On Taking Square Roots and Constructing Quadratic Nonresidues over Finite Fields

Tsz-Wo Sze (szetszwo@cs.umd.edu)

Preliminary version, October 28, 2007

Abstract

We present a novel idea to compute square roots over some families of finite fields. Our algorithms are deterministic polynomial time and can be proved by elementary means (without assuming any unproven hypothesis). In some particular finite fields $\mathbb{F}_q$, there are algorithms for taking square roots with $\tilde{O}(\log^2 q)$ bit operations.

As an application of our square root algorithms, we show a deterministic primality testing algorithm for some form of numbers. For some positive integer N, this primality testing algorithm runs in $\tilde{O}(\log^3 N)$.

1 Introduction

Let $\mathbb{F}_q$ be a finite field with q elements. Suppose q is odd in this paper. 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 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 [24], [23] and [1] 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 [24, 23], Adleman-Manders-Miller [1] and Cipolla-Lehmer [10, 14] require a quadratic nonresidue as an input. Berlekamp-Rabin [7, 19] takes square roots by polynomial factoring over finite fields. The idea of Peralta [20] is similar to Berlekamp-Rabin. For other results, see [4], [5], [6], [8], [9], [15], [17] [18] and [25].

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

By assuming the Extended Riemann Hypothesis, Ankeny [3] 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 $\mathbb{F}_p$. It can be proved that the least quadratic nonresidue must be a prime. 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 [22] 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

$$\zeta_5 = \frac{a + \sqrt{a^2 - 4}}{2}, \qquad \text{where } a = \frac{-1 + \sqrt{5}}{2}. \tag{1.1}$$

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 and Section 3, respectively. In Section 4, we construct a special group and discuss the computation of the operations in that group. In section 5, we provide algorithms for taking square roots and finding quadratic nonresidues in finite fields. A deterministic primality testing algorithm for some families of numbers is shown in section 6. In the appendix (by L. Washington), we show how to construct roots of unity needed for Theorem 2.2.

2 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. We also show a deterministic primality testing algorithm, for some particular form of numbers, constructed by our square root algorithms. For some positive integer N, such primality testing algorithm runs in $\tilde{O}(\log^3 N)$.

Theorem 2.1. *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 2.2. *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.*

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

Theorem 2.3. *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.*

3 Main Idea

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 can be computed efficiently 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.

4 A Group Isomorphic to $\mathbb{F}_q^\times$

Let $\mathbb{F}_q$ be a finite field with q odd and characteristic p. 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 $a_1 + a_2 \neq 0$,

$$\begin{aligned} [a] * [\infty] &= [\infty] * [a] = [a], && (4.1)\\ [a_1] * [-a_1] &= [\infty], && (4.2)\\ [a_1] * [a_2] &= \left[\frac{a_1 a_2 + \alpha^2}{a_1 + a_2}\right]. && (4.3) \end{aligned}$$

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

Proof. Define a bijective mapping

$$\psi : G_\alpha \longrightarrow \mathbb{F}_q^\times, \qquad [\infty] \longmapsto 1, \qquad [a] \longmapsto \frac{a+\alpha}{a-\alpha} \tag{4.4}$$

with inverse

$$\psi^{-1} : \mathbb{F}_q^\times \longrightarrow G_\alpha, \qquad 1 \longmapsto [\infty], \qquad b \longmapsto \left[\frac{\alpha(b+1)}{b-1}\right]. \tag{4.5}$$

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.

4.1 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 4.2. *Let $[a] \in G'_\alpha$ and $k > 0$. If the order of $[a]$ does not divide k, then*

$$[a]^k = \left[\alpha \cdot \frac{(a+\alpha)^k + (a-\alpha)^k}{(a+\alpha)^k - (a-\alpha)^k}\right] = \left[\frac{a^k + \binom{k}{2} a^{k-2}\alpha^2 + \cdots}{k a^{k-1} + \binom{k}{3} a^{k-3}\alpha^2 + \cdots}\right]. \tag{4.6}$$

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 (4.6) can be obtained by expanding $(a \pm \alpha)^k$. □

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

$$\gamma_k(x) = \frac{(x+\alpha)^k + (x-\alpha)^k}{2} = \sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{2j} x^{k-2j}\alpha^{2j}, \tag{4.7}$$

$$\Psi_k(x) = \frac{(x+\alpha)^k - (x-\alpha)^k}{2\alpha} = \sum_{j=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k}{2j+1} x^{k-1-2j}\alpha^{2j}. \tag{4.8}$$

Note that $\gamma_k(x), \Psi_k(x) \in \mathbb{F}_p[\alpha^2][x]$. We have the following recursion equations for $\gamma_k(x)$ and $\Psi_k(x)$.

