Trace Expression of r-th Root over 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

Efficient computation of r-th root in $\mathbb{F}_q$ has many applications in computational number theory and many other related areas. We present a new r-th root formula which generalizes Müller's result on square root, and which provides a possible improvement of the Cipolla-Lehmer type algorithms for general case. More precisely, for given r-th power $c \in \mathbb{F}_q$, we show 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 $Tr(\alpha) = \alpha + \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{r-1}}$ and α is a root of certain irreducible polynomial of degree r over $\mathbb{F}_q$.

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

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

1 Introduction

Let $r > 1$ be an integer and q be a power of a prime. Finding r-th root (or finding a root of $x^r = c$) in finite field $\mathbb{F}_q$ has many applications in computational number theory and in many other related topics. Some such examples include point halving and point compression on elliptic curves [16], where square root computations are needed. Similar applications for high genus curves require r-th root computation also.

Among several available root extraction methods of the equation $x^r - c = 0$, two algorithms are applicable for any integer $r > 1$; the Adleman-Manders-Miller [1] algorithm, a straightforward generalization of the Tonelli-Shanks square root algorithm [17, 19] to the case of r-th roots, and the Cipolla-Lehmer [6, 11] algorithms. Due to the cumbersome extension field arithmetic needed for the Cipolla-Lehmer algorithm, one usually prefers the Tonelli-Shanks or the Adleman-Manders-Miller, and other related researches [2, 10, 3] exist to improve the Tonelli-Shanks.

The efficiency of the Adleman-Manders-Miller algorithm depends on the exponent ν of r satisfying $r^\nu | q-1$ and $r^{\nu+1} \not| q-1$, which makes the worst case complexity of the Adleman-Manders-Miller $O(\log r \log^4 q)$ [1, 4] while the Cipolla-Lehmer can be executed in $O(r \log^3 q)$ [6, 11]. Even in the case of $r = 2$, it had been observed in [15] that, for a prime $p = 9 \times 2^{3354} + 1$, running the Tonelli-Shanks algorithm using various software such as Magma, Mathematica and Maple cost roughly 5 minutes, 45 minutes, 390 minutes, respectively while the Cipolla-Lehmer costs under 1 minute in any of the above softwares. It should be mentioned that such extreme cases (of p with $p-1$ divisible by high powers of 2) do happen in many cryptographic applications. For example, one of the NIST suggested curve [16] P-224 : $y^2 = x^3 - 3x + b$ over $\mathbb{F}_p$ uses a prime $p = 2^{224} - 2^{96} + 1$.

On the other hand, it is also true that the Adleman-Manders-Miller runs faster than the Cipolla-Lehmer for small exponents ν. A possible speed-up of the Cipolla-Lehmer comparable to the Tonelli-Shanks for low exponent ν was first given by Müller [15], where a special type of Lucas sequence corresponding to $f(x) = x^2 - Px + 1$ was used. The constant term 1 of $f(x)$ makes the given algorithm runs quite faster compared with the original Cipolla-Lehmer. A similar result for the case $r = 3$ was also obtained in [5].

In this paper, we show that the idea in [15] can be generalized to any integer $r > 1$. More precisely, for any r-th power c in $\mathbb{F}_q$, we can construct a polynomial $f(x) \in \mathbb{F}_q[x]$ of degree r with constant term ± 1 such that the irreducibility of f implies that $\left\{ Tr(\alpha^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}) \right\}^r = c$ where $f(\alpha) = 0$ and $Tr(\alpha) = \alpha + \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{r-1}}$. We mention that the case $r = 2$ (i.e., $\{Tr(\alpha^{\frac{q-1}{4}})\}^2 = c$) is the result in [15] and the case $r = 3$ (i.e., $\{Tr(\alpha^{\frac{q^2+q-2}{9}})\}^3 = c$) is shown in [5]. Therefore a possible existence of efficient linear recurrence relation computing $Tr(\alpha^m)$ guarantees the existence of efficient r-th root algorithm, where the case $r = 2, 3$ are well-known.

