that n is composite. Then, by Theorem 2.27, we know that a randomly chosen integer $0 < a < n$ passes the test with probability at most $\frac{1}{4}$. Since we choose the k integers $0 < a < n$ at random, the probability that all k integers $0 < a < n$ pass the test is at most $\frac{1}{4^k}$. This means that a composite n will be correctly identified as such with probability at least $\frac{4^k - 1}{4^k}$. □

Remark 2.33. As a consequence of Remark 2.30, note that we cannot improve the lower bound of $\frac{4^k-1}{4^k}$ on the probability that a composite number n is correctly identified as such, besides using a better upper bound for $|\mathbb{Z}_n^*|$ or doing additional checks to see if n is such a Carmichael number as described in Remark 2.30.

Remark 2.34. Note that a composite n is incorrectly identified by the Miller-Rabin primality test as prime with probability at most $\frac{1}{4^k}$, whereas the Solovay-Strassen primality test incorrectly identifies n as prime with probability at most $\frac{1}{2^k}$, where both probabilities are very good upper bounds by Remarks 2.22 and 2.33. One could argue that this is an improvement of Miller-Rabin over the Solovay-Strassen primality test [1]. However, the same confidence can be achieved with the Solovay-Strassen primality test by testing against $2k$ integers; this constant factor has no impact on the asymptotic running time of the algorithm, as we will see in Chapter 4.

3 Efficient Deterministic Primality Testing Algorithms

In the previous chapter we discussed several primality testing algorithms, but these were either not efficient for large numbers (Trial Division) or non-deterministic (Solovay-Strassen and Miller-Rabin). In this chapter we will discuss three efficient deterministic primality testing algorithms: a deterministic variant of both the Solovay-Strassen and Miller-Rabin algorithms, under the assumption that the Extended Riemann Hypothesis holds, and a deterministic algorithm without any assumptions, known as the AKS primality testing algorithm. We will later explicitly define an efficient algorithm as an algorithm which is polynomial. This will be defined in Chapter 4.

3.1 Assuming The Extended Riemann Hypothesis

Both the Solovay-Strassen and Miller-Rabin algorithms are efficient robabilistic primality testing algorithms. However, by Theorems 2.16 and 2.26, we know that, for any odd composite number n, there must be an integer $0 < a < n$ which is not an Euler pseudoprime base of n and therefore also not a strong pseudoprime base of n. We can therefore construct deterministic variants of the Solovay-Strassen and Miller-Rabin algorithms by testing all integers $0 < a < n$, instead of picking k of these integers at random. These algorithms would however not be very efficient, since they must test half or a quarter of all possible bases, respectively. The main idea which will allow us to construct more efficient variants of the aforementioned algorithms is to reduce the upper bound of n for the existence of an Euler pseudoprime to a much smaller bound than $\frac{1}{2}n$ or $\frac{1}{4}n$. We will be working towards the following theorem, which depends on the Extended Riemann Hypothesis: an unproven hypothesis in analytic number theory which is widely believed to be true [1] [8] [9].

Theorem 3.1. *For any odd composite number n, assuming the Extended Riemann Hypothesis (ERH), there exists an integer $0 < a < 2\ln^2 n$ which is not an Euler pseudoprime base of n.*

Except for Theorem 3.3, this theorem and all results leading up to it are our own, based on a remark in Chapter 5 of [1]. Before proving this theorem, we will first need to show an important property of the set of Euler pseudoprime bases, which will provide us with enough knowledge about its structure to prove Theorem 3.1.

Theorem 3.2. *Let n be an odd composite number and define $E_n \subset \mathbb{Z}_n^*$ as the set of Euler pseudoprime bases of n. Then E_n is a proper subgroup of $\mathbb{Z}_n^*$.*

Proof. By Theorem 2.16 we know that E_n is a proper subset of $\mathbb{Z}_n^*$ so it suffices to show that E_n is a subgroup of $\mathbb{Z}_n^*$. Note that $1 \in E_n$, since $(\frac{1}{n}) = 1 = 1^{(n-1)/2}$. Now suppose that $a \in E_n$; we will show that $a^{-1} \in E_n$. Since $a \in \mathbb{Z}_n^*$, we know that $(\frac{a}{n}) = \pm 1$ and so $(\frac{a^{-1}}{n}) = (\frac{a}{n})^{-1}$. Since a is an Euler pseudoprime base of n, it follows that

$$\left(a^{-1}\right)^{(n-1)/2} = \left(a^{(n-1)/2}\right)^{-1} \equiv \left(\frac{a}{n}\right)^{-1} = \left(\frac{a^{-1}}{n}\right) \mod n$$

and therefore $a^{-1} \in E_n$. Finally, suppose that $a, b \in E_n$; we will show that $ab \in E_n$. Since the Jacobi symbol is multiplicative and a, b are Euler pseudoprime bases of n, it follows that

$$(ab)^{(n-1)/2} = a^{(n-1)/2} b^{(n-1)/2} \equiv \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) = \left(\frac{ab}{n}\right) \mod n.$$

This implies that $ab \in E_n$, concluding the proof. □

The reason that we show that E_n is a proper subgroup of $\mathbb{Z}_n^*$ is that it allows us to make use of a very strong consequence of the ERH.

Theorem 3.3 (Bach, 1990)**.** *For any integer $n > 2$, assuming the ERH, the group $\mathbb{Z}_n^*$ is generated by its elements smaller than $2\ln^2 n$.*

Proof. This is proved by Bach in [9], who provides an upper bound for the constant in the bound of $O(\ln^2 n)$, originally given by Ankeny in [10]. □

Proof (Theorem 3.1). We present a proof by contradiction; assume there exists an odd composite number n for which every integer $0 < a < 2\ln^2 n$ is an Euler pseudoprime base to n. If we define E_n as in Theorem 3.2, this implies that $a \in E_n$ for every $a \in \mathbb{Z}_n^*$ with $0 < a < 2\ln^2 n$. By Theorem 3.3, these elements generate $\mathbb{Z}_n^*$ and since Theorem 3.2 implies that E_n is a group, $\mathbb{Z}_n^*$ is a subgroup of E_n. In particular, this implies that $\mathbb{Z}_n^* \subseteq E_n$. However, Theorem 3.2 implies that E_n is a proper subgroup of $\mathbb{Z}_n^*$, which in turn implies that $E_n \subset \mathbb{Z}_n^*$. Combining these statements, we see that $E_n \subset \mathbb{Z}_n^* \subseteq E_n$, a clear contradiction since E_n cannot be a proper subset of itself. □

Based on Theorem 3.1 we can now derive an efficient deterministic primality test based on the Solovay-Strassen test, under the assumption that the ERH is true.

Algorithm 3.4. Let $n \geq 2$ be an odd integer, whose primality we want to test. Then, for all integers $0 < a < \min\{n, 2\ln^2 n\}$, test if

(i) $(\frac{a}{n}) \neq 0$ and

(ii) $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$.

If this test fails for any a, then n is composite and the test stops. If n passes all of the tests, n is prime or the ERH is false.

Proof. Since this algorithm is almost exactly the same as the Solovay-Strassen algorithm, we present an almost identical proof. Firstly, suppose that n is prime. Then by the definition of the Jacobi symbol we have $(\frac{a}{n}) \neq 0$ for all integers $0 < a < \min\{n, 2\ln^2 n\} \leq n$ and, by Theorem 2.19, every integer $0 < a < \min\{n, 2\ln^2 n\} \leq n$ will also satisfy (ii), implying that the algorithm correctly identifies n as prime.

Now suppose that n is composite. By Theorem 3.1 we know that either the ERH is false, or at least one of the integers $0 < a < 2\ln^2 n$ is not an Euler pseudoprime base of n. By Theorem 2.16 we also know that at least one of the integers $0 < a < n$ is not an Euler pseudoprime base of n. Therefore either (ii) will fail, or the ERH is false. We thus see that if n is composite, the proposed algorithm will correctly identify n as such, or the ERH is false. From this we conclude that, if the algorithm identifies n as prime, either n is prime or the ERH is false, completing the proof. □

Since every strong pseudoprime base is also an Euler pseudoprime base by Theorem 2.26, we can derive the following corollary to Theorem 3.1.

Corollary 3.5. *For any odd composite number n, assuming the Extended Riemann Hypothesis (ERH), there exists an integer $0 < a < 2\ln^2 n$ which is not a strong pseudoprime base of n.*

Proof. From Theorem 3.1 we know that there exists an integer $0 < a < 2\ln^2 n$ which is not an Euler pseudoprime base of n. Since every strong pseudoprime base is also an Euler pseudoprime base by Theorem 2.26, a cannot be a strong pseudoprime. This concludes the proof. □

Based on Corollary 3.5, under the assumption that the ERH is true, we can also derive an efficient deterministic primality test based on the Miller-Rabin test: the Miller primality test.

Algorithm 3.6 (Miller)**.** Let $n \geq 2$ be an odd integer, whose primality we want to test. Firstly, we write $n-1 = 2^s t$, where t is odd. Then, for all integers $0 < a < \min\{n, 2\ln^2 n\}$, test if:

(i) If $a^t \equiv 1 \mod n$, the test fails.

(ii) Compute $a^{2^r t} \mod n$ for $0 \leq r < s$ by repeatedly squaring $a^t \mod n$. If $a^{2^r t} \equiv -1 \mod n$ for any $0 \leq r < s$, the test passes.

(iii) If we never obtain $a^{2^r t} \equiv -1 \mod n$, meaning that we find $a^{2^r t} \equiv 1 \mod n$ but $a^{2^{r-1} t} \not\equiv -1 \mod n$, the test fails.

If this test fails for any a, then n is composite and the test stops. If n passes all of the tests, n is prime or the ERH is false.

Proof. Since this algorithm is almost exactly the same as the Miller-Rabin algorithm, we present an almost identical proof. Firstly, suppose that n is prime. By Theorem 2.23, any $0 < a < \min\{n, 2\ln^2 n\} \leq n$ will pass the test and thus the algorithm will correctly identify n as prime. This means that any number n identified as composite must indeed be composite.

Now suppose that n is composite. By Corollary 3.5, we know that at least one of the integers $0 < a < 2\ln^2 n$ is not a strong pseudoprime base of n, assuming the ERH. By Theorem 2.27 we also know that at least one of the integers $0 < a < n$ is not a strong pseudoprime base of n. Therefore either the test will fail, assuming the ERH. We therefore see that if n is composite, the Miller algorithm will correctly identify n as such, or the ERH is false. From this we conclude that, if the algorithm identifies n as prime, either n is prime or the ERH is false, completing the proof. □

Remark 3.7. In actuality, the Miller primality test was published before the Miller-Rabin primality test. Miller derived the test in 1976, based on the ERH [8]. This test was then altered in 1980 by Rabin to remove the depencency on the ERH, resulting in the Miller-Rabin primality test [7].

