

On Solving Univariate Polynomial Equations over Finite Fields and Some Related Problems

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

Preliminary version, October 28, 2007

Abstract

We show deterministic polynomial time algorithms over some family of finite fields for solving univariate polynomial equations and some related problems such as taking n th roots, constructing n th nonresidues, constructing primitive elements and computing elliptic curve “ n th roots”. In addition, we present a deterministic polynomial time primality test for some family of integers. All algorithms can be proved by elementary means (without assuming any unproven hypothesis).

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

- constructing primitive n th roots of unity,
- taking n th roots,
- constructing n th 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 [21, 19] and its generalization for taking n th roots, constructing n th nonresidues and taking n th roots are polynomial time equivalent for all n . It is clear that primitive n th roots of unity can be computed efficiently from any n th nonresidue when n is prime. It is obvious that a primitive element is also an n th nonresidue. In [20], we showed that, for some families of finite fields, once we can compute a primitive n th 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 [15], for factoring polynomials over rational numbers. For polynomial factoring over finite fields, we have Berlekamp’s algorithm [4], which is deterministic but exponential time. So it only works well in small finite fields. There is a probabilistic version of Berlekamp’s algorithm [5] for large finite fields. We also have Cantor-Zassenhaus [7], which is a probabilistic algorithm, for polynomial factoring over finite fields. For a survey of polynomial factoring, see [10].

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 factorization¹ (see [26] and

¹Suppose 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.

[13]) and distinct degree factorization (see [10]) 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 [21, 19], Adleman-Manders-Miller [1] and Cipolla-Lehmer [8, 14] are polynomial time square root algorithms, which require a quadratic nonresidue as an input. Shanks' algorithm can be generalized to take n th root with a given n th nonresidue. Schoof's algorithm [18], 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. Some deterministic polynomial time algorithms for constructing primitive r th roots of unity and taking square roots over some families of finite fields are shown in [20].

All running times are given in term of bit operations. We ignore logarithmic factors in running time. Polynomial multiplication, division with remainder, gcd over finite fields can be implemented by fast Fourier transform (FFT) and other fast methods with running time $\tilde{O}(d \log q)$, where q is the number of element in the field and d is an upper bound of the degrees. See [9], [17], [13] and [23].

The main results are presented in Section 1. The problems of taking r th root, testing of primality, solving polynomial equations and computing elliptic curve " n th roots" are discussed in Section 2, Section 3, Section 4 and Section 5, respectively.

1 Main Results

In this paper, we show deterministic polynomial time algorithms for taking n th roots, constructing n th nonresidues, constructing primitive elements, solving polynomial equations and computing elliptic curve " n th roots" over some family of finite fields. In additional, we present a deterministic polynomial time primality test for some family of integers. The main results are summarized by the theorems at the end of the section. All theorems can be proved by elementary means.

Definition 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, \dots, 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) $r_1 + \cdots + r_m + t = O(\text{poly}(\log q))$.

(v) a primitive r_j th 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 ℓ th of unity can be computed for all prime factors ℓ of $q - 1$ except for $\ell|t$. For deterministic polynomial time algorithms constructing primitive r th root of unity over finite fields, see [20].

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

$$\overline{\mathcal{F}} \stackrel{\text{def}}{=} \bigcup_{t \geq 1} \mathcal{F}_t. \tag{1.1}$$

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. We summarize the main results below.

Theorem 1.2. Let $\mathbb{F}_q \in \overline{\mathcal{F}}$. For $r \in \{r_1, \dots, r_m\}$, there is a deterministic polynomial time algorithm computing an r th root of any r th residue in \mathbb{F}_q . Equivalently, there is a deterministic polynomial time algorithm constructing an r th nonresidue in \mathbb{F}_q .

Theorem 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 .

Theorem 1.4. 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 $\tilde{O}(\log^3 N)$.

Theorem 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 .

