Square Root Algorithm in $\mathbb{F}_q$ for $q \equiv 2^s + 1 \pmod{2^{s+1}}$

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 a square root algorithm in $\mathbb{F}_q$ which generalizes Atkins's square root algorithm [6] for $q \equiv 5 \pmod 8$ and Kong et al.'s algorithm [8] for $q \equiv 9 \pmod{16}$. Our algorithm precomputes a primitive 2^s-th root of unity ξ where s is the largest positive integer satisfying $2^s | q-1$, and is applicable for the cases when s is small. The proposed algorithm requires one exponentiation for square root computation and is favorably compared with the algorithms of Atkin, Müller and Kong et al.

Keywords : square root algorithm, finite field, Tonelli-Shanks algorithm, Cipolla-Lehmer algorithm

1 Introduction

There are two well-known algorithms for square root computation in finite field $\mathbb{F}_q$; the Tonelli-Shanks algorithm [1, 2] and the Cipolla-Lehmer [3, 4] algorithms. The Tonelli-Shanks algorithm depends on the exponent s of 2 satisfying $2^s | q-1$ and $2^{s+1} \not| q-1$, which makes the worst case complexity of the Tonelli-Shanks $O(\log^4 q)$ [5] while the Cipolla-Lehmer can be executed in $O(\log^3 q)$ [3, 4]. However, due to the cumbersome extension field arithmetic needed for the Cipolla-Lehmer algorithm, the Tonelli-Shanks algorithm performs better than the Cipolla-Lehmer for small exponents s, and related research can be found in [5, 9, 10, 11].

On the other hand, when the exponent s is small, there are some 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 [6], Müller [7] and Kong et al. [8], 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 exploited for square root extraction problem in $\mathbb{F}_q$. Our aim is to generalize this 'exponentiation only' approach for square root extraction and give concrete examples. We present a new square root algorithm using a precomputed primitive element and show that our algorithm requires only one exponentiation in $\mathbb{F}_q$.

2 Existing square root algorithms for small exponent

2.1 Atkin's algorithm

There is no known square root algorithm using one exponentiation for the case $q \equiv 1 \pmod 4$. However, if $q \equiv 5 \pmod 8$, a special case of $q \equiv 1 \pmod 4$, there is an efficient square root algorithm [6] 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.

The Algorithm 1 uses the fact that 2 is a quadratic nonresidue in $\mathbb{F}_q$ when $q \equiv 5 \pmod 8$ and hence $2a$ is also a quadratic nonresidue. So one can see that $(2a)^{\frac{q-1}{2}} = -1$ and $(2a)^{\frac{q-1}{4}} = \sqrt{-1}$. This algorithm also uses the fact that $(\sqrt{-1}-1)^2 = -2\sqrt{-1}$. Algorithm 1 needs 1 exponentiation and 4 multiplications, and is more efficient than the Tonelli-Shank's or the Cipolla-Lehmer.

2.2 Müller's generalization

Müller [7] extended Atkin's idea to the case $q \equiv 9 \pmod{16}$. In this case, 2 is no longer a quadratic nonresidue in $\mathbb{F}_q$. However, since $(2a)^{\frac{q-1}{4}}$ is 1 or -1, Atkin's idea can be extended if we have another parameter; a quadratic nonresidue when $(2a)^{\frac{q-1}{4}} = 1$ and a quadratic residue when $(2a)^{\frac{q-1}{4}} = -1$. 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.

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.

2.3 Kong et al.'s algorithm

Müller's algorithm requires 2 exponentiations in $\mathbb{F}_q$. If $(2a)^{\frac{q-1}{4}} = -1$, then Müller's algorithm can be improved further and it is realized in Kong's algorithm shown in Algorithm 3. Since the case $(2a)^{\frac{q-1}{4}} = -1$ is similar to the Atkin's algorithm, one needs only one exponentiation in step 1-4 of Kong's algorithm. If $(2a)^{\frac{q-1}{4}} = 1$, then the step 6-9 of Kong's algorithm finds a square root in a similar manner with Müller's algorithm. Hence Kong et al.'s algorithm is about 2 times faster than Müller's algorithm with probability $\frac{1}{2}$. That is, Kong's algorithm needs 1.5 exponentiations on average and is about 25% efficient than Müller's algorithm. Note that Kong et al.'s algorithm is also probabilistic since one needs a quadratic nonresidue in step 6.