3.2 The AKS Primality Test

While the algorithms mentioned above are efficient and deterministic, their correctness depends on the Extended Riemann Hypothesis. However, in a major breakthrough in 2004, an unconditional efficient deterministic primality test was published: the AKS primality test [11][12]. The information in this chapter will be heavily based on the original paper [11], with some ideas from other sources [13][14][15]. As with the primality tests mentioned in the previous chapter, the AKS primality test makes use of a stronger version of Fermat's Little Theorem.

Theorem 3.8. *Let p be a prime number and a any integer. Then we have the polynomial identity*

$$(X+a)^p \equiv X^p + a \mod p,$$

where X is a variable over $\mathbb{Z}_p$.

Proof. We present our own proof. Using the Binomial Theorem, we can expand $(X+a)^p$ as

$$(X+a)^p = \sum_{i=0}^{p} \binom{p}{i} a^i X^{p-i} = \sum_{i=0}^{p} \frac{p!}{i!(p-i)!} a^i X^{p-i}.$$

Since p is prime, we have $p \nmid i!(p-i)!$ for $0 < i < p$, since both factorials are products of integers smaller than p. Since $p \mid p!$, it follows that $p \mid \frac{p!}{i!(p-i)!}$ for $0 < i < p$. Therefore, modulo p, the above equation reduces to

$$(X+a)^p = \sum_{i=0}^{p} \frac{p!}{i!(p-i)!} a^i X^{p-i} \equiv X^p + a^p \mod p.$$

Now the result follows immediately from Fermat's Little Theorem. □

Remark 3.9. Note that this theorem is indeed a stronger version of Fermat's Little Theorem; this is implied by setting $X = 0$. However, what sets the above theorem apart from Fermat's Little Theorem (and any other stronger version mentioned thus far), is the fact that there are no a with $(a, p) = 1$ for which the above holds for composite p. There is thus no need to define an analog of pseudoprimes for the above theorem. This result is implied by the following theorem.

Theorem 3.10. *Let $n \geq 2$ and a be integers, with $(a, n) = 1$. If the polynomial identity*

$$(X + a)^n \equiv X^n + a \mod n$$

holds with X a variable over $\mathbb{Z}_n$, then n is prime.

The proof of this theorem will be our own. We first show the following lemma.

Lemma 3.11. *Let n be a composite number. Consider a prime q that is a factor of n and let q^k be the largest power of q that divides n. Then $q^k \nmid \binom{n}{q}$.*

Proof. We will count factors of q of $\binom{n}{q}$. Note that $q!$ has only a single factor of q, since q is prime. On the other hand, the quotient

$$\frac{n!}{(n-q)!} = (n - q + 1)(n - q + 2) \cdots (n - 1)n$$

has k factors of q, since none of the integers $n - q + 1, n - q + 2, \ldots, n - 1$ are divisible by q and n has exactly k factors of q. Therefore, the binomial coefficient $\binom{n}{p} = \frac{n!}{q!(n-q)!}$ only has $k - 1$ factors of q and thus $q^k \nmid \binom{n}{q}$. □

Using the above lemma, proving Theorem 3.10 becomes almost trivial.

Proof (Theorem 3.10). We present a proof by contradiction. Assume that n is composite. Let q be a prime factor of n and let q^k be the largest power of q that divides n. It follows from Lemma 3.11 that $q^k \nmid \binom{n}{q}$ and so $n \nmid \binom{n}{q}$. Since $(a, n) = 1$, we deduce that $n \nmid \binom{n}{q}a^q$, which is the coefficient of X^{n-q} in $(X + a)^n$. It follows that X^{n-q} has a non-zero coefficient modulo n in $(X + a)^n$. However, since $(X + a)^n \equiv X^n + a \mod n$, this implies that $n - q = 0$ or $n - q = n$. Since $q > 0$, we must have $n = q$. Since q is prime, this implies that n is also prime, a clear contradiction. □

We could use Theorems 3.8 and 3.10 to devise a deterministic primality test. However, this test would not be very efficient since the polynomial $(X + a)^n$ has $n + 1$ coefficients, which in the worst case must all be computed, leading to a similar efficiency to that of Trial Division. To solve this problem, we decrease the degree of polynomials we are working with by evaluating both sides modulo $X^r - 1$ for some small r [14][13]. The resulting equation becomes

$$(X + a)^n \equiv X^n + a \mod n, X^r - 1,$$

where the notation $f(X) = g(X) \mod n, X^r - 1$ represents the equation $f(X) = g(X)$ in the ring $\mathbb{Z}_n[X]/(X^r - 1)$ [11]. Clearly, any prime n will satisfy the congruence above for all values of a and r by Theorem 3.8. However, it is now also possible for a composite n and some a and r to satisfy the above equation, for example, we have

$$(X + 2)^6 = X^6 + 12X^5 + 60X^4 + 160X^3 + 240X^2 + 192X + 64 \equiv X^6 + 4X^3 + 4 \equiv X^6 + 2 \mod 6, X^3 - 1$$

but $6 = 2 \cdot 3$ certainly is not prime. The main idea of the AKS primality test to overcome this problem is to choose r large enough and test if $(X + a)^n \equiv X^n + a \mod n, X^r - 1$ for enough values of a to ensure that n must be prime [13]. This is illustrated by the following theorem.

Theorem 3.12. *Let $n > 1$ be an odd integer and r an integer such that $(a, n) = 1$ for all integers $1 \leq a \leq r$ and $o_r(n) > \lg^2 n$. Here $o_r(n)$ represents the order of n in the multiplicative group $\mathbb{Z}_r^*$ and $\lg n$ represents the base 2 logarithm of n. Further, suppose that for all integers $1 \leq a \leq \sqrt{r} \lg n$, the polynomial identity*

$$(X + a)^n \equiv X^n + a \mod n, X^r - 1$$

holds. Then n is either prime, or a power of a prime.

Since it is the underlying idea of the AKS primality test, proving Theorem 3.12 will be the main goal of the this section. However, we first need a definition and several theorems to aid us in the proof of Theorem 3.12. From this point on, let r be as in Theorem 3.12 and choose p as a prime divisor of n with $o_r(p) > 1$. Such a prime divisor must exist, since $o_r(n) > 1$.

Definition 3.13 (Introspective number)**.** Let $m > 0$ be a positive integer. We say that m is *introspective* for a polynomial $f(X) \in \mathbb{Z}[X]$ if [11]

$$(f(X))^m \equiv f(X^m) \mod p, X^r - 1.$$

The following two theorems serve to explore the structure of introspective numbers. Their proofs are based on the proofs given in [11].

Theorem 3.14. *If m and m' are introspective for a polynomial $f(X)$, then so is mm'.*

Proof. Since m' is introspective for $f(X)$, we know that

$$f\left(X^{m'}\right) \equiv (f(X))^{m'} \mod p, X^r - 1.$$

Replacing X with X^m, we see that

$$f\left(X^{mm'}\right) \equiv (f(X^m))^{m'} \mod p, X^{mr} - 1.$$

Since $r \mid mr$, we know that $X^r - 1 \mid X^{mr} - 1$ and so the above congruence implies that

$$f\left(X^{mm'}\right) \equiv (f(X^m))^{m'} \equiv (f(X))^{mm'} \mod p, X^r - 1,$$

where the second congruence follows from the fact that m is introspective for $f(X)$. This concludes the proof of Theorem 3.14. □

Theorem 3.15. *If m is introspective for $f(X)$ and $g(X)$, it is also introspective for $f(X)g(X)$.*

Proof. This follows immediately from the definition of introspectivity:

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

□

Lastly, we also require the following theorem to prove Theorem 3.12. This is our own theorem.

Theorem 3.16. *Let $f(X)$ and $g(X)$ be polynomials. If*

$$f(X^p) \equiv g(X^p) \mod p, X^r - 1.$$

Then $f(X)$ and $g(X)$ must also satisfy

$$f(X) \equiv g(X) \mod p, X^r - 1.$$

Proof. We can write $f(X) \equiv \sum_{i=0}^{r-1} a_i X^i \mod p, X^r - 1$ and $g(X) \equiv \sum_{i=0}^{r-1} b_i X^i \mod p, X^r - 1$ with $a_i, b_i \in \mathbb{Z}$ as their unique representatives in $\mathbb{Z}_p[X]$ with degree less than r and $0 \leq a_i, b_i < p$.

We now note that $X^u \equiv X^v \mod p, X^r - 1$ if and only if $u \equiv v \mod r$. If $u \equiv v \mod r$, it follows from $X^r \equiv 1 \mod p, X^r - 1$ that $X^u \equiv X^v \mod p, X^r - 1$. To prove the opposite implication, write $u \equiv u' \mod r$ and $v \equiv v' \mod r$ with $0 \leq u', v' < r$. Now since $X^r \equiv 1 \mod p, X^r - 1$, it follows that

$$X^{u'} \equiv X^u \equiv X^v \equiv X^{v'} \mod p, X^r - 1.$$

This implies that $X^r - 1$ divides $X^{u'} - X^{v'}$ in $\mathbb{Z}_p[X]$, but the latter polynomial has degree less than r. Therefore $X^{u'} - X^{v'} = 0$ in $\mathbb{Z}_p[X]$ and thus $u' = v'$. It follows that $u \equiv u' = v' \equiv v \mod r$.

For each $0 \leq i < r$, define $m_p(i)$ as the unique representation of $ip \pmod r$ with $0 \leq m_p(i) < r$. Since $(n,r) = 1$, we know that $(p,r) = 1$, so that the map $i \mapsto m_p(i) : \{0,1,\ldots,r-1\} \to \{0,1,\ldots,r-1\}$ is a bijection. Denote the inverse of this map by m_p^{-1}. By the fact that $X^u \equiv X^v \mod p, X^r - 1$ if and only if $u \equiv v \mod r$, we see that $X^{ip} \equiv X^{m_p(i)} \mod p, X^r-1$, so that we can write $f(X^p)$ as

$$f(X^p) \equiv \sum_{i=0}^{r-1} a_i \left(X^p\right)^i = \sum_{i=0}^{r-1} a_i X^{ip} \equiv \sum_{i=0}^{r-1} a_i X^{m_p(i)} = \sum_{j=0}^{r-1} a_{m_p^{-1}(j)} X^j \mod p, X^r - 1.$$

Completely analogous, we can write $g(X^p)$ as

$$g(X^p) \equiv \sum_{j=0}^{r-1} b_{m_p^{-1}(j)} X^j \mod p, X^r - 1.$$

The fact that $f(X^p) \equiv g(X^p) \mod p, X^r - 1$ now implies that

$$\sum_{j=0}^{r-1} a_{m_p^{-1}(j)} X^j \equiv \sum_{j=0}^{r-1} b_{m_p^{-1}(j)} X^j \mod p, X^r - 1.$$

Since both polynomials are of degrees less than r, this implies that $a_{m_p^{-1}} \equiv b_{m_p^{-1}(j)} \mod p$ for each $0 \leq j < r$. Since m_p^{-1} is the inverse of a bijective map, it is also bijective and thus we see that $a_i \equiv b_i \mod p$ for each $0 \leq i < r$. It now follows that

$$f(X) \equiv \sum_{i=0}^{r-1} a_i X^i \equiv \sum_{i=0}^{r-1} b_i X^i \equiv g(X) \mod p, X^r - 1,$$

as desired. $\square$

With this theorem, we now have all the tools we need to prove Theorem 3.12.

