Designs, Codes and Cryptography (2024) 92:1947–1959
https://doi.org/10.1007/s10623-024-01374-1

Square root computation in finite fields

Ebru Adiguzel-Goktas[1] · **Enver Ozdemir**[2]

Received: 26 December 2022 / Revised: 7 January 2024 / Accepted: 6 February 2024 /
Published online: 12 March 2024

Abstract
In this paper, we present a review of three widely-used practical square root algorithms. We then describe a unifying framework where each of these well-known algorithms can be seen as a special case of it. The framework with singular curves offers a broad perspective to compare and further improve the existing methods in addition to offering a new avenue for square root computation algorithms in finite fields.

Keywords Square roots · Singular curves · Elliptic curves

Mathematics Subject Classification Primary 11Y99 · 68Q99

1 Introduction

Finding square roots in finite fields has been an interest of many researchers in computational number theory [3]. In a strict sense, the problem is reduced to finding square roots modulo a prime number p. In other words, the problem is converted to computing square roots in a finite field $\mathbb{F}_p$ where p is a prime integer and $\mathbb{F}_p$ is a field with p elements. The square root computation is a dominating step of several other methods in computations. For example, any sieving method for factorization of integers or the elliptic curve primality proving method requires an enormous number of square root computations [3]. Even though frequently a new method for computing square root in finite fields appears [5, 7], the two oldest methods, Tonelli-Shanks and Cipolla [2, 11], are still the most popular algorithms in practice in addition to a relatively recent one presented by Peralta [10]. Each algorithm has its own advantages and disadvantages over the others. First of all, all three algorithms are probabilistic and in terms of probability of success for each trial, they are all the same considering the worst-

Communicated by D. Panario.

✉ Ebru Adiguzel-Goktas
ebru.adiguzel@agu.edu.tr

Enver Ozdemir
ozdemiren@itu.edu.tr

[1] Department of Computer Science, Abdullah Gul University, Kayseri, Turkey

[2] Informatics Institute, Istanbul Technical University, Istanbul, Turkey

case scenario. In other words, the probability of success for a single trial is around $1/2$ for each of these algorithms. However, considering a general case, Peralta's algorithm has higher probability of success. On the other hand, Tonelli–Shanks algorithm has lower computational load most of the time. We are going to present a method for square root computing which also allows us to observe aforementioned algorithms in the same context. The observation leads to an efficient way of analyzing and comparing of algorithms. For example, despite Peralta's algorithm success rate being stated to be at least 1/2 in the original manuscript, we show that the actual probability can be improved to be at least 3/4 and this can be achieved by adding one step while implementing Peralta's algorithm with a singular cubic. We present that the success rate for a prime p where $p-1=2^e m$ and m is odd depends on the exponent e.

The well-known three algorithms are quite efficient in practice but the scholarly work on finding a method for computing square roots continues in the direction of finding a practical and deterministic algorithm [7]. In what follows, we aim to explain these three algorithms as part of single method with singular curves to provide a wider insight that might help for further development toward a practical and deterministic square root algorithm. The singular curve explanation of these methods might also help understanding of current methods and their efficiency. In other words, each method is an analogue of finding special torsion points of certain curves where Tonelli–Shanks and Cipolla's algorithms seek sufficient conditions while Peralta's algorithm only seeks to fulfill the necessary condition for the same purpose. In this respect, we first present a brief description of Tonelli–Shanks, Cipolla, and Peralta's algorithms in the next section. The third section introduces the mathematical objects for the method and presentation of each algorithm in terms of these mathematical objects. We include performance analysis in the last section.

2 Square roots algorithms

Let p be an odd prime integer and $\mathbb{F}_p$ be a prime field with p elements. Let $a \neq 0 \in \mathbb{F}_p$. It is not hard to determine whether a has a square root in $\mathbb{F}_p$. In fact, the multiplicative group $G=(\mathbb{F}_p^*,\cdot)$ is cyclic and so $a=g^\beta$ for some positive integer β, where $g \in G$ is a generator of G. β is an even integer if and only if a has a square root. In other words, β being odd implies that a is a quadratic non-residue modulo p. It is not a tedious task to determine whether β is even or not. As g is a generator of G, then

$$g^{\frac{p-1}{2}} = -1 \in G.$$

In this respect;

$$a^{\frac{p-1}{2}} = (g^\beta)^{\frac{p-1}{2}} = (g^{\frac{p-1}{2}})^\beta = (-1)^\beta \text{ in } G.$$

Therefore, $a^{\frac{p-1}{2}} \mod p$ is 1 if and only if a is a quadratic residue modulo p. The formulation of quadratic residue/non-residue is given by the Legendre symbol;

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if a is quadratic residue modulo p and } a \not\equiv 0 \mod p \\ -1, & \text{if a is a non-quadratic residue modulo p} \\ 0, & a \equiv 0 \mod p \end{cases}$$

Once decided that a has a square root, the next task is to find it.

Let's assume for a moment that the prime integer $p \equiv 3 \mod 4$. We are also assuming a has a square in $\mathbb{F}_p$, i.e., $\left(\frac{a}{p}\right) = 1$. As

$$\left(a^{(p+1)/4}\right)^2 = a^{(p+1)/2} \equiv a^{(p-1)/2} \cdot a \equiv 1 \cdot a \mod p,$$

$x = a^{(p+1)/4}$ gives a square root of $a \in \mathbb{F}_p$.

