On r-th Root Extraction Algorithm in $\mathbb{F}_q$ For $q \equiv lr^s + 1 \pmod{r^{s+1}}$ with $0 < l < r$ and Small s

Namhun Koo, Gook Hwa Cho, and Soonhak Kwon
Dept. of Mathematics, Sungkyunkwan University, Suwon, S. Korea
komaton@skku.edu, achimheasal@nate.com, shkwon@skku.edu

Abstract

We present an r-th root extraction algorithm over a finite field $\mathbb{F}_q$. Our algorithm precomputes a primitive r^s-th root of unity ξ where s is the largest positive integer satisfying $r^s | q-1$, and is applicable for the cases when s is small. The proposed algorithm requires one exponentiation for the r-th root computation and is favorably compared to the existing algorithms.

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

1 Introduction

Let $r > 1$ be an integer. There are two well-known algorithms for r-th root computation in finite field $\mathbb{F}_q$; the Adleman-Manders-Miller algorithm [1, 2, 3, 6] (a natural extension of the Tonelli-Shanks square root algorithm) and the Cipolla-Lehmer [4, 5] algorithms. Assuming $r << \log q$, the Adleman-Manders-Miller algorithm depends on the exponent s of r satisfying $r^s | q-1$ and $r^{s+1} \not| q-1$, which makes the worst case complexity of the Adleman-Manders-Miller $O(\log r \log^4 q)$ [3, 6] while the Cipolla-Lehmer can be executed in $O(r \log^3 q)$ [4, 5].

However, due to the cumbersome extension field arithmetic needed for the Cipolla-Lehmer algorithm, the Adleman-Manders-Miller performs better than the Cipolla-Lehmer for small exponents s, and related research can be found in [10, 11].

On the other hand, when the exponent s is small, there are other approaches for finding a square root using one or two exponentiations, which are faster than the Tonelli-Shanks for some cases. One such example is a square root of c in $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$. That is, when c is a quadratic residue in $\mathbb{F}_q$, then a square root of c is given as $c^{\frac{q+1}{4}}$ since one can directly verify $(c^{\frac{q+1}{4}})^2 = c$. When $s = 2, 3$, there are similar approaches due to Atkin [7], Müller [8] and Kong et al. [9], and their methods have better performance when compared with the Tonelli-Shanks and the Cipolla-Lehmer.

However it seems that, these 'exponentiation only' approach is not well studied for general r-th root extraction problem in $\mathbb{F}_q$. Our aim is to generalize this 'exponentiation only' approach for r-th root extraction and give concrete examples. We present a new r-th root extraction algorithm using a precomputed primitive element in $\mathbb{F}_q$ and show that our algorithm requires only one exponentiation in $\mathbb{F}_q$ and efficient when s is small.

2 Existing root extraction algorithms for small exponent

2.1 Square root algorithms

There are some efficient square root extraction formulas for $q \equiv 5 \pmod 8$ and $q \equiv 9 \pmod{16}$ respectively. If $q \equiv 5 \pmod 8$, a special case of $q \equiv 1 \pmod 4$, there is an efficient square root algorithm [7] due to Atkin which uses only one exponentiation.

Algorithm 1 Atkin's algorithm when $q \equiv 5 \pmod 8$

Input : A square a in $\mathbb{F}_q$

Output : x satisfying $x^2 = a$ in $\mathbb{F}_q$

1: $b \leftarrow (2a)^{\frac{q-5}{8}}$
2: $i \leftarrow 2ab^2$
3: $x \leftarrow ab(i-1)$
4: **return** x.

