ABSTRACT

Title of dissertation:	ON SOLVING UNIVARIATE POLYNOMIAL EQUATIONS OVER FINITE FIELDS AND SOME RELATED PROBLEMS
	Tsz Wo Sze, Doctor of Philosophy, 2007
Dissertation directed by:	Professor Lawrence C. Washington Department of Mathematics

In this thesis, we are mainly interested in constructing *deterministic polynomial-time algorithms* for solving some computational problems that arise in number theory and cryptography. The problems we are interested in include finite field arithmetic, primality testing, and elliptic curve arithmetic.

We first present a novel idea to compute square roots over some families of finite fields. Our square root algorithm is deterministic polynomial-time and can be proved by elementary means. The approach of taking square roots is generalized to take nth roots. Then, we present a deterministic polynomial-time algorithm to solve polynomial equations over some families of finite fields. As applications, we construct a deterministic polynomial-time primality test for some forms of integers and show a deterministic polynomial-time algorithm computing elliptic curve "nth roots".

For example, we prove the following statements. Denote a finite field with q elements and characteristic p by $\mathbb{F}_q$.

(I) Suppose $p \equiv 1 \pmod{12}$, $q = 2^e 3^f t + 1$ for some $e, f \geq 1$ and some $t = O(\text{poly}(\log q))$. There is a deterministic polynomial time algorithm taking square roots over $\mathbb{F}_q$.

(II) Let $r_1^{e_1} \cdots r_m^{e_m}$ be the prime factorization of $q-1$. Suppose $r_j = O(\text{poly}(\log q))$ and a primitive r_jth root of unity can be computed efficiently for $1 \leq j \leq m$. There is a deterministic polynomial time algorithm solving any polynomial equation with degree $O(\text{poly}(\log q))$ over $\mathbb{F}_q$.

(III) Let $N = r^e t + 1$ for some prime r and some positive integers t and e with $r^e > t$. There is an $\tilde{O}(r^2(\log^2 N)(t + r \log N))$ deterministic primality testing algorithm. If r is a small constant and $t = O(\log N)$, the running time is $\tilde{O}(\log^3 N)$.

ON SOLVING UNIVARIATE POLYNOMIAL EQUATIONS OVER FINITE FIELDS AND SOME RELATED PROBLEMS

by

Tsz Wo Sze

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:

Professor Clyde P. Kruskal, Chair
Professor Lawrence C. Washington
Professor Jonathan Katz
Professor William Gasarch
Professor David Mount

Dedicated to Yuen-King Lau

Acknowledgments

This work would not be possible without the guidance provided from my adviser, Lawrence Washington. The most critical step of my Ph.D. study at the University of Maryland was meeting him in my first semester. Although he is a very intelligent and knowledgeable mathematician, he is exceptionally patient with his students. He has constantly given me non-trivial insights for solving problems and provided thoughtful comments on my work. I also deeply admire his personality. He is well-known to be one of the best professors, not only in the Department of Mathematics, but also the Department of Computer Science. I have learned a lot from him, from number theory to being a man.

Jonathan Katz has inspired me. I thank him for many useful discussions in cryptography. He also has recommended interesting problems to me and has carefully reviewed my work.

I am grateful to have Clyde Kruskal as my mentor. He has helped me a lot in my study and shown me different realms of knowledge, including the areas in mathematics, computer science and economics.

It was my luck to work with Nelson Padua-Pérez and Chau-Wen Tseng in teaching computer programming for several semesters. I highly respect their enthusiasm and commitment to their students.

I thank Amol Deshpande for serving on my preliminary examination committee, David Mount for serving on my dissertation committee and William Gasarch for serving on both committees.

At last, I would like to thank the following people and organizations (in alphabetical order) for providing high quality resources and making them freely available on the Internet.

`cr.yp.to` by Daniel Bernstein

MathWorld by Wolfram Research

PARI/GP

PlanetMath

The Prime Pages (`primes.utm.edu`) by Chris Caldwell

WikipediA

Table of Contents

List of Symbols

(a, b)	:	the greatest common divisor of a and b
$[a]$	:	an element in G_α (see below), where a is in some finite field
$\mathbb{C}$	:	the set of complex numbers
$E(F)$	:	the set of points on the curve E over a field F
$E_{ns}(F)$	:	the set of non-singular points of the curve E over a field F
$\mathbb{F}_q$	:	the finite field with q elements
$\overline{F}$	:	a fixed algebraic closure of a finite field F
G_α	:	a special group defined in Section 2.3
$\mathrm{ord}_a(b)$	:	the order of b in $(\mathbb{Z}/a\mathbb{Z})^\times$, given $(a, b) = 1$
$\Phi_n(x)$	:	the nth cyclotomic polynomial
$\Re(s)$	:	the real part of a complex number s
$\mathbb{Z}$	:	the set of integers
ζ_n	:	a fixed primitive nth root of unity

Chapter 1

Introduction

In this thesis, we are mainly interested in constructing *deterministic polynomial-time algorithms* for solving some computational problems that arise in number theory and cryptography. The problems we are interested in include finite field arithmetic, primality testing and elliptic curve arithmetic. We give a short overview of this thesis in Section 1.1. See [61], [60] and [62] for the related papers. Following the overview, we state some useful facts in the rest of this section.

1.1 Overview

The main problem we consider in this thesis is the problem of solving polynomial equations over finite fields. Let $\mathbb{F}_q$ denote a finite field with q elements. Let $f(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_0 \in \mathbb{F}_q[x]$ be a polynomial with $a_i \in \mathbb{F}_q$ for all i and $a_d \neq 0$. We assume $\deg f \stackrel{\mathsf{def}}{=} d = O(\mathrm{poly}(\log q))$. Then, the problem is to find the solutions of

$$f(x) = 0 \qquad \text{over } \mathbb{F}_q. \tag{1.1.1}$$

The input parameters are $\mathbb{F}_q$ and $f(x)$. Since there are d coefficients in f, the input size is $O(d \log q) = O(\text{poly}(\log q))$. There are at most d solutions for equation (1.1.1), therefore, the output size is also $O(d \log q) = O(\text{poly}(\log q))$.

Interestingly, there are efficient probabilistic algorithms, for example Berlekamp's algorithm [13], for this problem. These probabilistic algorithms work very well in practice. However, there is no known deterministic polynomial-time algorithm in the literature, even for solving quadratic equations over finite fields. For more background information about solving polynomial equations, see the introduction in Chapter 4.

Our main contribution is showing a deterministic algorithm to solve $f(x) = 0$ over $\mathbb{F}_q$ (equation (1.1.1)) for arbitrary f. The proof is totally elementary and does not assume any unproven hypotheses. The algorithm is polynomial-time if $\mathbb{F}_q$ is a finite field such that for each prime factor r of $q-1$, r is small and a primitive rth root of unity can be computed efficiently. We have the following theorem.[1]

Theorem 1.1.1. *Let $r_1^{e_1} \cdots r_m^{e_m}$ be the prime factorization of $q-1$. Suppose $r_j = O(\text{poly}(\log q))$ and a primitive r_jth root of unity can be computed efficiently for $1 \leq j \leq m$. There is a deterministic polynomial time algorithm solving any polynomial equation with degree $O(\text{poly}(\log q))$ over $\mathbb{F}_q$.*

The algorithm for solving polynomial equations is presented in Section 4.1. The algorithm relies on an algorithm for taking roots, which will be discussed in the next section.

[1] Theorem 1.1.1 and Theorem 4.1.5 are the same.

1.1.1 Taking Square Roots and rth Roots

In Chapter 2, we present an algorithm to take square roots. Clearly, the square root problem is equivalent to the problem of solving quadratic equations. For arbitrary finite fields $\mathbb{F}_q$ and arbitrary $\beta \in \mathbb{F}_q$, there is no known deterministic polynomial-time algorithm computing the square roots of β.

For arbitrary prime fields $\mathbb{F}_p$ and β with small absolute value, the square roots of β can be computed by Schoof's square root algorithm [55], which is deterministic polynomial-time. However, Schoof's algorithm becomes exponential-time for β with large absolute value.

Square roots over finite fields can be computed by probabilistic algorithms such as Tonelli-Shanks [63, 56]. Some of these probabilistic algorithms becomes deterministic polynomial-time if a quadratic nonresidue is given as an additional input. However, there is no known deterministic polynomial-time algorithms to find a quadratic nonresidue. Probabilistic algorithms for finding a quadratic nonresidue work very well in practice because half of the non-zero elements in a finite field[2] are quadratic nonresidues. For prime fields, a quadratic nonresidue exists in a small range if the Riemann Hypothesis is true (see [7]). For other results on the square root problem, see the introduction in Chapter 2.

We present a novel idea to compute square roots over some families of finite fields. The problem of taking square root of β with arbitrary size β is first reduced to the problem of constructing a primitive rth root of unity, ζ_r. In some cases, ζ_r can be

[2]We assume the characteristic of the field is odd since if the characteristic is even, the square root problem is easy.

constructed by taking square roots of some small elements $b_i \in \mathbb{F}_q$, which can be done by Schoof's algorithm. In some other situations, ζ_r can be constructed directly. One family of finite fields is described in the theorem[3] below. More generally, if all prime factors of $q-1$ are small and a primitive kth root of unity can be computed efficiently for certain factors k of $q-1$, then our square root algorithm is applicable to $\mathbb{F}_q$. See Section 2.1 for the other results.

Theorem 1.1.2. *Let $p \equiv 1 \pmod{12}$ be a prime and $q = p^n = 2^e 3^f t + 1$ for some $n, e, f \geq 1$. Suppose $t = O(\text{poly}(\log q))$. Both taking square roots in $\mathbb{F}_q$ and finding a quadratic nonresidue in $\mathbb{F}_q$ can be computed in deterministic polynomial time.*

In addition, we discuss how to construct primitive rth roots of unity, ζ_r, in Chapter 2. The interesting cases are $r|q-1$ and either $r = 4$ or r an odd prime. We summarize the cases that ζ_r can be constructed in deterministic polynomial time below. Let p be the characteristic of $\mathbb{F}_q$.

- $r = 4$ and $p \equiv 1 \pmod{4}$.
- $r = 3$ and -3 is a square mod p.
- $r = 2 \cdot 3^k + 1$ for some $k \geq 1$, $p \equiv 1 \pmod{r}$ and $p \equiv 13, 25 \pmod{36}$.
- $q = r^e t + 1$ with $r + t = O(\text{poly}(\log q))$.

In Chapter 3, the idea of our square root algorithm is generalized to take rth roots for some positive integer r with $r = O(\text{poly}(\log q))$. The requirements of the finite fields are similar to taking square roots.

[3]Theorem 1.1.2 and Theorem 2.1.1 are the same.

1.1.2 Applications

Based on our rth root algorithm, we construct a deterministic polynomial-time primality test for some forms of integers in Section 5.1. We have the following theorem[4].

Theorem 1.1.3. *Let $N = r^e t + 1$ for some prime r and some positive integers t and e with $r^e > t$. There is an $\tilde{O}(r^2(\log^2 N)(t + r \log N))$ deterministic primality testing algorithm. If r is a small constant and $t = O(\log N)$, the running time is $\tilde{O}(\log^3 N)$.*

In Section 4.2, we construct a deterministic polynomial-time algorithm to compute elliptic curve "n th root" by our algorithm for solving polynomial equations,

1.1.3 Other Works

In Section 3.7, we show a deterministic polynomial-time algorithm to compute a non-trivial factor of the r^2th cyclotomic polynomial $\Phi_{r^2}(x)$ over a finite field for some prime r. An efficient probabilistic algorithm for solving polynomial equations is presented in Section 4.3. We discuss a potentially fast primality test in Section 5.2.

1.2 Useful Facts

We state some useful facts in this section. Most of the proofs will be skipped. For the topics in algorithms, see [36] and [25]. For background material about number theory and computational number theory, see [32], [22] and [57]. For information about finite fields and abstract algebra, see [42] and [8]. For material about elliptic curves, see [69] and [58]. For the topics in cryptography, see [65] and [38].

[4]Theorem 1.1.3 and Theorem 5.1.1 are the same.

1.2.1 Algorithms

Deterministic algorithms always produce the same correct output. In contrast, *probabilistic algorithms* may sometimes give incorrect output or fail to give an output at all. An algorithm is *polynomial-time* if its *running time* is polynomial in the input size[5]. Running time means the number of operations required by the algorithm in order to finish the computation.

All running times in this thesis are measured in term of bit operations. We ignore logarithmic factors in running time and adopt the $\tilde{O}(\cdot)$ notation. For example, the running time of the Schönhage-Strassen integer multiplication algorithm [54] is $O(n \log n \log\log n)$, which will be denoted by $\tilde{O}(n)$.

Arithmetic Operations

Clearly addition and subtraction over integers can be done in linear time (i.e. $O(n)$, where n is the input size[6]). Multiplication, division with remainder and computing the greatest common divisor (GCD) over integers can be implemented by fast Fourier transform (FFT) and other fast methods in essentially linear time (i.e. $\tilde{O}(n)$). See [28], [35], [54], [64] and [24].

The case is similar when the operations are carried over finite fields and polynomial rings: addition and subtraction are $O(n)$; multiplication, division and computing GCD are $\tilde{O}(n)$. See [29] and [68].

[5]Output size is ignored since the output size is always polynomial in the input size for all the problems in considered this thesis.

[6]For a positive integer N, we have $n = O(\log N)$ because N can be represented by $O(\log N)$ bits.

For all cases, exponentiation a^e can be computed by *successive squaring*. It requires $O(\log e)$ multiplications.

1.2.2 Finite Fields

A *finite field* (also called Galois field) has finitely many elements. We denote a finite field with q elements by $\mathbb{F}_q$. The number of elements in $\mathbb{F}_q$ is always a prime power $q = p^n$ for some prime p and positive integer n. Conversely, there is a unique (up to isomorphism) finite field $\mathbb{F}_q$ for any prime power $q > 1$. The prime p is called the *characteristic* of $\mathbb{F}_q$. If $d|n$, then $\mathbb{F}_{p^d}$ is a subfield of $\mathbb{F}_{p^n}$. The subfield $\mathbb{F}_p$ is called *the prime field of* $\mathbb{F}_q$ or *a prime field*. A prime field is unique up to unique isomorphism. Note that $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ but $\mathbb{F}_{p^n} \not\simeq \mathbb{Z}/p^n\mathbb{Z}$ for $n > 1$.

Quadratic Residues and Nonresidues

In $\mathbb{F}_q$, we have

$$a^q = a \qquad \text{for all } a \in \mathbb{F}_q.$$

It is a generalization of Fermat's Little Theorem stated below.

Theorem 1.2.1. (Fermat's Little Theorem) *For any prime p,*

$$a^p \equiv a \pmod{p} \qquad \textit{for any integer } a. \tag{1.2.1}$$

If q is even (equivalently, the characteristic of $\mathbb{F}_q$ is 2), then for any element $a \in \mathbb{F}_q$, $a = (a^{q/2})^2$. The element $a^{q/2}$ is a square root of a. Therefore, taking square roots over $\mathbb{F}_q$ is trivial for q even.

Suppose q is odd. The multiplicative group $\mathbb{F}_q^\times$ has $q-1$ elements. Since $\mathbb{F}_q^\times$ is cyclic, there is a generator $g \in \mathbb{F}_q^\times$ such that $\mathbb{F}_q^\times = \langle g \rangle = \{1, g, g^2, \cdots, g^{q-2}\}$. The element g is also called *a primitive element* of $\mathbb{F}_q$. Squaring the elements of $\mathbb{F}_q^\times$, we obtain the set of all squares

$$\mathsf{QR} \stackrel{\mathsf{def}}{=} \left\{1, g^2, g^4, \cdots, g^{q-3}\right\}.$$

Since the product of two squares is a square and the inverse of a square is a square, QR is a subgroup of $\mathbb{F}_q^\times$. Let

$$\mathsf{NR} \stackrel{\mathsf{def}}{=} \mathbb{F}_q^\times \setminus \mathsf{QR} = \left\{g, g^3, g^5, \cdots, g^{q-2}\right\} = g\mathsf{QR}.$$

For any element $a \in \mathsf{NR}$, the square root of a does not exist in $\mathbb{F}_q$. The elements in QR and the elements in NR are called *quadratic residues* and *quadratic nonresidues*. We show in Theorem 1.2.2 below that the least[7] quadratic nonresidue in a prime field is a prime. We will discuss the problems of finding a quadratic nonresidue and taking square roots over finite fields in Chapter 2.

Theorem 1.2.2. *Let $p > 2$ be a prime. The least quadratic nonresidue in $\mathbb{F}_p$ is a prime.*

Proof. Let a be the least quadratic nonresidue. Suppose $a = uv$ with $1 < u, v < a$. Since a is the least quadratic nonresidue, u and v must be quadratic residues. However, the product uv is also a quadratic residue, which is a contradiction. □

[7]We consider the elements in $\mathbb{F}_p$ are integers from 0 to $p-1$.

Enumerating The Elements

In some algorithms given in the later chapters, a simple mechanism for enumerating a small portion of the elements in $\mathbb{F}_q^\times$ is required. In addition, we require the enumeration satisfying equation (1.2.3) below. For completeness, we illustrate a method to obtain an enumeration of half of the elements in any finite field $\mathbb{F}_q$ with $q = p^n$.

The finite field $\mathbb{F}_q$ can be viewed as a *vector space* over $\mathbb{F}_p$ with dimension n. Let $\{T_0, T_1, \cdots, T_{n-1}\}$ be a *basis* of $\mathbb{F}_q$ over $\mathbb{F}_p$. For $1 \leq m \leq \frac{q-1}{2}$, choose k such that $\frac{p^k-1}{2} < m \leq \frac{p^{k+1}-1}{2}$. Write $m - 1 - \frac{p^k-1}{2} = a_0 + a_1 p + \cdots + a_{k-1}p^{k-1} + a_k p^k$ with $0 \leq a_k < \frac{p-1}{2}$ and $0 \leq a_j < p$ for $0 \leq j < k$. The mth element of $\mathbb{F}_q$ in our enumeration is defined to be

$$a_0 T_0 + a_1 T_1 + \cdots + a_{k-1} T_{k-1} + (a_k + 1) T_k, \tag{1.2.2}$$

Let $\mathsf{element}(m)$ be the procedure returning the mth element in $\mathbb{F}_q$. It is easy to see that

$$\mathsf{element}(m) \neq -\mathsf{element}(j) \qquad \text{for } 1 \leq j, m \leq \frac{q-1}{2}. \tag{1.2.3}$$

1.2.3 Elliptic Curves

An *elliptic curve* E defined over a finite field F can be represented by the Weierstrass equation

$$E : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \qquad \text{for some } a_1, a_2, a_3, a_4, a_6 \in F.$$

Denote the points on E with coordinates in F by $E(F)$. Interestingly, $E(F)$ is a well-defined group, which has a lot of applications in number theory and cryptogra-

phy. The applications include primality proving [9], integer factoring [33], public key cryptography [37] and identity based encryption [17].

Computing Points

Fix the x-coordinate to $x_0 \in F$. If the quadratic equation

$$f(y) = y^2 + (a_1x_0 + a_3)y - (x_0^3 + a_2x_0^2 + a_4x_0 + a_6) = 0$$

has solutions in F, say y_1 and y_2, then (x, y_1) and (x, y_2) are points on $E(F)$. Similarly, fixing the y-coordinate to some $y_0 \in F$, if the cubic equation,

$$g(x) = x^3 + a_2x^2 + (a_4 - a_1y_0)x + a_6 - a_3y_0 - y_0^2 = 0$$

has solutions $x_1, x_2, x_3 \in F$, then (x_1, y_0), (x_2, y_0) and (x_3, y_0) are points on $E(F)$.

In either case, we need to solve a polynomial equation over F. It is obvious that solving quadratic equations and taking square roots are equivalent problems. By some algebraic manipulation, for example Cardano's method, cubic equations can be solved by taking square roots and cubic roots. We will describe how to take rth roots over finite fields in Chapter 3 and how to solve polynomial equations over finite fields in Chapter 4. In addition, the elliptic curve "nth root" problem will be discussed in Chapter 4.

1.2.4 Riemann Hypothesis

The Riemann Hypothesis (RH) is one of the most important open problems in mathematics. Most mathematicians believe RH is true. RH has a deep connection to

the distribution of prime numbers. Occasionally, there are results in computational number theory that assume RH or its generalizations. One typical example is Miller's primality test [44]. RH is not in our scope of study. We informally describe RH and two common generalizations below.

The *Riemann zeta-function* over the complex numbers is defined as

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} \qquad \text{for } s \in \mathbb{C}, \quad \Re(s) > 1 \tag{1.2.4}$$

and then analytically continued to all $s \neq 1$. The function $\zeta(s)$ has *trivial zeros* at the negative even integers -2, -4, $\cdots$. The Riemann Hypothesis, introduced in 1859, states that the non-trivial zeros of $\zeta(s)$ have real part equal to $1/2$ (i.e. $\Re(s) = 1/2$). See [52].

Extended Riemann Hypothesis

The *Dirichlet L-function* over the complex numbers is defined as

$$L(\chi, s) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s} \qquad \text{for } s \in \mathbb{C}, \quad \Re(s) > 1 \tag{1.2.5}$$

and then analytically continued to all s, where χ is a non-trivial Dirichlet character. The *Extended Riemann Hypothesis* (ERH) says that for any non-trivial Dirichlet character χ and any $s \in \mathbb{C}$ with $0 < \Re(s) \leq 1$, if $L(\chi, s) = 0$, then $\Re(s) = 1/2$.

Generalized Riemann Hypothesis

Let K be a number field and $\mathcal{O}_K$ be the set of algebraic integers in K. The *Dedekind zeta-function of* K is defined as

$$\zeta_K(s) = \sum_{\mathcal{I} \neq 0} \frac{1}{(N\mathcal{I})^s} \qquad \text{for } s \in \mathbb{C}, \quad \Re(s) > 1 \tag{1.2.6}$$

and then analytically continued to all $s \neq 1$, where $\mathcal{I}$ runs through all non-zero ideals of $\mathcal{O}_K$ and N denotes the norm function such that $N\mathcal{I} \stackrel{\mathsf{def}}{=} [\mathcal{O}_K : \mathcal{I}]$. The *Generalized Riemann Hypothesis* (GRH) states that for any $s \in \mathbb{C}$ with $0 < \Re(s) \leq 1$, if $\zeta_K(s) = 0$, then $\Re(s) = 1/2$.