Lemma 4.3. *For $k > 0$,*

$$\gamma_{k+1}(x) = x\gamma_k(x) + \alpha^2\Psi_k(x), \tag{4.9}$$

$$\Psi_{k+1}(x) = \gamma_k(x) + x\Psi_k(x). \tag{4.10}$$

Proof. By equations (4.7) and (4.8),

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

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 4.4. As a consequence, only $O(\log n)$ polynomial multiplications are required for computing γ_n and Ψ_n.

Lemma 4.4. *For $k > 0$,*

$$\gamma_{2k}(x) = \gamma_k(x)^2 + \alpha^2\Psi_k(x)^2, \tag{4.11}$$

$$\Psi_{2k}(x) = 2\gamma_k(x)\Psi_k(x), \tag{4.12}$$

$$\gamma_{2k+1}(x) = \gamma_k(x)\gamma_{k+1}(x) + \alpha^2\Psi_k(x)\Psi_{k+1}(x), \tag{4.13}$$

$$\Psi_{2k+1}(x) = \gamma_k(x)\Psi_{k+1}(x) + \gamma_{k+1}(x)\Psi_k(x). \tag{4.14}$$

Proof. By Lemma 4.3,

$$\begin{pmatrix} \gamma_k & \alpha^2\Psi_k \\ \Psi_k & \gamma_k \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix} \begin{pmatrix} \gamma_{k-1} & \alpha^2\Psi_{k-1} \\ \Psi_{k-1} & \gamma_{k-1} \end{pmatrix} = \begin{pmatrix} x & \alpha^2 \\ 1 & x \end{pmatrix}^{k-j} \begin{pmatrix} \gamma_j & \alpha^2\Psi_j \\ \Psi_j & \gamma_j \end{pmatrix} \tag{4.15}$$

for any $k > 0$ and $1 \le j \le k$. 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 (4.15),

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

which implies

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

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

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

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 4.5. *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,

$$\begin{aligned} & [a]^d = [\infty] & \\ \iff \quad & [a]^{d-1} = [-a] & \text{by equation (4.2)} \\ \iff \quad & \frac{\gamma_{d-1}(a)}{\Psi_{d-1}(a)} = -a & \text{by Lemma 4.2} \\ \iff \quad & \Psi_d(a) = 0 & \text{by equation (4.10).} \end{aligned}$$

Note that if $\Psi_d(a) = 0$, we have $\Psi_{d-1}(a) \neq 0$. Otherwise, if $\Psi_{d-1}(a) = 0$, equation (4.10) 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. □

4.2 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 [27] 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.

5 Taking Square Roots and Finding Quadratic Nonresidues

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

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

Consider the abelian group G_α defined in 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 $\zeta_m \in \mathbb{F}_q$. We have the following proposition.

Proposition 5.1. *Let $[0] \neq [a] \in G'_\alpha$. Suppose $[a]^d = [\infty]$ for some $d > 0$. Then,*

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

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

Lemma 5.2. *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 (4.2) and (4.3),

$$[g_1] * [g_2] = \begin{cases} [\infty] & \text{, if } g_1 = -g_2, \\ \left[\frac{g_1 g_2 + \beta}{g_1 + g_2}\right] & \text{, otherwise.} \end{cases} \tag{5.1}$$

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 (5.1). Then, $[g]^k$ can be evaluated efficiently by the successive squaring method.

Another method for computing $[g]^k$ is due to Proposition 4.5 and equation (4.6). 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 4.3 and 4.4. 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 [11], [21], [13] and [26]). 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. Let $\mathsf{power}(g, k, \beta)$ be a procedure computing $[g]^k$.