We do not have a generic square root formula for all other primes except for $p \equiv 5 \mod 8$. In fact, $a^{(p-1)/2} \equiv 1 \mod p$ implies either

$$a^{(p-1)/4} \equiv 1 \mod p \text{ or } a^{(p-1)/4} \equiv -1 \mod p.$$

- If $a^{(p-1)/4} \equiv -1 \mod p$, $x \equiv 2a(4a)^{(p-5)/8} \mod p$ gives a solution for $x^2 \equiv a \mod p$ as $2^{\frac{p-1}{2}} = \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} = -1$.
- Otherwise, $x \equiv a^{(p+3)/8} \mod p$ gives a square root of a.

From now on, we are assuming all primes $p \equiv 1 \mod 8$. The first method for square root computation that we are going to present briefly is due to Tonelli. The method is later improved by Shanks.

2.1 Tonelli–Shanks algorithm

The multiplicative group $G = (\mathbb{F}_p^*, \cdot)$ has order exactly $p - 1$ so for any element $s \in G$ we have

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

Lets write $p - 1 = 2^e m$ for some $e \geq 3$ and odd integer m. Then the group G has a 2-Sylow subgroup H of order 2^e. Let z be a generator of H and b be equal to a^m. Then b has order 2^j for some $j \geq 0$ since $b = a^m$ lies in H. As z is a generator of H, there must be an integer $r \geq 0$ such that $b \equiv z^r \mod p$. Since a is a quadratic residue, i.e., $a^{\frac{p-1}{2}} \equiv 1 \mod p$, so b is also a quadratic residue and then r must be even integer.

The fact that r is even implies $x = a^{(m+1)/2} z^{-r/2}$ a desired square root of a. Indeed,

$$x^2 \equiv a^{m+1} z^{-r} \equiv a^m \cdot a \cdot b^{-1} \equiv b \cdot a \cdot b^{-1} \equiv a \mod p$$

Now, the problem is how to find a generator z of the 2-Sylow subgroup H of G. The best way of finding the generator of H is to choose a random integer n such that $\left(\frac{n}{p}\right) = -1$. Then $z = n^m$ is a generator of H since $z^{2^{e-1}} \equiv n^{(m2^e)/2} \equiv n^{(p-1)/2} \equiv -1 \mod p$ i.e., z has order exactly 2^e in H. At the current state of the art, the only practical way to find such n is the trial and error method. Assuming GRH, n is smaller than $2 \log^2 p$ [1]. Let n be an element of G, as G is cyclic and g is a generator of it, there is a positive integer α such that $n = g^\alpha$. The random number n gives a generator of H if and only if the positive integer α is odd. In other words, for random $n \in G$, the probability that n^m is a generator of H is around $1/2$. This part of Tonelli–Shanks algorithm is probabilistic and the success rate for a single trial is almost 1/2. The resulting probabilistic algorithm for a square root computation performs the following steps:

Algorithm 1 Tonelli–Shanks Algorithm

Input: a quadratic residue a modulo p where p is an odd prime such that $p-1=2^e m$.
Output: $\sqrt{a} \mod p$
(1) Choose numbers n at random until $\left(\frac{n}{p}\right) = -1$
(2) Set $z = n^m \mod p$ and $b \equiv a^m \mod p$.
(4) Find the smallest integer $r \geq 0$ such that $b \equiv z^r \equiv n^{mr} \mod p$.
(5) Set $x \equiv a^{(m+1)/2} z^{-r/2} \equiv \sqrt{a} \mod p$.

We should also note that finding r might be a tedious task if e is a large number, say more than 99. In such a case, the usage of Tonelli–Shanks might not be suitable for practical purposes. However, Cipolla's algorithm stands as a little more practical alternative if e is large.

2.2 Cipolla's algorithm

Cipolla's Algorithm [2] is described as a square root method while using the extension field $\mathbb{F}_{p^2}$ of $\mathbb{F}_p$. Even though the probability of success is the same as the Tonelli–Shanks algorithm, computing in extension brings an extra burden for this method. As mentioned above, step 4 of Algorithm 1 handles the discrete logarithm problem in the 2-Sylow subgroup H of G. Once the size of H gets large, Cipolla's method becomes more practical. In what follows, we give brief details of Cipolla's method.