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 r-th order linear recurrence sequences. In Section 4, we propose a new r-th formula which has a possible application when combined with linear recurrence relations. Finally, in Section 5, we give concluding remarks and future works.

2 Existing r-th Root Extraction Methods in $\mathbb{F}_q$

In this section, we introduce two standard algorithms for computing r-th root in finite field, that is, the Adleman-Manders-Miller [1] algorithm and the Cipolla-Lehmer algorithm [6, 11].

2.1 Tonelli-Shanks and Adleman-Manders-Miller algorithm

The Adleman-Manders-Miller algorithm [1] is described in Table 1. Assuming $r << \log q$ and DLP (discrete logarithm problem) is easy in the multiplicative subgroup of order r in $\mathbb{F}_q^\times$, the complexity is given as $O(\nu_r(q-1) \log r \log^3 q)$, where $\nu_r(q-1)$ denotes the largest non-negative integer ν satisfying $r^\nu | q - 1$. Therefore, when $\nu_r(q-1)$ is small, the Adleman-Manders-Miller algorithm has the complexity $O(\log r \log^3 q)$, while has the worst complexity $O(\log r \log^4 q)$ when $\nu_r(q-1) \approx \log q$.

2.2 Cipolla-Lehmer algorithm

The Cipolla-Lehmer algorithm [6, 11] is described in Table 2. Its complexity is $O(r \log^3 q)$, which does not depend on $\nu = \nu_r(q-1)$ unlike the case of the Adleman-Manders-Miller. However, for small $\nu = \nu_r(q-1)$, the Adleman-Manders-Miller algorithm performs better than the Cipolla-Lehmer due to the relatively large constant term in the complexity estimation of the Cipolla-Lehmer usually omitted in the notation O. Hence the refinements of the Cipolla-Lehmer is desirable.

Let $c \in \mathbb{F}_q$ be an r-th power in $\mathbb{F}_q$ with $q \equiv 1 \pmod{r}$. To find an r-th root of c, the Cipolla-Lehmer algorithm needs an irreducible polynomial $f(x) = x^r - b_{r-1}x^{r-1} - b_{r-2}x^{r-2} - \cdots - b_1 x + (-1)^r c$ with constant term $(-1)^r c$. Letting $\alpha \in \mathbb{F}_{q^r}$ be a root of f, we get $\alpha^{1+q+q^2+\cdots+q^{r-1}} = c$ so that $\alpha^{\frac{\sum_{i=0}^{r-1} q^i}{r}}$ is an r-th root of c. Irreducibility testing of f and the exponentiation $\alpha^{\frac{\sum_{i=0}^{r-1} q^i}{r}}$

Table 1: Adleman-Manders-Miller r-th root algorithm

Input: An rth power δ in $\mathbb{F}_q$ with $r \mid q-1$ Output: An r-th root of δ
Step 1: Let $q-1 = r^s t$ with $(r,t)=1$ Compute the least nonnegative integer u such that $t \mid ru-1$ Choose ρ randomly in $\mathbb{F}_q$
Step 2: $a \leftarrow \rho^{r^{s-1}t}$, $c \leftarrow \rho^t$ **if** $a=1$, **go to** Step 1
Step 3: $b \leftarrow \delta^{ru-1}$, $h \leftarrow 1$
Step 4: **for** $i=1$ **to** $s-1$ $d \leftarrow b^{r^{s-1-i}}$ **if** $d=1$, **then** $j \leftarrow 0$ **else then** $j \leftarrow -\log_a d$ (compute the discrete logarithm) $b \leftarrow b(c^r)^j$, $h \leftarrow hc^j$ $c \leftarrow c^r$ **end for**
Step 5: **return** $\delta^u \cdot h$

Table 2: Cipolla-Lehmer r-th root algorithm

Input: An r-th power c in $\mathbb{F}_q$ Output: A r-th root of c
Step 1: Choose $b_1, b_2, \cdots, b_{r-1}$ randomly in $\mathbb{F}_q$
Step 2: $f(x) \leftarrow x^r - b_{r-1}x^{r-1} - b_{r-2}x^{r-2} - \cdots - b_1 x + (-1)^r c$ **if** f is reducible, **then** go to Step 1
Step 3: Return $x^{\frac{\sum_{i=0}^{r-1} q^i}{r}} \pmod{f(x)}$

(or computing $x^{\frac{\sum_{i=0}^{r-1} q^i}{r}} \pmod{f(x)}$) needs many multiplications in $\mathbb{F}_q$, and the number of such multiplications depends on the coefficients of f. One may choose a low hamming-weight polynomial (i.e., trinomial) to reduce the cost of computing $x^{\frac{\sum_{i=0}^{r-1} q^i}{r}} \pmod{f(x)}$.

