Memoirs of the Faculty of Engineering, Okayama University, Vol. 42, pp. 36-43, January 2008

Fast Exponentiation in Extension Field with Frobenius Mappings

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

Kenta NEKADO
Faculty of Engineering, Okayama University 3-1-1, Tsushima-naka Okayama Japan

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

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

(Received November 22, 2007)

Abstract

This paper proposes an exponentiation method with Frobenius mappings. Our method is closely related to so-called *interleaving exponentiation.* Different from the interleaving exponentiation methods, our method can carry out several exponentiations using same base at the same time. The efficiency to use Frobenius mappings for an exponentiation in extension field is well introduced by Avanzi and Mihailescu. This exponentiation method is based on so-called *simultaneous exponentiation* and uses many Frobenius mappings. Their method more decreased the number of multiplications; however, the number of Frobenius mappings inversely increased. Compared to their method , the number of multiplications needed for the proposed method becomes about 20% larger; however, that of Frobenius mappings becomes small enough.

Keywords: *exponentiation, Frobenius mapping, extension field.*

1 Introduction

Recently, pairing–based cryptographic applications such as ID–based cryptography [1], group signature authentication [2], and broadcast encryption [3] have been proposed. Pairings such as Weil, Tate, Eta, and Ate pairings are bilinear mappings from two rational points on a certain elliptic curve to a non-zero element in a certain extension field [4]–[6]. In general, a pairing calculation needs an exponentiation in extension field and the computation time of the exponentiation is about half of the total computation time of a pairing calculation. Such an exponentiation in extension field is one of targets of this paper. This paper proposes a new exponentiation method with Frobenius mappings.

The most widely used binary method calculates exponentiation a^n by efficiently using the binary representation the exponent n. It iterates squaring and multiplying. The binary method needs $\lfloor \log_2 n \rfloor$ squares and $\lfloor \log_2 n \rfloor/2$ multiplications on average. The sliding window method calculates exponentiations more efficiently; however, it still needs $\lfloor \log_2 n \rfloor - w$ squares, where w is the window size. In addition, it is not so efficient when the base a often changes. Our exponentiation method is closely related to so-called *interleaving exponentiation method* [7]. Different from the interleaving exponentiation methods, our method can carry out several exponentiations using same base at the same time. For example, when $A \in \mathbb{F}_{p^m}$ and x, y and z are large positive integers, our exponentiation can carry out A^x, A^y and A^z at the same time. The efficiency to use Frobenius mappings for an exponentiation in extension field is well introduced by Avanzi and Mihailescu [8]. This exponentiation method is based on so-called *simul-*

taneous exponentiation method [7] and uses Frobenius mappings. When the exponent n is enough larger than characteristic p and a Frobenius mapping is carried out efficiently in the concerned extension field, using Frobenius mappings is efficient because the number of squares required for an exponentiation decreases into $\lfloor \log_2(p-1) \rfloor$. On the other hand, $\lfloor \log_2 n \rfloor / 2$ multiplications on average are still needed. In order to decrease the number of multiplications, it is efficient to apply the sliding window method in addition to Frobenius mappings. Based on this idea, Avanzi and Mihailescu [8] have proposed an efficient exponentiation method. This method more decreased the number of multiplications; however, the number of Frobenius mappings inversely increased. Thus, it still has two problems: 1) it is not so efficient when the base a often changes and 2) it is not so efficient when a Frobenius mapping cannot be fast carried out in the concerned extension field.

In order to overcome the above problems, this paper proposes an exponentiation method in extension field that efficiently uses Frobenius mappings but is not based on the sliding window method. Compared to Avanzi's method [8], the number of multiplications needed for the proposed method becomes about 20% larger; however, that of Frobenius mappings is small enough. Evaluating the calculation costs, it is shown that the proposed exponentiation method is enough practical for the case that the base of exponentiation is often changed and a Frobenius mapping cannot be fast carried out in the concerned extension field. In this paper, we deal with an extension field $\mathbb{F}_{p^m}$ over a prime field $\mathbb{F}_p$, for instance; however, the proposed method can be easily applied for an extension field $\mathbb{F}_{q^m}$ over some extension field $\mathbb{F}_q, q = p^i$. In addition, the proposed method is also efficiently applied for a scalar multiplication of rational point of elliptic curve defined over extension field, for example.