Algorithm 1 needs 1 exponentiation and 4 multiplications, and is more efficient than the Tonelli-Shank's or the Cipolla-Lehmer. Müller [8] extended Atkin's idea to the case $q \equiv 9 \pmod{16}$. Müller's algorithm is probabilistic and requires 2 exponentiations. Details of the algorithm is given in Algorithm 2. Note that the probabilistic step finds $d \in \mathbb{F}_q$ satisfying $\eta(d) = -b$, where the quadratic character η satisfies $\eta(d) = 1$ if d is a square in $\mathbb{F}_q$, and $\eta(d) = -1$ if not. A further speed-up for the case $q \equiv 9 \pmod{16}$ is given by Kong et al. [9] where the average number of exponentiation is reduced to 1.5.

2.2 r-th root extraction algorithms for $r > 2$

For $r > 2$ there are no available simple algorithms similar to the algorithms of Atkin, Müller, and Kong et al. applicable to special types of q. Of course, we wish that such simple algorithms have better performance compared with general algorithms such as the Adleman-Manders-Miller and the Cipolla-Lehmer. In the following sections, we will show that there are natural

Algorithm 2 Müller's algorithm when $q \equiv 9 \pmod{16}$

Input : A square a in $\mathbb{F}_q$

Output : x satisfying $x^2 = a$ in $\mathbb{F}_q$

1: $b \leftarrow (2a)^{\frac{q-1}{4}}$
2: Find randomly d satisfying $-b = \eta(d)$
3: $u \leftarrow (2ad^2)^{\frac{q-9}{16}}$
4: $i \leftarrow 2u^2d^2a$
5: $x \leftarrow uda(i-1)$
6: **return** x.

generalization of the previously proposed ideas on the square root algorithms to the case of general r-th root extractions.

3 New r-th root extraction formula over $\mathbb{F}_q$ for $q \equiv lr^s + 1 \pmod{r^{s+1}}$

Let q be a power of a prime and let r be a prime dividing $q-1$. The reason why we only consider the case $r|q-1$ is as follows. If $r \not| q-1$, then one has the greatest common divisor $gcd(r, q-1) = 1$ 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.

Let s be the largest positive integer satisfying $r^s|q-1$ with $s \geq 1$. Since $\frac{q-1}{r^s} \equiv l \pmod{r}$ for some l with $0 < l < r$, one has $q \equiv lr^s + 1 \pmod{r^{s+1}}$. In other words, for any prime or prime power q and integer $r > 1$ with $r|q-1$, there exist unique positive integer s and $0 < l < r$ satisfying $q \equiv lr^s + 1 \pmod{r^{s+1}}$, where such unique s is the largest positive integer satisfying $r^s|q-1$.

Theorem 1. *For given r-th power c in $\mathbb{F}_q$, there exists $b \in \mathbb{F}_q$ such that $c^{r-1}b^r$ has an order r^t with $t < s$.*

Proof. We will use two elements α, β satisfying $r\beta + r - 1 = l\alpha$. Such α, β can be found as follows. Since $gcd(l, r) = 1$, there exists $0 \leq \beta < l$ satisfying $r\beta + r - 1 \equiv 0 \pmod{l}$ and letting $\alpha = \frac{r\beta+r-1}{l}$ we have $r\beta + r - 1 = l\alpha$. Therefore letting $\zeta = (c^\alpha)^{\frac{q-1}{r^s}}$,

$$\begin{aligned}\zeta &= (c^\alpha)^{\frac{q-1}{r^s}} \\ &= (c^\alpha)^{\frac{q-1}{r^s}} c^{r\beta+r-1-l\alpha} = c^{r-1}c^{r\beta}(c^\alpha)^{\frac{q-1}{r^s}}c^{-l\alpha} \\ &= c^{r-1}\{c^\beta(c^\alpha)^{\frac{q-1-lr^s}{r^{s+1}}}\}^r = c^{r-1}b^r, \end{aligned} \tag{1}$$

where we put $b = c^{\beta}(c^{\alpha})^{\frac{q-1-lr^s}{r^{s+1}}}$. Since c is an r-th power in $\mathbb{F}_q$, the order of ζ is r^t for some $0 \leq t < s$. □

Let ξ be a primitive r^s-th root of unity in $\mathbb{F}_q$, which will be computed once and will be fixed throughout this paper. Such ξ can be found by letting $\xi = d^{\frac{q-1}{r^s}}$ where d is not an r-th power in $\mathbb{F}_q$. Therefore our method is probabilistic (i.e., randomized). The heuristic probability that a randomly chosen $d \in \mathbb{F}_q$ is not an r-th power in $\mathbb{F}_q$ is $\frac{r-1}{r}$.