3 Linear Recurrence Sequences

Let $f(x) = x^r - b_{r-1}x^{r-1} - b_{r-2}x^{r-2} - \cdots - b_1x - b_0$ ($b_i \in \mathbb{F}_q$) be irreducible over $\mathbb{F}_q$. An r-th order linear recurrence sequence s_k corresponding to $f(x)$ is defined as

$$s_k = b_{r-1}s_{k-1} + b_{r-2}s_{k-2} + \cdots + b_0s_{k-r}, \qquad k \geq r.$$

It is well-known [18, 12] that such s_k is completely determined when f and the first r terms $s_0, s_1, \cdots, s_{r-1}$ are given. In fact, there is uniquely determined $\theta \in \mathbb{F}_{q^r}$ such that

$$s_k = Tr(\theta\alpha^k), \tag{1}$$

where α is a root of $f(x)$ and the trace map $Tr : \mathbb{F}_{q^r} \to \mathbb{F}_q$ is defined as $Tr(\beta) = \beta + \beta^q + \beta^{q^2} + \cdots + \beta^{q^{r-1}}$. We say that s_k is the characteristic sequence generated by $f(x)$ if $\theta = 1$, i.e., if s_k can be expressed as

$$s_k = Tr(\alpha^k) = \alpha^k + \alpha^{kq} + \alpha^{kq^2} + \cdots + \alpha^{kq^{r-1}}. \tag{2}$$

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 $s_k(f), s_k(b_0, \cdots, b_{r-1})$, or $s_k(\alpha)$. For small values of r, the sequence s_k can be computed using "double and add" method.

Example 1:

A. When $r = 2$ and $f(x) = x^2 - Px + Q$, one has the following Lucas relation [15] :

$$s_{2n} = s_n^2 - 2Q^n, \qquad s_{n+m} = s_ns_m - Q^ms_{n-m}$$

The exponentiation Q^n gives extra burden to the computation s_k, and one can compute the recurrence relation more efficiently letting $Q = 1$.

B. When $r = 3$ and $f(x) = x^3 - ax^2 + bx - c$, one has the following relation which can be found, for example, in the work of Gong and Harn [9] :

$$s_{2n} = s_n^2 - 2c^ns_{-n}, \qquad s_{n+m} = s_ns_m - c^ms_{n-m}s_{-m} + c^ms_{n-2m} \tag{3}$$

As in the case of second order recurrence relation, letting $c = 1$ makes the computation of the sequence cost effective.

Note that letting the constant term of $f(x)$ to be ± 1 makes it impossible to use the Cipolla-Lehmer. For example, when $r = 2$, to apply the Cipolla-Lehmer for the computation of the roots of $x^2 - c = 0$, one has to use the polynomial $x^2 - bx + c$ not $x^2 - bx + 1$. However, as is done by Müller [15] for the quadratic case, an wise choice of f of degree r gives a way to find the r-th root of $c \in \mathbb{F}_q$ as will be shown in the next sections.

From now on, we will consider the characteristic sequence s_k which comes from the irreducible polynomial $f(x) = x^r - b_{r-1}x^{r-1} - b_{r-2}x^{r-2} - \cdots - b_1x + (-1)^r$.

