Root Computation in Finite Fields*

Ryuichi HARASAWA^{†a)}, Yutaka SUEYOSHI[†], and Aichi KUDO[†], *Members*

SUMMARY We consider the computation of r -th roots in finite fields. For the computation of square roots (i.e., the case of $r = 2$), there are two typical methods: the Tonelli-Shanks method [7], [10] and the Cipolla-Lehmer method [3], [5]. The former method can be extended to the case of r -th roots with r prime, which is called the Adleman-Manders-Miller method [1]. In this paper, we generalize the Cipolla-Lehmer method to the case of r -th roots in $\mathbb{F}_q$ with r prime satisfying $r \mid q - 1$, and provide an efficient computational procedure of our method. Furthermore, we implement our method and the Adleman-Manders-Miller method, and compare the results.

key words: root computation, finite field, the Cipolla-Lehmer method

1. Introduction

Solving algebraic equations over finite fields $\mathbb{F}_q$ is one of the most fundamental topics in computer algebra. The typical equations are of the form

$$x^r = a,$$

given a natural integer $r (\geq 2)$ and an r -th residue element a in the base field.

An important application of r -th root computations in $\mathbb{F}_q$ to computer science is to construct an algebraic curve cryptosystem and a geometric Goppa code using curves over $\mathbb{F}_q$ of the form $y^r = f(x)$ (e.g., (hyper-)elliptic and super-elliptic curves). In order to find an $\mathbb{F}_q$ -rational point (α, β) on the curve $y^r = f(x)$, we need to compute an r -th root β (if exists) of $f(\alpha)$ for an element α in $\mathbb{F}_q$. Such situation also arises in the operation of hashing messages onto the curve.

For square roots (i.e., $r = 2$), there exist two well-known methods: the Tonelli-Shanks method [7], [10] and the Cipolla-Lehmer method [3], [5]. The idea of the former method is to reduce the computation of square roots in $\mathbb{F}_q$ to that in the Sylow 2-subgroup of $\mathbb{F}_q^*$, and the idea of the latter method is to apply the norm map of $\mathbb{F}_{q^2}$ into $\mathbb{F}_q$. The Tonelli-Shanks method can be extended to the case of r -th roots with r prime, which is called the Adleman-Manders-Miller method [1]. However, as pointed out in the paper [2], it seems to be difficult to extend the Cipolla-Lehmer method directly to more general cases.

Manuscript received September 20, 2012.

Manuscript revised January 4, 2013.

[†]The authors are with the Graduate School of Engineering, Nagasaki University, Nagasaki-shi, 852-8521 Japan.

*This work was partially supported by the Japan Society for the Promotion of Science (JSPS) under the Grant-in-Aid for challenging Exploratory Research No. 24650009.

a) E-mail: harasawa@cis.nagasaki-u.ac.jp

DOI: 10.1587/transfun.E96.A.1081

In this paper, we provide an explicit algorithm to realize the Cipolla-Lehmer method in the case of r -th roots in finite fields $\mathbb{F}_q$ with r prime satisfying $r \mid q - 1$. More precisely, we give a method to construct an irreducible monic polynomial of degree r satisfying some condition to apply the Cipolla-Lehmer method. We next describe an efficient computational procedure of our method. Furthermore, we implement our method and the Adleman-Manders-Miller method, and compare the results.

The remainder of this paper is organized as follows: In Sect. 2, we explain the Cipolla-Lehmer method for the square root computation. In Sect. 3, we consider the generalization of the Cipolla-Lehmer method to the r -th root computation. In Sect. 4, we provide an efficient computational procedure of our method, and estimate its complexity. In Sect. 5, we explain the Adleman-Manders-Miller method and its complexity. In Sect. 6, we implement our method and the Adleman-Manders-Miller method for various values of r over finite fields whose bit sizes are 500, 1000 and 2000. Based on the experimental results, we compare these methods and describe some observations. In Sect. 7, we give the conclusion and future works.

2. Square Root Computation Using the Cipolla-Lehmer Method

Before considering r -th root computations in $\mathbb{F}_q$ with r prime, we mention that it is easy to compute r -th roots in $\mathbb{F}_q$ for the cases $r \nmid q - 1$ and $r \parallel q - 1$. In the case of $r \nmid q - 1$, for each element a in $\mathbb{F}_q$, the r -th root of a is a^l with $l \equiv r^{-1} \pmod{q - 1}$. In the case of $r \parallel q - 1$, for an r -th residue element a in $\mathbb{F}_q$ (i.e., $a^{\frac{q-1}{r}} = 1$), an r -th root of a is a^l with $l \equiv r^{-1} \pmod{\frac{q-1}{r}}$. Furthermore, for the two cases above ($r \nmid q - 1$ and $r \parallel q - 1$), the paper [2] gives an efficient computational procedure for getting an r -th root of a (i.e., the value a^l).

In the rest of this paper, we assume that $\mathbb{F}_q$ is a finite field with q elements, r is a prime with $r \mid q - 1$ and a is an r -th residue element in $\mathbb{F}_q$.