Let t be an integer with $0 \leq t \leq p-1$ such that $u = t^2 - a$ is quadratic non-residue modulo p. The polynomial $x^2 - u$ is an irreducible polynomial in $\mathbb{F}_p$ and $\mathbb{F}_p[x]/(x^2-u)$ is isomorphic to the extension field $\mathbb{F}_{p^2}$. Let $w = \sqrt{u}$ in $\mathbb{F}_{p^2}$. As w can not be an element of $\mathbb{F}_p$, one can define the extension field as;

$$\mathbb{F}_p[\sqrt{u}] = \mathbb{F}_p[w] = \{x + yw : x, y \in \mathbb{F}_p\}.$$

For every element $x + yw \in \mathbb{F}_p[w]$, we have $(x+yw)^p = x - yw$, since

$$w^{p-1} \equiv (w^2)^{(p-1)/2} \equiv u^{(p-1)/2} \equiv -1 \mod p.$$

Theorem 2.1 *The element $b = (t+w)^{p+1/2}$ lies in $\mathbb{F}_p$ and it is a square root of a.*

Proof

$$b^2 = (t+w)^{p+1} = (t+w)(t+w)^p = (t+w)(t-w) = t^2 - w^2 = t^2 - (t^2 - a^2) = a.$$

□

The first aim of Cipolla's method is to find a primitive element u in $\mathbb{F}_{p^2}$ such that $u = t^2 - a$. For our case, u is a primitive element if and only if $t^2 - a$ is a quadratic non-residue modulo p. As in the case of Tonelli–Shanks algorithm, there is no deterministic method for finding such a primitive element. The desired primitive element is found via the trial and error method. Again, for a random t, the probability that $u = t^2 - a$ gives a primitive element is around $1/2$. The implementation of Cipolla's algorithm follows the following steps:

Algorithm 2 : Cipolla's Algorithm

Input: an odd prime p and a quadratic residue a modulo p.
Output: $\sqrt{a} \mod p$
(1) Find an integer t with $0 \leq t \leq p-1$ such that $u = t^2 - a$ is a quadratic non-residue $\mod p$.
(2) Return $(t + \sqrt{u})^{(p+1)/2}$.

As mentioned above the last step of the algorithm requires more multiplication operations than the Tonelli–Shanks method as w does not lie in $\mathbb{F}_p$. It lies strictly in the extension field $\mathbb{F}_{p^2}$. The running time of the algorithm is bounded by $O(M \log^2 p)$ while Tonelli–Shanks is $O(M(\log p + e^2))$ where M stands for complexity of multiplication in $\mathbb{F}_p$. Moreover, the complexity of the Tonelli–Shanks approach can be improved by using a faster algorithm for the discrete logarithm which yields a complexity of $O(M(\log p + e \log e / \log \log e))$ [12]. In practice, this approach will almost always outperform Cipolla unless e is very large.

2.3 Peralta's algorithm

A relatively new and practical method by Peralta [10] also exploits the group $(\mathbb{Z}_p^*, \cdot)$. Unlike Tonelli–Shanks and Cipolla's algorithms, the probability of success does not stand same all the time. If $p - 1 = 2^e m$ with m odd, the probability of success depends on e and it is at least 1/2. Consider a square a in the group $(\mathbb{Z}_p^*, \cdot)$. Let $f(y) = y^2 - a$ be a polynomial over $\mathbb{F}_p$. Note that $f(y)$ is reducible. The ring $R = \mathbb{Z}_p[y]/(f(y))$ is a factor ring and every nonzero element in

$$R = \mathbb{Z}_p[y]/(f(y)) = \{b_1 y + b_0 + (f(y)) \,|b_0, b_1 \in \mathbb{Z}_p\}$$

is either a unit or a zero-divisor. Since the set of units R^* in R form a finite abelian group under multiplication modulo $f(y)$ and every finite abelian group is a direct products of cyclic groups of prime order, R^* is isomorphic to $\mathbb{Z}^*{}_p \oplus \mathbb{Z}^*{}_p$. Hence,

$$e^{p-1} \equiv 1 \mod p \text{ for every e} \in R^*.$$

The ring R can also be considered as the ring $\mathbb{Z}_p[\sqrt{a}]$ and the norm of any unit in this ring should be 1. In other words, if $r + s\sqrt{a}$ is a multiplicative unit then $N(r + s\sqrt{a}) = (r + s\sqrt{a})(r - s\sqrt{a}) = r^2 - s^2 a = 1$ which also means that $r^2 - s^2 a \neq 0$ where N is the norm function. Therefore, R^* consists of elements of the form $r + s\sqrt{a}$ such that $r^2 \not\equiv s^2 a \mod p$. If $r \in \mathbb{Z}_p^*$ and $r^2 \equiv a \mod p$, then r would be our desired result. From now on, we will consider the case $r^2 \not\equiv a \mod p$ and we focus on the elements of R^* of the form $r + \sqrt{a}$. Consider

$$(r + \sqrt{a})^{(p-1)/2} = u + v\sqrt{a}.$$

then we have

$$(r + \sqrt{a})^{(p-1)} \equiv 1 \equiv u^2 + 2uv\sqrt{a} + v^2 a \mod p.$$

It implies that $2uv = 0$, i.e., either u or v is 0. In the case $u = 0$ we have,

$$(r + \sqrt{a})^{(p-1)/2} = 0 + v\sqrt{a} \implies (r + \sqrt{a})^{p-1} \equiv 1 \equiv v^2 a \mod p.$$

Then v^{-1} is our desired square root of a. Hence, if one can find a random $r \in \mathbb{Z}_p^*$ such that $r^2 \not\equiv a \mod p$ and $(r + \sqrt{a})^{(p-1)/2} = 0 + v\sqrt{a}$, then the solution of the square root problem will be v^{-1}. The algorithm based on this above observation follows the steps: A

