MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"A Primality Testing Journey: The Search for an Unconditional Deterministic Polynomial-Time Algorithm"

verfasst von / submitted by

Milena Damrau BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science

Vienna 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 821
Studienrichtung lt. Studienblatt / degree programme as is appears on the student record sheet:	Mathematik
Betreut von / Supervisor:	ao. Univ.-Prof. Mag. Dr. Leonhard Summerer

Acknowledgements

First of all, I would like to thank my supervisor Leonhard Summerer for his useful comments and remarks, and for suggesting this topic in the first place. Then I would also like to thank my family including Hernán, who are the best support I can think of. Finally, I want to thank Stephanie and Robert for giving me the freedom to take some days off of work to finish this thesis.

Abstract

In 2002, the three Indian computer scientists M. Agrawal, N. Kayal and N. Saxena presented the first unconditional deterministic polynomial-time primality test. The so called *AKS test*. Until then, it was not even known, if such an algorithm exists. In this thesis, we will take a journey through more than 2000 years of primality testing and discuss the most fundamental and most popular primality testing algorithms - from the Sieve of Eratosthenes over Fermat, Miller-Rabin, Lucas and Elliptic Curve tests to the celebrated AKS test and its improvements. At the end we will discuss which primality test to use for what purpose.

Contents

1. Introduction

People have been studying prime numbers since more than 2000 years. Ever since, primality testing was of big interest. Given any positive integer n, one would like to determine quickly if n is prime or not. The first primality test goes back to 300 BC: *the Sieve of Eratosthenes*. Eratosthenes found a method to make a list of all prime numbers up to fixed positive integer. This algorithm can be used to get a very easy primality test. We will discuss it in detail in chapter 3.

But *why* does one want to know if a number is prime in the first place? Even though in many mathematical fields prime numbers are important, they turn out to be essential in cryptography. The security of most public-key cryptosystems depends on large prime numbers. This is why primality testing has never been more important than nowadays.

One famous example for a public-key cryptosystem, whose security is based on the difficulty of factoring large numbers, is the famous *RSA cryptosystem*. It is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published it in 1977. We will shortly introduce it, to make clear, why large prime numbers play the key role for its security. You can find a more detailed description in [HPS08].

In cryptography the names *Alice* and *Bob* are commonly used for two people attempting to communicate with each other and *Eve* is the attacker. In the following, let (e, N) be the so called public encryption key and (d, N) the secret decryption key.

RSA Key Creation

1. Bob chooses two different prime numbers p, q and computes the public modulus
$$N := p \cdot q.$$
2. Then he computes Euler's totient function $\phi(N)$:
$$\phi(N) = (p-1) \cdot (q-1).$$
3. Bob chooses a public encryption exponent e coprime to $\phi(N)$ such that $1 < e < \phi(N)$.

RSA Enrcyption

1. Alice converts her plaintext into an integer m with $1 \leqslant m < N$.

2. Then she uses Bob's public key (e, N) to compute the ciphertext c:

$$c \equiv m^e \pmod{N}.$$

3. Alice then sends the ciphertext to Bob.

RSA Decryption

1. Bob computes the decryption exponent d as the modular multiplicative inverse of e modulo $\phi(N)$:

$$e \cdot d \equiv 1 \mod \phi(N).$$

2. He then takes the ciphertext c and computes the plaintext m:

$$m \equiv c^d \pmod{N}.$$

Note that Bob is only able to compute $\phi(N)$ and d since he knows p and q. In fact, he is the only one who knows these numbers. Eve on the other hand only knows the values of N, e and c, since they are public. One possibility for her to attack is to factorize N. If Bob chooses composite numbers instead of primes to compute N, Eve might be able to factor N. If p and q are small primes, factoring N would also be not that hard. So it is important that N is the product of two large prime numbers. So this is what Bob does. Therefore, Eve probably will not be successful with her attack. Note, that even if Eve knew c, she would not be able to compute d in general, since it is not known if the discrete logarithm problem is solvable in polynomial time (see for example [HPS08]).

So far we know, why large prime numbers are important in cryptographic applications. The next question which naturally arises is how does Bob know that the two numbers he chose are in fact prime?

This motivates us to find an algorithm which determines quickly whether an arbitrary number is prime or composite. As we will see, finding such an algorithm is not that easy.

As the title of this thesis reveals, our goal is to describe a *deterministic polynomial-time algorithm*. The problem is to have both: a deterministic test and a test which runs in polynomial-time. For a long time it was unknown, if such an algorithm even exists. But since 2002, we know, that the answer is yes. Before we introduce this so called *AKS Test* in chapter 7, we will give an overview of the results in primality testing, which have been accomplished before the AKS test had been published and which in a way made the finding of the algorithm possible in the first place.

We start in chapter 2 by introducing basic definitions and concepts, which will be important in the remainder of this thesis.

Then, in chapter 3 we will discuss some of the most fundamental primality tests: the *Sieve of Eratosthenes/Trial Division*, *Fermat's Test*, the *Miller-Rabin Test* and the *Lucas Test* with some variations.

In chapter 4 we will give a short introduction to *complexity theory* and analyze some of the primality tests introduced in chapter 3 regarding their running time.

Then, in chapter 5 we will introduce the so called *Lucas sequences*, which can be used for primality testing. Among the primality tests we introduce in this chapter is the *Lucas-Lehmer Test* with which most of the largest prime numbers were found.

In chapter 6 we introduce one of the most commonly used primality testing algorithms: the *Elliptic Curve Primality Test*.

After presenting the *AKS Test* and its improvements in chapter 7, we will compare the primality tests in chapter 8 to figure out, which one should be used depending on the situation.

2. Preliminaries

In this chapter we will focus on basic definitions and facts, which are related to primality testing.
More advanced readers might skip this chapter (or sections of it) and continue reading in chapter 3.

To avoid confusion we start by introducing the notations we will use.

- $\mathbb{Z}_n$ denotes the ring of integer numbers modulo n. $\mathbb{F}_q$ denotes the finite field with q elements. For a prime p, $\mathbb{Z}_p$ is a finite field, so we denote it by $\mathbb{F}_p$,
- $\gcd(n, m)$ and $\operatorname{lcm}(n, m)$ denote the greatest common divisor and the least common multiple of two integers n and m respectively. Two integers n, m are called *coprime* if $\gcd(n, m) = 1$,
- $\mathbb{Z}_n^*$ denotes the *group of units of* $\mathbb{Z}_n$. It consists of all elements a of $\mathbb{Z}_n$ with $\gcd(a, n) = 1$.

2.1. Prime numbers[1]

The concept of prime numbers is a mystery on its own. Whereas the basic definition of prime numbers was probably learned by most people in school, nobody seems to be able to have a complete picture of them.

A prime number p is a positive integer, which has exactly two positive divisors: one and itself. By this definition 2 is the smallest prime number (and the only one which is even). But until the 19th century, some people considered 1 as the first prime number (for example see [CRXK12] which gives is a great collection on the history of prime numbers, especially on the question "What is the smallest prime number?"). A number $n > 1$ which is not prime is called composite. 1 is considered neither prime nor composite.

The earliest surviving records about the study of prime numbers come from the ancient greeks. Around 300 BC, Euclid proved that *there are infinitely many prime numbers*. He used the fact that every integer greater than 1 is divisible by some prime number. The *fundamental theorem of arithmetics* was not explicitly stated by Euclid but follows directly from his results. The fundamental theorem of arithmetics states, that every positive integer has a unique prime factorization. This is by the way a reason, why it

[1]The main references for this section are [Bun08] and [CP05]

makes sense to exclude 1 as a prime number. The uniqueness would fail then, since any number can be written as $n = 1 \cdot n$. After Euclid, it got pretty quiet around the study of primes. No noteworthy statements about primes were made until the beginning of the 17th century. It was Fermat who then made some major discoveries about primes. His most significant discovery about primes is *Fermat's Little Theorem*. It states that if p is prime, then $a^p \equiv a \pmod p$ for every integer a. As we will see in chapter 3, this theorem plays an important role in the field of primality testing.

We will now give some essential results about the distribution of prime numbers. For that we first define the *prime counting function*:

Definition. The *prime-counting function* is defined by

$$\pi(x) = \#\{p \leqslant x \mid p \text{ prime}\}.$$

So $\pi(x)$ is the number of primes not exceeding x.
In the mid-19th century, the Russian mathematician Pafnuty Lvovich Chebyshev showed the following inequalities:

Theorem 2.1 (Chebyshev). *There are positive integers A, B such that for all $x \geqslant 3$,*

$$\frac{Ax}{\ln x} < \pi(x) < \frac{Bx}{\ln x}.$$

This theorem is for example true for $A = \frac{1}{2}$ and $B = 2$. Gauss had conjectured the asymptotic behavior of $\pi(x)$ in 1791, so Chebyshev's inequalities were a spectacular result. In 1896, Gauss conjecture was proved by J. Hadamard and C. de la Vallée-Poussin independently. It is now known as the *Prime Number Theorem*:

Theorem 2.2 (Prime Number Theorem). *As $x \to \infty$,*

$$\pi(x) \sim \frac{x}{\ln x}.$$

$f \sim g$ ("$f(x)$ is asymptotic to $g(x)$ as x tends to infinity"), means $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$.
See [CP05] or [Rib11] for the proof and more about the distribution of prime numbers.

We will now consider prime numbers of special kinds. These are interesting for themselves and also when it comes to primality testing as we will see. One example are *Fermat primes*.

Definition. An integer F_k is called *Fermat number* if it is of the form $F_k = 2^{2^k} + 1$. If a Fermat number is prime it is called a *Fermat prime*.

In 1637, Fermat claimed that every Fermat number is prime. But this is one of the few cases Fermat was wrong. The only known Fermat primes are those up to $F_4 = 65537$. Every other Fermat number for which we have been able to decide if it is prime or not is composite. The smallest Fermat number F_k for which its status is unknown is F_{33}. The largest Fermat number for which it is known is $F_{3329780}$ (as of October 2016). You

can find a list of some Fermat numbers and their factoring status in the appendix. A complete list of the current Fermat factoring status can be found at [Kel].
Apart from the fact that there aren't that many known Fermat primes, one gets some interesting results. If one is looking for primes of the form a power of two increased by one, then it is sufficient to look at Fermat numbers:

Theorem 2.3. *If* $p = 2^m + 1$ *is an odd prime, then* m *is a power of two.*

In chapter 3 we will introduce a primality test for Fermat numbers. There are more interesting facts about Fermat numbers. [CP05] is a good reference if you want to learn more about them.

Mersenne prime numbers are another example of particular prime numbers.

Definition. An integer M_p is called *Mersenne number* if it is of the form $M_p = 2^p - 1$. If a Mersenne number is prime it is called a *Mersenne prime*.

Mersenne numbers are named after the French Minim friar Marin Mersenne who stated in 1644 in the preface of his *Cogitata Physico-Mathematica* that M_p is prime for $p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127$ and 257. He was wrong with $p = 67$ and $p = 257$ and he missed out M_{61}, M_{89} and M_{107}.
When looking for Mersenne primes, the following result is helpful:

Theorem 2.4. *If* $M_p = 2^p - 1$ *is prime, then* p *is prime.*

So one might restrict oneself to prime exponents, which shortens the finding process a lot. Most of the largest known prime numbers are Mersenne primes.They were found with a software based on the Lucas-Lehmer primality test which we will introduce in chapter 5.

2.2. Modular Arithmetic[2]

In this section we will give the basic definitions and results from modular arithmetic.

Definition.
- Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$. The smallest number k for which $a^k \equiv 1 \pmod{n}$ is called the *order of* a *modulo* n. It is denoted by $\text{ord}_n(a)$.
- The *Euler's totient function* or *Euler's phi function* $\phi(n)$ counts the natural numbers up to a given number n that are coprime to n:

$$\phi(n) := \left|\{a \in \mathbb{N} \quad |1 \leqslant a \leqslant n \wedge \gcd(a, n) = 1\}\right|.$$

The following theorem is a famous and important result by Swiss mathematician and physicist Leonhard Euler from 1763. You can find a proof in nearly every book about number theory, for example in [Bun08].

Theorem 2.5 (Euler's Theorem)**.** *Let* n, a *be coprime positive integers. Then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

[2]The main reference for this section is [Bun08]

Remark.
- The number of elements in $\mathbb{Z}_n^*$, the group of units of $\mathbb{Z}_n$, is $\phi(n)$.
- For any $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$, $\mathrm{ord}_n(a) \mid \phi(n)$.

We will need the following simple fact about the least common multiple of the first m numbers (see [Nai82]):

Lemma 2.6. *Let* $\mathrm{LCM}(m)$ *denote the* lcm *of the first* m *numbers.*

$$\mathrm{LCM}(m) \geqslant 2^m$$

for $m \in \mathbb{N}$ *with* $m \geqslant 7$.

A famous result in number theory which we will need as well is the

Theorem 2.7 (The Chinese Remainder Theorem). *Let* $n_1, \ldots, n_k$ *be a collection of pairwise coprime integers and* $a_i, \ldots, a_k$ *arbitrary integers. Then the system*

$$x \equiv a_i \pmod{n_i} \quad \textit{for} \quad 1 \leqslant i \leqslant k$$

has a solution $x = c$. *Further, if* $x = c$ *and* $x = c'$ *are two solutions, then*

$$c \equiv c' \pmod{n_1 n_2 \cdots n_k}.$$

See [HPS08] for a proof.

Definition. A number g is called a *primitive root modulo* n if it is a generator of $\mathbb{Z}_n^*$.

Theorem 2.8 (Theorem on the primitive root (Gauss)). $\mathbb{Z}_n^*$ *is a cyclic group if and only if* n *is equal to* $2, 4, p^k$ *or* $2 \cdot p^k$ *where* p *is an odd prime number and* k *a positive integer.*

If g is a primitive root modulo a prime number n, then all of the primitive roots for n are of the form g^a, where a is coprime to $n - 1$ (since this is the number of elements in $\mathbb{Z}_n^*$). Thus:

Lemma 2.9. *If* n *is prime, then there are exactly* $\phi(n-1)$ *primitive roots modulo* n *in the interval* $[1, n]$.

The next definition divides the elements a of $\mathbb{Z}_n^*$ into two categories. a is a *quadratic residue modulo* p if there is an $x \neq 0$ such that $a \equiv x^2 \pmod{p}$ and a *quadratic non-residue modulo* p if such an x does not exist.

Definition. For an odd prime number p and an integer a, the *Legendre symbol* is defined as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if} \quad a \quad \text{is a quadratic residue modulo} \quad p \\ -1 & \text{if} \quad a \quad \text{is a quadratic non-residue modulo} \quad p \\ 0 & \text{if} \quad a \equiv 0 \pmod{p} \end{cases}$$

For a composite number n, we have the *Jacobi symbol* which is a generalization of the Legendre symbol. Both are denoted by the same symbol and coincide for n prime.

Definition. Let $n = p_1^{\nu_1} \cdot p_2^{\nu_2} \cdots p_k^{\nu_k}$. The *Jacobi symbol* is then defined by:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\nu_1} \cdot \left(\frac{a}{p_2}\right)^{\nu_2} \cdots \left(\frac{a}{p_k}\right)^{\nu_k}$$

We now give some relations for Legendre and Jacobi symbols.

Theorem 2.10 (Euler's Criterion). *Let p be an odd prime and a an integer coprime to p. Then*

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

Theorem 2.11. *Let n, m be positive odd integers. Then the following relations hold:*

$$\left(\frac{-1}{m}\right) = (-1)^{\frac{m-1}{2}}, \tag{2.1}$$

$$\left(\frac{2}{m}\right) = (-1)^{\frac{m^2-1}{8}}. \tag{2.2}$$

For m, n coprime we also have the law of quadratic reciprocity

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{(m-1)(n-1)}{4}}. \tag{2.3}$$

We will use the last two theorems in chapter 5, especially 2.10. Proofs can be found in [Bun08], for example.

2.3. Polynomial Arithmetic

For the AKS test we will need some basic facts about polynomial rings, especially about quotient rings of polynomial rings and their arithmetic.

Definition.
- The *polynomial ring* $R[x]$ *in* x *over a ring* R is the set of all polynomials with coefficients in R:

$$R[x] = \left\{ \sum_{i=0}^{k} a_i x^i \quad | \quad k \in \mathbb{N}_0 \wedge a_i \in R \right\}.$$

  The *degree of a polynomial* is the largest k such that the coefficient a_i of x^i is not zero.

- Let $R[x]$ be a polynomial ring and $I = (h(x))$ the ideal generated by the polynomial $h(x)$. The *quotient ring* $R[x]/I$ is the set of all equivalence classes modulo I:

$$R[x]/(h(x)) = \left\{ f(x) + (h(x)) | f(x) \in R[x] \right\}.$$

In particular, we will be interested in quotient rings of polynomial rings over finite fields:

Notations. For any prime p and any polynomial $h(x)$ with coefficients in $\mathbb{F}_p$, $\mathbb{F}_p[x]/(h(x))$ denotes the quotient ring over the finite field $\mathbb{F}_p$.
We will also use the notation

$$f(x) \equiv g(x) \pmod{h(x), n}$$

to indicate that the equation $f(x) \equiv g(x)$ holds in the ring $\mathbb{Z}_n[x]/(h(x))$.

Definition. A non-constant polynomial is *irreducible over a field* K, if its coefficients belong to K and it cannot be factored into a product of non-constant polynomials with coefficients in K.

Remark. If p is prime and $h(x)$ is a polynomial of degree d which is irreducible over $\mathbb{F}_p$, then $\mathbb{F}_p[x]/(h(x))$ is a finite field of order p^d.

3. Primality Testing

A primality test is an algorithm which decides whether or not a given number is prime. In general it will not provide the factorization of the number, but only a PRIME or COMPOSITE answer. However, it can happen, that an algorithm returns COMPOSITE because it found a factor of the number. But again, finding factors is NOT the purpose of a primality testing algorithm.
There are basically two categories for primality tests: *deterministic* and *probabilistic* tests. Deterministic primality tests are the "for sure" primality tests, they prove either the primality or the compositeness of the input. They are sometimes also called "perfect tests". Probabilistic primality tests on the other hand can in general prove the compositeness of a number, but are only able to prove that the number is prime with a certain probability. So actually they do not really earn the name "primality test" which is the reason why some call them *compositeness tests*.

In this chapter we will give an overview of some of the most fundamental and famous primality tests. The Fermat test in section 3.2.1 and the Lucas test in section 3.3 provide the basis for essentially all primality tests, of which the elliptic curve primality test and the AKS test are only two examples. We will introduce these in chapter 6 and 7. Of course there are more primality tests then presented here. A good collection can be found in [CP05].

3.1. A Deterministic Method: The Sieve of Eratosthenes

Let us start with one of the most famous and simplest methods: the Sieve of Eratosthenes. It is named after the Greek mathematician Eratosthenes of Cyrene who lived around 200 BC. The method is as follows. Given a number n for which we want to determine if it is prime. We start by making a list of all integers between 2 and m, where m is the largest integer less than or equal to $\sqrt{n}$. Next, we circle 2 and cross off all multiples of two. Then we circle 3 and cross off all multiples of three. We don't have to circle 4, since it is already crossed off (it is a multiple of 2). Hence we continue with 5. When we reached the end of the list, we use *trial division* to determine whether or not n is prime. So we test if one of the circled numbers, which are the prime numbers up to $\sqrt{n}$, divides n. If we checked all circled numbers and no divisor is found, then n is prime. The reason why we only have to check all primes less than or equal to $\sqrt{n}$ is that if n is composite it has at least one divisor smaller or equal to $\sqrt{n}$.

The Sieve of Eratosthenes is indeed a very simple deterministic method and the algorithm is quite straightforward and easy to implement. But it is by no means efficient. In most cases the numbers to test for primality are very large, usually more than 100 digits. If we want to decide whether a relatively small number with just 20 digits is prime by using the Sieve of Eratosthenes it will take decades. It is possible to reduce the running time (see chapter 4) but it still won't be practical to use. Hence we need faster algorithms.

In the following sections we will find a way to accomplish this goal.

3.2. Probabilistic Methods[1]

There are basically two options to find quicker algorithms. One is to find a pattern among primes and then check if a given number follows that pattern. But so far, no useful pattern has been found. The approaches in the following subsections are all based on the second option that is finding patterns that are unique to composite numbers. However, as we will see there are some composite numbers that do not fit these patterns. We call these numbers *pseudoprimes*. To make this clear, all numbers that do fit the patterns are definitely composite, but there are numbers which do not fit the pattern and are not prime. So they seem to be prime, but are in fact composite. We will see that there are relatively few pseudoprimes and hence the methods are good in practice.