Throughout this paper, $\#_{\text{SADD}}$ and $\#_{\text{SMUL}}$ denote the number of additions and that of multiplications, respectively. In this paper, a subtraction in $\mathbb{F}_p$ is counted up as an addition in $\mathbb{F}_p$. p and m denote the characteristic and the extension degree, respectively, where p is a prime number. $\mathbb{F}_{p^m}$ denotes an m-th extension field over $\mathbb{F}_p$ and $\mathbb{F}^*_{p^m}$ denotes the multiplicative group in $\mathbb{F}_{p^m}$. Without any additional explanation, lower and upper case letters show elements in prime field and extension field, respectively, and a Greek character shows a zero of modular polynomial.

2 Preparation

In this section, let us briefly go over basis of extension field, Frobenius mapping, Tate pairing, binary method, and sliding window method.

2.1 Basis of extension field

In order to construct the arithmetic operations in $\mathbb{F}_{p^m}$, we generally need an irreducible polynomial $f(x)$ of degree m over $\mathbb{F}_p$. Let ω be a zero of $f(x)$, that is a proper element[1] in $\mathbb{F}_{p^m}$, then the following set forms a basis of $\mathbb{F}_{p^m}$over $\mathbb{F}_p$;

$$\left\{1, \omega, \omega^2, \cdots, \omega^{m-1}\right\}, \tag{1}$$

which is called *polynomial basis*. An arbitrary element A in $\mathbb{F}_{p^m}$ is written as

$$A = a_0 + a_1\omega + \cdots + a_{m-1}\omega^{m-1}. \tag{2}$$

The vector representation of A is $\boldsymbol{v}_A = (a_0, a_2 \cdots, a_{m-1})$. A multiplication and inversion in $\mathbb{F}_{p^m}$ are carried out by using the relation $f(\omega) = 0$, and therefore $f(x)$ is called the *modular polynomial* of $\mathbb{F}_{p^m}$. When the following conjugates of ω with respect to F_p are linearly independent;

$$\left\{\omega, \omega^p, \omega^{p^2}, \cdots, \omega^{p^{m-1}}\right\}, \tag{3}$$

this set Eq.(3) is called *normal basis*, which is efficient for *Frobenius mapping.*

Consider a normal basis Eq.(3) in $\mathbb{F}_{p^m}$ and an arbitrary element A as follows;

$$\begin{aligned} A &= a_0\omega + a_1\omega^p + \cdots + a_{m-1}\omega^{p^{m-1}} \\ &= (a_0, a_1, \cdots, a_{m-1}). \end{aligned} \tag{4}$$

Then, Frobenius mapping is given as

$$\begin{aligned} A &\rightarrow A^p : \\ A^p &= a_0\omega^p + a_1\omega^{p^2} + \cdots + a_{m-2}\omega^{p^{m-1}} + a_{m-1}\omega \\ &= (a_{m-1}, a_0, \cdots, a_{m-2}). \end{aligned} \tag{5}$$

[1]In this paper, we call an element that belongs to $\mathbb{F}_{p^m}$ but not to its proper subfield *a proper element* in $\mathbb{F}_{p^m}$.

Thus, using a normal basis, we can carry out a Frobenius mapping without any arithmetic operations. In what follows, we use the notation $\varphi_i(A) = A^{p^i}$. By the way, even if the basis is not a normal basis, as optimal extension field (OEF) [9], there exist some bases that fast carries out Frobenius mappings; however, if not, a Frobenius mapping in $\mathbb{F}_{p^m}$ needs about m^2 $\mathbb{F}_p$–multiplications. It is almost equivalent to one multiplication in $\mathbb{F}_{p^m}$. For instance, a Frobenius mapping can be carried out by multiplying a certain $(m \times m)$ matrix.