In Table 1, we describe the Cipolla-Lehmer method for computing square roots in finite fields $\mathbb{F}_q$ of odd characteristic ([3], [5] or [6, Algorithm 3.39 (p. 101)]).

The correctness of the Cipolla-Lehmer method is obvious. Indeed, we see the polynomial $f(x) = x^2 - bx + a$

**In the case of even characteristic, for each element a in $\mathbb{F}_q$, the value $a^{\frac{q}{2}}$ is the square root of a .

Table 1 Cipolla-Lehmer algorithm.

Input:	A quadratic residue a in $\mathbb{F}_q$.
Output:	A square root of a .
Step 1:	Choose an element b in $\mathbb{F}_q$ at random. if $b^2 = a$, then return b as a square root of a .
Step 2:	if $b^2 - 4a = 0$, then return $\frac{b}{2}$ as a square root of a . if $b^2 - 4a$ is a quadratic residue, then go to Step 1.
Step 3:	$f(x) \leftarrow x^2 - bx + a$. Return $x^{\frac{q+1}{2}} \bmod f(x)$ as a square root of a .

obtained in Table 1 becomes a monic irreducible polynomial over $\mathbb{F}_q$. Let η be a root of $f(x)$, which is an element in $\mathbb{F}_{q^2} \setminus \mathbb{F}_q$. Since the other root of $f(x)$ becomes η^q , we have

$$f(x) = (x - \eta)(x - \eta^q).$$

Comparing the constant terms of both sides, we get the relation $\eta^{q+1} = a$, which implies that

$$\eta^{\frac{q+1}{2}} (= x^{\frac{q+1}{2}} \bmod f(x))$$

becomes a square root of a (recall that q is odd and that $f(\eta) = 0$). We note that the value η^{q+1} is the norm of η over $\mathbb{F}_q$.

3. Generalization of the Cipolla-Lehmer Method to r -th Roots Computation

In this section, using the Cipolla-Lehmer method, we consider the computation of r -th roots in $\mathbb{F}_q$, that is, the computation of χ satisfying

$$\chi^r = a$$

for a given r -th residue element a in $\mathbb{F}_q$ (i.e., $a^{\frac{q-1}{r}} = 1$).

In order to apply the Cipolla-Lehmer method, we have to construct an irreducible monic polynomial of degree r over $\mathbb{F}_q$ with the constant term $(-1)^r a$. If we construct such a polynomial, say $f(x)$, then we obtain a solution to $\chi^r = a$ as

$$\chi = x^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod f(x).$$

Indeed, let $f(x)$ be such a polynomial, and $\eta \in \mathbb{F}_{q^r}$ a root of $f(x)$. We note that the element η belongs to none of all proper subfields of $\mathbb{F}_{q^r}$ because $f(x)$ is irreducible in $\mathbb{F}_q[x]$. Since the other roots of $f(x)$ become η^{q^i} ($1 \leq i \leq r-1$), we have

$$f(x) = \prod_{0 \leq i \leq r-1} (x - \eta^{q^i}).$$

Comparing the constant terms of both sides, we get the relation

$$\eta^{q^{r-1} + \dots + q + 1} = a$$

(that is, the norm of η over $\mathbb{F}_q$ coincides with a). Therefore, we see the value

$$\eta^{\frac{q^{r-1} + \dots + q + 1}{r}} (= x^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod f(x))$$

becomes an r -th root of a . We note that $f(\eta) = 0$ and that $r \mid q-1$ (i.e., $q \equiv 1 \pmod{r}$) implies $r \mid (q^{r-1} + \dots + q + 1)$.

To describe our method for constructing the polynomial $f(x)$ above, we show a necessary and sufficient condition for a binomial to be irreducible.

Proposition 1: Let $r \geq 2$ be a prime with $r \mid q-1$ and $\beta \in \mathbb{F}_q^*$. Then the binomial $x^r - \beta$ is irreducible in $\mathbb{F}_q[x]$ if and only if $\beta^{\frac{q-1}{r}} \neq 1$ holds.

Proof. First, we assume that $\beta^{\frac{q-1}{r}} = 1$. Then there exists some $\gamma \in \mathbb{F}_q^*$ such that $\beta = \gamma^r$. From the relation

$$\begin{aligned} x^r - \beta &= (x - \gamma)(x^{r-1} + \gamma x^{r-2} + \gamma^2 x^{r-3} + \dots + \gamma^{r-2} x + \gamma^{r-1}), \end{aligned}$$

the binomial $x^r - \beta$ is not irreducible in $\mathbb{F}_q[x]$. This shows the “only if” part of the proposition.