We will now introduce some of these probabilistic methods.

3.2.1. Fermat Test

The simplest probabilistic method is the Fermat Primality Test. It is fundamental for many other primality tests but not used in practice, since it is not really efficient. The test is based on Fermat's Little Theorem. Fermat stated the theorem in 1640 without giving a proof. It was Leibniz who first proved it about 40 years later.

Theorem 3.1 (Fermat's Little Theorem). *Let p be a prime number. Then*

1.
$$a^p \equiv a \pmod{p} \tag{3.1}$$
for any positive integer a. And

2.
$$a^{p-1} \equiv 1 \pmod{p} \tag{3.2}$$
if $\gcd(p, a) = 1$.

Proof. By induction over a. □

For the primality testing algorithm we use the contrapositive of the second statement:

Theorem 3.2 (Fermat's Primality Test). *A positive integer n is composite if there exists a positive integer a such that*

$$\gcd(n, a) = 1 \quad \textit{and} \quad a^{n-1} \not\equiv 1 \pmod{n}. \tag{3.3}$$

Algorithm 1 Fermat's Primality Test

```
Input: positive integer $n$ to be tested, $k$: parameter that determines the number of
    times to test for primality
Output: COMPOSITE if $n$ is composite, otherwise PROBABLY PRIME
 1: Repeat $k$ times:
 2: Pick $a$ randomly in the range $[2, n-1]$
 3: if $\gcd(n, a) \neq 1$ then
 4:     return COMPOSITE
 5: end if
 6: if $a^{n-1} \not\equiv 1 \pmod{p}$ then
 7:     return COMPOSITE
 8: end if
 9: if composite is never returned then
10:     return PROBABLY PRIME
11: end if
```

Definition. An integer a not satisfying (3.3) in Theorem 3.2, is called a *Fermat witness for (the compositeness of)* n. If such a witness exists, n is definitely composite.

As mentioned above there are composite numbers which do not fit this pattern.

Definition. Let n be a composite number. If $a^{n-1} \equiv 1 \pmod{n}$ for some integer a with $\gcd(n, a) = 1$, then n is called *pseudoprime (to the base* a).

This would not be a big problem, if these numbers were only pseudoprime for few bases. But in fact there are Fermat pseudoprimes which fulfill equation (2.2) for all positive integers a:

Definition. If n is Fermat pseudoprime for every positive integer a with $\gcd(n, a) = 1$, then n is called a *Carmichael number*.

The smallest Carmichael number is $561 = 3 \cdot 11 \cdot 17$. Carmichael numbers are always the product of at least three different prime numbers.
William Robert Alford, Andrew Granville and Carl Pomerance have not only shown that there infinitely many such numbers but also that they are not as rare as one might think:

Theorem 3.3 (Alford, Granville, Pomerance)**.** *There are infinitely many Carmichael numbers. In particular, for* x *sufficiently large, the number of Carmichael numbers up to* x*, denoted by* $C(x)$*, satisfies* $C(x) > x^{\frac{2}{7}}$.

P. Erdös had given a heuristic argument for a similar statement in 1956. The proof of Alford et al. can be found in [AGP94].

However, there are way less Carmichael numbers than there are prime numbers. Richard Pinch has found in 2007 that there are 20 138 200 Carmichael numbers up

[1]The main references for this section are [MD99] and [HPS08]

to 10^{21} [Pin]. Below 10^{21}, there are 21 127 269 486 018 731 928 primes (for example, see [Cal]); so there is less than a one-in-a-billion chance that a number passing the Fermat test for all positive integers a is a Carmichael number.

Luckily, there are probabilistic primality tests that are better - both in the efficiency and in the probability of a true output.

One example is the commonly used *Miller-Rabin test*, which we will discuss below.

3.2.2. Miller-Rabin

In 1976 the US-american computer scientist Gary Lee Miller published a primality test, which is deterministic under the assumption of the extended Riemann-Hypothesis. The Israeli computer scientist Michael Oser Rabin turned it into a probabilistic test in the following year. However, the US-american mathematician John Lewis Selfridge already used the test in 1974. That's why the *Miller-Rabin test* is also called *Miller-Rabin-Selfridge test*. It is based on the following theorem:

Theorem 3.4. *Let p be an odd prime and write $p-1 = 2^s d$ with d odd. Let a be any number not divisible by p. Then one of the following two conditions holds:*

1. $a^d \equiv 1 \pmod{p}$.
2. *One of $a^d, a^{2d}, a^{4d}, \dots, a^{2^{s-1}d}, a^{2^s d}$ is congruent to -1 modulo p.*

Proof. According to Fermat's Little Theorem, where $a^{p-1} \equiv 1 \pmod{p}$, we see that the last number in the list $a^d, a^{2d}, a^{4d}, \dots, a^{2^{s-1}d}, a^{2^s d}$ which is equal to a^{p-1}, is congruent to 1 modulo p. Further, every number in the list is the square of the previous number. Therefore, either the first number in the list is congruent to 1 modulo p or some number in the list is not congruent to 1 modulo p, but when it is squared repeatedly, it becomes 1. But the only number satisfying $b \not\equiv 1 \pmod{p}$ and $b^2 \equiv 1 \pmod{p}$ is -1. $\square$

Definition. An integer a satisfying none of the conditions stated in Theorem 3.4, is called a *Miller-Rabin witness for (the compositeness of)* n. If such a witness exists, n is definitely composite.

Like the pseudoprimes in Fermat's test, we also have pseudoprimes in the Miller-Rabin test:

Definition. If a composite n has the properties described in theorem 3.4 for some base a, then n is called a *strong pseudoprime to the base* a. If n is either a prime or a pseudoprime, then n is called PROBABLY PRIME.

The following proposition leads us to the Miller-Rabin Primality Test.

Proposition 3.5 (Strong Pseudoprimality Test). *If $n-1 = 2^s d$ with d odd and s nonnegative, then n is probably prime if $a^d \equiv 1 \pmod{n}$ or $a^{2^r d} \equiv -1 \pmod{n}$ for some nonnegative r less than s.*

We now have an even stronger primality test than the Fermat test. It reduces the number of pseudoprimes by half. More important, there are no Carmichael-like numbers for the Miller-Rabin test, and in fact, every composite number has many Miller-Rabin witnesses, as we will see below.

Proposition 3.6. *Let $n > 1$ be an odd composite integer. Then n passes the Strong Pseudoprimality Test for at most $\frac{n-1}{4}$ bases a with $1 < a < n$.*

To prove this, we will need the following Lemma, for a proof see [Ros86], for example.

Lemma 3.7. *Let p be an odd prime and let e and q be positive integers. Then the number of incongruent solutions x ($1 \leqslant x \leqslant p^\alpha$) of the congruence $x^q \equiv 1 \pmod{p^\alpha}$ is $\gcd(q, p^{\alpha-1}(p-1))$.*

We will now give a sketch of the proof of Proposition 3.6:

Proof. Since the proof is quite long, we only prove the proposition for the case that n is not square free. See [Ros86] for a detailed proof.
Let $n = 1 + 2^s d$ be a strong pseudoprime to the base a. So we either have

$$a^d \equiv 1 \pmod{n}$$

or

$$a^{2^r d} \equiv -1 \pmod{n}$$

for some r less than s. In particular we have

$$a^{n-1} = a^{2^s d} = (a^{2^r d})^{2^{s-r}} \equiv 1 \pmod{n}.$$

Let $p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ be the prime factorization of n. By Lemma 3.7 we know that the number of incongruent solutions of $x^{n-1} \equiv 1 \pmod{p_i^{\alpha_i}}$ is

$$\gcd(n-1, p_i^{\alpha_i - 1}(p_i - 1)) = \gcd(n-1, p_i - 1)$$

for $i = 1, \ldots, k$. By the Chinese Remainder Theorem there are exactly

$$\prod_{i=1}^{k} \gcd(n-1, p_i - 1)$$

incongruent solutions of

$$x^{n-1} \equiv 1 \pmod{n}. \tag{3.4}$$

As we consider the case where the prime factorization of n contains a prime power $p_j^{\alpha_j}$ with $\alpha_i \geqslant 2$ and as n is odd and therefore $p_j \geqslant 3$, we have

$$\frac{p_j - 1}{p_j^{\alpha_j}} = \frac{1}{p_j^{\alpha_j - 1}} - \frac{1}{p_j^{\alpha_j}} \leqslant \frac{2}{9}.$$

For $n \geqslant 9$, we then get

$$\begin{aligned}
\prod_{i=1}^{k} \gcd(n-1, p_i - 1) &\leqslant \prod_{i=1}^{k}(p_i - 1) \\
&\leqslant \prod_{i \neq j}^{k}(p_i - 1)\frac{2 \cdot p_j^{\alpha_j}}{9} \\
&\leqslant \prod_{i \neq j}^{k} p_i \frac{2 \cdot p_j^{\alpha_j}}{9} \\
&\leqslant \frac{2}{9}n \\
&\leqslant \frac{n-1}{4}.
\end{aligned}$$

So there are at most $\frac{n-1}{4}$ incongruent solutions of (3.4) if $n \geqslant 9$, and therefore the number of bases $a \leqslant n-1$ for which n is a strong pseudoprime is less than or equal to $\frac{n-1}{4}$. In [Ros86] you can find the case where n is square free, which we did not treat here. □

Proposition 3.8 (Miller-Rabin Probabilistic Primality Test). *Suppose we test n for k randomly selected bases (as in the algorithm). If n is prime, then the result of the algorithm is always correct. If n is composite, then the probability that the algorithm outputs PRIME is at most $\frac{1}{4^k}$.*

Proof. The first claim is obviously true. So suppose n is composite. By Proposition 3.6, the probability that n passes the Strong Pseudoprimality Test for a randomly selected base a is at most $\frac{1}{4}$. Therefore, the probability that n passes the test for k different bases is at most $\frac{1}{4^k}$. □

So if k is large, the probability that a number is indeed prime, when the Miller-Rabin test determines that it is PROBABLY PRIME is very high. As we will see in chapter 4, the Miller-Rabin test is a very efficient probabilistic primality test and therefore often used in practice.

Note, that the Miller-Rabin test might provide a deterministic test :

Remark. If the Riemann Hypothesis is true and n passes the Strong Pseudoprimality Test for all integers a such that $1 < a < 2 \cdot \log^2(n)$, then n is prime.

See [Mil76] for more details.

3.3. Lucas Tests[2]

In 1876 the French mathematician Édouard Lucas proved a theorem on which the Lucas primality test is build on. The following version of *Lucas Theorem* from 1953 is due to the US-American mathematician Derrick Henry Lehmer.

[2]The main references for this section are [CP05] and [Pom02]

Theorem 3.9 (Lucas Theorem). *Let n, a be integers with $n > 1$. If*

$$a^{n-1} \equiv 1 \pmod{n} \tag{3.5}$$

and

$$a^{\frac{n-1}{q}} \not\equiv 1 \pmod{n} \tag{3.6}$$

for all prime divisors q of $n-1$, then n is prime.

Proof. By the congruences in (3.5) and (3.6), we infer that the order of a must divide $n-1$ but is does not divide $\frac{n-1}{q}$ for all prime divisors q of $n-1$, which means that the order of a must be equal to $n-1$. By Euler's Theorem it then follows that $n-1$ divides $\phi(n)$. If n is composite, $\phi(n) < n-1$. But then $n-1$ cannot be a divisor of $\phi(n)$, so n must be prime. □

Since this is the converse of Fermat's Little Theorem, it seems to be perfect to build a (deterministic) primality test upon. We only have to find the prime factors of $n-1$ and a number a such that (3.5) and (3.6) hold. But there is the problem. Whereas finding the number a will not be a big obstacle, finding all prime factors of a huge number will be. In general no fast factoring algorithm exists. Nevertheless, there are numbers, which can be easily factored and Theorem 3.9 provides a great primality test in that case.
Before we consider such a case, we still have to argue how we can find a suitable number a. By Lemma 2.9 we know that if n is prime there are $\phi(n-1)$ primitive roots modulo n in the interval $[1, n]$ (which satisfy (3.5) and (3.6)) and that $\phi(n) > \frac{n}{2\log\log n}$ for $n > 200560490131$ (see [CP05]). So the probability that a randomly chosen number a in the interval $[1, n]$, satisfies the congruences (3.5) and (3.6) is at least $\frac{1}{2\log\log n}$ if n is large. Thus, the expected number of random choices, until a suitable number a is found, is $2\log\log n$. So if we knew all prime divisors of $n-1$ we would be able to determine if n is prime in polynomial time. So let's consider a case, where we can factor $n-1$ easily, for example the Fermat numbers where we apply Theorem 3.9 with $a = 3$.

Corollary 3.10 (Pépin Test). *For $k \geqslant 1$, consider the Fermat numbers $F_k = 2^{2^k} + 1$. F_k is prime if and only if $3^{\frac{F_k-1}{2}} \equiv -1 \pmod{F_k}$.*

Proof. Assume that the congruence holds. Then we have that:

$$3^{F_k-1} \equiv 1 \pmod{F_k}$$

and

$$3^{\frac{F_k-1}{2}} \not\equiv 1 \pmod{F_k}$$

So by Theorem 3.9 F_k must be prime.
Conversely, suppose now that F_k is prime. $2^{2^k} \equiv 1 \pmod{3}$ since 2^k is even. It follows that $F_k \equiv 2 \pmod{3}$. We also have that $F_k \equiv 1 \pmod{4}$, which means, that 3 cannot be a square modulo F_k, so the Legendre symbol $\left(\frac{3}{F_k}\right) = -1$. By Theorem 2.10 (Euler's Criterion), the congruence in the corollary holds. □

The test is named after the French mathematician Théophile Pépin. In 1877, he gave a criterion similar to the above, in which 5 was used instead of 3 and with $k \geqslant 2$.
The largest F_k for which the Pépin test has been used is F_{24} (as of September 2016). This number is composite and as we already mentioned in 2.1 so are all Fermat numbers F_k for $k > 4$ which have been studied.
Since only few numbers can be proven prime by this primality test, it is not really satisfying. How else can we use Theorem 3.9 to get a primality test? We need to solve the problem of factoring $n-1$ completely. Let's assume, that we have not found the complete factorization of $n-1$ but a partial factorization. With the following theorem we will then get the *Pocklington primality test*.

Theorem 3.11 (Pocklington Theorem). *Let $n > 1$ be an integer with $n-1 = F \cdot R$, such that the complete factorization of F is known. Let a be an integer such that*

$$a^{n-1} \equiv 1 \pmod{n} \tag{3.7}$$

and

$$\gcd(a^{\frac{n-1}{q}}, n) = 1 \tag{3.8}$$

for every prime divisor q of F. If $F \geqslant \sqrt{n}$, then n is prime.

Proof. Let p be a prime dividing n. From (3.7) we have that $a^{n-1} \equiv 1 \pmod{p}$, so the order of a^R modulo p divides $\frac{n-1}{R} = F$. From (3.8), we have that $(a^R)^{\frac{F}{q}} \not\equiv 1 \pmod{p}$ for every prime q dividing F, so $\mathrm{ord}_p(a^R)$ cannot be a proper divisor of F and hence $\mathrm{ord}_p(a^R) = F$. Since $\mathrm{ord}_p(a^R) \leqslant p-1$ and $F \geqslant \sqrt{n}$ it follows that $p > \sqrt{n}$, so n must be prime. □

Konyagin and Pomerance have given a criterion for the primality of n that allows F to be even smaller than $\sqrt{n}$. The resulting probabilistic primality test goes by the name $n-1$-*Test*. It can be found in [CP05]

4. Time Complexity

Our goal is to find a deterministic primality testing algorithm with polynomial running time. But we did not mention yet, what we exactly mean by that. In this chapter we will focus on the basic theory of time complexity and explain what a "good" running time is.

The running time or time complexity of an algorithm quantifies the amount of time (by time we mean the number of operations) taken by the algorithm as a function of the length of the input. Since the input is normally binary, the length of the input is $\log n$. When we write $\log n$ we always mean the base 2 logarithm. We will also need the base e logarithm, which we will denote with $\ln n$

The time complexity of an algorithm is usually expressed using the *big-$\mathcal{O}$ notation*. So before we start talking about the running time in detail, we will introduce some basic time complexity concepts.

4.1. Notations and Basics

Definition. Let $f(x)$ and $g(x)$ be functions of x defined on a subset of $\mathbb{R}$. We say that f *is a big-$\mathcal{O}$ of* g as $x \to \infty$ and write

$$f(x) = \mathcal{O}(g(x))$$

if there is a positive real number c and a real number x_0 such that

$$|f(x)| \leqslant c|g(x)| \quad \text{for all} \quad x \geqslant x_0.$$

Note, that when we write $f(x) = \mathcal{O}(g(x))$ we actually mean $f(x) \in \mathcal{O}(g(x))$, since $\mathcal{O}(g(x))$ is a set of functions. When we write $\mathcal{O}(f(x)) = \mathcal{O}(g(x))$, we mean $\mathcal{O}(f(x)) \subseteq \mathcal{O}(g(x))$.

Example. Let $f(x) = 4x^3 - 7x^2 + 5$. Then: $f(x) = \mathcal{O}(x^3)$, since x^3 is the fastest growing term in $f(x)$. Formally: Let $x > x_0 = 1$ and $c = 16$. We then have:

$$\begin{aligned} |4x^3 - 7x^2 + 5| &\leqslant 4x^3 + 7x^2 + 5 \\ &\leqslant 4x^3 + 7x^3 + 5x^3 \\ &= 16x^3 \end{aligned}$$

So $|4x^3 - 7x^2 + 5| \leqslant 16|x^3|$ for all $x > 1$.

Definition. A function $f(x)$ *is a big-Ω of* g and write

$$f(x) = \Omega(g(x))$$

, if there are positive constants c and x_0 such that

$$f(x) \geqslant cg(x) \quad \text{for all} \quad x \geqslant x_0.$$

We will also use the *soft-$\mathcal{O}$ notation*:

Definition. A function $f(x)$ is called a *soft-$\mathcal{O}$* of $g(x)$, denoted by $f(x) = \hat{\mathcal{O}}(g(x))$, if

$$f(x) = \mathcal{O}(g(x)\log^k(g(x))) \quad \text{for some} \quad k.$$

It is essentially a big-$\mathcal{O}$ notation ignoring logarithmic factors.

Example. $\hat{\mathcal{O}}(\log^m(x)) = \mathcal{O}(\log^m x \log^k(\log(x))) = \mathcal{O}(\log^{m+\epsilon}(x))$ for any $\epsilon > 0$, since $\log^k(x) = \mathcal{O}(x^\epsilon)$ for any $\epsilon > 0$.

So what is a good running time? In our first example, the Sieve of Eratosthenes, the running time is ok if the number to be tested is small. But if the numbers get bigger, we might have to wait years until we know the result. This is obviously too long. So what we want is that even if we test big numbers the running time should not get too big. To ensure this we want our algorithms to have *polynomial time complexity*. Remember that the input is binary, so if we talk about polynomial running time we mean *polynomial in the length of the input*. Hence the algorithms should run in time $\mathcal{O}(\log^k(n))$, for some positive constant k.
An algorithm is said to have *exponential time complexity* if it runs in $\mathcal{O}(n^k) = \mathcal{O}(2^{\log^k n})$, for some positive constant k. An algorithm that is slower than polynomial but not as slow as exponential is said to have *quasi-polynomial time complexity*. For example, $\mathcal{O}((\log n)^{\log\log n})$ would be a quasi-polynomial running time.

For analyzing the time complexity of algorithms, we need some basic properties of the big-$\mathcal{O}$ notation:

Proposition 4.1 (Properties). *Let f, g, f_1, f_2, g_1, g_2 be functions and $f_1 = \mathcal{O}(g_1)$, $f_2 = \mathcal{O}(g_2)$, then*

1. $f_1 \cdot f_2 = \mathcal{O}(g_1 \cdot g_2)$.

2. $f \cdot \mathcal{O}(g) = \mathcal{O}(fg)$.

3. *for* $k \neq 0$: $\mathcal{O}(kg) = \mathcal{O}(g)$.

4. $f_1 + f_2 = \mathcal{O}(|g_1| + |g_2|)$.

5. $f_1 = \mathcal{O}(g)$ *and* $f_2 = \mathcal{O}(g) \Rightarrow f_1 + f_2 = \mathcal{O}(g)$.

