Memoirs of the Faculty of Engineering, Okayama University, Vol. 43, pp. 99-107, January 2009

A High-Speed Square Root Algorithm for Extension fields –Especially for Fast Extension Fields–

Hidehiro KATO *
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

Yasuyuki NOGAMI *
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

Yoshitaka MORIKAWA *
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

(Received November 19, 2008)

A square root (SQRT) algorithm in extension field $\mathbb{F}_{p^m}(m = r_0 r_1 \cdots r_{n-1} \cdot 2^d$, r_i : odd prime, d : positive integer) is proposed in this paper. First, a conventional SQRT algorithm, the Tonelli-Shanks algorithm, is modified to compute the inverse SQRT in $\mathbb{F}_{p^{2^d}}$, where most of the computations are performed in the corresponding subfields $\mathbb{F}_{p^{2^i}}$ for $0 \leqslant i \leqslant d-1$. Then the Frobenius mappings with addition chain are adopted for the proposed SQRT algorithm, in which a lot of computations in a given extension field $\mathbb{F}_{p^m}$ are also reduced to those in a proper subfield by the norm computations. Those reductions of the field degree increase efficiency in the SQRT implementation. The Tonelli-Shanks algorithm and the proposed algorithm in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$ were implemented on a Core2 (2.66 GHz) using the C++ programming language. The computer simulations showed that, on average, the proposed algorithm accelerated the SQRT computation by 6 times in $\mathbb{F}_{p^6}$, and by 10 times in $\mathbb{F}_{p^{10}}$, compared to the Tonelli-Shanks algorithm.

1 INTRODUCTION

The task of computing square roots (SQRTs) in a finite field $\mathbb{F}_{p^m}$ is a problem of considerable importance. Why is the SQRT important? Of course, we may simply be interested in the problem because it's there, but there are also applications to cryptography [1], [2]. In this paper, an efficient algorithm to compute the SQRT in $\mathbb{F}_{p^m}$ is proposed. First, however, we give a short overview of several conventional SQRT algorithms in a prime field $\mathbb{F}_p$. In order to find z such that $z^2 = x$ for a given square $x \in \mathbb{F}_p$, most of already known efficient methods are equivalent to or use the same basic ideas as either the Tonelli-Shanks [3] or the Cipolla-Lehmer algorithms [4]. The latter has the disadvantage that one has to require a quadratic non-residue (QNR) that depends on both p and x, while the QNR needed by the former can be reused for different x. This study is based on the Tonelli-Shanks algorithm.

The Tonelli-Shanks algorithm was originally devised in $\mathbb{F}_p$, but can easily be applied in $\mathbb{F}_{p^m}$ by simply replacing the operations in $\mathbb{F}_p$ with those in $\mathbb{F}_{p^m}$. We know the operations in $\mathbb{F}_{p^m}$ are more expensive than those in $\mathbb{F}_p$ for $m > 1$. For example, using the schoolbook method we need m^2 multiplications in $\mathbb{F}_p$ to compute one multiplication in $\mathbb{F}_{p^m}$. If directly applying the Tonelli-Shanks algorithm in $\mathbb{F}_{p^m}$, it will require a lot of computations. So, we need to develop a more efficient SQRT algorithm in extension fields $\mathbb{F}_{p^m}$.

An extension field $\mathbb{F}_{p^m}$ provides us flexibility to choose system parameters, such as characteristic p and extension degree m as well as irreducible polynomial. In what follows $m = r2^d$ $(r = r_0 r_1 \cdots r_{n-1})$ without further explanation.

If a given element $x \in \mathbb{F}_{p^m}$ is not a quadratic residue (QR), *i.e.* x has no SQRT in $\mathbb{F}_{p^m}$, then there is no need to compute its SQRTs. Before SQRT computations, we should thus use Euler's criterion $C^m(x) = x^{(p^m-1)/2}$ to

*{kato, nogami, morikawa}@trans.cne.okayama-u.ac.jp

This work is subjected to copyright.
All rights are reserved by this author/authors.

identify whether x is a QR or not, which is called QR test. Since $m = r2^d$, the exponent in Euler's criterion can be factorized into

$$\frac{p^m-1}{2} = e \cdot \frac{p^{2^d}-1}{2}, \quad e = 1+p^{2^d}+\cdots+(p^{2^d})^{r-1}. \quad (1)$$

Thus, $C^m(x)$ can be evaluated in the following steps:

$$C^m(x) = \bar{x}^{(p^{2^d}-1)/2}, \ \bar{x} = N^m_{2^d}(x) = x^e, \quad (2)$$

where $N^m_{2^d}(x)$, the norm of x with respect to the subfield $\mathbb{F}_{p^{2^d}}$, is always an element in $\mathbb{F}_{p^{2^d}}$. Instead of directly evaluating $C^m(x)$, this paper proposes a QR test by examining vector form of $\bar{x}^s$, where s is an odd number such that $p^{2^d}-1 = s2^T$ (T is an integer not less than 1). This method cuts the unnecessary computations for a QNR input.

The basic idea of the proposed SQRT algorithm is described as follows. From Eq.(2), *i.e.* $\bar{x} = x^e$, we have

$$\sqrt{x} = x^{(e+1)/2}\bar{x}^{-1/2}, \quad (3)$$

where e is an odd number because r in Eq.(1) is an odd number. In the proposed algorithm, the SQRT algorithm presented in [5], *i.e.* MW-ST algorithm, is used to compute the inverse SQRT $\bar{x}^{-1/2}$ in a given subfield $\mathbb{F}_{p^{2^d}}$, where most of the computations in $\mathbb{F}_{p^{2^d}}$ can be reduced to those in proper subfields $\mathbb{F}_{p^{2^{d-i}}}$ for $1 \leqslant i \leqslant d$. Therefore, the number of computation for $\bar{x}^{-1/2}$ is far smaller than that for $\sqrt{x}$. In addition, not only the binary method, but also the Frobenius mappings with addition chain are adopted for $x^{(e+1)/2}$ in Eq.(3). For exponentiation, we usually resort to the binary method, however, the number of computations for the Frobenius mapping $\phi(x) = x^p$ is far fewer than that for the binary method in important fast finite fields, such as optimal extension fields (OEFs) [7], all one polynomial fields (AOPFs) [8][9] and successive extension fields (SEFs) [10]. Thus, the square root $\sqrt{x}$ can be efficiently computed using the proposed algorithm in these extension fields. In addition, the simulation results show that the proposed SQRT algorithm is much faster than the conventional algorithm.

Throughout this paper, A_m, M_m, and ϕ_m denote an addition, multiplication, and Frobenius mapping, respectively, in $\mathbb{F}_{p^m}$, and $\#A_m$, $\#M_m$, and $\#\phi_m$ denote the respective numbers of these operations.