For our algorithms, we need to enumerate the elements of the field. Let $\{T_0, T_1, \cdots, T_{n-1}\}$ be a *basis* of $\mathbb{F}_q$ over $\mathbb{F}_p$. For $1 \le m \le \frac{q-1}{2}$, choose k such that $\frac{p^k-1}{2} < m \le \frac{p^{k+1}-1}{2}$. 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 $0 \le a_k < \frac{p-1}{2}$ and $0 \le a_j < p$ for $0 \le j < k$. The mth element of $\mathbb{F}_q$ in our enumeration is defined to be

$$a_0 T_0 + a_1 T_1 + \cdots + a_{k-1} T_{k-1} + (a_k + 1) T_k, \tag{5.2}$$

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

$$\mathsf{element}(m) \ne -\mathsf{element}(j) \qquad \text{for } 1 \le j, m \le \frac{q-1}{2}. \tag{5.3}$$

In the following sections, we present deterministic polynomial time algorithms to find square roots for some families of finite fields.

5.1 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 $e \ge 2$ and $f \ge 1$ 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 following algorithm for computing square roots in $\mathbb{F}_q$.

Algorithm 5.3. $\mathsf{squareRoot}(\beta)$

```
{
    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 power(g, (q-1)/3^k, β) = [∞]
    compute [a] = power(g, (q-1)/3^{k+1}, β)
    return ±a√-3
}
```

Lemma 5.4. *Algorithm 5.3 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 5.1, $\alpha = \pm\frac{a(\zeta_4-1)}{\zeta_4+1} = \pm a\sqrt{-1}$. 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 5.1, $\alpha = \pm\frac{a(\zeta_3-1)}{\zeta_3+1} = \pm a\sqrt{-3}$, 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 element() procedure (see equation (5.3)). 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 5.3 is $\tilde{O}(t \log^2 q + \log^9 p)$.

Proof of Theorem 2.1. Since $t = O(\text{poly}(\log q))$, square roots in $\mathbb{F}_q$ can be computed by Algorithm 5.3 with running time $\tilde{O}(\text{poly}(\log q)\log^2 q + \log^9 p)$. 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. □

5.2 Other Cases

Algorithm 5.3 in the previous section can be generalized as below.

Lemma 5.5. *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 $M + t = O(\text{poly}(\log q))$ 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 $M + t = O(\text{poly}(\log q))$ 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 5.3 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 5.6 below (see also Proposition 5.1). 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 5.6. $\mathsf{matchZeta}(r, \zeta_r, g, \beta)$

{
  find the largest k such that $\mathsf{power}(g, \frac{q-1}{r^k}, \beta) = [\infty]$
  compute $[a] = \mathsf{power}(g, \frac{q-1}{r^{k+1}}, \beta)$
  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 (1.1)) 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) and all square roots in the formula of ζ_r are in $\mathbb{F}_p$. Then, ζ_r can be computed in polynomial time. For any $q = p^n = r^{e_r}t + 1$ with $t = O(\text{poly}(\log q))$, 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 (1.1) 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.

5.3 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 5.7 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 shows a method to compute ζ_r in deterministic polynomial time. Therefore, the r-part of $\mathbb{F}_q^\times$ can be handled.

Lemma 5.7. *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 2.2. Since $p \equiv 13, 25 \pmod{36}$, cubic roots in $\mathbb{F}_p$ can be computed in polynomial time by Lemma 5.7. 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 $r_j = O(\mathrm{poly}(\log q))$, ζ_{r_j} can be computed in polynomial time (see the Appendix) for $j = 1, 2, \cdots, m$. Finally, Lemma 5.5 implies the theorem. □

5.4 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 (5.2)). 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 5.8 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 5.8. $\mathsf{findZeta}(r)$

{

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 5.3 can be constructed for computing the square roots. We have a for-loop, which is similar to the for-loop in Algorithm 5.3, in the algorithm. The running time of the for-loop is $\tilde{O}(t \log^2 q)$. Algorithm 5.6 (or $\mathsf{matchZeta4}()$ in Algorithm 5.3 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 5.8. 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 2.3. Since $t = O(\mathrm{poly}(\log q))$, ζ_r (or ζ_4 if $r = 2$) can be computed in polynomial time by Algorithm 5.8. Together with $r = O(\mathrm{poly}(\log q))$, Lemma 5.5 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 $0 \le k < p$, let