Algorithm 3 : Peralta's Algorithm I

Input: a quadratic residue a modulo p
Output: $x \equiv \sqrt{a} \mod p$.
(1) Choose $r \in \mathbb{Z}_p^*$ at random such that $r^2 \not\equiv a \mod p$, otherwise output is r.
(2) Compute $(r + \sqrt{a})^{(p-1)/2} = u + v\sqrt{a}$.
(3) If $u = 0$, output $x \equiv v^{-1} \mod p$ else go to (1).

slightly modified version of the above algorithm: The probability of success depends on the

Algorithm 4 : Peralta's Algorithm II

Input: a quadratic residue a modulo p.
Output: $x \equiv \sqrt{a} \mod p$.
(1) Choose r at random $\in \mathbb{Z}_p^*$.
(2) If $r^2 \equiv -a \mod p$, choose a new r.
(3) Compute $(r + \sqrt{-a})^m = u + v\sqrt{-a}$, where $p - 1 = 2^e m$ and m is an odd integer.
(4) If either u or v is 0, choose a new r.
(5) Compute $(u + v\sqrt{-a})^{2^i}$ for some $i = 1, 2, \cdots, e$ until $(u + v\sqrt{-a})^{2^i} = 0 + w\sqrt{-a}$ for some w.
(6) Let $(u + v\sqrt{-a})^{2^{i-1}} = k + l\sqrt{-a}$. Then $k^2 - l^2 a \equiv 0 \mod p$ and output k/l.

number e. In fact, for a random r, we have $1 - 1/2^{e-1}$ chance to end up with a square root of a (See [10, Theorem 4]). On the other hand, as mentioned above, when we look at Peralta's algorithm through singular curves, we would notice that the actual probability of success is $1 - 1/2^e$. Peralta's algorithm serves for all integers but as we do have a generic square root formula for $p \equiv 3 \mod 4$, the probability of success was defined to be 1/2 by Peralta considering the worst-case scenario ($e = 2$) event though we see in a moment that it is 3/4.

3 Geometric analogue of algorithms

In this section, we present analogues of all three algorithms for square root computing [2, 10, 11] using the Jacobian group of singular cubics. In the first part, we describe the group structure of a singular curve's Jacobian group and then show that the square root is embedded in certain points in this group [9].

3.1 Square root algorithm with curves

Let E be an elliptic curve over a finite field $\mathbb{F}_p$ with p elements where p is an odd prime integer. In particular, the curve E can be defined as

$$y^2 = x^3 + Ax + B \text{ over the field } \mathbb{F}_p. \tag{3.1}$$

3.2 Computing in an elliptic curve group

3.2.1 Affine coordinates

In case the affine coordinates is employed for representation of elements in the elliptic curve $E : y^2 = x^3 + Ax + B$ group, an addition operation can be performed as follows:

Addition

Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ be points on E such that $P + Q = P_3 = (x_3, y_3)$.

- If $P_1 \neq \pm P_2$ then $x_3 = k^2 - x_1 - x_2$, $y_3 = k(x_1 - x_3) - y_1$, $k = \frac{y_1 - y_2}{x_1 - x_2}$.
- If $x_1 = x_2$ but $y_1 \neq y_2$, then $P_1 + P_2 = \infty$.

Doubling

Let $2P_1 = (x_3, y_3)$. Then

$$x_3 = k^2 - 2x_1,\ y_3 = k(x_1 - x_3) - y_1,\ k = \frac{3x_1^2 + A}{2y_1}. \tag{3.2}$$

An addition and doubling operation require $(I + 2M + S)$ and $(I + 2M + 2S)$ calculations, where I is inversion, M is multiplication and S is squaring in finite field, respectively.[[14]]

Since the calculation of inversion is costly, representing elements with projective coordinates in E might be more suitable in practice.

3.2.2 Projective coordinates

For representing elements in projective coordinates, the equation of E is written as

$$Y^2Z = X^3 + AXZ^2 + BZ^3.$$

The point $P_1 = (X_1 : Y_1 : Z_1)$ on E corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$ when $Z_1 \neq 0$, otherwise it represents the point at infinity $P_\infty = (0 : 1 : 0)$. Also, $-P_1 = (X_1 : -Y_1 : Z_1)$.

Addition

Let $P_1 = (X_1 : Y_1 : Z_1)$, $P_2 = (X_2 : Y_2 : Z_2)$ on and $P_3 = P_1 + P_2 = (X_3 : Y_3 : Z_3)$.

- If $P_1 \neq P_2$, then set

$$K = Y_2Z_1 - Y_1Z_2,\ L = X_2Z_1 - X_1Z_2,\ M = K^2Z_1Z_2 - L^3 - 2L^2X_1Z_2$$

so that

$$X_3 = LM,\ Y_3 = K(L^2X_1Z_2 - M) - L^3Y_1Z_2,\ Z_3 = L^3Z_1Z_2.$$

- If $P_1 = -P_2$, then $P_1 + P_2 = \infty$.

Doubling

Let $2[P_1] = (X_3 : Y_3 : Z_3)$ then set

$$K = 4X_1Y_1^2,\ L = 3X_1^2 + AZ_1^4$$

and

$$X_3 = -2K + L^2,\ Y_3 = -8Y_1^4 + L(K - X_3),\ Z_3 = 2Y_1Z_1.$$

The complexities for an addition and doubling are $12M + 4S$, $4M + 6S$, respectively. No inversion is needed.