Proof (Theorem 3.12). This proof is mainly based on the proof given in [11], influenced by the proofs in [14] and [13]. We provide a proof by contradiction, so assume that n is composite and not a power of a prime and that for all integers $1 \leq a \leq \sqrt{r}\lg n$ we have

$$(X+a)^n \equiv X^n + a \mod n, X^r - 1.$$

This implies that n is introspective for the polynomials $X + a$ with $1 \leq a \leq \sqrt{r}\lg n$, since $p \mid n$. As a consequence of Theorem 3.8, we know that

$$(X+a)^p \equiv X^p + a \mod p, X^r - 1,$$

so that p is also introspective for the polynomials $X + a$ with $1 \leq a \leq \sqrt{r}\lg n$. We will now show that $\frac{n}{p}$ is introspective for these polynomials (this is an integer because $p \mid n$). Since n and p are introspective to any of the polynomials $X + a$ with $1 \leq a \leq \sqrt{r}\lg n$, we see that

$$(X^p + a)^{\frac{n}{p}} \equiv ((X+a)^p)^{\frac{n}{p}} = (X+a)^n \equiv X^n + a = (X^p)^{\frac{n}{p}} + a \mod p, X^r - 1$$

for any $1 \leq a \leq \sqrt{r}\lg n$. From Theorem 3.16 it now follows that

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

so that $\frac{n}{p}$ is also introspective for the polynomials $X + a$ with $1 \leq a \leq \sqrt{r}\lg n$.

Clearly, 1 is introspective for any polynomial and any number is introspective for X and 1. Together with the fact that $\frac{n}{p}$ is also introspective for the polynomials $X + a$ with $1 \leq a \leq \sqrt{r}\lg n$, Theorems 3.14 and 3.15 imply that any integer of the form $(\frac{n}{p})^i p^j$ with $i, j \geq 0$ is introspective for all polynomials of the form

$$\prod_{a=0}^{\lfloor \sqrt{r}\lg n \rfloor} (X+a)^{e_a}$$

with $e_a \geq 0$. Denote the set of all such polynomials as P and write $I = \{(\frac{n}{p})^i p^j \mid i,j \geq 0\}$. We will now show that the existence of this many introspective numbers is in violation of the factor theorem, which states that a polynomial over a field $\mathcal{F}$ has at most as many roots as its degree [13]. However, to apply the factor theorem, we need to work over a field, which $\mathbb{Z}_p[X]/(X^r-1)$ is not since $X^r - 1$ is reducible in $\mathbb{Z}_p[X]$. To remedy this problem, we consider the field

$$F := \mathbb{Z}_p[X]/(h(X)),$$

where $h(X)$ is an irreducible factor of (X^r-1). In particular, we choose $h(X)$ to be an irreducible factor of the r^{th} cyclotomic polynomial $\Phi_r(X)$ in $\mathbb{Z}_p[X]$. The r^{th} cyclotomic polynomial $\Phi_r(X)$ is defined as the unique polynomial of least degree which is a factor of $X^r - 1$, but not of $X^i - 1$ in $\mathbb{Z}[X]$ for any integer $1 \leq i < r$. It is a known fact that any irreducible factor of $\Phi_r(X)$ modulo p has degree $o_r(p)$ and so $h(X)$ has degree $o_r(p) > 1$ by choice of p. Since $h(X)$ is irreducible in $\mathbb{Z}_p[X]$, we know that F is indeed a field.

We now define two groups. The first group is the set of all residues of integers in I modulo r, given by

$$G := \{i \mod r \mid i \in I\}.$$

We further define $t = |G|$. The second group we consider is the set of all residues of polynomials in P modulo $h(X)$ and p, given by

$$\mathcal{G} := \{f(X) \mod p, h(X) \mid f(X) \in P\}.$$

In other words, $\mathcal{G}$ is the reduction of all polynomials in P to the field F. Since we chose $h(X)$ to have degree $o_r(p) > 1$, we see that none of the polynomials $X + i$ are 0 when reduced modulo $h(X)$ and p. We can even show that none of the elements of $\mathcal{G}$ are 0 [14]. The following two lemmata will now impose contradictory upper and lower bounds on the size of $\mathcal{G}$.

Lemma 3.17. *Let $\mathcal{G}$ be defined as above. Then $|\mathcal{G}| \leq n^{\sqrt{t}}$.*

Proof. This proof is based on the proofs given in [11] and [14]. To bound the number of elements of $|\mathcal{G}|$ from above, the idea is to construct a polynomial with coefficients in F for which all elements of $\mathcal{G}$ are its roots. If we can construct such a polynomial with small enough degree, the factor theorem yields an effective bound.

To construct this polynomial, we use the Pigeonhole Principle. Consider the elements $(\frac{n}{p})^i p^j \in I$ with $0 \leq i, j \leq \lfloor\sqrt{t}\rfloor$. Since n is a composite number that is not a prime power, we know that n must have a factor coprime to p. Therefore $(\frac{n}{p})^i p^j \neq (\frac{n}{p})^{i'} p^{j'}$ if $(i,j) \neq (i',j')$. This implies that there are $(\lfloor\sqrt{t}\rfloor + 1)^2 > t$ such elements of I and since G is a reduction of elements of I modulo r with $|G| = t$, by the Pigeonhole Principle two of these elements in I must reduce to the same element in G. In other words, we must have two distinct tuples (i,j) and (i',j') with $0 \leq i,j,i',j' \leq \lfloor\sqrt{t}\rfloor$ and

$$\left(\frac{n}{p}\right)^i p^j \equiv \left(\frac{n}{p}\right)^{i'} p^{j'} \mod r.$$

For simplicity, we will write $m = (\frac{n}{p})^i p^j$ and $m' = (\frac{n}{p})^{i'} p^{j'}$ and assume without loss of generality that $m > m'$. Then $m \equiv m' \mod r$ and so

$$X^m = X^{m'+kr} = (X^r)^k X^{m'} \equiv X^{m'} \mod X^r - 1.$$

We now define the polynomial $Q \in F[Y]$ as $Q(Y) = Y^m - Y^{m'}$. Since $m > m'$, the degree of Q is m, which we can bound by

$$\deg(Q) = m = \left(\frac{n}{p}\right)^i p^j \leq \left(\frac{n}{p}\right)^{\lfloor\sqrt{t}\rfloor} p^{\lfloor\sqrt{t}\rfloor} = n^{\lfloor\sqrt{t}\rfloor} \leq n^{\sqrt{t}}.$$

We will now show that every element of $\mathcal{G}$ is indeed a root of $Q(Y)$. Take any $g(X) \in \mathcal{G}$ and let $f(X) \in P$ be a polynomial which reduces to $g(X)$ modulo $h(X)$ and p, which exists by the definition of $\mathcal{G}$. Since every element of I is introspective for all polynomials in P, we know that m and m' are introspective for $f(X)$ and so

$$(f(X))^m \equiv f(X^m) \equiv f(X^{m'}) \equiv (f(X))^{m'} \mod p, X^r - 1,$$

where the second congruence follows from the fact that $X^m \equiv X^{m'} \mod X^r - 1$. Since $h(X)$ is a factor of $X^r - 1$, this implies that $(f(X))^m \equiv (f(X))^{m'} \mod p, h(X)$. Since $f(X)$ reduces to $g(X)$ modulo p and $h(X)$, this implies that $(g(X))^m = (g(X))^{m'}$ and thus

$$Q(g(X)) = (g(X))^m - (g(X))^{m'} = 0 \in F.$$

Since every element of $\mathcal{G}$ is a root of $Q(Y)$, we know that $Q(Y)$ must have at least $|\mathcal{G}|$ roots. On the other hand, $Q(Y)$ is a degree $m \leq n^{\sqrt{t}}$ non-zero polynomial and thus has at most $\deg(Q) \leq n^{\sqrt{t}}$ roots by the factor theorem. It follows that $|\mathcal{G}| \leq n^{\sqrt{t}}$. □

Lemma 3.18. *Let $\mathcal{G}$ be defined as above. Then* $|\mathcal{G}| \geq \binom{t+\lfloor\sqrt{r}\lg n\rfloor}{t-1}$.

Proof. This proof is based on the proof given in [11]. We again use our specific choice of $h(X)$ in this proof. To bound the number of elements of $|\mathcal{G}|$ from below, we explicitly construct several elements of $\mathcal{G}$ and prove all these elements are distinct.

We define $P' := \{f(X) \in P \mid \deg(f) < t\}$, where $\deg(f)$ is the degree of $f(X)$. First, we show that $|P'| = \binom{t+\lfloor\sqrt{r}\lg n\rfloor}{t-1}$. By definition, any polynomial $f(X) \in P'$ is of the form

$$f(X) = \prod_{a=0}^{\lfloor\sqrt{r}\lg n\rfloor} (X+a)^{e_a}$$

with $\sum_{a=0}^{\lfloor\sqrt{r}\lg n\rfloor} e_a < t$ and $e_a \geq 0$ for all a. Since $P \subseteq \mathbb{Z}[X]$, all of the polynomials $(X+a)$ are irreducible and distinct, implying that there is a trivial bijection between $f(X) \in P'$ and non-negative sequences $\{e_a\}_{a=0}^{\lfloor\sqrt{r}\lg n\rfloor}$ with $\sum_{a=0}^{\lfloor\sqrt{r}\lg n\rfloor} e_a < t$, given by the above formula for $f(X)$. By a stars and bars argument, we can deduce that the number of such sequences is exactly

$$\binom{t+\lfloor\sqrt{r}\lg n\rfloor}{t-1}.$$

Since there is a bijection between such sequences and elements of P', we indeed see that $|P'| = \binom{t+\lfloor\sqrt{r}\lg n\rfloor}{t-1}$. We will now show that every polynomial in P' reduces to a unique polynomial in F and thus to a unique element of $\mathcal{G}$.