2 MW-ST ALGORITHM

Let us briefly review QR test and SQRT algorithm called MW-ST algorithm in $\mathbb{F}_{p^{2^d}}$[5].

2.1 Quadratic Residue (QR) Test In $\mathbb{F}_{p^{2^d}}$

Generally, Euler's criterion is used to identify whether or not a nonzero element x in $\mathbb{F}_{p^{2^d}}$ is a QR:

$$C^{2^d}(x) = x^{(p^{2^d}-1)/2} = \begin{cases} 1, & \text{if } x \text{ is a QR} \\ -1, & \text{if } x \text{ is a QNR} \end{cases}. \quad (4)$$

Find an integer $T \geqslant 0$ and an odd number s such that

$$(p^{2^d}-1)/2 = 2^T s, \quad (5)$$

and then we have

$$C^{2^d}(x) = x^{(p^{2^d}-1)/2} = (x^s)^{2^T} = x_0^{2^T} \text{ for } x_0 = x^s. \quad (6)$$

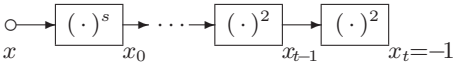
For the QR test of x, it is no need to compute the exponentiation $x^{(p^{2^d}-1)/2}$ directly. If $x_0 = 1$, then we can assert that x is a QR. If not, we repeatedly square from x_0 t times until -1 is obtained as shown in **Fig**.1(a). Note that we do not have to square x_0 for the case of $x_0 = -1$, which implies $t = 0$. If $t < T$, then x is a QR; if $t = T$, then x is a QNR. This QR test is a part of the authors' previous work[5]. When $x_0 = 1$, *i.e.* $x^s = 1$, multiplying both sides by x and taking SQRT of both sides results in $\sqrt{x} = x^{(s+1)/2}$. When computing x_0 in the above QR test, we thus first compute $x^{(s-1)/2}$, and then multiply $x^{(s-1)/2}$ by x to get $x^{(s+1)/2}$, finally multiply $x^{(s-1)/2}$ and $x^{(s+1)/2}$ together to get x_0, which can avoid the overlapping computations in QR test and SQRT computation.

2.2 MW-ST Algorithm In $\mathbb{F}_{p^{2^d}}$

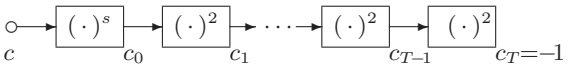
When $x_0 = 1$, *i.e.* $x^s = 1$, multiplying it by x and applying SQRT calculation to both sides, we have $\sqrt{x} = x^{(s+1)/2}$. In what follows, we mainly consider $x_0 \neq 1$. In this case, QR test is performed as described in **Sec**.2.1, where if $t < T$, as shown in **Fig**.1(a), then x is a QR. If not, as shown in **Fig**.1(b), then x denoted by c is a QNR.

From **Fig**.1, it is clear that $c_T x_t = 1$. Moving to the windows of c_{T-1} and x_{t-1}, we define $\sigma(1) = c_{T-1}x_{t-1}$. Since c_{T-1} and x_{t-1} are the SQRTs of c_T and x_t, respectively, and since the SQRT of $c_T x_t$ *i.e.* the SQRT of 1 must be 1 or -1 in any finite field, we have $\sigma(1) = \pm 1$. Then, we test the sign of $\sigma(1)$: if $\sigma(1) = 1$, moving to the windows of c_{T-2} and x_{t-2}, we define $\sigma(2) = c_{T-2}x_{t-2}$; if not, multiplying $\sigma(1)$ by c_T to get $c_T c_{T-1} x_t = 1$, and then moving to the windows of c_{T-1}, c_{T-2}, and x_{t-2}, we define $\sigma(2) = c_{T-1}c_{T-2}x_{t-2}$. This is the reason why the proposed SQRT algorithm is called moving window-sign testing (MW-ST) algorithm. In this way, for $1 \leqslant k \leqslant t$ we have

$$\sigma(k) = c_{T-1}^{i_{k-1}} c_{T-2}^{i_{k-2}} \cdots c_{T-k+1}^{i_1} c_{T-k} x_{t-k}. \quad (7)$$



(a) A power series of a QR



(b) A power series of a QNR ($t < T$)

Figure 1: QR test for $x \in \mathbb{F}_{p^{2^d}}$.

Same as $\sigma(1)$, we can assert $\sigma(k) = \pm 1$. If $\sigma(k) = 1$, then $i_k = 0$. If $\sigma(k) = -1$, then $i_k = 1$. For $k = t$, we have

$$\sigma(t) = c_{T-1}^{i_{t-1}} c_{T-2}^{i_{t-2}} \cdots c_{T-t+1}^{i_1} c_{T-t} x_0 = \pm 1. \quad (8)$$

Since $x_0 = x^s$ and $c_T = -1$,Eq.(8) can lead to

$$c_T^{i_t} c_{T-1}^{i_{t-1}} \cdots c_{T-t+1}^{i_1} c_{T-t} x^s = 1, \quad (9)$$

where for $1 \geqslant k \geqslant t$, i_k have the same definition as in Eq.(7). Multiplying both side of Eq.(9) by x and taking SQRT of both sides, we can easily get

$$\sqrt{x} = c_{T-1}^{i_t} \cdots c_{T-t}^{i_1} c_{T-t-1} x^{(s+1)/2}. \quad (10)$$

MW-ST algorithm is summarized as follows:

MW-ST Algorithm over $\mathbb{F}_{p^{2^d}}$

INPUT: A nonzero element $x \in \mathbb{F}_{p^{2^d}}$.

OUTPUT: A square root $\sqrt{x} \in \mathbb{F}_{p^{2^d}}$.

PRECOMPUTATION:

a. Factorize $(p^{2^d} - 1)/2$ as shown in Eq.(5).

b. Find an appropriate QNR $c \in \mathbb{F}_{p^{2^d}}$, compute $c_0, c_1, \cdots, c_T$ as shown in **Fig**.1(b), and save these values into the memory.

c. Execute the QR test as shown in **Fig**.1, and save the intermediate value of $x_0, x_1, \cdots, x_t$ and $x^{(s+1)/2}$ into the memory. If the input x is a QR execute the main procedure; if not, terminate.