Chapter 2

Taking Square Roots

In this chapter, we discuss the square root problem over finite fields. Let $\mathbb{F}_q$ be a finite field with q elements. Suppose q is odd in this chapter. Otherwise, the square root problem is trivial. Let β be a square in $\mathbb{F}_q$. The square root problem over $\mathbb{F}_q$ is to find $\alpha \in \mathbb{F}_q$ such that $\alpha^2 = \beta$, given $\mathbb{F}_q$ and β as inputs. The element α is called *a square root of* β. Note that $-\alpha \in \mathbb{F}_q$ is also a square root of β. Denote a fixed square root of β by $\sqrt{\beta}$ or $\beta^{1/2}$.

The problem of taking square roots over finite fields and the problem of constructing quadratic nonresidues over finite fields are closely related. If one can take square roots, one can compute $(-1)^{1/2} = \sqrt{-1}$, $(-1)^{1/4} = \sqrt{(-1)^{1/2}}$, $(-1)^{1/8} = \sqrt{(-1)^{1/4}}$, $\cdots$, and eventually obtain a quadratic nonresidue because the 2-part of the multiplicative group of the field is finite. Conversely, given a quadratic nonresidue as an input, there are deterministic polynomial time algorithms [63], [56] and [2] for computing square roots.

There is no known deterministic polynomial-time algorithm for constructing quadratic

nonresidues over a general finite field. However, the problem of deciding whether an element is a quadratic nonresidue in a finite field $\mathbb{F}_q$ is easy since, for any non-zero element $a \in \mathbb{F}_q$, a is a quadratic nonresidue if and only if $a^{(q-1)/2} = -1$.

Since the number of quadratic nonresidues is equal to the number of quadratic residues in $\mathbb{F}_q$, one could randomly pick an element $a \in \mathbb{F}_q^\times$ and then test whether a is a quadratic nonresidue by checking $a^{(q-1)/2} = -1$ in $\mathbb{F}_q$. Such simple strategy gives an efficient probabilistic algorithm for finding a quadratic nonresidue in $\mathbb{F}_q$.

There are several efficient probabilistic algorithms for taking square roots in finite fields. Tonelli-Shanks [63, 56], Adleman-Manders-Miller [2] and Cipolla-Lehmer [20, 40] require a quadratic nonresidue as an input. Berlekamp-Rabin [14, 50] takes square roots by polynomial factoring over finite fields. The idea of Peralta [51] is similar to Berlekamp-Rabin. For other results, see [10], [11], [12], [15], [18], [43], [45] [46] and [66].

For the following, let $\mathbb{F}_p$ be the finite field with p elements for some odd prime p.

By assuming the ERH (see Section 1.2.4), Ankeny [7] showed that the least quadratic nonresidue in $\mathbb{F}_p$ is less than $c \log^2 p$ for some constant c. It leads to a deterministic polynomial time algorithm for finding the least quadratic nonresidue in $\mathbb{F}_p$. The least quadratic nonresidue must be a prime (see Theorem 1.2.2). Since the problem of deciding quadratic nonresidues is easy, one could evaluate the Legendre symbol $\left(\frac{r}{p}\right) \equiv r^{(p-1)/2} \pmod{p}$ with the primes $r = 2, 3, 5, 7, \cdots$ until a quadratic nonresidue is found. Such quadratic nonresidue must be the least one.

Given β a square in $\mathbb{F}_p$, Schoof [55] showed a deterministic algorithm for computing

square roots of β in $\mathbb{F}_p$ with running time $O((|\beta|^{1/2+\epsilon}\log p)^9)$ bit operations[1] for all $\epsilon > 0$. Thus, his algorithm is polynomial time with any constant β.

We show below that a quadratic nonresidue in $\mathbb{F}_p$ can be computed in deterministic polynomial time for primes p with $p \not\equiv 1 \pmod{240}$. Let ζ_r be a primitive rth of unity. If $p \not\equiv 1 \pmod{16}$, at least one of

$$\zeta_2 = -1, \qquad \zeta_4 = \pm\sqrt{-1}, \qquad \zeta_8 = \pm\frac{1}{\sqrt{2}}(1 \pm \sqrt{-1})$$

is a quadratic nonresidue. Therefore, a quadratic nonresidue can be computed by Schoof's algorithm in this case. Suppose $p \equiv 1 \pmod 4$ for the following. If $p \equiv 2 \pmod 3$, then $\left(\frac{3}{p}\right) = \left(\frac{p}{3}\right) = \left(\frac{2}{3}\right) = -1$. So 3 is a quadratic nonresidue. Similarly, if $p \equiv 2, 3 \pmod 5$, then 5 is a quadratic nonresidue. Suppose $p \equiv 4 \pmod 5$. In this case, 5 is a square mod p. Let

$$\zeta_5 = \frac{a + \sqrt{a^2 - 4}}{2}, \qquad \text{where } a = \frac{-1 + \sqrt{5}}{2}. \tag{2.0.1}$$

Then, ζ_5 is a primitive 5th root of unity. Note that $a \in \mathbb{F}_p$ but $\zeta_5 \notin \mathbb{F}_p$. Therefore, $a^2 - 4$ must be a quadratic nonresidue. In conclusion, the problem of finding a quadratic nonresidue in $\mathbb{F}_p$ is hard only if $p \equiv 1 \pmod{16}$, $p \equiv 1 \pmod 3$ and $p \equiv 1 \pmod 5$, which is $p \equiv 1 \pmod{240}$.

We present our main results and the idea behind them in Section 2.1 and Section 2.2, respectively. In Section 2.3, we construct a special group and discuss the computation of the operations in that group. In Section 2.4, we provide algorithms for taking square roots and finding quadratic nonresidues in finite fields.

[1] $|\beta|$ denotes the absolute value of β, where β is considered as an integer in $(-\frac{p-1}{2}, \frac{p-1}{2}]$.

2.1 Main Results

We show deterministic polynomial time algorithms (without any unproven assumption) for computing square roots and finding quadratic nonresidues in some families of finite fields as stated in the following theorems. In some particular finite fields $\mathbb{F}_q$, there are algorithms for taking square roots with $\tilde{O}(\log^2 q)$ bit operations.

Theorem 2.1.1. *Let $p \equiv 1 \pmod{12}$ be a prime and $q = p^n = 2^e 3^f t + 1$ for some $n, e, f \geq 1$. Suppose $t = O(\mathrm{poly}(\log q))$. Both taking square roots in $\mathbb{F}_q$ and finding a quadratic nonresidue in $\mathbb{F}_q$ can be computed in deterministic polynomial time.*

Theorem 2.1.2. *Let p be a prime with $p \equiv 13, 25 \pmod{36}$. Let $r_1, r_2, \cdots, r_m$ be m distinct primes, where $r_j = 2 \cdot 3^{k_j} + 1 < M$ with $k_j \geq 0$ for some upper bound M and $j = 1, 2, \cdots, m$. Let $q = p^n = 2^e r_1^{f_1} r_2^{f_2} \cdots r_m^{f_m} t + 1$ for some $n, e, f_1, f_2, \cdots, f_m \geq 1$. Suppose $p \equiv 1 \pmod{r_1 r_2 \cdots r_m}$ and $M + t = O(\mathrm{poly}(\log q))$. Both taking square roots in $\mathbb{F}_q$ and finding a quadratic nonresidue in $\mathbb{F}_q$ can be computed in deterministic polynomial time.*

Theorem 2.1.3. *Let p, r be primes and $q = p^n = r^e t + 1$ for some $n, e \geq 1$. Suppose $r + t = O(\mathrm{poly}(\log q))$. Both taking square roots in $\mathbb{F}_q$ and finding a quadratic nonresidue in $\mathbb{F}_q$ can be computed in deterministic polynomial time.*

2.2 The Idea of Using Group Isomorphism

Let H be a cyclic group. Let G be another group such that G is isomorphic to H. However, we do not know the exact isomorphism, i.e. the isomorphism formula

contains an unknown parameter. Denote the isomorphism from G to H by ψ. Pick a non-identity element g in G. Let d be the order of g. Then, $\psi(g)$ is an order d element in H. Suppose we can compute an element $\zeta \in H$ with order d. We must have

$$\psi(g) = \zeta^j \qquad \text{for some } j \in (\mathbb{Z}/d\mathbb{Z})^\times.$$

since the group H is cyclic. If ψ is a simple formula and d is small, we can recover ψ by trying each possible j.

We further elaborate the group isomorphism idea for computing square root below. Let $\mathbb{F}_q$ be a finite field with q elements. Suppose $\beta \in \mathbb{F}_q^\times$ is a square. Then,

$$\alpha^2 = \beta \qquad \text{for some } \alpha \in \mathbb{F}_q^\times.$$

Let G_α be a group with the following properties:

(i) the group operation in G_α is efficiently computable with β but without the knowledge of α,

(ii) G_α is isomorphic to the multiplicative group $\mathbb{F}_q^\times$,

(iii) the isomorphism $\psi_\alpha : G_\alpha \to \mathbb{F}_q^\times$ depends on α as a parameter.

Since the isomorphism ψ_α depends on α while the value of α is unknown, ψ_α and its inverse are not at first efficiently computable. We try to match certain elements in G_α with the corresponding elements in $\mathbb{F}_q^\times$. In the cases we consider, a matched pair reveals the isomorphism ψ_α, and therefore α is obtained.

Let r be an odd prime factor of $q-1$. Then, $q = r^e t + 1$ for some $t, e > 0$ with $(t, r) = 1$. Denote the elements of G_α as $[g]$. Suppose the order of $[g]$ is rs for some

$s > 0$. The order of the element $[a] = [g]^s$ is r. Note that there are $(r^e - 1)t$ possible elements of $[g]$ leading to an element $[a]$ with order r.

Let ζ_r be a primitive rth of unity in $\mathbb{F}_q$ and suppose ζ_r could be computed efficiently. For some $0 < j < r$, the element $[a]$ with order r must be matched up with ζ_r^j, i.e.

$$\psi_\alpha([a]) = \zeta_r^j,$$

since $\mathbb{F}_q^\times$ is cyclic. If both the value of $[a]$ and the value of ζ_r^j are known, the parameter α of ψ_α can be computed. Suppose r is small. For $j = 1, 2, \cdots, r-1$, compute $\alpha = \alpha_j$ from ψ_α, $[a]$, and ζ_r^j. Check whether $\alpha_j^2 = \beta$. Eventually, the square roots of β are obtained.

2.3 A Special Group

Let $\mathbb{F}_q$ be a finite field with q odd. Define the set

$$G'_\alpha \stackrel{\mathsf{def}}{=} \{[a] \ : \ a \in \mathbb{F}_q, a \neq \pm\alpha\} \qquad \text{for some } \alpha \in \mathbb{F}_q^\times.$$

For distinguishing the elements in G'_α and the elements in $\mathbb{F}_q$, we denote the former by $[\,\cdot\,]$. The number of elements in G'_α is $q-2$. By adding the element $[\infty]$ to G'_α, we obtain

$$G_\alpha \stackrel{\mathsf{def}}{=} G'_\alpha \cup \{[\infty]\}\,.$$

Define an operator $*$ in G_α as following: $\forall\,[a] \in G_\alpha$ and $\forall\,[a_1], [a_2] \in G'_\alpha$ with $a_1 + a_2 \neq 0$,

$$\begin{aligned} [a] * [\infty] &= [\infty] * [a] = [a], && (2.3.1)\\ [a_1] * [-a_1] &= [\infty], && (2.3.2)\\ [a_1] * [a_2] &= \left[\frac{a_1 a_2 + \alpha^2}{a_1 + a_2}\right]. && (2.3.3) \end{aligned}$$

Interestingly, $(G_\alpha, *)$ is a well-defined group, which is isomorphic to the multiplicative group $\mathbb{F}_q^\times$. The group G_α provides a new computational point of view of the group $\mathbb{F}_q^\times$. We will use G_α to construct our square root algorithm later.

Theorem 2.3.1. *$(G_\alpha, *)$ is an abelian group with identity $[\infty]$. The group G_α is isomorphic to $\mathbb{F}_q^\times$.*

Proof. Define a bijective mapping

$$\psi : G_\alpha \longrightarrow \mathbb{F}_q^\times, \qquad [\infty] \longmapsto 1, \qquad [a] \longmapsto \frac{a+\alpha}{a-\alpha} \tag{2.3.4}$$

with inverse

$$\psi^{-1} : \mathbb{F}_q^\times \longrightarrow G_\alpha, \qquad 1 \longmapsto [\infty], \qquad b \longmapsto \left[\frac{\alpha(b+1)}{b-1}\right]. \tag{2.3.5}$$

A straightforward calculation shows that ψ is a homomorphism. □

Note that G_α is cyclic since $\mathbb{F}_q^\times$ is. Since q is odd, there is a unique order 2 element in $\mathbb{F}_q^\times$ or G_α. Clearly, -1 is the order 2 element in $\mathbb{F}_q^\times$. For any $\alpha \in \mathbb{F}_q^\times$, we have

$$\psi([0]) = \frac{0+\alpha}{0-\alpha} = -1.$$

Therefore, $[0]$ is the only order 2 element in G_α, independent of the choice of α.

2.3.1 The Power Formulas in G_α

Denote the power of an element in G_α by

$$[a]^k \stackrel{\mathsf{def}}{=} \underbrace{[a] * [a] * \cdots * [a]}_{k} \qquad \text{for all } [a] \in G_\alpha, k > 0.$$

We have the following formula for computing $[a]^k$.

Lemma 2.3.2. *Let $[a] \in G'_\alpha$ and $k > 0$. If the order of $[a]$ does not divide k, then*

$$[a]^k = \left[\alpha \cdot \frac{(a+\alpha)^k + (a-\alpha)^k}{(a+\alpha)^k - (a-\alpha)^k}\right] = \left[\frac{a^k + \binom{k}{2}a^{k-2}\alpha^2 + \cdots}{ka^{k-1} + \binom{k}{3}a^{k-3}\alpha^2 + \cdots}\right]. \tag{2.3.6}$$

Proof. If the order of $[a]$ does not divide k, then $\psi([a])^k \neq 1$. We have

$$\begin{aligned}
[a]^k &= \psi^{-1}(\psi([a])^k) \\
&= \left[\alpha \cdot \frac{\psi([a])^k + 1}{\psi([a])^k - 1}\right] \\
&= \left[\alpha \cdot \frac{(a+\alpha)^k + (a-\alpha)^k}{(a+\alpha)^k - (a-\alpha)^k}\right].
\end{aligned}$$

The last equality in equation (2.3.6) can be obtained by expanding $(a \pm \alpha)^k$. □

Define the following polynomials in $\mathbb{F}_q[x]$ for $k \geq 0$,

$$\gamma_k(x) = \frac{(x+\alpha)^k + (x-\alpha)^k}{2} = \sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{2j} x^{k-2j}\alpha^{2j}, \tag{2.3.7}$$

$$\Psi_k(x) = \frac{(x+\alpha)^k - (x-\alpha)^k}{2\alpha} = \sum_{j=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k}{2j+1} x^{k-1-2j}\alpha^{2j}. \tag{2.3.8}$$

Note that $\gamma_k(x), \Psi_k(x) \in \mathbb{F}_p[\alpha^2][x]$. The polynomials γ_k and Ψ_k can be defined recursively.

Lemma 2.3.3. *For $k \geq 0$, we have the following recursion equations:*

$$\gamma_{k+1} = x\gamma_k + \alpha^2\Psi_k, \tag{2.3.9}$$

$$\Psi_{k+1} = \gamma_k + x\Psi_k. \tag{2.3.10}$$

Proof. By equations (2.3.7) and (2.3.8),

$$\begin{aligned} x\gamma_k + \alpha^2\Psi_k &= \frac{x}{2}((x+\alpha)^k + (x-\alpha)^k) + \frac{\alpha}{2}((x+\alpha)^k - (x-\alpha)^k) \\ &= \gamma_{k+1}; \\ \gamma_k + x\Psi_k &= \frac{(x+\alpha)^k + (x-\alpha)^k}{2} + x\frac{(x+\alpha)^k - (x-\alpha)^k}{2\alpha} \\ &= \frac{1}{2}(x+\alpha)^k(1+\frac{x}{\alpha}) + \frac{1}{2}(x-\alpha)^k(1-\frac{x}{\alpha}) \\ &= \Psi_{k+1}. \end{aligned}$$

The lemma follows. □

By some algebraic manipulations, the polynomials γ_{2k}, Ψ_{2k}, γ_{2k+1} and Ψ_{2k+1} can be written in terms of γ_k, Ψ_k, γ_{k+1} and Ψ_{k+1} as shown in Lemma 2.3.4. As a consequence, only $O(\log n)$ polynomial multiplications are required for computing γ_n and Ψ_n.

Lemma 2.3.4. *For $k \geq 0$,*

$$\gamma_{2k} = \gamma_k^2 + \alpha^2\Psi_k^2, \tag{2.3.11}$$

$$\Psi_{2k} = 2\gamma_k\Psi_k, \tag{2.3.12}$$

$$\gamma_{2k+1} = \gamma_k\gamma_{k+1} + \alpha^2\Psi_k\Psi_{k+1}, \tag{2.3.13}$$

$$\Psi_{2k+1} = \gamma_k\Psi_{k+1} + \gamma_{k+1}\Psi_k. \tag{2.3.14}$$

Proof. By Lemma 2.3.3,

$$\begin{pmatrix} \gamma_k & \alpha^2\Psi_k \\ \Psi_k & \gamma_k \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix} \begin{pmatrix} \gamma_{k-1} & \alpha^2\Psi_{k-1} \\ \Psi_{k-1} & \gamma_{k-1} \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^{k-j} \begin{pmatrix} \gamma_j & \alpha^2\Psi_j \\ \Psi_j & \gamma_j \end{pmatrix} \tag{2.3.15}$$

for any $k > 0$ and $1 \leq j \leq k$. We have $\begin{pmatrix} \gamma_1 \\ \Psi_1 \end{pmatrix} = \begin{pmatrix} x \\ 1 \end{pmatrix}$. Then,

$$\begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^k = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^{k-1} \begin{pmatrix} \gamma_1 & \alpha^2\Psi_1 \\ \Psi_1 & \gamma_1 \end{pmatrix} = \begin{pmatrix} \gamma_k & \alpha^2\Psi_k \\ \Psi_k & \gamma_k \end{pmatrix}$$

for any $k > 0$. Finally, by equation (2.3.15),

$$\begin{pmatrix} \gamma_{2k} \\ \Psi_{2k} \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^k \begin{pmatrix} \gamma_k \\ \Psi_k \end{pmatrix} = \begin{pmatrix} \gamma_k & \alpha^2\Psi_k \\ \Psi_k & \gamma_k \end{pmatrix} \begin{pmatrix} \gamma_k \\ \Psi_k \end{pmatrix},$$

which implies

$$\begin{aligned} \gamma_{2k} &= \gamma_k^2 + \alpha^2\Psi_k^2, \\ \Psi_{2k} &= 2\gamma_k\Psi_k; \end{aligned}$$

and

$$\begin{pmatrix} \gamma_{2k+1} \\ \Psi_{2k+1} \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^k \begin{pmatrix} \gamma_{k+1} \\ \Psi_{k+1} \end{pmatrix} = \begin{pmatrix} \gamma_k & \alpha^2\Psi_k \\ \Psi_k & \gamma_k \end{pmatrix} \begin{pmatrix} \gamma_{k+1} \\ \Psi_{k+1} \end{pmatrix},$$

which implies

$$\begin{aligned} \gamma_{2k+1} &= \gamma_k\gamma_{k+1} + \alpha^2\Psi_k\Psi_{k+1}, \\ \Psi_{2k+1} &= \gamma_k\Psi_{k+1} + \gamma_{k+1}\Psi_k. \end{aligned}$$

The lemma follows. □

We use the polynomials γ_k and Ψ_k to compute the power of an element $[a]^k$ in G_α. With the recursion equations, $[a]^k$ can be computed efficiently by polynomial operations in $\mathbb{F}_q[x]$. It is not hard to see that the roots of Ψ_d, together with $[\infty]$, are the elements in the *d-torsion subgroup* of G_α.

Proposition 2.3.5. *Let $[a] \in G'_\alpha$. For $d > 0$, $[a]^d = [\infty]$ if and only if $\Psi_d(a) = 0$.*

Proof. Since $[a] \neq [\infty]$, we have $d > 1$ and the order of $[a]$ not dividing $d - 1$. Then,

$$\begin{aligned} [a]^d = [\infty] \quad &\Longleftrightarrow \quad [a]^{d-1} = [-a] && \text{by equation (2.3.2)} \\ &\Longleftrightarrow \quad \frac{\gamma_{d-1}(a)}{\Psi_{d-1}(a)} = -a && \text{by Lemma 2.3.2} \\ &\Longleftrightarrow \quad \Psi_d(a) = 0 && \text{by equation (2.3.10).} \end{aligned}$$

Note that if $\Psi_d(a) = 0$, we have $\Psi_{d-1}(a) \neq 0$. Otherwise, if $\Psi_{d-1}(a) = 0$, equation (2.3.10) implies $\gamma_{d-1}(a) = 0$. Then, $(a+\alpha)^{d-1} = 2\gamma_{d-1}(a) + 2\alpha\Psi_{d-1}(a) = 0$ leads to a contradiction. □

2.3.2 Singular Curves with a Double Root

We can reinterpret the group law in terms of "singular elliptic curves." Consider the curve

$$E : y^2 = x^2(x + \alpha^2).$$

Let $E(\mathbb{F}_q)$ be the points on the curve with coordinates in $\mathbb{F}_q$. The only singular point on $E(\mathbb{F}_q)$ is $(0, 0)$, which is a double root. Let $E_{ns}(\mathbb{F}_q)$ be the non-singular points on $E(\mathbb{F}_q)$. Then, the mapping

$$\tau : E_{ns}(F_q) \to \mathbb{F}_q^{\times}, \qquad \infty \longmapsto 1, \qquad (x, y) \longmapsto \frac{(y/x) + \alpha}{(y/x) - \alpha}$$

is an isomorphism from $E_{ns}(\mathbb{F}_q)$ to $\mathbb{F}_q^{\times}$. The inverse is

$$\tau^{-1} : \mathbb{F}_q^{\times} \to E_{ns}(F_q), \qquad 1 \longmapsto \infty, \qquad \lambda \longmapsto \left(\frac{4\alpha^2\lambda}{(\lambda-1)^2}, \frac{4\alpha^3(\lambda+1)}{(\lambda-1)^3} \right).$$

For proofs and details, see [69] p56 - p59. Together with the isomorphism ψ, we have

$$G_\alpha \simeq \mathbb{F}_q^{\times} \simeq E_{ns}(\mathbb{F}_q).$$

The isomorphism from $E_{ns}(\mathbb{F}_q)$ to G_α is surprisingly simple:

$$\psi^{-1} \circ \tau : E_{ns}(\mathbb{F}_q) \longrightarrow G_\alpha, \qquad \infty \longmapsto [\infty], \qquad (x, y) \longmapsto [y/x].$$

It is possible to formulate our algorithms given in the later sections in terms of the language of elliptic curves.