Assume for contradiction that $f(X), g(X) \in P'$ reduce to the same polynomial in F, so that

$$f(X) \equiv g(X) \mod p, h(X).$$

We will now once again use the factor theorem to arrive at a contradiction. Let $m \in I$. Since m is introspective for every polynomial in P, it certainly is for $f(X)$ and $g(X)$. Since $h(X)$ is a factor of $X^r - 1$, we see that

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

This implies that X^m is a root of the polynomial $Q'(Y) := f(Y) - g(Y)$ over F for any $m \in I$. In particular, we see that X^m is a root of $Q'(Y)$ in F for any $m \in G$, viewing m as an integer representative of the equivalence class modulo r. This is well-defined, since $X^r = 1$ in F, so $X^{m+kr} = X^m$ in F. We claim all these roots are distinct.

Suppose for contradiction that $X^m = X^{m'}$ in F for $m, m' \in G$ with $m > m'$, where we represent m and m' by their unique representative in $\{0, 1, \ldots, r-1\}$. Then $h(X)$ divides $X^m - X^{m'} = X^{m'}(X^{m-m'} - 1)$ in $\mathbb{Z}_p[X]$. It is clear that $h(X)$ does not divide $X^{m'}$ in $\mathbb{Z}_p[X]$. Since $h(X)$ is irreducible in $\mathbb{Z}_p[X]$, this implies that $h(X)$ must divide $X^{m-m'} - 1$ in $\mathbb{Z}_p[X]$. Now our choice of $h(X)$ turns out to be useful, since by definition $\Phi_r(X)$ does not divide $X^{m-m'} - 1$ in $\mathbb{Z}_p[X]$ $(0 < m - m' < r)$. Since $h(X)$ is an irreducible factor of $\Phi_r(X)$ reduced to $\mathbb{Z}_p[X]$, with some small technicalities, we can show that $h(X)$ cannot divide $X^{m-m'}-1$ in $\mathbb{Z}_p[X]$ [14].

Therefore, $Q(Y)$ has at least $|G| = t$ distinct roots. Note that this is where choosing a different irreducible factor of $X^r - 1$ in $\mathbb{Z}_p[X]$, such as $X - 1$, fails. In that case, the polynomials X^m do not need to be distinct in F for $m \in G$. In the case of $X - 1$, these polynomials are all identical.

On the other hand, since $f(X)$ and $g(X)$ have degree less than t and have different coefficients, $Q(Y)$ is not the zero-polynomial and of degree less than t. The fact that $Q(Y)$ has at least t roots is thus in direct contradiction with the factor theorem. This implies that every polynomial in P' reduces to a unique polynomial in F and thus to a unique element of $\mathcal{G}$ and so we see that

$$|\mathcal{G}| \geq |P'| = \binom{t + \lfloor\sqrt{r}\lg n\rfloor}{t-1}.$$

□

We will now show that the upper and lower bounds for $|\mathcal{G}|$ are contradictory. Note that $G \subseteq \mathbb{Z}_r^*$, since $(n, r) = (p, r) = 1$. This implies that $t = |G| < r$. Furthermore, since G is generated by n and p modulo r with $o_r(n) > \lg^2 n$, it follows that $t = |G| > \lg^2 n$. This implies that $\sqrt{t}\lg n < t$. It now follows from Lemma 3.18 that

$$|\mathcal{G}| \geq \binom{t + \lfloor\sqrt{r}\lg n\rfloor}{t-1} > \binom{t + \lfloor\sqrt{t}\lg n\rfloor}{t-1} = \binom{t + \lfloor\sqrt{t}\lg n\rfloor}{\lfloor\sqrt{t}\lg n\rfloor + 1} \geq \binom{2\lfloor\sqrt{t}\lg n\rfloor + 1}{\lfloor\sqrt{t}\lg n\rfloor + 1}.$$

Here we use the fact that $\binom{a}{b} = \binom{a}{a-b}$. The inequalities can intuitively be understood by noting that reducing the total number of choices reduces the number of possible combinations. For simplicity, we now write $\ell = \lfloor\sqrt{t}\lg n\rfloor$. Then we see that

$$|\mathcal{G}| > \binom{2\ell+1}{\ell} = \frac{(2\ell+1)!}{\ell!(\ell+1)!} = \frac{(\ell+2)\cdots(2\ell+1)}{1\cdots\ell} = (\ell+2)\prod_{k=2}^{\ell}\frac{\ell+k+1}{k} \geq (\ell+2)\cdot 2^{\ell-1}.$$

Since n is composite, we know that $n \geq 4$, so that $\lg n \geq 2$ and thus $\ell = \lfloor\sqrt{t}\lg n\rfloor \geq 2$. From this, it follows that

$$|\mathcal{G}| > (\ell+2)\cdot 2^{\ell-1} \geq 2^{\ell+1} = 2^{\lfloor\sqrt{t}\lg n\rfloor+1} > 2^{\sqrt{t}\lg n} = n^{\sqrt{t}}.$$

However, this is clearly in contradiction with Lemma 3.17. This completes the proof of Theorem 3.12. □

Having proved Theorem 3.12, we can almost introduce the AKS primality test. To complete the algorithm, we need to prove that there is a suitably small r with $o_r(n) > \lg^2 n$. To show this, we first prove the following theorem.

Theorem 3.19. *Let* $\mathrm{LCM}(m) = \mathrm{lcm}\{1, 2, \ldots, m\}$ *denote the least common multiple of the first* m *positive integers. For* $m \geq 7$ *we have*

$$\mathrm{LCM}(m) \geq 2^m.$$

We present our own proof of this theorem, but first we prove two additional lemmata.

Lemma 3.20. *Let* $m > 1$ *be an integer and let* $p \leq m$ *be any prime number. Define* $f_p(m)$ *to be the exponent of the largest power of* p *that is less than* m*. Then the number of factors of* p *in* $m!$ *is given by*

$$\sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor.$$

Proof. Define $v_p(a)$ as the greatest power of p that divides a. Then the number of factors of p in $m!$ is given by

$$\sum_{a=1}^{m} v_p(a).$$

We will now count the number of integers $1 \leq a \leq m$ for which $v_p(a) = k$, for each $0 \leq k \leq f_p(m)$. Note that in this way each integer $1 \leq a \leq m$ will be counted exactly once, since there cannot be an integer $1 \leq a \leq m$

with $v_p(a) \geq f_p(m)+1$. Would this be the case, then it would imply that $p^{f_p(m)+1} \leq a \leq m$, in contradiction with the maximality of $f_p(m)$. Let $0 \leq k \leq f_p(m)$ be fixed. Then $v_p(a) = k$ if $p^k \mid a$, but $p^{k+1} \nmid a$. There are a total of $\left\lfloor \frac{m}{p^k} \right\rfloor$ possible $1 \leq a \leq m$ that are multiples of p^k, of which $\left\lfloor \frac{m}{p^{k+1}} \right\rfloor$ are also multiples of p^{k+1}. Therefore, the total number of integers $1 \leq a \leq m$ with $v_p(a) = k$ is

$$\left\lfloor \frac{m}{p^k} \right\rfloor - \left\lfloor \frac{m}{p^{k+1}} \right\rfloor.$$

From this, we can now compute the number of factors p in $m!$ as

$$\sum_{a=1}^{m} v_p(a) = \sum_{k=0}^{k=f_p(m)} k\left(\left\lfloor \frac{m}{p^k} \right\rfloor - \left\lfloor \frac{m}{p^{k+1}} \right\rfloor\right) = 0 - 0 + \left\lfloor \frac{m}{p} \right\rfloor - \left\lfloor \frac{m}{p^2} \right\rfloor + 2\left\lfloor \frac{m}{p^2} \right\rfloor - 2\left\lfloor \frac{m}{p^3} \right\rfloor + \ldots = \sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor,$$

where the latter follows by recognizing the telescoping sum and the fact that $\left\lfloor \frac{m}{p^{f_p(m)+1}} \right\rfloor = 0$. This proves Lemma 3.20. □

Lemma 3.21. *For any integer $m > 1$ we have*

$$\mathrm{LCM}(m) \geq m\binom{m-1}{k}$$

for any integer $0 \leq k \leq m-1$.

Proof. Since $m\binom{m-1}{k} = \frac{m!}{k!(m-k-1)!}$, this expression clearly has no prime factors greater than m. Let $p \leq m$ be any prime number and let $f_p(m)$ be as in the statement of Lemma 3.20. Then, by Lemma 3.20 the number of factors of p in $m!$ is $\sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor$. In a similar way, we can also count the number of factors p in $k!$ and $(m-k-1)!$ to derive that the number of factors of p in $\frac{m!}{k!(m-k-1)!}$ is equal to

$$\begin{aligned}\sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor - \sum_{i=1}^{f_p(k)} \left\lfloor \frac{k}{p^i} \right\rfloor - \sum_{i=1}^{f_p(m-k-1)} \left\lfloor \frac{m-k-1}{p^i} \right\rfloor &= \sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor - \sum_{i=1}^{f_p(m)} \left\lfloor \frac{k}{p^i} \right\rfloor - \sum_{i=1}^{f_p(m)} \left\lfloor \frac{m-k-1}{p^i} \right\rfloor \\ &= \sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{k}{p^i} \right\rfloor - \left\lfloor \frac{m-k-1}{p^i} \right\rfloor.\end{aligned}$$

Here we used the fact that $\left\lfloor \frac{k}{p^i} \right\rfloor = 0$ if $i > f_p(k)$. We now claim that $\left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{k}{p^i} \right\rfloor - \left\lfloor \frac{m-k-1}{p^i} \right\rfloor \leq 1$ for any $1 \leq i \leq f_p(m)$. Suppose for contradiction that $\left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{k}{p^i} \right\rfloor - \left\lfloor \frac{m-k-1}{p^i} \right\rfloor \geq 2$. Then

$$\frac{m}{p^i} \geq \left\lfloor \frac{m}{p^i} \right\rfloor \geq \left\lfloor \frac{k}{p^i} \right\rfloor + \left\lfloor \frac{m-k-1}{p^i} \right\rfloor + 2 \geq \left(\frac{k+1}{p^i} - 1\right) + \left(\frac{m-k}{p^i} - 1\right) - 2 = \frac{m+1}{p^i},$$

a clear contradiction. Here we used the fact that $\left\lfloor \frac{a}{b} \right\rfloor \geq \frac{a+1}{b} - 1$ for any positive integers a, b. Since $\left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{k}{p^i} \right\rfloor - \left\lfloor \frac{m-k-1}{p^i} \right\rfloor \leq 1$ for any $1 \leq i \leq f_p(m)$, it follows that the number of factors of p in $m\binom{m-1}{k} = \frac{m!}{k!(m-k-1)!}$ is bounded by

$$\sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor - \sum_{i=1}^{f_p(k)} \left\lfloor \frac{k}{p^i} \right\rfloor - \sum_{i=1}^{f_p(m-k-1)} \left\lfloor \frac{m-k-1}{p^i} \right\rfloor = \sum_{i=1}^{f_p(m)} \left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{k}{p^i} \right\rfloor - \left\lfloor \frac{m-k-1}{p^i} \right\rfloor \leq f_p(m).$$

Since this holds for every prime $p \leq m$ and $m\binom{m-1}{k}$ does not have any prime factors larger than m, we see that

$$m\binom{m-1}{k} \leq \prod_{p \leq m, p \text{ prime}} p^{f_p(m)}.$$

On the other hand, for every prime $p \leq m$, we have $p^{f_p(m)} \leq m$ by definition, so $p^{f_p(m)} \mid \mathrm{LCM}(m)$. This implies that

$$\mathrm{LCM}(m) \geq \prod_{p \leq m, p \ prime} p^{f_p(m)} \geq m\binom{m-1}{k}.$$

□

Using the two lemmata above, we can now prove Theorem 3.19.