The idea of using singular curves for square root finding is based on the following observation. Let P_1, P_2, P_3 be points on the elliptic curve such that $P_1 = (x_1, 0)$, $P_2 = (x_2, 0)$ and $P_3 = (x_3, 0)$. In other words, x_1, x_2, x_3 are the x-coordinates where the graph of E is crossed with the x-axis and this means that the tangent line at the point P_i is vertical for $i = 1, 2, 3$. The group operation in the elliptic curve group $E(\mathbb{F}_p)$ indicates that P_i is equal to its inverse as the tangent line is vertical at these points. That implies $2P_i = \infty$ where ∞ represents the identity of the group $E(\mathbb{F}_p)$. Since the right side of Eq. (3.1) can have at most three distinct roots, there are at most three 2-torsion points in the curve E's group. The observation basically says reaching a 2-torsion on the curve gives a root of $x^3 + Ax + B$.

In particular, define $E : y^2 = (x - a)(x^2 - a)$ on $\mathbb{F}_p$ where a is a quadratic residue modulo p. The equation $(x - a)(x^2 - a)$ has three distinct roots in $\mathbb{F}_p$. Hence, the points $P_1 = (a, 0)$, $P_2 = (\sqrt{a}, 0)$ and $P_3 = (-\sqrt{a}, 0)$ are the only 2-torsion points of E. In this regard, two 2-torsion points on E give a square root of a. Since the points $\{P_1, P_2, P_3, \infty\}$ forms a subgroup of order 4, the order of the group on the elliptic curve should be divisible by 4. Therefore, we can say that $\#E(\mathbb{F}_p) = 2^e m$ where m is an odd integer and $e \geq 2$. We should note here that finding the number of points on the curve can take at most polynomial time [4] so we can assume that the order of E is computed in an efficient way. Consider a random point Q on E. Once mQ is not the identity in $E(\mathbb{F}_p)$, $2^j mQ$ gives a two torsion for some $0 \leq j < e$. One can define a square root algorithm based on this observation. However, the problematic part is locating a random point on the curve E. Even though, one can easily say $Q_1 = (0, a)$ and $Q_2 = (1, 1 - a)$ are points on the curve, the case $mQ_1 = mQ_2 = \infty$ requires locating another random point to be able to continue with the algorithm. Therefore, using a plain form of the method might stop most of the time as finding a point on the curve also requires square root computation. In order to avoid this issue, singular curves can be used.

3.3 Singular curves

The curves for our purposes are defined in a special format that resembles the well-known elliptic curves. Let K be a field with a characteristic different from 2, the curves that we work with are defined by an equation of the form $y^2 = xf^2(x)$ over K where $f(x)$ is a monic square- free polynomial. Even though, the normalization of such curves returns the basic geometric tool of the projective line, the Jacobian group of these curves has the potential to be utilized for problems in computational number theory [6–8]. The employed singular curves are defined in a special format. A singular cubic curve that is utilized for square root computation is defined by $E : y^2 = x(x + a)^2$ over the field $\mathbb{F}_p$. The non-singular points on $E(\mathbb{F}_p)$ form an abelian group and the group operation is performed as described above for the elliptic curves [14, Sect. 2.10]. Note that if a change of coordinates $x = x_1 - \frac{2a}{3}$ on an equation $y^2 = x(x + a)^2$ is applied then the equation becomes $y^2 = x_1^3 - \frac{a^2}{3}x_1 - \frac{2a^3}{27}$. Therefore, we can apply above group operation formulas for $E : y^2 = x_1^3 + Ax + B$ where $A = -\frac{a^2}{3}$ and $B = -\frac{2a^3}{27}$ in $\mathbb{F}_p$. For the sake of simplicity, we use affine coordinates for the proofs below, however the projective coordinates give better performance in practical applications. Since the point $(-a, 0)$ is a singular point, the point $(0, 0)$ is the only 2-torsion point on this curve. For any point $P \neq (0, 0)$ on E, if mP is neither ∞ nor $(0, 0)$ then the order of P must be divisible by 4 and $2^i mP$ is a 4-torsion point of E for some $0 < i < e$.

We show in a moment that the 4-torsion points on this curve give the square root of a. In what follows, we show that computing square root via finding a 4-torsion is actually what other algorithms implicitly do.

3.3.1 Tonelli–Shanks and Peralta's algorithms with singular curves

Let a be a quadratic residue modulo p. We define a special singular cubic curve E with the equation $y^2 = x^2(x-a)$. As we mentioned above the normalization of E at the singular point $(a, 0)$ returns the projective line. This can also be seen via the following re-formalization

$$\left(\frac{y}{x}\right)^2 = x - a \tag{3.3}$$

This representation strikes the observation that leads to Peralta's algorithm. In other words, the coordinate ring of E and the ring in Peralta's algorithm match. Even though, the change of coordinates of any singular cubic curve returns to the same coordinate ring, the concealed singular cubic in Peralta's algorithm is revealed via the resulting algorithm's behavior. We are now going to show that Peralta's algorithm is nothing but the method of finding 4-torsion points in the Jacobian of E where $E : y^2 = x(x+a)^2$. Note that the change of coordinates $(x = x_1 + a)$ on E states that E can also be defined by $y^2 = x^2(x-a)$.