Next, we assume that the binomial $x^r - \beta$ is not irreducible in $\mathbb{F}_q[x]$, and let $g(x) \in \mathbb{F}_q[x]$ be a proper factor of $x^r - \beta$. Let $\zeta \in \mathbb{F}_q^*$ be a primitive r -th root of unity and $\gamma \in \mathbb{F}_{\bar{\mathbb{F}}_q}^*$ an r -th root of β , where $\mathbb{F}_{\bar{\mathbb{F}}_q}$ is an algebraic closure of $\mathbb{F}_q$. Let s ($1 \leq s < r$) be the degree of $g(x)$, then from the relation

$$x^r - \beta = \prod_{0 \leq i \leq r-1} (x - \zeta^i \gamma),$$

the constant term of $g(x)$ is represented as

$$(-1)^s \zeta^t \gamma^s \in \mathbb{F}_q^*$$

for some t with $0 \leq t < r$, so the value $\zeta^t \gamma^s$ is in $\mathbb{F}_q^*$. For the integers l and m with $sl + rm = 1$, we have

$$(\zeta^t \gamma^s)^l = \zeta^{tl} \gamma^{-rm+1} = (\zeta^{tl} \beta^{-m}) \gamma.$$

Since $(\zeta^t \gamma^s)^l$ and $\zeta^{tl} \beta^{-m}$ are in $\mathbb{F}_q^*$, so is γ . Therefore, we have

$$\beta^{\frac{q-1}{r}} = (\gamma^r)^{\frac{q-1}{r}} = \gamma^{q-1} = 1,$$

which shows the “if” part of the proposition. $\square$

We further see, for a polynomial $g(x)$ over $\mathbb{F}_q$,

$g(x)$: irreducible

$$\iff g(x - \alpha): \text{irreducible for any } \alpha \text{ in } \mathbb{F}_q.$$

Using these facts, for a given r -th residue element a , we get a probabilistic algorithm for constructing an irreducible monic polynomial of degree r with constant term $(-1)^r a$ (Table 2), which is of the form

$$(x - \alpha)^r - \beta$$

for some elements α and β in $\mathbb{F}_q$. We remark that, assuming q is sufficiently large as compared with r (e.g., $r = O(\log q)$), the probability that the element $(-1)^r(\alpha^r - a)$ becomes an r -th non-residue for a given r -th residue element a and a random element α in $\mathbb{F}_q$ is approximately $1 - \frac{1}{r}$ (see Appendix).

For the computation of χ , an r -th root of a , we have

Table 2 Proposed algorithm.

Input:	An r -th residue a in $\mathbb{F}_q$.
Output:	An irreducible monic polynomial of degree r with constant term $(-1)^r a$ or an r -th root of a .
Step 1:	Choose an element α in $\mathbb{F}_q$ at random. $\beta \leftarrow (-1)^r(\alpha^r - a)$.
Step 2:	if $\beta = 0$, then return α as an r -th root of a .
Step 3:	if $\beta^{\frac{q-1}{r}} \neq 1$, then return $(x - \alpha)^r - \beta$ as a desired polynomial. else go to Step 1.

$$\begin{aligned}\chi &= x^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod \{(x - \alpha)^r - \beta\} \\ &= (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta),\end{aligned}$$

where $y := x - \alpha$. This gives an efficient method for computing χ because we can perform operations modulo the binomial $y^r - \beta$. For example, given a polynomial $g(y)$ of degree at most $2r - 2$, when we apply the classical method for operations on polynomials, it requires at most $r - 1$ multiplications on $\mathbb{F}_q$ to compute $g(y) \bmod (y^r - \beta)$, while it requires $O(r^2)$ multiplications on $\mathbb{F}_q$ to compute $g(y) \bmod f(y)$ for a general polynomial $f(y)$ of degree r .

We remark that, after the computation of

$$\chi = (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta),$$

we do not need the transformation from the indeterminate y to x (i.e., $y \leftarrow x - \alpha$) because χ is an element in $\mathbb{F}_q$.

4. Computational Procedure of Our Method

In this section, we give an efficient procedure for computing

$$\begin{aligned}\chi &= x^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod \{(x - \alpha)^r - \beta\} \\ &= (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta)\end{aligned}$$

described in the previous section.

We represent the exponent above as follows:

$$\begin{aligned}&\frac{q^{r-1} + \dots + q + 1}{r} \\ &= \frac{(q^{r-1} - 1) + \dots + (q - 1) + r}{r} \\ &= \{q^{r-2} + 2q^{r-3} + \dots + (r-2)q + (r-1)\} \frac{q-1}{r} + 1.\end{aligned}$$

Therefore, following the steps below, we compute an r -th root $\chi = (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta)$ of a :

Step 1: Compute

$$g(y) := (y + \alpha)^{q^{r-2} + 2q^{r-3} + \dots + (r-2)q + (r-1)} \bmod (y^r - \beta).$$

Step 2: Compute $h(y) := g(y)^{\frac{q-1}{r}} \bmod (y^r - \beta)$.

Step 3: Compute $\chi = h(y)(y + \alpha) \bmod (y^r - \beta)$.

In the remainder of this section, we consider the complexity of each step above. We evaluate one multiplication

of two polynomials over $\mathbb{F}_q$ of degree at most r as $O(r^{1+\delta})$ operations on $\mathbb{F}_q$ with $0 < \delta \leq 1$ ($\delta = 1$ for the classical method, $\delta = 0.59$ for the Karatsuba method, and $\delta = o(1)$ for the FFT method [4, Definition 8.26]).

