New Cube Root Algorithm Based on Third Order Linear Recurrence Relation in Finite Field

Gook Hwa Cho, Namhun Koo, Eunhye Ha, and Soonhak Kwon
Email: achimheasal@nate.com, komaton@skku.edu, grace.eh.ha@gmail.com, shkwon@skku.edu
Dept. of Mathematics, Sungkyunkwan University, Suwon, S. Korea

Abstract

In this paper, we present a new cube root algorithm in finite field $\mathbb{F}_q$ with q a power of prime, which extends the Cipolla-Lehmer type algorithms [4, 5]. Our cube root method is inspired by the work of Müller [8] on quadratic case. For given cubic residue $c \in \mathbb{F}_q$ with $q \equiv 1 \pmod 9$, we show that there is an irreducible polynomial $f(x) = x^3 - ax^2 + bx - 1$ with root $\alpha \in \mathbb{F}_{q^3}$ such that $Tr(\alpha^{\frac{q^2+q-2}{9}})$ is a cube root of c. Consequently we find an efficient cube root algorithm based on third order linear recurrence sequence arising from $f(x)$. Complexity estimation shows that our algorithm is better than previously proposed Cipolla-Lehmer type algorithms.

Keywords : finite field, cube root, linear recurrence relation, Tonelli-Shanks algorithm, Cipolla-Lehmer algorithm, Adleman-Manders-Miller algorithm

MSC 2010 Codes : 11T06, 11Y16, 68W40

1 Introduction

Finding r-th root in finite field $\mathbb{F}_q$ has many applications in computational number theory and in many other related topics. For square root computation, which is the simplest case among all r-th root extraction problems, there are two standard algorithms; the Tonelli-Shanks [1, 2] and the Cipolla-Lehmer [4, 5] algorithms. Due to the cumbersome extension field arithmetic needed in the Cipolla-Lehmer algorithm, one usually prefers the Tonelli-Shanks algorithm, and other related researches [8, 11, 12, 17, 18] exist.

There is also the Adleman-Manders-Miller [3] algorithm which is a straightforward generalization of the Tonelli-Shanks to the case of cube roots and r-th roots. Improving the Tonelli-Shanks or the Adleman-Manders-Miller algorithm via efficient computation of discrete logarithm in $\mathbb{F}_q^\times$ are reported in [13, 14]. However it should be mentioned that the worst case complexity of the Tonelli-Shanks is $O(\log^4 q)$ while the Cipolla-Lehmer can be executed in $O(\log^3 q)$. Other view points on r-th root extraction different from the Adleman-Manders-Miller and the Cipolla-Lehmer are suggested in [15, 16].

Müller [8] proposed a fast and simple square root algorithm which is a refinement of the Cipolla-Lehmer. Müller's idea is to use a special type of Lucas sequence with the polynomial $x^2 - Px + 1$ instead of the general $f(x) = x^2 - Px + Q$ required in the original Cipolla-Lehmer. The Cipolla-Lehmer algorithm is based on the simple idea that, when f is irreducible over $\mathbb{F}_q$, then a square root of Q is given by $\alpha^{\frac{q+1}{2}}$ with $f(\alpha) = 0$, so the arithmetic in $\mathbb{F}_{q^2}$ or the recurrence relation technique can be used to find a square root. However it is not clear to find an obvious (or natural) candidate for square root of Q when one use the polynomial $x^2 - Px + 1$.

Müller successfully find a suitable P for given Q so that the square roots of Q can be found efficiently.

For cubic case, it seems that there is not so many works regarding Cipolla-Lehmer algorithm other than that of Nishihara [9] for constructing cubic irreducible polynomials to speed up the Cipolla-Lehmer method.

In this paper, we propose a new cube root algorithm over $\mathbb{F}_q$ with prime power $q \equiv 1 \pmod 9$ by extending Müller's result on quadratic case. For the case $q \not\equiv 1 \pmod 9$, we have a simple closed formula (cost of one exponentiation in $\mathbb{F}_q$) for cube root. We show that, for given cubic residue $c \in \mathbb{F}_q$, we can efficiently construct an irreducible polynomial $f(x) = x^3 - ax^2 + bx - 1$ with root $\alpha \in \mathbb{F}_{q^3}$ such that $Tr(\alpha^{\frac{q^2+q-2}{9}})$ is a cube root of c, where $Tr(\beta) = Tr_{\mathbb{F}_{q^3}/\mathbb{F}_q}(\beta) = \beta + \beta^q + \beta^{q^2}$. Such element $Tr(\alpha^{\frac{q^2+q-2}{9}})$ can be computed via the recurrence relation found in [7].

The remainder of this paper is organized as follows: In Section 2, we introduce the root extraction algorithms in $\mathbb{F}_q$. In Section 3, we describe the third order linear recurrence sequences. In Section 4, we propose a new cube root algorithm based on the third order linear recurrence relation. In Section 5, we discuss the complexity estimation of the proposed algorithm and the implementation results. Finally, in Section 6, we give a conclusion.