Since $\xi^{r^{s-t}}$ is a primitive r^t-th root of unity, there exist unique i and j determined $\pmod{r^t}$ such that

$$\xi^{r^{s-t}} = \zeta^i, \quad \zeta = (\xi^{r^{s-t}})^j \tag{2}$$

From $\zeta = (\xi^{r^{s-t}})^j = \zeta^{ij}$, we have

$$ij \equiv 1 \pmod{r^t} \tag{3}$$

Now we present our new theorem which states that an r-th root of an r-th power residue c can be found using one exponentiation under suitable conditions.

Theorem 2. *Define u as $u \equiv j(r^t-1)r^{s-t-1} \equiv -jr^{s-t-1} \pmod{r^{s-1}}$. Then an r-th root of c in $\mathbb{F}_q$ is given as $cb\xi^u$ where b is given in Theorem 1.*

Proof. Letting $x = cb\xi^u$,

$$x^r = c \cdot c^{r-1}b^r \cdot \xi^{ru} = c \cdot \zeta \cdot \xi^{ru}$$

Since $u = j(r^t-1)r^{s-t-1} + r^{s-1}k$ for some integer k and using $\xi^{r^s} = 1$,

$$\begin{aligned}
\zeta\xi^{ru} &= (\xi^{r^{s-t}})^j \cdot \xi^{ru} = \xi^{j \cdot r^{s-t} + ru} && (4)\\
&= \xi^{j \cdot r^{s-t} + j(r^t-1)r^{s-t} + r^s k} && (5)\\
&= \xi^{j \cdot r^{s-t} + j(r^t-1)r^{s-t}} = \xi^{j \cdot r^{s-t}(1+r^t-1)} && (6)\\
&= \xi^{j \cdot r^s} = 1 && (7)
\end{aligned}$$

Therefore one has $x^r = c \cdot \zeta\xi^{ru} = c$. □

Remark 1. $r\beta + r - 1 = l\alpha$ implies $r(\beta + l) + r - 1 = l(\alpha + r)$. That is, α is determined $\pmod{r}$ and β is determined $\pmod{l}$. Thus for any α satisfying $\alpha\frac{q-1}{r^s} \equiv \alpha l \equiv -1 \pmod{r}$,

there is uniquely determined β satisfying $r\beta + r - 1 = l\alpha$ (i.e., $\beta = \frac{l\alpha+1-r}{r}$). Thus we may fix α first and determine the corresponding β. In fact, the condition $r\beta + r - 1 - l\alpha = 0$ can be weakened to $r\beta + r - 1 - l\alpha \equiv 0 \pmod{\frac{q-1}{r}}$ because $c^{\frac{q-1}{r}} = 1$.

Remark 2. The value cb can be simplified as follows.

$$cb = c \cdot c^{\beta}(c^{\alpha})^{\frac{q-1-lr^s}{r^{s+1}}} \tag{8}$$

$$= c^{\frac{r^{s+1}+\beta r^{s+1}+\alpha(q-1)-\alpha l r^s}{r^{s+1}}} \tag{9}$$

$$= c^{\frac{r^s(r+\beta r-l\alpha)+\alpha(q-1)}{r^{s+1}}} \tag{10}$$

$$= c^{\frac{r^s+\alpha(q-1)}{r^{s+1}}} = c^{\frac{1+\alpha\frac{q-1}{r^s}}{r}}, \tag{11}$$

where α is an integer with $0 < \alpha < r$ satisfying $1 + \alpha\frac{q-1}{r^s} \equiv 0 \pmod{r}$. Therefore one has an alternate expression of b (without using β) as $b = cb/c = c^{\frac{1+\alpha\frac{q-1}{r^s}-r}{r}}$.