For Step 1, from the representation

$$g(y) = (\dots((y + \alpha)^q(y + \alpha)^2) \dots)^q(y + \alpha)^{r-1} \bmod (y^r - \beta),$$

we can compute the polynomial $g(y)$ by performing the q -th power operations and the multiplications by $(y + \alpha)^i$ ($1 \leq i \leq r - 1$) on $\mathbb{F}_q[y]/(y^r - \beta)$. By the assumption $r \mid q - 1$, we first see

$$y^q \bmod (y^r - \beta) = \beta^l y$$

by $q = rl + 1$. Once we compute the value β^l with $O(\log q)$ multiplications on $\mathbb{F}_q$, we obtain the set of values

$$y^{iq} \bmod (y^r - \beta) = (\beta^l)^i y^i$$

for all i 's with $0 \leq i \leq r - 1$ by $O(r)$ multiplications on $\mathbb{F}_q$, and keep these values. Thus we can perform the q -th power operation on $\mathbb{F}_q[y]/(y^r - \beta)$ by $O(r)$ multiplications on $\mathbb{F}_q$ because the q -th power operation acts linearly on $\mathbb{F}_q[y]/(y^r - \beta)$.

Next, it takes $O(\sum_{1 \leq i \leq r-2} i) = O(r^2)$ multiplications on $\mathbb{F}_q$ to compute $(y + \alpha)^i$ for all i 's with $1 \leq i \leq r - 1$. Note that, for $1 \leq i \leq r - 1$, the equation

$$(y + \alpha)^i \bmod (y^r - \beta) = (y + \alpha)^i$$

holds from $\deg(y + \alpha)^i = i < \deg(y^r - \beta)$.

From these evaluations above, we perform Step 1 (i.e., the computation of the polynomial $g(y)$) by $O(r^2)$ multiplications on $\mathbb{F}_q$ and $O(r)$ multiplications on $\mathbb{F}_q[y]/(y^r - \beta)$.

Step 2 requires $O(\log \frac{q-1}{r}) = O(\log q)$ multiplications on $\mathbb{F}_q[y]/(y^r - \beta)$.

For Step 3, let

$$h(y) = b_{r-1}y^{r-1} + \dots + b_1y + b_0.$$

Since χ is the constant term of $h(y)(y + \alpha) \bmod (y^r - \beta)$, we get

$$\chi = \alpha b_0 + \beta b_{r-1}$$

with two multiplications on $\mathbb{F}_q$.

From the considerations above, we evaluate the complexity of the Cipolla-Lehmer method as the sum of $O(r^2)$ operations on $\mathbb{F}_q$ and $O(r + \log q)$ operations on $\mathbb{F}_q[y]/(y^r - \beta)$, which is evaluated as $O((r + \log q)r^{1+\delta})$ operations on $\mathbb{F}_q$. We note that the computation of $\chi = (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta)$ requires $O(r \log q)$ operations on $\mathbb{F}_q[y]/(y^r - \beta)$ by using the repeated square-and-multiply method directly.

5. The Adleman-Manders-Miller Method

In this section, we explain the Adleman-Manders-Miller method [1] (based on the Tonelli-Shanks method [7], [10])

for r -th root computations in finite fields $\mathbb{F}_q$, that is, the computation of χ satisfying $\chi^r = a$ for a given r -th residue element a in $\mathbb{F}_q$ (i.e., $a^{\frac{q-1}{r}} = 1$).

Let $q-1 = r^v s$ with $r \nmid s$ and b be an r -th non-residue in $\mathbb{F}_q$. Setting $g = b^s$ and $h = a^s$, we see that g is a primitive r^v -th root of unity, i.e., the multiplicative group $\langle g \rangle$ generated by g is the Sylow r -subgroup of $\mathbb{F}_q^*$, and that h belongs to the subgroup $\langle g^r \rangle$. If we get an r -th root of h , say α , then we compute an r -th root of a as

$$\chi = \alpha^l a^m,$$

where two integers l, m satisfy $sl + rm = 1$.

In order to obtain α , an r -th root of h , it is sufficient to find c_i 's ($0 \leq c_i < r, 1 \leq i \leq v-1$) such that

$$h = g^{c_1 r + c_2 r^2 + \dots + c_{v-1} r^{v-1}}.$$

Indeed, if we find such c_i 's above, then we obtain

$$\alpha = g^{c_1 + c_2 r + \dots + c_{v-1} r^{v-2}}.$$

When we raise both sides of the equation $h = g^{c_1 r + c_2 r^2 + \dots + c_{v-1} r^{v-1}}$ with unknowns c_i 's to r^{v-2} -th power, we get the equation

$$h^{r^{v-2}} = (g^{r^{v-1}})^{c_1},$$

from which we compute the value c_1 , that is, the solution to the discrete logarithm problem on the subgroup $\langle g^{r^{v-1}} \rangle$ of order r . After getting the value c_1 , we multiply $g^{-c_1 r}$ to the equation $h = g^{c_1 r + c_2 r^2 + \dots + c_{v-1} r^{v-1}}$ to obtain the relation

$$hg^{-c_1 r} = g^{c_2 r^2 + \dots + c_{v-1} r^{v-1}}.$$

We get the value c_2 by raising both sides of this equation to r^{v-3} -th power, after which we multiply $g^{-c_2 r^2}$. By performing the procedure above repeatedly, we get the values c_i 's to compute α .