2.4 The Square Root Algorithms

Suppose β is a square in $\mathbb{F}_q^\times$ for some odd q. We have

$$\alpha^2 = \beta \qquad \text{for some } \alpha \in \mathbb{F}_q^\times.$$

Consider the abelian group G_α defined in the previous section. Let ζ_m be a primitive mth root of unity in $\overline{\mathbb{F}_q}$, a fixed algebraic closure of $\mathbb{F}_q$. If m divides $q-1$, then ζ_m is in $\mathbb{F}_q$. We have the following proposition.

Proposition 2.4.1. *Let $[0] \neq [a] \in G'_\alpha$. Suppose $[a]^d = [\infty]$ for some $d > 0$. Then,*

$$\alpha = \pm\frac{a(\zeta_d^k - 1)}{\zeta_d^k + 1} \qquad \textit{for some } 0 < k < \frac{d}{2}.$$

Proof. Since ψ is an isomorphism, $\psi([a])^d = 1$ in $\mathbb{F}_q^\times$. Therefore, $\psi([a]) = \zeta_d^j$ for some $0 < j < d$. We have $j \neq \frac{d}{2}$, otherwise, $\zeta_d^j = -1$. But $[a] \neq [0]$, which is the only order 2 element in G_α. Then,

$$[a] = \psi^{-1}(\zeta_d^j) = \left[\frac{\alpha(\zeta_d^j + 1)}{\zeta_d^j - 1}\right],$$

which implies $\alpha = \frac{a(\zeta_d^j - 1)}{\zeta_d^j + 1}$. If $j < \frac{d}{2}$, we prove the proposition by setting $k = j$. If $j > \frac{d}{2}$, let $k = d - j < \frac{d}{2}$. Finally,

$$\frac{a(\zeta_d^k - 1)}{\zeta_d^k + 1} = \frac{a(\zeta_d^{-j} - 1)}{\zeta_d^{-j} + 1} = \frac{a(1 - \zeta_d^j)}{1 + \zeta_d^j} = -\alpha,$$

which implies the proposition. □

Proposition 2.4.1 suggests a method to compute α. It requires (1) an element $[a] \in G_\alpha$ such that $[a]^d = [\infty]$, (2) a primitive dth root of unity $\zeta_d \in \mathbb{F}_q$ and (3) the index k in the proof. The power of an element $[a]^n$ has to be efficiently computable.

Lemma 2.4.2. *Given β a square in $\mathbb{F}_q$, the group operation and the power of an element in G_α can be computed in polynomial time without the knowledge of α.*

Proof. Clearly, the computation of the group operation involving the identity element or the power of the identity element is trivial.

For any $[g_1], [g_2] \in G'_\alpha$, by equations (2.3.2 and 2.3.3),

$$[g_1] * [g_2] = \begin{cases} [\infty] & \text{, if } g_1 = -g_2, \\ \left[\frac{g_1 g_2 + \beta}{g_1 + g_2}\right] & \text{, otherwise.} \end{cases} \tag{2.4.1}$$

Therefore, the group operation with any elements can be computed in polynomial time. For any $[g] \in G'_\alpha$, $[g]^2$ can be computed by equation (2.4.1). Then, $[g]^k$ can be evaluated efficiently by the successive squaring method.

Another method for computing $[g]^k$ is due to Proposition 2.3.5 and equation (2.3.6). If $\Psi_k(g) = 0$, then $[g]^k = [\infty]$. Otherwise, $[g]^k = \left[\frac{\gamma_k(g)}{\Psi_k(g)}\right]$. The polynomials $\gamma_k(g)$ and $\Psi_k(g)$ can be evaluated by the recursion equations in Lemma 2.3.3 and 2.3.4. Note that the coefficients in γ's, Ψ's and the recursion equations only involve integers and β, but not α. □

The running time for computing a group operation is $\tilde{O}(\log q)$ since multiplication and division in finite fields can be done in $\tilde{O}(\log q)$ (see Section 1.2.1). Then, the running time for computing $[g]^k$ for $k < q$ is $\tilde{O}(\log^2 q)$ for either of the methods described in the proof of the Lemma above.

In the following sections, we present deterministic polynomial time algorithms to find square roots for some families of finite fields. Let $\mathsf{element}(m)$ be a procedure returning the mth element of $\mathbb{F}_q$ in a fixed enumeration such that the procedure

element satisfies equation (1.2.3). Let $\mathsf{power}(g, k, \beta)$ denote a procedure computing $[g]^k$ in G_α.

2.4.1 Case $q = 2^e 3^f t + 1$

Let $\mathbb{F}_q$ be the finite field with q elements and characteristic p such that $q = 2^e 3^f t + 1$ and $p \equiv 1 \pmod{12}$. Note that $e \geq 2$ and $f \geq 1$ since $p \equiv 1 \pmod{12}$. Then, -1 and -3 are squares in the prime field $\mathbb{F}_p$. In this case, $\sqrt{-1}$ and $\sqrt{-3}$ in $\mathbb{F}_p$ can be computed by Schoof's algorithm. We have the Algorithm 2.4.3 for computing square roots in $\mathbb{F}_q$.

Algorithm 2.4.3. squareRoot(β)

```
{
    for m = 1 to t
    {
        Set g = element(m)

        if g^2 = β
            return ±g

        else if power(g, (q-1)/2^(e-1), β) ≠ [∞]
            return matchZeta4(g, β)

        else if power(g, (q-1)/3^f, β) ≠ [∞]
            return matchZeta3(g, β)
    }
}

matchZeta4(g, β)
{
    find the largest k such that power(g, (q-1)/2^k, β) = [∞]
    compute [a] = power(g, (q-1)/2^(k+2), β)
    return ±a√(-1)
}

matchZeta3(g, β)
{
```

find the largest k such that $\mathsf{power}(g, \frac{q-1}{3^k}, \beta) = [\infty]$
compute $[a] = \mathsf{power}(g, \frac{q-1}{3^{k+1}}, \beta)$
return $\pm a\sqrt{-3}$
}

Lemma 2.4.4. *Algorithm 2.4.3 always returns the square roots of β.*

Proof. Inside the for-loop, if $g^2 = \beta$, clearly the Lemma is true.

Let $\alpha^2 = \beta$. Suppose $g \neq \pm\alpha$.

If $[g]^{\frac{q-1}{2^{e-1}}} \neq [\infty]$, there exists $0 \leq k < e-1$ such that $[g]^{\frac{q-1}{2^k}} = [\infty]$ and $[g]^{\frac{q-1}{2^{k+1}}} \neq [\infty]$. Let $[a] = [g]^{\frac{q-1}{2^{k+2}}}$. Then, $[a]^4 = 1$. By proposition 2.4.1, $\alpha = \pm\frac{a(\zeta_4-1)}{\zeta_4+1} = \pm a\sqrt{-1}$. Similarly, if $[g]^{\frac{q-1}{3^f}} \neq [\infty]$, there exists $0 \leq k < f$ such that $[g]^{\frac{q-1}{3^k}} = [\infty]$ and $[g]^{\frac{q-1}{3^{k+1}}} \neq [\infty]$. Let $[a] = [g]^{\frac{q-1}{3^{k+1}}}$. Then, $[a]^3 = 1$. By proposition 2.4.1, $\alpha = \pm\frac{a(\zeta_3-1)}{\zeta_3+1} = \pm a\sqrt{-3}$, where $\zeta_3 = \frac{-1\pm\sqrt{-3}}{2}$.

We show that the algorithm always returns an answer. If $[g]^{\frac{q-1}{2^{e-1}}} = [g]^{\frac{q-1}{3^f}} = [\infty]$, the order of $[g]$ divides $2t$. There is a unique subgroup H of G_α with $2t$ elements since G_α is cyclic. Then, $[g] \in H$. Since $[-g] = [g]^{-1}$, we have $[-g] \in H$. We also have $[\infty], [0] \in H$. Let $g_m = \mathsf{element}(m)$ for $1 \leq m \leq t$. There are $2t+2$ elements in the set $\{[\infty], [0], [\pm g_1], [\pm g_2], \cdots, [\pm g_t]\}$ by the property of the $\mathsf{element}()$ procedure (see equation (1.2.3)). Therefore, there exists some $1 \leq m_0 \leq t$ such that $g_{m_0} \notin H$. Then, g_{m_0} leads to the algorithm returning an answer. □

For running time, $\mathsf{element}(m)$, g^2 and $\mathsf{power}(g, j, \beta)$ for $j < q$ can be computed in $\tilde{O}(\log q)$, $\tilde{O}(\log q)$ and $\tilde{O}(\log^2 q)$, respectively. Once a condition in one of the three if-statements is satisfied, the algorithm must return without further looping. So it needs $\tilde{O}(t \log^2 q)$ for finishing the loop.

If $g^2 = \beta$, no further operations is required. If $\mathsf{power}(g, \frac{q-1}{2^{e-1}}, \beta) \neq [\infty]$, finding the required k needs $\tilde{O}(\log^2 q)$, computing the power is $\tilde{O}(\log^2 q)$ and computing the square roots of -1 in $\mathbb{F}_p$ by Schoof's algorithm is $O(\log^9 p)$. The overall running time is $\tilde{O}(\log^2 q + \log^9 p)$. It is similar for the case $\mathsf{power}(g, \frac{q-1}{3^f}, \beta) \neq [\infty]$. The running time is also $\tilde{O}(\log^2 q + \log^9 p)$.

Therefore, the running time of the Algorithm 2.4.3 is $\tilde{O}(t \log^2 q + \log^9 p)$.

Proof of Theorem 2.1.1. Since $t = O(\mathrm{poly}(\log q))$, square roots in $\mathbb{F}_q$ can be computed by Algorithm 2.4.3 with running time $\tilde{O}(\mathrm{poly}(\log q) \log^2 q + \log^9 p)$. For finding a quadratic nonresidue, we first take square root of $\sqrt{-1}$ and obtain $(-1)^{1/4}$. Then, keep taking square root of $(-1)^{1/4}$, $(-1)^{1/8}$, $\cdots$, $(-1)^{1/2^{e-1}}$. At last, we obtain $(-1)^{1/2^e}$ which is a quadratic nonresidue in $\mathbb{F}_q$. Clearly, such algorithm is polynomial time. □

2.4.2 Other Cases

Algorithm 2.4.3 in the previous section can be generalized as below.

Lemma 2.4.5. *Let $p_1, p_2, \cdots, p_m$ be m distinct odd primes such that $p_j < M$ for some upper bound M. Let $q = p^n = 2^{e_0} p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m} t + 1$ with $e_0 \geq 2$, $e_j \geq 1$ and $(2p_j, t) = 1$ for $j = 1, 2, \cdots, m$. Suppose $M + t = O(\mathrm{poly}(\log q))$ and $\zeta_4, \zeta_{p_1}, \zeta_{p_2}, \cdots, \zeta_{p_m} \in \mathbb{F}_q$ are polynomial time computable. Then, there is a deterministic polynomial time algorithm for taking square roots and finding quadratic nonresidues in $\mathbb{F}_q$.*

Sketch of proof. Since $M + t = O(\mathrm{poly}(\log q))$ and $\zeta_{p_1}, \zeta_{p_2}, \cdots, \zeta_{p_m} \in \mathbb{F}_q$ can be computed in polynomial time, a deterministic polynomial time algorithm similar to Algorithm 2.4.3 can be defined for taking square roots in $\mathbb{F}_q$. Note that for $p_j > 3$,

once an order d element $[g] \in G_\alpha$ with $p_j | d$ is found, we could compute an order p_j element in G_α and match it up with ζ_{p_j} as shown in Algorithm 2.4.6 below with $r = p_j$ (see also Proposition 2.4.1). Then, finding quadratic nonresidues can also be done in deterministic polynomial time

For the prime 2, we have $\zeta_2 = -1$. The relation $\psi([0]) = -1$ is independent of α. An order 4 element in G_α and a 4th root of unity $\zeta_4 \in \mathbb{F}_q^\times$ are required instead. Therefore, if $\sqrt{-1} \in \mathbb{F}_q$ can be computed efficiently, the 2-part of $\mathbb{F}_q^\times$ can be handled. □

Algorithm 2.4.6. matchZeta(r, ζ_r, g, β)

{

find the largest k such that power$(g, \frac{q-1}{r^k}, \beta) = [\infty]$

compute $[a] =$ power$(g, \frac{q-1}{r^{k+1}}, \beta)$

find $j \in \{1, 2, \cdots, (r-1)/2\}$ such that $\left(\frac{a(\zeta_r^j - 1)}{\zeta_r^j + 1}\right)^2 = \beta$

return $\pm \frac{a(\zeta_r^j - 1)}{\zeta_r^j + 1}$

}

For example, let p be a prime and r be a Fermat prime (i.e. $r = 2^{2^k} + 1$ for some $k \geq 0$). The rth root of unity ζ_r can be written in terms of square roots (e.g. equation (2.0.1)) by Gaussian period theory. Suppose taking square roots in $\mathbb{F}_p$ can be done in polynomial time (e.g. $p \equiv 3, 5, 6 \pmod 7$ or $p \not\equiv 1 \pmod{240}$ or $p = 2^e 3^f s + 1$ for some small s). Then, ζ_r can be computed in polynomial time. For any $q = r_1^{e_1} \cdots r_m^{e_m} t + 1$ with (1) q a power of p, (2) $r_1, \cdots, r_m$ Fermat primes and (3) $t = O(\text{poly}(\log q))$, taking square roots in $\mathbb{F}_q$ can be done in polynomial time.

In particular, let $r = 5$ and $p \equiv 1 \pmod{20}$. Suppose taking square roots in $\mathbb{F}_p$ can be done in polynomial time. Then, $a^2 - 4$ in equation (2.0.1) is a square in $\mathbb{F}_p$ and

$\zeta_5 \in \mathbb{F}_p$ can be computed in polynomial time. In this case, for any $q = p^n = 2^{e_2}5^{e_5}t+1$ with $t = \text{poly}(\log q)$, taking square roots in $\mathbb{F}_q$ can be done in polynomial time.

2.4.3 Computing $\zeta_{2\cdot 3^k+1}$

Suppose p be a prime with $p \equiv 1 \pmod 4$ and $p \equiv 4, 7 \pmod 9$. We show in Lemma 2.4.7 below that cubic roots in $\mathbb{F}_p$ can be computed efficiently. Let $r = 2 \cdot 3^k + 1$ be a prime for some $k \geq 1$. The appendix in [61] shows a method to compute ζ_r in deterministic polynomial time. Therefore, the r-part of $\mathbb{F}_q^\times$ can be handled.

Lemma 2.4.7. *Let p be a prime with $p \equiv 1 \pmod 4$ and $p \equiv 4, 7 \pmod 9$. If b is a cubic residue in $\mathbb{F}_p$, cube roots of b can be computed in polynomial time.*

Proof. Let $\zeta_3 = \frac{-1\pm\sqrt{-3}}{2} \in \mathbb{F}_p$, which can be computed by Schoof's algorithm in polynomial time. Since b is a cubic residue in $\mathbb{F}_p$, we have $b^{(p-1)/3} = 1$. If $p \equiv 4 \pmod 9$, let $a = b^{(2p+1)/9}$. Then, $a^3 = b^{(2p+1)/3} = b^{1+2(p-1)/3} = b$. Therefore, $b^{(2p+1)/9}$, $b^{(2p+1)/9}\zeta_3$ and $b^{(2p+1)/9}\zeta_3^2$ are cube roots of b. Similarly, if $p \equiv 7 \pmod 9$, let $a = b^{(p+2)/9}$. Then, $a^3 = b^{(p+2)/3} = b^{1+(p-1)/3} = b$. Therefore, $b^{(p+2)/9}$, $b^{(p+2)/9}\zeta_3$ and $b^{(p+2)/9}\zeta_3^2$ are cube roots of b. Clearly, every step can be computed in polynomial time and so the cube roots of b can be. □

Proof of Theorem 2.1.2. Since $p \equiv 13, 25 \pmod{36}$, cubic roots in $\mathbb{F}_p$ can be computed in polynomial time by Lemma 2.4.7. Compute $\sqrt{-1}, \sqrt{3}, \sqrt{r_j}$ by Schoof's algorithm. Then compute $\zeta_3 = \frac{-1\pm\sqrt{-3}}{2}$ and $\zeta_4 = \pm\sqrt{-1}$. With $p \equiv 1 \pmod{r_j}$ and $r_j = O(\text{poly}(\log q))$, ζ_{r_j} can be computed in polynomial time (see the appendix in [61]) for $j = 1, 2, \cdots, m$. Finally, Lemma 2.4.5 implies the theorem. □

2.4.4 Finding ζ_r by Searching

In the previous discussions, we first reduce the square root problem for arbitrary β to the problem of finding primitive roots of unity, which is further reduced to the square root problem for constant size β. Then, Schoof's algorithm can be used to compute the square roots of constant size β. In this section, we show another method to compute primitive roots of unity without the need of taking square roots.

Let p, r be primes and $q = p^n = r^e t + 1$ for some $n, e, t \geq 1$. Let H be the subgroup of $\mathbb{F}_q^\times$ with t elements. Let $g_m = \mathsf{element}(m)$ be the mth element in $\mathbb{F}_q^\times$ (see equation (1.2.2)). Consider the set $\{g_1, g_2, \cdots, g_{t+1}\}$ with $t+1$ elements. There exists an element g_{m_0} not in H for $1 \leq m_0 \leq t+1$. If t is small (i.e. r^e is large), such m_0 can be found. Let d be the order of g_{m_0}. We have $r|d$ and $\zeta_r = g_{m_0}^{d/r}$ is an rth root of unity in $\mathbb{F}_q$. We have Algorithm 2.4.8 below for finding ζ_r with running time $\tilde{O}(t \log^2 q)$, which is faster than our previous methods for computing a root of unity.

Algorithm 2.4.8. $\mathsf{findZeta}(r)$

{
    for $m = 1$ to $t+1$
        if $g_m^{(q-1)/r^e} \neq 1$, where $g_m = \mathsf{element}(m)$
            find the largest k such that $g_m^{(q-1)/r^k} = 1$ and then return $g_m^{(q-1)/r^{k+1}}$
}

An algorithm similar to Algorithm 2.4.3 can be constructed for computing the square roots. We have a for-loop, which is similar to the for-loop in Algorithm 2.4.3, in the algorithm. The running time of the for-loop is $\tilde{O}(t \log^2 q)$. Algorithm 2.4.6 (or $\mathsf{matchZeta4}()$ in Algorithm 2.4.3 if $r = 2$) is used for matching the elements. The running time is $\tilde{O}((\log q + r) \log q)$. Compute ζ_r (or ζ_4 if $r = 2$) by Algorithm

2.4.8. The total running time is $\tilde{O}((t \log q + r) \log q)$. If t is a small constant and $r = O(\log q)$, the running time becomes $\tilde{O}(\log^2 q)$. For example, if $p = 3^e \cdot 80 + 1$ is a prime, the running time of computing square roots in $\mathbb{F}_p$ is $\tilde{O}(\log^2 p)$. In particular, p is a prime for $e = 569$.

Proof of Theorem 2.1.3. Since $t = O(\mathrm{poly}(\log q))$, ζ_r (or ζ_4 if $r = 2$) can be computed in polynomial time by Algorithm 2.4.8. Together with $r = O(\mathrm{poly}(\log q))$, Lemma 2.4.5 implies the theorem. □

If n is large, we have a better strategy for computing ζ_r. Let $\mathbb{F}_q = \mathbb{F}_p[x]/f(x)$ for some monic irreducible polynomial $f(x) \in \mathbb{F}_p[x]$ with degree n. For $0 \leq k < p$, let

$$S_k \;=\; \left\{ \pm \prod_{m=0}^{k} (x+m)^{e_m} \in \mathbb{F}_q^\times \;:\; e_m \geq 0 \text{ and } \sum_{m=0}^{k} e_m < n \right\}.$$

be subsets of $\mathbb{F}_q^\times$. All the elements in S_k are distinct. The size of S_k is

$$|S_k| = 2 \sum_{j=0}^{n-1} \binom{j+k-1}{j} = 2 \binom{n+k-1}{k}.$$

Let $H \subset \mathbb{F}_q^\times$ be the subgroup of $\mathbb{F}_q^\times$ with t elements. If $|S_k| > |H| = t$, there exists $x + m_0 \notin H$ for some $0 \leq m_0 \leq k$. Find the largest d such that $(x+m_0)^{(q-1)/r^d} = 1$. Then $\zeta_r = (x+m_0)^{(q-1)/r^{d+1}}$ is an rth root of unity in $\mathbb{F}_q$.

For example, suppose $n = \lfloor \log^2 p \rfloor + 1 < p - 1$ and $t < \frac{p^{2 \log p}}{4\sqrt{\log p}}$. Set $k = \lfloor \log^2 p \rfloor$. By Lemma 2.4.9 below,

$$|S_k| = 2 \binom{2\lfloor \log^2 p \rfloor}{\lfloor \log^2 p \rfloor} > \frac{2^{2\lfloor \log^2 p \rfloor}}{\sqrt{\lfloor \log^2 p \rfloor}} > \frac{p^{2 \log p}}{4\sqrt{\log p}} > t.$$

There exists $0 \leq m_0 \leq \lfloor \log^2 p \rfloor$ such that the order of $x + m_0$ equals rs for some $s > 0$. Then, $\zeta_r = (x+m_0)^s \in \mathbb{F}_q$ can be computed in polynomial time.