2 Root Extraction Algorithms in $\mathbb{F}_q$

In this section, we introduce two standard algorithms for computing cube roots in finite field, that is, the Tonelli-Shanks [1, 2], the Adleman-Manders-Miller [3] algorithms and the Cipolla-Lehmer algorithm [4, 5]. Also, Müller's square root algorithm [8] will be briefly sketched.

2.1 Tonelli-Shanks and Adleman-Manders-Miller algorithm

The Tonelli-Shanks algorithm [1, 2] or the Adleman-Manders-Miller algorithm [3] is described in Table 1. Its complexity is $O(\nu_3(q-1)\log^3 q)$, where $\nu_3(q-1)$ denotes the largest non-negative power ν satisfying $3^\nu | q-1$. The Tonelli-Shanks algorithm has the complexity $O(\log^3 q)$ when $\nu_3(q-1)$ is small while has the worst complexity $O(\log^4 q)$ when $\nu_3(q-1) \approx \log_3 q$.

2.2 Cipolla-Lehmer algorithm

The Cipolla-Lehmer algorithm [4, 5] is described in Table 2. Its complexity is $O(\log^3 q)$, which does not depend on $\nu = \nu_3(q-1)$ unlike the case of the Tonelli-Shanks. However, for small $\nu = \nu_3(q-1)$, the Tonelli-Shanks algorithm runs faster than the Cipolla-Lehmer because the Cipolla-Lehmer relies on the relatively heavy extension field arithmetic in $\mathbb{F}_{q^3}$. Hence the refinements of the Cipolla-Lehmer is desirable.

To find a cube root of $c \in \mathbb{F}_q$ with $q \equiv 1 \pmod 3$, the Cipolla-Lehmer algorithm needs an irreducible cubic polynomial $f(x) = x^3 - ax^2 + bx - c$ with constant term $-c$. Letting $\alpha \in \mathbb{F}_{q^3}$ be a root of f, we get $\alpha^{1+q+q^2} = c$ so that $\alpha^{\frac{1+q+q^2}{3}}$ is a cube root of c. The irreducibility testing of f and the exponentiation $\alpha^{\frac{1+q+q^2}{3}}$ in $\mathbb{F}_{q^3}$ (or computing $x^{\frac{1+q+q^2}{3}} \pmod{f(x)}$) need many multiplications in $\mathbb{F}_q$, and the number of such multiplications depends on the coefficients of f. Nishihara [9] showed that a speed-up can be achieved if one use an irreducible cubic trinomials with constant term $-c$, and that Dickson's method [6] (Table 3) can be used to test the irreducibility of the trinomial. Dickson's irreducibility criterion for $f(x) = x^3 + bx - c$ says that $f(x)$ is irreducible over $\mathbb{F}_q$ if and only if the following two conditions are satisfied:

Table 1: Adleman-Manders-Miller cube root algorithm

Input: A cubic residue a in $\mathbb{F}_q$ with odd characteristic Output: A cube root of a
Step 1: Let $q-1 = 3^s t$ with $t = 3l \pm 1$
Step 2: Select a cubic non-residue b in $\mathbb{F}_q$ $c \leftarrow b^t$ $c' \leftarrow c^{3^{s-1}}$
Step 3: (Computation of the cube root of $(a^t)^{-1}$) $h \leftarrow 1$, $r \leftarrow a^t$ **for** $i = 1$ to $s-1$ $d \leftarrow r^{3^{s-i-1}}$ **if** $d = 1$, **then** $k \leftarrow 0$ **else if** $d = c'$, **then** $k \leftarrow 2$ **else** $k \leftarrow 1$ $h \leftarrow h \cdot c^k$, $r \leftarrow r \cdot (c^3)^k$ $c \leftarrow c^3$ **end for**
Step 4: $r \leftarrow a^l \cdot h$ **if** $t = 3l+1$, **then** $r \leftarrow r^{-1}$ Return r

Table 2: Cipolla-Lehmer cube root algorithm

Input: A cubic residue c in $\mathbb{F}_q$ Output: A cube root of c
Step 1: Choose a, b randomly in $\mathbb{F}_q$
Step 2: $f(x) \leftarrow x^3 - ax^2 + bx - c$ **if** f is reducible, **then** go to Step 1
Step 3: Return $x^{\frac{q^2+q+1}{3}} \pmod{f(x)}$

1. $D = -(4b^3 + 27c^2)$ is nonzero quadratic residue in $\mathbb{F}_q$
2. $\frac{1}{2}(c + 3^{-2}\sqrt{-3D})$ is a cubic non-residue in $\mathbb{F}_q$

For the cube root computation $x^{\frac{q^2+q+1}{3}} \pmod{f(x)}$ of c, this method is more efficient than the one using random cubic polynomial because $f(x)$ has a low hamming weight (i.e., trinomial).