2.2 Tate pairing

An elliptic curve over $\mathbb{F}_p$ is generally defined by

$$E/\mathbb{F}_p : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p. \qquad (6)$$

On the elliptic curve, $\mathbb{F}_p$-rational points form an additive Abelian group. In this paper, we denote this group and its order by $E(\mathbb{F}_p)$ and $\#E(\mathbb{F}_p)$, respectively.

2.2.1 Embedding degree of pairing:

Let r and $G[r]$ be a prime number such that $r \mid \#E(F_p)$ and a subgroup of the order r in $E(\mathbb{F}_p)$, respectively. It is said that the subgroup $G[r]$ has the embedding degree k if r divides $p^k - 1$ but does not divide $p^i - 1$, $1 \le i < k$. Then, the subgroup $E[r] \cong G[r] \times G[r]$ of r-torsion points lies in the elliptic curve $E(\mathbb{F}_{p^k})$ defined over $\mathbb{F}_{p^k}$ and r^2 divides $\#E(\mathbb{F}_{p^k})$ [10].

2.2.2 Tate pairing :

Let G_1 be the subgroup of the order r in $E(\mathbb{F}_p)$ and G_2 be a subgroup of order r in $E(\mathbb{F}_{p^k})$ such that $G_2 \ne G_1$. For rational points $P \in G_1$ and $Q \in G_2$, Tate pairing is given as the following bilinear map t:

$$\begin{aligned} t : G_1 \times G_2 &\longrightarrow G_T, \\ (P, Q) &\longrightarrow t(P, Q) = \left(f_P(D_Q)\right)^{(p^k-1)/r} = e. \end{aligned} \qquad (7)$$

e is an r-th primitive root of unity in $\mathbb{F}_{p^k}$ and $G_T = \{e \in \mathbb{F}_{p^k} : e^r = 1\}$. Tate pairing needs an exponentiation over a certain extension field $\mathbb{F}_{p^k}$. As shown in Eq.(7), Tate pairing t satisfies *bilinearity* and *non-degeneracy.* Such an exponentiation is one of targets of this paper.

2.3 Binary method

The well-known binary method fast calculates an exponentiation with a large integer exponent. Let n be a positive integer, it calculates a^n as follows;

Algorithm 1 (binary method)

```
1. $X \leftarrow 1$, $A \leftarrow a$.
2. If $n = 0$, output $X$.
3. Otherwise,
4.     if $(n \,\&\, 1) = 1$, $X \leftarrow X \cdot A$.
5.     $A \leftarrow A \cdot A$
6.     $n \leftarrow n \gg 1$, then go to Step.2.
```

(End of algorithm)

In what follows, & and $\gg$ in algorithms denote *bit-and* and *bit–shift* operators, respectively. As shown above, the binary method needs only $\lfloor \log_2 n \rfloor$ squares and $\lfloor \log_2 n \rfloor / 2$ multiplications on average.

2.4 Sliding window method

For instance, let us calculate a^n by the sliding window method whose window size is 3. First, let us prepare the following component;

$$a^2, a^3, a^4, \cdots, a^7, \qquad (8)$$

these exponents correspond to the following binary representations, respectively;

$$2 = (010)_2,\ 3 = (011)_2,\ 4 = (100)_2,\ \cdots, 7 = (111)_2. \qquad (9)$$

Then, we calculate a^n by combining and squaring these previously calculated components. For example, when the exponent n is 318, a^n is calculated as

$$\begin{aligned} a^{318} &= a^{(100111110)_2} \\ &= \left\{ \left(a^{(100)_2}\right)^{2^3} \left(a^{(111)_2}\right) \right\}^{2^3} a^{(110)_2}. \end{aligned} \qquad (10)$$

Thus, the *window size* w of the sliding window method means the bit length of each separation.

The sliding window method is efficient for repeatedly calculating exponentiations with the same base a. In addition, it is noted that the sliding window method still needs $\lfloor \log_2 n \rfloor - w$ squares as shown in Eq.(10). When the base a is often changed, the preparation Eq.(8) does not efficiently work. In other words, the preparation is needed every time.

2.5 Previous works

Let us briefly go over Avanzi's exponentiation method [8]. Note that as introduced in [8] it requires very fast Frobenius mappings.

