Bull. Korean Math. Soc. **56** (2019), No. 2, pp. 277–283
https://doi.org/10.4134/BKMS.b160769
pISSN: 1015-8634 / eISSN: 2234-3016

IMPROVING THE POCKLINGTON AND PADRÓ-SÁEZ CUBE ROOT ALGORITHM

Gook Hwa Cho and Hyang-Sook Lee

Abstract. In this paper, we present a cube root algorithm using a recurrence relation. Additionally, we compare the implementations of the Pocklington and Padró-Sáez algorithm with the Adleman-Manders-Miller algorithm. With the recurrence relations, we improve the Pocklington and Padró-Sáez algorithm by using a smaller base for exponentiation. Our method can reduce the average number of $\mathbb{F}_q$ multiplications.

1. Introduction

There are two standard algorithms for computing cube roots in finite fields; the Adleman-Manders-Miller algorithm [1, 2, 8, 9] and the Cipolla-Lehmer algorithm [3, 5]. The Pocklington and Padró-Sáez algorithm [6, 7] can also be used, although it is a different type of algorithm from the previously mentioned algorithms. Pocklington [7] proposed a square and cube root algorithm over $\mathbb{F}_p$ with prime p. Heo et al. [4] clarified and generalized the Pocklington and Padró-Sáez algorithm [6, 7]. Thus, Heo et al. [4] proposed a cube root algorithm in $\mathbb{F}_q$, where q is a power of a prime p.

In this paper, we present a cube root algorithm using a recurrence relation. We also provide the results of software implementations (using SAGE) of the Pocklington and Padró-Sáez algorithm [6, 7] compared with those of the Adleman-Manders-Miller algorithm [1, 2, 8, 9]. Our method can reduce the average number of $\mathbb{F}_q$ multiplications.

The remainder of this paper is organized as follows: In Section 2, we describe the Adleman-Manders-Miller algorithm [1, 2, 8, 9] and refined the Pocklington and Padró-Sáez [6, 7] for cube root computation. In Section 3, we implement our method with the Adleman-Manders-Miller algorithm [1, 2, 8, 9] and compare the results. In Section 4, we conclude the paper and discuss future works.

Received September 22, 2016; Revised February 6, 2019; Accepted March 4, 2019.
2010 *Mathematics Subject Classification.* 11T06, 11Y16, 68W40.
Key words and phrases. cube root algorithm, finite field, Pocklington algorithm, Adleman-Manders-Miller algorithm, Cipolla-Lehmer algorithm.

2. Computation of cube roots in finite fields

In this section, we consider a cube root algorithm in finite fields. We describe the Adleman-Manders-Miller algorithm [1,2,8,9] over prime fields. As is known, the Adleman-Manders-Miller algorithm [1,2,8,9] can be implemented over general finite fields. However, the algorithm is more complex when applied to prime fields, as well as the Pocklington and Padró-Sáez algorithm [6,7]. Subsequently, we describe the implementation of the Pocklington and Padró-Sáez algorithm [6,7] over general finite fields.

2.1. Adleman-Manders-Miller cube root algorithm

The Tonelli-Shanks [8,9] method for square root computation was extended to the general r-th roots computation by Adleman, Manders and Miller [1]. Let p be a prime such that $p \equiv 1 \pmod 3$. Let c be a cubic residue over $\mathbb{F}_p$.

TABLE 1. Adleman-Manders-Miller's cube root algorithm [1]

Input: A cubic residue c in $\mathbb{F}_p$ Output: A cube root of c
Step 1: Let $p-1 = 3^s t$ with $t = 3l \pm 1$
Step 2: Select a cubic non-residue b in $\mathbb{F}_p$ $a \leftarrow b^t$ $a' \leftarrow a^{3^{s-1}}$

Step 3: (Computation of a cube root of $(c^t)^{-1}$)
$h \leftarrow 1$, $r \leftarrow c^t$
for $i = 1$ to $s-1$
   $d \leftarrow r^{3^{s-i-1}}$
   if $d = 1$, **then** $k \leftarrow 0$
   else if $d = a'$, **then** $k \leftarrow 2$
   else $k \leftarrow 1$
   $h \leftarrow h \cdot a^k$, $r \leftarrow r \cdot (a^3)^k$
   $a \leftarrow a^3$
end for

Step 4: $r \leftarrow c^l \cdot h$ **if** $t = 3l+1$, **then** $r \leftarrow r^{-1}$ Return r

We require one multiplication for each "square" and "multiply" operation over $\mathbb{F}_p$. That is, 1.5 multiplications are required on average over $\mathbb{F}_p$. The Adleman-Manders-Miller algorithm requires approximately $(1.5 \times 3)\log t =$

$4.5 \log t$ (log=$\log_2$) for computing b^t, c^t and c^l ($\log 3 \approx 1$). Furthermore, it requires approximately $5(s-1)+\frac{(s-1)(s-2)}{2} = \frac{s^2+7s-8}{2}$ multiplications for computing the for-loop and a'. Therefore, the average number of $\mathbb{F}_q$ multiplications is $4.5 \log t + \frac{s^2+7s-8}{2}$ over $\mathbb{F}_p$, and the complexity of the Adleman-Manders-Miller algorithm is $O(\log^3 p + s^2 \log^2 p)$ [1, 2].