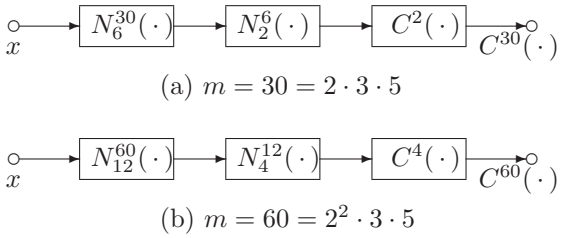
MAIN PROCEDURE:

1. Check whether x_0 equals to 1 or not. If $x_0 = 1$, output $\sqrt{x} = x^{(s+1)/2}$ and terminate; if not, execute **Main Procedure** 2 and 3 in order.
2. Let $\tau_0 = T - 1$, $\mu = 1$ and $k = 1$, and then repeatedly execute **Step** 1-3 until k becomes t.
   - **Step** 1 : Compute $\sigma = x_{t-k} \prod_{i=0}^{\mu-1} c_{\tau_i}$. If $\sigma = -1$, then $\tau_\mu = T - 1$ and $\mu = \mu + 1$; if not, then the values of τ_μ and μ are not modified.
   - **Step** 2 : Let $\tau_i = \tau_i - 1$ for $0 \leqslant < \mu$.
   - **Step** 3 : Let $k = k + 1$.
3. Compute the value of $\sqrt{x} = x^{(s+1)/2} \prod_{i=0}^{\mu-1} c_{\tau_i}$.

Remarks:

- The register τ_i is for the index memory of c_q appearing in Eq.(10).
- μ shows the number of c_q in Eq.(10).

As previously shown, MW-ST algorithm in $\mathbb{F}_{p^{2^d}}$ reduces the calculation cost by using the structure of Sylow 2-subgroup of order 2^t. In this paper, the authors propose SQRT algorithm for a more general case that $m = r_0 r_1 \cdots r_{n-1} 2^d$ where r_i is an odd prime number, and $r_i \geqslant r_j$ for $i \leqslant j$, and $d \geqslant 0$ is an integer. According to Eq.(10), using $x^{(s-1)/2}$ instead of $x^{(s+1)/2}$, we can obtain the inverse square root $x^{-\frac{1}{2}}$.



(a) $m = 30 = 2 \cdot 3 \cdot 5$

(b) $m = 60 = 2^2 \cdot 3 \cdot 5$

Figure 2: Some examples of the structure of $C^m(x)$

3 THE PROPOSED QR TEST IN $\mathbb{F}_{p^m}$

Same as $\mathbb{F}_{p^{2^d}}$, Euler's criterion is also used to identify whether or not a nonzero element $x \in \mathbb{F}_{p^m}$ is a QR.

$$C^m(x) = x^{(p^m-1)/2} = \begin{cases} 1, & \text{if } x \text{ is a QR} \\ -1, & \text{if } x \text{ is a QNR} \end{cases}. \quad (11)$$

In Eq.(11), $x^{(p^m-1)/2}$ can be directly computed by the binary method. However, in what follows, we give an efficient method for checking Eq.(11) that leads to the proposed efficient SQRT algorithm.

3.1 Generalized QR Test

For simplicity without loss of generality, assume $m = \bar{r}\bar{m}$ in this section, where $\bar{r}$ is an odd prime number and $\bar{m}$ is a composite number or 1, and then the exponent in Euler's criterion in Eq.(11) can be factorized as

$$\frac{p^m - 1}{2} = \left[1 + p^{\bar{m}} + \cdots + (p^{\bar{m}})^{\bar{r}-1}\right] \cdot \frac{(p^{\bar{m}} - 1)}{2}. \quad (12)$$

Thus, $C^m(x)$ can be evaluated in the following steps

$$C^m(x) = C^{\bar{m}}(\bar{x}),\ \bar{x} = N_{\bar{m}}^m(x) = x^{1+p^{\bar{m}}+\cdots+(p^{\bar{m}})^{\bar{r}-1}}. \quad (13)$$

Equation (13) shows that $C^m(x)$ will be reduced into $C^{\bar{m}}(\bar{x})$ by the norm computation of $N_{\bar{m}}^m(x)$. Since x is an element of an extension field $\mathbb{F}_{p^m}$ and $\bar{x} = N_{\bar{m}}^m(x)$ is an element of a subfield $\mathbb{F}_{p^{\bar{m}}}$, the number of computations for $C^{\bar{m}}(\bar{x})$ is thus far fewer than that for $C^m(x)$ *i.e.* $C^{\bar{r}\bar{m}}(x)$. As $\bar{r}$ becomes larger the efficiency of the reduction given by Eq.(13) becomes larger. Similar procedures can be applied to the remaining factors of $\bar{m}$. For example, the computation structures of $C^m(x)$ for $m = 30, 60$ are shown in **Fig**.2, in which the symbol $N_{\cdot}^{\cdot}(\cdot)$ denotes the norm computation as shown in Eq.(14).

Though $N_{\bar{m}}^m(x)$ can be computed by binary method, for efficiency, using the i-th iteration of the Frobenius mappings $\phi^{[i]}(x) = x^{p^i}$, $N_{\bar{m}}^m(x)$ in Eq.(13) can be expressed as follows:

$$N_{\bar{m}}^m(x) = \prod_{i=0}^{\bar{r}-1} \phi^{[i\bar{m}]}(x). \quad (14)$$

In what follows, for simplicity, the i-th iteration of the Frobenius mapping is regarded as the Frobenius mapping because their properties are all the same. Since the

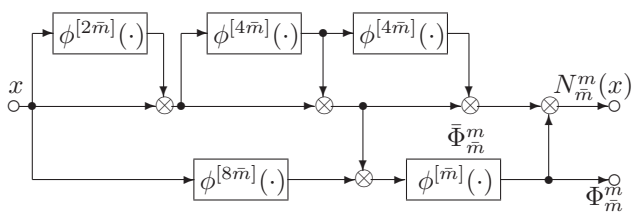


Figure 3: Addition chain to compute $N^m_{\bar{m}}(x)$ for $\bar{r}=11$.

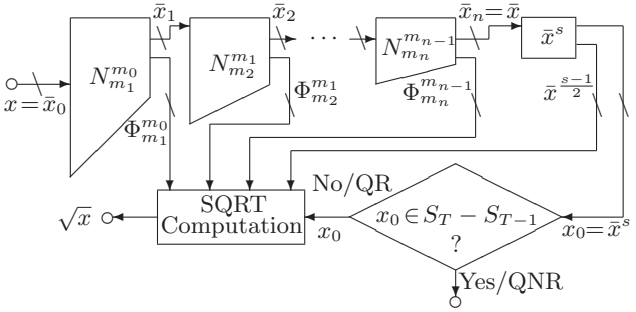


Figure 4: Schematic diagram for the proposed QR test.

Frobenius mapping $\phi^{[i\bar{m}]}(x)=x^{p^{i\bar{m}}}$ has the linearity

$$\phi^{[i\bar{m}]}(a\xi+b\zeta)=a\phi^{[i\bar{m}]}(\xi)+b\phi^{[i\bar{m}]}(\zeta), \tag{15}$$

for $\xi,\zeta\in\mathbb{F}_{p^m}$ and $a,b\in\mathbb{F}_p$. In addition, since any element $x\in\mathbb{F}_{p^m}$ is expressed as a linear combination of the basis, the Frobenius mapping of x can be easily computed when the Frobenius mappings of basis elements are easily obtained. In particular, the computational complexity for the Frobenius mapping of the basis is negligibly small in OEFs [7] and AOPFs [8].