Lemma 2.4.9. *We have the following lower bound for the central binomial coefficient*

$$\binom{2N}{N} > \frac{2^{2N-1}}{\sqrt{N}} \qquad \text{for } N > 1.$$

Proof. We show it by induction. For $N = 2$, we have $\binom{4}{2} = 6 > \frac{8}{\sqrt{2}}$. For $k > 2$, assume $\binom{2(k-1)}{k-1} > \frac{2^{2k-3}}{\sqrt{k-1}}$. Then,

$$\binom{2k}{k} = \frac{2(2k-1)}{k}\binom{2k-2}{k-1} > \frac{(2k-1)2^{2k-2}}{k\sqrt{k-1}} > \frac{2^{2k-1}}{\sqrt{k}},$$

since $\frac{2k-1}{2\sqrt{k(k-1)}} > 1$ for $k > 2$. □

2.5 Even Polynomials

Let β be a non-zero square over some finite field F. The problem of computing the square roots of β is obviously equivalent to the problem of factoring $x^2 - \beta$. It is possible to modify our square root algorithms (e.g. Algorithm 2.1.1) to find a non-trivial factor of $x^2 - \beta$. The idea is to do the computation over the ring $F[x]/(x^2 - \beta)$, instead of G_α. In Chapter 3, we will use this idea to generalize our square root algorithm to take rth roots. For taking rth roots, we will work in the ring $F[x]/(x^r - \theta)$, where θ is an rth power over F.

Let $f(x) \in F[x]$ be an even polynomial (i.e. $f(x) = f(-x)$) such that f is a product distinct[2] of linear polynomials. Then, $f(x) = \prod_i (x^2 - \beta_i)$ for some distinct squares $\beta_i \in F$. In this case, we can modify our square root algorithm to work on the ring $F[x]/(f(x))$ for finding a non-trivial factor of f. The modification is similar

[2]If f has repeated factors, it is easy to find a non-trivial factor of f. See Chapter 4, [71] and [29] for details.

to the rth root algorithm shown in Chapter 3. Since we already have an algorithm for solving arbitrary polynomial equations in Chapter 4, we skip the details of how to do the modification in this special case.

Chapter 3

Taking rth Roots

In this chapter, we extend the square root algorithms discussed in the previous chapter and show deterministic polynomial time algorithms for taking rth roots in some finite fields. Like the relationship between taking square roots and constructing quadratic nonresidues, the problem of constructing rth nonresidues, r a small positive integer and r not the characteristic of the field, is polynomial time reducible to the problem of taking rth roots, and vice versa. Clearly, if we can take rth roots, we can first pick a non-zero, non-identity element in a finite field, then keep taking rth roots and finally obtain an rth nonresidue. For the converse, Shanks' square root algorithm [56] can be generalized to take rth root with a given rth nonresidue.

In [12], Barreto and Voloch showed deterministic polynomial time algorithms for taking rth root in the finite field $\mathbb{F}_q$ when $(r, q-1) = 1$ or $r||q-1$ (i.e. $r|q-1$ and $((q-1)/r, r) = 1$). Buchmann and Shoup [18] provided a deterministic algorithm for constructing kth power nonresidues over finite fields. Their algorithm is polynomial time under the assumption of ERH (see Section 1.2.4). For other related results, see

the introduction in the previous chapter.

3.1 Main Results

We give a definition of a family of finite fields below.

Definition 3.1.1. *Let $\mathcal{F}_t$ be a family of finite fields such that for all $F \in \mathcal{F}_t$, F has q elements with*

(i) $q = r_1^{e_1} \cdots r_m^{e_m} t + 1$,

(ii) $r_1, \cdots, r_m$ *are distinct primes and* $(t, r_1 \cdots r_m) = 1$,

(iii) $e_j \geq 1$ *for* $1 \leq j \leq m$, *and*

(iv) $r_1 + \cdots + r_m + t = O(\mathrm{poly}(\log q))$.

(v) a primitive r_jth root of unity ζ_{r_j} can be computed efficiently for $1 \leq j \leq m$.

Informally, $\mathcal{F}_t$ is a set of finite fields in which a primitive ℓth of unity can be computed for all prime factors ℓ of $q-1$ except for $\ell | t$. For deterministic polynomial time algorithms constructing primitive rth roots of unity over finite fields, see the previous chapter or [61].

Denote the union of all $\mathcal{F}_t$ for $t \geq 1$ by

$$\overline{\mathcal{F}} \stackrel{\mathsf{def}}{=} \bigcup_{t \geq 1} \mathcal{F}_t. \tag{3.1.1}$$

Note that all prime factors of $q-1$ are small for $\mathbb{F}_q \in \overline{\mathcal{F}}$. Therefore, the factorization of $q-1$ can be computed efficiently. The main results are summarized in the theorems below.

Theorem 3.1.2. *Let $\mathbb{F}_q \in \overline{\mathcal{F}}$. For $r \in \{r_1, \cdots, r_m\}$, there is a deterministic polynomial time algorithm computing an rth root of any rth residue in $\mathbb{F}_q$. Equivalently, there is a deterministic polynomial time algorithm constructing an rth nonresidue in $\mathbb{F}_q$.*

Theorem 3.1.3. *Let $\mathbb{F}_q \in \mathcal{F}_1$. There is a deterministic polynomial time algorithm constructing a primitive element in $\mathbb{F}_q$.*

Proof. For any $\mathbb{F}_q \in \mathcal{F}_1$, an r_ith nonresidue $\zeta_{r_i^{e_i}} \in \mathbb{F}_q$ can be computed in deterministic polynomial for each i by Theorem 3.1.2. Then the product $\prod_{i=1}^{m} \zeta_{r_i^{e_i}}$ is a primitive element in $\mathbb{F}_q$. □

3.2 The rth Roots Problem

Let $\mathbb{F}_q \in \mathcal{F}_t$ (see Definition 3.1.1) be a finite field with q elements. Suppose

$$\beta = \alpha^r \qquad \text{for some } \alpha \in \mathbb{F}_q \text{ and some integer } r > 1. \tag{3.2.1}$$

The problem of taking rth roots over $\mathbb{F}_q$ is to find α, given a finite field $\mathbb{F}_q$, an element β and an integer r. If $q-1$ is not divisible by r^2, we can compute α easily. Therefore, assume

$$r \in \{r_1, \cdots, r_m\} \qquad \text{and} \qquad r^e || q-1 \text{ for some } e \geq 2$$

for the rest of the section. We show a deterministic polynomial time algorithm (Algorithm 3.6.1) for finding a non-trivial factor of $x^r - \beta$. Then α can be computed by Lemma 3.2.1 below. The input parameters are r, e, β and $\mathbb{F}_q$ (includes t, $r_1, \cdots, r_m$,

$e_1, \cdots, e_m$ defined in Definition 3.1.1), which are globally available. Unlike other algorithms for taking rth roots, Algorithm 3.6.1 does not require an rth nonresidue as an input and the associated proofs do not require any unproven assumption, like the Riemann Hypothesis. For the rest of this section, let

$$\rho = \zeta_r, \qquad \text{a fixed primitive } r \text{ root of unity in } \mathbb{F}_q. \tag{3.2.2}$$

Lemma 3.2.1. *Given a non-trivial factor of $x^r - \beta$, we can compute an rth root of β efficiently.*

Proof. Suppose $x^d + a_{d-1}x^{d-1} + \cdots + a_0 \in \mathbb{F}_q[x]$ is a non-trivial factor of $x^r - \beta$. Since $x^r - \beta = \prod_{j=0}^{r-1}(x - \rho^j\alpha)$, we have $a_0 = (-1)^d\rho^k\alpha^d$ for some integer k. We also have $(d, r) = 1$ because $d < r$ and r is prime. Find integers u, v by the Euclidean algorithm such that $ud + vr = 1$. Finally, $(-1)^{du}a_0^u\beta^v = \rho^{ku}\alpha$ is an rth root of β. □

The computations of the square root algorithms in the previous chapter are performed over the group G_α. It is possible to formulate the square root algorithms as algorithms for factoring the polynomial $x^2 - \beta$ over the ring $\mathbb{F}_q[x]/(x^2 - \beta)$. We generalize this idea and work on the ring $\mathbb{F}_q[x]/(x^r - \beta)$ for factoring the polynomial $x^r - \beta$. The "problem" of working on the ring $\mathbb{F}_q[x]/(x^r - \beta)$ is that there are non-zero, non-unit elements in $\mathbb{F}_q[x]/(x^r - \beta)$. However, if we can find a non-zero, non-unit element f, then $(f(x), x^r - \beta)$ is a non-trivial factor of $x^r - \beta$. This idea is similar to Lenstra's elliptic curve integer factoring algorithm [33]. He works on the ring $\mathbb{Z}/n\mathbb{Z}$ for some composite integer n and try to find a non-zero, non-unit element e in $\mathbb{Z}/n\mathbb{Z}$. Then, (e, n) is a non-trivial factor of n.

In our algorithms, we need to determine whether $f(x)$ is equal to zero in the ring $\mathbb{F}_q[x]/(x^r - \beta)$ for some polynomial $f(x) \in \mathbb{F}_q[x]$. Compute $h(x) = (f(x), x^r - \beta)$. If $h(x)$ is a non-trivial factor of $x^r - \beta$, we are done. Otherwise, $f(x)$ is either divisible by $x^r - \beta$ or relatively prime to $x^r - \beta$. We have the following algorithm.

Algorithm 3.2.2. isZero(f) `/* Comment:` *is* $f(x) \equiv 0 \pmod{x^r - \beta}$*?* `*/`

```
{
    compute h(x) = (f(x), x^r − β);
    if deg h = 0
        return FALSE;
    else if deg h = r
        return TRUE;
    else
        output h and halt;      /* Comment: found a non-trivial factor of x^r − β. */
}
```

3.3 Step 1: Find a Suitable Element a

Define a rational function $\psi_a(x)$ over $\mathbb{F}_q$ as

$$\psi_a(x) = \frac{a - x}{a - \rho x} \qquad \text{for some } a \in \mathbb{F}_q \text{ such that } a^r \neq \beta.$$

Then,

$$\psi_a(x) \equiv c_i \pmod{x - \rho^i \alpha} \qquad \text{for some } c_i \in \mathbb{F}_q^\times \text{ for each } i = 0, \cdots, r-1.$$

Consider $\psi_a(x)^{rt}$. We have three cases below:

(1) If the multiplicative order of c_i divides rt for all $0 \leq i \leq r-1$, we have $\psi_a(x)^{rt} \equiv 1 \pmod{x^r - \beta}$. This case is not useful to us. We will show that the number of possible values of a's falling into this case is small.

(2) The multiplicative order of at least one c_i's divides rt and the multiplicative order of at least one c_i's does not divide rt. Since $a - \rho x \in (\mathbb{F}_q[x]/(x^r - \beta))^\times$, let $h(x) = (\psi_a(x)^{rt} - 1) \bmod (x^r - \beta)$ be a polynomial. Then, $(h(x), x^r - \beta)$ is a non-trivial factor of $x^r - \beta$ and we are done.

(3) If the multiplicative orders of all c_i's do not divide rt, we have $\psi_a(x)^{rt} - 1 \in (\mathbb{F}_q[x]/(x^r - \beta))^\times$. We want to find such a in this step if we cannot discover a non-trivial factor of $x^r - \beta$.

Instead of working with the rational function ψ_a, we define a polynomial

$$g_k(x, y, z) = (y - x)^k - z(y - \rho x)^k \in \mathbb{F}_q[x, y, z] \qquad \text{for } k > 0. \tag{3.3.1}$$

It is easy to see that in case (1), we have $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{TRUE}$; in case (2), $\mathsf{isZero}(g_{rt}(x, a, 1))$ outputs a non-trivial factor of $x^r - \beta$; and in case (3), we have $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{FALSE}$. In step 1, we either find a non-trivial factor of $x^r - \beta$ or a value of a such that $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{FALSE}$. In general we have the following lemma.

Lemma 3.3.1. *Let $d_i = \operatorname{ord} c_i$, the order of c_i in $\mathbb{F}_q^\times$. If d_i divides k for all $0 \le i < r$, we have $\mathsf{isZero}(g_k(x, a, 1)) = \mathsf{TRUE}$. If d_i does not divide k for all $0 \le i < r$, we have $\mathsf{isZero}(g_k(x, a, 1)) = \mathsf{FALSE}$. If there exists i_0, i_1 such that d_{i_0} divides k but d_{i_1} does not divide k, $\mathsf{isZero}(g_k(x, a, 1))$ outputs a non-trivial factor of $x^r - \beta$.*

Proof. It is obvious. □

We have the following algorithm finding a desired a.

Algorithm 3.3.2. findA()

```
{
    for i = 1 to rt + 1
    {
        set a_i = ith element in F_q;
        if a_i^r = β
            output x − a_i and halt;

        if isZero(g_rt(x, a_i, 1)) = FALSE
            return a_i;
    }
}
```

Lemma 3.3.3. *There are at most rt distinct values of $a \in \mathbb{F}_q$ such that $a^r \neq \beta$ and* $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{TRUE}$.

Proof. Suppose $a^r \neq \beta$. The case $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{TRUE}$ implies $\psi_a(\alpha)^{rt} = 1$. So the multiplicative order of $\psi_a(\alpha)$ divides rt. There are only rt elements in $\mathbb{F}_q^\times$ having multiplicative order dividing rt since $\mathbb{F}_q^\times$ is cyclic. We also have $\psi_a(\alpha) \neq \psi_b(\alpha)$ whenever $a \neq b$. Therefore there are at most rt distinct values of a's such that $\psi_a(\alpha)^{rt} = 1$. The lemma follows. □

3.4 Step 2: Find a Suitable ℓ

In this section, suppose $a \in \mathbb{F}_q$, is a fixed element obtained in the previous step, i.e. $a^r \neq \beta$ and $\mathsf{isZero}(g_{rt}(x, a, 1)) = \mathsf{FALSE}$. Let $d_i = \operatorname{ord} \psi_a(\rho^i \alpha)$, the multiplicative order in $\mathbb{F}_q^\times$. We have d_i not dividing rt for all $0 \leq i < r$ by Lemma 3.3.1. We have the following algorithm to find a suitable $\ell \in \{r_1, \cdots, r_m\}$.

Algorithm 3.4.1. findL(a)

```
{
    for j = 1 to m
```

if $r_j \neq r$ and $\mathsf{isZero}(g_{(q-1)/r_j^{e_j}}(x,a,1)) = \mathsf{FALSE}$
return r_j;

if $\mathsf{isZero}(g_{(q-1)/r^{e-1}}(x,a,1)) = \mathsf{FALSE}$
return r;
}

All the return statements in Algorithm 3.4.1 are conditional. It might seem that $\mathsf{findL}(a)$ may terminate without returning any value and giving any output. We show below that it is not the case.

Lemma 3.4.2. *Suppose every call of* isZero *in* $\mathsf{findL}(a)$ *returns* TRUE *or* FALSE *but does not output a non-trivial factor of* $x^r - \beta$. *If* $\mathsf{isZero}(g_{(q-1)/r_j^{e_j}}(x,a,1)) = \mathsf{TRUE}$ *for all* $r_j \neq r$, *then* $\mathsf{isZero}(g_{(q-1)/r^{e-1}}(x,a,1)) = \mathsf{FALSE}$.

Proof. Suppose $\mathsf{isZero}(g_{(q-1)/r_j^{e_j}}(x,a,1)) = \mathsf{TRUE}$ for all $r_j \neq r$. We have d_i dividing $(q-1)/r_j^{e_j}$ for all $0 \leq i < r$ and all $1 \leq j \leq m$ such that $r_j \neq r$. Then, d_i divides $r^e t = \gcd_{\substack{1 \leq j \leq m \\ r_j \neq r}}((q-1)/r_j^{e_j})$ for all i. Since d_i does not divide rt for all i, we have $\mathsf{isZero}(g_{(q-1)/r^{e-1}}(x,a,1)) = \mathsf{FALSE}$. □

3.5 Step 3: Compute a Non-trivial Factor

Let

$$\ell = \mathsf{findL}(a) = \begin{cases} r_j & \text{, if } r_j \neq r \text{ and } \mathsf{isZero}(g_{(q-1)/r_j^{e_j}}(x,a,1)) = \mathsf{FALSE}; \\ r & \text{, otherwise.} \end{cases}$$

Case 1: Suppose $\ell = r_{j_0} \neq r$ for some fixed j_0, is the value obtained from the previous step. We have $\mathsf{isZero}(g_{(q-1)/\ell^{e_{j_0}}}(x,a,1)) = \mathsf{FALSE}$. Since $\psi_a(\rho^i\alpha)^{q-1} = 1$ for

all $0 \le i < r$, we have $\mathsf{isZero}(g_{q-1}(x,a,1)) = \mathsf{TRUE}$. Suppose $\mathsf{isZero}(g_{(q-1)/\ell^k}(x,a,1))$ returns either TRUE or FALSE but does not output a non-trivial factor for $0 \le k \le e_{j_0}$. By the lemma below, there exists $0 < k_0 < e_{j_0}$ such that $\mathsf{isZero}(g_{(q-1)/\ell^k}(x,a,1)) = \mathsf{TRUE}$ for $k = 0, 1, \cdots, k_0$ and $\mathsf{isZero}(g_{(q-1)/\ell^k}(x,a,1)) = \mathsf{FALSE}$ for $k = k_0+1, \cdots, e_{j_0}$.

Lemma 3.5.1. *If* $\mathsf{isZero}(g_k(x,a,1)) = \mathsf{TRUE}$*, then* $\mathsf{isZero}(g_{nk}(x,a,1)) = \mathsf{TRUE}$ *for any positive integer* n*.*

Proof. If $\mathsf{isZero}(g_k(x,a,1)) = \mathsf{TRUE}$, we have $\psi_a(\rho^i\alpha)^k = 1$ for all $0 \le i < r$. Then $\psi_a(\rho^i\alpha)^{nk} = 1$ for all $0 \le i < r$. Finally, $\mathsf{isZero}(g_{nk}(x,a,1)) = \mathsf{TRUE}$ by Lemma 3.3.1. □

Let $d = (q-1)/\ell^{k_0+1}$ and $d_i = \operatorname{ord} \psi_a(\rho^i\alpha)$. We have d_i dividing ℓd and d_i not dividing d for all $0 \le i < r$. Since $\mathbb{F}_q^\times$ is cyclic, we have

$$\psi_a(\rho^i\alpha)^d = \zeta_\ell^{n_i} \qquad \text{for some integer } n_i \in (\mathbb{Z}/\ell\mathbb{Z})^\times \text{ for } i = 0, \cdots, r-1,$$

where ζ_ℓ is a primitive ℓth root of unity which can be computed efficiently by the assumption that $\mathbb{F}_q \in \mathcal{F}_1$ and the property (v) of $\mathcal{F}_1$ in Definition 3.1.1.

Lemma 3.5.2. *Let* N *be a prime power with* $1 < N \ne r$*. For some positive integer* D*, suppose*

$$\psi_a(\rho^i\alpha)^D = \zeta_N^{n_i} \qquad \textit{for some integer } n_i \in (\mathbb{Z}/N\mathbb{Z})^\times \textit{ for } i = 0, \cdots, r-1,$$

Then, there exist i_0 *and* i_1 *such that* $n_{i_0} \ne n_{i_1}$*.*

Proof. Suppose $n_0 = \cdots = n_{r-1} = n$ for some integer n with $(n, N) = 1$. Let $\zeta = \zeta_N^n$. For all $0 \le i < r$, we have $\psi_a(\rho^i\alpha)^D = \zeta$, which is equivalent to $g_D(\rho^i\alpha, a, \zeta) = 0$.

Then,

$$(a-\alpha)^D(1-\zeta^r) = \sum_{i=0}^{r-1} \zeta^i g_D(\rho^i\alpha, a, \zeta) = 0.$$

Thus, $\zeta^r = 1$ since $a \neq \alpha$. We have $N|rn$ which is a contradiction. □

By Lemma 3.5.2 (with $N = \ell$ and $D = d$), $x - \alpha$ divides $g_d(x, a, \zeta_\ell^{n_0})$ and there exists $0 < i < r$ such that $x - \rho^i\alpha$ does not divide $g_d(x, a, \zeta_\ell^{n_0})$. Therefore, $(g_d(x, a, \zeta_\ell^{n_0}), x^r - \beta)$ is a non-trivial factor of $x^r - \beta$. We try $(g_d(x, a, \zeta_\ell^n), x^r - \beta)$ to find a non-trivial factor for $n = 1, \cdots, \ell - 1$. See procedure factorByZeta (with $N = \ell$) in Algorithm 3.6.1.

Case 2: Suppose $\ell = r$. The situation is similar: we have $\mathsf{isZero}(g_{(q-1)/r^{e-1}}(x, a, 1)) =$ FALSE. Suppose $\mathsf{isZero}(g_{(q-1)/\ell^k}(x, a, 1))$ returns either TRUE or FALSE but does not output a non-trivial factor for $0 \leq k \leq e - 1$. By Lemma 3.5.1 and the fact that $\psi_a(\rho^i\alpha)^{q-1} = 1$ for all $0 \leq i < r$, there exists $0 < k_0 < e - 1$ such that $\mathsf{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) =$ TRUE for $k = 0, 1, \cdots, k_0$ and $\mathsf{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) =$ FALSE for $k = k_0 + 1, \cdots, e - 1$. Let $d = (q-1)/r^{k_0+2}$. For $i = 0, \cdots, r - 1$, the element $\psi_a(\rho^i\alpha)^d$ is a primitive r^2th root of unity. Then,

$$\psi_a(\rho^i\alpha)^d = \zeta_{r^2}^{n_i} \qquad \text{for some integer } n_i \text{ such that } (n_i, r) = 1.$$

By Lemma 3.5.2 (with $N = r^2$ and $D = d$), $x-\alpha$ divides $g_d(x, a, \zeta_{r^2}^{n_0})$ and there exists $0 < i < r$ such that $x-\rho^i\alpha$ does not divide $g_d(x, a, \zeta_{r^2}^{n_0})$. Therefore, $(g_d(x, a, \zeta_{r^2}^{n_0}), x^r - \beta)$ is a non-trivial factor of $x^r - \beta$. We try $(g_d(x, a, \zeta_{r^2}^n), x^r - \beta)$ to find a non-trivial factor for each $n \in (\mathbb{Z}/r^2\mathbb{Z})^\times$. See factorByZeta (with $N = r^2$) in Algorithm 3.6.1.