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

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 $0 \le m_0 \le k$. 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 5.9 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 $0 \le m_0 \le \lfloor \log^2 p \rfloor$ 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 5.9. *We have the following lower bound for the central binomial coefficient*

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

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 $k > 2$. □

6 Primality Testing

Let $N = 2^e t + 1$ for some odd t with $2^e > t = \mathrm{poly}(\log N)$. Try to compute $(-1)^{1/2}$, $(-1)^{1/4}$, $\cdots$, $(-1)^{1/2^{e-1}}$ by the square root algorithm described in Section 5.4. If $(-1)^{1/2^{e-1}}$ is obtained, then N is a prime by Proth's Theorem (Theorem 6.1 below). Otherwise, since the square root algorithm is deterministic, it must fail in some point and then we conclude that N is composite. Such primality testing algorithm is deterministic and runs in $\tilde{O}(t \log^3 N)$.

If t is a small constant (e.g. 15,45,75), the algorithm above runs in $\tilde{O}(\log^3 N)$. For this kind of numbers, our algorithm is faster than other deterministic tests. The running time of the AKS test [2] and Lenstra-Pomerance's modified AKS test [12] are $\tilde{O}(\log^{7.5} N)$ and $\tilde{O}(\log^6 N)$, respectively. Assuming the Extended Riemann Hypothesis, Miller's test [16] is deterministic with running time $\tilde{O}(\log^4 N)$.

Theorem 6.1. (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 [28] for the details of Proth's Theorem.

References

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

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

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

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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`.

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

[14] 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.

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

[16] 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.

[17] 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.

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

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

[20] 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.

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

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

[23] 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.

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

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

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

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

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

Appendix: Computing roots of unity (by L. Washington)

Let $q = 2 \cdot 3^n + 1$ be prime. We show how to construct a qth root of unity mod p (where p is some prime) in polynomial time in $\log p$.

There are several such primes. The values of $n \leq 6000$ are 1, 2, 4, 5, 6, 9, 16, 17, 30, 54, 57, 60, 65, 132, 180, 320, 696, 782, 822, 897, 1252, 1454, 4217, 5480 corresponding to the primes $q = 7$, 19, 163, ... It is reasonable to conjecture that there are infinitely many such q (this is similar to the conjecture that there are infinitely many Mersenne primes).

Let ζ_q be a primitive qth root of unity and let ρ be a primitive cube root of unity. Let G be the Galois group of $\mathbb{Q}(\zeta_q, \rho)/\mathbb{Q}(\rho, \sqrt{-q})$. Then G is cyclic of order $(q-1)/2 = 3^n$. Let σ be a generator and let

$$\sigma_k = \sigma^{3^{n-k}}.$$

Then σ_k generates a subgroup of G of order 3^k. The fixed field K_k of σ_k is of degree 3^{n-k} over $\mathbb{Q}(\rho, \sqrt{-q})$.

We want to obtain an expression for a qth root of unity that involves only $\sqrt{-q}$ and taking cube roots. The basic idea is the following. Suppose we want to compute $r \in K_m$. Let $r_1 = r$, $r_2 = \sigma_{m+1}(r)$, $r_3 = \sigma_{m+1}^2(r)$ be the Galois conjugates of r over K_{m+1}. Let

$$f = r_1 + r_2 + r_3, \quad g = r_1 + \rho r_2 + \rho^2 r_3, \quad h = r_1 + \rho^2 r_2 + \rho r_3.$$

Then $\sigma_{m+1}(g) = \rho^2 g$, so g^3 is fixed by σ_{m+1} and therefore lies in K_{m+1}. Similarly, $f^3, h^3 \in K_{m+1}$. If we can determine the values of f^3, g^3, h^3, and if we can compute their cube roots, then we know f, g, h up to cube roots of unity. So, let's assume that we know f, g, h. Then $r_1 = (f + g + h)/3$, $r_2 = (f + \rho^2 g + \rho h)/3$, $r_3 = (f + \rho g + \rho^2 h)/3$, so we recover r_1, r_2, r_3.

