arXiv:1111.4877v1 [cs.SC] 21 Nov 2011

Adleman-Manders-Miller Root Extraction Method Revisited

Zhengjun Cao*, Qian Sha, Xiao Fan
Department of Mathematics, Shanghai University, Shanghai, China.
* caozhj@shu.edu.cn

Abstract

In 1977, Adleman, Manders and Miller had briefly described how to extend their square root extraction method to the general rth root extraction over finite fields, but not shown enough details. Actually, there is a dramatic difference between the square root extraction and the general rth root extraction because one has to solve discrete logarithms for rth root extraction. In this paper, we clarify their method and analyze its complexity. Our heuristic presentation is helpful to grasp the method entirely and deeply.

Keywords. square root extraction, rth root extraction

1 Introduction

Root extraction is a classical problem in computers algebra. It is essential to cryptosystems based on elliptic curves [2]. There are several efficient probabilistic algorithms for square root extraction in finite fields, such as Cipolla-Lehmer [6, 7], Tonelli-Shanks [10, 12] and Adleman-Manders-Miller [1]. All of them require a quadratic nonresidue as an additional input. In 2004, Müller investigated this topic in Ref.[8]. In 2011, Sze [11] presented a novel idea to compute square roots over finite fields, without being given any quadratic nonresidue, and without assuming any unproven hypothesis.

Adleman-Manders-Miller square root extraction method can be extended to solve the general rth root extraction problem. In recent, Nishihara et al. [9] have specified the Adleman-Manders-Miller method for cube root extraction. Barreto and Voloch [2] proposed an efficient algorithm to compute rth roots in F_{p^m} for certain choices of m and p. Besides, it requires that $r \,||\, p-1$ and $(m, r) = 1$, where the notation $a^b||c$ means that a^b is the highest power of a dividing c.

The basic idea of Adleman-Manders-Miller square root extraction in F_p can be described as follows. Write $p-1$ in the form $2^t \cdot s$, where s is odd. Given a quadratic residue δ and a quadratic nonresidue ρ, we have

$$(\delta^s)^{2^{t-1}} \equiv 1 \pmod{p}, \quad (\rho^s)^{2^{t-1}} \equiv -1 \pmod{p}$$

If $t \geq 2$, then $(\delta^s)^{2^{t-2}} \pmod p \in \{1, -1\}$. Take $k_1 = 0$ or 1 such that

$$(\delta^s)^{2^{t-2}} (\rho^s)^{2^{t-1} \cdot k_1} \equiv 1 \pmod p$$

Since $(\delta^s)^{2^{t-3}} (\rho^s)^{2^{t-2} \cdot k_1} \pmod p \in \{1, -1\}$, take $k_2 = 0$ or 1 such that

$$(\delta^s)^{2^{t-3}} (\rho^s)^{2^{t-2} \cdot k_1} (\rho^s)^{2^{t-1} \cdot k_2} \equiv 1 \pmod p$$

Likewise, we can obtain $k_3, \cdots, k_{t-1} \in \{0, 1\}$ such that

$$(\delta^s) (\rho^s)^{2 \cdot k_1 + 2^2 \cdot k_2 + \cdots + 2^{t-1} \cdot k_{t-1}} \equiv 1 \pmod p$$

Thus, we have

$$\left(\delta^{\frac{s+1}{2}}\right)^2 \left((\rho^s)^{k_1 + 2 \cdot k_2 + \cdots + 2^{t-2} \cdot k_{t-1}}\right)^2 \equiv \delta \pmod p$$

It should be stressed, however, that there is a dramatic difference between the square root extraction and the general rth root extraction. Write $p - 1$ in the form $r^t \cdot s$, where $(r, s) = 1$. Given a rth residue δ and a rth nonresidue ρ, we have

$$(\delta^s)^{r^{t-1}} \equiv 1 \pmod p, \quad (\rho^s)^{r^{t-1}} \not\equiv 1 \pmod p$$