Proof (Theorem 3.19). We prove that

$$m\binom{m-1}{\lfloor\frac{m-1}{2}\rfloor} \geq 2^m$$

for any $m \geq 7$. The result then immediately follows from Lemma 3.21. We present a proof by induction.

First, we will prove the claim for odd $m \geq 7$. For $m = 7$, we see that the claim indeed holds. Now suppose that for some odd $m = k = 2\ell + 1$ the claim holds. Then we see that

$$(k+2)\binom{k+1}{\lfloor\frac{k+1}{2}\rfloor} = (2\ell+3)\binom{2\ell+2}{\ell+1} = (2\ell+3)\frac{(2\ell+1)(2\ell+2)}{(\ell+1)^2}\binom{2\ell}{\ell} > 4(2\ell+1)\binom{2\ell}{\ell} = 4k\binom{k-1}{\lfloor\frac{k-1}{2}\rfloor} \geq 2^{k+2},$$

and so the claim also holds for $m = k + 2$, completing the induction.

We will now prove the claim for even $m \geq 7$. For $m = 8$, it can again be checked that the claim indeed holds. Now suppose that for some even $m = k = 2\ell$ the claim holds. Then we see that

$$(k+2)\binom{k+1}{\lfloor\frac{k+1}{2}\rfloor} = (2\ell+2)\binom{2\ell+1}{\ell} = (2\ell+2)\frac{2\ell(2\ell+1)}{\ell^2}\binom{2\ell-1}{\ell-1} > 4\cdot 2\ell\binom{2\ell-1}{\ell-1} = 4k\binom{k-1}{\lfloor\frac{k-1}{2}\rfloor} \geq 2^{k+2},$$

proving the claim also holds for $m = k + 2$, which completes the induction. □

Remark 3.22. Theorem 3.19 allows us to derive our own crude lower bound of the prime counting function $\pi(m)$, which is defined as the number of primes $p \leq m$. Let $m \geq 7$ be an integer and consider the product $\prod_{p \leq m, p \ prime} p^{f_p(m)}$. Any $1 \leq i \leq m$ divides this product, since $i \leq m$ has at most $f_p(m)$ factors of a prime $p \leq m$. It follows that

$$\mathrm{LCM}(m) \leq \prod_{p \leq m, p \ prime} p^{f_p(m)}.$$

By Theorem 3.19 we know that $\mathrm{LCM}(m) \geq 2^m$, since $m \geq 7$, and by definition of $f_p(m)$, we also know that $p^{f_p(m)} \leq m$ for every prime $p \leq m$. This implies that

$$2^m \leq \mathrm{LCM}(m) \leq \prod_{p \leq m, p \ prime} p^{f_p(m)} \leq \prod_{p \leq m, p \ prime} m = m^{\pi(m)} = 2^{\lg m \pi(m)}.$$

We therefore conclude that $\pi(m) \geq \frac{m}{\lg m}$ for any $m \geq 7$. This bound differs only by a constant factor $\frac{\ln m}{\lg m} = \ln(2)$ from the asymptotically optimal bound implied by the prime number theorem [14].

The consequence of Theorem 3.19 which we are most interested in is the fact that we can prove the existence of a suitably small r with $o_r(n) > \lg^2 n$. We present an altered version of the proof given in [11], which simplifies the proof, under the condition that we also accept r with $(n, r) > 1$ besides r with $o_r(n) > \lg^2 n$.

Theorem 3.23. *Let $n > 1$ be an odd integer. There exists an integer $1 \leq r \leq \lceil\lg^5 n\rceil$ such that either $(n, r) > 1$ or $o_r(n) > \lg^2 n$.*

Proof. Suppose for contradiction that $(n, r) = 1$ and $o_r(n) \leq \lg^2 n$ for any $1 \leq r \leq \lceil\lg^5 n\rceil$. This implies that $r \mid n^{i_r} - 1$ for some $1 \leq i_r \leq \lg^2 n$ for any $1 \leq r \leq \lceil\lg^5 n\rceil$ and therefore the product

$$\prod_{i=1}^{\lfloor\lg^2 n\rfloor} (n^i - 1)$$

is divisible by any integer $1 \leq r \leq \lceil \lg^5 n \rceil$ and from this it follows that it is also divisible by $\mathrm{LCM}\left(\lceil \lg^5 n \rceil\right)$. Since the product is clearly positive, this implies that

$$\prod_{i=1}^{\lfloor \lg^2 n \rfloor} (n^i - 1) \geq \mathrm{LCM}\left(\lceil \lg^5 n \rceil\right) \geq 2^{\lceil \lg^5 n \rceil} \geq 2^{\lg^5 n} = n^{\lg^4 n},$$

where the second inequality follows from Theorem 3.19, since $\lceil \lg^5 n \rceil \geq 7$ because $n \geq 3$. On the other hand, since $\lg^2 n \geq 1$, we have

$$\prod_{i=1}^{\lfloor \lg^2 n \rfloor} (n^i - 1) < \prod_{i=1}^{\lfloor \lg^2 n \rfloor} n^i = n^{\sum_{i=1}^{\lfloor \lg^2 n \rfloor} i} = n^{\frac{1}{2}\lfloor \lg^2 n \rfloor \left(\lfloor \lg^2 n \rfloor + 1\right)} \leq n^{\frac{1}{2}\lg^2 n \left(\lg^2 n + 1\right)} \leq n^{\lg^4 n},$$

a clear contradiction. This completes the proof. □

Combining all these ideas, we can finally describe the final algorithm presented in this thesis: the AKS primality test [11] [15].

Algorithm 3.24 (Agrawal–Kayal–Saxena)**.** Let $n \geq 2$ be an odd integer, whose primality we want to test. The following algorithm will determine the primality of n.

(i) If $n = a^b$ for some integers a, b with $b > 1$, n is composite.

(ii) Loop through all $1 \leq r \leq \lceil \lg^5 n \rceil$. If $r = n$, n is prime (this small technicality can only occur for small values of n for which $\lceil \lg^5 n \rceil \geq n$). Otherwise, test if $(n, r) > 1$. If so, n is composite. If not, test if $o_r(n) > \lg^2 n$ and if this is the case, store r and continue with step (iii).

(iii) For each integer $1 \leq a \leq \lfloor \sqrt{r} \lg n \rfloor$ test if

$$(X + a)^n \equiv X^n + a \mod n, X^r - 1.$$

If the above equation does not hold, n is composite. If the equation holds for every $1 \leq a \leq \lfloor \sqrt{r} \lg n \rfloor$, n is prime.

Proof. Note that there will always be an $1 \leq r \leq \lceil \lg^5 n \rceil$ with either $(n, r) > 1$ or $o_r(n) > \lg^2 n$ by Theorem 3.23, which means that the algorithm will always determine the primality of n in these three steps. We will now prove that the algorithm always returns the correct result.

Suppose n is prime, then step (i) will never determine that n is composite. Since n is prime and r will never exceed n in step (ii) (because we determine that n is prime as soon as $r = n$), every tested r will be coprime to n (except if $r = n$, but then (ii) determines that n is prime). Therefore step (ii) will also never determine that n is composite. Finally, as a direct corollary of Theorem 3.8, n will satify the polynomial identity

$$(X + a)^n \equiv X^n + a \mod n, X^r - 1$$

for any integer a. Therefore, step (iii) will correctly determine that n is prime, if this was not already decided in step (ii).

Now suppose that n is composite. The algorithm can only wrongly determine that n is prime in steps (ii) and (iii). However, if step (ii) determines that n is prime, then this must be since $n = r$ and so any previous $1 \leq r < n$ would have to have satisfied $(n, r) = 1$, which is impossible since n is composite. On the other hand, as a direct corollary of Theorem 3.12, step (iii) will only determine that n is prime if n is prime, or a power of a prime. Since the latter would have been determined as composite in step (iii), we see that step (iii) cannot determine n as prime if it is composite and passed step (i). We see that composite n will also always correctly be identified as such by the algorithm. This completes the proof that the algorithm will always return the correct result. □

4 Complexity Analysis

In the previous chapters, we have seen several primality testing algorithms. However, apart from the original motivation for looking into different algorithms than Trial Division, we have not really investigated the difference in speed between the different algorithms. This is mainly because we did not have the required tools. This chapter aims to provide the reader with such tools, serving as a basic introduction to complexity analysis. We will quantify the speed of the presented algorithms and make a comparison between the algorithms. However, we will see that this analysis needs to be backed by a practical analysis of the algorithms, this will be done in the next chapter.

4.1 Basic Definitions

In order to compare the speed of two different algorithms, we will count the number of *bit operations* both perform. We can rigorously define a bit operation as the number of steps of a Turing or register machine, or the number of gates of a Boolean circuit implementing the algorithm [16]. However, these definitions are quite technical; for this section it suffices to informally see bit operations as elementary operations on single bits, such as adding or subtracting two bits. From bit operations, we can define a measure for the complexity of an algorithm, by simply counting the required number of bit operations.

Definition 4.1. We call an algorithm with input n an $O(f(n))$ algorithm if there exists a $C \geq 0$ such that the algorithm applied to any n requires at most $Cf(n)$ bit operations. We say that the algorithm is of (time) complexity $O(f(n))$.

Remark 4.2. We will often denote the complexity of an algorithm in terms of the number of bits of the input. We can do this since the number of bits of n is simply another function of n.

Using the above definition of time complexity, we can finally rigorously define what we mean with an efficient algorithm: an algorithm which is polynomial.