4 Trace Expression of r-th Root

Our main result is the Theorem 2, and we will discuss the necessary prerequisites first. Let r be an integer > 1 and let b be in $\mathbb{F}_q$ with $q \equiv 1 \pmod{r}$ such that

$$f(x) = (x + (-1)^r)^r + (-1)^{r+1}(b + (-1)^r r)x \tag{4}$$

is irreducible over $\mathbb{F}_q$. Also we define a polynomial $h(x)$ as

$$h(x) = x^r + (-1)^{r+1}(b + (-1)^r r)(x-1). \tag{5}$$

Then one has the following relation

$$h(1 + (-1)^r x) = (-1)^r f(x), \tag{6}$$

because

$$\begin{aligned} h(1 + (-1)^r x) &= (1 + (-1)^r x)^r + (-1)^{r+1}(b + (-1)^r r)(1 + (-1)^r x - 1) && (7)\\ &= (-1)^r (x + (-1)^r)^r + (-1)^{r+1}(b + (-1)^r r)(-1)^r x && (8)\\ &= (-1)^r \{(x + (-1)^r)^r + (-1)^{r+1}(b + (-1)^r r)x\} = (-1)^r f(x). && (9) \end{aligned}$$

The above equations implies that one has the following when r is even,

$$f(x) = (x+1)^r - (b+r)x, \quad h(x) = x^r - (b+r)x + (b+r), \qquad h(1+x) = f(x), \tag{10}$$

and when r is odd, one has

$$f(x) = (x-1)^r + (b-r)x, \quad h(x) = x^r + (b-r)x - (b-r), \qquad h(1-x) = -f(x). \tag{11}$$

In particular, the irreducibility of f implies the irreducibility of h and vice versa.

Suppose that α is a root of $f(x)$. Since $f(0) = (-1)^r$, we find that the norm of f (i.e., the product of all the conjugates of α) is

$$\alpha^{1+q+q^2+\cdots+q^{r-1}} = 1. \tag{12}$$

A classical result of Hilbert Theorem 90 says that there exists $\beta \in \mathbb{F}_{q^r}$ such that $\beta^r = \alpha$. More precisely, using the equation (12), one can show that

$$\alpha(1 + \alpha + \alpha^{1+q} + \cdots + \alpha^{1+q+\cdots+q^{r-2}})^q = 1 + \alpha + \alpha^{1+q} + \cdots + \alpha^{1+q+\cdots+q^{r-2}}. \tag{13}$$

Therefore letting $\beta = (1 + \alpha + \alpha^{1+q} + \cdots + \alpha^{1+q+\cdots+q^{r-2}})^{\frac{1-q}{r}}$, from the equation (13), we get

$$\beta^r = \alpha.$$

Theorem 1. *Assuming $f(\alpha) = 0$ and $q \equiv 1 \pmod r$, we have*

$$\begin{aligned} &\alpha^{\frac{1+q+q^2+\cdots+q^{r-1}}{r}} = (b+r)^{-\frac{q-1}{2}} && \text{if } r \text{ is } \textit{even},\\ &\alpha^{\frac{1+q+q^2+\cdots+q^{r-1}}{r}} = 1 && \text{if } r \text{ is } \textit{odd}. \end{aligned}$$

In particular, when r is even and $b+r$ is a square in $\mathbb{F}_q$, one gets $\alpha^{\frac{1+q+q^2+\cdots+q^{r-1}}{r}} = 1$.

Proof. Since $h(1 + (-1)^r \alpha) = f(\alpha) = 0$ and $h(0) = (-1)^r(b + (-1)^r r)$,

$$(1 + (-1)^r \alpha)^{\sum_{i=0}^{r-1} q^i} = b + (-1)^r r. \tag{14}$$

On the other hand, by simplifying the equation (7), we have

$$h(1 + (-1)^r x) = (1 + (-1)^r x)^r - (b + (-1)^r r)x = (-1)^r f(x), \tag{15}$$

which implies

$$(1+(-1)^r\alpha)^r = (b+(-1)^r r)\alpha. \tag{16}$$

By taking $\frac{\sum_{i=0}^{r-1} q^i}{r}$-th power to both sides of the above expression, one has

$$(1+(-1)^r\alpha)^{\sum_{i=0}^{r-1} q^i} = (b+(-1)^r r)^{\frac{\sum_{i=0}^{r-1} q^i}{r}} \alpha^{\frac{\sum_{i=0}^{r-1} q^i}{r}}. \tag{17}$$