Since $(\delta^s)^{r^{t-2}} \pmod p$ is a root of the equation $X^r \equiv 1 \pmod p$ and the equation has r different roots (these roots can be represented by $(\rho^s)^{k_i \cdot r^{t-1}}$, $k_i \in \{0, 1, \cdots, r-1\}$), it becomes difficult to find k_1 such that

$$(\delta^s)^{r^{t-2}} (\rho^s)^{r^{t-1} \cdot k_1} \equiv 1 \pmod p$$

In 1977, Adleman, Manders and Miller [1] had presented a brief description on how to extend their square root extraction method to the general rth root extraction over finite fields, but not shown enough details. By the way, it is the only known method for the general rth root extraction over finite fields. In this paper, we clarify their method and analyze its complexity.

2 Preliminary

Let $Z_n = \{0, 1, \cdots, n-1\}$ be the set of all numbers smaller than n, $Z_n^* = \{x \,|\, 1 \leq x \leq n \text{ and } \gcd(x, n) = 1\}$ be the set of numbers in Z_n that are coprime to n. The following definitions and results can be found in Ref.[4].

Definition 1. *A residue $a \in Z_n^*$ is said to be a quadratic residue if there exists some $x \in Z_n^*$ such that $x^2 \equiv a \pmod n$. If a is not a quadratic residue, then it is referred to as a quadratic non-residue.*

Theorem 2. (Euler's Criterion) *For prime p, an element $a \in Z_p^*$ is a quadratic residue if and only if $a^{\frac{p-1}{2}} \equiv 1 \pmod p$.*

Definition 3. (Legendre Symbol) *For any prime p and $a \in Z_p^*$, we define the Legendre symbol*

$$\left[\frac{a}{p}\right] = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue (mod } p) \\ -1 & \text{if } a \text{ is a quadratic non-residue (mod } p) \end{cases}$$

For an integer a, we define $\log(a)$ to be the number of bits in the binary representation of $|a|$; more precisely,

$$\log(a) = \begin{cases} \lfloor \log_2 |a| \rfloor + 1 & \text{if } a \neq 0 \\ 1 & \text{if } a = 0 \end{cases}$$

Given $a \in Z_n$ and a non-negative integer e, the repeated-squaring algorithm computes $a^e \pmod n$ using just $\mathcal{O}(\log(e))$ multiplications in Z_n, thus taking time $\mathcal{O}(\log(e)\log^2 n)$. Therefore, we have the following result:

Proposition 4. *For an odd prime p, we can test whether an integer a is a quadratic residue modulo p by either performing the exponentiation $a^{\frac{(p-1)}{2}} \pmod p$ or by computing the Legendre symbol $\left[\frac{a}{p}\right]$. Assume that $0 < a < p$. Using a standard repeated squaring algorithm, the former method takes time $\mathcal{O}(\log^3 p)$, while using Euclidean-like algorithm, the latter method takes time $\mathcal{O}(\log^2 p)$.*

Proof. See [5].

Let R be a ring. Let us define the length of a polynomial $f(X) \in R[X]$, denoted by $\log(f)$, to be the length of its coefficient vector; more precisely, we define

$$\log(f) = \begin{cases} \deg(f) + 1 & \text{if } f \neq 0 \\ 1 & \text{if } f = 0 \end{cases}$$

Analogous to algorithms for modular integer arithmetic, we can also do arithmetic in the residue class ring $R[X]/(f)$, where $f \in R[X]$ is a polynomial of $\deg(f) > 0$ whose leading coefficient $\mathrm{lc}(f)$ is a unit.

Proposition 5. *Let $R[X]/(f)$ be a residue class ring, where $f \in R[X]$ is a polynomial of $\deg(f) > 0$ whose leading coefficient $\mathrm{lc}(f)$ is a unit. Given $g \in R[X]/(f)$ and a non-negative exponent e, using repeated-squaring algorithm we can compute g^e taking $\mathcal{O}(\log(e)\deg(f)^2)$ operations in R.*

Proof. See [3].

Notice that using a standard representation for F_p, each operation in F_p takes time $\mathcal{O}(log^2 p)$.