Start with $r = \zeta_q$. We will actually use the procedure for r and its Galois conjugates $\sigma^3(r)$, $\sigma^6(r)$, $\sigma^9(r)$, ... The above reduces the computation of ζ_q and its Galois conjugates to finding the cube roots of certain elements of K_1. In fact, these elements of K_1 are f^3, g^3, h^3 and their Galois conjugates over $K_n = \mathbb{Q}(\sqrt{-q}, \rho)$. We then reduce the computation of these elements to finding the cube roots of certain elements of K_2 and their conjugates. Continuing in this manner, we eventually reduce the problem to computing cube roots of elements of K_n. Note that each time that we formed a sum g, we also formed a sum h. These are conjugate via the automorphism that sends ρ to ρ^2 and fixes ζ_q. Therefore, the elements of K_n that we obtain are in pairs z_1, z_2 that are conjugate over $\mathbb{Q}(\sqrt{-q})$. Both $z_1 + z_2$ and $(z_1 - z_2)/\sqrt{-3}$ are fixed by $\mathrm{Gal}(K_n/\mathbb{Q}(\sqrt{-q}))$, so they lie in $\mathbb{Q}(\sqrt{-q})$. The real and imaginary parts are rational numbers, and it is easy to bound the denominators. Therefore, we can recognize these as rational numbers by floating point computations. Working back through the preceding and taking the necessary cube roots, we obtain an expression for ζ_q.

The expression obtained for ζ_q can be reduced mod p. There will be some ambiguity caused by the cube roots being determined only up to powers of ρ, so we obtain a finite list of possibilities of ζ_q. Taking their qth powers identifies a primitive qth root of unity.

The above is best understood via an example. Let $q = 19$. The Galois group G is generated by σ, which maps ζ_{19} to ζ_{19}^4. Also, $\sigma_1 = \sigma^3$ maps ζ_{19} to ζ_{19}^7. Form

$$f_0 = \zeta_{19} + \zeta_{19}^7 + \zeta_{19}^{49}.$$

The Galois conjugates are $f_0, \sigma(f_0), \sigma^2(f_0)$.

It is classical, and easily verified numerically, that

$$f_0 + \sigma(f_0) + \sigma^2(f_0) = \frac{-1 + \sqrt{-19}}{2}.$$

Define

$$\begin{aligned} x_0 &= (f_0 + \rho\sigma(f_0) + \rho^2\sigma^2(f_0))^3 \\ x_1 &= (f_0 + \rho^2\sigma(f_0) + \rho\sigma^2(f_0))^3. \end{aligned}$$

Then σ fixes x_0 and x_1, so they lie in $\mathbb{Q}(\sqrt{-19}, \rho)$. Moreover, the map that switches ρ and ρ^2 and fixes ζ_{19} switches x_0 and x_1. Therefore, $x_0 + x_1$ and $(x_0 - x_1)/\sqrt{-3}$ are in $\mathbb{Q}(\sqrt{-19})$. Numerical

computation shows that

$$\begin{aligned} x_0 + x_1 &= \frac{1}{2}(19 - 17\sqrt{-19}) \\ \frac{x_0 - x_1}{\sqrt{-3}} &= \frac{1}{2}(-57 - 9\sqrt{-19}). \end{aligned}$$

(Note that these numbers are algebraic integers, so rounding the results of a floating point computation yields correct exact answers.) Therefore,

$$\begin{aligned} x_0 &= \frac{1}{4}(19 - 17\sqrt{-19} - 57\sqrt{-3} + 9\sqrt{57}) \\ x_1 &= \frac{1}{4}(19 - 17\sqrt{-19} + 57\sqrt{-3} - 9\sqrt{57}), \end{aligned}$$

where $\sqrt{57} = -\sqrt{-3}\sqrt{-19}$.

Therefore, since $1 + \rho + \rho^2 = 0$, we obtain

$$f_0 = \frac{1}{3}(\frac{-1 + \sqrt{-19}}{2} + x_0^{1/3} + x_1^{1/3}),$$

with an appropriate choice of cube roots of x_0 and x_1.