The proof is quite simple. It can be found in nearly every book or lecture notes about complexity theory and related topics.

Obviously, different algorithms can lead to very different time complexities. We already talked about how probabilistic primality tests can lead to faster algorithms. However, we are always (not only when it comes to primality testing) interested in an algorithm which can actually solve a problem (= a deterministic algorithm) in the best running time possible. By "problem" we mean a *decision problem*, that is a question which has a *yes* or *no* answer. For example, PRIME is a decision problem. A set of decision problems which have related complexity, is called a *complexity class*. Lets shortly (informally) introduce the most important (at least for our purposes) complexity classes:

- **P** is the set of all decision problems, which are solvable in polynomial time.
- **NP** is the set of all decision problems, where the yes-instance of the problem can be verified in polynomial time.

In which complexity class a problem is categorized is based on the fastest *known* algorithm. As such, the class a problem belongs to may change over time if a faster algorithm is discovered. Based on the primality tests above we don't know if PRIMES is in **P**, since none of them is a deterministic polynomial time algorithm. But in chapter 7 (and we already mentioned this in the introduction) we will see that such an algorithm exists.

Remark. It is an open problem in computer science if $\mathbf{P} = \mathbf{NP}$. It is widely believed that the answer is no.

4.2. Analyzing the running time of some algorithms

In this section we analyze the time complexity of some of the primality tests above. Even if we are in general interested in a deterministic polynomial-time algorithm, we will also analyze the time complexity of the (probabilistic) Miller-Rabin test. As we saw in the last chapter, the probability of a true output is very high for the Miller-Rabin test. So if it is much faster than the deterministic tests we considered so far, we might want to compromise and take the (little) risk, that Miller-Rabin tells us our number is prime even if it is not.

Before we start, we will give the time complexity of the some fundamental operations:

Lemma 4.2. *For any numbers of length* $\log n$*,*

1. *addition and subtraction can be done in time* $\mathcal{O}(\log n)$*,*
2. *multiplication and division can be done in time* $\mathcal{O}(\log^2 n)$*,*
3. *multiplication modulo* n *can be done in time* $\mathcal{O}(\log^2 n)$*,*

4. *finding out if* $n = a^b$ *can be done in time* $\hat{\mathcal{O}}(\log^3 n)$*,*

5. *the greatest common divisor of two numbers can be computed in time* $\mathcal{O}(\log n)$*.*

Proof. 1. is simple: since the sum of two numbers of size $\log n$ is at most of size $\log n + 1$ and each bit of the sum gets computed in a constant time, the time complexity is linear, thus $\mathcal{O}(\log n)$.
For 2., we use long multiplication (the one we all learned at school), for multiplying two numbers of size $\log n$: in base 2, we get $\log n$ rows of length at most $2 \log n$, which we then have to add up. Since every addition takes at most time $\mathcal{O}(\log n)$, the running time of multiplying two numbers is $\mathcal{O}(\log^2 n)$. Note that to divide two numbers n and m, we have to find numbers q, r, such that $n = q \cdot m + r$, where $r > m$. It is easy to see, that like multiplication, this takes time $\mathcal{O}(\log^2 n)$.
3. follows from 2: we first multiply the numbers, which gives a number of length at most $2 \log n$ and then compute the remainder upon dividing the product by the modulus n. Since both operations take time $\mathcal{O}(\log^2 n)$, multiplication modulo n can still be done in time $\mathcal{O}(\log^2 n)$.
For 4. and 5., see [vzGG99]. □

Using *Fast Fourier transforms* we can improve the running time for multiplication (and therefore for division and for multiplication modulo n):

Lemma 4.3. *With the Schönhage-Strassen algorithm, multiplication, division can be done in time* $\hat{\mathcal{O}}(\log n)$*. Similarly, these operations on two degree* d *polynomials with coefficients of length at most* $\log n$ *can be done in time* $\hat{\mathcal{O}}(d \cdot \log n)$*.*

For more details, see [vzGG99], for example.

4.2.1. Sieve of Eratosthenes

To analyze the time complexity of the Sieve of Eratosthenes, lets recall how the method works:
First, we write down all numbers between 2 and n. Then we compute the multiples of all prime numbers $p \leqslant n$, that is $p, 2p, 3p \ldots kp$, until $kp \geqslant n$. So for each prime p, the number of multiplications is smaller than or equal to $\frac{n}{p}$. Therefore, the total number of multiplications is

$$\sum_{p \leqslant \sqrt{n}} \frac{n}{p}.$$

The following theorem follows directly from Theorem 427 (page 351) in [HW75]:

Theorem 4.4. *Let* n *be a positive integer and* p *be prime. Then* $\sum_{p \leqslant \sqrt{n}} \frac{n}{p} = n \ln \ln \sqrt{n} + \mathcal{O}(n)$*.*

Since $\mathcal{O}(\ln \ln x) = \mathcal{O}(\log \log x)$ (changing the base of a logarithms only results in a constant change) the running time of the Sieve of Eratosthenes for finding all primes up to $\sqrt{n}$ is $\mathcal{O}(n \log \log \sqrt{n})$.

Now, to find out if n is prime, we have to test if one of the primes $p \leqslant n$ divides n. In the worst case, that is if n is prime, this gives another $\pi(\sqrt{n})$ operations.

The running time of the Sieve of Eratosthenes can be improved. Instead of writing down all numbers greater than or equal to 2, we write down 2 and all odd numbers, since we already know, that all even numbers cannot be prime. Calculating the multiples of 2 takes $\frac{n}{2}$ operations, so it seems like we could save much time with this. But unfortunately, it is not that easy. In fact, by deleting the multiples of 2 in our list we indeed save up some time, but we also have to change indices in the algorithm, which in total gives us a larger running time. But if we also deleting the multiples of $3, 5, 7, 11$ and 13, we get in total a better running time. If we delete the multiples of larger prime numbers, we again get a worse running time [MO].

Another possibility to improve the running time, is to keep prime numbers in a database. But this would not be very efficient concerning the space complexity.

Overall, primality proving with the Sieve of Eratosthenes is not very efficient. The running time is exponential. Lets see, what probabilistic tests can offer.

4.2.2. Miller-Rabin

Recall the Miller-Rabin Test: for k randomly chosen integers a, it tests if n is a strong pseudoprime to base a. Lets see the pseudocode for details:

Algorithm 2 Miller-Rabin Primality Test

```
Input: positive integer $n = 2^s d + 1$ to be tested, $k$: parameter that determines the
    number of times to test for primality
Output: COMPOSITE if $n$ is composite, otherwise PROBABLY PRIME
 1: $i := 1$
 2: for $i = 1$ to $k$ do
 3:    Randomly choose an integer $a$ with $1 < a < n$ and $p \nmid a$
 4:    if $a^d \equiv 1 \pmod{n}$ then
 5:       $i := i + 1$
 6:       Repeat step 3
 7:    end if
 8:    if $a^{2^r d} \equiv -1 \pmod{n}$ for some nonnegative $r$ less than $s$ then
 9:       $i := i + 1$
10:       Repeat step 3
11:    end if
12:    return COMPOSITE
13: end for
14: return PROBABLY PRIME
```

To determine if n is a strong pseudoprime, the algorithm computes the powers $a^d, a^2 d, \ldots, a^{2^s d}$ modulo n. By using the binary representation of d and repeated squaring, computing a^d requires $\mathcal{O}(\log n) \mod n$-multiplication operations. By Lemma

4.2 every mod n-multiplication requires time $\mathcal{O}(\log^2 n)$ using the usual algorithms for integer multiplication and division. The remaining powers of a^d can then be computed via repeated squaring modulo n, which again can be done in time $\mathcal{O}(\log n)$. Therefore, for each number a, it takes time $\mathcal{O}(\log^3 n)$ to determine if n is a strong pseudoprime. So testing k different bases a yields a time complexity of $\mathcal{O}(k \log^3 n)$ in total.

The time complexity can be improved by using the Schönhage-Strassen multiplication algorithm. The Miller-Rabin test then takes time $\hat{\mathcal{O}}(k \log^2 n)$.

The Miller-Rabin Test is therefore a probabilistic polynomial-time algorithm.

4.2.3. Pépin's Test

We will now analyze Pépin's test for Fermat numbers. Given a Fermat number $F_k = 2^{2^k} + 1$, we only have to check if $3^{\frac{F_k - 1}{2}} \equiv -1 \pmod{F_k}$ (see Algorithm 3). This involves $\log F_k$ multiplications modulo F_k. By Lemma 4.2 each multiplication can be done in time $\mathcal{O}(\log^2 F_k)$, which gives a total running time of $\mathcal{O}(\log^3 F_k)$.

Using the Schönhage-Strassen multiplication algorithm, we get a time complexity of $\hat{\mathcal{O}}(\log^2 F_k)$.

So Pépin's test is a deterministic polynomial-time algorithm for Fermat numbers.

Algorithm 3 Pepin Test

```
Input: positive integer $F_k = 2^{2^k} + 1$, $k \geqslant 1$, to be tested
Output: COMPOSITE if $n$ is composite, otherwise PRIME
1: if $3^{\frac{F_k - 1}{2}} \equiv -1 \pmod{F_k}$ then
2:     return PRIME
3: end if
4: return COMPOSITE
```

5. Primality Tests based on Lucas Sequences

We saw in chapter 3 that if n is of a special form, it could be more easily verified, if it is prime. Especially the Pepin Test for Fermat numbers is very efficient. In this chapter we introduce *Lucas sequences* and some primality tests based on them. For Mersenne prime numbers this will lead us to the very efficient *Lucas-Lehmer-Test*.

5.1. Lucas Sequences

Let $a, b \in \mathbb{Z}, a, b \neq 0$. Consider the polynomial $f(x) = x^2 - ax + b$ and it's discriminant $\Delta = a^2 - 4b$. The two roots α and β of f are

$$\alpha = \frac{a + \sqrt{\Delta}}{2} \quad \text{and} \quad \beta = \frac{a - \sqrt{\Delta}}{2}.$$

The following relations among a, b, Δ, α and β hold, as one can easily check:

$$\alpha + \beta = a, \alpha - \beta = \sqrt{\Delta}, \alpha \cdot \beta = b.$$

We assume that $\Delta \neq 0$. The *Lucas sequences* $U(a, b)$ and $V(a, b)$ are then defined as follows:

Definition. For each $k \geqslant 0$ consider the integers

$$U_k(a, b) = \frac{\alpha^k - \beta^k}{\alpha - \beta}$$

and

$$V_k(a, b) = \alpha^k + \beta^k.$$

The *Lucas sequences* of the pair (a, b) are defined by

$$U(a, b) = (U_0(a, b), U_1(a, b), U_2(a, b), \dots)$$

$$V(a, b) = (V_0(a, b), V_1(a, b), V_2(a, b), \dots)$$

A famous example of a Lucas sequence is the sequence of *Fibonacci numbers*. It is obtained by choosing $a = 1, b = -1$ in $U(a, b)$. The numbers of the corresponding sequence V are called *Lucas numbers*.

[1]The main references for this chapter are [CP05] and [Rib11]

For every $k \geqslant 2$ we have

$$\begin{aligned} U_k(a,b) &= aU_{k-1}(a,b) - bU_{k-2}(a,b) \\ V_k(a,b) &= aV_{k-1}(a,b) - bV_{k-2}(a,b) \end{aligned}$$

(to see this, simply substitute $a = \alpha + \beta$ and $b = \alpha\beta$). Since $U_0 = 0, V_0 = 2$, we see that the Lucas sequences are defined by a linear recurrence (and indeed consist of integers). From now on, we will write U_k, V_k instead of $U_k(a,b), V_k(a,b)$.

We will need the following properties:

Lemma 5.1. *Let n be a positive integer. Then*

$$U_{2n} = U_n V_n.$$

Proof.

$$U_n V_n = \frac{\alpha^n - \beta^n}{\alpha - \beta}(\alpha^n + \beta^n) = \frac{\alpha^{2n} - \alpha^{2n}}{\alpha - \beta} = U_{2n}.$$

□

Lemma 5.2. *For all integers n, we have:*

$$V_{2n} = V_n^2 - 2b^n.$$

Proof.

$$V_{2n} = \alpha^{2n} + \beta^{2n} = (\alpha^n + \beta^n)^2 - 2\alpha^n\beta^n = V_n^2 - 2b^n.$$

□

Lemma 5.3. *Let n, m be positive integers. Then*

$$2V_{m+n} = V_m V_n + \Delta U_m U_n.$$

Proof.

$$\begin{aligned} V_m V_n + \Delta U_m U_n &= (\alpha^m + \beta^m)(\alpha^n + \beta^n) + \Delta \frac{\alpha^m - \beta^m}{\alpha - \beta}\frac{\alpha^n - \beta^n}{\alpha - \beta} \\ &= \alpha^{m+n} + \beta^{m+n} + \alpha^m\beta^n + \alpha^n\beta^m + \Delta\frac{\alpha^{m+n} + \beta^{m+n} - \alpha^m\beta^n - \alpha^n\beta^m}{\Delta} \\ &= 2(\alpha^{m+n} + \beta^{m+n}) = 2V_{m+n}. \end{aligned}$$

□

Lemma 5.4. *For all positive integers $m > n$, we have $U_{m+n} = U_m V_n - b^n U_{m-n}$.*

Proof. Let $m > n$. Then:

$$\begin{aligned}
U_m V_n - b^n U_{m-n} &= \frac{\alpha^m - \beta^m}{\alpha - \beta}(\alpha^n + \beta^n) - (\alpha\beta)^n \frac{\alpha^{m-n} - \beta^{m-n}}{\alpha - \beta} \\
&= \frac{\alpha^m\alpha^n - \beta^m\beta^n + \alpha^m\beta^n - \alpha^n\beta^m}{\alpha - \beta} - \frac{\alpha^m\beta^n - \alpha^n\beta^m}{\alpha - \beta} \\
&= \frac{\alpha^m\alpha^n - \beta^m\beta^n}{\alpha - \beta} \\
&= U_{m+n}.
\end{aligned}$$

□

Lemma 5.5. $V_n^2 = \Delta U_n^2 + 4b^n$ *for every integer* n.

Proof. By Lemma 5.3 we have $2V_{2n} = V_n^2 - \Delta U_n$. With Lemma 5.2 we get:

$$\begin{aligned}
2(V_n^2 - 2b^n) &= V_n^2 + \Delta U_n^2 \\
V_n^2 &= \Delta U_n^2 + 4b^n.
\end{aligned}$$

□

Lemma 5.6. *For every integer* n *we have*

$$2^{n-1}U_n = \sum_{k\leqslant n,\text{odd}} \binom{n}{k} a^{n-k}\Delta^{\frac{k-1}{2}}.$$

Proof.

$$\begin{aligned}
U_n &= \frac{\alpha^n - \beta^n}{\alpha - \beta} = \frac{(\frac{a+\sqrt{\Delta}}{2})^n - (\frac{a-\sqrt{\Delta}}{2})^n}{\sqrt{\Delta}} \\
&= \frac{1}{2}^n \frac{1}{\sqrt{\Delta}}(\sum_{k=0}^{n} \binom{n}{k} a^{n-k}(\sqrt{\Delta})^k - \sum_{k=0}^{n} \binom{n}{k} a^{n-k}(-\sqrt{\Delta})^k) \\
2^n\sqrt{\Delta}U_n &= 2\sum_{k\leqslant n,\text{odd}} \binom{n}{k} a^{n-k}(\sqrt{\Delta})^k \\
2^{n-1}U_n &= \sum_{k\leqslant n,\text{odd}} \binom{n}{k} a^{n-k}(\sqrt{\Delta})^{k-1} \\
2^{n-1}U_n &= \sum_{k\leqslant n,\text{odd}} \binom{n}{k} a^{n-k}\Delta^{\frac{k-1}{2}}
\end{aligned}$$

□

Lemma 5.7. *Let* p *be an odd prime. Then*

$$U_p \equiv (\frac{\Delta}{p}) \pmod{p} \tag{5.1}$$

Proof. By Lemma 5.6 we have

$$2^{p-1}U_p = \sum_{k \leqslant n, \text{odd}} \binom{p}{k} a^{p-k} \Delta^{\frac{k-1}{2}}.$$

Since $\binom{p}{k} \equiv 0 \pmod{p}$ for all k with $1 \leqslant k \leqslant p-1$, we get (with Fermat's Little theorem and Euler's Criterion)

$$\begin{aligned} 2^{p-1}U_p &\equiv \Delta^{\frac{p-1}{2}} \pmod{p} \\ U_p \equiv 2^{p-1}U_p &\equiv \left(\frac{\Delta}{p}\right) \pmod{p}. \end{aligned}$$

□

Lemma 5.8. *Let p be an odd prime. Then*

$$V_p \equiv a \pmod{p}. \tag{5.2}$$

See [Rib11] for a proof.

Lemma 5.9. *For all positive integers n, k, we have that $U_n \mid U_{kn}$.*

Proof. Induction over k and Lemma 5.4. □

Definition. Let n be a positive integer. The *rank of appearance of* n, denoted by $r_f(n)$, is the least positive integer r with $U_r \equiv 0 \pmod{n}$.

Lemma 5.10. *Suppose $r_f(n)$ exists, with $\gcd(n, b) = 1$. Then $U_k \equiv 0 \pmod{n}$ if and only if $r_f(n) \mid k$.*

Proof. Suppose $r_f(n) \mid k$. By Lemma 5.9 we then have that $U_{r_f(n)} \mid U_k$. Since $n \mid U_{r_f(n)}$ we therefore have $n \mid U_k$.
Now suppose $r_f(n) \nmid k$ and write $k = lr_f + m$, $l > 0, 0 < m < r_f$ (if $k < r_f$ we are done by definition of r_f). We then have to show that $n \nmid U_k$.
We will use induction over l. For $l = 1$ we have:

$$U_k = U_{r_f+m} = U_{r_f}V_m - b^m U_{r_f-m},$$

where the first part is divisible by n but the second part is not, since $\gcd(n, b) = 1$ and $r_f - m < r_f$. So $U_k = U_{r_f+m}$ is not divisible by n.
Let's suppose $n \nmid U_k = U_{(l-1)r_f+m}$ for every m with $0 < m < r_f$. We then have:

$$U_k = U_{lr_f+m} = U_{lr_f}V_m - b^m U_{lr_f-m}.$$

The first part is divisible by n, since $n \mid U_{lr_f}$. The second part can be written as $b^m U_{lr_f-m} = b^m U_{(l-1)r_f+(r_f-m)}$. By induction hypothesis and since $\gcd(n, b) = 1$, $n \nmid b^m U_{(l-1)r_f+(r_f-m)}$ and therefore $n \nmid U_k = U_{lr_f+m}$. □

Lemma 5.11. *Let p be an odd prime with $p \nmid ab\Delta$. Then $r_f(p) \mid p - \left(\frac{\Delta}{p}\right)$.*

Proof. Suppose $\left(\frac{\Delta}{p}\right) = -1$. By Lemma 5.6 we have

$$2^p U_{p+1} = \sum_{k \leqslant p+1, \text{odd}} \binom{p+1}{k} a^{p+1-k} \Delta^{\frac{k-1}{2}}.$$

$p \mid \binom{p+1}{k} a^{p+1-k} \Delta^{\frac{k-1}{2}}$ for all k with $1 < k < p$. Therefore, we get

$$\begin{aligned} 2^p U_{p+1} &\equiv (p+1)a^p + (p+1)a\Delta^{\frac{p-1}{2}} \pmod{p} \\ &\equiv a + a\Delta^{\frac{p-1}{2}}, \end{aligned}$$

where the last equivalence follows by Fermat's Little Theorem. By Theorem 2.10, $\Delta^{\frac{p-1}{2}} \equiv \left(\frac{\Delta}{p}\right) \pmod{p}$. Since $\left(\frac{\Delta}{p}\right) = -1$, we thus get

$$2^p U_{p+1} \equiv a(1 + (\frac{\Delta}{p})) \equiv 0 \pmod{p}.$$

So $p \mid U_{p+1}$ and therefore (by Lemma 5.10) $r_f(p) \mid p+1$.