3 Adleman-Manders-Miller square root extraction method

The Adleman-Manders-Miller square root extraction method requires a quadratic non-residue as an additional input. We classify the method into two kinds because there is a gap between

the base field F_p and the extension F_{p^m} to test whether an element is a quadratic non-residue.

3.1 Adleman-Manders-Miller square root extraction method in F_p

Consider the problem to find a solution to the congruence $X^2 \equiv \delta \pmod{p}$ over finite field F_p, where p is an odd prime.

Adleman, Manders and Miller [1] proposed an algorithm to solve the problem. Their square root extraction method is based on the following facts. Write $p-1$ in the form $2^t \cdot s$, where s is odd. Given a quadratic residue δ and a quadratic nonresidue ρ, we have

$$(\delta^s)^{2^{t-1}} \equiv 1 \pmod{p}, \quad (\rho^s)^{2^{t-1}} \equiv -1 \pmod{p}$$

If $t = 1$, then $\delta^s \equiv 1 \pmod{p}$. Hence, we have $\left(\delta^{\frac{s+1}{2}}\right)^2 \equiv \delta \pmod{p}$. It means that $\delta^{\frac{s+1}{2}}$ is a square root of δ. In this case, it only takes time $\mathcal{O}(\log(s)\log^2 p)$.

If $t \geq 2$, then $(\delta^s)^{2^{t-2}} \pmod{p} \in \{1, -1\}$. Take $k_1 = 0$ or 1 such that

$$(\delta^s)^{2^{t-2}} (\rho^s)^{2^{t-1} \cdot k_1} \equiv 1 \pmod{p}$$

Take $k_2 = 0$ or 1 such that

$$(\delta^s)^{2^{t-3}} (\rho^s)^{2^{t-2} \cdot k_1} (\rho^s)^{2^{t-1} \cdot k_2} \equiv 1 \pmod{p}$$

Likewise, we obtain $k_3, \cdots, k_{t-1} \in \{0, 1\}$ such that

$$(\delta^s) (\rho^s)^{2 \cdot k_1 + 2^2 \cdot k_2 + \cdots + 2^{t-1} \cdot k_{t-1}} \equiv 1 \pmod{p}$$

Finally, we have

$$\left(\delta^{\frac{s+1}{2}}\right)^2 \left((\rho^s)^{k_1 + 2 \cdot k_2 + \cdots + 2^{t-2} \cdot k_{t-1}}\right)^2 \equiv \delta \pmod{p}$$

To find a quadratic non-residue ρ, it requires to check that $[\frac{\rho}{p}] \neq 1$. The computation takes time $\mathcal{O}(\log^2 p)$. If we do this for more than $\mathcal{O}(1)\log p$ different randomly chosen ρ, then with probability $> 1 - (\frac{1}{p})^{\mathcal{O}(1)}$ at least one of them will give a quadratic non-residue. Thus, to find a quadratic nonresidue ρ, it takes expected time $\mathcal{O}(\log^3 p)$. To compute $b^{2^{t-i-1}} \pmod{p}$, it takes time $\mathcal{O}((t-i-1)\log^2 p)$. Since there are $1 + 2 + \cdots + (t-1) = \frac{t(t-1)}{2}$ steps, the loop takes time $\mathcal{O}(t^2 \log^2 p)$. Thus, the total estimate is $\mathcal{O}(\log^3 p + t^2 \log^2 p)$. At worst (if almost all of $p-1$ is a power of 2), this is $\mathcal{O}(\log^4 p)$.

3.2 Adleman-Manders-Miller square root extraction method in F_{p^m}

As we mentioned before, the Adleman-Manders-Miller method in the extension field F_{p^m} differs from the method in the base field F_p because one can not determine a quadratic non-residue by computing the Legendre Symbol.