Comparing two expressions (14) and (17), we get

$$\begin{aligned} \alpha^{\frac{\sum_{i=0}^{r-1} q^i}{r}} &= (b+(-1)^r r)^{-\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}} = (b+(-1)^r r)^{-\frac{\sum_{i=0}^{r-1}(q^i-1)}{r}} \\ &= (b+(-1)^r r)^{-(q-1)\frac{\sum_{i=0}^{r-2}\sum_{j=0}^{i} q^j}{r}}. \end{aligned} \tag{18}$$

Since $q \equiv 1 \pmod r$, we have

$$\sum_{i=0}^{r-2}\sum_{j=0}^{i} q^j \equiv \frac{r(r-1)}{2} \pmod r, \tag{19}$$

which is $\frac{r}{2} \pmod r$ when r is even, and is $0 \pmod r$ when r is odd. Noticing $b+(-1)^r r \in \mathbb{F}_q$, one has the desired result. □

Corollary 1. *Assume $q \equiv 1 \pmod r$. If r is even, further assume that $b+r$ is a square in $\mathbb{F}_q$. Then $s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}}(\beta)^r = s_{\sum_{i=0}^{r-2} q^i}(\beta)^r$.*

Proof. Letting $\beta^{\frac{\sum_{i=0}^{r-1} q^i}{r}} = \omega$ with $\beta^r = \alpha$ and using Theorem 1, we have $\omega^r = \beta^{\sum_{i=0}^{r-1} q^i} = \alpha^{\frac{\sum_{i=0}^{r-1} q^i}{r}} = 1$ and $\omega^q = \omega$. Therefore

$$\begin{aligned} s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}}(\beta)^r &= Tr(\beta^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}})^r \\ &= (\beta^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}} + \beta^{q\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}} + \beta^{q^2\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}} + \cdots + \beta^{q^{r-1}\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}})^r \\ &= (\omega\beta^{-1} + \omega^q\beta^{-q} + \omega^{q^2}\beta^{-q^2} + \cdots + \omega^{q^{r-1}}\beta^{-q^2})^r \\ &= (\beta^{(\sum_{i=0}^{r-1} q^i)-1} + \beta^{(\sum_{i=0}^{r-1} q^i)-q} + \beta^{(\sum_{i=0}^{r-1} q^i)-q^2} + \cdots + \beta^{(\sum_{i=0}^{r-1} q^i)-q^{r-1}})^r \\ &= Tr(\beta^{\sum_{i=0}^{r-2} q^i})^r = s_{\sum_{i=0}^{r-2} q^i}(\beta)^r. \end{aligned} \tag{20}$$

□

Corollary 2. *Assuming the same conditions as in the Corollary 1 and also assuming $q \equiv 1 \pmod{r^2}$, one has $s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(\alpha)^r = s_{\sum_{i=0}^{r-2} q^i}(\beta)^r$.*

Proof.

$$\begin{aligned} s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(\alpha)^r &= Tr(\alpha^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}})^r = Tr((\beta^r)^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}})^r \\ &= Tr(\beta^{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r}})^r = s_{\sum_{i=0}^{r-2} q^i}(\beta)^r, \end{aligned} \tag{21}$$

where the last equality comes from the Corollary 1. □

If $b+(-1)^r r$ is an r-th power in $\mathbb{F}_q$, one can explicitly find r-th root of $b+(-1)^r r$ as follows.