Table 3: Dickson's irreducibility testing of cubic polynomial

Input: A cubic residue c in $\mathbb{F}_q$ Output: An irreducible cubic polynomial with constant term $-c$
Step 1: Choose b randomly in $\mathbb{F}_q$ $f(x) \leftarrow x^3 + bx - c, \quad D(f) \leftarrow -(4b^3 + 27c^2)$
Step 2: **if** $D(f)$ is zero or a quadratic non-residue, **then** go to Step 1 **else** $\alpha \leftarrow \frac{1}{2}(c + 3^{-2}\sqrt{-3D(f)})$ (i.e., α: a root of $x^2 - cx - 3^{-3}b^3 = 0$)
Step 3: **if** α is a cubic non-residue in $\mathbb{F}_q$, **then** return $f(x)$ **else** go to Step 1

2.3 Müller's square root algorithm with Lucas sequences

For given irreducible quadratic $f(x) = x^2 - Px + Q \in \mathbb{F}_q[x]$ with roots α and β, one has the corresponding Lucas sequence $s_k = \alpha^k + \beta^k$. Computing s_k via the relation $s_k = Ps_{k-1} - Qs_{k-2}$ can be simple if one lets $Q = 1$, that is, if $f(x) = x^2 - Px + 1$.

Let Q be a square in $\mathbb{F}_q$. Assume that $q \equiv 1 \pmod 4$ and $f(x) = x^2 - Px + 1$ with $P = Q - 2$ is irreducible over $\mathbb{F}_q$. Letting α, α^{-1} be roots of $f(x)$, Müller [8] found a square root of Q as

$$\begin{aligned} s^2_{\frac{q-1}{4}} &= (\alpha^{\frac{q-1}{4}} + \alpha^{-\frac{q-1}{4}})^2 \\ &= \alpha^{-1}\alpha^{\frac{q+1}{2}} + \alpha\alpha^{-\frac{q+1}{2}} + 2 \\ &= \alpha^{-1} + \alpha + 2 = P + 2 = Q. \end{aligned}$$

For detailed explanation, see [8]. The cost of computing $s_{\frac{q-1}{4}}$ is small because it comes from $x^2 - Px + 1$ not from $x^2 - Px + Q$. In fact, it is shown [8] that one needs $2\log_2 q$ multiplications in $\mathbb{F}_q$ to compute $s_{\frac{q-1}{4}}$. Our purpose is to generalize and extend Müller's idea [8] to the cubic case, and it will be shown in the next two sections.

3 The Third Order Linear Recurrence Sequences

Let $f(x) = x^3 - ax^2 + bx - c$ $(a, b, c \in \mathbb{F}_q)$ be irreducible over $\mathbb{F}_q$. A third-order linear recurrence sequence s_k with characteristic polynomial $f(x)$ is defined as

$$s_k = as_{k-1} - bs_{k-2} + cs_{k-3}, \qquad k \geq 3.$$

If s_k has the initial state $s_0 = 3, s_1 = a$, and $s_2 = a^2 - 2b$, then s_k is called the characteristic sequence generated by $f(x)$. Letting $f(\alpha) = 0$, the characteristic sequence s_k can be expressed as

$$s_k = Tr(\alpha^k) = \alpha^k + \alpha^{kq} + \alpha^{kq^2}.$$

When it is needed to emphasize that the characteristic sequence s_k comes from the polynomial f, we denote such s_k using various notations such as $s_k(f), s_k(a, b, c)$, or $s_k(\alpha)$. It is well-known [7] that the sequence s_k satisfies

1. $s_{2n} = s_n^2 - 2c^n s_{-n}$,

2. $s_{n+m} = s_n s_m - c^m s_{n-m} s_{-m} + c^m s_{n-2m}$

As in the case of second order recurrence relation, the exponentiation c^n gives extra burden to the computation of s_k. Therefore we are mainly interested in the polynomial $f(x)$ with $c = 1$, i.e. $f(x) = x^3 - ax^2 + bx - 1$. Note that, to apply the Cipolla-Lehmer for the computation of the cube root of c, one has to use the polynomial $x^3 - ax^2 + bx - c$, so letting the constant term -1 makes it impossible to use the Cipolla-Lehmer. However, as is done by Müller for the quadratic case, an wise choice of f gives a way to find the cube root of $c \in \mathbb{F}_q$ as will be shown in the next section.

From now on, we will consider the characteristic sequence s_k which comes from the irreducible cubic $f(x) = x^3 - ax^2 + bx - 1$. Then the sequence s_k has much simpler relation;

A. $s_{2n} = s_n^2 - 2s_{-n}$,

B. $s_{n+m} = s_n s_m - s_{n-m} s_{-m} + s_{n-2m}$

Now Let $k = \sum_{i=0}^{r} k_i 2^{r-i}$ be a binary representation of k, and let the partial sum z_j be defined as $z_0 = k_0 = 1, z_j = 2z_{j-1} + k_j$, $j = 1, 2, \cdots, r$. Then one has $z_r = k$, and s_k can be computed using the above two relations A and B as follows (See [7] for more explanation):