Definition 4.3. An algorithm is called *polynomial* if it has polynomial complexity in the number of bits of the input. That is, if the algorithm has time complexity $O(\lg^k n)$ for some $k \geq 0$.

4.2 Elementary Operations

In order to derive the time complexity of the presented algorithms, we first need to derive the number of bit operations required for the elementary operations such as addition and multiplication.

Theorem 4.4 (Addition)**.** *The addition of two ℓ-bit integers can be performed with an $O(\ell)$ algorithm.*

Proof. We consider the basic algorithm of adding two ℓ-bit numbers by bit-wise addition from right to left, using a carry bit. For every bit of the input integers, this algorithm performs the bit-wise addition of two bits of the input numbers, and a single carry bit from the previous addition. Since this operation can be performed in a constant number of bit-wise operations and the algorithm performs ℓ such operations in total, the total complexity of the proposed algorithm is $O(\ell)$. □

Remark 4.5. To ensure correctness of the result and not lose any information, each of the ℓ bits in the input must be used in a bit operation at least once by the algorithm. Therefore, the complexity of $O(\ell)$ for addition is asymptotically optimal.

Theorem 4.6 (Subtraction)**.** *The subtraction of two ℓ-bit integers can be performed with an $O(\ell)$ algorithm.*

Proof. Simply calculate the inverse of one of the inputs (two's complement in the case of a fixed number of bits), which can be done in $O(\ell)$ bit operations and add this to the other input. By the previous theorem, this addition can also be performed in $O(\ell)$ bit operations, so the total complexity is $O(\ell)$. □

Remark 4.7. Again, since each of the ℓ input bits must be used in a bit operation at least once, the complexity of $O(\ell)$ for subtraction is asymptotically optimal.

Theorem 4.8 (Multiplication)**.** *The multiplication of two ℓ-bit integers can be performed with an $O(\ell^2)$ algorithm.*

Proof. We consider another well-known algorithm. To multiply two ℓ-bit integers a and b, we traverse the bits of a from right to left, keeping count of the index of the current bit (where the rightmost bit has index 0). If the current bit of a is a 0, we continue on to the next bit, but if the current bit of a is a 1, then we bit-shift b to the left by the same amount of bits as the index of the current bit of a and add this to a list. After processing all bits of a in this way, we add all the numbers in the created list together, one by one. Since we add at most ℓ numbers, with at most 2ℓ bits, the total time complexity becomes $O(\ell^2)$, as desired. □

Remark 4.9. The complexity of $O(\ell^2)$ for multiplication is not asymptotically optimal. In recent history, many algorithms for multiplication with better complexity have been derived. In 2019, an algorithm with a complexity of $O(\ell \ln \ell)$ was presented, which is predicted to be asymptotically optimal, although this is still an open problem [17].

In the following, let $M(\ell)$ denote the complexity of any multiplication algorithm of two ℓ-bit integers.

Theorem 4.10 (Division)**.** *The division of two ℓ-bit integers can be performed with an $O(M(\ell))$ algorithm.*

Proof. This can be achieved through Newton iteration. See [16] for an algorithm and proof. □

Theorem 4.11 (Modular exponentiation)**.** *Let x, y be two ℓ-bit integers and let e be a k-bit integer. The value of $x^e \bmod y$ can be computed with an $O(kM(\ell))$ algorithm.*

Proof. This can be achieved through repeated squaring. See [16] for an algorithm and proof. □

4.3 The Presented Algorithms

Using the tools of complexity analysis, we can now determine and compare the time complexity of the different algorithms presented. First, we see that Trial Division is indeed non-polynomial.

Theorem 4.12. *The Trial Division primality test can test an ℓ-bit integer n for primality with complexity $O(2^{\ell/2}M(\ell))$.*

Proof. For each integer $1 < a \leq \sqrt{n}$, the Trial Division algorithm checks if $a|n$. This check can be computed in $O(M(\ell))$ by Theorem 4.10, since any such a will have at most as many bits as n. Therefore, the total running time will be no more than

$$O\left(\sqrt{n}M(\ell)\right) = O\left(\sqrt{2^{\lg n}}M(\ell)\right) = O\left(\sqrt{2^{\ell}}M(\ell)\right) = O\left(2^{\ell/2}M(\ell)\right).$$

□

Remark 4.13. The Trial Division algorithm is indeed non-polynomial, since for a prime number p, Trial Division tests $\sqrt{p}$ integers for divisibility against p. Every test takes at least a single bit operation, so that the total number of bit operations is at least $\sqrt{p}$, which is exponential in the number of bits of p.

We now proceed to show that all the other presented algorithms are polynomial. We first derive the time complexities for the two probabilistic algorithms.

Theorem 4.14. *The Solovay-Strassen primality test with k iterations can test an ℓ-bit integer n for primality with complexity $O(\ell k M(\ell))$.*

Proof. For each of the k iterations, a random integer $0 < a < n$ is chosen. Since $a < n$, this integer will have at most ℓ bits. The Jacobi symbol $(\frac{a}{n})$ can be calculated in $O(\ell M(\ell))$[1], implying that test (i) of an iteration can be computed in $O(\ell M(\ell))$ bit operations. Theorem 4.11 shows us that $a^{(n-1)/2} \bmod n$ can also be computed with an $O(\ell M(\ell))$ algorithm, so that test (ii) of an iteration can be computed in $O(\ell M(\ell))$ bit operations, having already calculated $(\frac{a}{n})$. We see that both tests have complexity $O(\ell M(\ell))$. Therefore the total running time of a single iteration is $O(\ell M(\ell))$ and the complexity with k iterations becomes $O(\ell k M(\ell))$. □

Theorem 4.15. *The Miller-Rabin primality test with k iterations can test an ℓ-bit integer n for primality with complexity $O(\ell k M(\ell))$.*

Proof. Again, for each of the k iterations, a random integer $0 < a < n$ is chosen. Since $a < n$, this integer will have at most ℓ bits. Since $n - 1 = 2^s t$ with $t, s \geq 1$, we see that $t \leq \frac{n-1}{2}$ and $s \leq \lg n$. In particular it follows that t has at most ℓ bits, so that $a^t \bmod n$ can be computed with an $O(\ell M(\ell))$ algorithm by Theorem 4.11. This implies that test (i) of an iteration can be computed in $O(\ell M(\ell))$ bit operations. The remainder of an iteration consists of squaring $a^t \bmod n$ at most $s-1$ times. Every squaring operation has complexity $O(M(\ell))$ by Theorem 4.11, so that the remainder of an iteration has complexity $O((s-1)M(\ell)) = O(\lg n M(\ell)) = O(\ell M(\ell))$. Since we compute k iterations, the total complexity becomes $O(\ell k M(\ell))$. □

The time complexities of the deterministic variants of the Solovay-Strassen and Miller-Rabin algorithms now follow quite easily from the theorems above.

Corollary 4.16. *The deterministic version of the Solovay-Strassen primality test, assuming the ERH, can test an ℓ-bit integer n for primality with complexity $O(\ell^3 M(\ell))$.*

Proof. In the proof of Theorem 4.14, we saw that a single Solovay-Strassen iteration can be computed in $O(\ell M(\ell))$ bit operations. Since we compute at most $2\ln^2(n) - 1 = O(\ell^2)$ iterations, the total complexity is $O(\ell^3 M(\ell))$. □

Corollary 4.17. *The Miller primality test, assuming the ERH, can test an ℓ-bit integer n for primality with complexity $O(\ell^3 M(\ell))$.*

Proof. In the proof of Theorem 4.15, we saw that a single Miller iteration can be computed in $O(\ell M(\ell))$ bit operations. Since we compute at most $2\ln^2(n) - 1 = O(\ell^2)$ iterations, the total complexity is $O(\ell^3 M(\ell))$. □

Finally, we can derive the time complexity of the AKS primality testing algorithm to see that the AKS algorithm is indeed polynomial, as claimed.

Theorem 4.18. *The AKS primality test can test an ℓ-bit integer n for primality with complexity $O(\ell^{9/2} M(\ell^6))$.*

Proof. We will again derive the time complexity of each step of the algorithm. First note that if $n = a^b$ for some $b > 1$, then $b \leq \lg n$, since $a \geq 2$. We now simply test if $\sqrt[b]{n}$ is an integer for every integer $1 < b \leq \lg n$. Using Newton iteration, this test can be performed in $O(M(\ell))$ for any b, so the total complexity of step (i) is $O(\ell M(\ell))$.

For step (ii), in each iteration, $r = n$ can be tested with $O(\ell)$ bit operations. Furthermore, we can calculate (n, r) in $O(\ell M(\ell))$ bit operations [16]. Testing if $o_r(n) > \lg^2 n$ can be done by simply calculating $n^k \mod r$ for every $1 \leq k \leq \lg^2 n$. This takes at most $\lg^2 n$ multiplications modulo r, for a total of $O(\lg^2 n \lg r) = O(\lg^2 n \lg\lg n) = O(\ell^2 \lg \ell)$ bit operations. This is the most complex step of the iteration (if we take $M(\ell) = O(\ell \lg \ell)$, so the entire iteration has complexity $O(\ell^2 \lg \ell)$. We do at most $\lg^5 n = \ell^5$ iterations, so step (ii) has time complexity $O(\ell^7 \lg \ell)$.
For step (iii), we verify at most $\sqrt{r} \lg n$ polynomial equations. Calculating $(X + a)^n \mod n, X^r - 1$ can be done in $O(\lg n) = O(\ell)$ polynomial multiplications by repeated squaring. Since we work modulo n and $X^r - 1$, the polynomials we multiply have degree less than r and coefficients with at most ℓ bits. Such a multiplication has time complexity $O(M(r(\ell + \lg r))) = O(M(\ell^6 + \ell^5 \lg \ell)) = O(M(\ell^6))$. Therefore, calculating $(X + a)^n \bmod n, X^r - 1$ has complexity $O(\ell M(\ell^6))$. Verifying the polynomial equation

$$(X + a)^n \equiv X^n + a \mod n, X^r - 1$$

can now be done with complexity $O(r\ell)$, so that the calculation of $(X+a)^n \mod n, X^r - 1$ is the most complex in every iteration. Since we do at most $\sqrt{r} \lg n$ in total, step (iii) has time complexity $O(\sqrt{r} \lg n \ell M(\ell^6)) = O(\ell^{9/2} M(\ell^6))$. This is the most complex step of the entire algorithm, from which the result follows. □

We see that the AKS primality testing algorithm has a much worse time complexity than the deterministic variant of the Solovay-Strassen algorithm and the Miller algorithm. If we assume the ERH to be true, these latter two algorithms would thus be much faster deterministic primality tests. However, the probabilistic algorithms have the least time complexity of all the presented algorithms, since we can choose k to be a small constant. We thus expect the AKS algorithm to be the slowest of the polynomial primality testing algorithms presented, and the Solovay-Strassen and Miller-Rabin algorithms to be the fastest.