Write the exponent n as

$$n = \sum_{i=0}^{s} n_i p^i,\ 0 \le n_i \le p-1,\ s = \lfloor \log_p n \rfloor, \quad (11)$$

and for some $w \ll u = \lfloor \log_2 p \rfloor,\ K = \lfloor u/w \rfloor$,

$$n_i = \sum_{j=0}^{K-1} n_{ij} 2^{jw}. \quad (12)$$

Then, calculate A^n in $\mathbb{F}_{p^m}$ as follows;

$$\begin{aligned} A^n &= \prod_{i=0}^{s} \varphi_i(A^{n_i}) = \prod_{i=0}^{s} \prod_{j=0}^{K-1} \varphi_i(A^{n_{ij} 2^{jw}}) \\ &= \prod_{j=0}^{K-1} \left(\prod_{i=0}^{s} \varphi_i(A^{n_{ij}}) \right)^{2^{jw}}. \end{aligned} \quad (13)$$

For calculating $A^{n_{ij}}$, it efficiently uses the sliding window method. Compared to just applying the sliding method in addition to Frobenius mappings, Eq.(13) decreases the number of multiplications but increases Frobenius mappings. This algorithm needs $u+1$ squaring, $(s+1)K + 2^w - 2$ multiplications and sK Frobenius mappings.

3 Exponentiation in extension field

This paper proposes a new exponentiation method in extension field using Frobenius mapping with respect to $\mathbb{F}_p$. In what follows, let us consider an exponentiation $A^n, A \in \mathbb{F}_{p^m}$ and let m be larger than or equal to 2.

3.1 Main idea

Consider the p–adic representation of the exponent n as Eq.(11). Then, using Frobenius mapping φ, the exponentiation A^n is calculated by

$$A^n = \prod_{i=0}^{s} \varphi_i\left(A^{n_i}\right). \quad (14)$$

Let these components A^{n_i} be respectively denoted by $A[i]$ in **Alg**.2. Then, an exponentiation A^n is calculated by the binary method with Frobenius mappings as follows;

Algorithm 2

```
 1. $B \leftarrow A,\ s \leftarrow \lfloor \log_p n \rfloor$,
    $t \leftarrow \lfloor \log_2(p-1) \rfloor$.
 2. For $0 \le i \le s$, $A[i] = 1$.
 3. If $n = 0$, output 1.
 4. Otherwise,
 5.     calculate the $p$–adic representation of $n$ as Eq.(11),
 6.     for $0 \le j < t$,
 7.         for $0 \le i \le s$,
 8.             if $(n_i \ \&\ 1) = 1$,
                $A[i] \leftarrow A[i] \cdot B$,
 9.         $B \leftarrow B \cdot B$, $n_1 \leftarrow n_1 \gg 1$.
10.     $C = A[s]$.
11.     for $s-1 \ge j \ge 0$,
12.         $C \leftarrow \varphi_1(C)$, $C \leftarrow C \cdot A[j]$.
13.     output $C$.
```

(End of algorithm)

This algorithm needs t squares and on average $st/2$ multiplications in $\mathbb{F}_{p^m}$, after that, Eq.(14) needs $(s-1)$ Frobenius mappings and $(s-1)$ multiplications. In order to decrease the number of multiplications with respect to [8], it is efficient to additionally apply the sliding window method for the calculation of A^{n_i}. Since the proposed method described below does not use the sliding window method, it has some advantages different from Avanzi's method.

Let us improve **Alg**.2. For instance, let $s = \lfloor \log_p n \rfloor$ and $t = \lfloor \log_2(p-1) \rfloor$ be 5 and 4, respectively. Suppose the binary representations of n_0, $n_1, \cdots, n_5$ as

$$\begin{aligned} n_1 &= (1001)_2,\ n_0 = (1110)_2, && \text{(15a)}\\ n_3 &= (1101)_2,\ n_2 = (1110)_2, && \text{(15b)}\\ n_5 &= (1111)_2,\ n_4 = (0101)_2. && \text{(15c)} \end{aligned}$$