Corollary 3. *Assume that $q \equiv 1 \pmod r$ and $b+(-1)^r r$ is an r-th power in $\mathbb{F}_q$, then $s_{\sum_{i=0}^{r-2} q^i}(\beta)^r = b+(-1)^r r$.*

Proof. Since $\alpha = \beta^r \in \mathbb{F}_{q^r}$, we may rewrite the equation (16) as

$$(1+(-1)^r\alpha)^r = (b+(-1)^r r)\beta^r. \tag{22}$$

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

$$(1+(-1)^r\alpha) = \omega_0 u\beta \tag{23}$$

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

$$\begin{aligned} Tr(\beta^{\sum_{i=0}^{r-2} q^i}) &= \frac{1}{\omega_0^{r-1}u^{r-1}} Tr((1+(-1)^r\alpha)^{\sum_{i=0}^{r-2} q^i}) \\ &= \frac{1}{\omega_0^{r-1}u^{r-1}}(b+(-1)^r r) \\ &= \omega_0 u, \end{aligned} \tag{24}$$

where the first equality comes from $\omega_0 u \in \mathbb{F}_q$ and the second equality comes from the coefficient $(-1)^{r+1}(b+(-1)^r r)$ of x in $h(x) = x^r + (-1)^{r+1}(b+(-1)^r r)(x-1)$. We also have the last equality because $\omega_0^r = 1$ and $b+(-1)^r r = u^r$. Therefore we get

$$Tr(\beta^{\sum_{i=0}^{r-2} q^i})^r = (\omega_0 u)^r = b+(-1)^r r. \tag{25}$$

□

Finally, combining the Corollaries 2 and 3, we have the following theorem.

Theorem 2. *Suppose that $q \equiv 1 \pmod{r^2}$ and $f(x) = (x+(-1)^r)^r + (-1)^{r+1}(b+(-1)^r r)x$ is an irreducible polynomial over $\mathbb{F}_q$ with $f(\alpha) = 0$. Assume $b+(-1)^r r$ is an r-th power in $\mathbb{F}_q$. Then $s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(\alpha)^r = b+(-1)^r r$.*

Proof. We have

$$s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(\alpha)^r = s_{\sum_{i=0}^{r-2} q^i}(\beta)^r = b+(-1)^r r,$$

where the first equality comes from Corollary 2 and the second equality is Corollary 3. □

Now using the polynomial $f(x)$, we can find an r-th root for given r-th power c in $\mathbb{F}_q$. For given r-th power $c \in \mathbb{F}_q$, define $b = c-(-1)^r r$. If $f(x)$ with given coefficient b is irreducible, then $s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(f)$ is an r-th root of c. That is,

$$s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(f)^r = b+(-1)^r r = c.$$

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 $b = ct^r - (-1)^r r$. Then

$$s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(f)^r = b+(-1)^r r = ct^r,$$

Table 3: New r-th root algorithm for $\mathbb{F}_q$ with $q \equiv 1 \pmod{r^2}$

Input: An r-th power c in $\mathbb{F}_q$ Output: s satisfying $s^r = c$
Step 1: $t \leftarrow 1$, $b \leftarrow ct^r - (-1)^r r$, $f(x) \leftarrow (x + (-1)^r)^r + (-1)^{r+1}(b + (-1)^r r)x$
Step 2: **while** $f(x)$ is reducible over $\mathbb{F}_q$ Choose random $t \in \mathbb{F}_q$ $b \leftarrow ct^r - (-1)^r r$, $f(x) \leftarrow (x + (-1)^r)^r + (-1)^{r+1}(b + (-1)^r r)x$ **end while**
Step 3: $s \leftarrow s_{\frac{(\sum_{i=0}^{r-1} q^i) - r}{r^2}}(f) \cdot t^{-1}$

which implies $t^{-1} s_{\frac{(\sum_{i=0}^{r-1} q^i) - r}{r^2}}(f)$ is an r-th root of c (See Table 3).

Example 2:

A. $r = 2$**:** For given square $c \in \mathbb{F}_q$, we have $f(x) = (x+1)^2 - (b+2)x = x^2 - bx + 1$ with $b = c - 2$. If f is irreducible over $\mathbb{F}_q$, one has $s_{\frac{q-1}{4}}(f)^2 = b + 2 = c$, and such $s_{\frac{q-1}{4}}$ can be computed via Lucas sequence $s_k = bs_{k-1} - s_{k-2}$ (See [15]).