3.5.1 Computing a Primitive r^2th Root of Unity

In case 2, we need to find a primitive r^2th root of unity, ζ_{r^2}. We have ρ, a primitive rth root of unity, by assumption. Then ζ_{r^2} can be computed by finding a non-trivial factor of $x^r - \rho$. Compute $a = \mathsf{findA}()$ in step 1 and $\ell = \mathsf{findL}(a)$ in step 2. Suppose all evaluations of isZero in step 1 and 2 do not output. If $\ell \neq r$, we continue with case 1 in step 3 and a non-trivial factor of $x^r - \rho$ is obtained.

Suppose $\ell = r$. We are in case 2. We find d as before. Then $(g_d(x, a, \zeta_{r^2}^n), x^r - \rho)$ is a non-trivial factor of $x^r - \rho$ for some n. Nevertheless, we cannot compute the gcd directly because we do not have ζ_{r^2}. Suppose

$$\psi_a(\rho^i \zeta_{r^2})^d = \zeta_{r^2}^{n_i} \qquad \text{for some integer } n_i \in (\mathbb{Z}/r^2\mathbb{Z})^\times \text{ for } i = 0, \cdots, r-1.$$

Consider the polynomial $g_d(x, a, x^{n_0})$. We have $x - \zeta_{r^2}$ dividing $g_d(x, a, x^{n_0})$. We show in the lemma below that there exists $0 < i < r$ such that $x - \rho^i \zeta_{r^2}$ does not divide $g_d(x, a, x^{n_0})$. We try $(g_d(x, a, x^n), x^r - \rho)$ to find a non-trivial factor for each $n \in (\mathbb{Z}/r^2\mathbb{Z})^\times$. See $\mathsf{factorByX}$ in Algorithm 3.6.1.

Lemma 3.5.3. *Suppose $x - \zeta_{r^2}$ divides $g_d(x, a, x^n)$ for some $n \in (\mathbb{Z}/r\mathbb{Z})^\times$. There exists $0 < i < r$ such that $x - \rho^i \zeta_{r^2}$ does not divide $g_d(x, a, x^n)$.*

Proof. Let $\zeta = \zeta_{r^2}$. Suppose $x - \rho^i \zeta$ divides $g_d(x, a, x^n)$ for all $0 < i < r$. We have

$$g_d(\rho^i \zeta, a, (\rho^i \zeta)^n) = (a - \rho^i \zeta)^d - \rho^{in} \zeta^n (a - \rho^{i+1} \zeta)^d = 0 \qquad \text{for } i = 0, \cdots, r-1.$$

Let $s_k = \sum_{i=0}^{k-1} i = k(k-1)/2$. Then,

$$0 = \sum_{i=0}^{r-1} \rho^{s_i n} \zeta^{in} g_d(\rho^i \zeta, a, (\rho^i \zeta)^n) = (a - \zeta)^d (1 - \rho^{s_r n} \zeta^{rn}) = (a - \zeta)^d (1 - \zeta^{rn}),$$

which implies $\zeta^{rn} = 1$ since $a \neq \zeta$. We have $r^2 | rn$, a contradiction. □

3.6 The Algorithm

We present the entire algorithm below and prove Theorem 3.1.2 at the end of the section.

Algorithm 3.6.1. factor$(x^r - \beta)$

{
    set $a =$ findA();
    set $\ell =$ findL(a);

    $k_0 =$ the largest k such that isZero$(g_{(q-1)/\ell^k}(x, a, 1)) =$ TRUE;

    if $\ell \neq r$
        factorByZeta$(\ell, \frac{q-1}{\ell^{k_0+1}}, a)$; `/* Comment:` defined below `*/`
    else
        if $\beta = \zeta_r$
            factorByX$(\frac{q-1}{r^{k_0+2}}, a)$; `/* Comment:` defined below `*/`
        else
            factorByZeta$(r^2, \frac{q-1}{r^{k_0+2}}, a)$;
}

factorByZeta(N, d, a) `/* Comment:` $\beta \neq \zeta_r$ in this case `*/`
{
    find ζ_N, a primitive Nth root of unity;
    for each $n \in (\mathbb{Z}/N\mathbb{Z})^\times$
        isZero$(g_d(x, a, \zeta_N^n))$;
}

factorByX(d, a) `/* Comment:` $\beta = \zeta_r$ in this case `*/`
{
    for each $n \in (\mathbb{Z}/r^2\mathbb{Z})^\times$
        isZero$(g_d(x, a, x^n))$;
}

Proof of Theorem 3.1.2. From our discussion in this section, Algorithm 3.6.1 (together with Lemma 3.2.1) is a deterministic algorithm for computing an rth root of any rth residue in $\mathbb{F}_q$. We list out the running times below:

Procedures	Running time (bit operations)
Factoring $q-1$	$O(\text{poly}(\log q))$
findA	$\tilde{O}(r^2 t \log q)$
findL	$\tilde{O}(mr \log^2 q)$
Computing k_0	$\tilde{O}(r \log^2 q)$
factorByZeta	$\tilde{O}(Nr \log^2 q)$
factorByX	$\tilde{O}(r^3 \log^2 q)$
Computing an rth root from a non-trivial factor of $x^r - \beta$	$\tilde{O}(\log^2 q)$

Therefore, the overall running time is polynomial in the input size.

For constructing an rth nonresidue in $\mathbb{F}_q$, we keep taking rth roots beginning from ζ_r and obtain $\zeta_{r^2} = \sqrt[r]{\zeta_r}$, $\zeta_{r^3} = \sqrt[r]{\zeta_{r^2}}$, $\cdots$, $\zeta_{r^e} = \sqrt[r]{\zeta_{r^{e-1}}}$. Finally, ζ_{r^e} is an rth nonresidue in $\mathbb{F}_q$. □

3.7 Finding a Non-trivial Factor of $\Phi_{r^2}(x)$

Let $\mathbb{F}_q$ be a finite field with q elements. Let $\zeta_n \in \overline{\mathbb{F}_q}$ be a primitive nth root of unity, where $\overline{\mathbb{F}_q}$ denotes a fixed algebraic closure of $\mathbb{F}_q$. Denote the nth cyclotomic polynomial by

$$\Phi_n(x) \stackrel{\text{def}}{=} \prod_{i \in (\mathbb{Z}/n\mathbb{Z})^\times} (x - \zeta_n^i).$$

It is easy to see that $\Phi_n(x) \in \mathbb{F}_q[x]$ since $\Phi_n(x)$ is fixed by any automorphism which fixes $\mathbb{F}_q$. Let r be a prime such that $r^2 || q-1$. We have

$$\Phi_{r^2}(x) = \Phi_r(x^r) = \sum_{i=0}^{r-1} x^{ir}.$$

We consider the problem of finding a non-trivial factor of $\Phi_{r^2}(x)$ in this section.

We may be able to construct a primitive rth root of unity from a non-trivial factor of $\Phi_{r^2}(x)$. If we have a non-trivial factor of $\Phi_{r^2}(x)$ with constant term not equal to ± 1, a primitive rth root of unity can be constructed from the constant term. Unfortunately, it is possible to have a non-trivial factor of $\Phi_{r^2}(x)$ with constant term equal ± 1, although the number of such cases is small.

3.7.1 Method 1

Write $q = p_1^{e_1} \cdots p_m^{e_m} + 1$ for some distinct primes $p_1, \cdots, p_m$ and some positive integers $e_1, \cdots, e_m$, where $p_1 = r$ and $e_1 \geq 2$. Suppose $\zeta_{p_2}, \cdots, \zeta_{p_m}$ are available. Let

$$\zeta = \zeta_{r^2} \in \mathbb{F}_q, \qquad \rho = \zeta_r \in \mathbb{F}_q$$

We try to factor $\Phi_{r^2}(x)$ by Algorithm 3.6.1 with some modifications.

We use $\Phi_{r^2}(x)$ as an input, instead of $x^r - \beta$. In Algorithm 3.6.1, we work on $\psi_a(x)$. Here, we work on the rational polynomial

$$\frac{a - x}{a - x^{r+1}} \pmod{\Phi_{r^2}(x)}.$$

With the corresponding modifications, compute[1] a and ℓ. If $\ell \neq r$, we proceed with the Case 1 in Section 3.5 to obtain a non-trivial factor of $\Phi_{r^2}(a)$. The bad case is when $\ell = r$. We find d as the Case 2 in Section 3.5. Define a polynomial

$$h_j(x) \stackrel{\mathsf{def}}{=} (a - x)^d - x^j (a - x^{r+1})^d.$$

[1]We assume $\Phi_r(a) \neq 0$ and $\Phi_{r^2}(a) \neq 0$. If $\Phi_r(a) = 0$ or $\Phi_{r^2}(a) = 0$, a primitive rth root of unity can be constructed. We can compute the complete factorization of $\Phi_{r^2}(a)$ by the algorithm in Chapter 4.

Then, a non-trivial factor of $\Phi_{r^2}(x)$ can be discovered by the lemma below.

Lemma 3.7.1.

$$(h_j(x), \Phi_{r^2}(x)) \qquad \textit{for some } j \in (\mathbb{Z}/r^2\mathbb{Z})^\times$$

is a non-trivial factor of $\Phi_{r^2}(x)$.

Proof. By the construction of a, ℓ and d,

$$\left(\frac{a-x}{a-x^{r+1}}\right)^d = \zeta^{n_i} \pmod{x-\zeta^i} \qquad \text{for some } n_i \in (\mathbb{Z}/r^2\mathbb{Z})^\times \text{ for all } i \in (\mathbb{Z}/r^2\mathbb{Z})^\times.$$

If there exist i_0, i_1 such that $n_{i_0}/i_0 \neq n_{i_1}/i_1$, we are done.

Suppose $n_i/i = j \in (\mathbb{Z}/r^2\mathbb{Z})^\times$ for all $i \in (\mathbb{Z}/r^2\mathbb{Z})^\times$. Then,

$$\left(\frac{a-\zeta^i}{a-\zeta^{i(r+1)}}\right)^d = \zeta^{ij} \qquad \text{for all } i \in (\mathbb{Z}/r^2\mathbb{Z})^\times.$$

Equivalently,

$$h_j(\zeta^i) = 0 \qquad \text{for all } i \in (\mathbb{Z}/r^2\mathbb{Z})^\times.$$

Let $s_n = \sum_{k=0}^{n-1}(r+1)^k = \frac{(r+1)^n-1}{r}$ and

$$T_n \stackrel{\text{def}}{=} \sum_{k=0}^{n-1} \zeta^{js_k} h_j(\zeta^{(r+1)^k}) = 0 \qquad \text{for all } n > 0.$$

We also have $0 = T_n = (a-\zeta)^d - \zeta^{js_n}(a-\zeta^{(r+1)^n})^d$ for all $n > 0$. By the fact that $(r+1)^r \equiv 1 \pmod{r^2}$, we have

$$0 = T_r = (a-\zeta)^d - \zeta^{js_r}(a-\zeta^{(r+1)^r})^d = (a-\zeta)^d(1-\zeta^{j((r+1)^r-1)/r}).$$

Finally, $\zeta^{j((r+1)^r-1)/r} = 1$ since $a \neq \zeta$. We have r^2 dividing $j((r+1)^r-1)/r$, which is a contradiction. □

3.7.2 Method 2

We show a second method for finding a non-trivial factor of $\Phi_{r^2}(x)$ in this section. Suppose we have the same situation as in the previous section. We consider the rational polynomial

$$\frac{a - x^r}{a - x^{(r-1)r}} \pmod{\Phi_{r^2}(x)}$$

and define a polynomial

$$h'_j(x) \stackrel{\mathsf{def}}{=} (a - x^r)^d - x^j(a - x^{(r-1)r})^d.$$

Replace $h_j(x)$ by $h'_j(x)$ and find a, ℓ and d as before. Suppose $\ell = r$ (otherwise, it falls in Case 1 which is an easy case). Then, a non-trivial factor of $\Phi_{r^2}(x)$ can be discovered by the lemma below.

Lemma 3.7.2.

$$(h'_j(x), \Phi_{r^2}(x)) \qquad \textit{for some } j \in (\mathbb{Z}/r^2\mathbb{Z})^\times$$

is a non-trivial factor of $\Phi_{r^2}(x)$.

Proof. We use a similar technique as in the proof of Lemma 3.7.1. Suppose

$$\left(\frac{a - \zeta^{ir}}{a - \zeta^{i(r-1)r}}\right)^d = \zeta^{ij} \qquad \text{for some } j \in (\mathbb{Z}/r^2\mathbb{Z})^\times, \text{ for all } i \in (\mathbb{Z}/r^2\mathbb{Z})^\times.$$

Then,

$$h'_j(\zeta^i) = 0 \qquad \text{for all } i \in (\mathbb{Z}/r^2\mathbb{Z})^\times.$$

However,

$$\begin{aligned} 0 &= h'_j(\zeta) + \zeta^j h'_j(\zeta^{r-1}) \\ &= (a - \zeta^r)^d - \zeta^{rj}(a - \zeta^{(r-1)^2 r})^d \\ &= (a - \zeta^r)^d(1 - \zeta^{rj}), \end{aligned}$$

which implies $r^2|rj$, a contradiction. $\square$

3.7.3 More Variations

Suppose $g_1(x), \cdots, g_m(x)$ are the non-trivial factors obtained in Method 1. It guarantees that $f_1(x) \stackrel{\mathsf{def}}{=} (x-\zeta^{r+1})(x-\zeta^{(r+1)^2})\cdots(x-\zeta^{(r+1)^r})$ does not divide $g_i(x)$ for all $i = 1, \cdots, m$. Similarly, Method 2 guarantees the non-trivial factors obtained are not divisible by $f_2(x) \stackrel{\mathsf{def}}{=} (x-\zeta^r)(x-\zeta^{(r-1)r})$.

We can have more variations by beginning with the rational polynomials

$$\tau_k(x) \stackrel{\mathsf{def}}{=} \frac{a-x}{a-x^{kr+1}} \qquad \text{and} \qquad \sigma_k(x) \stackrel{\mathsf{def}}{=} \frac{a-x^r}{a-x^{(kr-1)r}} \qquad \text{for } 0 < k < r.$$

Then, the non-trivial factors computed from $\tau_k(x)$ are not divisible by

$$(x-\zeta^{kr+1})(x-\zeta^{(kr+1)^2})\cdots(x-\zeta^{(kr+1)^r}) = f_1(x)$$

and the non-trivial factors computed from $\sigma_k(x)$ are not divisible by

$$(x-\zeta^r)(x-\zeta^{(kr-1)r}) = f_2(x).$$

Although the polynomials (i.e. f_1, f_2) are the same, the computation processes are different, therefore, the non-trivial factors computed may be different. If any of the non-trivial factors obtained has a constant term not equal to ± 1, then a primitive rth root of unity can be constructed.

Chapter 4

Solving Polynomial Equations

The problem of solving polynomial equations over finite fields is a generalization of the following problems over finite fields

- constructing primitive nth roots of unity,
- taking nth roots,
- constructing nth nonresidues,
- constructing primitive elements (generators of the multiplicative group)

for any positive n dividing the number of elements of the underlying field. By the Tonelli-Shanks square root algorithm [63, 56] and its generalization for taking nth roots, constructing nth nonresidues and taking nth roots are polynomial time equivalent for all n. It is clear that primitive nth roots of unity can be computed efficiently from any nth nonresidue when n is prime. It is obvious that a primitive element is also an nth nonresidue. In [61], we showed that, for some families of finite fields, once

we can compute a primitive nth root of unity for some suitably chosen n, we can take square roots.

The problem of solving polynomial equations is a special case of the problem of polynomial factoring. There is a deterministic polynomial time algorithm, Lenstra-Lenstra-Lovász [41], for factoring polynomials over rational numbers. For polynomial factoring over finite fields, we have Berlekamp's algorithm [13], which is deterministic but exponential time. So it only works well in small finite fields. There is a probabilistic version of Berlekamp's algorithm [14] for large finite fields. We also have Cantor-Zassenhaus [19], which is a probabilistic algorithm, for polynomial factoring over finite fields. For a survey of polynomial factoring, see [29].

The problem of solving polynomial equations is to find the solutions of $f(x) = 0$ over $\mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements and $f(x) \in \mathbb{F}_q[x]$ is a polynomial with $\deg f = O(\text{poly}(\log q))$. We may assume f is a product of distinct linear factors since squarefree factorization[1] (see [71] and [35]) and distinct degree factorization (see [29]) can be done efficiently. If f has a multiple root, then (f, f') is a non-trivial factor of f, where f' is the first derivative of f. Since $x^q - x$ is a product of all linear polynomials in $\mathbb{F}_q[x]$, we can work on $(f(x), x^q - x)$ instead of $f(x)$.

In addition to the polynomial factoring algorithms discussed above, algorithms related to solving univariate polynomial equations in finite fields include the following: Tonelli-Shanks [63, 56], Adleman-Manders-Miller [2] and Cipolla-Lehmer [20, 40] are

[1]Suppose the input polynomial is a product of some irreducible factors with multiplicity ≥ 1. Squarefree factorization is the process finding the product of the same set of irreducible factors with multiplicity equal 1.

polynomial time square root algorithms, which require a quadratic nonresidue as an input. Shanks' algorithm can be generalized to take nth root with a given nth nonresidue. Schoof's algorithm [55], which takes square root for the elements in prime fields, is deterministic but the running time is polynomial only if the input element is small. In the previous chapters, we have shown deterministic polynomial time algorithms for constructing primitive rth roots of unity, taking square roots and taking rth roots over some families of finite fields.

In this chapter, we prove that there is a deterministic polynomial time algorithm solving polynomial equations over any finite field F in $\mathcal{F}_1$ (see Definition 3.1.1). As an application of our algorithm for solving polynomial equations, we show a deterministic polynomial time algorithm computing elliptic curve "nth roots" over F. At last, we show a probabilistic algorithm for solving polynomial equations over arbitrary finite fields with odd characteristic.

4.1 Factoring by Searching

Let $\mathbb{F}_q$ be the finite field of q elements. Let $f(x) \in \mathbb{F}_q[x]$ be a polynomial. In this section, we consider the problem of solving the polynomial equation $f(x) = 0$, i.e. finding roots of f. As in the discussion in the introduction, we may assume f is a product of two or more distinct linear factors. With some algebraic and combinatorial techniques, we show below that, in some finite fields $\mathbb{F}_q$, the problem of solving polynomial equations is polynomial time reducible to the problem of taking ℓth roots for all $\ell | q - 1$.

Write $q = p_1^{e_1} \cdots p_m^{e_m} + 1$ for some distinct primes $p_1, \cdots, p_m$. Suppose we can compute the p_jth roots of a for any $a \in \mathbb{F}_q$ and $1 \leq j \leq m$. We show a deterministic algorithm (Algorithm 4.1.1 below) to factor f, where f is product of two or more distinct linear factors and $f(0) \neq 0$.

The idea is simple: suppose $f(x)|x^d - a$ for some integer $d|q-1$ and some $a \in \mathbb{F}_q$ with $\text{ord}(a) = (q-1)/d$. Let ℓ be a prime factor of d and ζ_ℓ be a primitive ℓth root of unity in $\mathbb{F}_q$. Then, $x^d - a = \prod_{i=0}^{\ell-1}(x^{d/\ell} - \zeta_\ell^i \sqrt[\ell]{a})$. We have

$$f(x) = \prod_{i=0}^{\ell-1}(f(x), x^{d/\ell} - \zeta_\ell^i \sqrt[\ell]{a}).$$

We compute $(f(x), x^{d/\ell} - \zeta_\ell^i \sqrt[\ell]{a})$ for each $i = 0, \cdots, \ell - 1$. If $(f(x), x^{d/\ell} - \zeta_\ell^i \sqrt[\ell]{a})$ is a non-trivial factor of $f(x)$ for some $0 \leq i < \ell$, we are done (or keep factoring until the complete factorization of $f(x)$ is obtained). Otherwise, $f(x)|x^{d/\ell} - \zeta_\ell^i \sqrt[\ell]{a}$ for some $0 \leq i < \ell$. Then, repeat the process with $d' = d/\ell$ and $a' = \zeta_\ell^i \sqrt[\ell]{a}$. In the beginning, we have $f(x)|x^{q-1} - 1$ (i.e. $d = q - 1$ and $a = 1$).

Algorithm 4.1.1. factorBySearching(f)

```
{
    set a = 1;
    set d = q − 1;

    for j = 1 to m
        for k = 1 to e_j
        {
            set d = d/p_j;
Label 1:
            set b = a^{1/p_j};
            set i_0 = search(f, j, d, b);
            set a = ζ_{p_j}^{i_0} b;
        }
}
```

search(f, j, d, b)

```
{
    for i = 0 to p_j − 1
    {
        compute g(x) = (f(x), x^d − ζ^i_{p_j} b);
        if 1 < deg g < deg f
Label 2:        output g and halt;
        else if deg g = deg f
            return i;
    }
}
```

Lemma 4.1.2. *Suppose $f(x) \in \mathbb{F}_q[x]$ is product of linear factors such that $f(0) \neq 0$. Algorithm 4.1.1 always outputs a non-trivial factor of f.*

Proof. It is easy to see that a always is a p_jth residue in $\mathbb{F}_q$ at Label 1.

We show by induction that $f(x)$ divides $x^{p_j d} - a$ at Label 1 whenever the algorithm is still running. When $j = k = 1$, we have $a = 1$ and $d = (q-1)/p_1$. Obviously, $f(x)$ divides $x^{q-1} - 1$. For $j = j_0$ and $k = k_0$, denote $a_{j_0,k_0} = a$ and $d_{j_0,k_0} = d$ at Label 1. Assume $f(x)$ divides $x^{p_{j_0} d_{j_0,k_0}} - a_{j_0,k_0}$. Let $b = a_{j_0,k_0}^{1/p_{j_0}}$ and $g_i(x) = (f(x),\ x^{d_{j_0,k_0}} - \zeta^i_{p_{j_0}} b)$ for $i = 0, \cdots, p_{j_0} - 1$. Then,

$$x^{p_{j_0} d_{j_0,k_0}} - a_{j_0,k_0} = \prod_{i=0}^{p_{j_0}-1} \left(x^{d_{j_0,k_0}} - \zeta^i_{p_{j_0}} b \right) \qquad \text{and} \qquad f(x) = c \prod_{i=0}^{p_{j_0}-1} g_i(x)$$

for some constant c. If there exists g_i such that $1 < \deg g_i < \deg f$, then g_i is a non-trivial factor of f. The algorithm outputs g_i and halts. Otherwise, there exists a unique i_0 such that $\deg g_{i_0} = \deg f$ and i_0 is returned. Denote the pair of j, k followed j_0, k_0 by j_1, k_1. For $j = j_1$ and $k = k_1$, we have $a = \zeta^{i_0}_{p_{j_0}} a_{j_0,k_0}^{1/p_{j_0}}$ and $d = d_{j_0,k_0}/p_{j_1}$ at Label 1. By the definition of g_{i_0}, $f(x)$ divides $x^{p_{j_1} d} - a$.