Let us separate the exponent n into three parts as follows (*see* **Fig**.1);

$$n = (n_5 p + n_4)p^4 + (n_3 p + n_2)p^2 + (n_1 p + n_0). \quad (16)$$

Then, consider two sets $G_1 = \{A^{n_5}, A^{n_3}, A^{n_1}\}$ and $G_2 = \{A^{n_4}, A^{n_2}, A^{n_0}\}$.

$$\begin{aligned} A^{n_1} &= A^8A^1, & A^{n_0} &= A^8A^4A^2, && \text{(17a)}\\ A^{n_3} &= A^8A^4A^1, & A^{n_2} &= A^8A^4A^2, && \text{(17b)}\\ A^{n_5} &= A^8A^4A^2A^1, & A^{n_4} &= A^4A^1. && \text{(17c)} \end{aligned}$$

As shown in Eqs.(15) and Eqs.(17), for example, the component A^2 is needed for A^{n_5} in G_1 and A^{n_0}, A^{n_2} in G_2. The component A^4 is needed for A^{n_3}, A^{n_5} in G_1 and A^{n_0}, A^{n_2}, A^{n_4} in G_2. Let us calculate C_{001}, $C_{010}, \cdots, C_{111}$ as follows;

$$\begin{aligned} &C_{100} = \varphi_1(A^2)A^1,\ C_{010} = 1,\ C_{001} = 1, && \text{(18a)}\\ &C_{011} = A^8A^2,\ C_{101} = 1,\ C_{110} = \varphi_1(A^4), && \text{(18b)}\\ &C_{111} = \varphi_1(A^8A^1)A^4. && \text{(18c)} \end{aligned}$$

Then, we can calculate A^n as follows;

$$\begin{aligned} R_0 &= C_{100}C_{101}C_{110}C_{111}, && \text{(19a)}\\ R_1 &= C_{010}C_{011}C_{110}C_{111}, && \text{(19b)}\\ R_2 &= C_{001}C_{011}C_{101}C_{111}, && \text{(19c)}\\ A^n &= \varphi_4(R_2)\,\varphi_2(R_1)\,R_0. && \text{(19d)} \end{aligned}$$

Eq.(14) with Eqs.(17) needs 3 squares and 16 multiplications without using the sliding window method; however, the improved Eqs.(18) and Eqs.(19) need 3 squares and only 14 multiplications. Thus, the p–adic representation and Frobenius mappings shown in Eq.(14) help for decreasing the number of squares and the calculation technique introduced with Eqs.(18) helps for decreasing that of multiplications. **Fig**.1 shows an image of the calculations of Eqs.(15), Eq.(16), Eqs.(17), and Eqs.(18).

3.2 Proposed algorithm

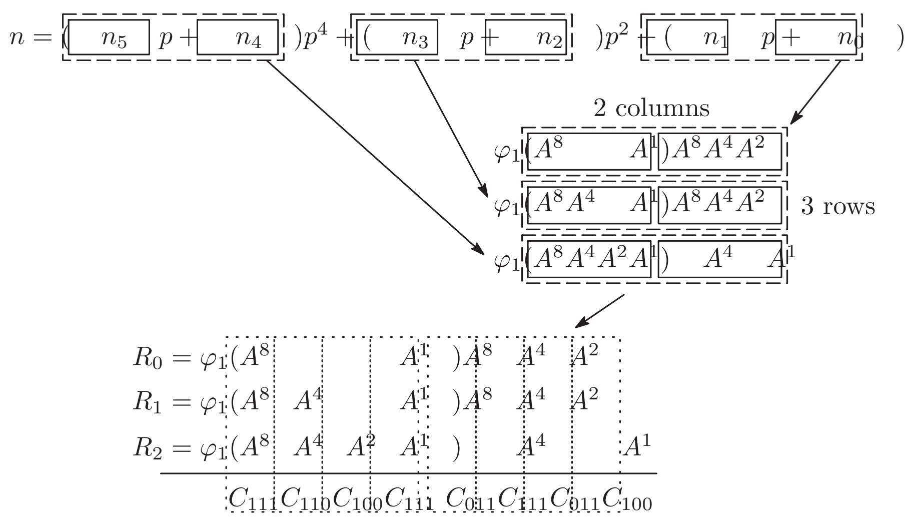
As shown in **Fig**.1, let the numbers of rows and columns be r and c, respectively. In this case, consider the exponent n as