Remark 3.1 If an element $x \in \mathbb{F}_p$ is not a quadratic residue then $y^2 = x(x+a)^2$ has no solution modulo p. Because of this, for all points (x, y) on the given curve the first coordinate must be a quadratic residue. This observation allows for the parametrization $(t^2, t(t^2+a))$.

Theorem 3.2 *Let a be a square in $\mathbb{F}_p$, $E : y^2 = x(x+a)^2$ be a singular curve over $\mathbb{F}_p$. Then the points on E of order 4 give the square root of a.*

Proof Let $P = (x_1, y_1) = (t^2, t(t^2+a))$ be a point on E. We are going to find $2P = (x_3, y_3)$ in terms of t. The tangent line $y = k(x - x_1) + y_1$ intersects $E : y^2 = x(x+a)^2 = x^3 + 2ax^2 + xa^2$ and we have,

$$(k(x - x_1) + y_1)^2 = x(x+a)^2. \implies 0 = x^3 - (k^2 - 2a)x^2 + \cdots$$

Since the sum of three roots of the given cubic equation is $k^2 - 2a$, we have

$$x_1 + x_1 + x_3 = k^2 - 2a \implies x_3 = k^2 - 2a - 2x_1, y_3 = k(x_1 - x_3) - y_1.$$

Since

$$2y\frac{dy}{dx} = (x+a)(3x+a)$$

then we have

$$k = \frac{dy}{dx} = \frac{(x+a)(3x+a)}{2y}$$

This implies

$$x_3 = k^2 - 2a - 2x_1 = \frac{(t^2-a)^2}{(2t)^2}, y_3 = \frac{(t^2+a)^2(t^2-a)}{(2t)^3}.$$

It follows that;

$$2P = \left(\frac{(a-t^2)^2}{4t^2}, \frac{(t^2+a)^2(t^2-a)}{8t^3}\right). \tag{3.4}$$

If the order of the point P on E is 4, then $4P = \infty$ i.e., $(x_3, y_3) = 2P = -2P = (x_3, -y_3)$ which implies $y_3 = 0$ i.e, $(t^2 + a)^2(t^2 - a) = 0$. It follows that $t = \pm\sqrt{a}$ or $t = \pm\sqrt{-a}$. $(-a, 0)$ is a singular point, it is not in group and so t can not be $\pm\sqrt{-a}$. Therefore, the points that have order 4 are $(a, 2a\sqrt{a})$, $(a, -2a\sqrt{a})$ as t can only be $\sqrt{a}$ or $-\sqrt{a}$. □

Theorem 3.2 suggests a way of finding square root of a in $\mathbb{F}_p$. Let R be a random point on the curve E. The singular cubic curve E has attached abelian group consisting of non-singular points in addition to the point at infinity [14, Theorem 2.31]. Counting the points on E modulo p depends on the following observation: Every quadratic residue x other than $-a$ generates two points. That means there exists $p - 3$ remaining points. Adding the points $(0, 0)$ and ∞ yields $p - 1$ points. The group $E(\mathbb{F}_p)$ has order exactly $p - 1$. Let $p - 1 = 2^e m$ for some odd integer m and a positive integer $e \geq 2$. Let $R = (\ell^2, \ell(\ell^2 + a))$ be a random point on $E(\mathbb{F}_p)$ for some $\ell \neq 0 \in \mathbb{F}_p$. If $T = mR$ and 2T are not the identity then $2^i T$ is definitely a 4-torsion point in $E(\mathbb{F}_p)$ for some $0 \leq i \leq e - 1$. The above discussion leads to us the following algorithm which is implicitly the method presented by Peralta [10].

Algorithm 5 .

Input: An odd prime number p and a quadratic residue a modulo p.
$E : y^2 = x(x + a)^2$ over the field $\mathbb{F}_p$. Assume $\#E(\mathbb{F}_p) = 2^e m$ such that $(m, 2) = 1$.
Output: $\sqrt{a} \mod p$.
(1) Choose a point $R = (t^2, t(t^2 + a))$ on E such that $Q = mR \neq \infty$ and $Q = mR \neq (0, 0)$.
(2) $2^i Q = (z, w)$ must be a 4-torsion (i.e $z = a$) for some $i = 0, 1, \cdots e - 1$.
(3) Compute $w/2a$ which gives $\sqrt{a} \mod p$.

Theorem 3.3 *Let E be as above and R be a random point on it. The probability of reaching a point of order 4 via computing $2^i mR$ for some $0 \leq i < e - 1$ is $1 - \frac{1}{2^{e-1}}$.*

Proof Since the attached abelian group, $Jac(E)$, to E is cyclic group of order $p - 1$, where $p - 1 = 2^e m$, $Jac(E) \cong \mathbb{Z}_{2^e} \oplus \mathbb{Z}_m$. The probability that a random point's order is not divisible by 4 is $2/2^e = 1/2^{e-1}$ and this completes the proof. □

The same probability of success reveals that the curve is the actual singular cubic curve that Peralta's algorithm employed. The following lemma indicates an additional step can improve the success rate of Peralta's algorithm.