In the rest of this section, we consider the complexity of the Adleman-Manders-Miller method. We denote by $M(r)$ the number of operations on $\mathbb{F}_q$ to solve a discrete logarithm problem on the subgroup $\langle g^{r^{v-1}} \rangle$ of order r , that is, for example $M(r) = O(r)$ for the exhaustive search and $M(r) = O(\sqrt{r})$ for the ρ method.

Before evaluating the cost of each process in the Adleman-Manders-Miller method, we see the cost for computing the four values (v, s, l, m) is negligible as compared with the costs of the other processes. Therefore, we omit the evaluation of this procedure.

Firstly, from the fact $s \leq q$, it requires $O(\log q)$ operations on $\mathbb{F}_q$ to compute three values b, g and h . We note that the probability that a random element b in $\mathbb{F}_q^*$ is an r -th non-residue is $1 - \frac{1}{r}$. Secondly, it takes

$$\begin{aligned} & \sum_{1 \leq i \leq v-1} \{O(\log r^{v-1-i}) + M(r) + O(\log r)\} \\ &= O(v^2 \log r + vM(r)) \end{aligned}$$

operations on $\mathbb{F}_q$ to compute α (i.e., the values c_i 's). After

that, it requires $O(\log q)$ operations on $\mathbb{F}_q$ to compute the value $\chi = \alpha^l a^m$, an r -th root of a .

Summing up these costs, we obtain an r -th root of a by $O(\log q + v^2 \log r + vM(r))$ operations on $\mathbb{F}_q$ using the Adleman-Manders-Miller method. This implies that, for a fixed size of q , the size of v rather than the size of r influences the computational task of the Adleman-Manders-Miller method, which differs from the case of the Cipolla-Lehmer method, namely, as described in Sect. 4, the complexity of the Cipolla-Lehmer method (i.e., $O((r + \log q)r^{1+\delta})$ operations on $\mathbb{F}_q$) depends only on r and not on v . We further see that the Cipolla-Lehmer method needs operations on the residue ring $\mathbb{F}_q[y]/(y^r - \beta)$, while the Adleman-Manders-Miller method needs operations on only $\mathbb{F}_q$.

6. Experimental Results

In this section, using the following four methods, we implement the r -th root computation in prime finite fields $\mathbb{F}_p$ with the bit sizes of p being 500, 1000 and 2000.

CL 1: The Cipolla-Lehmer method based on the computation

$$\chi = x^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod \{(x - \alpha)^r - \beta\},$$

using the repeated square-and-multiply method.

CL 2: The Cipolla-Lehmer method based on the computation

$$\chi = (y + \alpha)^{\frac{q^{r-1} + \dots + q + 1}{r}} \bmod (y^r - \beta),$$

using the repeated square-and-multiply method.

CL 3: The Cipolla-Lehmer method based on the computation

$$\begin{aligned} \chi &= (y + \alpha)^{\{q^{r-2} + 2 \cdot q^{r-3} + \dots + (r-2)q + (r-1)\} \frac{q-1}{r} + 1} \\ &\bmod (y^r - \beta), \end{aligned}$$

using the procedure described in Sec. 4.

AMM: The Adleman-Manders-Miller method [1].

For each size of p , we implement 100 times r -th root computations. We present the average times in the tables below. All computations are performed on a 2.8 GHz Pentium G6950 with 1.6 GB RAM. The language is C with gcc 4.1.2 compiler and with no mathematical library.

We apply the classical method for multiplications of two polynomials in the Cipolla-Lehmer method, and the exhaustive method for solving the discrete logarithm problem on the subgroup of order r in $\mathbb{F}_p^*$ in the Adleman-Manders-Miller method. For an integer x , by $v_r(x)$ we denote the largest non-negative integer v such that $r^v \mid x$.

6.1 The Cipolla-Lehmer Method

First, we implement the Cipolla-Lehmer method (CL 1 – 3) for some values of r .

[†]Therefore, the Adleman-Manders-Miller method is probabilistic.

Table 3 Running time (s) of the Cipolla-Lehmer method for r -th root in $\mathbb{F}_p$.

p (bit)	method \ r	3	5	7	11	13
500	CL 1	0.15	0.72	1.94	7.47	12.26
	CL 2	0.11	0.46	1.10	3.68	5.73
	CL 3	0.10	0.20	0.32	0.66	0.81
1000	CL 1	1.24	5.11	14.11	53.93	68.56
	CL 2	0.97	3.24	7.97	26.31	41.22
	CL 3	0.86	1.41	2.35	4.64	6.10
2000	CL 1	8.93	39.54	108.05	414.57	680.09
	CL 2	6.91	24.96	60.81	202.33	316.80
	CL 3	5.94	11.01	17.85	36.42	47.33

From Table 3, the procedure (CL 3) becomes more efficient than the others (CL 1), (CL 2) as r grows.