If $k_j = 0$, then $s_{z_j} = s_{2z_{j-1}}$ so let

1. $s_{z_j - 1} = s_{z_{j-1}} s_{z_{j-1}-1} - b s_{-z_{j-1}} + s_{-(z_{j-1}+1)}$

2. $s_{z_j} = s_{z_{j-1}}^2 - 2s_{-z_{j-1}}$

3. $s_{z_j+1} = s_{z_{j-1}} s_{z_{j-1}+1} - a s_{-z_{j-1}} + s_{-(z_{j-1}-1)}$

If $k_j = 1$, then $s_{z_j} = s_{2z_{j-1}+1}$ so let

1. $s_{z_j - 1} = s_{z_{j-1}}^2 - 2s_{-z_{j-1}}$

2. $s_{z_j} = s_{z_{j-1}} s_{z_{j-1}+1} - a s_{-z_{j-1}} + s_{-(z_{j-1}-1)}$

3. $s_{z_j+1} = s_{z_{j-1}+1}^2 - 2s_{-(z_{j-1}+1)}$

An algorithm realizing the above relations is given in Table 5. The complexity of computing both of s_k and s_{-k} is $9 \log_2 k$ multiplications in $\mathbb{F}_q$ on average. Also when a is a small integer as will be shown in the next section, the complexity is further reduced to $7 \log_2 k$ multiplications in $\mathbb{F}_q$. Moreover if one considers the cost using both squarings and multiplications, then the case $k_j = 1$ has the cost of 3 squarings using Karatsuba type technique (i.e., $2s_{z_{j-1}} s_{z_{j-1}+1} = (s_{z_{j-1}} + s_{z_{j-1}+1})^2 - s_{z_{j-1}}^2 - s_{z_{j-1}+1}^2$). Therefore the average cost is further reduced to $4 \log_2 k$ multiplications and $3 \log_2 k$ squarings in $\mathbb{F}_q$. Note that our method of using the recurrence relation is much more efficient than the method of directly computing $x^{\frac{1+q+q^2}{3}} \pmod{f(x)}$, since computing $(b_2 x^2 + b_1 x + b_0)^2 \pmod{f(x)}$ already requires at least 12 multiplications in $\mathbb{F}_q$.

4 New Cube Root Algorithm

Let b be in $\mathbb{F}_q$ and let $f(x) = x^3 - 3x^2 + bx - 1$ be irreducible over $\mathbb{F}_q$ with $q \equiv 1 \pmod 3$. Suppose $f(\alpha) = 0$. Then the norm of f (the product of all the conjugates of α) is

$$\alpha^{1+q+q^2} = 1.$$

A classical result of Hilbert Theorem 90 [19] or direct calculation over the field extension $\mathbb{F}_{q^3}/\mathbb{F}_q$ says that there exists $\beta \in \mathbb{F}_{q^3}$ such that $\beta^3 = \alpha$. More precisely, using the property $\alpha^{1+q+q^2} = 1$, one can show that

$$\alpha(1 + \alpha + \alpha^{1+q})^q = 1 + \alpha + \alpha^{1+q}.$$

Therefore putting $\beta = (1 + \alpha + \alpha^{1+q})^{\frac{1-q}{3}}$, we get

$$\beta^3 = (1 + \alpha + \alpha^{1+q})^{1-q} = \alpha. \tag{1}$$

Letting $h(x) = x^3 + (b-3)x - (b-3)$, it is easy to check $h(1-\alpha) = 0$ since $Tr(\alpha) = 3$. In fact, one has $h(1-x) = -f(x)$. Therefore, the irreducibility of f implies the irreducibility of h and vice versa.

Theorem 1. *Assuming $f(\alpha) = 0$ and $q \equiv 1 \pmod 3$, we have $\alpha^{\frac{1+q+q^2}{3}} = 1$.*

Proof. Since $h(x) = x^3 + (b-3)x - (b-3)$ is also irreducible over $\mathbb{F}_q$,

$$(1-\alpha)^{1+q+q^2} = (b-3). \tag{2}$$

On the other hand, from $0 = h(1-\alpha) = (1-\alpha)^3 + (b-3)(1-\alpha) - (b-3)$, we get

$$(1-\alpha)^3 = (b-3)\alpha \tag{3}$$

By taking $\frac{1+q+q^2}{3}$-th power to both sides of the above expression,

$$(1-\alpha)^{1+q+q^2} = (b-3)^{\frac{1+q+q^2}{3}} \alpha^{\frac{1+q+q^2}{3}} \tag{4}$$

Comparing two expressions (2) and (4), we get

$$\alpha^{\frac{1+q+q^2}{3}} = (b-3)^{-\frac{q^2+q-2}{3}} = (b-3)^{-\frac{(q-1)(q+2)}{3}} = 1 \tag{5}$$

since $q \equiv 1 \pmod 3$ and $b - 3 \in \mathbb{F}_q$. □

Now letting $g(x) = x^3 - a'x^2 + b'x - c'$ $(a', b', c' \in \mathbb{F}_q)$ be the irreducible polynomial of β over $\mathbb{F}_q$, we get $c' = 1$ from the above theorem because

$$c' = \beta^{1+q+q^2} = \alpha^{\frac{1+q+q^2}{3}} = 1. \tag{6}$$

Lemma 1. $b'^3 = b + 3b'a' - 3$

Proof. Using the following simple identity

$$(A+B+C)^3 = A^3+B^3+C^3+3(A+B+C)(AB+BC+CA)-3ABC$$

with $A=\beta^{1+q}, B=\beta^{q+q^2}, C=\beta^{1+q^2}$, we get

$$\begin{aligned}&(\beta^{1+q}+\beta^{q+q^2}+\beta^{1+q^2})^3 = \\ &\alpha^{1+q}+\alpha^{q+q^2}+\alpha^{1+q^2}+3(\beta^{1+q}+\beta^{q+q^2}+\beta^{1+q^2})(\beta+\beta^q+\beta^{q^2})-3\end{aligned} \tag{7}$$