Table 1: Adleman-Manders-Miller square root extraction algorithm in F_p

Input: Odd prime p and a quadratic residue δ.
Output: A square root of δ.
Step 1: Choose ρ uniformly at random from F_p^*.
Compute $[\frac{\rho}{p}]$ using Euclidean-like algorithm.
Step 2: **if** $[\frac{\rho}{p}] = 1$, **go to** Step 1.
Step 3: Compute t, s such that $p-1 = 2^t s$, where s is odd.
Compute $a \leftarrow \rho^s, b \leftarrow \delta^s$, $h \leftarrow 1$.
Step 4: **for** $i = 1$ **to** $t-1$
compute $d = b^{2^{t-1-i}}$
if $d = 1$, $k \leftarrow 0$
else $k \leftarrow 1$
$b \leftarrow b \cdot (a^2)^k$, $h \leftarrow h \cdot a^k$
$a \leftarrow a^2$
end for
Step 5: **return** $\delta^{\frac{s+1}{2}} \cdot h$

Set $q = p^m$. To find a quadratic non-residue ρ, it requires to check that $\rho^{\frac{q-1}{2}} \neq 1$. The computation takes time $\mathcal{O}(\log^3 q)$. If we do this for more than $\mathcal{O}(1)\log q$ different randomly chosen ρ, then with probability $> 1 - (\frac{1}{q})^{\mathcal{O}(1)}$ at least one of them will give a quadratic non-residue. Thus, to find a quadratic nonresidue ρ, it takes expected time $\mathcal{O}(\log^4 q)$.

To compute $b^{2^{t-i-1}}$, it takes time $\mathcal{O}((t-i-1)\log^2 q)$. Since there are $1 + 2 + \cdots + (t-1)$ steps, the loop takes time $\mathcal{O}(t^2 \log^2 q)$. Thus, the final estimate is $\mathcal{O}(\log^4 q + t^2 \log^2 q)$.

4 Adleman-Manders-Miller cubic root extraction method

In 2009, Nishihara et al. [9] specified the Adleman-Manders-Miller method for cube root extraction. See the following description.

Set $q = p^m$. The cubic root extraction algorithm takes time $\mathcal{O}(\log^4 q + t^2 \log^2 q)$. As for this claim, we refer to the complexity analysis of Adleman-Manders-Miller square root extraction algorithm in Section 3.2.

Table 2: Adleman-Manders-Miller square root extraction algorithm in F_{p^m}

Input: Odd prime p, a positive integer m and a quadratic residue δ.

Output: A square root of δ.

Step 1: Choose ρ uniformly at random from $F_{p^m}^*$.

Step 2: **if** $\rho^{\frac{p^m-1}{2}} = 1$, **go to** Step 1.

Step 3: Compute t, s such that $p^m - 1 = 2^t s$, where s is odd.

Compute $a \leftarrow \rho^s, b \leftarrow \delta^s$, $h \leftarrow 1$.

Step 4: **for** $i = 1$ **to** $t - 1$

compute $d = b^{2^{t-1-i}}$

if $d = 1$, $k \leftarrow 0$

else $k \leftarrow 1$

$b \leftarrow b \cdot (a^2)^k$, $h \leftarrow h \cdot a^k$

$a \leftarrow a^2$

end for

Step 5: **return** $\delta^{\frac{s+1}{2}} \cdot h$

5 Specification of Adleman-Manders-Miller rth Root Extraction Method

Consider the general problem to find a solution to $X^r = \delta$ in F_q. Clearly, it suffices to consider the following two cases:

$$(1)\ (r, q-1) = 1; \qquad (2)\ r | q-1.$$

If $(r, q-1) = 1$, then $\delta^{r^{-1}}$ is a rth root of δ. Therefore, it suffices to consider the case that $r|q-1$.

Adleman, Manders and Miller [1] had mentioned how to extend their square root extraction method to rth root extraction, but not specified it. We now clarify it as follows.

If $r|q-1$, we write $p-1$ in the form $r^t \cdot s$, where $(s, r) = 1$. Given a rth residue δ, we have $(\delta^s)^{r^{t-1}} = 1$. Since $(s, r) = 1$, it is easy to find the least nonnegative integer α such that $s | r\alpha - 1$. Hence,

$$\left(\delta^{r\alpha-1}\right)^{r^{t-1}} = 1 \tag{1}$$

If $t - 1 = 0$, then δ^α is a rth root of δ. From now on, we assume that $t \geq 2$.