Moreover, in order to increase the computational efficiency of $N^m_{\bar{m}}(x)$, we can adopt an addition chain, which reuses the previously obtained values of the Frobenius mappings. In the proposed addition chain, we first compute $\Phi^m_{\bar{m}}$ and $\bar{\Phi}^m_{\bar{m}}$, and then multiply them to obtain $N^m_{\bar{m}}(x)$, where

$$\Phi^m_{\bar{m}}=\prod_{i=1}^{(\bar{r}-1)/2}\phi^{[(2i-1)\bar{m}]}(x), \tag{16a}$$

$$\bar{\Phi}^m_{\bar{m}}=\prod_{i=0}^{(\bar{r}-1)/2}\phi^{[(2i)\bar{m}]}(x). \tag{16b}$$

In the proposed addition chain, we obtain and save the value of $\Phi^m_{\bar{m}}$ that is required for the proposed SQRT algorithm. An example of the chain for computing $N^m_{\bar{m}}(x)$ with $\bar{r}=m/\bar{m}=11$ is shown in **Fig**.3, where $\phi^{[i\bar{m}]}(\,\cdot\,)$ denotes the Frobenius mapping of the input $\cdot$ and $\otimes$ denotes the multiplication in $\mathbb{F}_{p^m}$. If $N^m_{\bar{m}}(x)$ is computed by directly using Eq.(14), then 10 multiplications and 10 Frobenius mappings in $\mathbb{F}_{p^m}$ are required. On the other hand, using the proposed chain, only 5 multiplications and 5 Frobenius mappings in $\mathbb{F}_{p^m}$ are required as shown in **Fig**.3.

Using the Frobenius mappings with the addition chain, $N^m_{\bar{m}}(x)$ requires

$$\#M_m=\#\phi_m=\lfloor\log_2(\bar{r})\rfloor+w(\bar{r})-1, \tag{17}$$

where $\lfloor\cdot\rfloor$ and $w(\cdot)$ denote the maximum integer not more than $\cdot$ and the Hamming weight of $\cdot$, respectively. In what follows, the expression $\lfloor\log_2(\cdot)\rfloor+w(\cdot)-1$ is abbreviated as $LW(\cdot)$ for convenience.

3.1.1 Degree Reduction By Odd Prime Factors In m

In the general case that $m=r_0r_1\cdots r_{n-1}2^d$, where r_i is an odd prime number, and $r_i\geqslant r_j$ for $i\leqslant j$ and $d\geqslant 0$, let $m_n=2^d$ and define

$$m_j=r_j\cdots r_{n-1}2^d,\quad 0\leqslant j\leqslant n-1. \tag{18}$$

Then, we have $m_0=m$. As in **Fig**.2, $C^m(x)$ can be reduced into $C^{2^d}(\bar{x}_n)$ by the series of norm computations $\bar{x}_{j+1}:=N^{m_j}_{m_{j+1}}(\bar{x}_j)$ with the Frobenius mappings

$$\bar{x}_{j+1}=N^{m_j}_{m_{j+1}}(\bar{x}_j)=\prod_{i=0}^{r_j-1}\phi^{[im_{j+1}]}(\bar{x}_j) \in\mathbb{F}_{p^{m_{j+1}}},\ 0\leqslant j\leqslant n-1, \tag{19}$$

where $\bar{x}_0=x$. When implement by the above computations, we first use the proposed addition chain to compute $\Phi^{m_i}_{m_{j+1}}$ and $\bar{\Phi}^{m_i}_{m_{j+1}}$, respectively, and then multiply them to get $\bar{x}_{j+1}$, where

$$\Phi^{m_j}_{m_{j+1}}=\prod_{i=1}^{(r_j-1)/2}\phi^{[(2i-1)m_{j+1}]}(\bar{x}_j), \tag{20a}$$

$$\bar{\Phi}^{m_j}_{m_{j+1}}=\prod_{i=0}^{(r_j-1)/2}\phi^{[(2i)m_{j+1}]}(\bar{x}_j). \tag{20b}$$

In the QR test, all the values of $\Phi^{m_j}_{m_{j+1}}$ $(0\leqslant j\leqslant n-1)$ will be saved as shown in **Fig**.4 because the following SQRT algorithm will use them.

From Eq.(19), we can see that the computations in a given extension field $\mathbb{F}_{p^m}$, *i.e.* $\mathbb{F}_{pr_0\cdots r_{n-1}2^d}$, can be reduced to those in proper subfields $\mathbb{F}_{p^{m_j}}$ $(1\leqslant j\leqslant n-1)$ by the series of norm computations. When taking the reduction by the prime factors in m, the largest r_0 is performed first because it will achieve the maximam reduction in computation cost.

3.1.2 Degree Reduction By Factors 2 In m

Find an integer $T>0$ and an odd number s such that

$$p^{2^d}-1=2^Ts. \tag{21}$$

The multiplicative group $\mathbb{F}^*_{p^{2^d}}$ is a cyclic group of order 2^Ts. Therefore, $\mathbb{F}^*_{p^{2^d}}$ contains a cyclic group of order 2^T, which is called Sylow 2-subgroup, and there is a descending chain of subgroups of S_T,

$$S_T\supset S_{T-1}\supset\cdots\supset S_2\supset S_1=\{\pm1\}\supset S_0=\{1\}. \tag{22}$$

For any QNR c in $\mathbb{F}_{p^m}$, $\bar{c}=c^e$ must be a QNR in $\mathbb{F}_{p^{2^d}}$, where e is given by Eq.(1). It follows that S_T of order 2^T is generated by $c_0:=(\bar{c})^s$, S_{T-1} of order 2^{T-1} is generated by $c_1:=(c_0)^2$, and in general, S_{T-k} of order 2^{T-k} is generated by $c_k:=(c_{k-1})^2$. In general, for $1\leqslant k\leqslant d-1$, we have

$$c_k\in S_{T-k}-S_{T-k-1}\subset\mathbb{F}_{p^{2^{d-k}}} \quad . \tag{23a}$$

For the other k, *i.e.* $d\leqslant k\leqslant T-2$, we have

$$c_k\in S_{T-k}-S_{T-k-1}\subset\mathbb{F}_p, \qquad \text{if } 4|(p-1), \tag{23b}$$
$$c_k\in S_{T-k}-S_{T-k-1}\subset\mathbb{F}_{p^2}, \qquad \text{if } 4\nmid(p-1), \tag{23c}$$

for $k=T-1$, we have

$$c_{T-1}\in S_1-S_0=\{-1\}\subset\mathbb{F}_p. \tag{23d}$$

This property is also introduced in [17]