The algorithm always outputs a non-trivial factor and halts at Label 2. Otherwise, for $j = m$ and $k = e_m$, we have $f(x)$ dividing $x^{p_m} - a$ at Label 1 but $x - \zeta^i_{p_{j_m}} a^{1/p_{j_m}}$

does not divide $f(x)$ for all $i = 0, \cdots, p_{j_m} - 1$ since the algorithm does not output. It leads to a contradiction. □

Theorem 4.1.3. *Let $\mathbb{F}_q$ be a finite field of q elements such that $\ell = O(\text{poly}(\log q))$ for every prime factor ℓ of $q-1$. Suppose there is a deterministic polynomial time algorithm to compute ℓth roots. Then, there is a deterministic polynomial time algorithm solving any polynomial equation over $\mathbb{F}_q$.*

Proof. By our previous discussion, we may assume the input polynomial $f(x) \in \mathbb{F}_q[x]$ is a product of two or more distinct linear factors and $f(0) \neq 0$. Since ℓth roots can be computed in deterministic polynomial time, Algorithm 4.1.1 can factor f in deterministic polynomial time by Lemma 4.1.2. □

Theorem 4.1.4. *Let $\mathbb{F}_q$ be a finite field of q elements such that $\ell = O(\text{poly}(\log q))$ for every prime factor ℓ of $q-1$. Given a primitive element in $\mathbb{F}_q$, there is a deterministic polynomial time algorithm solving any polynomial equation over $\mathbb{F}_q$.*

Proof. Denote the given primitive element by a. Note that a is an ℓth nonresidue in $\mathbb{F}_q$ for all prime factor ℓ of $q-1$. Then, ℓth roots can be computed by a generalized Shanks' algorithm with a as an input. See Section 3.2 in [67] for modifying Shanks' algorithm to take rth roots. Finally, the theorem follows from Theorem 4.1.3. □

Theorem 4.1.5. *Let $\mathbb{F}_q \in \mathcal{F}_1$. There is a deterministic polynomial time algorithm solving any polynomial equation over $\mathbb{F}_q$.*

Proof. It is an obvious consequence of Theorem 3.1.2 and Theorem 4.1.3. □

4.2 Elliptic curve "nth root" problem

As an application of our algorithms for solving polynomial equations, we show a deterministic polynomial-time algorithm to solve the elliptic curve "nth root" problem in this section.

Let $F \in \mathcal{F}_1$ (see Definition 3.1.1) be a finite field. Denote an elliptic curve E defined over F by the Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad \text{for some } a_1, a_2, a_3, a_4, a_6 \in F.$$

Consider the following problem: given a point $Q \in E(F)$ and a positive integer $n \in O(\mathrm{poly}(\log q))$,

(I) decide whether $Q = nP$ for some $\infty \neq P \in E(F)$,

(II) find P if such P exists.

Although we write the elliptic curve group operation additively, the nature of the problem above is closer to the nth root problem in finite fields than the multiplicative inverse problem.

It is well known that multiplication by n is an endomorphism and

$$n(x, y) = \left(\frac{R_1(x)}{S_1(x)}, y\frac{R_2(x)}{S_2(x)} \right)$$

for some $R_1(x), S_1(x), R_2(x), S_2(x) \in F[x]$ with $(R_1, S_1) = (R_2, S_2) = 1$, $\deg R_1 = n^2$ and $\deg S_1 \leq n^2 - 1$. We have $S_1(x) = \Psi(x)^2$ for some $\Psi(x) \in F[x]$. All polynomials R_1, S_1, R_2, S_2 and Ψ can be computed in deterministic polynomial time. See [69] for the details.

Suppose $(a, b) = Q \neq \infty$. Then, $Q = n(x, y)$ implies x is a solution of

$$f(x) \stackrel{\mathsf{def}}{=} R_1(x) - aS_1(x) = 0 \tag{4.2.1}$$

over F. Let $\alpha_1, \cdots, \alpha_m \in F$ be the roots of the equation (4.2.1). For (I), a solution of $Q = nP$ exists if and only if $m > 0$. For each α_i, compute $\beta_i = b\frac{S_2(\alpha)}{R_2(\alpha)}$. For (II), $\{(\alpha_i, \beta_i) \ : \ 1 \leq i \leq m\}$ is the complete set of solutions of P.

Suppose $Q = \infty$. Then, $P \in E[n](F) \stackrel{\mathsf{def}}{=} E[n] \cap E(F)$, where $E[n]$ is the n-torsion subgroup of $E(\overline{F})$, where $\overline{F}$ denotes a fixed algebraic closure of F. Let $\alpha_1, \cdots, \alpha_m \in F$ be the roots of the equation $\Psi(x) = 0$. Consider the quadratic equation

$$g_i(y) \stackrel{\mathsf{def}}{=} y^2 + (a_1\alpha_i + a_3)y - (\alpha_i^3 + a_2\alpha_i^2 + a_4\alpha_i + a_6) = 0.$$

Let $J = \{j \ : \ g_j \text{ has a root in } F, 1 \leq j \leq m\}$ be an index set. In this case, $P = \infty$ always is a solution of $Q = nP$. For (I), a solution $P \neq \infty$ of $Q = nP$ exists if and only if J is non-empty. Let $\beta_{j,1}, \beta_{j,2} \in F$ be the roots of g_j for $j \in J$. For (II), $\{(\alpha_j, \beta_{j,k}) \ : \ j \in J \text{ and } k = 1, 2\} \cup \{\infty\}$ is the complete set of solutions of P.

Theorem 4.2.1. *Let $\mathbb{F}_q \in \mathcal{F}_1$. There is a deterministic polynomial time algorithm computing elliptic curve "nth roots" over $\mathbb{F}_q$.*

Proof. The polynomial equations can be solved in deterministic polynomial time by Theorem 4.1.5. By the discussion above, the theorem follows. □

4.3 A Probabilistic Algorithm

In this section, we discuss a probabilistic algorithm for factoring products of linear polynomial over an arbitrary finite field F with characteristic not equal to 2. Our

idea is to use the fact that half of the elements in $F^\times$ are quadratic residues and the other half of the elements in $F^\times$ are quadratic nonresidues.

Let $f(x) = (x - \alpha_1) \cdots (x - \alpha_d) \in F[x]$ be a polynomial. We may assume α_i are non-zero and distinct. Suppose $\alpha_1, \cdots, \alpha_m$ are quadratic residues and $\alpha_{m+1}, \cdots, \alpha_d$ are quadratic nonresidues for some $0 < m < d$. Compute $g(x) = (f(x^2), x^{q-1} - 1)$. Then, $g(x) = (x^2 - \alpha_1) \cdots (x^2 - \alpha_m)$. Therefore, $g(\sqrt{x})$ is a non-trivial factor of f.

If all the roots of f are quadratic residues or all the roots of f are quadratic nonresidues, we can shift the roots of f by an arbitrary element $a \in F$ and try to factor the shifted polynomial. The algorithm shown below captures this idea with a as an input. If the algorithm fails for some a, we can try again with a different a.

Algorithm 4.3.1. $\mathsf{factor}(f, a)$

```
{
    if f(a) = 0
        output x − a;

    compute g(x) = (f(x^2 + a), x^{q−1} − 1);
    if 0 < deg g < 2 deg f
        output g(√(x − a));
}
```

Proposition 4.3.2. *For any finite field F, let $f(x) = (x - \alpha_1) \cdots (x - \alpha_d) \in F[x]$ for some distinct $\alpha_i \in F^\times$. The success probability of Algorithm 4.3.1 for any $a \in F$ is approximately $1 - 2^{1-d}$, where $d = \deg f \geq 2$.*

Proof. The probability of each $\alpha_i - a$ being a quadratic residue is approximately $1/2$. Algorithm 4.3.1 does not work when all $(\alpha_i - a)$'s are quadratic residues or all $(\alpha_i - a)$'s are quadratic nonresidues. Therefore, the overall success probability is approximately $1 - 2^{1-d}$. □

The running time of Algorithm 4.3.1 is $\tilde{O}(d \log q)$ bit operations, where q is the number of element in F and d is the degree of f. The most costly operation in the algorithm is computing GCD, which takes $\tilde{O}(d \log q)$ bit operations. The other operations in the algorithm are obviously bounded by it.

Algorithm 4.3.1 can be generalized by replacing the shift operation by a with some 1-1 mapping σ over F. The mapping σ should

- be efficiently computable for both σ and σ^{-1};
- induce an 1-1 mapping from polynomial to polynomial;
- map quadratic residues and quadratic nonresidues randomly.

In Algorithm 4.3.1, we use $\sigma_a : x \longmapsto x - a$ and the induced polynomial map is $\tau_a : f(x) \longmapsto f(x + a)$. A more general map is

$$\sigma_{a,b,c,d} : x \longmapsto \frac{ax + b}{cx + d} \qquad \text{for some } a, b, c, d \in F \text{ with } ad - bc \neq 0$$

and an induced polynomial map is

$$\tau_{a,b,c,d} : f(x) \longmapsto (cx + d)^{\deg f} f(\frac{ax + b}{cx + d}).$$

Chapter 5

Primality Testing

Primality testing is the process of checking whether a positive integer is a prime. The problem of primality testing is in great interest in modern research since many modern cryptographic schemes rely on finding large prime numbers. One typical example is the RSA public key cryptosystem [53].

Directly from the definition, if N is composite, then there exists a prime $p \leq \sqrt{N}$ such that N is divisible by p. However, checking all prime $p \leq \sqrt{N}$ requires exponential time to the input size. This idea was known to ancient Greeks.

Another test, called Fermat's test, is to find an integer a with $(a, N) = 1$ such that $a^{N-1} \not\equiv 1 \pmod{N}$. If such a exists, then N is composite by Fermat's Little Theorem (Theorem 1.2.1). Fermat's test does not prove primality. Even if

$$a^{N-1} \equiv 1 \pmod{N} \qquad \text{for all } a \text{ with } (a, N) = 1, \tag{5.0.1}$$

the integer N may not be a prime since there exist infinitely many composite numbers, called Carmichael numbers (see [6]), satisfying equation (5.0.1). Fermat's test also

fails to prove the compositeness of Carmichael numbers.

For some forms of numbers, there are specific primality tests. We have Lucas-Lehmer (see [70]) for Mersenne numbers ($N = 2^q - 1$ for some prime q) and Pépin's test [47] for Fermat number ($N = 2^{2^n} + 1$). Both algorithms are $\tilde{O}(\log^2 N)$ and deterministic.

In 2002, Agrawal, Kayal and Saxena (AKS) gave the first deterministic polynomial-time primality testing algorithm [5]. See also [26]. We will discuss more details of AKS and the related works in Section 5.2. Before AKS, there were many primality testing algorithms which are either probabilistic or not polynomial-time: Pocklington-Lehmer [48, 39], Miller-Rabin [44, 49], Solovay-Strassen [59], Adleman-Pomerance-Rumely [3], elliptic curve primality proving [30, 9], and some other tests [23], [1] and [4]. See [31] for a survey.

In this chapter, we show a deterministic polynomial-time primality test for some form of numbers in Section 5.1 and present a potentially fast primality test based on AKS in Section 5.2.

5.1 rth Root Primality Test

In this section, we show a deterministic primality testing algorithm, for some form of numbers. The primality test is constructed by our rth root algorithms presented in Chapter 3. The idea of our primality test is similar to the Pocklington-Lehmer primality test [48, 39]. We have the following theorem.

Theorem 5.1.1. *Let $N = r^e t + 1$ for some prime r and some positive integers t and*

e with $r^e > t$. There is an $\tilde{O}(r^2(\log^2 N)(t + r\log N))$ deterministic primality testing algorithm. If r is a small constant and $t = O(\log N)$, the running time is $\tilde{O}(\log^3 N)$.

Proof. Firstly, we try to find a primitive rth root of unity ζ_r over $(\mathbb{Z}/N\mathbb{Z})^\times$ by Algorithm 2.4.8. If $\zeta_r \notin (\mathbb{Z}/N\mathbb{Z})^\times$, Algorithm 2.4.8 will fail and we conclude that N is composite. Otherwise, try computing $\zeta_{r^2} = \sqrt[r]{\zeta_r}$, $\zeta_{r^3} = \sqrt[r]{\zeta_{r^2}}$, $\cdots$, $\zeta_{r^e} = \sqrt[r]{\zeta_{r^{e-1}}}$ over the ring $\mathbb{Z}/N\mathbb{Z}$ by Algorithm 3.6.1. If N is prime, we will obtain ζ_{r^e} eventually. If N is composite, ζ_{r^e} does not exist in $\mathbb{Z}/N\mathbb{Z}$ by a generalized Proth's Theorem (Theorem 5.1.4 in the next section). Since Algorithm 3.6.1 is deterministic, it must fail in some point during computing ζ_{r^e}. Therefore, N is prime if and only if ζ_{r^e} can be computed successfully by the procedure described.

The running time of Algorithm 2.4.8 and Algorithm 3.6.1 are $\tilde{O}(t\log^2 N)$ and $\tilde{O}(r^2\log N(t + r\log N))$, respectively. Since Algorithm 3.6.1 is used $e - 1$ times, the overall running time is $\tilde{O}(r^2\log^2 N(t + r\log N))$. The theorem follows. □

For $N = r^e t + 1$ with r a small constant and $t = O(\log N)$, the running time our algorithm is $\tilde{O}(\log^3 N)$, which is faster than other deterministic primality tests which are applicable. The running time of the AKS test [5] and Lenstra-Pomerance's modified AKS test [34] are $\tilde{O}(\log^{7.5} N)$ and $\tilde{O}(\log^6 N)$, respectively. Assuming ERH (see Section 1.2.4), Miller's test [44] is deterministic with running time $\tilde{O}(\log^4 N)$.

5.1.1 Proth's Theorem

In 1878, a self-taught farmer, Francois Proth, proved the following theorem.

Theorem 5.1.2. (Proth's Theorem) *Let $N = 2^e t + 1$ for some odd t with $2^e > t$. If*

$$a^{(N-1)/2} \equiv -1 \pmod{N}$$

for some a, then N is a prime.

See [70] for the details of Proth's Theorem. We will show a generalization of Proth's theorem (Theorem 5.1.4). This generalization of Proth's theorem is well known. The following lemma is used in the proof of Theorem 5.1.4.

Lemma 5.1.3. *Let $n = \ell^k$ for some prime ℓ and $k \geq 1$. Let r^e be a prime power with $r \neq \ell$. If $r^e | \phi(n)$ and $r^e > \sqrt{n}$, then n is a prime (i.e. $k = 1$).*

Proof. We have r^e dividing $\phi(n) = (\ell - 1)\ell^{k-1}$, therefore, $r^e | \ell - 1$. If $k > 1$, then $\phi(n) \geq (\ell - 1)\ell > r^{2e} > n$, which is a contradiction. Thus, $k = 1$ and n is a prime. □

Theorem 5.1.4. (Generalized Proth's Theorem) *Let $N = r^e t + 1$ for some prime r and integers $e, t \geq 1$. Suppose $r^e > t$. If*

$$a^{N-1} \equiv 1 \pmod{N} \qquad \text{and} \qquad a^{(N-1)/r} \not\equiv 1 \pmod{N}, \tag{5.1.1}$$

for some integer a, then N is a prime.

Proof. Suppose there exists an integer a satisfying equations (5.1.1). Let d be the order of a in $(\mathbb{Z}/N\mathbb{Z})^\times$. Then $r^e | d$. Let $b \equiv a^{d/r^e} \pmod{N}$. The order of b in $(\mathbb{Z}/N\mathbb{Z})^\times$ is r^e. Note that $r^e > \sqrt{N}$.

Suppose $N = \ell^k$ for some prime ℓ and $k \geq 1$. Since $(N, r) = 1$, we have $\ell \neq r$. The order of b, r^e divides $\phi(N)$. By Lemma 5.1.3, $k = 1$ and N is a prime.

Suppose $N = \ell_1^{k_1} \cdots \ell_m^{k_m}$ for $m > 1$, some distinct primes $\ell_1, \cdots, \ell_m$ and integers $k_1, \cdots, k_m \geq 1$. Let d_i be the order of b in $(\mathbb{Z}/\ell_i^{k_i}\mathbb{Z})^\times$. Since $b^{r^e} \equiv 1 \pmod{\ell_i^{k_i}}$, we have $d_i = r^{s_i}$ for some $0 \leq s_i \leq e$. Without loss of generality, assume $s_1 = \max(s_1, \cdots, s_m)$. If $s_1 < e$, we have $b^{r^{s_1}} \equiv 1 \pmod{\ell_i^{k_i}}$ for all $1 \leq i \leq m$. By the Chinese Remainder Theorem, $b^{r^{s_1}} \equiv 1 \pmod{N}$ but r^e does not divide r^{s_1}, contradiction. Therefore, $s_1 = e$. We have $r^e | \phi(\ell_1^{k_1})$, which implies $k_1 = 1$ by Lemma 5.1.3. Write $\ell_1 = r^e t_1 + 1$ and $N/\ell_1 = r^{e_0} t_0 + 1$ with $(r, t_0) = 1$. Then $N = (t_0 t_1 r^{e_0} + t_1 + t_0 r^{e_0 - e}) r^e + 1$. We have $e_0 \geq e$, otherwise, $t = t_0 t_1 r^{e_0} + t_1 + t_0 r^{e_0 - e}$ is not an integer. However, $N = \ell_1(N/\ell_1) > r^{e+e_0} \geq r^{2e} > N$, contradiction. □

5.2 A Potentially Fast Primality Test

In 2002, Agrawal, Kayal and Saxena [5] gave the first deterministic, polynomial-time primality testing algorithm. The main step was the following.

Theorem 5.2.1. (AKS) *Given an integer $n > 1$, let r be an integer such that $\mathrm{ord}_r(n) > \log^2 n$. Suppose*

$$(x + a)^n \equiv x^n + a \pmod{n, x^r - 1} \qquad \text{for } a = 1, \cdots, \lfloor \sqrt{\phi(r)} \log n \rfloor. \tag{5.2.1}$$

Then, n has a prime factor $\leq r$ or n is a prime power.

The running time is $\tilde{O}(r^{1.5} \log^3 n)$. It can be shown by elementary means that the required r is $O(\log^5 n)$. So the running time is $\tilde{O}(\log^{10.5} n)$. Moreover, by Fouvry's Theorem [27], such r exists in $O(\log^3 n)$, so the running time becomes $\tilde{O}(\log^{7.5} n)$. It is

conjectured that such r exists in $O(\log^2 n)$, which makes the running time $\tilde{O}(\log^6 n)$. However, the conjecture is still not proved yet.

In [34], Lenstra and Pomerance showed that the AKS primality test can be improved by replacing the polynomial $x^r - 1$ in equation (5.2.1) with a specially constructed polynomial $f(x)$, so that the degree of $f(x)$ is $O(\log^2 n)$. The overall running time of their algorithm is $\tilde{O}(\log^6 n)$.

With an extra input integer a, Berrizbeitia [16] has provided a deterministic primality test with time complexity $2^{-\min(k,\lfloor 2\log\log n\rfloor)}\tilde{O}(\log^6 n)$, where $2^k||n-1$ if $n \equiv 1 \pmod 4$ and $2^k||n+1$ if $n \equiv 3 \pmod 4$. If $k \geq \lfloor 2\log\log n\rfloor$, this algorithm runs in $\tilde{O}(\log^4 n)$. The algorithm is also a modification of AKS by verifying the congruence

$$(1+mx)^n \equiv 1 + mx^n \pmod{n, x^{2^s} - a}$$

for a fixed s and some clever choices of m. The main drawback of this algorithm is that it requires a satisfying the Jacobi symbol $\left(\frac{a}{n}\right) = -1$ if $n \equiv 1 \pmod 4$ and $\left(\frac{a}{n}\right) = \left(\frac{1-a}{n}\right) = -1$ if $n \equiv 3 \pmod 4$. Since there is no deterministic algorithm to find such a yet, Berrizbeitia's algorithm is considered as a probabilistic test.

We attempt to improve the AKS primality test in another direction. We suggest that equation (5.2.1) may be checked with only the single value $a = -1$. If a certain conjecture (Conjecture 5.2.8) about cyclotomic polynomials holds, we obtain a deterministic primality testing algorithm with running time $\tilde{O}(r\log^2 n)$. The requirement of r is exactly the same as in AKS. Therefore, the running time would be $\tilde{O}(\log^5 n)$ if r is $O(\log^3 n)$.

5.2.1 The Algorithm

Let SimplePrimalityTest(n) be an $O(\sqrt{n})$ primality test algorithm. We show our algorithm below.

Algorithm 5.2.2. PrimalityTest(n)

{
    if $n < n_0 = 8 \times 10^5$, return SimplePrimalityTest(n);
    if $n = a^e$ for some prime a and some $e > 1$, return COMPOSITE;

    find smallest r such that $\mathrm{ord}_r(n) > \log^2 n$;
    if $1 < (a, n) < n$ for some $a \leq r$, return COMPOSITE;
    if $n \leq r$, return PRIME;

    if $(x-1)^n \not\equiv x^n - 1 \pmod{n, x^r - 1}$, return COMPOSITE;
    return PRIME;
}

Throughout this section, suppose $n > 1$ is an integer and Algorithm 5.2.2 returns PRIME at the last line. Therefore,

- n is not a non-trivial power of a prime (that is, $n \neq p^e$ for some $e > 1$),
- $\mathrm{ord}_r(n) > \log^2 n$,
- all prime divisors of n are greater than r
- $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$.