Given a rth non-residue $\rho \in F_q$, we have

$$(\rho^s)^{i \cdot r^{t-1}} \neq (\rho^s)^{j \cdot r^{t-1}} \text{ where } i \neq j,\ i, j \in \{0, 1, \cdots, r-1\}$$

Table 3: Adleman-Manders-Miller cubic root extraction algorithm in F_{p^m}

Input: Odd prime p, a positive integer m and a cubic residue δ.

Output: A cubi root of δ.

Step 1: Choose ρ uniformly at random from $F_{p^m}^*$.

Step 2: **if** $\rho^{\frac{p^m-1}{3}} = 1$, **go to** Step 1.

Step 3: Compute t, s such that $p^m - 1 = 3^t s$, where $s = 3l \pm 1$.

Compute $a \leftarrow \rho^s, a' \leftarrow \rho^{3^{t-1} \cdot s}, b \leftarrow \delta^s$, $h \leftarrow 1$.

Step 4: **for** $i = 1$ **to** $t - 1$

compute $d = b^{3^{t-1-i}}$

if $d = 1$, $k \leftarrow 0$,

else if $d = a'$, $k \leftarrow 2$

else $k \leftarrow 1$

$b \leftarrow b \cdot (a^3)^k$, $h \leftarrow h \cdot a^k$

$a \leftarrow a^3$

end for

Step 5: $r \leftarrow \delta^l h$

if $s = 3l + 1$, $r \leftarrow r^{-1}$

return r

Set

$$K_i = (\rho^s)^{i \cdot r^{t-1}} \text{ and } K = \{K_0, K_1, \cdots, K_{r-1}\}$$

It is easy to find that all K_i satisfy $X^r = 1$. Since

$$\left(\left(\delta^{r\alpha-1}\right)^{r^{t-2}}\right)^r = 1$$

there is a unique $j_1 \in \{0, 1, \cdots, r-1\}$ such that

$$\left(\delta^{r\alpha-1}\right)^{r^{t-2}} = K_{r-j_1}$$

where $K_r = K_0$. Hence,

$$\left(\delta^{r\alpha-1}\right)^{r^{t-2}} K_{j_1} = 1$$

That is

$$\left(\delta^{r\alpha-1}\right)^{r^{t-2}} (\rho^s)^{j_1 \cdot r^{t-1}} = 1 \tag{2}$$

By the way, to obtain j_1 one has to solve a discrete logarithm.

Likewise, there is a unique $j_2 \in \{0, 1, \cdots, r-1\}$ such that

$$\left(\delta^{r\alpha-1}\right)^{r^{t-3}} (\rho^s)^{j_1 \cdot r^{t-2}} (\rho^s)^{j_2 \cdot r^{t-1}} = 1 \tag{3}$$

Consequently, we can obtain $j_1, \cdots, j_{t-1}$ such that

$$\left(\delta^{r\alpha-1}\right) (\rho^s)^{j_1 \cdot r} (\rho^s)^{j_2 \cdot r^2} \cdots (\rho^s)^{j_{t-1} \cdot r^{t-1}} = 1 \tag{4}$$

Thus, we have

$$(\delta^\alpha)^r \left((\rho^s)^{j_1 + j_2 \cdot r + \cdots j_{t-1} \cdot r^{t-2}}\right)^r = \delta \tag{5}$$

It means that

$$\delta^\alpha (\rho^s)^{j_1 + j_2 \cdot r + \cdots j_{t-1} \cdot r^{t-2}}$$

is a rth root of δ.

Table 4: Adleman-Manders-Miller rth root extraction algorithm in F_q

Input: F_q and a rth residue δ, $r|q-1$.

Output: A rth root of δ.

Step 1: Choose ρ uniformly at random from F_q^*.

Step 2: **if** $\rho^{\frac{q-1}{r}} = 1$, **go to** Step 1.

Step 3: Compute t, s such that $q - 1 = r^t s$, where $(r, s) = 1$.

Compute the least nonnegative integer α such that $s|r\alpha - 1$.