3.1.3 QR Test In $\mathbb{F}_{p^{2^d}}$

In the conventional QR test, for a nonzero element $x\in\mathbb{F}_{p^m}$, we directly compute $C^m(x)=x^{(p^m-1)/2}$ to identify whether or not x is a QR. In the proposed QR test, the computation of $C^m(x)$ is reduced into that of $C^{2^d}(\bar{x})$ by a series of norm computations as described in 3.1.1, where $\bar{x}\in\mathbb{F}_{p^{2^d}}$. If $\bar{x}$ is a QR in $\mathbb{F}_{p^{2^d}}$, then x is also a QR in $\mathbb{F}_{p^m}$. If $\bar{x}$ is a QNR in $\mathbb{F}_{p^{2^d}}$, then x is also a QNR in $\mathbb{F}_{p^m}$. Since any element in S_T-S_{T-1} must always be a QNR in $\mathbb{F}_{p^{2^d}}$, and since $x_0=\bar{x}^s$ must belong to one of sets S_k-S_{k-1} for $1\leqslant k\leqslant T$, we only need to identify whether or not $x_0\in S_T-S_{T-1}$ as shown in **Fig**.4. From Eqs.(23), if $T>2$, or $T=1$ and $4|(p-1)$, then checking whether x_0 belongs to S_T-S_{T-1} is equivalent to checking whether it belongs to $\mathbb{F}_{p^{2^d}}-\mathbb{F}_{p^{2^{d-1}}}$.

If $\mathbb{F}_{p^m}$ is an OEF with polynomial basis or TypeI-X AOPF with normal basis, we can identify whether x_0 belongs to $\mathbb{F}_{p^{2^d}}-\mathbb{F}_{p^{2^{d-1}}}$ from the form of vector representation of x_0. For example, in TypeI-X AOPF $\mathbb{F}_{p^{2^d}}$ with normal basis, an element A in the subfield $\mathbb{F}_{p^{2^{d-1}}}$ is represented by

$$A=\{a_0,a_1,\ldots,a_{2^{d-1}-1},a_0,a_1,\ldots,a_{2^{d-1}-1}\}. \tag{24}$$

For a more general case, an element A in $\mathbb{F}_{p^{2^{d-i}}}$ has $i+1$ repetitions of vector coefficients. Moreover, there exist a lot of extension fields isomorphic with TypeI-X AOPF, *i.e.* p and m are identical to those of TypeI-X AOPF but only modular polynomial is different from the all one polynomial. Even in those cases, we can apply the above QR test after performing basis translation from the objective field to TypeI-X AOPF [12][14].

3.2 COMPLEXITY OF THE QR TEST

When we directly compute $x^{(p^m-1)/2}$ for QR test by binary method, it requires

$$\#M_m=LW\left(\frac{p^m-1}{2}\right). \tag{25}$$

In the proposed QR test, we first carry out a series of norm computations $N^{m_j}_{m_{j+1}}(\cdot)$ for $0\leqslant j\leqslant n-1$ to get $\bar{x}$, using the Frobenius mappings with a certain addition chain. From Eq.(17), the norm computations $N^{m_j}_{m_{j+1}}(\cdot)$ for $0\leqslant j\leqslant n-1$ in total require the sum of

$$\#\phi_{m_j}=\#M_{m_j}=LW(r_j), \quad 0\leqslant j\leqslant n-1, \tag{26}$$

where m_j is given by Eq.(18).

Then, we compute $\bar{x}^s$. To avoid the overlapping computations in QR test and SQRT computation, we first compute $\bar{x}^{(s-1)/2}$, and then take its square to get $\bar{x}^{s-1}$, finally multiply $\bar{x}^{s-1}$ by $\bar{x}$ to get $\bar{x}^s$ (*see* **Fig**.4). The computation of $\bar{x}^s$ requires

$$\#M_{2^d}=LW\left(\frac{s_1-1}{2}\right)+\frac{d(d+3)}{2}+(d-1)U(p), \tag{27a}$$
$$\#\phi_{2^d}=\frac{d(d-1)}{2}, \tag{27b}$$

where s_1 is the odd number such that $p^2=2^{T_1}s_1+1$ and $U(p)$ is defined as follows (*see* Appendix A);

$$U(p)=\begin{cases} LW\left(\dfrac{p-1}{4}\right)+1, & \text{if } 4|(p-1),\\ LW\left(\dfrac{p-3}{4}\right)+LW\left(\dfrac{p+1}{2}\right)+2, & \text{otherwise,}\end{cases} \tag{28a}$$

Finally, we identify whether or not $\bar{x}^s=x_0\in S_T-S_{T-1}\subset\mathbb{F}_{p^{2^d}}$ from the form of x_0, which almost does not need a computation as described in Sect.3.1.3.

4 PROPOSED SQRT ALGORITHM IN $\mathbb{F}_{p^m}$

4.1 SQRT Algorithm

From the norm definition Eqs.(13) and Eq.(19), we have

$$\bar{x}_{j+1}=\bar{x}_j^{(p^{m_{j+1}})^{r_j-1}+\cdots+p^{m_{j+1}}+1}. \tag{29}$$

Dividing both sides by $\bar{x}_j\bar{x}_{j+1}$ and then taking SQRTs of both sides, for $0\leqslant j\leqslant n-1$, we have

$$(\bar{x}_j)^{-\frac{1}{2}}=(\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}\cdot(\bar{x}_{j+1})^{-\frac{1}{2}}, \tag{30}$$

where $\Phi^{m_j}_{m_{j+1}}$ is given by Eq.(20a). It follows that

$$(\bar{x}_0)^{-\frac{1}{2}}=(\bar{x}_n)^{-\frac{1}{2}}\cdot\prod_{j=0}^{n-1}(\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}, \tag{31}$$

where $\bar{x}_0=x$ and $\bar{x}_n=\bar{x}$. Therefore, we have

$$\sqrt{x}=\bar{x}^{-\frac{1}{2}}\cdot x\cdot\prod_{j=0}^{n-1}(\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}. \tag{32}$$

When implementing the exponentiation with $(1+p^{m_{j+1}})/2$, which can be expressed as

$$\frac{p^{m_{j+1}}+1}{2}=\left(1+p+\cdots+p^{(m_{j+1})-1}\right)\cdot\frac{p-1}{2}+1, \tag{33}$$

we apply the Frobenius mappings with the addition chain to the part in the parenthesis, and then apply

the binary method for $\left(\frac{p-1}{2}\right)$–th power followed by a multiplication.

Based on Eq.(32), the SQRT in $\mathbb{F}_{p^m}$ can be efficiently computed using the following algorithm via the Frobenius mappings with the addition chain:

Proposed SQRT Algorithm over $\mathbb{F}_{p^m}$

INPUT: An odd prime number p and an integer m and a random nonzero element $x \in \mathbb{F}_{p^m}$.