Lemma 3.4 *Let E be as above over the field $\mathbb{F}_p$ and $R = (\ell^2, \ell(\ell^2 + a))$ be a random point on $E(\mathbb{F}_p)$. If $mR = (0, 0)$ then the point $Q = \left(\frac{m+1}{2}\right)R$ has order divisible by 4 where again $p - 1 = m2^e$ for some positive integer e and odd integer m.*

Proof The point $Q = \left(\frac{m+1}{2}\right)R$ is in the group $E(\mathbb{F}_p)$ where the order of $E(\mathbb{F}_p) = m2^e$ with odd integer m and positive integer e. It is given that $mR = (0, 0)$ where $(0, 0)$ is the only point $E(\mathbb{F}_p)$ of order 2. Lets look at:

$$4Q = 2mR + 2R = 2(0, 0) + 2R = \infty + 2R = 2R$$

Since we have $2mQ = mR = (0, 0)$, the order of Q is divisible by 4 and mQ has order exactly 4. □

Remark 3.5 Lemma 3.4 implies that the success rate of Algorithm 5 can easily be improved. In other words, unless one reaches $mR = \infty$ in step 1, one can still find a square root of a.

Because, in the case of $mR = (0, 0)$ then $Q = \left(\frac{m+1}{2}\right)R$ gives a point of order divisible by 4. This observation reduces the probability of failure to $1/2^e$ and increases the success rate to $1 - 1/2^e$ in a single trial. Considering the worst case where $e = 2$, we have at least $3/4$ chance to reach a square root in a single trial with a little modification in Algorithm 5.

Theorem 3.6 *Let E be as above and R be a random point on it. The probability of mR being a generator of the Sylow-2 subgroup of $E(\mathbb{F}_p)$ is 1/2.*

Proof Consider the group $\mathbb{Z}_p^* \cong \mathbb{Z}_{2^e} \oplus \mathbb{Z}_m$ and an element n of it where $p - 1 = m2^e$ for some odd integer m. The element n^m is a generator of the Sylow-2 subgroup of $\mathbb{Z}_p^*$ if it satisfies $n^{(p-1/2)} \equiv -1 \mod p$. Similarly, if a point R in the cyclic group $E(\mathbb{F}_p)$ satisfies $\left(\frac{p-1}{2}\right)R = (0, 0)$ then mR becomes a generator of the Sylow-2 subgroup of $E(\mathbb{F}_p)$. As the groups $\mathbb{Z}_p^*$ and $E(\mathbb{F}_p)$ are isomorphic, the probability of reaching a generator of the Sylow-2 subgroup of $E(\mathbb{F}_p)$ via computing mR for a random point R is 1/2. □

Once one finds a generator mR of Sylow-2 subgroup of $E(\mathbb{F}_p)$, one can easily reach a square root by computing a 4-torsion via $2^{e-2}mR$. In other words, like Tonelli–Shanks algorithm, finding a generator is sufficient to reach a desired square root in a deterministic way. However, Peralta's algorithm shows that this is not a necessary condition as one can reach a 4-torsion without finding a generator for the 2-Sylow subgroup of $E(\mathbb{F}_p)$.

3.3.2 Cipolla's algorithm with singular curves

In the same context, we now present a geometric analogue of Cipolla's algorithm.

Theorem 3.7 *Let a be a square in $\mathbb{F}_p$, $E : y^2 = x(x + a)^2$ be a singular curve over $\mathbb{F}_p$ and $P = (t^2, t(t^2 + a))$ be any point on E, where $t \neq 0$. If $t^2 + a$ is quadratic non-residue, then employing P with Algorithm 5 definitely returns a square root of a.*

Proof We first show that if $t^2 + a$ is a quadratic non-residue, then it is impossible to have $mP = \infty$. Suppose $mP = \infty$. Consider the point

$$Q = \frac{(m + 1)}{2} P.$$

Then we have

$$2Q = mP + P = P.$$

Let the x-coordinate of Q be b where $b \in \mathbb{F}_p$. Then the x-coordinates of $2Q$ would be $\frac{(a - b^2)^2}{(2b)^2}$ by (2.4). So, we should have

$$\frac{(a - b^2)^2}{(2b)^2} = t^2 \text{ and then } a - b^2 = \pm 2bt.$$

Since the discriminant of $b^2 \pm 2bt - a = 0$ is $\Delta = 4t^2 + 4a = 4(t^2 + a)$ and $t^2 + a$ is quadratic non-residue, it is impossible to have such b^2 such that $mP = \infty$.

If $mP = (0, 0)$ Lemma 3.4 then Remark 3.5 implies that $Q = \left(\frac{m+1}{2}\right)P$ has order divisible by 4 which means a modified version of Algorithm 5 returns a square root. □

The above theorem indicates that Cipolla's method also seeks a sufficient condition in Algorithm 5 similar to the Tonelli–Shanks method. Employing 4-torsion points on E returns a square root algorithm where each of three practical algorithms can be explained in the same context. Towards a deterministic and polynomial-time algorithm for square root computation in the finite field, it might be beneficial to work with other torsion points. For example, designing a singular curve where 5-torsion points return square root increases the success rate.