2.2. Pocklington and Padró-Sáez cube root algorithm

Assume that q is a power of prime p such that $q \equiv 1 \pmod 3$. Let c be a cubic residue over $\mathbb{F}_q$. Heo et al. [4] clarified and generalized the Pocklington and Padró-Sáez algorithm [6, 7]. That is, Heo et al. presented a cube root algorithm over $\mathbb{F}_q$, which is described in Table 2.

Using the ring isomorphism $\mathbb{F}_q[X]/\langle X^3 - c\rangle \cong \mathbb{F}_q \oplus \mathbb{F}_q \oplus \mathbb{F}_q$, $N(z)$ is defined as the product of all the conjugates of $z = \alpha + \beta X + \gamma X^2 \in \mathbb{F}_q[X]/\langle X^3 - c\rangle$. That is, $N(z) = z\bar{z}\bar{\bar{z}} \in \mathbb{F}_q$, where $\bar{z} = \alpha + \beta\epsilon X + \gamma\epsilon^2 X^2$ and a primitive cube root of unity ϵ.

Table 2. Pocklington and Padró-Sáez cube root algorithm [4]

Input : A cubic residue c in $\mathbb{F}_q$ with $q-1 = 3^s t, \gcd(3,t) = 1$ Output : x satisfying $x^3 = c$ in $\mathbb{F}_q$
Step 1: **if** $q \equiv 4 \pmod 9$ **then** $x \leftarrow c^{\frac{2q+1}{9}}$ and **return** x
Step 2: **if** $q \equiv 7 \pmod 9$ **then** $x \leftarrow c^{\frac{q+2}{9}}$ and **return** x
Step 3: Choose random $\alpha, \beta, \gamma \in \mathbb{F}_q$ and let $z := \alpha + \beta X + \gamma X^2 \in \mathbb{F}_q[X]/\langle X^3 - c\rangle$ **if** $N(z) = 0$ **then** go to STEP 3 $z \leftarrow z^t$
Step 4: **if** $\alpha = \beta = 0$ or $\beta = \gamma = 0$ or $\gamma = \alpha = 0$ **then** choose new z again **while** $\alpha\beta \neq 0$ or $\beta\gamma \neq 0$ or $\gamma\alpha \neq 0$ **do** $z_0 := \alpha_0 + \beta_0 X + \gamma_0 X^2 \leftarrow z$ $z \leftarrow z^3$
Step 5: **if** $\beta = \gamma = 0$ **then** $x \leftarrow \frac{\alpha_0}{\beta_0}$ **else if** $\gamma = \alpha = 0$ **then** $x \leftarrow -\frac{9c\alpha_0\beta_0\gamma_0}{\beta}$ **else then** $x \leftarrow -\frac{\gamma}{9\alpha_0\beta_0\gamma_0}$ **return** x

If $q \equiv 1 \pmod 9$, then the algorithm computes z^t by repeated cubing z^3 until z has two coefficients equal to zero. That is, the complexity of the algorithm is $O(\log^3 q)$. When s is large, $s \approx \log q$, the Pocklington and Padró-Sáez algorithm is more efficient than the Adleman-Manders-Miller algorithm, as it is independent of the size of s. Because the while-loop (in Step 4) operates probabilistically until $z^{3^m t}$ has exactly 2 zero coefficients with $0 \leq m \leq s-1$, where m is the number of times that the loop is executed in Step 4. The probability that a chosen invertible $z = a + X \in \mathbb{F}_q[X]/\langle X^3 - c\rangle$ satisfies $z^{3^m} = \alpha + \beta X + \gamma X^2$ with 2 zero coefficients is $\frac{1}{3^{2s-2m-1}}$. (We can verify that the probability is identical for both random z and special z from [4].) Therefore the expected number of iterations of the while-loop is $s - \frac{3}{8}\left(1 - \frac{1}{9^s}\right)$ [4].