OUTPUT: A SQRT $z = \sqrt{x} \in \mathbb{F}_{p^m}$ such that $z^2 \equiv x$, or "UNSOLVABLE" if no such solution exists.

PRECOMPUTATION: Obtain all factors of extension degree m as $m = r_0 r_1 \cdots r_{n-1} 2^d$ and factorize the order of $\mathbb{F}^*_{p^{2^d}}$ as Eq.(21).

MAIN PROCEDURE:

1. If $x = 1$ then return 1. Otherwise, execute the proposed QR test in Sect.3.1. If the input x is a QR then save the values of $\bar{x}$, $\alpha_j \leftarrow \Phi^{m_j}_{m_{j+1}}$ for $0 \leqslant j \leqslant n-1$, and else return "UNSOLVABLE". (*see* Eq.(19)). Note $\bar{x} = \bar{x}_n = x^e \in \mathbb{F}_{p^{2^d}}$.
2. $z \leftarrow \bar{x}^{-\frac{1}{2}}$ using MW-ST algorithm[5]
3. About the computations for $x \cdot \prod_{j=0}^{n-1} (\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}$ in Eq.(32):

   $\beta_j \leftarrow \alpha_j^{\frac{1+p^{m_{j+1}}}{2}}$, for each $0 \leqslant j \leqslant n-1$,
   and $\gamma \leftarrow \prod_{j=0}^{n-1} \beta_j$.
4. $z \leftarrow zx\gamma$.
5. Return(z).

4.2 Complexity Of The Proposed SQRT Algorithm

In the proposed SQRT algorithm, Step 1 is just the QR test whose complexity has been evaluated in Sect.3.2. Since α_j has been computed in the QR test, we do not count the complexity of Step 1.

In Step 2, the value $\bar{x} \in \mathbb{F}_{p^{2^d}}$ has been computed in the QR test, recomputation by Step 2 is thus not necessary. We only need to modify the MW-ST algorithm [5] to compute $\bar{x}^{-1/2}$ in $\mathbb{F}_{p^{2^d}}$. As described in [5] [1], when $4 \mid (p-1)$ and $d > 1$, Step 2 on average requires the sum of

$$\#M_1 = \frac{T^2 - T + d^2 + 5d}{4} - \frac{2^d(d^2+3d)}{2^T}, \tag{34a}$$

$$\#M_{2^{d+1-j}} = \frac{j^2+3j-2}{4}, \quad 1 \leqslant j \leqslant d. \tag{34b}$$

[1] Also in MW-ST algorithm, we take advantage of degree reduction of the descending subgroup chain.

When $4 \nmid (p-1)$ and $d > 2$, Step 2 requires the sum of

$$\#M_2 = \frac{T^2 - T + d^2 + 3d - 4}{4} - \frac{2^d(d^2+d-2)}{2^{T+1}}, \tag{35a}$$

$$\#M_{2^{d+1-j}} = \frac{j^2+3j-2}{4}, \ 1 \leqslant j \leqslant d-1. \tag{35b}$$

In Step 3, we take the $(1 + p^{m_{j+1}})/2$-th power of α for each $0 \leqslant j \leqslant n-1$ and multiply them together. Using the binary method and the Frobenius mappings with the addition chain, Step 3 requires the sum of the following computations:

$$\#\phi_{m_j} = \#M_{m_j} = LW(m_{j+1}), \ 0 \leqslant j \leqslant n-1, \tag{36a}$$

$$\#M_{m_j} = LW\left(\frac{p-1}{2}\right) + 1, \ 0 \leqslant j \leqslant n-1, \tag{36b}$$

$$\#M_{m_j} = r_{j-1}, \ 1 \leqslant j \leqslant n-1, \tag{36c}$$

where m_j is given by Eq.(18). Since $x, \ \gamma \in \mathbb{F}_{p^m}$ and $z \in \mathbb{F}_{p^{2^d}}$, Step 4 requires

$$\#M_m = 1, \tag{37a}$$

$$\#M_{2^d} = r_0 r_1 \cdots r_{n-1}. \tag{37b}$$

4.3 Complexity Of The Tonelli-Shanks Algorithm

From Eqs.(1) and Eq.(21), it follows that

$$p^m - 1 = e \cdot s \cdot 2^T \tag{38}$$

Based on the result in [13], the average complexity of the Tonelli-Shanks algorithm over $\mathbb{F}_{p^m}$ is given by

$$\#M_m = \frac{1}{4}(T^2 + 7T - 16) + \frac{1}{2^{T-1}} + LW\left(\frac{e \cdot s - 1}{2}\right) + 2. \tag{39}$$

5 COMPUTER SIMULATIONS

Pairing-based cryptographic applications over *pairing-friendly* elliptic curves have received much attention. They are defined over a certain extension field. For example, MNT(Miyaji-Nakabayashi-Takano) curve [16] and Freeman's curve [15] are defined over $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$, respectively. They need some SQRT calculations for preparing rational points. Thus, in this section, let us simulate SQRT calculations over these extension field with the following parameters:

$$m = 6 = 2 \times 3, \tag{40a}$$

$$p = 53956\ 14237\ 76153\ 20457\ 34007\ 60106\ 31315\ 18176\ 97922\ 60564\ 49333\ 63744\ 98577\ [216\text{bits}]. \tag{40b}$$

$$m = 10 = 2 \times 5, \tag{41a}$$

$$p = 61099\ 96327\ 10831\ 28746\ 07376\ 95679\ 44870\ 35427\ 01616\ 46150\ 91479\ 4603\ [196\text{bits}], \tag{41b}$$

Table 1: Computational Complexity

Field $\mathbb{F}_{p^m}$	Method	A. Complexity				
		$\#\phi_2$	$\#\phi_m$	$\#M_1$	$\#M_2$	$\#M_m$
$m=6$ p : 216bits	Conventional QR test	–	–	–	–	1957
	Proposed QR test	–	2	–	1244	2
	Tonelli-Shanks algorithm	–	–	–	–	1974
	Proposed SQRT algorithm	–	–	–	12	325
$m=10$ p : 196bits	Conventional QR test	–	–	–	–	2943
	Proposed QR test	–	3	–	1174	3
	Tonelli-Shanks algorithm	–	–	–	–	2949
	Proposed SQRT algorithm	–	–	–	4	295

SQRT calculations are required for preparing rational points on the pairing-friendly curve.

The conventional QR test, the proposed QR test, the plain Tonelli-Shanks algorithm and the proposed SQRT algorithm over the extension fields $\mathbb{F}_{p^6}$, and $\mathbb{F}_{p^{10}}$ were implemented on a Core2 (2.66 GHz) computer using the C++ programming language with the number theory library (NTL).