Remark 3. Finding i (or j) in the equation (2) is in general difficult if r^t is large (i.e., if $t > 1$ and the discrete logarithm in the cyclic group of order r^t is intractable). So our method is useful only when r and t are small, and can be viewed as a parallelized version of the Adleman-Manders-Miller algorithm.

4 Examples and Algorithms

In this section, using the result of previous section, we will give some examples and algorithms for efficient r-th root computation.

Example 1. **r-th root for $q \equiv lr + 1 \pmod{r^2}$ with $0 < l < r$:**

This is the case $s = 1$. Therefore we get $t = 0$ and $u = 0$, and an r-th root of c is given as $cb = c^{\frac{1+\alpha\frac{q-1}{r}}{r}}$, where $\alpha \in \mathbb{F}_q$ satisfies $1 + \alpha\frac{q-1}{r} \equiv 0 \pmod{r}$.

When $r = 2$, the condition $s = 1$ implies that $q \equiv 3 \pmod 4$ and a square root of c is given as $c^{\frac{1+1\cdot\frac{q-1}{2}}{2}} = c^{\frac{q+3}{4}}$ which is well known.

When $r = 3$, the condition $q \equiv lr + 1 \pmod{r^2}$ (i.e., $s = 1$) implies that either $q \equiv 4 \pmod 9$ or $q \equiv 7 \pmod 9$ depending on the value $l = 1, 2$. Thus a cube root of c is given by $c^{\frac{1+2\frac{q-1}{3}}{3}} = c^{\frac{2q+1}{9}}$ when $q \equiv 4 \pmod 9$, and $c^{\frac{1+1\frac{q-1}{3}}{3}} = c^{\frac{q+2}{9}}$ when $q \equiv 7 \pmod 9$. Note that the cases $q \equiv 4, 7 \pmod 9$ cover $\frac{2}{3} \approx 66.7\%$ of all primes of the form $q \equiv 1 \pmod 3$.

Table 1. r-th roots of c over $\mathbb{F}_q$ for $r = 3, 5, 7$

$r = 3$	$r = 5$	$r = 7$
$q \equiv 4 \pmod 9$: $c^{\frac{2q+1}{9}}$	$q \equiv 6 \pmod{25}$: $c^{\frac{4q+1}{25}}$	$q \equiv 8 \pmod{49}$: $c^{\frac{6q+1}{49}}$
$q \equiv 7 \pmod 9$: $c^{\frac{q+2}{9}}$	$q \equiv 11 \pmod{25}$: $c^{\frac{2q+3}{25}}$	$q \equiv 15 \pmod{49}$: $c^{\frac{3q+4}{49}}$
	$q \equiv 16 \pmod{25}$: $c^{\frac{3q+2}{25}}$	$q \equiv 22 \pmod{49}$: $c^{\frac{2q+5}{9}}$
	$q \equiv 21 \pmod{25}$: $c^{\frac{q+4}{25}}$	$q \equiv 29 \pmod{49}$: $c^{\frac{5q+2}{9}}$
		$q \equiv 36 \pmod{49}$: $c^{\frac{4q+3}{49}}$
		$q \equiv 43 \pmod{49}$: $c^{\frac{q+6}{49}}$
Exceptional cases :		
$q \equiv 1 \pmod 9$	$q \equiv 1 \pmod{25}$	$q \equiv 1 \pmod{49}$

In general, for any prime r, the case $s = 1$ covers $\frac{r-1}{r}$ of all primes q with $q \equiv 1 \pmod r$ because the prime $q \equiv lr+1 \pmod{r^2}$ with $l = 1, 2, \cdots, r-1$ covers all possible primes except the case $q \equiv 1 \pmod{r^2}$ (i.e., $l = 0$). Therefore the method of single exponentiation covers most of the cases as r becomes larger. Table 1 shows r-th root of r-th power residue of $c \in \mathbb{F}_q$ for the cases $r = 3, 5, 7$. Tables of the other cases $r > 7$ can also be constructed similarly.