which can be rewritten as

$$b'^3 = b+3b'a'-3. \tag{8}$$

□

Corollary 1. $s_{\frac{q^2+q-2}{3}}(\beta)^3 = s_{q+1}(\beta)^3 = b'^3$

Proof. Letting $\beta^{\frac{1+q+q^2}{3}} = \omega$, we have $\omega^3 = \beta^{1+q+q^2} = 1$ and $\omega^q = \omega$. Therefore

$$\begin{aligned}s_{\frac{q^2+q-2}{3}}(\beta)^3 &= Tr(\beta^{\frac{q^2+q-2}{3}})^3 = (\beta^{\frac{q^2+q-2}{3}}+\beta^{q\frac{q^2+q-2}{3}}+\beta^{q^2\frac{q^2+q-2}{3}})^3 \\ &= (\omega\beta^{-1}+\omega^q\beta^{-q}+\omega^{q^2}\beta^{-q^2})^3 \\ &= (\beta^{q+q^2}+\beta^{1+q^2}+\beta^{1+q})^3 = s_{q+1}(\beta)^3 = b'^3.\end{aligned} \tag{9}$$

□

Corollary 2. *If* $q \equiv 1 \pmod 9$, *then* $s_{\frac{q^2+q-2}{9}}(\alpha)^3 = b'^3$.

Proof.

$$\begin{aligned}s_{\frac{q^2+q-2}{9}}(\alpha)^3 &= Tr(\alpha^{\frac{q^2+q-2}{9}})^3 = Tr((\beta^3)^{\frac{q^2+q-2}{9}})^3 \\ &= Tr(\beta^{\frac{q^2+q-2}{3}})^3 = s_{\frac{q^2+q-2}{3}}(\beta)^3 = b'^3\end{aligned} \tag{10}$$

□

If $b-3$ is a cubic residue in $\mathbb{F}_q$, one can also show $a'=0$ as follows.

Corollary 3. *Assume* $b-3$ *is a nonzero cube in* $\mathbb{F}_q$. *Then one has* $\beta+\beta^q+\beta^{q^2}=0$.

Proof. Since $\alpha = \beta^3 \in \mathbb{F}_{q^3}$, we may rewrite the equation (3) as

$$(1-\alpha)^3 = (b-3)\beta^3 \tag{11}$$

Assume $b-3=u^3$ for some u in $\mathbb{F}_q$. Then from $(1-\alpha)^3 = u^3\beta^3$, we get

$$(1-\alpha) = \omega_0 u\beta \tag{12}$$

for some cube root of unity ω_0 in $\mathbb{F}_q$. Therefore we get

$$\begin{aligned}\omega_0 u Tr(\beta) &= Tr(\omega_0 u\beta) = Tr(1-\alpha) \\ &= (1-\alpha)+(1-\alpha)^q+(1-\alpha)^{q^2} \\ &= 3-(\alpha+\alpha^q+\alpha^{q^2}) = 0.\end{aligned} \tag{13}$$

Since $u \neq 0$, we get $a' = Tr(\beta) = 0$. □

Table 4: New cube root algorithm for $\mathbb{F}_q$ with $q \equiv 1 \pmod 9$

Input: A cubic residue c in $\mathbb{F}_q$ Output: s satisfying $s^3 = c$
Step 1: $t \leftarrow 1$, $b \leftarrow ct^3 + 3$, $f(x) \leftarrow x^3 - 3x^2 + bx - 1$
Step 2: **while** $f(x)$ is reducible over $\mathbb{F}_q$ Choose random $t \in \mathbb{F}_q$ $b \leftarrow ct^3 + 3$, $f(x) \leftarrow x^3 - 3x^2 + bx - 1$ **end while**
Step 3: $s \leftarrow s_{\frac{q^2+q-2}{9}}(f) \cdot t^{-1}$

Table 5: Algorithm for computing s_m

Input: $b \in \mathbb{F}_q$ and $m = \sum_{j=0}^{l} k_j 2^j$ Output: s_m
Step 1: (Initialize) $d_0 \leftarrow 3,\ d_1 \leftarrow 3,\ d_2 \leftarrow 9 - 2b$ $a_0 \leftarrow 3,\ a_1 \leftarrow b,\ a_2 \leftarrow b^2 - 6$

Step 2: (Iterate on j)
for j from $l-1$ down to 1 **do**
if $k_j = 0$ **then** $d_0 \leftarrow a_2 - ba_1 + d_0d_1,\ d_1 \leftarrow d_1^2 - 2a_1,\ d_2 \leftarrow a_0 - 3a_1 + d_1d_2$
$a_0 \leftarrow d_2 - 3d_1 + a_0a_1,\ a_1 \leftarrow a_1^2 - 2d_1,\ a_2 \leftarrow d_0 - bd_1 + a_1a_2$
if $k_j = 1$ **then** $d_0 \leftarrow d_1^2 - 2a_1,\ d_1 \leftarrow a_0 - ba_1 + d_1d_2,\ d_2 \leftarrow d_2^2 - 2a_2$
$a_0 \leftarrow a_1^2 - 2d_1,\ a_1 \leftarrow d_0 - 3d_1 + a_1a_2,\ a_2 \leftarrow a_2^2 - 2d_2$
end for

Step 3: (Evaluate) $w_1 \leftarrow a_0 - ba_1 + d_1d_2,\ w_2 \leftarrow d_1^2 - 2a_1$ **if** $k_0 = 1$ **then** return w_1, **else** return w_2

Finally, combining Lemma 1 and Corollary 2,3, we have the following theorem.