The authors constructed the target extension field $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$ as TypeI-X all one polynomial field, because we can easily prepare subfield arithmetic operations in TypeI-X AOPF [14].

Based on Eqs.(21) and (40), we get T and s. Then, from Eq.(38), we know T_1 and s_1. Inputting p, m, T, s, T_1 and s_1 to Eqs.(25), (26), (27), (34), (36), (37), and (39), we explicitly evaluate the complexity of the algorithms over $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$ as shown in the column A of **Table** 1. **Table** 2 shows the number of algebraic operations required for ϕ_i and M_j, where $i=2,\ 6,\ 10$ and $j=1,\ 2,\ 6,\ 10$. According to the data in **Table** 2, we know $\#A_1$ and $\#M_1$ in the column A of **Table** 3.

From the column A in **Table** 1, we see that $\#M_m$ required in the proposed SQRT algorithm is much smaller than that in the Tonelli-Shanks algorithm, primarily because in the proposed SQRT algorithm, most of multiplications in a given extension field are replaced by those in its proper subfields.

Inputting a lot of random QRs, the computation time for the algorithms was measured as shown in the column B of **Table** 3. The column A of **Table** 3 shows that, using the proposed QR test, the numbers of computations in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$ show about 10-fold and 30-fold reductions, compared to using the conventional QR test, respectively. Using the proposed SQRT algorithm, the numbers of computations in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{10}}$ show 6-fold and 10-fold reductions for $p \cong 2^{216}$ and $p \cong 2^{196}$, respectively, compared to using the Tonelli-Shanks algorithm. The computer simulations show that, on average, the proposed QR test accelerates the QR test by 10 times in $\mathbb{F}_{p^6}$ and by 30 times in $\mathbb{F}_{p^{10}}$, compared to the conventional QR test. They also show that, on average, the proposed algorithm accelerates the SQRT computation by 6 times in $\mathbb{F}_{p^6}$ and by 10 times in $\mathbb{F}_{p^{10}}$, compared to the Tonelli-Shanks algorithm, which is supported by the evaluation of the number of computations.

6 CONCLUSION

This paper has proposed an efficient SQRT algorithm over $\mathbb{F}_{p^m}$ based on Eq.(32). Although the main idea of the proposed SQRT algorithm is based on the Tonelli-Shanks algorithm, in the proposed SQRT algorithm over $\mathbb{F}_{p^m}$, most of the computations required in extension fields $\mathbb{F}_{p^m}$ can be reduced to those in proper subfields $\mathbb{F}_{p^{m_j}}$ and $\mathbb{F}_{p^{2^d-i}}$ for $1 \leqslant i \leqslant d$, where

$$m = r_0 \cdots r_{n-1} 2^d,\ m_j = r_j \cdots r_{n-1} 2^d,\ 0 \leqslant j \leqslant n \quad (42)$$

However, all computations required for the plain Tonelli-Shanks algorithm over $\mathbb{F}_{p^m}$ must be executed in extension fields $\mathbb{F}_{p^m}$. In addition, the proposed SQRT algorithm can reuse the intermediate data of the QR test, such as $\Phi_{m_{j+1}}^{m_j}$ for $0 \leqslant j \leqslant n-1$. The computer simulations showed that, on average, the proposed algorithm accelerates the SQRT computation by 6 times in $\mathbb{F}_{p^6}$ and by 10 times in $\mathbb{F}_{p^{10}}$, compared to the Tonelli-Shanks algorithm, which is supported by the evaluation of the number of computations. This concludes that the proposed SQRT algorithm over $\mathbb{F}_{p^m}$ is very efficient.

REFERENCES

[1] D. Hankerson, A. Menezes and S. Vanstone:Guide to Elliptic Curve Cryptography, Springer (2003).

[2] K. Kurosawa, T. Ito and M. Takeuchi:Cryptologia, 12-4 (1988), 225-233.

[3] A. Tonelli:Bemerkung über die Auflösung quadratischer Congruenzen, Göttinger Nachrichten (1891), 344-346.

[4] M. Cipolla: "Un metodo per la risolutione della congruenza di secondo grado," Rendiconto

Table 2: The number of algebraic operations required for ϕ_i and M_j, where $i=2,\ 6,\ 10$ and $j=1,\ 2,\ 6,\ 10$.

	ϕ_2	ϕ_6	ϕ_{10}	$^{\dagger}M_2$	$^{\ddagger}M_2$	M_6	M_{10}
$\#A_1$	–	–	–	9	21	57	245
$\#M_1$	–	–	–	4	4	22	56

†subfield of $\mathbb{F}_{p^6}$, ‡subfield of $\mathbb{F}_{p^{10}}$

Table 3: Computational Amount and Running Time (CPU: Core2 (2.66GHz))

Field $\mathbb{F}_{p^m}$	Method	A. Amount $\#A_1$	A. Amount $\#M_1$	B. Time [μs]
$m=6$, p : 216bits	Conventional QR test	111549	43054	4.1×10^4
	Proposed QR test	11310	5020	4.1×10^3
	Tonelli-Shanks algorithm	112518	43428	3.9×10^4
	Proposed algorithm	18633	7198	6.6×10^3
$m=10$, p : 196bits	Conventional QR test	721035	164808	1.9×10^5
	Proposed QR test	25389	4864	5.7×10^3
	Tonelli-Shanks algorithm	722505	165144	1.9×10^5
	Proposed algorithm	72359	16536	1.9×10^4

dell'Accademia Scienze Fisiche e Matematiche, 9-3 (1903), 154-163.

[5] F. Wang, Y. Nogami and Y. Morikawa:IEICE Trans., E88-A-10 (2005), 2792-2799.

[6] F. Wang, Y. Nogami and Y. Morikawa:Proc. ICICS2003 LNCS2836 (2003), 1-10.

[7] D. V. Bailey and C. Paar:Proc. Crypto., (1998), 472-485.

[8] Y. Nogami, A. Saito and Y. Morikawa:IEICE Trans., E86-A-9 (2003), 2376-2387.

[9] Y. Nogami, S. Shinonaga and Y. Morikawa:IEICE Trans., E88-A-5 (2005), 1200-1208.

[10] J. L. Fan and C. Paar:Proc. ISIT1997, (1997), 20.

[11] T. Yoshida, Y. Nogami and Y. Morikawa:Proc. CSS2006, (2006), 43-48.

[12] Y. Nogami, R. Namba and Y. Morikawa:ETRI journal, 30-2 (2008), 326-334.

[13] S. Lindhurst:CRM Proceeding and Lecture Notes, 19 (1999), 231-242.

[14] H. Kato, Y. Nogami, T. Yoshida and Y. Morikawa:ETRI journal, 29-6 (2007), 769-778.