Example 2. **r-th root for $q \equiv lr^2 + 1 \pmod{r^3}$ with $0 < l < r$:**

This is the case $s = 2$. Therefore $\zeta = (c^\alpha)^{\frac{q-1}{r^2}} = 1$ or a primitive r-th root of unity, i.e., ζ is of order r^t with $t = 0$ or 1. Also ξ is a primitive r^2-th root of unity satisfying $\xi^{r^{2-t}} = \zeta^i$ and $(\xi^{r^{2-t}})^j = \zeta$ with $ij \equiv 1 \pmod{r^t}$. Thus an r-th root of c is given as $cb\xi^u$ where $u \equiv j(r^t - 1)r^{1-t} \equiv -jr^{1-t} \pmod r$ and cb is given in Remark 2. When $t = 0$, one has $u = 0$ and $x = cb$ is a square root, and when $t = 1$, one has $u \equiv -j \pmod r$ and a square root is given $x = cb\xi^{-j}$.

Algorithm 3 Our cube root algorithm when $q \equiv 1 \pmod 9$ and $q \not\equiv 1 \pmod{27}$

Input : A cube c in $\mathbb{F}_q$ $\quad (q = 9l + 1 \pmod{27}$ with $l = 1, 2$, i.e., $q \equiv 10, 19 \pmod{27})$

Output : x satisfying $x^3 = c$ in $\mathbb{F}_q$

1: $b \leftarrow c^{(3-l)\frac{q-(9l+1)}{27}}$
2: $X \leftarrow cb, \zeta \leftarrow X^2 b, A \leftarrow \xi, B \leftarrow A^3$ $\quad (X = c^{\frac{2q+7}{27}}$ if $q \equiv 10 \pmod{27}$ and $X = c^{\frac{q+8}{27}}$ if $q \equiv 19 \pmod{27})$
3: **if** $\zeta = 1$ **then** $x \leftarrow X$
4: **else if** $\zeta = B$ **then** $x \leftarrow XA^2$
5: **else** **then** $x \leftarrow XA$
6: **return** x.

When $r = 3$, from the Remark 2, one has $b = c^{\frac{1+\alpha\frac{q-1}{9}-3}{3}}$. Thus one has $b = c^{\frac{2q-20}{27}}$ when $\frac{q-1}{9} \equiv 1 \pmod 3$, and $b = c^{\frac{q-19}{27}}$ when $\frac{q-1}{9} \equiv 2 \pmod 3$. Algorithm 3 shows the proposed cube root algorithm for the case $s = 2$. Algorithm 3 requires only 1 exponentiation (in step 1) plus at most 6 multiplications in $\mathbb{F}_q$. Table 1 and Algorithm 3 (i.e., combining the cases $s = 1$ and $s = 2$) show that a cube root can be found using just 1 exponentiation for $\frac{2}{3} + \frac{1}{3} \cdot \frac{2}{3} = \frac{8}{9} \approx 89\%$ of all primes of the form $q \equiv 1 \pmod 3$. When $q \equiv 2 \pmod 3$, we already mentioned that the cost of cube root computation is 1 exponentiation in the beginning of Section 3. Thus we conclude that the cost of cube root computation is 1 exponentiation for $\frac{1}{2} + \frac{1}{2} \cdot \frac{8}{9} = \frac{17}{18} \approx 94\%$ of all primes q.