Algorithm 3 Kong el al.'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: $b \leftarrow (2a)^{\frac{q-9}{16}}$
2: $i \leftarrow 2ab^2,\ r \leftarrow i^2$
3: **if** $r = -1$ **then**
4: $\quad x \leftarrow ab(i-1)$
5: **else**
6: $\quad$ Find a quadratic nonresidue $d \in \mathbb{F}_q$
7: $\quad u \leftarrow bd^{\frac{q-9}{8}}$
8: $\quad i \leftarrow 2u^2d^2a$
9: $\quad x \leftarrow uda(i-1)$
10: **end if**
11: **return** x.

3 New square root formula for $\mathbb{F}_q$ with $q \equiv 2^s + 1 \pmod{2^{s+1}}$

Let q be a power of an odd prime and let s be the largest positive integer satisfying $2^s | q-1$. Since $\frac{q-1}{2^s} \equiv 1 \pmod 2$, one has $q \equiv 2^s + 1 \pmod{2^{s+1}}$. That is, for any odd prime power q, there exists unique positive integer s satisfying $q \equiv 2^s + 1 \pmod{2^{s+1}}$, where s is the largest positive integer satisfying $2^s | q-1$.

For given square c in $\mathbb{F}_q$, define

$$b = c^{\frac{q-(2^s+1)}{2^{s+1}}} \tag{1}$$

and

$$\zeta = c^{\frac{q-1}{2^s}} = c \cdot c^{\frac{q-(2^s+1)}{2^s}} = c\{c^{\frac{q-(2^s+1)}{2^{s+1}}}\}^2 = cb^2 \tag{2}$$

Since c is a square in $\mathbb{F}_q$, ζ is a primitive 2^t-th root of unity in $\mathbb{F}_q$ for some uniquely determined $t \leq s-1$, that is, there is $t < s$ satisfying

$$\zeta^{2^t} = 1, \quad \zeta^{2^{t-1}} = -1 \tag{3}$$

Let ξ be a primitive 2^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 $\xi = d^{\frac{q-1}{2^s}}$ where d is a quadratic nonresidue in $\mathbb{F}_q$. Therefore our method is also probabilistic (i.e., randomized).

Since $\xi^{2^{s-t}}$ is a primitive 2^t-th root of unity, there exist unique i and j determined $\pmod{2^t}$ such that

$$\xi^{2^{s-t}} = \zeta^i, \quad (\xi^{2^{s-t}})^j = \zeta \tag{4}$$

From $\xi^{2^{s-t}} = \zeta^i = (\xi^{2^{s-t}})^{ij}$, we have

$$ij \equiv 1 \pmod{2^t} \tag{5}$$

Now we present our new theorem which states that a square root can be found using one exponentiation under suitable conditions.

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

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

$$x^2 = c \cdot cb^2 \cdot \xi^{2u} = c \cdot \zeta \cdot \xi^{2u}$$

Since $u = j(2^t-1)2^{s-t-1} + 2^{s-1}k$ for some integer k and using $\xi^{2^s} = 1$,

$$\zeta\xi^{2u} = (\xi^{2^{s-t}})^j \cdot \xi^{2u} = \xi^{j\cdot 2^{s-t}+2u} \tag{6}$$

$$= \xi^{j\cdot 2^{s-t}+j(2^t-1)2^{s-t}+2^s k} \tag{7}$$

$$= \xi^{j\cdot 2^{s-t}+j(2^t-1)2^{s-t}} = \xi^{j\cdot 2^{s-t}(1+2^t-1)} \tag{8}$$

$$= \xi^{j\cdot 2^s} = 1 \tag{9}$$

Therefore one has $x^2 = c \cdot \zeta\xi^{2u} = c$. □

Finding i or j in the equation (5) is difficult if 2^t is large. So our method is useful only when t is relatively small, and can be viewed as a parallelized version of the Tonelli-Shanks algorithm.