We consider the average number of $\mathbb{F}_q$ multiplications. The algorithm selected a random $\alpha, \beta, \gamma \in \mathbb{F}_q$ with $z = \alpha + \beta X + \gamma X^2 \in \mathbb{F}_q[X]/\langle X^3 - c\rangle$. To compute z^t, we consider the classical "double and add" relations. As one can express $z^m = \alpha_m + \beta_m X + \gamma_m X^2$ by the array $[\alpha_m, \beta_m, \gamma_m]$ with respect to the basis $\{1, X, X^2\}$, we obtain the following "double" and "add" operations:

$$\begin{aligned} \alpha_{2n} &= \alpha_n^2 + 2c\beta_n\gamma_n, \\ \beta_{2n} &= c\gamma_n^2 + 2\alpha_n\beta_n, \\ \gamma_{2n} &= \beta_n^2 + 2\alpha_n\gamma_n, \end{aligned} \tag{1}$$

$$\begin{aligned} \alpha_{m+1} &= \alpha_1\alpha_m + c(\beta_1\gamma_m + \gamma_1\beta_m), \\ \beta_{m+1} &= \alpha_1\beta_m + \alpha_m\beta_1 + c\gamma_1\gamma_m, \\ \gamma_{m+1} &= \alpha_1\gamma_m + \beta_m\beta_1 + \gamma_1\alpha_m, \end{aligned} \tag{2}$$

where the initial values are $\alpha_1 = \alpha, \beta_1 = \beta, \gamma_1 = \gamma$ ($z^1 = \alpha + \beta X + \gamma X^2$). The Pocklington and Padró-Sáez algorithm uses 7 multiplications for each "square" and 11 multiplications for each "add". Therefore the algorithm can be accomplished by $12.5 \log t (= 7 + \frac{11}{2})$ multiplications on average for computing z^t. To extract a cube root of c, the algorithm requires $12.5 \log t + 18s$ multiplications (original PPS).

The number of "square" multiplications depends on the coefficients of $f(x) = x^3 - c$. Furthermore, the number of "add" multiplications depends on the initial values. For a given $f(x) = x^3 - c$, there are a fixed number of "square" multiplications. Thus, our aim is to reduce the number of "add" multiplications. We select a special $z = a + X \in \mathbb{F}_q[X]/\langle X^3 - c\rangle$ with random a (numerically small a). From the choice of the (special) initial values, we obtain

$$\begin{aligned} \alpha_{m+1} &= a\alpha_m + c\gamma_m, \\ \beta_{m+1} &= \alpha_m + a\beta_m, \\ \gamma_{m+1} &= \beta_m + a\gamma_m. \end{aligned} \tag{3}$$

As a is (numerically) small, we calculate z^t in $7 \log t$ multiplications for "square", and $\frac{1}{2} \log t$ multiplications for "add" on average. Subsequently, to compute a cube root of c, the algorithm requires $7.5 \log t + 18s (= 7 + 11)$ multiplications (Refined PPS when a is small). Unfortunately, if we choose an unconstrained $a \in \mathbb{F}_q$, then the algorithm requires $9 \log t (= 7 + \frac{4}{2}) + 18s$ multiplications (Refined PPS when a is random).

TABLE 3. Theoretical estimation (average number of $\mathbb{F}_q$ multiplications)

AMM	$4.5\log t + \frac{s^2+7s-8}{2}$
PPS	$12.5\log t + 18s$
Refined PPS(random a)	$9\log t + 18s$
Refined PPS(small a)	$7.5\log t + 18s$

TABLE 4. Running time (in seconds) for cube root computation with $p \approx 2^{2000}$

s	10	30	50	80	100	150	200	300	400	500	600
AMM	0.013	0.016	0.021	0.032	0.041	0.074	0.126	0.274	0.475	0.792	1.077
PPS	0.055	0.056	0.056	0.057	0.055	0.056	0.057	0.056	0.056	0.055	0.057
Refined PPS(random a)	0.035	0.035	0.040	0.041	0.039	0.040	0.041	0.040	0.039	0.040	0.041
Refined PPS(small a)	0.028	0.030	0.031	0.032	0.032	0.031	0.030	0.031	0.031	0.030	0.032

3. Experimental results

In this section, we implement our proposed algorithm with the AMM (Adleman-Manders-Miller) algorithm [1, 2, 8, 9] in a finite field $\mathbb{F}_q$. The complexity of the AMM cube root algorithm is $O(\log^4 q)$, where $q - 1 = 3^s t$ with $\gcd(3, t) = 1$ and $s \approx \log q$, and the complexity of the PPS (Pocklington and Padró-Sáez) cube root algorithm is $O(\log^3 q)$ [4, 6, 7].