Example 3.8 We are going to find the square root of $a = 2$ modulo $p = 2017$ via a singular cubic analogue of each algorithm. We define the curve E by the equation $y^2 = x(x+2)^2$ over the finite field $\mathbb{F}_{2017}$. As

$$2017 - 1 = 2016 = 2^5 \cdot 63$$

$m = 63$ and $e = 5$.

Peralta's Algorithm We select a random point P on E, say $P = (1, 3)$. We compute

$$Q = mP = 63P = (2, 90).$$

Since Q is neither the identity nor two torsion point $(0, 0)$, the order of P is divisible by 4 which means that the order of $2^i Q$ is divisible by 4 for some $i = 0, \ldots, 3$. As $2Q = (0, 0)$, the point Q has order 4. Note that

$$Q = (a, 2a\sqrt{a}) = (2, 4\sqrt{2}) = (2, 90) \text{ which implies } \sqrt{2} = 1031.$$

Tonelli–Shanks Algorithm We search a point P such that

$$\left(\frac{p-1}{2}\right) P \neq \infty$$

The first random points $P = (1, 3)$ and $P = (25, 135)$ do not work. The third point $P = (289, 913)$ works as $1008P = (0, 0)$. That means that $mP = Q = 63P = (138, 258)$ is a generator of the Sylow-2 subgroup where

$$8Q = 8 \cdot 63P = (2, 1927)$$

is a point of order 4. Note that $(2, 1927) = (2, 4\sqrt{2}) = (2, 4 \cdot 986)$ which implies $\sqrt{2} \equiv 986 \equiv -1031 \mod 2017$

Cipolla Algorithm We first search a random point t such that $t^2 + a = t^2 + 2$ is a quadratic non-residue mod 2017. We have found such t on the second try where we set $t = 611$ and $P = (t^2, t(t^2+2)) = (176, 1857)$. The order of the point P must be divisible by 4. We compute $mP = (1379, 1791)$ then $2mP = (1553, 936)$, $4mP = (96, 384)$ and $8mP = (2, 90)$ where we get $4\sqrt{2} = 90$ and $\sqrt{2} = 1031$.

We implemented all algorithms in the same environment where we use a C++ library, PARI/GP [13], for operations involving large integers. The real time tests are conducted on the computer running Linux OS with an Intel i7-11370H processor and 32 GB main memory. The discrete logarithm problem in Tonelli–Shanks algorithm is handled via a naive brute force method. The first implementation of Tonelli–Shanks seeks a quadratic non-residue randomly in the field $\mathbb{F}_p$ for a prime integer p. On the other hand, we also implement Tonelli–Shanks with the quadratic reciprocity law to find a quadratic non-residue while searching only numbers less than 100. The following table briefly summarize the test results (Table 1).

Table 1 The tests are conducted for primes p where $p-1=2^e m$ and the time (in millisecond) is average of 1000 runs for each algorithm

Finite field size (size of p)	256-bit, e = 4	512-bit, e = 5	1024-bit, e = 8
Tonelli–Shanks	0.753	1.548	4.792
Tonelli–Shanks (Quadratic Reciprocity)	0.328	0.642	2.372
Cipolla	0.583	1.391	4.484
Peralta	0.407	0.720	2.188
Singular cubics	0.317	0.992	4.298

Acknowledgements We thank Andrew Sutherland and Rewievers for their corrections and suggestions to improve the quality of paper.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

References

1. Bach E.: Explicit bounds for primality testing and related problems. Math. Comput. **55**(191), 355–380 (1990).
2. Cipolla M.: Un metodo per la risoluzione della congruenza di secondo grado. Napoli Rend. **9**, 154–163 (1903).
3. Cohen H.: A Course in Computational Algebraic Number Theory. Springer, New York (2000).
4. Cohen H., Frey G.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman & Hall/CRC, Boca Raton (2005).
5. Müller S.: On strong lucas pseudoprimes. Comb. Gen. Algebra **10**, 237–249 (1998).
6. Nari K., Ozdemir E., Ozkirisci A.N.: Strong pseudo primes to base 2. Ramanujan J. **59**, 1323–1332 (2022).
7. Ozdemir E.: Computing square roots in finite fields. IEEE Trans. Inf. Theory **59**, 9 (2013).
8. Ozdemir E.: Factoring polynomials over finite field. Int. J. Number Theory **17**(07), 1517–1536 (2021).
9. Ozdemir E..: Curves and Their Applications to Factoring Polynomials, PhD Thesis (2009).
10. Peralta R.. C.: A simple and fast probabilistic algorithm for computing square roots modulo a prime number. IEEE Trans. Inf. Theory **32**(6,), 846–847 (1986).
11. Shanks D.: Class number, a theory of factorization, and genera. In: Proc. Symp Pure Maths. 20, AMS, Providence pp. 415–440 (1971).
12. Sutherland A.V.: Structure computation and discrete logarithms in finite abelian p-groups. Math. Comput. **80**, 477–500 (2011).
13. The PARI Group. PARI/GP version `2.13.4`, Univ. Bordeaux, 2022, http://pari.math.u-bordeaux.fr/.
14. Washington L.C.: Elliptic Curves: Number Theory and Cryptography, 2nd edn Chapman & Hall/CRC, Boca Raton (2008).