Theorem 1.6. Let $\mathbb{F}_q \in \mathcal{F}_1$. There is a deterministic polynomial time algorithm computing elliptic curve “ n th roots” over \mathbb{F}_q .

2 Taking r th Roots

In this chapter, we extend the square root algorithms in [20] and show deterministic polynomial time algorithms for taking r th roots in some finite fields. Like the relationship between taking square roots and constructing quadratic nonresidues, the problem of constructing r th nonresidues, r a small positive integer and r not the characteristic of the field, is polynomial time reducible to the problem of taking r th roots, and vice versa. Clearly, if we can take r th roots, we can first pick a non-zero, non-identity element in a finite field, then keep taking r th roots and finally obtain an r th nonresidue. For the converse, Shanks’ square root algorithm [19] can be generalized to take r th root with a given r th nonresidue.

In [3], Barreto and Voloch showed deterministic polynomial time algorithms for taking r th 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 [6] provided a deterministic algorithm for constructing k th power nonresidues over finite fields. Their algorithm is polynomial time under the assumption of Extended Riemann Hypothesis. For other related results, see [20].

Let $\mathbb{F}_q \in \mathcal{F}_t$ (see Definition 1.1) be a finite field with q elements. Suppose

$$\beta = \alpha^r \quad \text{for some } \alpha \in \mathbb{F}_q \text{ and some integer } r > 1. \quad (2.1)$$

The problem of taking r th 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

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

for the rest of the section. We show a deterministic polynomial time algorithm (Algorithm 2.11) for finding a non-trivial factor of $x^r - \beta$. Then α can be computed by Lemma 2.1 below. The input parameters are r , e , β and \mathbb{F}_q (includes t , r_1, \dots, r_m , e_1, \dots, e_m defined in Definition 1.1), which are globally available. Unlike other algorithms for taking r th roots, Algorithm 2.11 does not require an r th nonresidue as an input and the associated proofs do not require any unproven hypothesis, like the Riemann Hypothesis. For the rest of this section, let

$$\rho = \zeta_r, \quad \text{a fixed primitive } r \text{ root of unity in } \mathbb{F}_q. \quad (2.2)$$

Lemma 2.1. Given a non-trivial factor of $x^r - \beta$, we can compute an r th root of β efficiently.

Proof. Suppose $x^d + a_{d-1}x^{d-1} + \dots + a_0 \in \mathbb{F}_q[x]$ 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 r th root of β . \square

The computations of the square root algorithms in [20] are performed over a group G_α . It is possible to formulate the square root algorithms over $\mathbb{F}_q[x]/(x^2 - \beta)$ for factoring $x^2 - \beta$. We generalize this idea and work on $\mathbb{F}_q[x]/(x^r - \beta)$ for factoring $x^r - \beta$. The “problem” of working on $\mathbb{F}_q[x]/(x^r - \beta)$ is that there are non-units 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 [11]. He works on the ring $\mathbb{Z}/n\mathbb{Z}$ for some composite 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 2.2. `isZero(f)` */* Comment: is $f(x) \equiv 0 \pmod{x^r - \beta}$? */*
{
 compute $h(x) = (f(x), x^r - \beta)$;
 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 $x^r - \beta$. */*
}

2.1 Step 1: Find a Suitable Element a

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

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

Then,

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

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 $h(x) = (\psi_a(x)^{rt} - 1) \pmod{x^r - \beta}$ 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

$$g_k(x, y, z) = (y - x)^k - z(y - \rho x)^k \in \mathbb{F}_q[x, y, z] \quad \text{for } k > 0. \quad (2.3)$$

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

Lemma 2.3. *Let $d_i = \text{ord } c_i$, the order of c_i in \mathbb{F}_q^\times . If d_i divides k for all $0 \leq i < r$, we have $\text{isZero}(g_k(x, a, 1)) = \text{TRUE}$. If d_i does not divide k for all $0 \leq i < r$, we have $\text{isZero}(g_k(x, a, 1)) = \text{FALSE}$. If there exists i_0, i_1 such that d_{i_0} divides k but d_{i_1} does not divide k , $\text{isZero}(g_k(x, a, 1))$ outputs a non-trivial factor of $x^r - \beta$.*

Proof. It is obvious. □

We have the following algorithm finding a desired a .