Let p be a prime dividing n such that $\mathrm{ord}_r(p) > 1$. Since $\mathrm{ord}_r(n) > 1$, such a prime p exists. Let

$$G = \left\{ (\frac{n}{p})^i p^j \pmod{r} \ : \ i, j \in \mathbb{Z} \right\} \subset (\mathbb{Z}/r\mathbb{Z})^\times.$$

Let $t = |G|$. Let $h(x)$ be an irreducible factor of $\Phi_r(x)$ in $\mathbb{F}_p$. Then, $\deg(h) = \mathrm{ord}_r(p) > 1$. Let

$$F = (\mathbb{Z}/p\mathbb{Z})[x]/(h(x)),$$

which is isomorphic to the finite field $\mathbb{F}_{p^{\deg(h)}}$. Let

$$P = \{f \in \mathbb{Z}[x] \ : \ f(x)^n \equiv f(x^n) \pmod{p, \Phi_r(x)}\}$$

and

$$\mathcal{G} = \{f(x) \pmod{p, h(x)} \ : \ f \in P\} \subset F^\times.$$

In F, it can be shown that $f(x)^n = f(x^n)$ implies $f(x)^{n/p} = f(x^{n/p})$ (see [5] for a proof). Since p is the characteristic of F, we have $f(x)^p = f(x^p)$. Therefore, for all $f \in \mathcal{G}$, we have $f(x)^m = f(x^m)$ for all $m \in G$.

5.2.2 Upper bounds of $|\mathcal{G}|$

Some upper bounds of the size of $|\mathcal{G}|$ can be shown as follows.

Lemma 5.2.3. *Suppose n is not a power of p. Then, $|\mathcal{G}| \leq n^{\sqrt{t}}$.*

Proof. The proof is essentially the same as the proof of Lemma 4.8 in [5]. □

Lemma 5.2.4. *Suppose n is not a power of p. If $\mathrm{ord}_r(n) > \sqrt{t} \geq \sqrt{384}$, then $|\mathcal{G}| \leq n^{\sqrt{t}-2/5}$.*

Proof. Suppose $n^{2/5} \leq p \leq n^{3/5}$. Let

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

The size of $\hat{I}$ satisfies $|\hat{I}| = (\lfloor\sqrt{t}\rfloor + 1)^2 > t$. Since $G = \hat{I} \pmod{r}$ and $|G| = t$, there exist $m_1, m_2 \in \hat{I}$ with $m_1 < m_2$ such that $m_1 \equiv m_2 \pmod{r}$. Consider the polynomial $\psi(T) = T^{m_2-m_1} - 1 \in F[T]$. For all $f(x) \in \mathcal{G}$,

$$\begin{aligned}\psi(f(x)) &= f(x)^{m_2-m_1} - 1 \\ &= \frac{f(x^{m_2})}{f(x^{m_1})} - 1 \\ &= 0.\end{aligned}$$

Therefore, $\psi(T)$ has at least $|\mathcal{G}|$ roots in F.

Let

$$M = \max\left\{(\frac{n}{p})^{\lfloor\sqrt{t}\rfloor}p^{\lfloor\sqrt{t}\rfloor-1}, (\frac{n}{p})^{\lfloor\sqrt{t}\rfloor-1}p^{\lfloor\sqrt{t}\rfloor}\right\}.$$

Note that $M \leq n^{\lfloor\sqrt{t}\rfloor-2/5} \leq n^{\sqrt{t}-2/5}$ since both p, $\frac{n}{p} \geq n^{2/5}$. We claim that m_1, m_2 can be chosen such that $m_2 - m_1 \leq M$. This implies that $|\mathcal{G}| \leq \deg(\psi) = m_2 - m_1 \leq n^{\lfloor\sqrt{t}\rfloor-2/5}$.

To prove the claim, let $m_1' \equiv m_2' \pmod{r}$ with $m_1', m_2' \in \hat{I}$ and $m_1' < m_2'$.

If $m_2' < n^{\lfloor\sqrt{t}\rfloor}$, then $m_2' \leq (n/p)^i p^j$ with either $i < \lfloor\sqrt{t}\rfloor$ or $j < \lfloor\sqrt{t}\rfloor$. Then $m_2' \leq M$, so $m_2' - m_1' \leq M$. We can set $m_1 = m_1'$ and $m_2 = m_2'$. The case $m_1' = 1$ and $m_2' = n^{\lfloor\sqrt{t}\rfloor}$ is not possible; otherwise, $1 \equiv n^{\lfloor\sqrt{t}\rfloor} \pmod{r}$, so $\mathrm{ord}_r(n) \leq \lfloor\sqrt{t}\rfloor$. Finally, assume $1 \neq m_1' < m_2' = n^{\lfloor\sqrt{t}\rfloor}$. The definition of $\hat{I}$ shows that $m_1' | n^{\lfloor\sqrt{t}\rfloor} = m_2'$. Choose $m_1 = 1$ and $m_2 = m_2'/m_1'$. Since $m_1' \geq \min\left\{p, \frac{n}{p}\right\} \geq n^{2/5}$, this completes the proof of the claim.

Now suppose that $p < n^{2/5}$ or $p > n^{3/5}$, Let $n^\delta = \min\left\{p, \frac{n}{p}\right\}$ with $0 < \delta < \frac{2}{5}$. Then $n^{1-\delta} = \max\left\{p, \frac{n}{p}\right\}$. Let

$$\tilde{I} = \left\{n^{\delta i}n^{(1-\delta)j} \ : \ 0 \leq i \leq A \text{ and } 0 \leq j \leq B\right\},$$

where $A = \left\lfloor \sqrt{\frac{t(1-\delta)}{\delta}} \right\rfloor$ and $B = \left\lfloor \sqrt{\frac{t\delta}{1-\delta}} \right\rfloor$. Then $|\tilde{I}| = (A+1)(B+1) > t$. As before, there exist $m_3, m_4 \in \tilde{I}$, such that $m_3 \equiv m_4 \pmod{r}$ with $m_3 < m_4$. Note that $m_4 \leq n^{A\delta} n^{B(1-\delta)} \leq n^{2\sqrt{t\delta(1-\delta)}} \leq n^{\sqrt{24t/25}} < n^{\sqrt{t}-2/5}$ for $t \geq 384$. All the elements in $\mathcal{G}$ are roots of the polynomial $T^{m_4} - T^{m_3}$. Therefore, $|\mathcal{G}| \leq m_4 \leq n^{\sqrt{t}-2/5}$. □

5.2.3 Producing elements of $\mathcal{G}$

One way to find a lower bound on the size of $\mathcal{G}$ is to produce a large number of elements of $\mathcal{G}$. If we have chosen r so that n is a primitive root mod r, then this is easy.

Lemma 5.2.5. *Assume that n is a primitive root mod r and that $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$. If $(m, r) = 1$, then*

$$x^m - 1 \equiv (x-1)^e \pmod{n, x^r - 1}$$

for some integer e.

Proof. Write $m \equiv n^f \pmod{r}$. Then

$$x^m - 1 \equiv x^{n^f} - 1 \equiv (x-1)^{n^f}.$$

□

Consider the cyclotomic field of rth roots of unity $\mathbb{Q}(\zeta)$, where ζ is a primitive rth root of unity. The cyclotomic units are generated by the quotients $(\zeta^a - 1)/(\zeta - 1)$ with $(a, r) = 1$. The index of these units in the full group of units of the ring $\mathbb{Z}[\zeta]$ is the class number of the real subfield $\mathbb{Q}(\zeta + \zeta^{-1})$. This class number tends to be rather

small, so the cyclotomic units are of small index in the full group of units. Let p be prime and let $\mathfrak{p}$ be a prime ideal of $\mathbb{Z}[\zeta]$ dividing p. The field $\mathbb{Z}[\zeta]/\mathfrak{p}$ is isomorphic to $\mathbb{F}_p[x]/(p, h(x))$, where $h(x)$ is an irreducible factor mod p of $\Phi_r(x)$. Work on Artin's primitive root conjecture (see, for example, [21]) shows that the reduction mod $\mathfrak{p}$ of the group of units of $\mathbb{Z}[\zeta]$ should often be quite large. In fact, it is conjectured to be the full multiplicative group of $\mathbb{Z}[\zeta]/\mathfrak{p}$ for a positive density of primes p. Since the index of the cyclotomic units tends to be small, we expect that the cyclotomic units also generate a large subgroup of the multiplicative group. Therefore, the polynomials $x^m - 1$ should generate a large subgroup of $\mathbb{F}_p[x]/(p, h(x))$, so we expect that the group $\mathcal{G}$ should be large for many p.

In the next section, we formulate a conjecture on cyclotomic polynomials (Conjecture 5.2.8) that can be regarded as a way of producing a large number of polynomials in $\mathcal{G}$. In the case that n is a primitive root mod r, the following lemma shows that the group obtained is contained in the group generated by $x - 1$.

Lemma 5.2.6. *If n is a primitive root mod r and $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$, then*

$$\Phi_m(x) \in \{(x-1)^e \ : \ e \in \mathbb{Z}\} \subset F^\times$$

for $(m, r) = 1$.

Proof. Since

$$\Phi_m(x) = \prod_{d|m} (x^{m/d} - 1)^{\mu(d)}$$

where $\mu(d)$ is the Möbius function, the previous lemma yields the result. □

5.2.4 Cyclotomic polynomials

We conjecture that once equation (5.2.1) is verified with $a = -1$, the size of $\mathcal{G}$ is larger than the upper bounds in Lemma 5.2.3 and 5.2.4. If the conjecture is true, then n must be a prime when Algorithm 5.2.2 returns PRIME at the last line.

In particular, we have the mth cyclotomic polynomial $\Phi_m(x) \in \mathcal{G}$ for all $m > 0$ with $(m, r) = 1$ as shown in Lemma 5.2.7. Since there are infinitely many distinct $\Phi_m(x)$ in $\mathbb{Z}[x]$, some of them must be congruent to each other in F. We will show that there exist r distinct $\Phi_q(x)$'s in F for q prime (see Lemma 5.2.14). By Lemma 5.2.15, for primes p_1 and q_1, $\Phi_{p_1}(x)$ and $\Phi_{q_1}(x)$ are distinct whenever $p_1 \not\equiv q_1 \pmod{r}$. Conjecture 5.2.8 suggests a generalized situation that $\Phi_{p_1 \cdots p_k}(x)$ and $\Phi_{q_1 \cdots q_k}(x)$ are distinct unless $p_i \equiv q_{\sigma(i)} \pmod{r}$ for all $1 \leq i \leq k$ and some permutation σ. Proposition 5.2.16 proves Conjecture 5.2.8 with $k = 2$.

Lemma 5.2.7. *If* $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$*, then for* $k \geq 1$ *with* $(k, r) = 1$*,*

$$\Phi_k(x)^n \equiv \Phi_k(x^n) \pmod{p, \Phi_r(x)}. \tag{5.2.2}$$

Proof. We use induction. By the hypothesis, $\Phi_1(x) = x - 1$ satisfies the conclusion because $p|n$ and $\Phi_r(x)$ divides $x^r - 1$. Suppose $\Phi_i(x)^n \equiv \Phi_i(x^n) \pmod{p, \Phi_r(x)}$ for $1 \leq i < k$ with $(i, r) = 1$.

For $k > 1$ with $(k, r) = 1$, the congruence $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$ implies that

$$(x^k - 1)^n \equiv x^{kn} - 1 \pmod{n, x^{kr} - 1}.$$

Since $p|n$ and $\Phi_r(x)$ divides $x^{kr}-1$,

$$(x^k-1)^n \equiv (x^n)^k - 1 \pmod{p, \Phi_r(x)}.$$

By the identity $T^k - 1 = \prod_{d|k} \Phi_d(T)$,

$$\left(\prod_{d|k} \Phi_d(x)\right)^n \equiv \prod_{d|k} \Phi_d(x^n) \pmod{p, \Phi_r(x)}. \tag{5.2.3}$$

For any proper divisor d' of k, $(d', r) = 1$ and $\Phi_{d'}(x)^n \equiv \Phi_{d'}(x^n) \pmod{p, \Phi_r(x)}$ by the induction assumption. Let $g(x) = (\Phi_{d'}(x), \Phi_r(x)) \in \mathbb{F}_p[x]$. If $g(x) \neq 1$, let $\alpha \in \overline{\mathbb{F}}_p$ be a root of $g(x)$. Then, $\alpha^{d'} = 1$ and $\alpha^r = 1$. But $(d', r) = 1$ implies that $\alpha = 1$. However, $\Phi_r(1) = \sum_{i=0}^{r-1} 1 = r \neq 0$ in $\mathbb{F}_p$ since r, p are distinct primes. Therefore, $(\Phi_{d'}(x), \Phi_r(x)) = 1$. so equation 5.2.3 yields

$$\Phi_k(x)^n \equiv \Phi_k(x^n) \pmod{p, \Phi_r(x)}.$$

□

Conjecture 5.2.8. *Let $p_1, p_2, \cdots, p_k$ be prime numbers that are distinct mod r and none of them are congruent to $-1, 0, 1 \pmod r$. Similarly, let $q_1, q_2, \cdots, q_k$ be primes that are distinct mod r and none of them are congruent to $-1, 0, 1 \pmod r$. Let $h(x)$ be an irreducible factor of $\Phi_r(x)$ mod p. Then,*

$$\Phi_{p_1 p_2 \cdots p_k}(x) \equiv \Phi_{q_1 q_2 \cdots q_k}(x) \pmod{p, h(x)}$$

if and only if there is a permutation σ of $\{1, 2, \ldots, k\}$ such that

$$p_i \equiv q_{\sigma(i)} \pmod r \qquad \text{for } i = 1, 2, \cdots, k.$$

One direction of this conjecture can be proved. In Section 5.2.6, we give evidence for the other direction.

Proof of "⇐". We prove a stronger version of the statement:

$$p_i \equiv q_i \pmod{r} \qquad \text{for } i = 1, 2, \cdots, k,$$

implies

$$\Phi_{p_1 p_2 \cdots p_k}(x) \equiv \Phi_{q_1 q_2 \cdots q_k}(x) \pmod{p, \Phi_r(x)}.$$

We show it by induction. For $k = 1$, the statement is true by Lemma 5.2.15. For $k > 1$, suppose $p_i \equiv q_i \pmod{r}$ for $i = 1, 2, \cdots, k$. By the induction assumption,

$$\Phi_{p_1 p_2 \cdots p_{k-1}}(y) \equiv \Phi_{q_1 q_2 \cdots q_{k-1}}(y) \pmod{p, \Phi_r(y)}.$$

Put $y = x^{p_k}$. We have

$$\Phi_{p_1 p_2 \cdots p_{k-1}}(x^{p_k}) \equiv \Phi_{q_1 q_2 \cdots q_{k-1}}(x^{p_k}) \pmod{p, \Phi_r(x^{p_k})}.$$

Since $\Phi_r(x)$ divides $\Phi_r(x^{p_k})$,

$$\Phi_{p_1 p_2 \cdots p_{k-1}}(x^{p_k}) \equiv \Phi_{q_1 q_2 \cdots q_{k-1}}(x^{p_k}) \pmod{p, \Phi_r(x)}.$$

We claim that $\Phi_{p_1 p_2 \cdots p_{k-1}}(x)$ and $\Phi_r(x)$ are relatively prime mod p. To see this, let $\alpha \in \overline{\mathbb{F}_p}$ be a common root mod p of the two polynomials. Then $\alpha^{p_1 p_2 \cdots p_{k-1}} = 1 = \alpha^r$, so $\alpha = 1$. But $\Phi_r(1) = r \not\equiv 0 \pmod{p}$. Therefore, the two polynomials have no common root mod p, which proves the claim. So $\Phi_{p_1 p_2 \cdots p_{k-1}}(x)$ is a unit mod $\Phi_r(x)$. Finally,

$$\begin{aligned}
\Phi_{p_1 p_2 \cdots p_k}(x) &= \frac{\Phi_{p_1 p_2 \cdots p_{k-1}}(x^{p_k})}{\Phi_{p_1 p_2 \cdots p_{k-1}}(x)} \\
&\equiv \frac{\Phi_{q_1 q_2 \cdots q_{k-1}}(x^{p_k})}{\Phi_{q_1 q_2 \cdots q_{k-1}}(x)} \pmod{p, \Phi_r(x)} \\
&\equiv \frac{\Phi_{q_1 q_2 \cdots q_{k-1}}(x^{q_k})}{\Phi_{q_1 q_2 \cdots q_{k-1}}(x)} \pmod{p, \Phi_r(x)} \\
&= \Phi_{q_1 q_2 \cdots q_k}(x).
\end{aligned}$$

□

In Conjecture 5.2.8, we require $p_i, q_i \not\equiv -1, 0, 1 \pmod{r}$ for $i = 1, 2, \cdots, k$, otherwise, the conjecture is obviously false. For any prime q, if $q \equiv 0 \pmod{r}$, then $q = r$ and $\Phi_q(x) \equiv 0 \pmod{\Phi_r(x)}$ is not a unit. If $q \equiv 1 \pmod{r}$, we have $\Phi_q(x) \equiv 1 \pmod{\Phi_r(x)}$, which is the multiplicative identity. Then, $\Phi_{qm}(x) \equiv 1 \pmod{\Phi_r(x)}$ for any integer $m > 0$. If $q \equiv -1 \pmod{r}$, then $\Phi_q(x) \equiv -x^{-1} \pmod{\Phi_r(x)}$. The subgroup of $F^\times$ generated by $-x^{-1}$ has only $2r$ elements, where $F = (\mathbb{Z}/p\mathbb{Z})[x]/(h(x))$. We have $\Phi_{qm_1}(x) \equiv \Phi_{qm_2}(x) \pmod{\Phi_r(x)}$ for some[1] $m_1 \equiv m_2 \pmod{2r}$.

5.2.5 Lower bound for $|\mathcal{G}|$

Assuming Conjecture 5.2.8 is true, we establish a lower bound for $|\mathcal{G}|$ in Lemma 5.2.10 that implies the correctness of Algorithm 5.2.2. See Theorem 5.2.11.

Recall the following.

Theorem 5.2.9. (Stirling's approximation) *For $N > 0$,*

$$\sqrt{2\pi N}(\frac{N}{e})^N e^{1/(12N+1)} < N! < \sqrt{2\pi N}(\frac{N}{e})^N e^{1/(12N)}.$$

Lemma 5.2.10. *If Conjecture 5.2.8 is true, then $|\mathcal{G}| > \frac{1}{11}\frac{2^r}{\sqrt{r}}$.*

Proof. If $r \leq 5$, then $\frac{1}{11}\frac{2^r}{\sqrt{r}} < 2 \leq |\mathcal{G}|$ since $\mathcal{G}$ has as least two elements, x and $x - 1$.

Suppose $r > 5$, i.e. $r \geq 7$. If Conjecture 5.2.8 is true, there are $\binom{r-3}{k}$ distinct $\Phi_{p_1 p_2 \cdots p_k}(x)$ in $\mathcal{G}$ by Lemma 5.2.7 and Dirichlet's Theorem (Theorem 5.2.13). Consider

[1]For example, let m_1 and m_2 be distinct primes such that $m_1 \equiv m_2 \equiv m \pmod{2r}$ with $m \not\equiv -1, 0, 1 \pmod{r}$.

$k = \frac{r-3}{2}$. By Theorem 5.2.9,

$$\begin{aligned}
\frac{(r-3)!}{(((r-3)/2)!)^2} &> \frac{(\frac{r-3}{e})^{r-3} e^{1/(12r-35)} \sqrt{2\pi(r-3)}}{\left((\frac{r-3}{2e})^{(r-3)/2} e^{1/(6r-18)} \sqrt{2\pi(\frac{r-3}{2})}\right)^2} \\
&= \frac{2^r}{\sqrt{32\pi(r-3)}} e^{\frac{1}{12r-35} - \frac{1}{3r-9}} \\
&> \frac{e^{\frac{1}{49} - \frac{1}{12}}}{\sqrt{32\pi}} \frac{2^r}{\sqrt{r}} \\
&> \frac{1}{11} \frac{2^r}{\sqrt{r}}.
\end{aligned}$$

Therefore, $|\mathcal{G}| \geq \binom{r-3}{(r-3)/2} > \frac{1}{11}\frac{2^r}{\sqrt{r}}$, as required. □

Theorem 5.2.11. *If Conjecture 5.2.8 is true, then Algorithm 5.2.2 returns PRIME at the last line only if n is a prime.*

Proof. If algorithm 5.2.2 returns PRIME at the last line, then r is an odd prime, n is not a nontrivial power of a prime, $n \geq n_0 = 8 \times 10^5$, and $\mathrm{ord}_r(n) = r - 1 > \log^2 n$. Moreover, all prime divisors of n are greater than r, and $(x-1)^n \equiv x^n - 1 \pmod{n, x^r - 1}$.

Suppose $\mathrm{ord}_r(n) > \sqrt{t} \geq \sqrt{384}$. Let $c = (\log^2 n)/r > 1$ and let

$$f(c, n) = \frac{n^{2/5}}{\log n} \left(\frac{n^{(c-\sqrt{c})\log n}}{\sqrt{c}} \right).$$

Note that $\frac{n^{2/5}}{\log n}$ is increasing for $n > \sqrt{32}$. So $f(c, n)$ is increasing in n for $n > \sqrt{32}$ and $c \geq 1$. The term $\frac{n^{(c-\sqrt{c})\log n}}{\sqrt{c}}$ is increasing in c for $c \geq 1$.

For $n > 392$ and $c \geq c_0 = 1.084$,

$$\frac{n^{2/5}}{\log n} \left(\frac{n^{(c-\sqrt{c})\log n}}{\sqrt{c}} \right) = f(c, n) > f(c_0, 384) > 11,$$

which implies that

$$\frac{1}{11}\frac{2^r}{\sqrt{r}} = \frac{1}{11} \left(\frac{n^{c \log n}}{\sqrt{c} \log n} \right) > n^{\sqrt{c}\log n - \frac{2}{5}} = n^{\sqrt{r} - \frac{2}{5}}.$$

If $1 \leq c < c_0$ and $n \geq n_0$, then $\frac{n^{2/5}}{\log n} > 11\sqrt{c_0}$ for $n \geq n_0$ and

$$\frac{1}{11}\frac{2^r}{\sqrt{r}} > \frac{1}{11}\left(\frac{n^{\sqrt{r}}}{\sqrt{c_0}\log n}\right) > n^{\sqrt{r}-\frac{2}{5}}.$$

If n is not a power of p, then Lemma 5.2.4 and Lemma 5.2.10 together imply that

$$n^{\sqrt{n}-\frac{2}{5}} \geq |\mathcal{G}| > n^{\sqrt{n}-\frac{2}{5}},$$

which is a contradiction. Since the algorithm removes nontrivial powers of primes, we must have that n is a prime.