Remark 4.19. Lenstra and Pomerance managed to improve the time complexity of the AKS algorithm to $O(\ell^6 \lg^k \ell)$ for some $k \geq 1$, which is still worse than the other polynomial algorithms presented, but a massive improvement over $O(\ell^{9/2}M(\ell^6))$. Their idea is to replace the cyclotomic polynomial $\Phi_r(X)$ by a polynomial $f(X)$ with some specific properties [14].

5 Practical Results

While complexity analysis provides an asymptotic upper bound for the running time of an algorithm, in practice the relative performance of two algorithms might differ from what we expect from complexity analysis. This can occur if the constants in the big-O notation differ significantly or if the algorithm almost always performs better than its worst case performance. It is therefore essential to perform an additional analysis to effectively compare the speed of different algorithms. In this section, we provide a practical analysis based on Python implementations of the different algorithms presented. The code for these implementations can be found in Appendix A.

5.1 Performance on Prime Numbers

Doing meaningful analysis on the presented primality testing algorithms is hard, since the total computing time always depends on the number of iterations before termination, which varies greatly between different integers. One way to overcome this problem is to only consider the computing time of the algorithms on prime numbers. This is because in all the presented algorithms, a prime number is guaranteed to always require the maximum amount of iterations required. We can thus interpret the required computing time of a primality testing algorithm on a prime number as a *worst-case* performance, which provides a meaningful way to compare different primality testing algorithms.

The result of this comparison on primes of different orders of magnitude is shown in Table 1. The computing time was measured in seconds and averaged over 10 tests. To prevent the slower algorithms from never terminating, an algorithm taking more than 60 seconds was terminated externally; these external terminations are denoted by N/A. The algorithms have been abbreviated, but are listed in the same order as they were introduced in the previous chapters. For the probabilistic tests, we opt for $k = 80$ for Solovay-Strassen and $k = 40$ for Miller-Rabin, so that both have a probability of at least $1 - \frac{1}{2^{80}}$ of determining a composite input as such.

n	TD	SS$_{80}$	MR$_{40}$	SS$_{ERH}$	M	AKS
$10^5 + 3$	0.0000	0.0003	0.0001	0.0008	0.0004	N/A
$10^{10} + 19$	0.0084	0.0009	0.0003	0.0086	0.0067	N/A
$10^{15} + 37$	2.7371	0.0015	0.0004	0.0303	0.0231	N/A
$10^{20} + 39$	N/A	0.0036	0.0008	0.0767	0.0695	N/A
$10^{50} + 151$	N/A	0.0111	0.0033	1.7961	1.8323	N/A
$10^{100} + 267$	N/A	0.0358	0.0111	27.0716	26.7191	N/A
$10^{200} + 357$	N/A	0.1424	0.0557	N/A	N/A	N/A
$10^{500} + 961$	N/A	1.2643	0.5892	N/A	N/A	N/A
$10^{1000} + 453$	N/A	8.6627	4.0552	N/A	N/A	N/A

Table 1: Computing time of different primality testing algorithms on primes.

Most of the results in Table 1 were already expected in the previous chapter. The first thing we see is that the two probabilistic algorithms greatly outperform the deterministic primality testing algorithms for large values of n. This can be attributed to the fact that these algorithms perform a fixed number of iterations, while the number of iterations for the deterministic algorithms depends on the size of n. Furthermore, the deterministic variants of Solovay-Strassen and the Miller primality test perform almost the same, implying that the Miller and Solovay-Strassen iterations take about the same amount of time, since both perform the same number of iterations ($2\ln^2 n$). This is further strengthened by the fact that the Solovay-Strassen algorithm consistently takes around twice as long as the Miller-Rabin algorithm, which can be explained by the fact that the former requires twice as many iterations to achieve the same level of confidence. Since every strong pseudoprime base is also an Euler pseudoprime base of n by Theorem 2.26, this implies that it is always favourable to use the Miller-Rabin algorithm over the Solovay-Strassen algorithm (at least when using the versions of the algorithms described in Appendix A).

It should be noted that our implementation of none of the algorithms is the asymptotically fastest possible one, but our implementation of the AKS algorithm is exceptionally sub-optimal. The step where we calculate $(X+a)^n \mod n, X^r - 1$, which we saw to be the most time consuming step in the previous chapter, can be sped up significantly, by using the method of Single- or Multipoint Evaluation instead of multiplying every pair of coefficients [14][16]. However, the fact remains that the AKS algorithm has far worse asymptotical behaviour than the other non-trivial algorithms presented. To give at least an idea of how our implementation of the AKS algorithm performs, Table 2 lists the computing time used by the AKS algorithm on several smaller primes. Again, the average of 10 tests was taken.

n	AKS
$10^0 + 3$	0.0000
$10^1 + 1$	0.0000
$10^2 + 1$	1.0887
$10^3 + 9$	14.0129
$10^4 + 7$	92.1654
$10^5 + 3$	440.7000

Table 2: Computing time of the AKS primality testing algorithms on primes.

All of the other primality testing algorithms (including Trial Division) can test the primes in Table 2 in less than a millisecond. It should be clear that while the AKS primality testing algorithm was a major theoretical breakthrough, it has no place in practical applications.

5.2 Performing Fewer Iterations

The computing time required by all the algorithms in the worst case is mainly dictated by the amount of iterations performed. Therefore, if we can decrease the number of required iterations of a certain algorithm, it will become more efficient.

By Theorem 3.1 and Corollary 3.5, we know there must be a strong (and Euler) pseudoprime non-base to a composite number n that is less than $2\ln^2 n$. However, Figures 1 and 2 suggest we can do better than this upper bound. These graphs respectively show the minimal Euler pseudoprime non-base and minimal strong pseudoprime non-base to a composite number n.

In Figure 1, we see a maximum of $a = 37$, attained at $n = 14469841$, which is still much lower than the upper bound of $2\ln^2 n = 543.68\ldots$. We do however see a quadratic trend, which would imply that there is no better upper bound than $C\ln^2 n$. Based on the limited data in Figure 1, we propose $C = 0.137$, which yields an upper bound for all composite $n \leq 10^8$.

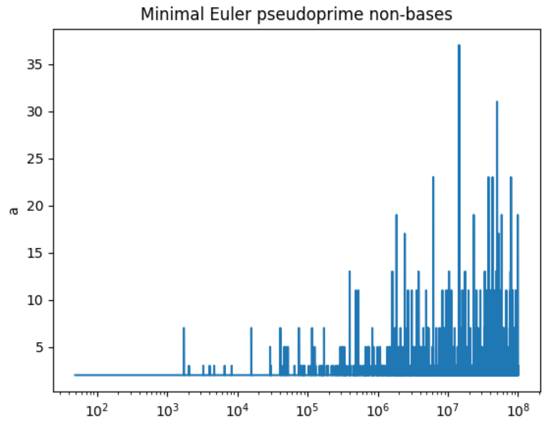

Figure 1: Minimal Euler pseudoprime non-bases.

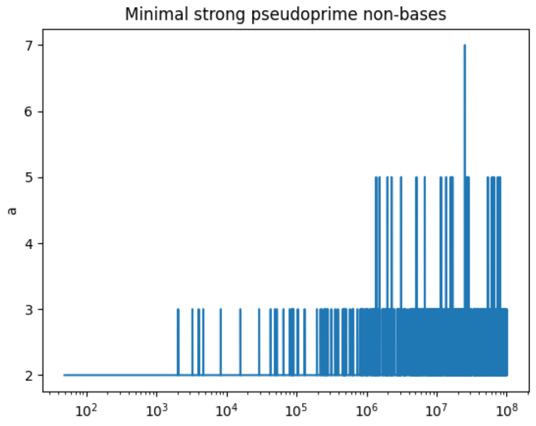

Figure 2: Minimal strong pseudoprime non-bases.

In Figure 2, we see a maximum of $a = 7$, attained at $n = 25326001$. Again, this is much lower than the upper bound of $2\ln^2 n = 581.22\ldots$. Here, we do not see a clear trend, but under the assumption that there is a quadratic trend and there is no better upper bound than $C\ln^2 n$, we propose $C = 0.026$, which yields an upper bound for all composite $n \leq 10^8$. However, it could well be that a bound of $C\ln^{2-\epsilon} n$ is possible.

It is also remarkable that both the minimal Euler pseudoprime non-bases and the minimal strong pseudoprime non-bases seem to always be prime numbers. The fact that the minimal Euler pseudoprime non-base is prime follows immediately from Theorem 3.2, which in particular states that the set of all Euler pseudoprime non-bases to n is a group. Therefore, if a composite number $k = ab$ is an Euler pseudoprime non-base to n, at least one of a and b must also be an Euler pseudoprime non-base to n and so, a composite number can never be a minimal Euler pseudoprime non-base. The fact that the minimal strong pseudoprime non-bases also seem to always be prime numbers is more difficult to prove and we leave this as a point of future research.

Unfortunately, due to the speed of the implementation, no meaningful graphs as in Figures 1 and 2 could be produced for the AKS primality test in a reasonable timespan. However, we can do an analysis on the required iterations in the probabilistic Solovay-Strassen and Miller-Rabin primality tests.

In the previous section, we used 80 and 40 iterations respectively for the Solovay-Strassen and Miller-Rabin algorithm. However, it turns out that in most cases a single iteration will suffice. This is shown in Figures 3 and 4. These stacked graphs indicate the percentage of tests which determined a composite number as such in a given number of tries. The graphs were created by taking 100 logarithmically spaced points x_i and for each of these points x_k, counting the number of iterations required when testing 10000 surrounding points. These surrounding points were chosen from the interval $[x_k, x_{k+1}]$.
Clearly, for large n, performing a single iteration is almost always sufficient. Do note that we take the 10000 surrounding points at random, so that an n which on average needs more tests is almost invisible in the graph. However, the graphs do show us that for a random large n, there is a big chance that a single iteration will suffice. This is partly clear from Remark 2.30; for a lot of composite numbers n, the fraction of possible bases that is a strong pseudoprime base to n is much less than $\frac{1}{4}$. One other thing that is very interesting to note is that the graphs in Figures 3 and 4 are almost identical, while the theoretical confidence of the Solovay-Strassen primality test is much lower than that of the Miller-Rabin primality test. This might also be an interesting topic for future research.

In any case, Miller-Rabin seems to be the preferred algorithm to use in practice, since it requires only a fixed number of iterations, but still always has the same confidence, regardless of the size of n. It is not surprising that this is indeed the case: the Miller-Rabin algorithm is one of the most widely used primality testing algorithms in practice [18]. Furthermore, we saw that Miller-Rabin should always be used over Solovay-Strassen when using implementations similar to those in Appendix A.