Example 1. Square root for $q \equiv 3 \pmod 4$: This is the case $s = 1$. Therefore $\zeta = c^{\frac{q-1}{2}} = 1$ and thus $t = 0$. Also one has $\xi = -1$ and $u = 0$. Thus a square root of c is given as $cb = c \cdot c^{\frac{q-3}{4}} = c^{\frac{q+1}{4}}$, which is well-known.

Example 2. Square root for $q \equiv 5 \pmod 8$: This is the case $s = 2$. Therefore $\zeta = c^{\frac{q-1}{4}} = \pm 1$ and thus $t = 0$ or 1. Also ξ is a primitive 2^2-th root of unity satisfying $\xi^{2^{2-t}} = \zeta^i$ for some $i \pmod{2^t}$. Thus a square root of c is given as $cb\xi^u$ where $u \equiv j(2^t-1)2^{1-t} \pmod 2$. When $t = 0$, one has $u = 0$ and $x = cb$ is a square root, and when $t = 1$, one has $i = j = 1$ and also $u = 1$ and a square root is given $x = cb\xi$.

Example 3. Square root for $q \equiv 9 \pmod{16}$: This is the case $s = 3$. Therefore $\zeta = c^{\frac{q-1}{2^3}}$ has order 2^t with $t = 0, 1, 2$. Also ξ is a primitive 2^3-th root of unity satisfying $\xi^{2^{3-t}} = \zeta^i$ for some $i \pmod{2^t}$ with $ij \equiv 1 \pmod{2^t}$. Thus a square root of c is given as $cb\xi^u$ where $u \equiv j(2^t-1)2^{2-t} \pmod{2^2}$. To summarize,

- When $t = 0$, one has $u = 0$ and $x = cb$ is a square root.
- When $t = 1$, then one has $i = j = 1$ and also $u \equiv 2j \equiv 2 \pmod{2^2}$ and a square root is given as $x = cb\xi^2$.
- When $t = 2$, then, from $\xi^2 = \zeta^i$, one has $(i, j) = (1, 1), (3, 3)$ and also $u \equiv 3j \pmod{2^2}$. Therefore a square root is given as $x = cb\xi^{3j}$ where $\xi^2 = \zeta^j$. That is,
  - When $\xi^2 = \zeta$, then a square root is given as $x = cb\xi^3$.
  - When $\xi^2 = \zeta^3$ (i.e., $\zeta = -\xi^2$), then a square root is given as $x = cb\xi$.

Example 4. Square root for $q \equiv 17 \pmod{32}$: This is the case $s = 4$. Therefore $\zeta = c^{\frac{q-1}{2^4}}$ has order 2^t with $t = 0, 1, 2, 3$. Also ξ is a primitive 2^4-th root of unity satisfying $\xi^{2^{4-t}} = \zeta^i$ for some $i \pmod{2^t}$ with $ij \equiv 1 \pmod{2^t}$. Thus a square root of c is given as $cb\xi^u$ where $u \equiv j(2^t - 1)2^{3-t} \pmod{2^3}$. To summarize,

- When $t = 0$, one has $u = 0$ and $x = cb$ is a square root.
- When $t = 1$, then from $\xi^8 = \zeta$ with $\zeta^2 = 1$, one has $i = j = 1$ and also $u \equiv 4j \equiv 4 \pmod{2^3}$ and a square root is given $x = cb\xi^4$.
- When $t = 2$, then, from $\xi^4 = \zeta^i$, one has $(i, j) = (1, 1), (3, 3) \pmod{2^2}$ and also $u \equiv 6j \pmod{2^3}$ and a square root is given $x = cb\xi^{6j}$ where $\xi^4 = \zeta^j$. That is,
  - When $\xi^4 = \zeta$, then a square root is given as $x = cb\xi^6$.
  - When $\xi^4 = \zeta^3$ (i.e., $\zeta = -\xi^4$), then a square root is given as $x = cb\xi^2$.
- When $t = 3$, then, from $\xi^2 = \zeta^i$, one has $(i, j) = (1, 1), (3, 3), (5, 5), (7, 7) \pmod{2^3}$ and also $u \equiv 7j \pmod{2^3}$ and a square root is given $x = cb\xi^{7j}$ where $\xi^2 = \zeta^j$. That is,
  - When $\xi^2 = \zeta$, then a square root is given as $x = cb\xi^7$.
  - When $\xi^2 = \zeta^3$ (i.e., $\zeta = \xi^6$), then a square root is given as $x = cb\xi^5$.
  - When $\xi^2 = \zeta^5$ (i.e., $\zeta = -\xi^2$), then a square root is given as $x = cb\xi^3$.
  - When $\xi^2 = \zeta^7$ (i.e., $\zeta = -\xi^6$), then a square root is given as $x = cb\xi$.