Now suppose that $\left(\frac{\Delta}{p}\right) = 1$. That means, there exists an x such that $\Delta = a^2 - 4b \equiv x^2 \pmod{p}$. By Lemma 5.6 we have

$$2^{p-2} U_{p-1} = \sum_{k \leqslant p-1, \text{odd}} \binom{p-1}{k} a^{p-1-k} \Delta^{\frac{k-1}{2}}.$$

Since $\binom{p-1}{k} \equiv -1 \pmod{p}$ for k odd, we get

$$\begin{aligned} 2^{p-2} U_{p-1} &\equiv -\sum_{k \leqslant p-1, \text{odd}} a^{p-1-k} \Delta^{\frac{k-1}{2}} \\ &\equiv -a(a^{p-3} + a^{p-5}\Delta + \cdots + \Delta^{\frac{p-3}{2}}) \\ &\equiv -a\frac{a^{p-1} - \Delta^{\frac{p-1}{2}}}{a^2 - \Delta} \\ &\equiv -a\frac{a^{p-1} - x^{p-1}}{a^2 - \Delta} \\ &\equiv 0 \pmod{p}. \end{aligned}$$

Therefore, $p \mid U_{p-1}$ and (by Lemma 5.10) $r_f(p) \mid p-1$ (Note, that since $p \nmid ab\Delta$, $a^2 - \Delta \not\equiv 0$ and $p \nmid x$). Since $p \nmid \Delta$, $\left(\frac{\Delta}{p}\right)$ cannot be equal to 0, so overall we have

$$r_f(p) \mid p - (\frac{\Delta}{p}).$$

□

5.2. Primality Tests

The following theorem by Michael Morrison from 1975 is similar to Theorem 3.11. It is often referred to as "$n+1$ test".

Theorem 5.12 (Morrison). *Suppose* $n > 1$ *is an odd integer,* a, b *are integers with* $\gcd(b,n) = 1$ *and that the Jacobi symbol* $\left(\frac{\Delta}{n}\right) = -1$. *Also assume that* F *is an integer with* $F \mid n+1, F > \sqrt{n}+1$ *and that*

$$U_{n+1} \equiv 0 \pmod{n}, U_{\frac{n+1}{q}} \in \mathbb{Z}_n^*$$

for every prime divisor q *of* F. *Then* n *is prime.*

Proof. Since $U_{n+1} \equiv 0 \pmod{n}$, $r_f(n) \mid n+1$ and therefore $r_f(p) \mid n+1$ for every prime factor p of n. But since $p \nmid U_{\frac{n+1}{q}}$, by Lemma 5.10 we have $r_f(p) \nmid \frac{n+1}{q}$ for every prime divisor q of F. So every prime factor q must be a divisor of $r_f(p)$. By Lemma 5.11 we thus have $F \mid p - \left(\frac{\Delta}{p}\right)$. Therefore, $p+1 \geqslant p - \left(\frac{\Delta}{p}\right) \geqslant F > \sqrt{n}+1$. So $p > \sqrt{n}$, for every prime factor of n, which means n is prime. □

To turn Theorem 5.12 into a primality test, we first have to find suitable numbers a, b. Given a prime number n, we may choose a, b at random such that the conditions of Theorem 5.12 are fulfilled. Indeed, the expected number of choices we have to make is not that large [CP05].

Theorem 5.13 (Primality Test for Mersenne prime numbers). *Let* $a = 2$, $b = -2$ *and let* U, V *be the related Lucas sequences with discriminant* $\Delta = 12$. *Then* $n = M_p = 2^p - 1 > 3$ *is prime if and only if* $V_{\frac{n+1}{2}} \equiv 0 \pmod{n}$.

Proof. Let $n > 3$ be prime. By Lemma 5.2 we have

$$\begin{aligned} V^2_{\frac{n+1}{2}} &= V_{n+1} + 2(-2)^{\frac{n+1}{2}} = V_{n+1} - 4(-2)^{\frac{n-1}{2}} \\ &\overset{\text{by } 2.10}{\equiv} V_{n+1} - 4\left(\frac{-2}{n}\right) \pmod{n}. \end{aligned}$$

With $\left(\frac{-2}{n}\right) = \left(\frac{-1}{n}\right)\left(\frac{2}{n}\right) = -1$ (this follows by Theorem 2.11) we get

$$V^2_{\frac{n+1}{2}} \equiv V_{n+1} + 4 \pmod{n}.$$

We will show, that $V_{n+1} \equiv -4 \pmod{n}$. By Lemma 5.3 we have

$$\begin{aligned} 2V_{n+1} &= V_n V_1 + \Delta U_n U_1 \\ 2V_{n+1} &= 2V_n + 12U_n \\ V_{n+1} &= V_n + 6U_n. \end{aligned}$$

Thus, with (5.1) and (5.2) we get

$$\begin{aligned} V_{n+1} &\equiv a + 6\left(\frac{12}{n}\right) \\ &\equiv 2 + 6\left(\frac{2}{n}\right)\left(\frac{2}{n}\right)\left(\frac{3}{n}\right) \\ &\equiv 2 - 6 \equiv -4 \pmod{n}. \end{aligned}$$

Now, let $V_{\frac{n+1}{2}} \equiv 0 \pmod{n}$. By Lemma 5.1 U_{n+1} is also divisible by n. With Lemma 5.5 we get

$$V^2_{\frac{n+1}{2}} - 12U_{\frac{n+1}{2}} = 4(-2)^{\frac{n+1}{2}}.$$

Therefore, $\gcd(U_{\frac{n+1}{2}}, n) = 1$. And since $\gcd(2, n) = 1$, $\left(\frac{12}{n}\right) = -1$ and since 2 is the only prime factor of $n + 1 = 2^p$, n must be prime by Theorem 5.12. $\square$

For practical reasons, instead of the Lucas sequence V we now consider the recursive sequence $S = (S_0, S_1, \dots)$, which is defined as follows:

$$S_0 = 4, \quad S_{k+1} = S_k^2 - 2.$$

In this way, we get the promised *Lucas-Lehmer-Test* for Mersenne prime numbers:

Theorem 5.14 (Lucas-Lehmer-Test). $M_p = 2^p - 1$ *is prime if and only if* $S_{p-2} \equiv 0 \pmod{M_p}$.

Proof. First, we will show by induction that $S_k = \frac{V_{2^{k+1}}}{2^{2^k}}$. $S_0 = 4 = \frac{V-2}{2}$, so suppose, that $S_{k-1} = \frac{V_{2^k}}{2^{2^{k-1}}}$ holds. Then

$$S_k = S_{k-1}^2 - 2 = \frac{V_{2^k}^2}{2^{2^k}} - 2 \overset{\text{by Lemma 5.2}}{=} \frac{V_{2^{k+1}} + 2 \cdot 2^{2^k} - 2 \cdot 2^{2^k}}{2^{2^k}} = \frac{V_{2^{k+1}}}{2^{2^k}}.$$

By Theorem 5.13, M_p is prime, if and only if it divides $V_{\frac{M_p+1}{2}}$. And since

$$V_{\frac{M_p+1}{2}} = V_{2^{p-1}} = 2^{2^{p-1}} S_{p-1},$$

M_p is prime, if and only if it divides S_{p-2}. $\square$

5.2.1. Time Complexity Analysis - Lucas-Lehmer

We now analyze the Lucas-Lehmer primality test. If we perform the reduction $\pmod{M_p}$ at each iteration, we ensure that the intermediate results do not get bigger than the length of the input. The most expensive operation of each iteration is the squaring. The number of iterations is $\mathcal{O}(\log M_p) = \mathcal{O}(p)$. By Lemma 4.2 and since the length of all intermediate results is at most $\log M_p$, each squaring takes time $\mathcal{O}(\log^2 M_p) = \mathcal{O}(p^2)$. Therefore, the total running time of the algorithm is $\mathcal{O}(\log^3 M_p) = \mathcal{O}(p^3)$.

Using the Schönhage-Strassen algorithm, we get an improved time complexity of $\hat{\mathcal{O}}(\log^2 M_p) = \hat{\mathcal{O}}(p^2)$.

The Lucas-Lehmer test is therefore an efficient deterministic polynomial-time algorithm for Mersenne numbers.

Algorithm 4 Lucas-Lehmer Test

```
Input: positive integer $M_p = 2^p - 1$ to be tested
Output: COMPOSITE if $M_p$ is composite, otherwise PRIME
  Let $S_{p-2} = S_{p-3}^2 - 2$ with $S_0 = 4$.
  if $S_{p-2} \equiv 0 \pmod{M_p}$ then
    return PRIME
  end if
  return COMPOSITE
```

5.2.2. GIMPS

The Lucas-Lehmer test is the basis of the *Great Internet Mersenne Prime Search* (GIMPS). GIMPS is a collaborative project of volunteers who use freely available software to search for big Mersenne prime numbers. It was founded in 1996 by computer scientist George Woltman. The project has found fifteen Mersenne prime numbers as of October 2016, most of them were the greatest known primes at their respective times of discovery. The largest known prime as of October 2016, $2^{74207281} - 1$, was discovered on September 17, 2015 (but actually not noticed until January 2016) by US-American mathematician Curtis Niles Cooper using a GIMPS software. It has 22 228 618 decimal digits.

6. The Elliptic Curve Primality Test

One of the most powerful primality tests that is also most widely used in practice is the elliptic curve primality test. In 1985, H. W. Lenstra developed a concept of using elliptic curves in factorization. The use of elliptic curves in primality testing algorithms was an implication of his work. The elliptic curve primality test was first described by Shafrira Goldwasser and Joe Kilian in 1986 and then turned into an effective algorithm by A.O.L. Atkin in the same year. The algorithm was improved notably by A. O. L. Atkin and Francois Morain in 1993.

6.1. Motivation[1]

We will first consider a generalization of the idea of Lucas. It is the motivation for a theorem, on which the Elliptic Curve Primality Test is based on.

Theorem 6.1. *Let $n \in \mathbb{N}, n > 1$. Suppose that there is an element $a \in \mathbb{Z}_n$ and a number $s > 0$ such that*

$$\begin{aligned} a^s &\equiv 1 \pmod n \\ a^{\frac{s}{q}} - 1 &\in \mathbb{Z}_n^*, \quad \textit{for every prime factor } q \textit{ of } s. \end{aligned}$$

Then $p \equiv 1 \pmod s$ for any prime p dividing n. In particular: if $s \geqslant \sqrt{n} \Rightarrow n$ is prime.

Proof. Let p be prime with $p \mid n$. Then we have:

$$\begin{aligned} a^s &\equiv 1 \pmod p \\ a^{\frac{s}{q}} &\not\equiv 1 \pmod p \quad \text{for every prime factor } q \text{ of } s, \end{aligned}$$

since $a^{\frac{s}{q}} - 1 \in \mathbb{Z}_n^* \Rightarrow a^{\frac{s}{q}} - 1 \in \mathbb{Z}_p^* \Rightarrow a^{\frac{s}{q}} - 1 \not\equiv 0 \pmod p$. Therefore, $s = \text{ord}_p(a) \mid |\mathbb{Z}_p^*| = p - 1$, which means $p \equiv 1 \mod s$. For the second statement, suppose that n is composite and choose a prime $p \leqslant \sqrt{n}$. Let $s \geqslant \sqrt{n}$. We then have:

$$p - 1 = k \cdot s \geqslant k\sqrt{n} \geqslant \sqrt{n} \Rightarrow p - 1 \geqslant \sqrt{n} \Rightarrow p > \sqrt{n}.$$

Therefore, n must be prime. $\square$

In order to test if $a^{\frac{s}{q}} - 1 \in \mathbb{Z}_n^*$ for every prime factor q of n, one has to check if $\gcd(n, a^{\frac{s}{q}} - 1)$. So first we need to know all prime factors q of s. But s needs to be large. In fact, it has to be larger than $\sqrt{n}$ to conclude that n is prime. In applications, s is a completely factored divisor of $n - 1$. Computing such a divisor takes usually much

[1]The motivation is based on [Sch14]

time, if n is large. So in general this theorem won't prove directly, if n is prime.
But it may happen, that one can compute a divisor $r > 1$ of $n-1$ that has the property that $s = \frac{n-1}{r}$ is *probably* prime. This would make our problem much easyer: one chooses a random $x \in \mathbb{Z}_n$ and computes $a = x^r$. With high probability we have $a^s \equiv 1 \pmod{n}$ and $a - 1 \in \mathbb{Z}_n^*$. Since $s > \sqrt{n}$, Theorem 6.1 implies now that n is prime if the smaller number s is prime. But don't get too excited: the chance that $n-1$ factors this way is very low (on average $\mathcal{O}(\frac{1}{\log n})$). As we will see, elliptic curves will give a solution to this problem.

6.2. Preliminaries

In this section we will give some basic definitions and facts related to elliptic curves. While there are whole books about the theory of elliptic curves we will concentrate on the very basics which is enough for our purpose. Advanced readers might want to skip this section and continue reading in section 6.3. To define elliptic curves we shortly introduce the concept of affine and projective planes:

6.2.1. Affine and Projective Plane

First remember how an affine plane is defined:

Definition. Let $\mathbb{A}$ be a set of points and $\mathbb{G} \subset \mathcal{P}(\mathbb{A})$, where $\mathcal{P}(\mathbb{A})$ denotes the power set of A, be a set of lines. $(\mathbb{A}, \mathbb{G})$ is called *affine plane* if:

(A1) For any two elements $p, q \in \mathbb{A}, p \neq q$ there exists exactly one $L \in \mathbb{G}$ such that $p, q \in L$

(A2) For $L \in \mathbb{G}$ and $p \in \mathbb{A} \backslash L$ there exists exactly one $L' \in \mathbb{G}$ with $p \in L'$ and $L \cap L' = \emptyset$. (Parallel postulate)

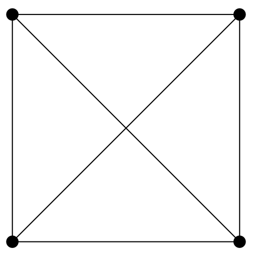
(A3) There are $p, q, r \in \mathbb{A}$ and $L \in \mathbb{G}$ with $p, q \in L$ but $r \notin L$.

Lemma 6.2. *Let K^2 be the two dimensional vector space over the field K. For $\mathbb{A} := K^2$ and $\mathbb{G} := \{a + Kb \mid a, b \in K, b \neq 0\}$, $\mathbb{A}^2(K) := (\mathbb{A}, \mathbb{G})$ is an affine plane.*

Remark. For $K = \mathbb{R}$, $\mathbb{A}$ and $\mathbb{G}$ are points and lines in the usual interpretation.

The smallest affine plane is the following:

Example. Let $\mathbb{A} = \{p, q, r, s\}$ and $\mathbb{G} = \{\{p, q\}, \{p, r\}, \{p, s\}, \{q, r\}, \{q, s\}, \{r, s\}\}$. So any two different lines are either parallel or intersect in exactly one point.



Although the concept of an affine plane is nice and clear, we don't like the fact that there are lines which do not intersect each other. The projective plane solves this problem:

Definition. Let $\mathbb{P}$ be a set of points and $\mathbb{G} \subset \mathcal{P}(\mathbb{P})$ be a set of lines. $(\mathbb{P}, \mathbb{G})$ is called *projective plane* if:

(A1) For any two points $P, Q \in \mathbb{P}$ with $P \neq Q$ there exists exactly one $L \in \mathbb{G}$ such that $P, Q \in L$

(P2) For any two lines $L_1, L_2 \in \mathbb{G}$, $|L_1 \cap L_2| = 1$.

(P3) There are $P, Q, R, S \in \mathbb{P}$ such that any three of them do not belong to the same line $L \in \mathbb{G}$.

So in a projective plane any two lines intersect in exactly one point. There are no non-identical parallel lines anymore. To avoid confusion, we usually denote the coordinates of a point in the projective plane with capital letters and those in the affine plane with small letters.
Now, let K^3 be the three dimensional vector space over K. We will define an *equivalence relation* for points $P, Q \in K^3 \backslash \{0\}$:

$$P \sim Q :\Leftrightarrow \exists \lambda \in K \backslash \{0\} \quad \text{with} \quad \lambda P = Q.$$

For the equivalence classes of $a = (a_1, a_2, a_3) \in K^3 \backslash \{0\}$ we write $(a_1 : a_2 : a_3)$ or for simplicity reasons $[a]$. On these equivalence classes, we now define a set of points:

$$\mathbb{P} := (K^3 \backslash \{0\})/_\sim = \{[a] \mid a \in K^3 \backslash \{0\}\}.$$

For $P \neq Q \in \mathbb{P}, P = [a], Q = [b]$, the line connecting the points is defined by:

$$\overline{PQ} := \{[\lambda a + \mu b] \mid \lambda, \mu \in K, (\lambda, \mu) \neq (0,0)\}.$$

Then the set of all lines is

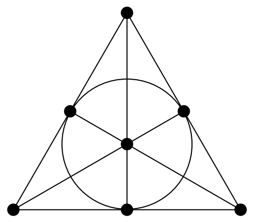
$$\mathbb{G} := \{\overline{PQ} \mid P, Q \in \mathbb{P}, P \neq Q\}$$

and we get:

Lemma 6.3. *For $\mathbb{P}$ and $\mathbb{G}$ definded as above, $\mathbb{P}^2(K) := (\mathbb{P}, \mathbb{G})$ is a projective plane.*

The smallest projective plane is the so called *Fano plane*:

Example (Fano plane). $|\mathbb{P}| = 7$.



For $Z \neq 0$ we can transform every point of the project plane as follows:

$$(X : Y : Z) = (\frac{X}{Z} : \frac{Y}{Z} : 1) = (x : y : 1).$$

We can now identify the point $(x : y : 1)$ of the project plane with (x, y) in the affine plane. To include the points where $Z = 0$ we define the so called *points at infinity*:

$$U := \{(U : V : W) \in \mathbb{P} \mid W = 0\}.$$

We then get the projective plane by adding the points at infinity to the affine plane:

$$\mathbb{P}^2(K) = \mathbb{A}^2(K) \cup U.$$

6.2.2. Elliptic Curves[2]

The field of elliptic curves is huge. There are lots of good books devoted to elliptic curves. We will focus on the very basics, just enough for applying them to primality testing. For further reading see [Mil06], for example.

Before we start, note that elliptic curves are (as we will see) not ellipses. See [Mil06] for their (sparse) connection.

Consider the following polynomial:

$$F(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3.$$

F is a homogenous polynomial (which is a polynomial whose nonzero terms all have the same degree) of degree 3. The strange numbering of the coefficients has historical reasons.

Definition. The equation $F(X, Y, Z) = 0$ is called *Weierstrass equation*.

Let

$$E := \{(U : V : W) \in \mathbb{P} \mid F(U, V, W) = 0\}$$

be the zero set of F. E defines a curve in the projective plane $\mathbb{P}^2(K)$.

Definition. A point $P = (U : V : W) \in E$ is called *singular*, if the partial derivatives of F at P vanish:

$$\frac{\partial F}{\partial X}(P) = \frac{\partial F}{\partial Y}(P) = \frac{\partial F}{\partial Z}(P) = 0.$$

A curve E is called singular if there exist a singular point in E and non-singular, if not.

We can now give a definition of an elliptic curve:

Definition. A non-singular curve, described by a Weierstrass equation, is called an *elliptic curve*.

[2]The main reference for this section is [HPS08]

Remark. Let E be an elliptic curve. Then E has exactly one point at infinity, namely: $\mathcal{O} := (0 : 1 : 0)$. (Simply substitute $Z = 0$ in the equation...)

In many cases, and in particular all the cases we will consider, we can simplify the Weierstrass equation:

Lemma 6.4. *Let E be an elliptic curve over K. If $\mathrm{char}(K) \neq 2, 3$ then E is isomorphic to an elliptic curve E' of the form:*

$$E' : 0 = Y^2Z - X^3 - aXZ^2 - bZ^3$$

and the equation

$$0 = Y^2Z - X^3 - aXZ^2 - bZ^3 \tag{6.1}$$

is called short Weierstrass equation.

By dividing equation (6.1) by Z^3 and replacing X by $\frac{X}{Z}$ and Y by $\frac{Y}{Z}$, we get the short Weierstrass equation in the affine plane:

$$0 = y^2 - x^3 - ax - b.$$

Since $\mathcal{O}$ is the only point at infinity on E and all other points can be transformed into points of the form $P = (U : V : 1)$, we get the affine representation of E:

$$E = \{(x, y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \mathcal{O}. \tag{6.2}$$

But how can we ensure, that our curve is non-singular? The following lemma will give the answer to this question:

Lemma 6.5. *Let E be a curve given by (6.2). E is non-singular, and therefore an elliptic curve, if and only if*

$$4a^3 + 27b^2 \neq 0$$

Proof. Let $f(x, y) = y^2 - x^3 - ax - b$. We then have the partial derivatives $f_x = -3x^2 - a$ and $f_y = 2y$, where $f_x = 0 \Leftrightarrow x = \sqrt{-\frac{a}{3}}$ and $f_y = 0 \Leftrightarrow y = 0$. If we plug these in the short Weierstrass equation, we get:

$$\begin{aligned} 0 &= \left(\sqrt{-\frac{a}{3}}\right)^3 + a \cdot \sqrt{-\frac{a}{3}} + b \\ \Leftrightarrow 4a^3 + 27b^2 &= 0 \end{aligned}$$

So for E being non-singular, $4a^3 + 27b^2 \neq 0$ must hold and this is also sufficient. □

Remark. If $4a^3 + 27b^2 > 0$, E has only one component, if $4a^3 + 27b^2 < 0$, it has two.