Algorithm 4 Our 5-th root algorithm when $q \equiv 1 \pmod{25}$ and $q \not\equiv 1 \pmod{125}$

Input : A fifth power c in $\mathbb{F}_q$ ($q = 25l + 1 \pmod{125}$ with $l = 1, 2, 3, 4$, i.e., $q \equiv 26, 51, 76, 101 \pmod{125}$)

Output : x satisfying $x^5 = c$ in $\mathbb{F}_q$

1: Compute $0 < \alpha < 5$ satisfying $1 + \alpha l \equiv 0 \pmod 5$ (i.e., $(l, \alpha) = (1, 4), (2, 2), (3, 3), (4, 1)$)
2: $b \leftarrow c^{\frac{\alpha(q-1)-100}{125}}$
3: $X \leftarrow cb, \zeta \leftarrow X^4 b, A \leftarrow \xi, B \leftarrow A^2, C \leftarrow AB^2, D \leftarrow C^2$
4: **if** $\zeta = 1$ **then** $x \leftarrow X$
5: **if** $\zeta = C$ **then** $x \leftarrow XB^2$
6: **if** $\zeta = D$ **then** $x \leftarrow XAB$
7: **if** $\zeta = CD$ **then** $x \leftarrow XB$
8: **if** $\zeta = D^2$ **then** $x \leftarrow XA$
9: **return** x.

When $r = 5$, similarly one has $b = c^{\frac{1+\alpha\frac{q-1}{25}-5}{5}} = c^{\frac{\alpha(q-1)-100}{125}}$, where $\alpha \cdot \frac{q-1}{25} \equiv -1 \pmod 5$. Algorithm 4 shows the proposed 5-th root algorithm for the case $s = 2$. Algorithm 4 also requires only 1 exponentiation (in step 2) plus at most 10 multiplications in $\mathbb{F}_q$. In a similar manner, combining Table 1 and Algorithm 4 show that the cost of 5-th root computation over $\mathbb{F}_q$ is 1 exponentiation for $\frac{4}{5} + \frac{1}{5} \cdot \frac{4}{5} = \frac{24}{25} = 96\%$ of all primes of the form $q \equiv 1 \pmod 5$, and also for $\frac{3}{4} + \frac{1}{4} \cdot \frac{24}{25} = \frac{99}{100} = 99\%$ of all primes q.

Example 3. ***r*-th root for $q \equiv lr^3 + 1 \pmod{r^4}$ with $0 < l < r$:**

This is the case $s = 3$. Therefore $\zeta = (c^\alpha)^{\frac{q-1}{r^3}} = 1$ has order r^t with $t = 0, 1, 2$ and ξ is a primitive r^3-th root of unity satisfying $\xi^{r^{3-t}} = \zeta^i$ and $(\xi^{r^{3-t}})^j = \zeta$ with $ij \equiv 1 \pmod{r^t}$. Thus an r-th root of c is given as $cb\xi^u$ where $u \equiv j(r^t - 1)r^{2-t} \equiv -jr^{2-t} \pmod{r^2}$ where cb is given

in Remark 1. When $t = 0$, one has $u = 0$ and $x = cb$ is an r-th root. When $t = 1$, one has $u \equiv -jr \pmod{r^2}$, and when $t = 2$, one has $u \equiv -j \pmod{r^2}$, and an r-th root is given as $x = cb\xi^u$. Similarly as in Example 2, one can construct algorithms for r-th root (like the cases of Algorithm 3 and 4), and the number of cases (of u) we need to consider is r^2. That is, when $r = 3$, we have to consider 9 cases, and when $r = 5$, we have to consider 25 cases.