Algorithm 2.4. findA()
{
 for $i = 1$ to $rt + 1$
 {
 set $a_i = i$ th element in \mathbb{F}_q ;
 if $a_i^r = \beta$
 output $x - a_i$ and halt;

 if $\text{isZero}(g_{rt}(x, a_i, 1)) = \text{FALSE}$
 return a_i ;
 }
}

Lemma 2.5. *There are at most rt distinct values of $a \in \mathbb{F}_q$ such that $a^r \neq \beta$ and $\text{isZero}(g_{rt}(x, a, 1)) = \text{TRUE}$.*

Proof. Suppose $a^r \neq \beta$. The case $\text{isZero}(g_{rt}(x, a, 1)) = \text{TRUE}$ 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. □

2.2 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 $\text{isZero}(g_{rt}(x, a, 1)) = \text{FALSE}$. Let $d_i = \text{ord } \psi_a(\rho^i \alpha)$, the multiplicative order in \mathbb{F}_q^\times . We have d_i not dividing rt for all $0 \leq i < r$ by Lemma 2.3. We have the following algorithm to find a suitable $\ell \in \{r_1, \dots, r_m\}$.

Algorithm 2.6. findL(a)
{
 for $j = 1$ to m
 if $r_j \neq r$ and $\text{isZero}(g_{(q-1)/r_j^{e_j}}(x, a, 1)) = \text{FALSE}$
 return r_j ;

```

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

```

All the return statements in Algorithm 2.6 are conditional. It might seem that $\text{findL}(a)$ may terminate without returning any value and giving any output. We show below that it is not the case.

Lemma 2.7. *Suppose every call of isZero in $\text{findL}(a)$ returns TRUE or FALSE but does not output a non-trivial factor of $x^r - \beta$. If $\text{isZero}(g_{(q-1)/r_j^{e_j}}(x, a, 1)) = \text{TRUE}$ for all $r_j \neq r$, then $\text{isZero}(g_{(q-1)/r^{e-1}}(x, a, 1)) = \text{FALSE}$.*

Proof. Suppose $\text{isZero}(g_{(q-1)/r_j^{e_j}}(x, a, 1)) = \text{TRUE}$ 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 $r^{et} = \gcd_{\substack{1 \leq j \leq m \\ r_j \neq r}}((q-1)/r_j^{e_j})$ for all i . Since d_i does not divide rt for all i , we have $\text{isZero}(g_{(q-1)/r^{e-1}}(x, a, 1)) = \text{FALSE}$. \square

2.3 Step 3: Compute a Non-trivial Factor

Let

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

Case 1: Suppose $\ell = r_{j_0} \neq r$ for some fixed j_0 , is the value obtained from the previous step. We have $\text{isZero}(g_{(q-1)/\ell^{e_{j_0}}}(x, a, 1)) = \text{FALSE}$. Since $\psi_a(\rho^i \alpha)^{q-1} = 1$ for all $0 \leq i < r$, we have $\text{isZero}(g_{q-1}(x, a, 1)) = \text{TRUE}$. Suppose $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1))$ returns either TRUE or FALSE but does not output a non-trivial factor for $0 \leq k \leq e_{j_0}$. By the lemma below, there exists $0 < k_0 < e_{j_0}$ such that $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) = \text{TRUE}$ for $k = 0, 1, \dots, k_0$ and $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) = \text{FALSE}$ for $k = k_0 + 1, \dots, e_{j_0}$.