Theorem 2. *Suppose that $q \equiv 1 \pmod 9$ and $f(x) = x^3 - 3x^2 + bx - 1$ is an irreducible polynomial over $\mathbb{F}_q$ with $f(\alpha) = 0$. Assume $b - 3$ is a cubic residue in $\mathbb{F}_q$. Then $s_{\frac{q^2+q-2}{9}}(\alpha)^3 = b - 3$.*

Proof. We have

$$s_{\frac{q^2+q-2}{9}}(\alpha)^3 = b'^3 = b + 3b'a' - 3 = b - 3,$$

where the first equality is the Corollary 2, the second equality is the Lemma 1, and the third equality holds because of the Corollary 3. □

Now using the polynomial $f(x) = x^3 - 3x^2 + bx - 1$, we can find a cube root for given cube c in $\mathbb{F}_q$. For given cubic residue $c \in \mathbb{F}_q$, define $b = c + 3$. If $f(x)$ with given coefficient b is irreducible, then $s_{\frac{q^2+q-2}{9}}(f)$ is a cube root of c. That is,

$$s_{\frac{q^2+q-2}{9}}(f)^3 = b - 3 = c.$$

If the given f is not irreducible over $\mathbb{F}_q$, then we may twist c by random $t \in \mathbb{F}_q$ until we get irreducible f with $b = ct^3 + 3$. Then

$$s_{\frac{q^2+q-2}{9}}(f)^3 = b - 3 = ct^3,$$

which implies $t^{-1}s_{\frac{q^2+q-2}{9}}(f)$ is a cube root of c. Table 4 shows our proposed cube root algorithm and Table 5 explains the algorithm for computing s_m.

Our proposed algorithm works only for those primes $q \equiv 1 \pmod 9$. However, when $q \not\equiv 1 \pmod 9$, one has a much simpler formula for the cube root of c as follows; When $q \equiv 2 \pmod 3$, a cube root of c is given as $c^{\frac{2q-1}{3}}$. When $q \equiv 4 \pmod 9$, a cube root of cubic residue c is given by $c^{\frac{2q+1}{9}}$. When $q \equiv 7 \pmod 9$, a cube root of cubic residue c is given by $c^{\frac{q+2}{9}}$.

Thus the computational cost of finding cube root of c when $q \not\equiv 1 \pmod 9$ is just one exponentiation in $\mathbb{F}_q$. In fact, this kind of approach can be generalized following the manner of Atkin [11], Kong et al. [12] and Müller [8] for quadratic case. Atkin showed that the cost of finding a square root of $c \in \mathbb{F}_q$ is just one exponentiation when $q \equiv 5 \pmod 8$, and Müller extended Atkin's result by showing that the cost is two exponentiations when $q \equiv 9 \pmod{16}$. However, our method is slightly different from [8, 11, 12] and is rather a simplified version of the Tonelli-Shanks in the sense that we need only one exponentiation even in the case of $q \equiv 9 \pmod{16}$ by fixing primitive 8-th root of unity in $\mathbb{F}_q$. We state the result for the case of cube root computation.

Proposition 1. *Suppose that $q \equiv 1 \pmod 9$ but $q \not\equiv 1 \pmod{27}$, and suppose that ξ is a primitive 9-th root of unity in $\mathbb{F}_q$. Then one can find a cube root of $c \in \mathbb{F}_q$ with cost of one exponentiation in $\mathbb{F}_q$.*

Proof. From the condition on q, we have either $q \equiv 10 \pmod{27}$ or $q \equiv 19 \pmod{27}$. Thus we may write $q \equiv 9\epsilon + 1 \pmod{27}$ where $\epsilon = 1$ or 2. Now define

$$b = \left(c^{3-\epsilon}\right)^{\frac{q-9\epsilon-1}{27}}, \quad \text{and } \zeta = b^3c^2. \tag{14}$$

Since $\zeta = b^3c^2 = ((c^{3-\epsilon})^{\frac{q-9\epsilon-1}{27}})^3c^2 = c^{\frac{(3-\epsilon)(q-9\epsilon-1)+18}{9}} = (c^{(3-\epsilon)})^{\frac{q-1}{9}}$ with $\epsilon = 1, 2$ and c is a cubic residue in $\mathbb{F}_q$, we get $\zeta^3 = 1$. Therefore, we have either $\zeta = 1$ or $\xi^3 = \zeta^j$ for some $j \in \{1, 2\}$. Then a cube root of c is given by bc (when $\zeta = 1$) and $\xi^{3-j}bc$ (when $\xi^3 = \zeta^j$) because

$$(bc)^3 = b^3c^2c = \zeta c = c \qquad (\zeta = 1), \tag{15}$$

$$(\xi^{3-j}bc)^3 = \zeta^{j(3-j)}b^3c^2c = \zeta^{j(3-j)+1}c = c \quad (\xi^3 = \zeta^j), \tag{16}$$

where $\zeta^{j(3-j)+1} = \zeta^3 = 1$ if $j = 1, 2$. The computational cost (other than several multiplications) is just one exponentiation needed for computing b. □