Compute $a \leftarrow \rho^{r^{t-1}s}, b \leftarrow \delta^{r\alpha-1}$, $c \leftarrow \rho^s$, $h \leftarrow 1$

Step 4: **for** $i = 1$ **to** $t - 1$

compute $d = b^{r^{t-1-i}}$

if $d = 1$, $j \leftarrow 0$,

else $j \leftarrow -\log_a d$ (compute the discrete logarithm)

$b \leftarrow b\,(c^r)^j$, $h \leftarrow h\,c^j$

$c \leftarrow c^r$

end for

Step 5: **return** $\delta^\alpha \cdot h$

6 Complexity analysis of Adleman-Manders-Miller rth Root Extraction Method

We now discuss the time estimate for this rth root extraction algorithm.

To find a rth non-residue ρ, it requires to check that $\rho^{\frac{q-1}{r}} \neq 1$. The computation takes time $\mathcal{O}(\log^3 q)$. If we do this for more than $\mathcal{O}(1)\log q$ different randomly chosen ρ, then with probability $> 1-(\frac{1}{q})^{\mathcal{O}(1)}$ at least one of them will give a rth non-residue. Therefore, the expected time of finding a rth non-residue is $\mathcal{O}(\log^4 q)$.

The work done outside the loop amounts to just a handful of exponentiations. Hence, it takes time $\mathcal{O}(\log^3 q)$. To compute $b^{r^{t-i-1}}$, it takes time $\mathcal{O}((t-i-1)\log r \log^2 q)$. Since there are $1+2+\cdots+(t-1)$ steps, it takes time $\mathcal{O}(t^2 \log r \log^2 q)$.

To compute the discrete logarithm $\log_a d$, it takes time $\mathcal{O}(r\log^2 q)$ using brute-force search. Since there are $t-1$ discrete logarithms at worst, it takes time $\mathcal{O}(tr\log^2 q)$.

Thus, the final estimate is $\mathcal{O}(\log^4 q + r\log^3 q)$. Notice that the algorithm can not run in polynomial time if r is sufficiently large.

7 Conclusion

The basic idea of Adleman-Manders-Miller root extraction method and its complexity analysis have not specified in the past decades. In this paper, we clarify the method and analyze its complexity. We think our heuristic presentation is helpful to grasp the method entirely and deeply.

Acknowledgements. We thank the anonymous referees' for their detailed suggestions. This work is supported by the National Natural Science Foundation of China (Project 60873227, 11171205), and the Key Disciplines of Shanghai Municipality (S30104).

References

[1] Adleman L., Manders K., Miller G.: On Taking Roots in Finite Fields. In: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 175–177. IEEE Press, New York (1977)

[2] Barreto P., Voloch J.: Efficient Computation of Roots in Finite Fields. Designs, Codes and Cryptography, 39, 275–280 (2006)

[3] Gathen J., Gerhard J.: Modern Computer Algebra, 2nd ed., Cambridge University Press (2003)

[4] Lidl R., Niederreiter H.: Introduction to finite fields and their applications, Cambridge University Press (1986)

[5] Shoup V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press (2005)

[6] Cipolla M.: 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] Lehmer D.: Computer technology applied to the theory of numbers, Studies in Number Theory, Englewood Cliffs, NJ: Pretice-Hall, pp. 117-151 (1969)

[8] Müller S.: On the computation of square roots in finite fields, Designs, Codes and Cryptography, 31, 301–312 (2004)

[9] Nishihara N., Harasawa R., Sueyoshi Y., and Kudo A.: A remark on the computation of cube roots in finite fields, `eprint.iacr.org/2009/457`

[10] Shanks D.: Five Number-theoretic Algorithms. In: Proc. 2nd Manitoba Conf., pp. 51–70. Numer. Math. (1972)

[11] Sze T., On taking square roots without quadratic nonresidues over finite fields, Mathematics of Computation, 80, 1797-1811 (2011)

[12] Tonelli A.: Bemerkungüber die Auflösung quadratischer Congruenzen. Nachrichten der Akademie der Wissenschaften in Göttingen. 344-346 (1891)