Lemma 2.8. *If $\text{isZero}(g_k(x, a, 1)) = \text{TRUE}$, then $\text{isZero}(g_{nk}(x, a, 1)) = \text{TRUE}$ for any positive integer n .*

Proof. If $\text{isZero}(g_k(x, a, 1)) = \text{TRUE}$, we have $\psi_a(\rho^i \alpha)^k = 1$ for all $0 \leq i < r$. Then $\psi_a(\rho^i \alpha)^{nk} = 1$ for all $0 \leq i < r$. Finally, $\text{isZero}(g_{nk}(x, a, 1)) = \text{TRUE}$ by Lemma 2.3. \square

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

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

where ζ_ℓ is a primitive ℓ th 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 1.1.

Lemma 2.9. *Let N be a prime power with $1 < N \neq r$. For some positive integer D , suppose*

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

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

Proof. Suppose $n_0 = \dots = n_{r-1} = n$ for some integer n with $(n, N) = 1$. Let $\zeta = \zeta_N^n$. For all $0 \leq i < r$, 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. \square

By Lemma 2.9 (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 $n = 1, \dots, \ell - 1$. See procedure `factorByZeta` (with $N = \ell$) in Algorithm 2.11.

Case 2: Suppose $\ell = r$. The situation is similar: we have $\text{isZero}(g_{(q-1)/r^{e-1}}(x, a, 1)) = \text{FALSE}$. Suppose $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1))$ returns either TRUE or FALSE but does not output a non-trivial factor for $0 \leq k \leq e - 1$. By Lemma 2.8 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 $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) = \text{TRUE}$ for $k = 0, 1, \dots, k_0$ and $\text{isZero}(g_{(q-1)/\ell^k}(x, a, 1)) = \text{FALSE}$ for $k = k_0 + 1, \dots, e - 1$. Let $d = (q-1)/r^{k_0+2}$. For $i = 0, \dots, r-1$, the element $\psi_a(\rho^i \alpha)^d$ is a primitive r^2 th root of unity. Then,

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

By Lemma 2.9 (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 2.11.

Computing a Primitive r^2 th Root of Unity In case 2, we need to find a primitive r^2 th root of unity, ζ_{r^2} . We have ρ , a primitive r th root of unity, by assumption. Then ζ_{r^2} can be computed by finding a non-trivial factor of $x^r - \rho$. Compute $a = \text{findA}()$ in step 1 and $\ell = \text{findL}(a)$ 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

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

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 `factorByX` in Algorithm 2.11.

Lemma 2.10. *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

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

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

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

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

2.4 The Algorithm

We present the entire algorithm below and prove Theorem 1.2 at the end of the section.

Algorithm 2.11. `factor($x^r - \beta$)`

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

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

  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  $\zeta_N$ , a primitive  $N$ th root of unity;
  for each  $n \in (\mathbb{Z}/N\mathbb{Z})^\times$ 
     $\text{isZero}(g_d(x, a, \zeta_N^n))$ ;
}

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

Proof of Theorem 1.2. From our discussion in this section, Algorithm 2.11 (together with Lemma 2.1) is a deterministic algorithm for computing an r th root of any r th 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 r th 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 r th nonresidue in \mathbb{F}_q , we keep taking r th roots beginning from ζ_r and obtain $\zeta_{r^2} = \sqrt[r]{\zeta_r}$, $\zeta_{r^3} = \sqrt[r]{\zeta_{r^2}}$, \dots , $\zeta_{r^e} = \sqrt[r]{\zeta_{r^{e-1}}}$. Finally, ζ_{r^e} is an r th nonresidue in \mathbb{F}_q . \square

Proof of Theorem 1.3. For any $\mathbb{F}_q \in \mathcal{F}_1$, an r_i th nonresidue $\zeta_{r_i^{e_i}} \in \mathbb{F}_q$ can be computed in deterministic polynomial for each i by Theorem 1.2. Then the product $\prod_{i=1}^m \zeta_{r_i^{e_i}}$ is a primitive element in \mathbb{F}_q . \square

3 Primality testing

Let N be a positive integer. Consider the problem of determining whether N is a prime. In [20], a deterministic primality test is constructed from a deterministic square root algorithm and Proth's theorem. See [25] for the details of Proth's theorem. The idea is generalized: a deterministic primality test can be constructed from the deterministic r th root algorithm presented in the previous section and a generalized Proth's theorem (Theorem 3.2 below). This generalization of Proth's theorem is well known. The idea of our primality test is similar to Pocklington-Lehmer primality test. Theorem 1.4 is proved in the following.