Table 6: Running time (in seconds) for cube root computation with $p \approx 2^{3000}$

$\nu_3(p-1)$	100	300	500	700	1000	1400	1800
Adleman et al.	2.12	6.30	14.76	27.58	53.80	105.22	172.76
Cipolla-Lehmer	6.78	6.36	6.21	6.36	6.25	6.27	6.87
Proposed Alg.	5.24	5.09	5.02	4.89	4.93	5.07	4.93

Table 7: Running time (in seconds) for cube root computation with $p \approx 2^{4000}$

$\nu_3(p-1)$	100	300	500	700	1000	1500	2000	2500
Adleman et al.	2.96	9.14	21.67	40.36	80.78	177.89	316.84	486.99
Cipolla-Lehmer	9.78	9.78	9.97	10.04	9.97	9.95	9.97	9.84
Proposed Alg.	7.63	7.61	7.85	7.69	7.75	7.66	7.75	7.69

Table 8: Running time (in seconds) for cube root computation with $p \approx 2^{5000}$

$\nu_3(p-1)$	100	300	500	700	1000	1500	2000	2500	3000
Adleman et al.	4.02	12.37	29.72	55.35	109.12	241.71	430.45	671.40	989.10
Cipolla-Lehmer	15.54	15.07	15.07	15.32	15.04	15.01	14.93	15.08	15.07
Proposed Alg.	11.92	11.62	11.59	11.70	11.53	11.46	11.50	11.76	11.53

Example: The above result says that, when $q \equiv 10 \pmod{27}$, a cube root of c is given as

1. bc when $\zeta = 1$, because $(bc)^3 = (b^3c^2)c = \zeta c = c$
2. $\xi^2 bc$ when $\xi^3 = \zeta$, because $(\xi^2 bc)^3 = \zeta^2(b^3c^2)c = \zeta^2\zeta c = c$
3. ξbc when $\xi^3 = \zeta^2$, because $(\xi bc)^3 = \zeta^2(b^3c^2)c = \zeta\zeta^2 c = c$

Remark: Using similar technique, one can actually compute a cube root in $\mathbb{F}_q$ with just one exponentiation when $q \equiv 1 \pmod{3^s}$ and $q \not\equiv 1 \pmod{3^{s+1}}$ for small s. However, this method suddenly loses its merit when s gets larger because the the number cases we consider increases exponentially.

5 Complexity Estimation

A randomly selected monic polynomial over $\mathbb{F}_q$ of degree 3 with nonzero constant term is irreducible with probability $\frac{1}{3}$ (For an explanation, see [21]). Even if our choice of f is not really random, experimental evidence implies that one third of such f is irreducible.

As is mentioned at the end of Section 3, the cost of computing $s_{\frac{q^2+q-2}{9}}$ needs $7\log_2 \frac{q^2+q-2}{9} \approx 14\log_2 q$ $\mathbb{F}_q$-multiplications. The cost of irreducibility testing, using Dickson's formula in Table 3, needs essentially $2\log_2 q$ $\mathbb{F}_q$-multiplications, which is the cost of square root computation

via Müller's recurrence relation in step 2 of Dickson's algorithm. Step 3 of Dickson's algorithm asks one to find the cubic residuosity of certain $a \in \mathbb{F}_q$. An obvious way of determining cubic residuosity is computing $a^{\frac{q-1}{3}}$ whose cost is one exponentiation. However, similarly as in the case of quadratic residuosity, one can use the cubic reciprocity law and Euclidean algorithm in the ring $\mathbb{Z} + \mathbb{Z}\frac{-1+\sqrt{-3}}{2}$ to reduce the cost to constant number of multiplications. Detailed explanation is given by Damgård and Frandsen [10].

Therefore the total cost of our algorithm including irreducibility testing is $(2 \cdot 3+14)\log_2 q = 20\log_2 q$ multiplications in $\mathbb{F}_q$. Tables 6,7,8 show the comparison of the implementation results with Maple of the 3 algorithms; the Adleman-Manders-Miller algorithm, the Cipolla-Lehmer algorithm, and the proposed algorithm (in Table 4). The implementation was performed on Pentium(R) Dual-Core 2.70GHz with 2GB memory.

For convenience, we used prime fields $\mathbb{F}_p$ with three different size of primes p: $3000, 4000, 5000$ bits. Average timings for 5 different inputs $c \in \mathbb{F}_p$ are computed for those cases $\nu_3(p-1) = 100, 300, 500, \cdots$, etc. As one can see in the tables, the timings of Adleman-Manders-Miller increase drastically as $\nu_3(p-1)$ becomes larger, while the timings of the Cipolla-Lehmer and our algorithm are independent of $\nu_3(p-1)$. Also, the tables show that our proposed algorithm is consistently faster than the Cipolla-Lehmer. For example, when $p \approx 2^{5000}$, the average timing of the Cipolla-Lehmer is 15.13 (seconds) which are 30% slower than the average timing 11.62 (seconds) of our algorithm.