Now suppose that $\mathrm{ord}_r(n) \leq \sqrt{t}$. Then, $r > t \geq (\mathrm{ord}_r\, n)^2 > \log^4 n$. We have

$$\frac{1}{11}\frac{2^r}{\sqrt{r}} \geq \frac{1}{11}\frac{n^{\log n\sqrt{r}}}{\sqrt{r}} > n^{\sqrt{r}}$$

for $n \geq 5$ and $r \geq 3$, which includes all possible values of n and r. By Lemma 5.2.3 and Lemma 5.2.10, n is a prime. □

Note that it is possible to minimize the value of n_0 by manipulating the parameters in the proof of Theorem 5.2.11. However, such minimization is unnecessary for any practical use of Algorithm 5.2.2 because $n_0 = 8 \times 10^5$ is small enough for running an $O(\sqrt{n})$ algorithm.

5.2.6 Evidence for Conjecture 5.2.8

In this section, we give evidence for Conjecture 5.2.8. In particular, we prove it for $k = 1, 2$. Recall that p and r are primes as in Conjecture 5.2.8.

Lemma 5.2.12. *For any positive integer M,*

$$\sum_{k=0}^{M-1} x^k \equiv 0 \pmod{p, \Phi_r(x)} \qquad \Longleftrightarrow \qquad M \equiv 0 \pmod{r}.$$

Proof. Let $m \equiv M \pmod{r}$ with $0 \leq m < r$. Then,

$$\begin{aligned}
& \sum_{k=0}^{M-1} x^k \equiv 0 && \pmod{p, \Phi_r(x)} \\
\iff \quad & \sum_{k=0}^{m-1} x^k \equiv 0 && \pmod{p, \Phi_r(x)} \\
\iff \quad & m = 0 \\
\iff \quad & M \equiv 0 && \pmod{r}
\end{aligned}$$

□

The following result is well known.

Theorem 5.2.13. (Dirichlet's theorem) *Let a, d be two positive coprime integers. Then, there are infinitely many primes congruent to a mod d.*

Lemma 5.2.14. *For any positive integer N with $(N, r) = 1$, there exist infinitely many primes q such that*

$$\Phi_q(x) \equiv \sum_{k=0}^{N-1} x^k \pmod{p, \Phi_r(x)}.$$

Proof. Given $N > 0$ and $(N, r) = 1$, by Theorem 5.2.13 there exists a prime q with $q \equiv N \pmod{r}$. By Lemma 5.2.12,

$$\Phi_q(x) = \sum_{k=0}^{q-1} x^k = x^N \left(\sum_{k=0}^{q-N-1} x^k \right) + \sum_{k=0}^{N-1} x^k \equiv \sum_{k=0}^{N-1} x^k \pmod{p, \Phi_r(x)}.$$

□

Proposition 5.2.15.

$$\Phi_{p_1}(x) \equiv \Phi_{q_1}(x) \pmod{p, \Phi_r(x)} \qquad \iff \qquad p_1 \equiv q_1 \pmod{r},$$

where p_1, q_1 are primes.

Proof. If $p_1 = q_1$, the proposition is trivially true.

Without loss of generality, suppose $p_1 < q_1$. Then,

$$\begin{aligned}
& \Phi_{p_1}(x) \equiv \Phi_{q_1}(x) && \pmod{p, \Phi_r(x)} \\
\iff \quad & \sum_{k=0}^{p_1-1} x^k \equiv \sum_{k=0}^{q_1-1} x^k && \pmod{p, \Phi_r(x)} \\
\iff \quad & \sum_{k=0}^{q_1-p_1-1} x^k \equiv 0 && \pmod{p, \Phi_r(x)} \\
\iff \quad & p_1 \equiv q_1 && \pmod{r},
\end{aligned}$$

by Lemma 5.2.12. □

Proposition 5.2.16. *Let p_1, p_2, q_1, q_2 be primes with p_1, p_2 distinct mod r, and with q_1, q_2 distinct mod r. Moreover, assume that $p_i \not\equiv 1 \pmod{r}$ for $i = 1, 2$. Then,*

$$\Phi_{p_1p_2}(x) \equiv \Phi_{q_1q_2}(x) \pmod{p, \Phi_r(x)} \tag{5.2.4}$$

implies

$$p_1 \equiv q_i \pmod{r} \qquad \textit{and} \qquad p_2 \equiv q_j \pmod{r},$$

where $\{i, j\} = \{1, 2\}$.

Proof. **Case 1:** Suppose that all p_1, p_2, q_1, q_2 are distinct mod r. Then, r is at least 7. For primes $p_0 \neq q_0$, $\Phi_{p_0q_0}(x) = \frac{(x^{p_0q_0}-1)(x-1)}{(x^{p_0}-1)(x^{q_0}-1)}$. Therefore, $\Phi_{p_1p_2}(x) \equiv \Phi_{q_1q_2}(x) \pmod{p, \Phi_r(x)}$ implies

$$\frac{(x^{p_1p_2}-1)(x-1)}{(x^{p_1}-1)(x^{p_2}-1)} \equiv \frac{(x^{q_1q_2}-1)(x-1)}{(x^{q_1}-1)(x^{q_2}-1)} \pmod{p, \Phi_r(x)}$$

Multiply both sides by the denominators:

$$(x^{p_1p_2}-1)(x^{q_1}-1)(x^{q_2}-1) \equiv (x^{q_1q_2}-1)(x^{p_1}-1)(x^{p_2}-1) \pmod{p, \Phi_r(x)}. \tag{5.2.5}$$

If $p_1p_2 \equiv q_1q_2 \pmod{r}$, congruence (5.2.5) becomes

$$\begin{aligned}(x^{q_1}-1)(x^{q_2}-1) &\equiv (x^{p_1}-1)(x^{p_2}-1) \pmod{p, \Phi_r(x)},\\ x^{q_1+q_2}+x^{p_1}+x^{p_2} &\equiv x^{p_1+p_2}+x^{q_1}+x^{q_2} \pmod{p, \Phi_r(x)}.\end{aligned}$$

Note that $p_1+p_2 \not\equiv q_1+q_2 \pmod{r}$. Otherwise, p_1, p_2, q_1, q_2 are distinct roots of $T^2-(p_1+p_2)T+p_1p_2$ in $\mathbb{F}_r$, which is a contradiction. Then, $x^{q_1+q_2}+x^{p_1}+x^{p_2} \pmod{p, x^r-1}$ and $x^{p_1+p_2}+x^{q_1}+x^{q_2} \pmod{p, x^r-1}$ are polynomials with different degrees since the three terms $x^{q_1+q_2}$, x^{p_1}, x^{p_2} are not congruent to any of $x^{p_1+p_2}$, x^{q_1}, x^{q_2}. Therefore, since $\Phi_r(x) = (x^r-1)/(x-1)$,

$$\begin{aligned} & x^{q_1+q_2}+x^{p_1}+x^{p_2} \not\equiv x^{p_1+p_2}+x^{q_1}+x^{q_2} && \pmod{p, x^r-1}\\ \Longrightarrow \quad & \frac{x^{q_1+q_2}-x^{p_1+p_2}}{x-1} \not\equiv \frac{x^{q_1}-x^{p_1}}{x-1}+\frac{x^{q_2}-x^{p_2}}{x-1} && \pmod{p, \Phi_r(x)}\\ \Longrightarrow \quad & x^{q_1+q_2}+x^{p_1}+x^{p_2} \not\equiv x^{p_1+p_2}+x^{q_1}+x^{q_2} && \pmod{p, \Phi_r(x)}\end{aligned}$$

This contradiction implies that $p_1p_2 \not\equiv q_1q_2 \pmod{r}$.

Expanding the terms in congruence (5.2.5), we have

$$\begin{aligned} & x^{p_1p_2+q_1+q_2}-x^{p_1p_2+q_1}-x^{p_1p_2+q_2}-x^{q_1+q_2}+x^{p_1p_2}+x^{q_1}+x^{q_2}-1\\ \equiv \; & x^{q_1q_2+p_1+p_2}-x^{q_1q_2+p_1}-x^{q_1q_2+p_2}-x^{p_1+p_2}+x^{q_1q_2}+x^{p_1}+x^{p_2}-1\\ & \pmod{p, \Phi_r(x)}\end{aligned}$$

$$\begin{aligned}\text{Let } f(x) &= x^{p_1p_2+q_1+q_2}+x^{q_1q_2+p_1}+x^{q_1q_2+p_2}+x^{p_1+p_2}+x^{p_1p_2}+x^{q_1}+x^{q_2},\\ g(x) &= x^{q_1q_2+p_1+p_2}+x^{p_1p_2+q_1}+x^{p_1p_2+q_2}+x^{q_1+q_2}+x^{q_1q_2}+x^{p_1}+x^{p_2}.\end{aligned}$$

As before, we first show that $f(x) \not\equiv g(x) \pmod{p, x^r-1}$. Since $x-1$ divides

$f(x) - g(x)$, we must have $f(x) \not\equiv g(x) \pmod{p, \Phi_r(x)}$. As a result, congruence (5.2.5) leads to a contradiction.

The sum of the coefficients in $f(x) \pmod{x^r - 1}$ is exactly 7. There are only 7 terms in $f(x)$. Since each power of x is congruent mod $x^r - 1$ to a power x^j with $0 \leq j < r$, we see that $f(x)$ is congruent mod $x^r - 1$ to a sum of seven not necessarily distinct such powers x^j. Since $p > r \geq 7$, these cannot cancel each other mod p. A similar result holds for $g(x)$. If $f(x) \equiv g(x) \pmod{p, x^r - 1}$, then x^{p_2} is congruent to some term in $f(x)$. The only possibilities are $x^{p_1 p_2 + q_1 + q_2}$ and $x^{q_1 q_2 + p_1}$. Similarly, x^{p_1} must be congruent to $x^{p_1 p_2 + q_1 + q_2}$ or $x^{q_1 q_2 + p_2}$.

If

$$p_2 \equiv q_1 q_2 + p_1 \pmod{r}, \tag{5.2.6}$$

then $p_1 \not\equiv q_1 q_2 + p_2 \pmod{r}$; otherwise, $p_2 - p_1 \equiv q_1 q_2 \equiv p_1 - p_2 \pmod{r}$, which is impossible. Therefore, $p_1 \equiv p_1 p_2 + q_1 + q_2 \pmod{r}$. Then, using these congruences for p_1 and p_2, we obtain

$$\begin{aligned} & x^{q_1 q_2 + p_2} + x^{p_1 + p_2} + x^{p_1 p_2} + x^{q_1} + x^{q_2} \\ \equiv \;& x^{q_1 q_2 + p_1 + p_2} + x^{p_1 p_2 + q_1} + x^{p_1 p_2 + q_2} + x^{q_1 + q_2} + x^{q_1 q_2} \pmod{p, x^r - 1} \end{aligned}$$

The only possible term in the left-hand side congruent to $x^{q_1 q_2}$ is $x^{p_1 + p_2}$. But congruence (5.2.6) implies $q_1 q_2 \equiv p_2 - p_1 \pmod{r}$. So $q_1 q_2 \not\equiv p_1 + p_2 \pmod{r}$. Hence, $f(x) \not\equiv g(x) \pmod{p, x^r - 1}$.

If $p_2 \equiv p_1 p_2 + q_1 + q_2 \pmod{r}$, then $p_1 \equiv q_1 q_2 + p_2 \pmod{r}$. The case is the same as before by switching the roles of p_1 and p_2.

Case 2: Suppose that some p_i is congruent to some q_j mod r. we may assume that $p_1 \equiv q_1 \equiv m \pmod{r}$ for some $1 < m < r$. Note that $m \neq 1$ by assumption. By Lemma 5.2.15, $\Phi_{p_1}(x) \equiv \Phi_{q_1}(x) \equiv \sum_{k=0}^{m-1} x^k \pmod{p, \Phi_r(x)}$. Therefore,

$$\begin{aligned}
& \Phi_{p_1 p_2}(x) \equiv \Phi_{q_1 q_2}(x) && \pmod{p, x^r - 1} \\
\implies \quad & \Phi_{p_1}(x)\Phi_{p_1 p_2}(x) \equiv \Phi_{q_1}(x)\Phi_{q_1 q_2}(x) && \pmod{p, x^r - 1} \\
\implies \quad & \Phi_{p_1}(x^{p_2}) \equiv \Phi_{q_1}(x^{q_2}) && \pmod{p, x^r - 1} \\
\implies \quad & \sum_{k=1}^{m-1} x^{kp_2} \equiv \sum_{k=1}^{m-1} x^{kq_2} && \pmod{p, x^r - 1}
\end{aligned}$$

Let $M = \{1, 2, \cdots, m-1\}$. We see that $p_2 M = q_2 M$ as subsets of $\mathbb{Z}/r\mathbb{Z}$. Let $a \equiv p_2 q_2^{-1} \pmod{r}$. Then multiplication by $a \pmod{r}$ is a permutation of M. By Lemma 5.2.17 below, $a = 1$. Therefore, $p_2 \equiv q_2 \pmod{r}$. □

Lemma 5.2.17. *Let q be a prime and let $1 < m < q$. Let $M = \{1, 2, \cdots, m-1\}$. Suppose $0 \leq a < q$ and $aM = M$ in $\mathbb{F}_q$. Then, $a = 1$.*

Proof. If $a = 0$, then $aM = \{0\} \neq M$, so we may assume that $a \geq 1$. For any $1 \leq a < q$, multiplication by $a \pmod{q}$ is a permutation of $\{1, 2, \cdots, q-1\}$. If $aM = M$, multiplication by $a \pmod{q}$ is also a permutation of M. As a consequence, multiplication by $a \pmod{q}$ also permutes $\overline{M} \stackrel{\text{def}}{=} \{m, \cdots, q-1\}$. Both M and $\overline{M}$ are not empty since $1 < m < q$.

Suppose $a \neq 1$. Let $q = ua + v$, where the quotient $u = \lfloor q/a \rfloor \geq 1$ and the remainder $v = q - ua < a \leq ua$. This implies $ua > q/2$. We claim that $\{1, 2, \cdots, ua\} \subseteq M$ and $\{q-1, q-2, \cdots, q-ua\} \subseteq \overline{M}$. Then, $|M| + |\overline{M}| > q$, which leads to a contradiction.

We show by induction that $A_k \stackrel{\mathsf{def}}{=} \{1, 2, \cdots, ak\} \subseteq M$ for $1 \leq k \leq u$. Note that A_k is a set of exactly ak elements in $\mathbb{F}_q$ because $ak \leq au < q$. Since $1 \in M$, we have $a \cdot 1 \in M$. Therefore, $1 \leq a \leq m - 1$, so $A_1 \subseteq M$. Assume $A_{k-1} \subseteq M$ for $k > 1$. We have $k \in A_{k-1}$ because $k \leq 2(k-1) \leq a(k-1)$. Then, $ak \in aM = M$, which implies $A_k \subseteq M$.

The statement $\{q - 1, q - 2, \cdots, q - ak\} \subseteq \overline{M}$ can be shown by a similar argument, beginning with $q - 1 \in \overline{M}$. $\square$

Bibliography

[1] William Adams and Daniel Shanks. Strong primality tests that are not sufficient. *Math. Comp.*, 39(159):255–300, 1982.

[2] Leonard M. Adleman, Kenneth L. Manders, and Gary L. Miller. On taking roots in finite fields. In *FOCS*, pages 175–178. IEEE, 1977.

[3] Leonard M. Adleman, Carl Pomerance, and Robert S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. of Math.*, 117(1):173–206, jan 1983.

[4] Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, 2003.

[5] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[6] W. R. Alford, Andrew Granville, and Carl Pomerance. There are infinitely many Carmichael numbers. *Ann. of Math.*, 139:703–722, 1994.

[7] Nesmith C. Ankeny. The least quadratic non residue. *Ann. of Math.*, 55(1):65–72, jan 1952.

[8] Michael Artin. *Algebra*. Prentic Hall, 1991.

[9] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61(203):29–68, 1993.

[10] Eric Bach. A note on square roots in finite fields. *IEEE Transactions on Information Theory*, 36(6):1494–1498, nov 1990.

[11] Eric Bach and Klaus Huber. Note on taking square-roots modulo N. *IEEE Transactions on Information Theory*, 45(2):807–809, mar 1999.

[12] Paulo S. Barreto and José Felipe Voloch. Efficient computation of roots in finite fields. *Des. Codes Cryptography*, 39(2):275–280, 2006.

[13] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.

[14] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713–735, 1970.

[15] Daniel J. Bernstein. Faster square roots in annoying finite fields. Preprint.

[16] Pedro Berrizbeitia. Sharpening "PRIMES is in P" for a large family of numbers. *Math. Comp.*, 74(252):2043–2059 (electronic), 2005.

[17] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32:586–615, 2003.

[18] Johannes Buchmann and Victor Shoup. Constructing nonresidues in finite fields and the extended riemann hypothesis. *Math. Comp.*, 65(215):1311–1326, jul 1996.

[19] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36(154):587–592, 1981.

[20] Michele Cipolla. Un metodo per la risoluzione della congruenza di secondo grado. *Rendiconto dell'Accademia delle Scienze Fisiche e Matematiche Napoli*, 9:154–163, 1903.

[21] David A. Clark and M. Ram Murty. The Euclidean algorithm for Galois extensions of $\mathbb{Q}$. *J. Reine Angew. Math.*, 459:151–162, 1995.

[22] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, 1993.

[23] Henri Cohen and Hendrik W. Lenstra Jr. Primality testing and Jacobi sums. *Math. Comp.*, 42(165):297–330, 1984.

[24] Stephen A. Cook. *On the minimum computation time of functions*. PhD thesis, Harvard University, 1966.

[25] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[26] Martin Dietzfelbinger. *Primality Testing in Polynomial Time: From Randomized Algorithms to "PRIMES Is in P"*. Springer, 2004.

[27] Etienne Fouvry. Théorème de Brun-Titchmarsh; application au théorème de Fermat. *Invent. Math.*, 79(2):383–407, 1985.

[28] Martin Fürer. Faster integer multiplication. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 2007. ACM Press.

[29] Joachim Von Zur Gathen and Daniel Panario. Factoring polynomials over finite fields: a survey. *J. Symb. Comput.*, 31(1-2):3–17, 2001.

[30] Shafi Goldwasser and Joe Kilian. Primality testing using elliptic curves. *J. ACM*, 46(4):450–472, 1999.

[31] Andrew Granville. It is easy to determine whether a given integer is prime. *Bull. Amer. Math. Soc.*, 42:3–38, 2005.

[32] Gareth A. Jones and Josephine M. Jones. *Elementary Number Theory*. Springer-Verlag, 1998.

[33] Hendrik W. Lenstra Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126:649–673, 1987.

[34] Hendrik W. Lenstra Jr. and Carl Pomerance. Primality testing with gaussian periods, 2005. Preliminary version. Available at `http://www.math.dartmouth.edu/~carlp/PDF/complexity12.pdf`.

[35] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, 1st (2nd printing) edition, 1971.

[36] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, Reading, 3rd edition, 1997.

[37] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987.

[38] Neal Koblitz. *A Course in Number Theory and Cryptography*. Graduate Texts in Mathematics. Springer-Verlag, 1994.

[39] Derrick H. Lehmer. Tests for primality by the converse of Fermat's theorem. *Bull. AMS*, 33:327–340, 1927.

[40] Derrick H. Lehmer. Computer technology applied to the theory of numbers. In *Studies in Number Theory*, pages 117–151. Math. Assoc. Amer. (distributed by Prentice-Hall, Englewood Cliffs, N.J.), 1969.

[41] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515–534, 1982.

[42] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, 1994.

[43] Scott Lindhurst. An analysis of Shanks's algorithm for computing square roots in finite fields. *CRM Proceedings and Lecture Notes*, 19:231–242, 1999.

[44] Gary L. Miller. Riemann's hypothesis and tests for primality. In *STOC '75: Proceedings of seventh annual ACM symposium on Theory of computing*, pages 234–239, New York, NY, USA, 1975. ACM Press.

[45] Siguna Müller. On probable prime testing and the computation of square roots mod n. In Wieb Bosma, editor, *Algorithmic number theory: ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 423–437, Berlin, 2000. Springer-Verlag.

[46] Siguna Müller. On the computation of square roots in finite fields. *Des. Codes Cryptography*, 31(3):301–312, 2004.

[47] P. Pépin. Sur la formule $2^{2^n} + 1$. *Comptes Rendus Acad. Sci. Paris*, 85:329–333, 1877.

[48] Henry C. Pocklington. The determination of the prime or composite nature of large numbers by Fermat's theorem. *Proc. of the Cambridge Philosophical Soc.*, 18:29–30, 1914.

[49] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.

[50] Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

[51] René C. Peralta. A simple and fast probabilistic algorithm for computing square roots modulo a prime number. *IEEE Transactions on Information Theory*, 32(6):846–847, nov 1986.

[52] Bernhard Riemann. Ueber die Anzahl der Primzahlen unter einer gegebenen Grösse. *Monatsberichte der Berliner Akademie*, pages 671–680, nov 1859.

[53] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[54] Arnold Schönhage and Volker Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[55] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, apr 1985.

[56] Daniel Shanks. Five number-theoretic algorithms. In *Proc. 2nd Manitoba Conf. Numer. Math.*, volume VII of *Congressus Numerantium*, pages 51–70, Winnipeg, Manitoba, 1972. Utilitas Mathematica.

[57] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[58] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer, 1994.

[59] Robert Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977.

[60] Tsz-Wo Sze. On solving univariate polynomial equations over finite fields and some related problems, 2007. Preliminary version.

[61] Tsz-Wo Sze. On taking square roots and constructing quadratic nonresidues over finite fields, 2007. Preliminary version.

[62] Tsz-Wo Sze. A potentially fast primality test, 2007. Preliminary version.

[63] Alberto Tonelli. Bemerkung über die Auflösung quadratischer Congruenzen. *Nachrichten der Akademie der Wissenschaften in Göttingen*, pages 344–346, 1891.

[64] Andrei L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[65] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, 2nd edition, 2005.

[66] Stephen M. Turner. Square roots mod p. *The American Mathematical Monthly*, 101(5):443–449, may 1994.

[67] Christiaan van de Woestijne. *Deterministic equation solving over finite fields*. PhD thesis, Universiteit Leiden, Leiden, Netherlands, 2006.

[68] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, United Kingdom, 2nd edition, 2003.

[69] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2003.

[70] Hugh C. Williams. *Édouard Lucas and Primality Testing*, volume 22 of *Canadian Mathematical Society Series of Monographs and Advanced Texts*. Wiley-Interscience, 1998.

[71] David Y.Y. Yun. On square-free decomposition algorithms. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 26–35, New York, NY, USA, 1976. ACM Press.