See figures 6.1 and 6.2 for examples of elliptic curves. And figure 6.3 for examples of singular curves.

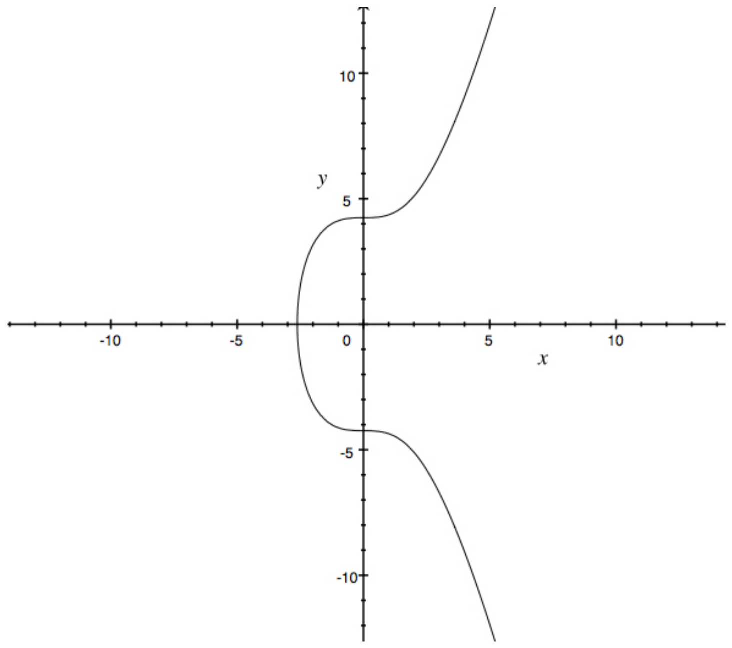


Figure 6.1.: An elliptic curve with one component: $y^2 = x^3 + 18$

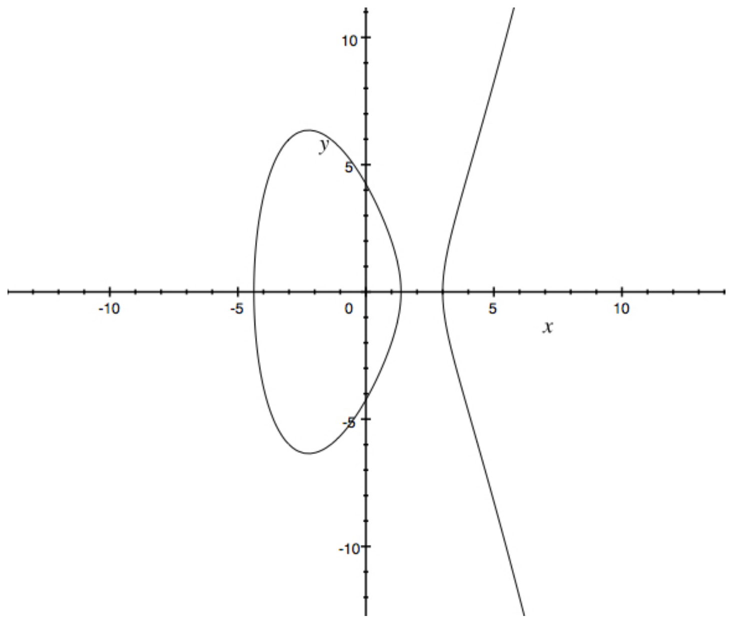


Figure 6.2.: An elliptic curve with two components: $y^2 = x^3 - 15x + 18$

From now on, we assume that E is given in its affine representation and that Lemma 6.5 holds, so we have indeed an elliptic curve.

Elliptic curves have the great property that one can take any two points on the curve and add them together to get a third point. We will now define, what we mean by "adding":

Addition Law. Let E be an elliptic curve. The *addition law* is then defined by:

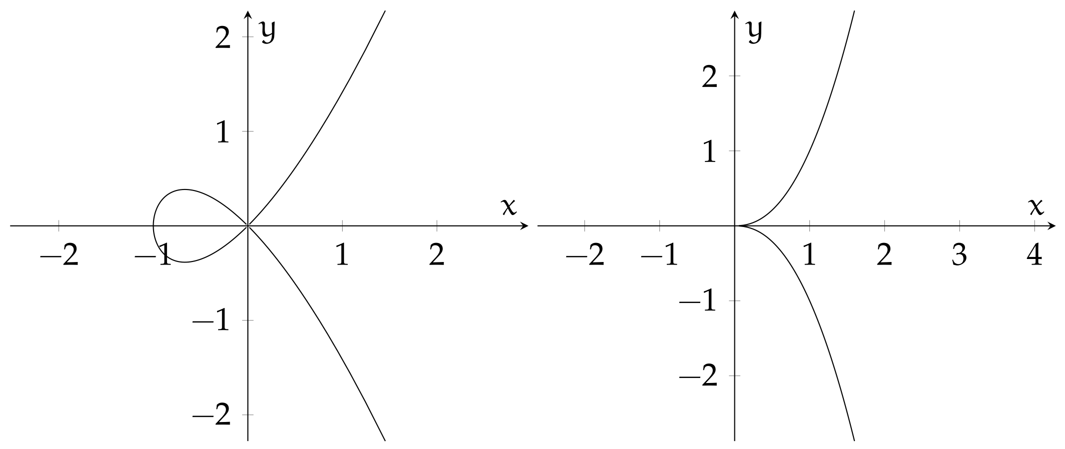


Figure 6.3.: Two singular curves, one with self intersection, one with a cusp

- Let P and Q be two points on E, $P \neq Q$ and let L be the line connecting them. L intersects E in three points: P, Q and a third point which we call $R = (s,t)$. We take R and reflect it across the x-axis. So we get:

$$P + Q := R' = (s, -t).$$

- Let $Q = P'$. In that case, L is the vertical line through P and it intersects E in three points: P, P' and in the point at infinity $\mathcal{O}$. With this we get:

$$P + P' := \mathcal{O}.$$

- If $Q = P$, then L is the tangent line to E at P. Suppose that L intersects E in P and in a different point $Q \neq P$. Then

$$P + P := -Q$$

- $P + \mathcal{O} := P$, since the line connecting P and $\mathcal{O}$ is vertical it intersects E in $P, \mathcal{O}$ and P'. Reflecting P' gets us back to P

Note that the reflection point is always in E, since elliptic curves are symmetric about the x-axis.
To make this addition law clear, let's do an example:

Example. Let $E : y^2 = x^3 - 15x + 18$ be an elliptic curve and $P = (7,16), Q = (1,2)$ be two points on E. To determine $P + Q$, we first have to find the line $L : y = kx + d$ connecting them, so we have to find k, d. Computation leads to $L : y = \frac{7}{3}x - \frac{1}{3}$. Now we have to find the third intersection point of E and L:

$$\left(\frac{7}{3}x - \frac{1}{3}\right)^2 = x^3 - 15x + 18 \tag{6.3}$$

$$\Leftrightarrow x^3 - \frac{49}{9}x^2 - \frac{121}{9}x + \frac{161}{9} = 0 \tag{6.4}$$

In general, finding the roots of a cubic polynomial is not that easy, but in this case we already know two of its roots: $x = 7$ and $x = 1$. Therefore the polynomial in equation (6.4) factors into $(x-7)(x-1)(x+\frac{23}{9})$. So we found the x-coordinate of the third point, which is $x = -\frac{23}{9}$, and by substituting this into L we also get the y-coordinate: $y = -\frac{170}{27}$. By reflecting the point $(-\frac{23}{9}, -\frac{170}{27})$ across the x-axis, the get:

$$P + Q = (-\frac{23}{9}, \frac{170}{27}).$$

Note that this is indeed a point in E!

Notations. If $P = (s, t)$ we denote the reflection point by $P' = (s, -t)$. If we add P n times to itself, we denote it by nP.

Theorem 6.6. *Let E be an elliptic curve. Then the addition law gives the points of E the structure of an abelian group:*

(a) $P + \mathcal{O} = \mathcal{O} + P = P$ *for all* $P \in E$ *(Identity)*

(b) $P + P' = \mathcal{O}$ *for all* $P \in E$ *(Inverse)*

(c) $(P + Q) + R = P + (Q + R)$ *for all* $P, Q, R \in E$ *(Associative)*

(d) $P + Q = Q + P$ *for all* $P, Q \in E$ *(Commutative)*

Proof. (a) and (b) are clear.
(d) is also clear, since every line through P, Q is the same as the line through Q, P.
The associative law (c) is harder to prove. One can find it in [Mil06] for example. □

Algorithm 5 computes the sum of two points on an elliptic curve.

Algorithm 5 Elliptic Curve Addition Algorithm

Input: An elliptic curve $E(\mathbb{F}_p) : y^2 = x^3 + ax + b$ and two points $P, Q \in E(\mathbb{F}_p)$
Output: The sum $P + Q$

```
1: if $P = \mathbb{O}$ then
2:     $P + Q = Q$
3: end if
4: if $Q = \mathbb{O}$ then
5:     $P + Q = P$
6: end if
7: Write $P = (x_1, y_1)$, $Q = (x_2, y_2)$.
8: if $x_1 = x_2$ and $y_1 = -y_2$ then
9:     $P + Q = \mathbb{O}$
10: end if
11: Define $\lambda$ by
12: $\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$
13: and let $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$
14: then $P + Q = (x_3, y_3)$
```

Proof of the correctness of the algorithm. The steps 1, 4 and 8 obviously compute the correct sum.
We show, that step 11 also leads to the correct point on $E(\mathbb{F}_p)$.
If $P \neq Q$, then λ is the slope of the line through P and Q. We have

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

If $P = Q$, then λ is the slope of the tangent line at P. By differentiating implicitly we get

$$2y\frac{dy}{dx} = 3x^2 + a \Leftrightarrow \frac{dy}{dx} = \frac{3x^2 + a}{2y}.$$

Substituting P yields

$$\lambda = \frac{3x_1^2 + a}{2y_1}.$$

In both cases, the line is then given by

$$L : y = \lambda x + \nu \quad , \quad \nu = y_1 - \lambda x_1.$$

Substituting this into $E(\mathbb{F}_p)$ yields:

$$(\lambda x + \nu)^2 = x^3 + ax + b$$

$$\Leftrightarrow x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + (b - \nu^2) = 0. \tag{6.5}$$

This cubic equation has roots x_1 and x_2. Let x_3 denote the third root, then 6.5 factors into

$$x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + (b - \nu^2) = (x - x_1)(x - x_2)(x - x_3).$$

By equating coefficients, we get:

$$\lambda^2 = x_1 + x_2 + x_3 \Leftrightarrow x_3 = \lambda^2 - x_1 - x_2.$$

Substituting x_3 into L yields $y_3' = \lambda(x_3 - x_1) + y_1$. By reflecting across the x-axis we get $y_3 = \lambda(x_1 - x_3) - y_1$ and we have

$$P + Q = (x_3, y_3).$$

□

Algorithm 5 shows, that it is necessary to have a non-singular curve, since otherwise we would not have unique tangent lines in every point and the addition would not be defined for all points on the elliptic curve.

For our purpose we will consider elliptic curves over finite fields:

Definition. Let $a, b \in \mathbb{F}_p$. An *elliptic curve over the finite field* $\mathbb{F}_p$ is then given by:

$$\begin{aligned} E_{a,b}(\mathbb{F}_p) \ := \ & \{(x, y) \mid x, y \in \mathbb{F}_p \text{with } y^2 \equiv x^3 + ax + b \pmod p \\ & \text{with } 4a^3 + 27b^2 \not\equiv 0 \pmod p)\} \cup \mathcal{O}. \end{aligned}$$

Notations. $|E_{a,b}(\mathbb{F}_p)|$ denotes the number of points of $E_{a,b}(\mathbb{F}_p)$.

Note that this is not a continuous curve anymore, but a discrete set of points in the xy-plane, as the following example will show:

Example. Let $E_{3,8} : y^2 \equiv x^3 + 3x + 8$ over $\mathbb{F}_{13}$. To find the points of $E_{3,8}(\mathbb{F}_{13})$ we substitute all possible values $x = 0, \dots, 12$ into the equation and check if the result is a square modulo 13:

$$x = 0 \Rightarrow y^2 = 8 \quad \text{but 8 is not a square modulo 13, so we discard } x$$

$$x = 1 \Rightarrow y^2 = 12 \quad \text{we get two square roots modulo 13:}$$
$$5^2 \equiv 12 \pmod{13} \quad \text{and} \quad 8^2 \equiv 12 \pmod{13}$$

which gives us two points in $E_{3,8}(\mathbb{F}_{13})$: $(1,5)$ and $(1,8)$. By repeating the same procedure for all possible values x, we get:

$$E_{3,8}(\mathbb{F}_{13}) = \{\mathcal{O}, (1,5), (1,8), (2,3), (2,10), (9,6), (9,7), (12,2), (12,11)\}$$

and $|E_{3,8}(\mathbb{F}_{13})| = 9$.

Remark. The addition law still gives $E_{a,b}(\mathbb{F}_p)$ the structure of an abelian group. And the elliptic curve addition applied to two points of $E_{a,b}(\mathbb{F}_p)$ still yields a point in $E_{a,b}(\mathbb{F}_p)$.

Note, that the elliptic curve addition algorithm involves division. So if n is composite, $\mathbb{Z}_n$ is not a finite field anymore, but a ring and we have zero divisors. Therefore, there are points P, Q such that $P + Q$ is not defined. Hence, in this case the points of $E_{a,b}(\mathbb{Z}_n)$ will not form a group anymore. Since we often do not know, if n is prime or not, we call a curve over the ring $\mathbb{Z}_n$ an *elliptic pseudocurve*:

Definition. For $a, b \in \mathbb{Z}_n$ with $\gcd(4a^3 + 27b^2, n) = 1$ and $\gcd(n, 6) = 1$, an *elliptic pseudocurve over the ring* $\mathbb{Z}_n$ is the set

$$E_{a,b}(\mathbb{Z}_n) = \{(x, y) \mid x, y \in \mathbb{Z}_n : y^2 = x^3 + ax + b\} \cup \mathcal{O}.$$

Note, that every elliptic curve over $\mathbb{F}_p$ is also an elliptic pseudocurve.

In order to use elliptic curves for primality tests, we need an estimate for the number of points on an elliptic curve over a finite field.

Theorem 6.7 (Hasse's Theorem)**.** *Let $E_{a,b}$ be an elliptic curve over the finite field $\mathbb{F}_p$. Then:*

$$\big| |E_{a,b}(\mathbb{F}_p)| - (p+1) \big| \leqslant 2\sqrt{p}.$$

6.3. Goldwasser-Kilian Primality Test[3]

We can finally apply Theorem 6.1 from the beginning of this chapter to elliptic curves:

[3]The main references for this section are [Sch14] and [CP05]

Theorem 6.8. *Let $n \in \mathbb{N}, n > 1$ and let $E_{a,b}(\mathbb{Z}_n)$ be an elliptic pseudocurve. Suppose that there is a point $P \in E_{a,b}(\mathbb{Z}_n)$ and an integer $s > 0$, such that:*

$$\begin{aligned} sP &= \mathcal{O} \quad \textit{in } E_{a,b}(\mathbb{Z}_n) \\ \frac{s}{q} &\neq \mathcal{O} \quad \textit{in } E_{a,b}(\mathbb{Z}_p) \end{aligned}$$

for any prime factors q of s and any prime factors p of n. Then:

$$|E_{a,b}(\mathbb{Z}_p)| \equiv 0 \pmod{s}$$

for every prime p dividing n. In particular, if $s > (\sqrt[4]{n}+1)^2$, then n is prime.

Proof. Let p be any prime divisor of n. Since $sP = \mathcal{O}$ in $E_{a,b}(\mathbb{Z}_n)$, the order of P in $E_{a,b}(\mathbb{Z}_p)$ is also s. Therefore $s \mid |E_{a,b}(\mathbb{F}_p)|$ and $|E_{a,b}(\mathbb{F}_p)| \equiv 0 \pmod{s}$. By Hasse's Theorem we know that $|E_{a,b}(\mathbb{F}_p)| \leqslant (\sqrt{p}+1)^2$. Therefore, if $s > (\sqrt[4]{n}+1)^2$, we have:

$$(\sqrt{p}+1)^2 \geqslant |E_{a,b}(\mathbb{F}_p)| \geqslant s > (\sqrt[4]{n}+1)^2 \Rightarrow p > \sqrt{n}$$

As before, if n is composite it must have a prime factor $p \leqslant \sqrt{n}$, since this is not the case, n must be prime. □

Theorem 6.8 is due to Shafrira Goldwasser and Joe Kilian.

Now the question is how do we get a primality test out of Theorem 6.8? The idea of Goldwasser and Kilian is the following:

Suppose n is prime. Then, for any given a, b, the order of $E_{a,b}(\mathbb{Z}_n)$ lies in the Hasse interval $((\sqrt{n}-1)^2, (\sqrt{n}+1)^2)$ by Theorem 6.7. To apply Theorem 6.8 we have to find a number $s > (\sqrt[4]{n}+1)^2$, which is a completely factored divisor of $|E_{a,b}(\mathbb{Z}_n)|$. As we already noticed, in generak it will not be easy to find such a number s. But now, we are dealing with elliptic curves, which gives us more possibilities:
We choose a, b at random, with $\gcd(4a^3 + 27b^2, n) = 1$ and determine $|E_{a,b}(\mathbb{Z}_n)|$. So we first have to know, if there is an algorithm, that quickly determines the order of $E_{a,b}(\mathbb{Z}_n)$, provided n is prime. In fact, such an algorithm exists:
In 1985, the Dutch mathematician René Schoof published a deterministic polynomial-time algorithm to count the number of points on elliptic curves over finite fields [Sch85]. Until then, only algorithms with exponential running time were known. Therefore, Schoof's algorithm was a great discovery.

So now that we know that determining the order will not be a problem, we try to find a completely factored divisor s of this order with $s > (\sqrt[4]{n}+1)^2$. If trying to factor the order of the curve takes too long, we just choose another pair a, b and try again with those.
Suppose, that we were lucky and we found suitable numbers a, b and s. We now have to find a point P on $E_{a,b}(\mathbb{Z}_n)$ which satisfies the conditions of Theorem 6.8. For thhis purpose, we simply choose an integer x at random and determine $x^3 + ax + b$. The probability that this number is a quadratic residue modulo n is approximately $1/2$.

Then, finding its square root modulo n also will not be a big challenge. There are several algorithms to provide it.
Also, finding a point on $E_{a,b}(\mathbb{Z}_n)$ whose order is a multiple of s is not too difficult. The probability of finding such a point is large enough and if we found one, getting a multiple of that point whose order is exactly s is simple.

So if we find a large and completely factored divisor s of $|E_{a,b}(\mathbb{Z}_n)|$, deciding whether n is prime or composite is ensured. We therefore have to think about what kind of numbers are easy to factor. Well, prime numbers might be able to do the job, since once you know you have a prime number, it obviously is completely factored.
So as with Theorem 6.1, instead of proving that n is prime, we try to show that a smaller number s is prime. Concretely we proceed as follows:

Given a number n that is likely to be prime, we choose randomly elliptic (pseudo)curves $E_{a,b}(\mathbb{F}_n)$ and compute the order $|E_{a,b}(\mathbb{F}_n)|$ until we find one with $|E_{a,b}(\mathbb{F}_n)| = r \cdot s$, where s is probably prime with $s > (\sqrt[4]{n}+1)^2$. Now randomly choose a point $Q \in E_{a,b}(\mathbb{F}_n)$ and compute $P = rQ$. Then check, if $sP = \mathcal{O}$ in $E_{a,b}(\mathbb{F}_n)$ and that $P \neq \mathcal{O}$ in $E_{a,b}(\mathbb{F}_p)$ for every prime divisor p of n (If we take the projective notation $P = (U : V : W)$, the last equation simply means, that $\gcd(n, W) = 1$). Theorem 6.8 now ensures that n is prime if s is prime. In order to prove that s is prime, one might have to apply the method again, until one is able to prove the primality of some divisor by trial division.