Proof of Theorem 1.4. Firstly, we try to find a primitive r th root of unity ζ_r by Algorithm 4.8 in [20] if $\zeta_r \in (\mathbb{Z}/N\mathbb{Z})^\times$. If $\zeta_r \notin (\mathbb{Z}/N\mathbb{Z})^\times$, Algorithm 4.8 in [20] will fail and we conclude that N is composite. Otherwise, try computing $\zeta_{r^2} = \sqrt[r]{\zeta_r}$, $\zeta_{r^3} = \sqrt[r]{\zeta_{r^2}}$, \dots , $\zeta_{r^e} = \sqrt[r]{\zeta_{r^{e-1}}}$ over the ring $\mathbb{Z}/N\mathbb{Z}$ by Algorithm 2.11. 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 Theorem 3.2 below. Since Algorithm 2.11 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 4.8 in [20] and Algorithm 2.11 are $\tilde{O}(t \log^2 N)$ and $\tilde{O}(r^2 \log N(t + r \log N))$, respectively. The overall running time is $\tilde{O}(r^2 \log^2 N(t + r \log N))$. The theorem follows. \square

We will use the following lemma to prove Theorem 3.2.

Lemma 3.1. *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. \square

Theorem 3.2. (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*

$$a^{N-1} \equiv 1 \pmod{N} \quad \text{and} \quad a^{(N-1)/r} \not\equiv 1 \pmod{N}, \quad (3.1)$$

for some integer a , then N is a prime.

Proof. Suppose there exists an integer a satisfying equations (3.1). 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 3.1, $k = 1$ and N is a prime.

Suppose $N = \ell_1^{k_1} \cdots \ell_m^{k_m}$ for $m > 1$, some distinct primes ℓ_1, \dots, ℓ_m and integers $k_1, \dots, k_m \geq 1$. 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 $s_1 = \max(s_1, \dots, s_m)$. 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 3.1. 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. \square

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 [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 $\tilde{O}(\log^4 N)$.

4 Solving polynomial equations

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 ℓ th roots for all $\ell | q - 1$.

Write $q = p_1^{e_1} \cdots p_m^{e_m} + 1$ for some distinct primes p_1, \dots, p_m . Suppose we can compute the p_j th roots of a for any $a \in \mathbb{F}_q$ and $1 \leq j \leq m$. We show a deterministic algorithm (Algorithm 4.1 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 ℓ th 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 $i = 0, \dots, \ell - 1$. 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. factorBySearching(f)

<pre> { 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 $i_0 = \text{search}(f, j, d, b)$; set $a = \zeta_{p_j}^{i_0} b$; } } </pre>	<pre> search(f, j, d, b) { for $i = 0$ to $p_j - 1$ { compute $g(x) = (f(x), x^d - \zeta_{p_j}^i b)$; if $1 < \deg g < \deg f$ Label 2: output g and halt; else if $\deg g = \deg f$ return i; } } </pre>
--	--

Lemma 4.2. *Suppose $f(x) \in \mathbb{F}_q[x]$ is product of linear factors such that $f(0) \neq 0$. Algorithm 4.1 always outputs a non-trivial factor of f .*

Proof. It is easy to see that a always is a p_j th 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 $g_i(x) = (f(x), x^{d_{j_0, k_0}} - \zeta_{p_{j_0}}^i b)$ for $i = 0, \dots, p_{j_0} - 1$. Then,