B. $r = 3$**:** For given cube $c \in \mathbb{F}_q$, we have $f(x) = (x-1)^3 + (b-3)x = x^3 - 3x^2 + bx - 1$ with $b = c + 3$. If f is irreducible over $\mathbb{F}_q$, one has $s_{\frac{q^2+q-2}{9}}(f)^3 = b - 3 = c$, and such $s_{\frac{q^2+q-2}{9}}$ can be computed via the third order linear recurrence sequence $s_k = 3s_{k-1} - bs_{k-2} + s_{k-3}$ using the relation in the equation (3) (See [5]).

Our theorem and examples were explained on the assumption of $q \equiv 1 \pmod{r^2}$. However it should be mentioned that one can find an r-th root of c when $q \not\equiv 1 \pmod{r^2}$ easily. For example, when $r = 2$ and $q \equiv 3 \pmod 4$, a square root of a quadratic residue c is given by $c^{\frac{q+1}{4}}$. Also when $r = 3$ and $q \not\equiv 1 \pmod 9$, one has the followings. 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 $\mathbb{F}_q$.

These closed formulas are not obtained by ad-hoc method. In fact, one has the following simple result of r-th root when $q \not\equiv 1 \pmod{r^2}$.

Proposition 1. *Let q be a prime power such that $q \equiv 1 \pmod r$ but $q \not\equiv 1 \pmod{r^2}$. Assume that $gcd(\frac{q-1}{r}, r) = 1$. Then, for given r-th power c in $\mathbb{F}_q$, an r-th root of c can be computed by the cost of one exponentiation in $\mathbb{F}_q$. In particular, if r is a prime, then the condition $gcd(\frac{q-1}{r}, r) = 1$ is automatically satisfied so that the cost of finding r-th root of c is just one exponentiation.*

Proof. We claim that there is an integer θ depending only on r and q but not on c such that

$$(A)\ \theta < rq, \qquad (B)\ r^2|\theta, \qquad (C)\ \left(c^{\frac{\theta}{r^2}}\right)^r = c \tag{26}$$

The condition (C) of the above equation says that $c^{\frac{\theta}{r}} = c$, i.e., $c^{\frac{\theta-r}{r}} = 1$. Since c is an r-th power in $\mathbb{F}_q$, this condition can be satisfied if $\theta \equiv r \pmod{(q-1)}$. Therefore writing $\theta = r + k(q-1)$, the condition (B) says that one should have $r + k(q-1) \equiv 0 \pmod{r^2}$, which is equivalent to the following equation

$$1 + k\frac{q-1}{r} \equiv 0 \pmod{r}. \tag{27}$$

Since $gcd(\frac{q-1}{r}, r) = 1$, the above equation has unique solution $k \pmod{r}$. Now the condition (C) is satisfied because $\theta = kq + r - k \leq (r-1)q + 1 < rq$. Finally, if r is a prime, then the assumption $q \not\equiv 1 \pmod{r^2}$ implies $gcd(\frac{q-1}{r}, r) = 1$. □

Example 3:

A. $r = 3$**:** When $r = 3$, the equation (27) becomes $1 + k\frac{q-1}{3} \equiv 0 \pmod{3}$. Therefore depending on the values of $\frac{q-1}{3} \pmod{3}$, the corresponding $k \pmod{3}$ is uniquely determined and they are

$$\left(\frac{q-1}{3}, k\right) = (1,2), (2,1). \tag{28}$$

Since $\frac{q-1}{3} \equiv j \pmod{3}$ implies $q \equiv 3j + 1 \pmod{3^2}$, we have the following table of pairs of $q \pmod{3^2}$ and corresponding $\theta = kq + 3 - k$

$$(q \pmod 9,\ \theta) = (4, 2q+1), (7, q+2). \tag{29}$$

That is, when $q \equiv 4 \pmod 9$, the a cube root of c is given as $c^{\frac{2q+1}{9}}$, and when $q \equiv 7 \pmod 9$, the a cube root of c is given as $c^{\frac{q+2}{9}}$.