[15] D. Freeman:In Algorithmic Number Theory Symposium ANTS-VII, LNCS4076, (2006), 452-465.

[16] A. Miyaji, M. Nakabayashi and S. Takano:IEICE Trans., E84-A-5 (2001), 1234-1243.

[17] E. Berlekamp:Algebraic Coding Theory, McGraw-Hill, (1968).

A ALGORITHM FOR $\bar{x}^s$

For $1\leqslant i\leqslant d$, find positive integer T_i and odd integer s_i such that $p^{2^i}-1=s_i2^{T_i}$. Comparing to Eq.(21), we see $s_d=s$, $T_d=T$, and so $\bar{x}^{(s-1)/2}=\bar{x}^{(s_d-1)/2}$. As p is odd prime, $(p^{2^{i-1}}+1)/2$ is also odd for $i\geqslant 1$, implying

$$s_i=s_{i-1}\frac{p^{2^{i-1}}+1}{2},\ 1\leqslant i\leqslant d. \tag{A.43}$$

On the other hand,

$$\frac{p^2+1}{2}=\frac{p^2-1}{2}+1, \tag{A.44a}$$

and for $3\leqslant i\leqslant d$,

$$\frac{p^{2^{i-1}}+1}{2}=\frac{p^2-1}{2}\cdot\prod_{k=1}^{i-2}(p^{2^k}+1)+1. \tag{A.44b}$$

From Eqs.(A.43) and Eq.(A.44), we obtain

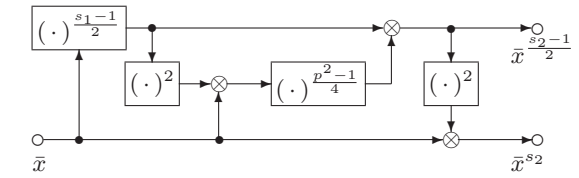
$$\frac{s_2-1}{2}=\left(\frac{s_1-1}{2}\cdot 2+1\right)\frac{p^2-1}{4}+\frac{s_1-1}{2}, \tag{A.45a}$$

and for $3\leqslant i\leqslant d$,

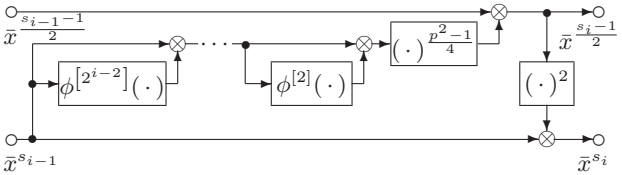
$$\frac{s_i-1}{2}=s_{i-1}\cdot\frac{p^2-1}{4}\cdot\prod_{k=1}^{i-2}(p^{2^k}+1)+\frac{s_{i-1}-1}{2}. \tag{A.45b}$$

Equation (A.45a) implies that $\bar{x}^{(s_2-1)/2}$ can be computed by first taking the $(s_1-1)/2$-th powers of $\bar{x}$ and its

square, second multiplying x^{s_1-1} and x together, third taking $(p^2-1)/4$-th power of $\bar{x}^{s_1}$ and finally multiplying $\bar{x}^{s_1(p^2-1)/4}$ and $\bar{x}^{(s_1-1)/2}$ together, as shown in Fig.5(a). Note we may simultaneously get $\bar{x}^{s_2}$ by multiplying $\bar{x}$ and the square of $\bar{x}^{(s_2-1)/2}$. Similarly Eq.(A.45b) implies that a pair of $\bar{x}^{(s_i-1)/2}$ and $\bar{x}^{s_i}$ can be computed, as shown in Fig.5(b), using the Frobenius mappings with the input pair of $\bar{x}^{(s_{i-1}-1)/2}$ and $\bar{x}^{s_{i-1}}$. The objective value of $\bar{x}^{s_d}$ is obtained at the last stage of ladder connections of the units shown in Fig.5(b) preceded by the unit shown in Fig.5(a).



(a) 1-st unit



(b) i-th unit, $3 \leqslant i \leqslant d$

Figure 5: Addition chain units for computing $\bar{x}^{s_d}$.

B ALGORITHM FOR $\bar{x}^s$

Since $\bar{x} \in \mathbb{F}_{p^{2^d}}$ as describe in Sect.3, all the arithmetic operations for $\bar{x}$ are implemented in $\mathbb{F}_{p^{2^d}}$. We begin with the first unit shown in Fig.5(a). From the Figure we see that the required operations are only

$$\#M_{2^d} = LW\left(\frac{s_1-1}{2}\right) + LW\left(\frac{p^2-1}{4}\right) + 5, \quad \text{(A.46)}$$

where we counted a square as 1 multiplication.

On the other hand, the i-th unit, shown in Fig.5 (b), consists of $(i-2)$ Frobenius mappings, a $(p^2-1)/4$-th power, i multiplications, and a square, which totally require the following operations

$$\#M_{2^d} = LW\left(\frac{p^2-1}{4}\right) + i + 1, \quad \text{(A.47a)}$$

$$\#\phi_{2^d} = i - 2. \quad \text{(A.47b)}$$

Adding up Eqs.(A.47) for $3 \leqslant i \leqslant d$ and Eq.(A.46), the total counts for computation of $\bar{x}^{s_d}$ result in

$$\#M_{2^d} = (d-1) \cdot LW\left(\frac{p^2-1}{4}\right) + LW\left(\frac{s_1-1}{2}\right) + \frac{d(d+3)}{2}, \quad \text{(A.48a)}$$

$$\#\phi_{2^d} = \frac{(d-1)(d-2)}{2}. \quad \text{(A.48b)}$$

However, a room of applying the Frobenius mapping remains in the $(p^2-1)/4$-th power in Eq.(A.48a).

· When $4|(p-1)$, as $(p-1)/4$ is an integer and $(p+1)$-th power is implemented by 1 Frobenius mapping and 1 multiplication, $LW((p^2-1)/4)$ may be replaced as

$$LW\left(\frac{p^2-1}{4}\right) M_{2^d} \leftarrow \left\{LW\left(\frac{p-1}{4}\right)+1\right\} M_{d^2} + \phi_{d^2}. \quad \text{(A.49a)}$$

· When $4 \nmid (p-1)$, we have

$$\frac{p^2-1}{4} = \frac{p-3}{4}(p+1) + \frac{p+1}{2}, \quad \text{(A.49b)}$$

and so the following replacement may be induced,

$$LW\left(\frac{p^2-1}{4}\right) M_{2^d} \leftarrow \left\{LW\left(\frac{p-3}{4}\right) + LW\left(\frac{p+1}{2}\right) + 2\right\} M_{d^2} + \phi_{d^2}. \quad \text{(A.49c)}$$

Equation (A.48a) with the replacement of Eqs.(A.49) leads us to Eqs.(27) in the main part of the paper.