Our examples are realized in the following square root algorithms 4,5,6. All the algorithms need a primitive 2^s-th root of unity ξ which can be precomputed and fixed once q is given. Our proposed algorithms need only one exponentiation (that is, for computing b in step 1). On the other hand, Müller's algorithm needs 2 exponentiations (in step 1 and 3) and Kong's algorithm needs 1.5 exponentiations on average.

Note that no precomputation (to further improve the complexity) is possible in both Müller's and Kong's algorithms. It should also be mentioned that the cost of multiplications in each algorithm is negligible compared to the cost of exponentiations. For example, in Algorithm 6, one needs at most 7 multiplications (5 multiplications in step 2 and at most 2 multiplications in the searching step 3-10), and this cost is negligible compared to the cost of one exponentiation.

Algorithm 4 Our square root algorithm when $q \equiv 5 \pmod 8$

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

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

$b \leftarrow c^{\frac{q-5}{8}}$.
$\zeta \leftarrow cb^2$
if $\zeta = 1$, **then** $x \leftarrow cb$
else **then** $x \leftarrow cb\xi$
return x.

Algorithm 5 Our square root algorithm when $q \equiv 9 \pmod{16}$

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

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

$b \leftarrow c^{\frac{q-9}{16}}$
$\zeta \leftarrow cb^2$
if $\zeta = 1$ **then** $x \leftarrow cb$
else if $\zeta = -1$ **then** $x \leftarrow cb\xi^2$
else if $\zeta = \xi^2$ **then** $x \leftarrow cb\xi^3$
else **then** $x \leftarrow cb\xi$
return x.

Algorithm 6 Our square root algorithm when $q \equiv 17 \pmod{32}$

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

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

$b \leftarrow c^{\frac{q-17}{32}}$
$X \leftarrow cb, \zeta \leftarrow Xb, A \leftarrow \xi, B \leftarrow A^2, C \leftarrow B^2, D \leftarrow BC$
if $\zeta = 1$ **then** $x \leftarrow X$
if $\zeta = -1$ **then** $x \leftarrow XC$
if $\zeta = B$ **then** $x \leftarrow XAD$
if $\zeta = -B$ **then** $x \leftarrow XAB$
if $\zeta = C$ **then** $x \leftarrow XD$
if $\zeta = -C$ **then** $x \leftarrow XB$
if $\zeta = D$ **then** $x \leftarrow XAC$
if $\zeta = -D$ **then** $x \leftarrow XA$
return x.

4 Conclusion

We proposed a new square root algorithm in $\mathbb{F}_q$ for $q \equiv 2^s + 1 \pmod{2^{s+1}}$ which can be successfully implemented for small s. We presented our algorithm for the cases $s = 2, 3, 4$ and all the given algorithms need only one exponentiation, while the Algorithm 2 in [7] needs 2 exponentiations and the Algorithm 3 in [8] needs 1.5 exponentiations. The technique of precomputation is not possible in [7, 8] while our algorithm supports it. Also, the case $s = 4$ (i.e., Algorithm 6 with $q \equiv 17 \pmod{32}$) has not been previously considered in [6, 7, 8]. Our algorithm is useful for relatively small values of s since the number of cases we need to consider is 2^{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] 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.

[4] 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.

[5] S. Lindhurst, "An Analysis for Computing Square Roots in Finite Fields", *CRM Proceeding and Lecture Notes*, vol. 19, pp. 231-242, 1999.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] G. Adj, and F. Rodríguez-Henríquez, "Square root computation over even extension fields", preprint, available at http://eprint.iacr.org/2012/685.pdf.