We compared the average number of $\mathbb{F}_q$ multiplications between the cube root algorithms. The AMM algorithm always required $4.5\log t + \frac{s^2+7s-8}{2}$ multiplications. However, the PPS algorithm has a flexible number of multiplications, as shown in Table 3.

Tables 4 and 5 show the comparison of the implementation results, of the proposed methods for various cases, obtained using the software, SAGE. The implementation was performed using Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz with 8GB memory.

For convenience, we used prime fields $\mathbb{F}_p$ with two different sizes of primes p : 2000 and 3000 bits. The average timings of the cube root computations for 5 different inputs of cubic residue are computed for the cases of $s = 10, 30, 50, \ldots$, etc.

If s is relatively smaller, the AMM algorithm is more efficient than the PPS type algorithms. However, if s is large, the PPS type algorithms are more efficient. The tables show that our proposed methods are faster than the original PPS and AMM algorithms for large value of s.

4. Conclusion and future works

We proposed a refined the Pocklington and Padró-Sáez cube root algorithm over $\mathbb{F}_q$, and successfully reduced the average number of $\mathbb{F}_q$ multiplications

Table 5. Running time (in seconds) for cube root computation with $p \approx 2^{3000}$

s	10	30	50	80	100	150	200	300	400	500	600
AMM	0.035	0.039	0.047	0.068	0.083	0.143	0.217	0.474	0.861	1.209	1.868
PPS	0.131	0.129	0.130	0.132	0.131	0.129	0.131	0.130	0.132	0.131	0.131
Refined PPS(random a)	0.091	0.087	0.089	0.090	0.091	0.090	0.089	0.089	0.090	0.091	0.091
Refined PPS(small a)	0.075	0.072	0.075	0.073	0.075	0.074	0.072	0.074	0.075	0.072	0.074

from the original Pocklington and Padró-Sáez algorithm [4, 6, 7]. Furthermore, software implementations via SAGE also indicate that the proposed methods are faster than the original Pocklington and Padró-Sáez algorithm when s is large [4, 6, 7]. However, the Adleman-Manders-Miller algorithm [1, 2, 8, 9] is more efficient than the Pocklington and Padró-Sáez algorithm for small value of s. Thus, we will examine the development of an efficient algorithm for small value of s in a future work.

Acknowledgements. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2009-0093827). The work of G. H. Cho was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2018R1D1A1B07041716).

References

[1] L. Adleman, K. Manders, and G. Miller, *On taking roots in finite fields*, in 18th Annual Symposium on Foundations of Computer Science (Providence, R.I., 1977), 175–178, IEEE Comput. Sci., Long Beach, CA, 1977.

[2] Z. Cao, Q. Sha, and X. Fan, *Adleman-Manders-Miller root extraction method revisited*, in Information security and cryptology, 77–85, Lecture Notes in Comput. Sci., 7537, Springer, Heidelberg, 2012.

[3] M. Cipolla, *Un metodo per la risolutione della congruenza di secondo grado*, Rendiconto dell'Accademia Scienze Fisiche e Matematiche, Napoli, Ser. 3, **9** (1903), 154–163.

[4] G. Heo, S. Choi, K. H. Lee, N. Koo, and S. Kwon, *Remarks on the Pocklington and Padró-Sáez cube root algorithm in F_q*, Electronics Letters **50** (2014), no. 14, 1002–1003.

[5] D. H. Lehmer, *Computer technology applied to the theory of numbers*, in Studies in Number Theory, 117–151, Math. Assoc. Amer. (distributed by Prentice-Hall, Englewood Cliffs, N.J.), 1969.

[6] C. Padró and G. Sáez, *Taking cube roots in $\mathbb{Z}_m$*, Appl. Math. Lett. **15** (2002), no. 6, 703–708.

[7] H. C. Pocklington, *The direct solution of the quadratic and cubic binomial congruences with prime moduli*, Proceedings of the Cambridge Philosophical Society **19** (1917), 57–59.

[8] D. Shanks, *Five number-theoretic algorithms*, in Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972), 51–70. Congressus Numerantium, VII, Utilitas Math., Winnipeg, MB, 1973.

[9] A. Tonelli, *Bemerkung über die Auflösung quadratischer Congruenzen*, Nachrichten von der Königl. Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen, pp. 344–346, 1891.

Gook Hwa Cho
Institute of Mathematical Sciences
Ewha Womans University
Seoul 03760, Korea
Email address: `ghcho@ewha.ac.kr`

Hyang-Sook Lee
Department of Mathematics
Ewha Womans University
Seoul 03760, Korea
Email address: `hsl@ewha.ac.kr`