6.2 Comparison between the Cipolla-Lehmer Method and the Adleman-Manders-Miller Method

Next, we implement our Cipolla-Lehmer method (CL 3) and the Adleman-Manders-Miller method (AMM) for several values of r and $v_r(p-1)$. In the tables below, we mainly list the experimental results for the $v_r(p-1)$'s near the cross-over point of the running time of the Cipolla-Lehmer method and that of the Adleman-Manders-Miller method.

The bit size of p is 500:

Table 4 Running time (s) for cubic root in $\mathbb{F}_p$ (p : 500 bit).

$v_3(p-1)$	3	21	41	60	81	100
CL 3	0.10	0.11	0.10	0.10	0.09	0.10
AMM	0.03	0.04	0.07	0.10	0.13	0.16

Table 5 Running time (s) for 5-th root in $\mathbb{F}_p$ (p : 500 bit).

$v_5(p-1)$	2	40	60	82	100	120
CL 3	0.20	0.19	0.19	0.19	0.20	0.20
AMM	0.03	0.06	0.10	0.19	0.22	0.30

Table 6 Running time (s) for 7-th root in $\mathbb{F}_p$ (p : 500 bit).

$v_7(p-1)$	3	61	80	100	121	140
CL 3	0.33	0.32	0.32	0.32	0.34	0.32
AMM	0.03	0.13	0.26	0.28	0.42	0.54

Table 7 Running time (s) for 11-th root in $\mathbb{F}_p$ (p : 500 bit).

$v_{11}(p-1)$	2	100	120	140	160 (p : 561 bit)
CL 3	0.66	0.66	0.64	0.65	0.87
AMM	0.02	0.35	0.49	0.64	1.03

The bit size of p is 1000:

Table 8 Running time (s) for cubic root in $\mathbb{F}_p$ (p : 1000 bit).

$v_3(p-1)$	3	80	100	120	140	162
CL 3	0.86	0.81	0.71	0.73	0.71	0.79
AMM	0.26	0.56	0.68	0.92	1.13	1.43

Table 9 Running time (s) for 5-th root in $\mathbb{F}_p$ (p : 1000 bit).

$v_5(p-1)$	2	100	120	140	160	182
CL 3	1.41	1.48	1.41	1.41	1.41	1.42
AMM	0.21	0.90	1.32	1.53	1.93	2.50

Table 10 Running time (s) for 7-th root in $\mathbb{F}_p$ (p : 1000 bit).

$v_7(p-1)$	2	120	140	160	180	201
CL 3	2.35	2.29	2.27	2.37	2.33	2.29
AMM	0.21	1.55	2.03	2.58	3.21	3.94

Table 11 Running time (s) for 11-th root in $\mathbb{F}_p$ (p : 1000 bit).

$v_{11}(p-1)$	2	160	180	200	220	240
CL 3	4.64	4.67	4.65	4.74	4.74	4.68
AMM	0.25	3.11	3.92	4.74	5.68	6.70

Table 12 Running time (s) for 13-th root in $\mathbb{F}_p$ (p : 1000 bit).

$v_{13}(p-1)$	2	180	200	220	240	260
CL 3	6.10	6.19	6.13	6.21	6.08	6.13
AMM	0.21	3.87	4.72	5.67	6.71	7.82

Table 13 Running time (s) for 17-th root in $\mathbb{F}_p$ (p : 1000 bit).

$v_{17}(p-1)$	2	200	220	240	260 (p : 1072 bit)
CL 3	9.78	9.57	9.70	9.51	11.92
AMM	0.21	4.54	5.47	6.49	8.57

The bit size of p is 2000:

Table 14 Running time (s) for cubic root in $\mathbb{F}_p$ (p : 2000 bit).

$v_3(p-1)$	2	100	120	140	160	181
CL 3	5.94	5.42	6.07	5.38	5.41	5.91
AMM	1.66	3.48	4.58	5.50	6.27	7.62

Table 15 Running time (s) for 5-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_5(p-1)$	2	140	160	180	200	220
CL 3	11.03	11.03	10.92	10.92	10.91	11.00
AMM	1.97	6.79	8.37	10.08	12.13	14.26

Table 16 Running time (s) for 7-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_7(p-1)$	2	160	180	201	220	240
CL 3	17.78	17.80	18.17	17.87	17.97	17.91
AMM	1.67	10.89	13.25	16.11	18.96	22.22

Table 17 Running time (s) for 11-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_{11}(p-1)$	2	240	260	280	300	320
CL 3	36.49	36.13	36.13	36.39	36.01	36.16
AMM	1.66	26.81	31.52	35.90	40.90	46.29

Table 18 Running time (s) for 13-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_{13}(p-1)$	2	280	301	320	340	360
CL 3	47.35	47.70	48.56	48.62	47.89	48.37
AMM	1.66	35.94	41.22	46.36	52.12	58.28

Table 19 Running time (s) for 17-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_{17}(p-1)$	2	380	400	420	440	460
CL 3	75.76	73.98	74.16	76.27	75.41	75.16
AMM	1.65	62.19	68.70	75.52	82.69	90.18

Table 20 Running time (s) for 19-th root in $\mathbb{F}_p$ (p : 2000 bit).