Define

$$\begin{aligned} f_1 &= (\zeta_{19} + \rho\sigma_1(\zeta_{19}) + \rho^2\sigma_1^2(\zeta_{19}))^3 \\ f_2 &= (\zeta_{19} + \rho^2\sigma_1(\zeta_{19}) + \rho\sigma_1^2(\zeta_{19}))^3. \end{aligned}$$

Then f_1 and f_2 are fixed by σ_1, hence lie in K_1. Let

$$\begin{aligned} y_1 &= f_1 + \sigma(f_1) + \sigma^2(f_1) \\ y_2 &= f_2 + \sigma(f_2) + \sigma^2(f_2). \end{aligned}$$

Then y_1 and y_2 lie in $\mathbb{Q}(\sqrt{-19}, \rho)$. Numerical computation yields

$$\begin{aligned} y_1 + y_2 &= 38 - \sqrt{-19} \\ \frac{y_1 - y_2}{\sqrt{-3}} &= 3\sqrt{-19}, \end{aligned}$$

hence

$$\begin{aligned} y_1 &= \frac{1}{2}(38 - \sqrt{-19} - 3\sqrt{57}) \\ y_2 &= \frac{1}{2}(38 - \sqrt{-19} + 3\sqrt{57}). \end{aligned}$$

Let

$$\begin{aligned} x_2 &= (f_1 + \rho\sigma(f_1) + \rho^2\sigma^2(f_1))^3 \\ x_3 &= (f_1 + \rho^2\sigma(f_1) + \rho\sigma^2(f_1))^3 \\ x_4 &= (f_2 + \rho\sigma(f_2) + \rho^2\sigma^2(f_2))^3 \\ x_5 &= (f_2 + \rho^2\sigma(f_2) + \rho\sigma^2(f_2))^3. \end{aligned}$$

Then

$$\begin{aligned} x_2 + x_5 &= \frac{1}{2}(-1007 + 4373\sqrt{-19}) \\ \frac{x_2 - x_5}{\sqrt{-3}} &= \frac{1}{2}(-10659 - 99\sqrt{-19}) \\ x_3 + x_4 &= 1292 - 1121\sqrt{-19} \\ \frac{x_3 - x_4}{\sqrt{-3}} &= 2850 + 171\sqrt{-19}. \end{aligned}$$

Solving yields

$$\begin{aligned} x_2 &= \frac{1}{4}(-1007 + 4373\sqrt{-19} - 10659\sqrt{-3} + 99\sqrt{57}) \\ x_3 &= \frac{1}{2}(1292 - 1121\sqrt{-19} + 2850\sqrt{-3} - 171\sqrt{57}) \\ x_4 &= \frac{1}{2}(1292 - 1121\sqrt{-19} - 2850\sqrt{-3} + 171\sqrt{57}) \\ x_5 &= \frac{1}{4}(-1007 + 4373\sqrt{-19} + 10659\sqrt{-3} - 99\sqrt{57}). \end{aligned}$$

Again, since $1 + \rho + \rho^2 = 0$, we have

$$\begin{aligned} f_1 &= \frac{1}{3}(x_2^{1/3} + x_3^{1/3} + y_1) \\ f_2 &= \frac{1}{3}(x_4^{1/3} + x_5^{1/3} + y_2), \end{aligned}$$

with an appropriate choice of cube roots. The search for the appropriate cube roots can be shortened, for example, by using the fact that $x_2^{1/3}x_5^{1/3}$ is fixed by σ and is unchanged under the automorphism that maps ρ to ρ^2 and which fixes ζ_{19}. It therefore lies in $\mathbb{Q}(\sqrt{-19})$. Numerical computations show that $x_2^{1/3}x_5^{1/3} = -114 - 4\sqrt{-19}$. Therefore, the choice of cube root for one of $x_2^{1/3}$ and $x_5^{1/3}$ determines the other.

Putting all of the above together, we obtain

$$\zeta_{19} = \frac{1}{3}(f_0 + f_1^{1/3} + f_2^{1/3})$$

with an appropriate choice of square roots.

Schoof's algorithm allows us to calculate $\sqrt{-3}$ and $\sqrt{-19}$ in time polynomial in $\log p$. If taking cube roots mod p is easy (for example, if $p \equiv 4, 7 \pmod 9$), then the above quickly calculates several possibilities for ζ_{19}, corresponding to the choices of cube roots. Each possibility can be tested to determine whether of not it is a primitive 19th root of unity. This will yield the desired ζ_{19} in time polynomial in $\log p$.