6 Conclusions and Future Works

We proposed a new cube root algorithm in $\mathbb{F}_q$ using a linear recurrence relation arising from a cubic polynomial with constant term -1, which is an improvement over the original Cipolla-Lehmer algorithm. The related linear recurrence is easy to compute and has low computational complexity. Complexity estimation shows that the proposed algorithm is better than the Adleman-Manders-Miller algorithm when $\nu_3(q-1)$ is sufficiently large. Our idea can be generalized to the case of r-th root extraction. We obtained a closed formula for r-th root for any positive integer r. More precisely, for given r-th power $c \in \mathbb{F}_q$, we showed that there exists $\alpha \in \mathbb{F}_{q^r}$ such that $Tr\left(\alpha^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}\right)^r = c$ where $Tr(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{r-1}}$ and α is a root of irreducible $f(x)$, where $f(x) = (x-1)^r + (b-r)x$ if $r = odd$ and $f(x) = (x+1)^r - (b+r)x$ if $r = even$. Bottleneck of our approach is efficient "double and add" formula for r-th order linear recurrence sequence, which is more or less similar to the one in Section 3. Other than the cases for $r = 2, 3$, it seems difficult to find an efficient "double and add" formula for general r and further study is needed to settle this problem.

Acknowledgement:
The preliminary version of this paper was presented at *10'th Algorithmic Number Theory Symposium* (ANTS X) Poster Session, July 9–13, 2012. No proceeding will be published for the poster session.

References

[1] D. Shanks, *Five number-theoretic algorithms*, Proc. 2nd Manitoba Conf. Number. Math., Manitoba, Canada, pp. 51-70, 1972

[2] A. Tonelli, *Bemerkung über die Auflösung quadratischer Congruenzen*, Göttinger Nachrichten, pp. 344-346, 1891

[3] L. Adleman, K. Manders and G. Miller, *On taking roots in finite fields*, Proc. 18th IEEE Symposium on Foundations on Computer Science (FOCS), pp. 175-177, 1977

[4] M. Cipolla, *Un metodo per la risolutione della congruenza di secondo grado*, Rendiconto dell'Accademia Scienze Fisiche e Matematiche, Napoli, Ser.3,Vol. IX, pp. 154-163, 1903

[5] D.H. Lehmer, *Computer technology applied to the theory of numbers*, Studies in Number Theory, Englewood Cliffs, NJ: Pretice-Hall, pp. 117-151, 1969

[6] L.E. Dickson, *Criteria for the irreducibility of functions in a finite field*, Bull. Amer. Math. Soc., Vol. 13, No. 1, pp. 1-8, 1906

[7] G. Gong and L. Harn, *Public key cryptosystems based on cubic finite field extensions*, IEEE Trans. Information Theory, Vol.45, pp. 2601-2605, 1999

[8] S. Müller, *On the computation of square roots in finite fields*, Design, Codes and Cryptography, Vol.31, pp. 301-312, 2004

[9] N. Nishihara, R. Harasawa, Y. Sueyoshi, and A. Kudo, *A remark on the computation of cube roots in finite fields*, preprint, available at http://eprint.iacr.org/2009/457.pdf.

[10] I.B. Damgård and G.S. Frandsen, *Efficient algorithm for the gcd and cubic residuosity in the ring of Eisenstein integers*, J. Symbolic Computation, Vol. 39, pp. 643-652, 2005

[11] A. O. L. Atkin, *Probabilistic primality testing*, summary by F. Morain, Inria Research Report 1779, pp.159-163, 1992

[12] F. Kong, Z. Cai, J. Yu, and D. Li, *Improved generalized atkin algorithm for computing square roots in finite fields*, Info. Processing Letters, Vol. 98, no. 1, pp. 1-5, 2006.

[13] A. V. Sutherland, *Structure computation and discrete logarithms in finite abelian p-groups*, Math. Comp., Vol. 80, pp. 477-500, 2011.

[14] D. Bernstein, *Faster square roots in annoying finite fields*, preprint, available at http://cr.yp.to/papers/sqroot.pdf.

[15] R. C. Peralta, *A simple and fast probabilistic algorithm for computing square roots modulo a prime number*, IEEE Trans. Information Theory, Vol.32, pp. 846-847, 1986

[16] P. S. Barreto and J. F. Voloch, *Efficient computation of roots in finite fields*, Design, Codes and Cryptography, Vol.39, pp. 275-280, 2006

[17] S. Lindhurst, *An analysis of Shanks's algorithm for computing square roots in finite fields*, CRM Proc. and Lecture Notes, vol. 19, pp. 231-242, 1999

[18] D. Han, D. Choi, and H. Kim, *Improved computation of square roots in specific finite fields*, IEEE Trans. on Computers, Vol. 58, pp. 188-196, 2009.

[19] S. Lang, *Algebra*, Springer, 2005

[20] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press,1997

[21] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Springer, 1992