Recall, that if n is not prime, $E_{a,b}(\mathbb{F}_n)$ is not an elliptic curve but an elliptic pseudocurve and there are zero divisors. In practice, this is not a problem, since whenever zero divisors occur, we know for sure that n is composite.
Algorithm 6 summarizes the details of the Goldwasser-Kilian Primality Test.

Algorithm 6 Goldwasser-Kilian Primality Test

```
Input: positive nonsquare integer $n$, that has already passed a probabilistic primality test
Output: COMPOSITE if $n$ is composite, otherwise PRIME IF $s$ IS PRIME
1: [Choose an elliptic pseudocurve over $\mathbb{Z}_n$]
   Choose random $a, b \in [0, n-1]$ such that $\gcd(4a^3 + 27b^2, n) = 1$
2: [Assess curve order]
   Calculating $|E_{a,b}(\mathbb{F}_n)| =: o$ as if $n$ were prime using Algorithm 8.
3: if the point-counting algorithm fails then
4:     return COMPOSITE
5: end if
6: [Attempt to factor]
   Attempt to factor $o = r \cdot s$ where $r > 1$ and $s$ probably prime with $s > (\sqrt[4]{n} + 1)^2$.
7: if the factoring cannot be done according to some time-limit criterion then
8:     goto 1
9: end if
10: [Choose point on $E_{a,b}(\mathbb{Z}_n)$]
   Choose random $x \in [0, n-1]$ such that $t \equiv x^3 + ax + b \pmod{n}$ has $\left(\frac{t}{n}\right) \neq -1$
   Find an integer $y$ such that $y^2 \equiv t \pmod{n}$ (under the assumption that $n$ is prime).
11: if $y^2 \not\equiv t \pmod{n}$ then
12:     return COMPOSITE
13: end if
14: $Q = (x, y)$
15: [Operate on point]
   Compute the multiple $P = rQ$
16: if zero divisors occur then
17:     return COMPOSITE
18: end if
19: if $P = \mathcal{O}$ then
20:     goto 10
21: end if
22: Compute $R = sP$
23: if zero divisors occur then
24:     return COMPOSITE
25: end if
26: if $R \neq \mathcal{O}$ then
27:     return COMPOSITE
28: end if
29: return PRIME IF $s$ IS PRIME
```

But the Goldwasser-Kilian Primality Test is not only able to decide whether a number is prime or not, it also provides a very short proof of primality, a so called *certificate of primality*. That means, once proven that a certain number is prime, one can quickly verify this result by using the certificate. In the case of the Goldwasser-Kilian test, the certificate consists of a chain

$$(n = n_1, a_1, b_1, o_1, r_1, s_1, Q_1), (s_1 = n_2, a_2, b_2, o_2, r_2, s_2, Q_2),$$

$$\ldots, (s_{i-1} = n_i, a_i, b_i, o_i, r_i, s_i, Q_i),$$

where s_i can be proven prime by trial division.

6.3.1. Time Complexity Analysis

Whereas the time complexity analysis of the primality tests in chapter 3 was quite straight forward, the time complexity analysis of the Goldwasser-Kilian Primality Test is more complicated. This is basically because we do not really know how prime numbers are distributed in certain intervals. Therefore, we do not know how long it will take until we find a curve whose order we are able to factor in a limited time.
However, there are some conjectures related to this:

It can be shown that if

$$\pi((\sqrt{x}+1)^2) - \pi((\sqrt{x}-1)^2) > A\frac{\sqrt{x}}{\ln^c x}$$

for positive constants A, c, then the expected time complexity of the algorithm is $\mathcal{O}(\ln^{9+c} n)$ (see [GK99]).

The most time consuming step is computing the curve order via Schoof's algorithm. Schoof's algorithm takes time $\mathcal{O}(\ln^8 n)$ (see [GK99]). The algorithm can be improved such that one gets a time complexity of $\mathcal{O}(\ln^6 n)$ (see [CP05]).

Therefore the Goldwasser-Kilian Primality Test has expected polynomial running time for almost all inputs.

6.4. Atkin-Morain Primality Test

In 1988, Arthur Oliver Lonsdale Atkin and François Morain were able to transform the Goldwasser-Kilian Primality Test into a computer practical test. The problem with the Goldwasser-Kilian Primality Test is determining the curve order with Schoof's Algorithm. Even though it is a deterministic, polynomial-time algorithm, it is not very practical to apply it repeatedly with very large prime numbers.

By using the theory of *complex multiplication* on elliptic curves, the method of Atkin and Morain first finds the order, and if one can quickly find a completely factored s as in Theorem 6.8 then finding a corresponding curve is not so hard (see [AM93] for the details).

The method seems to work very well in practice. The largest numbers that were proven prime by the Atkin-Morain Primality Test have more than 2 000 digits.

However, in general, the number of choices for the order we have to make is not known, since the potential curve orders that one wants to factor have an unknown distribution.
But heuristic estimates for the time complexity of the algorithm are polynomial, e.g. $\mathcal{O}(\log^{4+\epsilon} n)$. Like the Goldwasser-Kilian Primality Test, the Atkin-Morain Primality Test also provides a certificate of primality.

7. The AKS Test

On August 6, 2002, Manindra Agrawal, Neeraj Kayal and Nitin Saxena, computer scientists at the Indian Institute of Technology Kanpur, published a paper called "PRIMES is in P", in which they introduced a deterministic polynomial time primality testing algorithm. The AKS test, named after its inventors, is the first primality-proving algorithm with polynomial running time. For this breakthrough Agrawal, Kayal and Saxena received the 2006 Goedel Prize and the 2006 Fulkerson Prize. [AKS04] is the main reference for this chapter.

7.1. The Idea

The test is based on the following lemma:

Lemma 7.1. *Let* $a \in \mathbb{Z}, n \in \mathbb{N}, n \geqslant 2$ *and* $\gcd(a, n) = 1$. *Then* n *is prime if and only if*

$$(x + a)^n \equiv x^n + a \pmod{n}. \tag{7.1}$$

Proof. The coefficient of x^k in $(x + a)^n$ is $\binom{n}{k} a^{n-k}$ for $0 < k < n$.

(i) Suppose n is prime. Then $n \mid \binom{n}{k}$ and $\binom{n}{k} \equiv 0 \pmod{n}$. Hence all coefficients are zero modulo n.

(ii) Suppose n is composite. Consider a prime factor q of n and let q^r be the greatest power of q dividing n. Then $q^r \nmid \binom{n}{q}$, since

$$\begin{aligned} \binom{n}{q} &= \frac{n(n-1)\cdots(n-q+1)}{q(q-1)\cdots 1} \\ &= \frac{q^r m(n-1)\cdots(n-q+1)}{q(q-1)\cdots 1} = \frac{q^{r-1} m(n-1)\cdots(n-q+1)}{(q-1)\cdots 1} \end{aligned}$$

Since $\gcd(a, n) = 1$, q^r is coprime to a^{n-q} and hence the coefficient of x^q is not zero modulo n. Thus $(x + a)^n \not\equiv x^n + a \pmod{n}$.

□

So if we want to find out, if a given number is prime or not the lemma above gives us a simple way to find out: if the congruence 7.1 is satisfied for at least one a, we know for sure, that n is prime. So far so good. The only problem is: in general this involves the computation of all coefficients of the polynomial and this cannot be done in polynomial time. In fact, if there are n coefficients it takes time $\Omega(n)$. The idea is now to reduce the number of coefficients. A simple way to do so is evaluating both sides of 7.1 modulo a polynomial of the form $x^r - 1$ for an appropriately chosen small r. So we test if the following equation is satisfied:

$$(x+a)^n \equiv x^n + a \pmod{x^r - 1, n}. \tag{7.2}$$

If n is prime, obviously 7.2 is satisfied for all values of a and r. The problem now is that there are composite numbers which also satisfy the congruence for some values of a and r. However, we will show that for appropriately chosen r, if the congruence 7.2 is satisfied for several values of a, then n must be a prime power. With this we get a deterministic primality testing algorithm with polynomial running time.

7.2. The Algorithm

The AKS primality testing algorithm is as follows:

Algorithm 7 The AKS Test

```
Input: integer n>1
Output: COMPOSITE if n is composite, PRIME if n is prime.
 1: if $n = a^b$ for $a \in \mathbb{N}$ and $b > 1$ then
 2:    return COMPOSITE
 3: end if
 4: Find the smallest r such that $\mathrm{ord}_r(n) > \log^2 n$.
 5: if $1 < \gcd(a, n) < n$ for some $a \leqslant r$ then
 6:    return COMPOSITE
 7: end if
 8: if $n \leqslant r$ then
 9:    return PRIME
10: end if
11: for $a = 1$ to $\lfloor \sqrt{\phi}(r) \log n \rfloor$ do
12:
13:    if $(x+a)^n \not\equiv x^n + a \pmod{x^r - 1, n}$ then
14:       return COMPOSITE
15:    end if
16:    return PRIME
17: end for
```

In the next section we will show that this algorithm determines, if a given number is prime or not and that the running time is polynomial.

7.2.1. Proof of Correctness

Theorem 7.2. *The algorithm above returns PRIME if and only if n is prime.*

We will prove the theorem through a sequence of lemmas. One direction is trivial:

Lemma 7.3. *If n is prime, the algorithm returns PRIME.*

Proof. If n is prime, step 2 and 6 of the algorithm can never return COMPOSITE. The for loop also cannot return COMPOSITE, because of Lemma 7.1. Therefore the algorithm will identify n either in step 9 or in step16 as a prime number. □

The other direction is not that easy. It is clear that if the algorithm identifies n as PRIME in step 9, n is definitely prime, since otherwise step 5 would have found a non-trivial divisor of n. So what remains is the case when the algorithm returns PRIME in step 16. We assume now that this is the case, so we have to prove that this implies that n is in fact prime.

In step 4 of the algorithm we have to find the smallest r such that $\operatorname{ord}_r(n) > \log^2 n$. This can be done by simply testing successive values for r. But then the question will be how large r can get. Since we want our algorithm to have polynomial running time, the answer to the answer to this question is very important. Luckily, we can bound the magnitude of the appropriate r:

Lemma 7.4. *There exist an* $r \leqslant \max\{3, \lceil\log^5 n\rceil\}$ *such that* $\operatorname{ord}_r(n) > \log^2 n$.

Proof. For $n = 2$ this is trivially true: $r = 3$ satisfies all conditions. So assume that $n > 2$. Consider the set of all numbers r_i such that either $\operatorname{ord}_{r_i}(n) \leqslant \log^2 n$ or $r_i \mid n$. Then every such r_i divides $n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor}(n^i - 1)$. We have:

$$\begin{aligned} n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor}(n^i - 1) &< n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor} n^i \\ &= n \cdot n^{1+2+\cdots+\lfloor\log^2 n\rfloor} \\ &= n^{\frac{\lfloor\log^2 n\rfloor\cdot(\lfloor\log^2 n\rfloor+1)}{2}+1} \\ &= n^{\frac{\lfloor\log^2 n\rfloor^2+\lfloor\log^2 n\rfloor+2}{2}} \end{aligned}$$

Since $\log^2 n > 2$ for every $n > 2$, we have $2 < \lfloor\log^2 n\rfloor^2 - \lfloor\log^2 n\rfloor$. With this we get:

$$\begin{aligned} n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor}(n^i - 1) &< n^{\frac{\lfloor\log^2 n\rfloor^2+\lfloor\log^2 n\rfloor+2}{2}} \\ &< n^{\lfloor\log^2 n\rfloor^2} \leqslant n^{(\log^2 n)^2} \\ &= n^{\log^4 n} = 2^{\log^5 n} \end{aligned}$$

Overall we have:

$$n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor}(n^i - 1) < 2^{\log^5 n}.$$

From Lemma 2.6 we know, that the $\operatorname{LCM}(\lceil\log^5 n\rceil) \geqslant 2^{\lceil\log^5 n\rceil}$. So:

$$n \cdot \prod_{i=1}^{\lfloor\log^2 n\rfloor}(n^i - 1) < \operatorname{LCM}(\lceil\log^5 n\rceil),$$

which means, that there must be a number smaller than $\lceil \log^5 n \rceil$, which does not divide the product. For this number, lets call it r, we have $\text{ord}_r(n) > \log^2 n$. If $\gcd(r, n) = 1$, we are done. So suppose $\gcd(r, n) > 1$. Then we define r' as $r' := \frac{r}{\gcd(r,n)}$. Since r does not divide the product and $\gcd(r', n) = 1$, r' cannot divide the product. Therefore, $\text{ord}_{r'}(n) > \log^2 n$. □

Remark.

- Since $\text{ord}_r(n) > 1$, there is a prime factor p of n such that $\text{ord}_r(p) > 1$. (Let $n = p_1 \cdots p_k$. Suppose $r \mid (p_i - 1)$ for all $i \in \{1, \dots, k\}$. Then $r \mid \prod_{i=1}^{k}(p_i - 1)$. But then $r \mid (n-1)$, which is a contradiction to $\text{ord}_r(n) > 1$.)

- $p > r$, since otherwise the algorithm would have returned COMPOSITE in step 6 or PRIME in step 9.

- $\gcd(n, r) = 1$, since otherwise either step 6 or step 9 of the algorithm would have identified n correctly as composite resp. prime. So $p, n \in \mathbb{Z}_r^*$.

p and r will be fixed in the remainder of this section. Let $l := \lfloor \phi(r) \log n \rfloor$.

In step 11 of the algorithm, l congruences are verified. Since the algorithm does not return COMPOSITE, we have:

$$(x + a)^n \equiv x^n + a \pmod{x^r - 1, n} \tag{7.3}$$

for every a, $0 < a \leqslant l$. This implies:

$$\begin{aligned} (x + a)^n &\equiv x^n + a \pmod{x^r - 1, p} && (7.4)\\ \xRightarrow{\text{Lemma 7.1}} (x + a)^p &\equiv x^p + a \pmod{x^r - 1, p} && (7.5) \end{aligned}$$

With (7.4) und (7.5) we get

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

And since $x^p \equiv x \pmod{x^r - 1, p}$ we have:

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

We see that both n and $\frac{n}{p}$ behave like the prime p with respect to the polynomial $x + a$. We give numbers with this property a name:

Definition. Let $f(x)$ be a polynomial and $m \in \mathbb{N}$. We say m is *introspective for* $f(x)$ *if*

$$f(x)^m \equiv f(x^m) \pmod{x^r - 1, p} \tag{7.6}$$

Introspective numbers are closed under multiplication:

Lemma 7.5. *Let* m *and* m' *be introspective for* $f(x)$*. Then* $m \cdot m'$ *is also introspective for* $f(x)$*.*

Proof. Since m is introspective for $f(x)$ we have:

$$((f(x))^m)^{m'} \equiv (f(x^m))^{m'} \pmod{x^r - 1, p}. \tag{7.7}$$

Since m' is introspective for $f(x)$ and by replacing x by x^m we get:

$$\begin{aligned}(f(x^m))^{m'} &\equiv f((x^m)^{m'}) \pmod{(x^m)^r - 1, p} &(7.8)\\ &\equiv f(x^{m\cdot m'}) \pmod{x^r - 1, p} &(7.9)\end{aligned}$$

The second relation holds because $(x^r - 1) \mid (x^{m\cdot r} - 1)$. Putting together (7.7) and (7.9) we get:

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

□

The following lemma shows that the set of polynomials for which a fixed number m is introspective, is also closed under multiplication:

Lemma 7.6. *Let m be introspective for $f(x)$ and $g(x)$. Then it is also introspective for $f(x) \cdot g(x)$.*

Proof. We have:

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

□

Let's define P as the set of all products of powers of $(x + a)$ for $0 \leqslant a \leqslant l$:

$$P := \Big\{ \prod_{a=0}^{l} (x+a)^{e_a} \quad | \quad e_a \geqslant 0 \Big\}.$$

Then Lemma 7.6 tells us that $\frac{n}{p}$ and p are introspective for all polynomials in P.
Let now I be the set of all products of powers of p^i and $(\frac{n}{p})^j$:

$$I := \Big\{ p^i \cdot \Big(\frac{n}{p}\Big)^j \quad | \quad i, j \geqslant 0 \Big\}.$$

Then we know from Lemma 7.5 that every number in I is introspective for every polynomial in P.

Based on the sets P and I we now define two groups:
The first group, let's call it G, is the set of all residues of numbers in $I \pmod r$. G is a subgroup of $\mathbb{Z}_r^*$ since $\gcd(n, r) = \gcd(p, r) = 1$. Let $|G| = t$. G is generated by n and p modulo r and since $\mathrm{ord}_r(n) > \log^2 n$, $t > \log^2 n$.
To define the second group, we need some basic facts about cyclotomic polynomials over finite fields. But first let's define what cyclotomic polynomials are:

Definition. Let p be prime and r be any positive integer. The r^{th} *cyclotomic polynomial over* $\mathbb{F}_p$ is then given by:

$$\Phi_r(x) := \prod_{\substack{1\leqslant k\leqslant r\\ \gcd(k,r)=1}} \Big(x - e^{\frac{2i\pi k}{r}}\Big) \in \mathbb{F}_p[x]$$

, where $e^{\frac{2i\pi k}{r}}$ is a primitive r^{th} root of unity.

Whereas cyclotomic polynomials over the integers are always irreducible (this is a well known fact, proved by several mathematicians such as Gauss, Landau, Kronecker and Dedekind; see [Wei]), for cyclotomic polynomials over finite fields this must not be the case, as the following example will show:

Example. The 4th cyclotomic polynomial is given by:

$$\begin{aligned}\Phi_4(x) &= \prod_{\substack{1\leqslant k\leqslant 4\\ \gcd(k,4)=1}} \left(x - e^{\frac{2i\pi k}{4}}\right) = (x - e^{\frac{2i\pi}{4}})(x - e^{\frac{2i\pi\cdot 3}{4}}) \\ &= (x-i)(x+i) = x^2 - ix + ix + 1 = x^2 + 1.\end{aligned}$$

Over the integers, x^2+1 is indeed irreducible, where as in $\mathbb{F}_5$ for example, we can factor it as follows:

$$(x+2)(x-2) = x^2 - 4 \equiv x^2 + 1 \pmod 5.$$

Lemma 7.7 (Cyclotomic Polynomials over finite fields). *Let $\Phi_r(x)$ be the r^{th} cyclotomic polynomial over $\mathbb{F}_p$. Then $\Phi_r(x)$ divides $x^r - 1$ and factors into irreducible polynomials of degree* $\mathrm{ord}_r(p)$.

A proof can be found in [LN86] for example.

Let now $h(x)$ be a factor of $\Phi_r(x)$ with degree $\mathrm{ord}_r(p)$. The degree of $h(x)$ is greater than 1 since $\mathrm{ord}_r(p) > 1$. The second group we have to consider then consists of all residues of polynomials in P modulo $h(x)$ and p. Let's call this group $\mathcal{G}$. It is generated by $x, x+1, x+2, \ldots, x+l$ in the field $\mathbb{F} := \mathbb{F}_p[x]/(h(x))$. It is a subgroup of the multiplicative group of $\mathbb{F}$. We now give a lower bound on the size of the subgroup $\mathcal{G}$:

Lemma 7.8 (Hendrik Lenstra Jr.). $|\mathcal{G}| \geqslant \binom{t+l}{t-1}$.