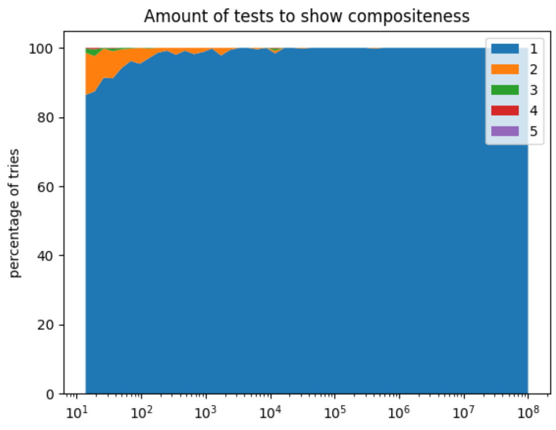

Figure 3: Solovay-Strassen.

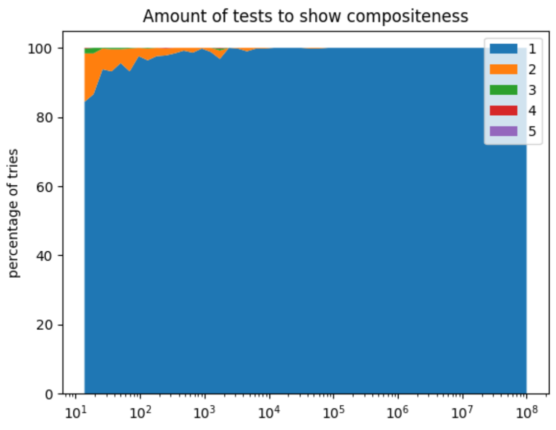

Figure 4: Miller-Rabin.

6 Final Remarks

We have seen several primality testing algorithms, but there are many other algorithms which were not discussed in this thesis. These algorithms include very efficient algorithms which only work on integers of certain special types, such as the Lucas-Lehmer primality test. This primality test only works on numbers of the form $2^\ell - 1$, but is deterministic and has a time complexity of $O(\ell M(\ell))$. This is much faster than all of the other deterministic algorithms presented in this thesis and in most cases even outperforms the probabilistic Miller-Rabin primality test. Other general primality testing algorithms include the widely used probabilistic Baillie-PSW algorithm and the deterministic elliptic curve primality test [15][14].

We saw that, in practice, the Miller-Rabin primality test has the best performance out of all the presented algorithms, both in theory and in practice. The fact that a constant number of iterations can guarantee a level of certainty, regardless of the size of the input, allows the probabilistic Solovay-Strassen and Miller-Rabin primality tests to achieve a much better asymptotic time complexity than their deterministic counterparts based on the ERH. It should therefore not be surprising that the Miller-Rabin primality tests is one of the most used primality testing algorithms in practice. While the AKS primality test is a very interesting theoretical algorithm, it performs much worse than the other polynomial algorithms and has no use in practical applications.

Based on the practical results in Chapter 5, we hypothesized on improving the upper bounds for a minimal Euler or strong pseudoprime base. It is not unlikely that there exist numbers greater than those computed which have a minimal Euler or strong pseudoprime base above the speculated upper bound. We do however think that the actual bounds will not be much greater than those presented. We also discovered the interesting result that in practice, we expect the probabilistic Solovay-Strassen and Miller-Rabin primality tests to almost always require only a single test to show compositeness. We also pointed out several ideas for potential future research, based on the practical results gathered.

Concluding, the main idea the reader should retain from this thesis is that almost any criterion which holds mainly for prime numbers can be turned into a primality test. The speed of the resulting test mostly depends on the speed with which this criterion can be computed, but a slow test can sometimes be made significantly faster by considering a weaker version of the criterion and turning the test into a probabilistic primality test.

A Algorithm Implementations in Python 3

A.1 Trial Division

```
def TrialDivision(n):
    for a in range(2, math.floor(math.sqrt(n)) + 1):
        if n % a == 0:
            return False
    return True
```

A.2 Solovay-Strassen

```
def SolovayStrassen(n, k):
    while (k > 0):
        a = random.randrange(1, n)
        if not SolovayStrassenIteration(a, n):
            return False
        k -= 1
    return True

def SolovayStrassenERH(n):
    for a in range(1, math.floor(2*math.log(n)**2)):
        if not SolovayStrassenIteration(a, n):
            return False
    return True

def SolovayStrassenIteration(a, n):
    j = Jacobi(a, n)
    if (j == 0):
        return False
    p = pow(a, (n-1)//2, n)
    if ((p - j) % n != 0):
        return False
    return True

# From http://rosettacode.org/wiki/Jacobi_symbol#Python
def Jacobi(a, n):
    a %= n
    result = 1
    while a != 0:
        while a % 2 == 0:
            a //= 2
            n_mod_8 = n % 8
            if n_mod_8 in (3, 5):
                result = -result
        a, n = n, a
        if a % 4 == 3 and n % 4 == 3:
            result = -result
        a %= n
    if n == 1:
        return result
    else:
        return 0
```

A.3 Miller and Miller-Rabin

```
def MillerRabin(n, k):
    (s, t) = DivideEvens(n-1)
    while (k > 0):
        a = random.randrange(1, n)
        if not MillerIteration(a, n, s, t):
            return False
        k = k - 1
    return True

def Miller(n):
    (s, t) = DivideEvens(n-1)
    for a in range(1, math.floor(2*math.log(n)**2)):
        if not MillerIteration(a, n, s, t):
            return False
    return True

def MillerIteration(a, n, s, t):
    p = pow(a, t, n)
    if (p == 1):
        return True
    for i in range(s):
        if ((p + 1) % n == 0):
            return True
        p = (p * p) % n
    return False

def DivideEvens(n):
    s = 0
    t = n
    while (t % 2 == 0):
        t //= 2
        s += 1
    return (s, t)
```

A.4 Agrawal-Kayal-Saxena (AKS)

```
def AKS(n):
    if PerfectPower(n):
        return False
    for r in range(2, math.floor(math.log2(n)**5) + 1):
        if r == n:
            return True
        if math.gcd(r, n) > 1:
            return False
        if HasBigOrder(n, r):
            break
    for a in range(1, math.floor(math.sqrt(r) * math.log2(n)) + 1):
        if not AKSPolynomial(a, n, r):
            return False
    return True
```

```
def PerfectPower(n):
    for k in range(2, math.floor(math.log2(n)) + 1):
        if IsKthPower(n, k):
            return True
    return False

def IsKthPower(n, k):
    r = n ** (1 / k)
    return n == int(math.floor(r)) ** k or n == int(math.ceil(r)) ** k

def HasBigOrder(n, r):
    e = n % r
    for k in range(math.floor(math.log2(n)**2) + 1):
        if e == 1:
            return False
        e = (e * n) % r
    return True

def AKSPolynomial(a, n, r):
    P = [0] * (r-2) + [1, a]
    LHS = PolynomialModPow(P, n, r, n)
    l = len(LHS)
    if LHS[l - 1] != a % n:
        return False
    for d in range(1, l):
        if d == n % r:
            if LHS[l - d - 1] != 1:
                return False
        else:
            if LHS[l - d - 1] != 0:
                return False
    return True

def PolynomialModPow(P, n, m, e):
    k = e
    B = P
    Q = [0] * (m-1) + [1]
    while k > 0:
        if k % 2 == 1:
            Q = PolynomialModMultiply(Q, B, n, m)
        B = PolynomialModMultiply(B, B, n, m)
        k //= 2
    return Q

def PolynomialModMultiply(P, Q, n, m):
    d1 = len(P)
    d2 = len(Q)
    res = [0] * m;
    for i in range(d1):
        for j in range(d2):
            res[(i + j - d1 - d2 + 1) % m] += P[i] * Q[j];
            res[(i + j - d1 - d2 + 1) % m] %= n
    return res;
```

References

[1] Neal Koblitz. *A Course in Number Theory and Cryptography*. 2nd ed. Springer-Verlag, 1994. ISBN: 9780387942933.

[2] Thomas L. Heath. *The Thirteen Books of Euclid's Elements*. 1st ed. Vol. 2. Cambridge University Press, 1908.

[3] Solomon W. Golomb. "Combinatorial Proof of Fermat's "Little" Theorem". In: *The American Mathematical Monthly* 63.10 (1956), p. 718.

[4] Robert D. Carmichael. "Note on a new number theory function". In: *Bulletin of the American Mathematical Society* 16.5 (1910), pp. 232–238.

[5] William R. Alford, Andrew Granville, and Carl Pomerance. "There are infinitely many Carmichael numbers". In: *Annals of Mathematics* 140.3 (1994), pp. 703–722.

[6] Lorenzo Di Biagio. "Euler Pseudoprimes for Half of the Bases". In: *Integers* 12.6 (2012), pp. 1231–1237. DOI: `10.1515/integers-2012-0037`.

[7] Michael O. Rabin. "Probabilistic Algorithm for Testing Primality". In: *Journal of Number Theory* 12.1 (1980), pp. 128–138.

[8] Gary L. Miller. "Riemann's Hypothesis and Tests for Primality". In: *Journal of Computer and System Sciences* 13.3 (1976), pp. 300–317.

[9] Eric Bach. "Explicit Bounds for Primality Testing and Related Problems". In: *Mathematics of Computation* 55.191 (1990), pp. 355–380.

[10] Nesmith C. Ankeny. "The Least Quadratic Non Residue". In: *Annals of Mathematics* 55.1 (1952), pp. 65–72.

[11] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of Mathematics* 160.2 (2004). First description and proof of the AKS primality test, pp. 781–793.

[12] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "Errata: PRIMES is in P". In: *Annals of Mathematics* 189.1 (2019), pp. 317–318.

[13] Terence Tao. *The AKS primality test*. Blog post by Terence Tao. 2009. URL: `https://terrytao.wordpress.com/2009/08/11/the-aks-primality-test/`.

[14] Andrew Granville. "It is easy to determine whether a given integer is prime". In: *Bulletin of the American Mathematical Society* 42.1 (2004), pp. 3–38.

[15] Richard Crandall and Carl Pomerance. *Prime Numbers: A Computational Perspective*. 2nd ed. Springer Science+Business Media, 2005.

[16] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge University Press, 2013.

[17] David Harvey and Joris van der Hoeven. "Integer multiplication in time O(n log n)". In: *Annals of Mathematics* 193.2 (2021), pp. 563–617.

[18] Keith Conrad. "The Miller-Rabin Test". In: *Encyclopedia of Cryptography and Security*. Springer Science+Business Media, 2011.