B. $r = 5$**:** When $r = 5$, the equation (27) becomes $1 + k\frac{q-1}{5} \equiv 0 \pmod{5}$. Therefore depending on the values of $\frac{q-1}{5} \pmod{5}$, the corresponding $k \pmod{5}$ is uniquely determined and they are

$$\left(\frac{q-1}{5}, k\right) = (1,4), (2,2), (3,3), (4,1). \tag{30}$$

Since $\frac{q-1}{5} \equiv j \pmod{5}$ implies $q \equiv 5j + 1 \pmod{5^2}$, we have the following table of pairs of $q \pmod{5^2}$ and corresponding $\theta = kq + 5 - k$

$$(q \pmod{25},\ \theta) = (6, 4q+1), (11, 2q+3), (16, 3q+2), (21, q+4). \tag{31}$$

For example, when $q \equiv 6 \pmod{25}$, the an 5-th root of c is given as $c^{\frac{4q+1}{25}}$, and when $q \equiv 11 \pmod{25}$, the an 5-th root of c is given as $c^{\frac{2q+3}{25}}$, etc.

Remarks:

1. The reason why we only consider the case $r|q-1$ (i.e., $q \equiv 1 \pmod{r}$) is as follows. If $r \not| q-1$, then one has $gcd(r, q-1) = 1$ and there are a, b satisfying $ra + (q-1)b = 1$. Thus for any $c \in \mathbb{F}_q$, we have $c = c^{ra+(q-1)b} = (c^a)^r$. That is, any element c is an r-th powers of c^a.

2. For r-th root extraction, considering the cases $r = prime$ is enough for practical purposes. For example, to find 4-th root of $c \in \mathbb{F}_q$, we only have to use square root algorithm twice instead of using 4-th root algorithm once, and the complexity of two applications of square root algorithm is lower than that of one application of 4-th root algorithm.

5 Conclusions

Randomly selected monic polynomial over $\mathbb{F}_q$ of degree r with nonzero constant term is irreducible with probability $\frac{1}{r}$ (For an explanation, see [18, 14]). Even if our choice of f in (4) is not really random, experimental evidence (using software tools such as MAPLE and SAGE) shows that $\frac{1}{r}$ of such f is irreducible, which implies that an irreducible f can be found after r random tries. Irreducibility testings of low degree polynomials are well understood and can be implemented efficiently, see [7, 18, 12, 14]. Therefore the algorithm in Table 3 is dominated by the complexity of step 3 which computes $s_{\frac{(\sum_{i=0}^{r-1} q^i)-r}{r^2}}(f)$. For $r = 2, 3$, i.e., for quadratic and cubic polynomials, the well-known linear recurrence sequences give faster algorithms than previously proposed Cipolla-Lehmer type algorithms. For $r > 3$, there are some known recurrence relations, for example in [8]. However those sequences do not seem to give efficient algorithms to compute $s_m(f)$ and further study is needed.

References

[1] 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

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

[3] D. Bernstein, *Faster square root in annoyng finite field*, preprint, *Available from* http://cr.yp.to/papers/sqroot.pdf, 2001

[4] Z. Cao, Q. Sha, and X. Fan, *Adlemen-Manders-Miller root extraction method revisited*, preprint, *available from* http://arxiv.org/abs/1111.4877, 2011

[5] G.H. Cho, N. Koo, E. Ha, and S. Kwon, *New cube root algorithm based on third order linear recurrence relation in finite field*, preprint, *available from* http://eprint.iacr.org/2013/024.pdf, 2013

[6] 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

[7] 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

[8] K.J. Giuliani and G. Gong, *A New Algorithm to compute remote terms in special types of characteristic sequences*, Proc. International Conference on Sequences and Their Applications (SETA), LNCS 4086, pp. 237-247, 2006

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

[10] F. Kong, Z. Cai, J. Yu, and D. Li, *Improved Generalized Atkin Algorithm for Computing Square Roots in Finite Fields*, Information Processing Letters, Vol. 98, no. 1, pp. 1-5, 2006.

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

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

[13] 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

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

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

[16] NIST, *Digital Signature Standard*, Federal Information Processing Standard 186-3, National Institute of Standards and Technology, *Available from* http://csrc.nist.gov/publications/fips/, 2000

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

[18] I. Shparlinski, *Finite Fields: Theory and Computation*, Springer, 1999

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