Proof. First note, that x is a primitive r^{th} root of unity in $\mathbb{F}$ since $h(x)$ is a factor of $\Phi_r(x)$, so in particular a factor of $x^r - 1$.
We first show, that any two distinct polynomials of degree less than $|G| = t$ in P will map to different elements in $\mathcal{G}$. Let $f(x)$ and $g(x)$ be two such polynomials in P and suppose that $f(x) \equiv g(x)$ in $\mathbb{F}$.Then $f(x)^m \equiv g(x)^m$ in $\mathbb{F}$ for every $m \in I$. Since m is introspective for both $f(x)$ and $g(x)$, by definition $f(x^m) \equiv g(x^m) \pmod{x^r - 1, p}$ and since $h(x)$ divides $x^r - 1$, $f(x^m) \equiv g(x^m)$ in $\mathbb{F}$. This implies that x^m is a root of the polynomial $Q(y) = f(y) - g(y)$ for every $m \in G$. As mentioned above, G is a subgroup of $\mathbb{Z}_r^*$ and $\gcd(m, r) = 1$, so each x^m is a primitive r^{th} root of unity. Hence there are $|G| = t$ distinct roots of $Q(y)$ in $\mathbb{F}$. But by the choice of f and g the degree of $Q(y)$ is less than t. This is contradiction and therefore $f(x) \not\equiv g(x)$ in $\mathbb{F}$.
Since $l = \lfloor \sqrt{\phi}(r) \log n \rfloor < \sqrt{r} \log n < r$ (the last inequality follows from $r \geqslant \mathrm{ord}_r(p) > \log^2 n$) and $p > r$, $i \not\equiv j \pmod p$ for $1 \leqslant i \neq j \leqslant l$. So $x, x+1, x+2, \ldots, x+l$ are all distinct in $\mathbb{F}$. Also $x + a \not\equiv 0$ in $\mathbb{F}$ for every a with $0 \leqslant a \leqslant l$, since $\deg(h) = \mathrm{ord}_r(p) > 1$. So there are et least $l+1$ distinct polynomials of degree one in $\mathcal{G}$. So there are at least

$$\sum_{i=0}^{t-1} \binom{t-1+l+1}{i} = \sum_{i=0}^{t-1} \binom{t+l}{i} = \binom{t+l}{t-1}$$

distinct polynomials with degree smaller than t in $\mathcal{G}$. □

Lemma 7.9. *If* n *is not a power of* p *then* $|\mathcal{G}| \leqslant n^{\sqrt{t}}$.

Proof. Let $\hat{I} := \left\{p^i \cdot \left(\frac{n}{p}\right)^j \mid 0 \leqslant i,j \leqslant \lfloor\sqrt{t}\rfloor\right\} \subset I$. If n is not a power of p then the set $\hat{I}$ has $(\sqrt{t}+1)^2 > t$ distinct elements. At least two numbers in $\hat{I}$ must be equal modulo r, since $|G| = t$. Let these be m_1 and m_2 and let $m_1 > m_2$. We then have:

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

Let $f(x) \in P$. Then the following holds:

$$\begin{aligned} f(x)^{m_1} &\equiv f(x^{m_1}) \pmod{x^r-1, p} \\ &\equiv f(x^{m_2}) \pmod{x^r-1, p} \\ &\equiv f(x)^{m_2} \pmod{x^r-1, p}. \end{aligned}$$

Therefore, $f(x)$ is a root of the polynomial $Q(y) := y^{m_1} - y^{m_2}$ in the field $\mathbb{F}$. Since $f(x)$ can be any polynomial in $\mathcal{G}$, the polynomial $Q(y)$ has at least $|\mathcal{G}|$ distinct roots in $\mathbb{F}$. Since the degree of $Q(y)$ is $m_1 \leqslant (\frac{n}{p} \cdot p)^{\lfloor\sqrt{t}\rfloor} \leqslant n^{\lfloor\sqrt{t}\rfloor}$, $|\mathcal{G}| \leqslant n^{\sqrt{t}}$. □

With these estimates on the size of $\mathcal{G}$ we are now able to prove the correctness of the algorithm:

Lemma 7.10. *If the algorithm returns PRIME then* n *is prime.*

Proof. First note that since $t > \log^2 n$, $t > \sqrt{t}\log n$. Also, since G is a subgroup of $\mathbb{Z}_r^*$, $|G| = t \leqslant \phi(r)$. With this we have $l = \lfloor\sqrt{\phi(r)}\log n\rfloor \geqslant \lfloor\sqrt{t}\log n\rfloor$. Now suppose the algorithm returns PRIME. Lemma 7.8 implies that for $|G| = t$ and $l = \lfloor\sqrt{\phi(r)}\log n\rfloor$ the following holds:

$$\begin{aligned} |\mathcal{G}| &\geqslant \binom{t+l}{t-1} \\ &\geqslant \binom{l+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor} \\ &\geqslant \binom{2\cdot\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor} \\ &> 2^{\lfloor\sqrt{t}\log n\rfloor+1} \\ &\geqslant n^{\sqrt{t}}. \end{aligned}$$

By Lemma 7.9, $|\mathcal{G}| \leqslant n^{\sqrt{t}}$, if n is not a power of p. Therefore, $n = p^k$ for some $k > 0$. If $k > 1$ the algorithm would have returned COMPOSITE in step 2. Therefore n must be prime. □

We have thus proven that the algorithm returns PRIME if and only if n is prime. What we still have to show is that the running time of the algorithm is polynomial. This will be part of the next section.

7.3. Time Complexity Analysis

Theorem 7.11. *The asymptotic time complexity of the AKS-algorithm is* $\hat{\mathcal{O}}(\log^{\frac{21}{2}} n)$.

Proof. . For the proof, we use Lemma 4.2 and 4.3. In the first step of the algorithm, we have to test if $n = a^b$. Since b is bounded by $\log n$ ($n = 2^{\log n}$), this takes time $\hat{\mathcal{O}}(\log^3 n)$.
We then have to find an r with $\operatorname{ord}_r(n) > \log^2 n$. To do so, we try out successive values of r and test if $n^k \not\equiv 1 \pmod r$ for all $k \leqslant \log^2 n$. For each r (with input length $\log r$), this takes at most $\mathcal{O}(\log^2 n)$ multiplications modulo r, which gives us a time complexity of $\hat{\mathcal{O}}(\log^2 n \log r)$. By Lemma 7.4 we know that we have to try at most $\mathcal{O}(\log^5 n)$ different r's. So the total time complexity of step 4 is $\hat{\mathcal{O}}(\log^7 n)$.
In step 5 we have to compute the greatest common divisor of at most r numbers, where every computation takes time $\mathcal{O}(\log n)$.Therefore, the total time complexity of step 5 is $\mathcal{O}(r \log n) = \mathcal{O}(\log^6 n)$.
The time complexity of step 8 (testing, if $n \leqslant r$), is $\mathcal{O}(\log n)$.
In step 11 we have to check at most $\lfloor \sqrt{\phi(r)} \log n \rfloor$ equations. Each equation involves $\mathcal{O}(\log n)$ multiplications of polynomials with degree r and coefficients of size $\mathcal{O}(\log n)$. Thus, each equation can be checked in time $\hat{\mathcal{O}}(r \log^2 n)$. So the total time complexity of step 11 is $\hat{\mathcal{O}}(r\sqrt{\phi}(r) \log^3 n) = \hat{\mathcal{O}}(r^{\frac{3}{2}} \log^3 n) = \hat{\mathcal{O}}((\log^5)^{\frac{3}{2}} \log^3 n) = \hat{\mathcal{O}}(\log^{\frac{21}{2}} n)$. So the time complexity of this step is dominant, hence it is the total time complexity of the AKS-algorithm. □

7.3.1. Improvements

We know now, that the AKS-algorithm has polynomial time complexity. But we would like to have an even better running time, since we know there are very good probabilistic primality tests with a much better time complexity. So what can we do? We could try to improve the time complexity by improving the estimate for r. The best possible value for r is $r = \mathcal{O}(\log^2 n)$, since for smaller values of r, $r > \operatorname{ord}_r(n) > \log^2 n$ would not be possible. In this case of $r = \mathcal{O}(\log^2 n)$ the time complexity of the algorithm would be $\hat{\mathcal{O}}(log^6 n)$, as one can easily check.
There are two conjectures that support the possibility of such an estimate for r:

Conjecture 1 (Artin's Conjecture)**.** Given any number $n \in \mathbb{N}$ that is not a perfect square, the number of primes $q \leqslant m$ for which $\operatorname{ord}_r(n) = q - 1$ is asymptotically $A(n) \cdot \frac{m}{\ln m}$ where $A(n)$ is Artin's constant with $A(n) > 0.35$.

For $m = \mathcal{O}(log^2 n)$, Artin's Conjecture would show that there is an $r = \mathcal{O}(\log^2 n)$ with the required properties.It is known, that if the *Generalized Riemann Hypothesis* holds, Artin's Conjecture is true.

Conjecture 2 (Sophie-Germain Prime Density Conjecture)**.** The number of primes $q \leqslant m$ such that $2q + 1$ is also prime is asymptotically $\frac{2C_2 m}{\ln^2 m}$ where C_2 is the twin prime constant (estimated to be approximately 0.66). Primes q with this property are called *Sophie-Germain Primes*.

If this conjecture holds, there must exist at least $\log^2 n$ Sophie-Germain prime numbers between $8\log^2 n$ and $c\log^2 n(\log\log n)^2$ for suitable constant c. Let $r = 2q+1$. Then either $\mathrm{ord}_r(n) \leqslant 2$ or $\mathrm{ord}_r(n) \geqslant q$. If $\mathrm{ord}_r(n) \leqslant 2$, then r divides $n^2 - 1$. Since the number of such numbers r is bounded by $\mathcal{O}(\log n)$, this implies that there is a prime $r = \hat{\mathcal{O}}(\log^2 n)$ such that $\mathrm{ord}_r(n) > \log^2 n$. Such an r therefore qualifies for the AKS-algorithm and it would yield a time complexity of $\hat{\mathcal{O}}(\log^6 n)$.

Apart from these conjectures, there is a lemma proved by *Fouvry* in 1985 [Fou85], which also improves the estimate of r:

Lemma 7.12 (Fouvry's Theorem). *Let* $P(m)$ *denote the greatest prime divisor of* m. *There exist constants* $c > 0$ *and* n_0 *such that*

$$|\{q \mid q \textit{ is prime, } q \leqslant x \textit{ and } P(q-1) > q^{\frac{2}{3}}\}| \geqslant c\frac{x}{\ln x}$$

for all $x \geqslant n_0$.

Specifying the constant c is quite difficult and so is the proof of the lemma. However, R. C. Baker and G. Harman had shown in [BH96] that the lemma holds for constants up to 0.6683.

With Fouvry's theorem one can show, that r may be chosen with $r = \mathcal{O}(\log^3 n)$, which gives the following improvement:

Theorem 7.13. *The asymptotic time complexity of the AKS-algorithm is* $\hat{\mathcal{O}}(\log^{\frac{15}{2}} n)$.

See [AKS04] for the proof. The fact that the proof of Fouvry's theorem is not only very hard but also ineffective (meaning that from the proof it is not possible to give a numerical explicit upper bound for the number of operations) makes the last result not that satisfying.

Hendrik Lenstra and Carl Pomerance have found a way to improve the time complexity even more and without using such a deep result as Fouvry's theorem. We will explain their idea in the next subsection.

7.3.2. The Version of Lenstra and Pomerance[1]

Agrawal, Kayal and Saxena reduced the number of coefficients in (7.1) by evaluating it modulo a polynomial of the form $x^r - 1$. The approach of Lenstra and Pomerance (first given in 2005) is via a more general polynomial $f(x)$.

Theorem 7.14 (Lenstra and Pomerance). *Suppose* n *is an integer with* $n \geqslant 2$, $f(x)$ *is a monic polynomial in* $\mathbb{Z}_n[x]$ *of degree* t, *where* $t > \log^2 n$,

$$f(x^n) \equiv 0 \pmod{f(x)}, \quad x^{n^t} \equiv x \pmod{f(x)}, \tag{7.10}$$

and

$$x^{n^{\frac{t}{q}}} - x \textit{ and } f(x) \textit{ are coprime for all primes} q \textit{ dividing } t. \tag{7.11}$$

[1]This subsection is based on [Gra04], [CP05] and [JP15]

Then p *is prime if and only if the following three conditions hold:* n *is not a perfect power,* n *does not have any prime factor* $p \leqslant t$ *and*

$$(x+a)^n \equiv x^n + a \pmod{f(x)} \tag{7.12}$$

for each integer a *with* $0 \leqslant a \leqslant \sqrt{t}\log n$.

Proof. As in the proof of the AKS algorithm, one direction is trivial, that is if n is prime. So suppose n is composite and that the three hypotheses hold. Let p be a prime dividing n (with $p > t$) and let $h(x)$ be an irreducible factor of $f(x) \pmod p$, so $\mathbb{F} = \mathbb{Z}_p[x]/(h(x))$ is a finite field.
Let $l = \lfloor \sqrt{t}\log n \rfloor$. As in subsection 7.2.1, let $\mathcal{G}$ be the subgroup of $\mathbb{F}$ generated multiplicatively by $x, x+1, x+2, \dots, x+l$ and let I be the set of positive integers of the form $p^i \cdot n^j$ with $i, j \geqslant 0$. Further, let r be the order of x in $\mathbb{Z}_p[x]/(f(x))$. By the second condition of (7.10) and by (7.11) we thus have that t is the order of $n \pmod r$. The powers $x^{n^0}, x^{n^1}, \dots, x^{n^{t-1}}$ are distinct in $\mathbb{Z}_p[x]/(f(x))$ and also in $\mathbb{F}$. Therefore, the roots of the polynomial $g(y) := \prod_{i=0}^{t-1}(y - x^{n^i}) \in \mathbb{F}[y]$ are distinct. Since $g(x^p) \equiv g(x)^p \equiv 0$ in $\mathbb{F}$, $x^p \equiv x^{n^j}$ in $\mathbb{F}$ for some j. Therefore, $p \equiv n^j \pmod r$. Thus, if G denotes the set of residues of numbers in I modulo r, generated by n and p, then G is in fact generated by n alone and has therefore t elements (as in subsection 7.2.1).
The proof that I is closed under multiplication, works like the proof of Lemma 7.7, except using the following observation (instead of using the fact that $(x^r - 1) \mid (x^{m\cdot r} - 1)$.): $f(x^k) \equiv 0 \pmod{f(x)}$ for all $k \in I$, which holds by the first condition of 7.10 and since $f(x^p) \equiv f(x)^p \pmod p \equiv 0 \pmod{f(x)}$.
Note that in 7.2.1 we had $x + a \in \mathcal{G}$ for every a with $0 \leqslant a \leqslant l = \lfloor \sqrt{\phi(r)}\log n \rfloor$ but in the proof of Lemma 7.10 we only used that $0 \leqslant a \leqslant \lfloor \sqrt{t}\log n \rfloor$, which is the condition we now have. Now, we can simply follow the proof in subsection 7.2.1. So like in the proof of 7.10 we get a contradiction and thus n must be prime. $\square$

With the AKS test, we conjectured that there are suitable values for r with $r = \mathcal{O}(\log^2 n)$. But the best estimate for r that we could actually prove was $r \leqslant \log^5 n$ (besides the improvement due to Fouvry's theorem).
We now consider a more general polynomial $f(x)$, more precisely, we can consider any monic polynomial with degree smaller than $\log^2 n$, such that (7.10) and (7.11) are satisfied. Note, that if n is prime, then a polynomial $f(x)$ satisfies (7.10) and (7.11) if and only if $f(x)$ is irreducible in $\mathbb{Z}_n[x]$ (see [CP05], Theorem 2.2.8). Also, there are many monic irreducible polynomials of any given degree (see [CP05], (2.5)). So the first idea would be, to just let $t = \lfloor \log^2 n \rfloor + 1$ and choose a polynomial of degree t that would be irreducible if n were prime.
As usual, things are not that easy. The problem is that most deterministic polynomial algorithms for finding an irreducible polynomial of any given degree depend on the ERH. But in [JP15] it is shown how to deterministically find an irreducible polynomial modulo a prime number p with degree in $[t, 2t]$:

Theorem 7.15. *There is an effectively computable positive integer* c *and a deterministic algorithm such that the following holds: Given a prime* p *and positive integer* $t > \log^{\frac{46}{25}} p$*, the algorithm computes an irreducible polynomial* f *in* $\mathbb{F}_p[x]$ *of degree* t'*, where* $t \leqslant t' \leqslant 2t$*. Moreover, the running time of the algorithm is at most* $(t\log p)\cdot(2 + \log t + \log\log p)^c$.

So lets take $t = \lfloor \log^2 n \rfloor + 1$ and run the algorithm of Lenstra and Pomerance on a large number n. If n is prime, then the algorithm finds an irreducible polynomial with degree in $[t, 2t]$. If n is composite, then two things can happen: either the algorithm finds a polynomial with degree in $[t, 2t]$ such that (7.10) and (7.11) hold or the algorithm will crash. In the latter case, n will then have been proven composite.
If the algorithm succeeds in finding a suitable polynomial, one tests if (7.12) holds for the required values of a. This step takes time $\hat{\mathcal{O}}(t^{\frac{3}{2}} \log^3 n) = \hat{\mathcal{O}}(\log^6 n)$ and since this step was dominant, this is the total time complexity of deciding whether n is prime or composite.

The polynomial construction method based on *Gaussian periods* that Lenstra and Pomerance are using is quite complicated. A detailed description can be found in [JP15].
Note, that in practice one should always find a small r so that the running time of the algorithm does not get too big. Therefore, the version of Lenstra and Pomerance gives no practical advantage over the AKS algorithm. However, we now have the proof that deciding whether n is prime or composite can be done in time $\hat{\mathcal{O}}(\log^6 n)$.

7.3.3. Further Improvements

Since the best possible estimate for r and d, respectively, was $r = \mathcal{O}(\log^2 n)$, we already reached the best possible time complexity of $\hat{\mathcal{O}}(\log^6 n)$. For getting a deterministic algorithm with running time having an exponent smaller than 6, a fundamental new idea would be required [JP15].
However, Daniel J. Bernstein presented an algorithm that, given a prime n, finds and verifies a proof of primality of n in random time $\hat{\mathcal{O}}(\log^4 n)$ [Ber07]. His approach was build on an idea of Pedro Berrizbeitia [Ber05].

8. Which Primality Testing Algorithm should be used?

The answer to this question depends obviously on the implementation. However, we will discuss this issue only concentrating on the time complexities stated in the above chapters.

Maybe, the most interesting question is: Should we use the AKS test in practice? As we saw, with the original version and its improvements we are able to decide whether or not n is prime in time $\hat{O}(\log^6 n)$. Although this is a great (theoretical) result, it is not yet very practical. If one needs to decide quickly, if a number is prime - for cryptographic applications for example - the AKS test is still not fast enough. In fact, the AKS test is not used in practice at all. So lets see, what other methods can offer.

A (deterministic) alternative is the Atkin-Morain elliptic curve method. The heuristic time complexity is $\hat{O}(\log^4 n)$, however, the worst case running time is not known. Recall that elliptic curve primality tests have the great advantage that they also provide a certificate for primality. Since it works very well in practice, elliptic curve primality proving is one of the most widely used methods.

In many cases, one might want to take the (very little) risk of a false output and use a probabilistic method. One of the fastest probabilistic algorithms is the Miller-Rabin test. As we saw, using the Schönhage-Strassen algorithm for fast multiplication we got a time complexity of $\hat{O}(k \log^2 n)$. This is so much faster than the AKS test and also faster than the Atkin-Morain test, that this seems to be a very good option. However, note that for prime numbers of unknown origin, such a probabilistic test should not be used to verify primality. This is especially relevant in cryptography, since an adversary might try to send you a pseudoprime in a place where a prime number is needed. But for choosing a number at random and then check, if it is prime, a test like Miller-Rabin could be used (and it is).

Tests like Pépin or Lucas-Lehmer, which do not work for general (prime) numbers, are not very useful in cryptography, since the number of such primes is very limited. However, when it comes to prime number records, they are key. Especially the Lucas-Lehmer test helped to find many of the largest prime numbers known today. We already mentioned this in subsection 5.2.2.

But what about the Sieve of Eratosthenes and trial division? Even though they are very inefficient, for smaller numbers they still could be used (and they are). Recall, that some methods where the primality of a number depends on the primality of a smaller number - like the Goldwasser-Kilian test - have to be applied several times, until some divisor can be proven prime by trial division. The trial division is in fact used in practice - if only as a part of some other test.

Overall, deciding which primality test to use highly depends on the reason why one wants to know that a number is prime. Very often, a probabilistic test like Miller-Rabin is used first, and when the output is (pseudo)prime, a test like Atkin-Morain is used to verify the result.

Note, that there are other commonly used primality tests that we did not present, for example the *Baillie-Pomerance-Selfridge-Wagstaff test (Baillie-PSW)*. It is a combination of a Miller-Rabin test and a Lucas test. See [Nic12] for a description.

Appendices

A. List of Prime Numbers and Prime Number Records

A.1. Prime numbers

Table A.1 shows all prime numbers up to 2000. [Cal] provides more lists of prime numbers. One can also test if a certain number is prime or not.