$$n = \sum_{i=0}^{r-1}\sum_{j=0}^{c-1} n_{ij}p^{ci+j}. \quad (20)$$

According to the bit size n and the number r of rows, the number c of columns is automatically determined. Then, the proposed method calculates the exponentiation A^n as follows;

$$\begin{aligned} S_{jl} &= \left\{ x \mid \sum_{i=0}^{r-1} 2^i(n_{ij}\ \&\ 2^x) = l,\ 0 \le x < t \right\},\\ &\quad 0 \le j < c,\ 1 \le l < 2^r. && \text{(21a)}\\ T_i &= \left\{ y \mid (2^i\ \&\ y) = 2^i,\ 1 \le y < 2^r - 1 \right\},\\ &\quad 0 \le i < r. && \text{(21b)}\\ C_l &= \prod_{j=0}^{c-1} \varphi_j \left(\prod_{k \in S_{jl}} A^{2^k} \right). && \text{(21c)}\\ R_i &= \prod_{j=0}^{c-1} \varphi_j\left(A^{n_{ij}}\right) = \prod_{k \in T_i} C_k,\ 0 \le i < r. && \text{(21d)}\\ A^n &= \prod_{i=0}^{r-1} \varphi_{ci}(R_i). && \text{(21e)} \end{aligned}$$

It is found that $|S_{jl}| \le t$ and $|T_i| < 2^r - 1$. In the proposed method, the number of the temporary variables C_l and R_i is $(2^r - 1) + r$ as shown in Eqs.(21). The preparation of C_l, $1 \le l < 2^r$ needs multiplications less than or equal to $c \cdot t$ times, where $t = \lfloor \log_2(p-1) \rfloor$. Using this temporary data, A^n is calculated with several multiplications less than $r(2^r - 1) + (r - 1)$ times as shown in Eq.(21d) and Eq.(21e). In addition, the proposed algorithm needs $(c-1)(2^r - 1) + (r - 1)$ Frobenius mappings. In the Algorithm3, if we finish at line 20, we can calculate r exponentiations at the same time.



$n = (n_5\ p + n_4)p^4 + (n_3\ p + n_2)p^2 + (n_1\ p + n_0)$
2 columns
$\varphi_1(A^8\ A^1)A^8A^4A^2$
$\varphi_1(A^8A^4\ A^1)A^8A^4A^2$
$\varphi_1(A^8A^4A^2A^1)\ A^4\ A^1$
3 rows
$R_0 = \varphi_1(A^8\ A^1)A^8\ A^4\ A^2$
$R_1 = \varphi_1(A^8\ A^4\ A^1)A^8\ A^4\ A^2$
$R_2 = \varphi_1(A^8\ A^4\ A^2\ A^1)\ A^4\ A^1$
$C_{111}\ C_{110}\ C_{100}\ C_{111}\ C_{011}\ C_{111}\ C_{011}\ C_{100}$

Figure 1: Image of the calculations of Eqs.(15), Eq.(16), Eqs.(17), and Eqs.(18)

Algorithm 3 (the proposed algorithm)

```
1. $B[0] \leftarrow A,\ t \leftarrow \lfloor \log_2(p-1) \rfloor$.
2. For $1 \le i < t$, $B[i] \leftarrow B[i-1] \cdot B[i-1]$.
3. For $0 < i \le 2^r$, $C[i] = 1$.
4. For $0 \le i < r$, $R[i] = 1$.
5. If $n = 0$, output 1.
6. Otherwise,
7.     calculate the $p$–adic representation of $n$ as Eq.(20),
8.     for $c > j \ge 0$,
9.         for $0 \le k < t$,
10.            $M = 0$.
11.            for $0 \le i < r$,
12.                if $(n_{ij}\ \&\ 1) = 1$,
                   $M \leftarrow M + 2^i$.
13.                $n_{ij} \leftarrow n_{ij} \gg 1$.
14.            if $M \ne 0$,
               $C[M] \leftarrow C[M] \cdot B[k]$.
15.        if $j \ne 0$,
16.            for $1 \le i \le 2^r$,
17.                $C[i] \leftarrow \varphi_1(C[i])$.
18.    for $0 \le i < r$,
19.        for $1 \le j \le 2^r$,
20.            if $(2^i\ \&\ j) \ne 0$,
               $R[i] \leftarrow R[i] \cdot C[j]$.
21.    $D \leftarrow R[r-1]$.
22.    for $r-2 \ge i \ge 0$,
23.        $D \leftarrow \varphi_c(D)$, $D \leftarrow D \cdot R[i]$.
24.    output $D$.
```