$v_{19}(p-1)$	2	380	400	420	440	460
CL 3	92.35	91.58	91.28	90.43	90.73	92.23
AMM	2.02	76.12	84.44	92.65	101.50	110.76

Table 21 Running time (s) for 23-rd root in $\mathbb{F}_p$ (p : 2000 bit).

$v_{23}(p-1)$	2	400	420	440 (p : 2003 bit)
CL 3	128.90	125.50	127.79	126.12
AMM	1.67	99.62	109.78	120.83

From Tables 4–21, we see the following facts.

- The Adleman-Manders-Miller method is more efficient than the Cipolla-Lehmer method provided that $v_r(p-1)$ is not so large. In the case of p with 500 bit (resp. 1000 bit and 2000 bit), it seems that the Adleman-Manders-Miller method is always more efficient than the Cipolla-Lehmer method for $r \geq 11$ (resp. $r \geq 17$ and $r \geq 23$).
- For a fixed size of p , as pointed out in Sect. 5, the running time of the Adleman-Manders-Miller method depends on the values r and $v_r(p-1)$ and, more precisely, the value $v_r(p-1)$ is more influential than the value r . On the other hand, the running time of the Cipolla-Lehmer method depends only on the value r .
- For a fixed size of r , the value $v_r(p-1)$ of the cross-over point of the running time of the Cipolla-Lehmer method and that of the Adleman-Manders-Miller method becomes large as the size of p grows.

7. Conclusion and Future Works

In this paper, for a prime r with $r \mid q-1$, we considered the computation of r -th roots in finite fields $\mathbb{F}_q$ based on the Cipolla-Lehmer method, and provided an efficient computational procedure for realizing our method. We further implemented our method (based on the Cipolla-Lehmer method) and the Adleman-Manders-Miller method.

Our future works are, for example

- the evaluation of the probability that the element $(-1)^r(\alpha^r - a)$ is an r -th non-residue for a given r -th residue element a and a random element α in $\mathbb{F}_q$ as r becomes large;
- the feasibility of r -th root computations in $\mathbb{F}_q$ as r becomes large;
- the theoretical comparison of our method with the Adleman-Manders-Miller method;
- the generalization to n -th roots with n composite without applying the factorization of n ;
- the existence of an efficient deterministic algorithm for

root computations in finite fields.

We mention that, for the last topic in the list of our future works, T. Sze recently proposed a deterministic algorithm for root computations in finite fields [9]. The idea is to reduce the r -th root computation of an r -th residue element a in $\mathbb{F}_q$ to the computation of a non-trivial monic factor of $x^r - a$. Indeed, let $g(x)$ be a non-trivial monic factor of $x^r - a$, and n (resp. c) the degree of $g(x)$ (resp. the constant term of $g(x)$). Similar to the latter part of the proof of Prop. 1, we have

$$c = (-1)^n \zeta^k \gamma^n$$

for some integer k , where $\zeta \in \mathbb{F}_q^*$ (resp. γ) is a primitive r -th root of unity (resp. an r -th root of a). Then we see the element

$$(-1)^{ln} c^l a^m = \zeta^{kl} \gamma$$

is an r -th root of a , where l and m are integers with $ln + mr = 1$. Setting $q-1 = r^s s$ with $r \nmid s$, we evaluate the complexity as $O((r(r+s) + \log q)r^{1+\delta})$ operations on $\mathbb{F}_q$ [9, Theorem 3.22]. Therefore, the method is efficient if both r and s are sufficiently small as compared with q (e.g., $O(\log q)$).

Acknowledgments

We are grateful to the referees for giving a lot of useful comments and suggestions which make this paper more valuable.

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.175–178, 1977.
- [2] P.S.L.M. Barreto and J.F. Voloch, "Efficient computation of roots in finite fields," Des. Codes Cryptogr., vol.39, pp.275–280, 2006.
- [3] M. Cipolla, "Un metodo per la risoluzione della congruenza di secondo grado," Rendiconto dell'Accademia delle Scienze Fisiche e Matematiche, Napoli, vol.9, pp.154–163, 1903.
- [4] J. von zur Gathen and J. Gerhard, Modern Computer Algebra, Second ed., Cambridge, 2003.
- [5] D. H. Lehmer, Computer technology applied to the theory of numbers, Studies in Number Theory, pp.117–151, Prentice-Hall, Englewood Cliffs, NJ, 1969.
- [6] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.
- [7] D. Shanks, Five number-theoretic algorithms, Proc. 2nd Manitoba Conf. Numer. Math., pp.51–70, Manitoba, Canada, 1972.
- [8] H. Stichtenoth, Algebraic Function Fields and Codes, Springer Universitext, Springer-Verlag, Berlin Heidelberg, 1993.
- [9] T. Sze, On taking r -th roots without r -th nonresidues over finite fields and its applications, preprint, <http://arxiv.org/pdf/1105.5852v1.pdf>, 2011.
- [10] A. Tonelli, Bemerkung über die Auflösung quadratischer Congruenzen, pp.344–346, Göttinger Nachrichten, 1891.