Table A.1.: The prime numbers up to 2000

2	3	5	7	11	13	17	19	23	29	31	37
41	43	47	53	59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131	137	139	149	151
157	163	167	173	179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409	419	421	431	433
439	443	449	457	461	463	467	479	487	491	499	503
509	521	523	541	547	557	563	569	571	577	587	593
599	601	607	613	617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719	727	733	739	743
751	757	761	769	773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881	883	887	907	911
919	929	937	941	947	953	967	971	977	983	991	997
1009	1013	1019	1021	1031	1033	1039	1049	1051	1061	1063	1069
1087	1091	1093	1097	1103	1109	1117	1123	1129	1151	1153	1163
1171	1181	1187	1193	1201	1213	1217	1223	1229	1231	1237	1249
1259	1277	1279	1283	1289	1291	1297	1301	1303	1307	1319	1321
1327	1361	1367	1373	1381	1399	1409	1423	1427	1429	1433	1439
1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499	1511
1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	1597	1601
1607	1609	1613	1619	1621	1627	1637	1657	1663	1667	1669	1693
1697	1699	1709	1721	1723	1733	1741	1747	1753	1759	1777	1783
1787	1789	1801	1811	1823	1831	1847	1861	1867	1871	1873	1877
1879	1889	1901	1907	1913	1931	1933	1949	1951	1973	1979	1987
1993	1997	1999									

A.2. Fermat numbers

The first five Fermat numbers $F_n = 2^{2^n} + 1$ are:

$$F_0 = 3, \quad F_1 = 5, \quad F_2 = 17, \quad F_3 = 257, \quad F_4 = 65537.$$

Table A.2 shows the status of composite Fermat numbers up to F_{20}. The date indicates the year when the first factor was found. The current status of other Fermat numbers can be found (HERE).

Table A.2.: Composite Fermat numbers

n	Status	Date	Discovered by
5	completely factored	1732	L. Euler
6	completely factored	1855	T. Clausen
7	completely factored	1970	M. A. Morrison & J. Brillhart
8	completely factored	1980	R. P. Brent & J. M. Pollard
9	completely factored	1903	A.E. Western
10	completely factored	1962	J. Brillhart
11	completely factored	1899	A. Cunningham
12	factorization incomplete	1877	I. M. Pervushin; E. Lucas
13	factorization incomplete	1974	J. C. Hallyburton & J. Brillhart
14	factorization incomplete	2010	T. Rajala & Woltman
15	factorization incomplete	1925	M. B. Kraitchik
16	factorization incomplete	1953	J. L. Selfridge
17	factorization incomplete	1978	G. B. Gostin
18	factorization incomplete	1903	A.E. Western
19	factorization incomplete	1962	H. Riesel
20	without known factors	1987	D. A. Buell & J. Young

A.3. Mersenne numbers

Table A.3 shows all known Mersenne prime numbers (as of October 2016).

Table A.3.: Mersenne prime numbers

#	$2^p - 1$	Date	Discovered by	Method
1	$2^2 - 1$	500 BCE	Ancient Greek mathematicians	
2	$2^3 - 1$	500 BCE	Ancient Greek mathematicians	
3	$2^5 - 1$	275 BCE	Ancient Greek mathematicians	
4	$2^7 - 1$	275 BCE	Ancient Greek mathematicians	
5	$2^{13} - 1$	1456	Anonymous	trial division
6	$2^{17} - 1$	1588	P. Cataldi	trial division
7	$2^{19} - 1$	1588	P. Cataldi	trial division

Table A.3.: Mersenne prime numbers

#	$2^p - 1$	Date	Discovered by	Method
8	$2^{31}-1$	1772	L. Euler	Enhanced trial division
9	$2^{61}-1$	1883	I. M. Pervushin	Lucas Sequences
10	$2^{89}-1$	1911	R. E. Powers	Lucas Sequence
11	$2^{107}-1$	1914	R. E. Powers	Lucas Sequence
12	$2^{127}-1$	1876	É. Lucas	Lucas Sequence
13	$2^{521}-1$	1952	R. M. Robinson	Lucas-Lehmer
14	$2^{607}-1$	1952	R. M. Robinson	Lucas-Lehmer
15	$2^{1279}-1$	1952	R. M. Robinson	Lucas-Lehmer
16	$2^{2203}-1$	1952	R. M. Robinson	Lucas-Lehmer
17	$2^{2281}-1$	1952	R. M. Robinson	Lucas-Lehmer
18	$2^{3217}-1$	1957	H. Riesel	Lucas-Lehmer
19	$2^{4253}-1$	1961	A. Hurwitz	Lucas-Lehmer
20	$2^{4423}-1$	1961	A. Hurwitz	Lucas-Lehmer
21	$2^{9689}-1$	1993	D. B. Gillies	Lucas-Lehmer
22	$2^{9941}-1$	1963	D. B. Gillies	Lucas-Lehmer
23	$2^{11213}-1$	1963	D. B. Gillies	Lucas-Lehmer
24	$2^{19937}-1$	1871	B. Tuckerman	Lucas-Lehmer
25	$2^{21701}-1$	1978	L. C. Noll & L. Nickel	Lucas-Lehmer
26	$2^{23209}-1$	1997	L. C. Noll	Lucas-Lehmer
27	$2^{44497}-1$	1979	H. L. Nelson & D. Slowinski	Lucas-Lehmer
28	$2^{86243}-1$	1982	D. Slowinski	Lucas-Lehmer
29	$2^{110503}-1$	1988	W. Colquitt & L. Welsh	Lucas-Lehmer
30	$2^{132049}-1$	1983	D. Slowinski	Lucas-Lehmer
31	$2^{216091}-1$	1985	D. Slowinski	Lucas-Lehmer
32	$2^{756839}-1$	1992	D. Slowinski & P. Gage	Lucas-Lehmer
33	$2^{859433}-1$	1994	D. Slowinski & P. Gage	Lucas-Lehmer
34	$2^{1257787}-1$	1996	D. Slowinski & P. Gage	Lucas-Lehmer
35	$2^{1398269}-1$	1996	GIMPS - J. Armengaud	Lucas-Lehmer
36	$2^{2976221}-1$	1997	GIMPS - G. Spence	Lucas-Lehmer
37	$2^{3021377}-1$	1998	GIMPS - R. Clarkson	Lucas-Lehmer
38	$2^{6972593}-1$	1999	GIMPS - N. Hajratwala	Lucas-Lehmer
39	$2^{13466917}-1$	2001	GIMPS - M. Cameron	Lucas-Lehmer
40	$2^{20996011}-1$	2003	GIMPS - M. Shafer	Lucas-Lehmer
41	$2^{24036583}-1$	2004	GIMPS - J. Findley	Lucas-Lehmer
42	$2^{25964951}-1$	2005	GIMPS - M. Nowak	Lucas-Lehmer
43	$2^{30402457}-1$	2005	GIMPS - C. Cooper & S. Boone	Lucas-Lehmer
44	$2^{32582657}-1$	2006	GIMPS - C. Cooper & S. Boone	Lucas-Lehmer
45	$2^{37156667}-1$	2008	GIMPS - H.-M. Elvenich	Lucas-Lehmer
46	$2^{42643801}-1$	2009	GIMPS - O. M. Strindmo	Lucas-Lehmer
47	$2^{43112609}-1$	2008	GIMPS - E. Smith	Lucas-Lehmer

Table A.3.: Mersenne prime numbers

#	$2^p - 1$	Date	Discovered by	Method
48	$2^{57885161} - 1$	2013	GIMPS - C. Cooper	Lucas-Lehmer
49	$2^{74207281} - 1$	2016	GIMPS - C. Cooper	Lucas-Lehmer

A.4. Sophie Germain prime numbers

Table A.4 shows the ten greatest known Sophie Germain prime numbers (as of October 2016).

Table A.4.: Top 10 Sophie Germain prime numbers

x	prime	digits	Date
1	$2618163402417 \cdot 2^{1290000} - 1$	388 342	2016
2	$18543637900515 \cdot 2^{666667} - 1$	200 701	2012
3	$183027 \cdot 2^{265440} - 1$	79 911	2010
4	$648621027630345 \cdot 2^{253824} - 1$	76 424	2009
5	$620366307356565 \cdot 2^{253824} - 1$	76474	2009
6	$99064503957 \cdot 2^{200008} - 1$	60 220	2016
7	$607095 \cdot 2^{176311} - 1$	53 081	2009
8	$48047305725 \cdot 2^{172403} - 1$	51 910	2007
9	$137211941292195 \cdot 2^{171960} - 1$	51 780	2006
10	$31737014565 \cdot 2^{140003} - 1$	42 156	2010

Table A.5 shows the actual numbers of Sophie Germain primes up to a certain number m compared to the estimated number of the Sophie Germain prime density conjecture.

Table A.5.: Sophie Germain primes up to m

m	actual	estimate
1 000	37	39
100 000	1 171	1 166
10 000 000	56 032	56 128
100 000 000	423 140	423 295
1 000 000 000	3 308 859	3 307 888
10 000 000 000	26 569 515	26 568 824

B. Schoof's Algorithm[1]

To be able to explain the idea of Schoof, we need a few more tools for elliptic curves over finite fields:
For any elliptic curve $E_{a,b}(\mathbb{F}_p)$ defined over a finite field $\mathbb{F}_p$, that is whose points have coordinates in $\mathbb{F}_p$, we add those points of E whose coordinates lie in an algebraic closure $\overline{\mathbb{F}}_p$ of $\mathbb{F}_p$. The so defined elliptic curve defined over $\overline{\mathbb{F}}_p$ is denoted by $E_{a,b}(\overline{\mathbb{F}}_p)$. The *Frobenius endomorphism* Φ on $E_{a,b}(\overline{\mathbb{F}}_p)$ is defined by $\Phi(x,y) = (x^p, y^p)$ and $\Phi(\mathcal{O}) = \mathcal{O}$.

The following theorem establishes a connection between the points of E defined over $\mathbb{F}_p$ and those over $\mathbb{F}_p$:

Theorem B.1. *Let* $|E_{a,b}(\mathbb{F}_p)| = p + 1 - t$, *then*

$$\Phi^2(P) - t\Phi(P) + pP = \mathcal{O},$$

for every point $P \in E_{a,b}(\overline{\mathbb{F}}_p)$.

If we know the value of t, we obviously know the order of $E_{a,b}(\mathbb{F}_p)$.
For any positive integer n, we now consider those points P of $E_{a,b}(\overline{\mathbb{F}}_p)$ for which $nP = \mathcal{O}$. Those are the points of order dividing n in the group, the so called n-*torsion* points. We denote this set by $E[n]$. By Theorem B.1, we have

$$\Phi^2(P) - (t \mod n)\Phi(P) + (p \mod n)P = \mathcal{O}, \text{ for all } P \in E[n]. \tag{B.1}$$

Schoof's idea was to use this equation to compute the residue $t \mod n$ by trial and error procedure until the correct value that satisfies (B.1) is found. To do this, the *division polynomials* are used. These polynomials both simulate elliptic multiplication and pick out n-torsion points. They are defined as follows:

Definition. To an elliptic curve $E_{a,b}(\mathbb{F}_p)$ we associate the *division polynomials* $\Psi_n(x,y) \in \mathbb{F}_p[x,y]/(y^2 - x^3 - ax - b)$ as follows:

$$\begin{aligned}\Psi_{-1} &= -1, \Psi_0 = 0, \Psi_1 = 1, \Psi_2 = 2y,\\ \Psi_3 &= 3x^4 + 6ax^2 + 12bx - a^2,\\ \Psi_4 &= 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3),\end{aligned}$$

while all further cases are given by

$$\begin{aligned}\Psi_{2n} &= \Psi_n \frac{\Psi_{n+2}\Psi_{n-1}^2 - \Psi_{n-2}\Psi_{n+1}^2}{2y},\\ \Psi_{2n+1} &= \Psi_{n+2}\Psi_n^3 - \Psi_{n+1}^3\Psi_{n-1}.\end{aligned}$$

[1]This chapter is based on [CP05] and [Wik]

Algorithm 8 summarizes Schoof's algorithm. For a detailed explanation see [CP05] or [Sch85].

Algorithm 8 Schoof's Algorithm for curve order

```
Input: A prime number $p > 3$ and an elliptic curve $E_{a,b}(\mathbb{F}_p) : y^2 = x^3 + ax + b$.
Output: The curve order $|E_{a,b}(\mathbb{F}_p)|$.
 1: Choose a set $S$ of odd primes not containing $p$ such that $N = \prod_{l \in S} l > 4\sqrt{p}$.
 2: if $\gcd(x^p - x, x^3 + ax + b) \neq 1$ then
 3:   $t_2 = 0$
 4: else
 5:   $t_2 = 1$
 6: end if
 7: Compute the division polynomials $\Psi_l$.
    All computations in the loop below are performed in the ring $\mathbb{F}_p[x,y]/(y^2 - x^3 - ax - b, \Psi_l)$.
 8: for $l \in S$ do
 9:   Let $\overline{p}$ be the unique integer such that $p \equiv \overline{p} \pmod{l}$ and $|\overline{p} < \frac{l}{2}|$.
      Compute $(x^p, y^p)$, $(x^{p^2}, y^{p^2})$ and $(x_{\overline{p}}, x_{\overline{p}}) = \left(x - \frac{\Psi_{\overline{p}-1}\Psi_{\overline{p}+1}}{\Psi_{\overline{p}}^2}, \frac{\Psi_{2\overline{p}}}{2\Psi_{\overline{p}}^4}\right)$.
10:   if $x^{p^2} \neq x_{\overline{p}}$ then
11:     Compute $(X, Y) = (x^{p^2}, y^{p^2}) + (x_{\overline{p}}, x_{\overline{p}})$.
12:     for $1 \leqslant \overline{t} \leqslant \frac{l-1}{2}$ do
13:
14:       if $X = x_{\overline{t}}^p$ then
15:
16:         if $Y = y_{\overline{t}}^p$ then
17:           $t_l = \overline{t}$
18:         else
19:           $t_l = -\overline{t}$.
20:         end if
21:       end if
22:     end for
23:   else if $q$ is a square modulo $l$ then
24:     compute $w$ with $q \equiv w^2 \pmod{l}$
        compute $w(x^p, y^p)$
25:     if $w(x^p, y^p) = (x^{p^2}, y^{p^2})$ then
26:       $t_l = 2w$
27:     else if $w(x^p, y^p) = (x^{p^2}, -y^{p^2})$ then
28:       $t_l = -2w$
29:     else
30:       $t_l = 0$
31:     end if
32:   else
33:     $t_l = 0$
34:   end if
35: end for
36: Use the Chinese Remainder Theorem to compute $t$ modulo $N$ from the equations
    $x \equiv t_l \pmod{l}$, where $l \in S$.
37: OUTPUT $q + 1 - t$
```

C. Zusammenfassung

Im August 2002 veröffentlichten Manindra Agrawal, Neeraj Kayal und Nitin Saxena, alle drei Informatiker und Mathematiker am "Indian Institute of Technology Kanpur", den ersten deterministischen Primzahltest mit polynomialer Laufzeit. Der sogenannte *AKS Test* war eine Sensation, denn bis zur Veröffentlichung war nicht bekannt, ob es überhaupt einen derartigen Primzahltest gibt. In der vorliegenden Arbeit machen wir eine Reise durch mehr als 2000 Jahre Primzahltests. Dabei werden wir die fundamentalsten und bekanntesten Primzahltests vorstellen - angefangen beim Sieb des Eratosthenes über Fermat, Miller-Rabin, Lucas und Primzahltests basierend auf elliptischen Kurven bis hin zum gefeierten AKS Test. Am Ende gehen wir der Frage nach, welcher Primzahltest für welchen Zweck verwendet werden sollte.

Bibliography

[AGP94] William Robert Alford, Andrew Granville, and Carl Pomerance. There are infinitely many Carmichael numbers. *Annals of Mathematics*, 140:703–722, 1994.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160:781–793, 2004.

[AM93] A. O. L. Atkin and François Morain. Elliptic Curves and Primality Proving. *Mathematics of Computation*, 61(203):29–68, 1993.

[Ber05] Pedro Berrizbeitia. Sharpening "PRIMES IS IN P" for a large family of numbers. *Mathematics of Computation*, 74(252):2043–2059, 2005.

[Ber07] Daniel J. Bernstein. PROVING PRIMALITY IN ESSENTIALLY QUARTIC RANDOM TIME. *Mathematics of Computation*, 76(257):389–403, 2007.

[BH96] R. C. Baker and G. Harman. The Brun-Titchmarsh Theorem on average. *Proceedings of a conference in Honor of Heini Halberstam*, 1:39–103, 1996.

[Bun08] Peter Bundschuh. *Einführung in die Zahlentheorie*. Springer, 6 edition, 2008.

[Cal] Chris K. Caldwell. The Prime Pages. `https://primes.utm.edu`. last checked: 17.11.2016.

[CP05] Richard Crandall and Carl Pomerance. *Prime Numbers - A Computational Perspective*. Springer, 2 edition, 2005.

[CRXK12] Chris K. Caldwell, Angela Reddick, Yeng Xiong, and Wilfrid Keller. The History of the Primality of One: A Selection of Sources. *Journal of Integer Sequences*, 15(12.9.8), 2012. last checked: 17.11.2016.

[DPV06] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. Algorithms with numbers. `https://people.eecs.berkeley.edu/~vazirani/algorithms/chap1.pdf`, 2006. lecture notes, last checked: 17.11.2016.

[Fou85] E. Fouvry. Theoreme de Brun-Titchmarsh; application au theoreme de Fermat. *Invent. Math.*, 79:383–407, 1985.

[GK99] Shafi Goldwasser and Joe Kilian. Primality Testing Using Elliptic Curves. *Journal of the ACM*, 46(4):450–472, 1999.

[Gra04] Andrew Granville. It is easy to determine whether a given integer is prime. *Bulletin of the American Mathematical Society*, 42(1):3–38, 2004.

[HPS08] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008.

[HW75] G.H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 4 edition, 1975.

[JP15] H. W. Lenstra Jr. and Carl Pomerance. Primality Testing with Gaussian periods. `https://math.dartmouth.edu/~carlp/aks06-2015.pdf`, 2015. Revision of preprint from 2005.

[Kel] Wilfrid Keller. Prime factors $k \cdot 2^n + 1$ of Fermat numbers F_m and complete factoring status. `http://www.prothsearch.net/fermat.html`. last checked: 17.11.2016.

[LN86] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, 1986.

[MD99] Zachary S. McGregor-Dorsey. Methods of Primality Testing. *MIT Undergraduate Journal of Mathematics*, pages 133–141, 1999.

[mer] Great Internet Mersenne Prime Search. `http://www.mersenne.org`. last checked: 17.11.2016.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. System Sci.*, 13(3):300–317, 1976.

[Mil06] J.S. Milne. *Elliptic Curves*. BookSurge Publishers, 2006.

[MO] Rolf Möhring and Martin Oellrich. Das Sieb des Eratosthenes: Wie schnell kann man eine Primzahl berechnen? `https://prof.beuth-hochschule.de/fileadmin/user/oellrich/eratosthenes.pdf`. last checked: 17.11.2016.

[Nai82] M. Nair. On Chebychev-type inequalities for primes. *Amer. Math. Monthly*, 89:126–129, 1982.

[Nic12] Thomas R. Nicely. The Baillie-PSW primality test. `http://www.trnicely.net/misc/bpsw.html`, 2012. last checked: 17.11.2016.

[Pin] Richard G.E. Pinch. The Carmichael Numbers up to 10^{21}. `http://math.ucalgary.ca/ants/files/ants/pinch.pdf`. last checked: 17.11.2016.

[Pom02] Carl Pomerance. PRIMALITY TESTING: VARIATIONS ON A THEME OF LUCAS, 2002. lecture notes.

[Rib11] Paulo Ribenboim. *Die Welt der Primzahlen*. Springer, 2 edition, 2011.

[Ros86] Kenneth H. Rosen. *Elementary Number Theory and its Applications*. Addison-Wesley Publishing Company, 1986.

[Sch85] René Schoof. Elliptic Curves Over Finite Fields and the Computation of Square Roots $\mod p$. *Mathematics of Computation*, 44(170):483–494, 1985.

[Sch14] René Schoof. Four primality testing algorithms, 2014.

[vzGG99] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[Wei] Steven H. Weintraub. Several Proofs of the Irreducibility of the Cyclotomic Polynomials. `http://www.lehigh.edu/~shw2/c-poly/several_proofs.pdf`. last checked: 17.11.2016.

[Wik] Wikipedia. Schoof's Algorithm. `https://en.wikipedia.org/wiki/Schoof\%27s_algorithm`. last checked: 17.11.2016.