(End of algorithm)

4 Application and simulation

As introduced in the preceding sections, one of targets of the proposed exponentiation method is the case that the base of exponentiation is often changed and a Frobenius mapping cannot be fast carried out in the concerned extension field. In this simulation, we consider an exponentiation in such a case.

4.1 Application

As an application of the proposed method, this section considers Tate pairing over elliptic curve. As introduced in **Sec**.2.2, a pairing needs an exponentiation in extension field. Barreto et al. [11] have proposed a class of *non super–singular pairing friendly* (NSPF) elliptic curves defined as follows;

$$E/\mathbb{F}_p : y^2 = x^3 + b,\ b \in \mathbb{F}_p, \tag{22}$$

for which, using a certain integer χ, the characteristic p and its order $\#E(\mathbb{F}_p)$ need to be given as follows;

$$\begin{aligned} p &= 36\chi^4 + 36\chi^3 + 24\chi^2 + 6\chi + 1, && \text{(23a)} \\ \#E(\mathbb{F}_p) &= 36\chi^4 + 36\chi^3 + 18\chi^2 + 6\chi + 1. && \text{(23b)} \end{aligned}$$

The embedding degree k is equal to 12. In other words, let $r = \#E(\mathbb{F}_p)$ be a prime number, we have

$$r \mid \#E(\mathbb{F}_{p^k}) \quad \text{and} \quad r \mid (p^k - 1). \tag{24}$$

Noting that the embedding degree k is the least positive integer such that r divides $(p^k - 1)$, the exponentiation of the pairing calculation Eq.(7) becomes

$$\begin{aligned} A^{(p^{12}-1)/r} &= \left\{A^{(p^6-1)(p^2+1)}\right\}^{(p^4-p^2+1)/r} \\ &= \left\{B^{(p^6-1)}\right\}^{(p^4-p^2+1)/r} \\ &= C^{(p^4-p^2+1)/r}, \end{aligned} \tag{25}$$

where $A = f_P(D_Q)$, $B = A^{p^2+1} = \varphi_2(A)A$, and $C = B^{p^6}B^{-1} = \varphi_6(B)B^{-1}$. Therefore, the major computation is the exponentiation $C^{(p^4-p^2+1)/r}, C \in \mathbb{F}^*_{p^{12}}$. Noting that $\log_2 r = \log_2 p$, it is easily found that

$$\log_p\bigl((p^4 - p^2 + 1)/r\bigr) = 3. \tag{26}$$

In this case, the proposed method Eqs.(21) carries out this exponentiation with $n = 160 \times 3$, $r = 3$, $c = 1$, and $t = 160$. This exponentiation needs 160 squares, 142 multiplications on average, and 2 Frobenius mappings. The proposed algorithm is also efficient for a scalar multiplication of rational point of elliptic curve defined over extension field that uses Frobenius mappings such as [12].

4.2 Simulation result

Table 1 shows the comparison of the calculation cost for an exponentiation in $\mathbb{F}_{p^m}$ between the proposed method and Avanzi's method [8]. The authors considered a 160–bit prime number as the characteristic p.

As shown in the table, the number of $\mathbb{F}_p$–multiplications needed for the proposed method Eqs.(21) is about 20% larger than that of Avanzi's method; however, the number of Frobenius mappings needed for the proposed method is quite small. As introduced in **Sec**.2.1, if a Frobenius mapping cannot be fast carried out in the concerned extension field $\mathbb{F}_{p^m}$, it is almost equivalent to a multiplication in $\mathbb{F}_{p^m}$. In such a case, the proposed method will efficiently work than the other previous works such as Avanzi's method [8].