Table 2. Comparison of timing using MAPLE (in seconds)

	Our Algorithm		AMM [3, 6]		Cipolla-Lehmer [4, 5]	
	cubic	quintic	cubic	quintic	cubic	quintic
$q \approx 2^{1024}$ with $s = 1$	0.140	0.141	0.452	0.546	2.356	8.081
$q \approx 2^{1024}$ with $s = 2$	0.156	0.156	0.468	0.561	2.351	8.077
$q \approx 2^{2048}$ with $s = 1$	0.655	0.671	2.364	2.637	10.795	40.763
$q \approx 2^{2048}$ with $s = 2$	0.681	0.690	2.387	2.698	10.764	41.153

Table 3. Comparison of Timing using SAGE (in seconds)

	Our Algorithm		AMM [3, 6]		Cipolla-Lehmer [4, 5]	
	cubic	quintic	cubic	quintic	cubic	quintic
$p \approx 2^{1024}$ with $s = 1$	0.014	0.014	0.026	0.027	0.350	0.891
$p \approx 2^{1024}$ with $s = 2$	0.014	0.015	0.026	0.032	0.351	0.867
$p \approx 2^{2048}$ with $s = 1$	0.044	0.048	0.080	0.094	1.165	3.360
$p \approx 2^{2048}$ with $s = 2$	0.046	0.048	0.093	0.095	1.162	3.362

Tables 2 and 3 show the comparison of the implementation results of our algorithm, the AMM (Adleman-Manders-Miller) algorithm [3, 6] and the Cipolla-Lehmer algorithm [4, 5]. For the implementation, we used two available softwares for symbolic computation, MAPLE and SAGE, where SAGE is a newly appeared (since 2005) open source software based on Python language [12]. The computations were performed on 2.80Ghz CPU with 8GB RAM, where primes p of sizes 1024-bit and 2048-bit were chosen. Our method uses the formulas in Table 1 for the case $s = 1$, and uses Algorithm 3 (cubic) and Algorithm 4 (quintic) for the case $s = 2$. For a fair comparison, we did not take the timing of primitive root finding in AMM into account (because the precomputation can also be used in AMM). Also we used trinomials for Cipolla-Lehmer and the timing of the irreducibility testing is not taken into account either.

The comparison shows that our algorithms (of Table 1, Algorithms 3 and 4) have significant advantage over the AMM and the Cipolla-Lehmer. It should be mentioned that the complexity of the Cipolla-Lehmer does not depend on s but on the finite field extension $\mathbb{F}_{q^r}/\mathbb{F}_q$, as one sees the timing of Cipolla-Lehmer for $r = 5$ is quite slower than the timing for $r = 3$.

We expect that our algorithms are only efficient when s is small because the number of cases of ζ increases exponentially with respect to s. However, as is mentioned already, the cases $s = 1, 2$ already cover 94% (for cubic) and 99% (for quintic) of all primes q, and our algorithms can be used in most of the situations.

5 Conclusion

We proposed a new r-th root extraction method in $\mathbb{F}_q$ for $q \equiv lr^s + 1 \pmod{r^{s+1}}$ with $0 < l < r$ which can be successfully implemented for small s. We presented our algorithms for the cubic and quintic cases with $s = 1, 2$, and the given algorithms need only one exponentiation. For the case $s = 1$, we did not need any precomputation but, when $s \geq 1$, a precomputation of ξ was needed. The implementation results imply that the proposed algorithms have significant advantage over the existing algorithms. Our algorithm is useful for relatively small values of s since the number of cases we need to consider is r^{s-1}, which will increase exponentially as s gets larger.

References

[1] D. Shanks "Five Number-Theoretic Algorithms," *Proceeding of Second Manitoba Conference of Numerical Mathematics*, 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", In *Studies in Number Theory*, Prentice-Hall Enblewood Cliffs, NJ pp.117-151, 1969.

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

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

[8] S. Müller, "On the Computation of Square Roots in Finite Fields", *Designs, Codes and Cryptography*, vol. 31, pp. 301-312, 2004.

[9] 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.

[10] D. Han, D. Choi, and H. Kim, "Improved Computation of Square roots in Specific Finite Fields", *IEEE Transactions on Computers*, vol. 58, no. 2, pp.188-196, 2009.

[11] 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.

[12] SAGE, available at http://www.sagemath.org/index.html.