Appendix: The Fermat Curve

Unless explicitly stated otherwise, we assume that r is prime

with $r \mid q - 1$. In this appendix, for an r -th residue a in $\mathbb{F}_q$, we evaluate the number

$$N := \#\{x \in \mathbb{F}_q \mid (-1)^r(x^r - a) \text{ is an } r\text{-th non-residue}\}.$$

Let $a = h^r$ with $h \in \mathbb{F}_q^*$. Since we have

$$(-1)^r(x^r - a) = (-h^{-1}x)^r - 1,$$

we consider the number of $\mathbb{F}_q$ -rational points on the curve

$$C/\mathbb{F}_q : x^r - 1 = y^r.$$

If we set $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$, the defining equation of $C/\mathbb{F}_q$ becomes

$$X^r = Y^r + Z^r$$

(the Fermat curve of genus $g = \frac{(r-1)(r-2)}{2}$). We see that the set of $\mathbb{F}_q$ -rational points of C , say $C(\mathbb{F}_q)$, has r points at infinity

$$[X : Y : Z] = [1 : \zeta^i : 0] \quad (0 \leq i < r),$$

where ζ is a primitive r -th root of unity. For a given x in $\mathbb{F}_q$, the number of y 's in $\mathbb{F}_q$ satisfying $x^r - 1 = y^r$ is as follows:

$$\begin{cases} 0 & (x^r - 1: r\text{-th non-residue}), \\ 1 & (x^r - 1 = 0, \text{ i.e. } x = \omega^i), \\ r & (x^r - 1: r\text{-th residue}). \end{cases}$$

Therefore, we obtain

$$\begin{aligned} N &= q - \{r + (\#C(\mathbb{F}_q) - 2r)/r\} \\ &= q - \#C(\mathbb{F}_q)/r - (r - 2). \end{aligned}$$

Since the Fermat curve $C/\mathbb{F}_q$ has genus $g = \frac{(r-1)(r-2)}{2}$ in the case of $\text{char } \mathbb{F}_q \nmid r$ [8], we have

$$|\#C(\mathbb{F}_q) - (q + 1)| \leq 2g\sqrt{q} = (r - 1)(r - 2)\sqrt{q}$$

from the Hasse-Weil theorem [8]. Therefore, the number N is approximately $(1 - \frac{1}{r})q$ for sufficiently large q compared with r (e.g., $r = O(\log q)$).

Yutaka Sueyoshi received the B.S., M.S. and D.S. degrees in mathematics from Kyushu University, Japan, in 1975, 1977 and 1992, respectively. From 1979 to 1994, he was with the Department of Mathematics at Kyushu University. From 1994 to 2000, he was a Lecturer in the Graduate School of Mathematics at Kyushu University. From 2000 to 2012, he was an Associate Professor in the Department of Computer and Information Sciences at Nagasaki University. Since 2012, he has been a Professor in the Graduate School of Engineering at Nagasaki University. His research interests are in algebraic number theory, class field theory and algebraic theory of cryptography. Dr. Sueyoshi is a member of the Mathematical Society of Japan, the Japan Society for Industrial and Applied Mathematics, the Information Processing Society of Japan and American Mathematical Society.

Aichi Kudo received the B.S., M.S. and D.S. degrees in mathematics from Kyushu University, Japan, in 1970, 1972 and 1977, respectively. He was with Kyushu University from 1977 to 1978, and with Nagoya University from 1978 to 1988. Since 1988, he has been a Professor in Nagasaki University. Since 1998, he has been with the Graduate School of Engineering at Nagasaki University. His research interests are in algebraic number theory, p -adic analysis, computational algorithm and algebraic theory of

cryptography. Dr. Kudo is a member of the Mathematical Society of Japan and the Japan Society for Industrial and Applied Mathematics.

Ryuichi Harasawa received the B.S. degree in mathematics from Osaka City University, Japan, in 1997, and the M.S. and D.S. degrees in mathematics from Osaka University, Japan in 1999 and 2003, respectively. From 2003 to 2007, he was a Research Associate in the Department of Computer and Information Sciences at Nagasaki University. Since 2007, he has been an Assistant Professor in the Graduate School of Engineering at Nagasaki University. His research interests are in algebraic curve

cryptography and its applications, provable security of cryptosystems, and computer algebra. Dr. Harasawa is a member of the Mathematical Society of Japan and the Japan Society for Industrial and Applied Mathematics.