5 Conclusion

In this paper, the authors have proposed an exponentiation method in extension field that efficiently uses Frobenius mappings but was not based on the sliding window method. Compared to Avanzi's method [8], the number of multiplications needed for the proposed method becomes about 20% larger; however, that of Frobenius mappings is sufficiently small. Evaluating the calculation costs, it was shown that the proposed exponentiation method was practical for the case that the base of exponentiation was often changed and a Frobenius mapping could not be fast carried out in the concerned extension field.

References

[1] D.Boneh, B.Lynn, and H.Shacham, "Short signatures from the Weil pairing," Proc. of Asiacrypt2001, LNCS 2248, pp.514-532, 2001.

[2] T.Nakanishi and N.Funabiki, "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps," Asiacrypt2005, LNCS 3788, Springer-Verlag, pp.443-454, 2005.

[3] D.Boneh, C.Gentry, and B.Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," In Advances in Cryptology (CRYPTO2005), LNCS 3621, Springer-Verlag, pp.258-275, 2005.

[4] J. Silverman, The arithmetic of elliptic curve, Springer-Verlag, 1986.

[5] H.Cohen and G.Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, *Discrete Mathematics and Its Applications*, Chapman & Hall CRC, pp.280-285, p.458, 2005.

Table 1: Comparison of the calculation cost for an exponentiation in F_{p^m}

degree m	3	4	5	6	7
proposal Eqs.(21)	$r=3$ (159,150,2)	$r=4$ (159,180,3)	$r=5$ (159,234,4)	$r=3$ (159,290,9)	$r=4$ (159,320,18)
Avanzi Eq.(13)	$w=5$ (160,125,61)	$w=5$ (160,156,92)	$w=5$ (160,187,123)	$w=5$ (160,218,154)	$w=6$ (156,249,159)

degree m	8	9	10	11	12
proposal Eqs.(21)	$r=4$ (159,330,18)	$r=5$ (159,383,35)	$r=5$ (159,388,35)	$r=4$ (159,470,33)	$r=4$ (159,480,33)
Avanzi Eq.(13)	$w=6$ (156,275,185)	$w=6$ (156,302,212)	$w=6$ (156,328,238)	$w=6$ (156,354,264)	$w=6$ (156,381,291)

Remark : ($\#S$,$\#M$,$\#F$) means $\#S$ squares, $\#M$ multiplications and $\#F$ Frobenius mappings in $\mathbb{F}_{p^m}$, respectively. r and w are the *row size* in the proposed algorithm and the *window size* in Avanzi's algorithm Eq.(13), respectively. The characteristic p is a 160–bit prime number.

[6] F.Hess, N.Smart, and F.Vercauteren, "The Eta Pairing Revisited," IEEE Transactions on Information Theory, Vol.52, No.10, pp.4595-4602, 2006.

[7] Bodo Moller, "Algorithms for Multi-exponentiation," Proc. of SAC2001, LNCS 2259, Springer-Verlag, LNCS pp.165-180, 2001.

[8] R.Avanzi, and P.Mihailescu, "Generic Efficient Arithmetic Algorithms for PAFFs (Processor Adequate Finite Fields) and Related Algebraic Structures," Proc. of SAC2003, LNCS 3006, Springer-Verlag, LNCS pp.320-334, 2003.

[9] D.Bailey and C.Paar, "Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms," Proc. Asiacrypt2000, LNCS 1976, pp.248-258, 2000.

[10] R.Balasubramanian, and N.Koblitz, "The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm," Journal of Cryptology, 11(2):141-145, 1998.

[11] P.Barreto, H.Kim, B.Lynn, and M.Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," CRYPTO 2002, LNCS 2442, pp.354-368, 2002.

[12] T.Kobayashi, H.Morita, K.Kobayashi, and F.Hoshino, "Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic," EUROCRYPT '99, LNCS 1592, p.176, 1999.