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

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 $a = \zeta_{p_{j_0}}^{i_0} a_{j_0, k_0}^{1/p_{j_0}}$ 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 $x - \zeta_{p_{j_m}}^i a^{1/p_{j_m}}$ does not divide $f(x)$ for all $i = 0, \dots, p_{j_m} - 1$ since the algorithm does not output. It leads to a contradiction. \square

Theorem 4.3. *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 ℓ th 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 ℓ th roots can be computed in deterministic polynomial time, Algorithm 4.1 can factor f in deterministic polynomial time by Lemma 4.2. \square

Theorem 4.4. *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 ℓ th nonresidue in \mathbb{F}_q for all prime factor ℓ of $q - 1$. Then, ℓ th roots can be computed by a generalized Shanks' algorithm with a as an input. See Section 3.2 in [22] for modifying Shanks' algorithm to take r th roots. Finally, the theorem follows from Theorem 4.3. \square

Proof of Theorem 1.5. It is an obvious consequence of Theorem 1.2 and Theorem 4.3. \square

5 Elliptic curve “ n th root” problem

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

Let $F \in \mathcal{F}_1$ (see Definition 1.1) be a finite field. Denote an elliptic curve E defined over F by the Weierstrass equation

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

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

- (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 n th 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 [24] for the details.

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

$$f(x) \stackrel{\text{def}}{=} R_1(x) - aS_1(x) = 0 \tag{5.1}$$

over F . Let $\alpha_1, \dots, \alpha_m \in F$ be the roots of the equation (5.1). 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, $P \in E[n](F) \stackrel{\text{def}}{=} E[n] \cap E(F)$, where $E[n]$ is the n -torsion subgroup of $E(\overline{F})$, where \overline{F} denotes a fixed algebraic closure of F . Let $\alpha_1, \dots, \alpha_m \in F$ be the roots of the equation $\Psi(x) = 0$. Consider the quadratic equation

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

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 .

Proof of Theorem 1.6. The polynomial equations can be solved in deterministic polynomial time by Theorem 1.5. By the discussion above, the theorem follows. \square

Acknowledgments: We thank Lawrence C. Washington for his guidance of the study. He provided a lot of invaluable comments in this work.

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] Paulo S. Barreto and José Felipe Voloch. Efficient computation of roots in finite fields. *Des. Codes Cryptography*, 39(2):275–280, 2006.
- [4] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.
- [5] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713–735, 1970.
- [6] Johannes Buchmann and Victor Shoup. Constructing nonresidues in finite fields and the extended riemann hypothesis. *Math. Comp.*, 65(215):1311–1326, jul 1996.
- [7] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36(154):587–592, 1981.
- [8] 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.
- [9] 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.
- [10] Joachim Von Zur Gathen and Daniel Panario. Factoring polynomials over finite fields: a survey. *J. Symb. Comput.*, 31(1-2):3–17, 2001.
- [11] Hendrik W. Lenstra Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126:649–673, 1987.
- [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] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515–534, 1982.

- [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] Arnold Schönhage and Volker Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.
- [18] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p . *Math. Comp.*, 44(170):483–494, apr 1985.
- [19] 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.
- [20] Tsz-Wo Sze. On taking square roots and constructing quadratic non-residues over finite fields, 2007. Preliminary version. Available at <http://www.cs.umd.edu/~szetszwo/papers/qnr.pdf>.
- [21] Alberto Tonelli. Bemerkung über die Auflösung quadratischer Congruenzen. *Nachrichten der Akademie der Wissenschaften in Göttingen*, pages 344–346, 1891.
- [22] Christiaan van de Woestijne. *Deterministic equation solving over finite fields*. PhD thesis, Universiteit Leiden, Leiden, Netherlands, 2006.
- [23] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, United Kingdom, 2nd edition, 2003.
- [24] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2003.
- [25] Hugh C. Williams. *Édouard Lucas and Primality Testing*, volume 22 of *Canadian Mathematical Society Series of Monographs and Advanced Texts*. Wiley-Interscience, 1998.
- [26] 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.