Efficient Computation of Eta Pairing over Binary Field with Vandermonde Matrix

Masaaki Shirase, Tsuyoshi Takagi, Dooho Choi, DongGuk Han, and Howon Kim

This paper provides an efficient algorithm for computing the η_T pairing on supersingular elliptic curves over fields of characteristic two. In the proposed algorithm, we deploy a modified multiplication in $F_{2^{4n}}$ using the Vandermonde matrix. For $F, G \in F_{2^{4n}}$, the proposed multiplication method computes $\beta \cdot F \cdot G$ instead of $F \cdot G$ with some $\beta \in F_{2^n}^*$ because β is eliminated by the final exponentiation of the η_T pairing computation. The proposed multiplication method asymptotically requires only 7 multiplications in F_{2^n} as $n \to \infty$, while the cost of the previously fastest Karatsuba method is 9 multiplications in F_{2^n}. Consequently, the cost of the η_T pairing computation is reduced by 14.3%.

Keywords: Cryptography, η_T pairing, finite field, loop unrolling, Vandermonde matrix.

Manuscript received May 31, 2008; revised Dec 17, 2008; accepted Dec. 24, 2008.

This work was supported by the IT R&D program of MKE/IITA [2009-F-055-01, Development of the Technology of Side Channel Attack Countermeasure Primitives and Security Validation], Rep. of Korea.

Masaaki Shirase (phone: +81 138 34 6476, email: shirase@fun.ac.jp) and Tsuyoshi Takagi (email: takagi@fun.ac.jp) are with School of Systems Information Science, Future University Hakodate, Hokkaido, Japan.

Dooho Choi (email: dhchoi@etri.re.kr) is with Software & Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

DongGuk Han (email: christa@kookmin.ac.kr) was with Software & Content Research Laboratory, ETRI, Daejeon, Rep. of Korea., and is currently with the Department of Mathematics, Kookmin University, Seoul, Rep. of Korea.

Howon Kim (email: howonkim@pusan.ac.kr) is with the Department of Computer Engineering Information, Pusan National University, Busan, Rep. of Korea.

I. Introduction

Pairings have been attractive for cryptography, and the η_T pairing proposed by Barreto and others [1] is one of the fastest pairings. The η_T pairing is defined on supersingular elliptic curves over finite fields of characteristic two or three. In this paper, we particularly focus on the η_T pairing on supersingular elliptic curves in characteristic two because it is more suitable for implementation on computers. The speed for the η_T pairing in characteristic two bottlenecks at the multiplications in the 4th extension field $F_{2^{4n}}$. For η_T pairing in characteristic three, there are two methods to improve the speed of the extension field: the loop unrolling technique [2] and an efficient multiplication by discrete Fourier transformation (DFT) [3]. This paper extends these methods for extension field $F_{2^{4n}}$.

A loop unrolling technique tries to reduce the number of multiplications in a finite field by merging two or more iterations of a loop into one. In the case of characteristic three, the loop unrolling technique makes the η_T pairing fast [2]. On the other hand, if we directly apply it to the case of characteristic two, it becomes no more efficient than it would be without loop unrolling. However, the loop unrolling technique provides us with an efficient pairing algorithm if we modify an efficient multiplication method in $F_{2^{4n}}$.

The Karatsuba method [4] has been used to efficiently compute multiplications in extension fields. Gorla and others proposed an efficient multiplication method in $F_{3^{6n}}$ using the DFT matrix [3]. We call it the GPS multiplication in this paper, in accordance with the authors' names. However, if we apply this multiplication directly to a case of characteristic two, the cost of multiplication in $F_{2^{4n}}$ with this method is the same as

 © 2009 ETRI

that with the conventional Karatsuba method.

This paper proposes a faster algorithm for computing the η_T pairing over F_{2^n} using loop unrolling and the improved extension field multiplication in $F_{2^{4n}}$ with the Vandermonde matrix. We may compute $\beta \cdot F \cdot G$ instead of $F \cdot G$ for any $F, G \in F_{2^{4n}}$ and $\beta \in F_{2^n}^*$ in an algorithm for computing the η_T pairing because β powers to 1 in the final exponentiation, and has no effect on the pairing value. Therefore, the improved multiplication method computes the multiplication $\beta \cdot F \cdot G$. The cost of the improved multiplication method is *asymptotically* 7 multiplications in F_{2^n} as $n \to \infty$. Note that a multiplication $F \cdot G$ requires 9 multiplications in F_{2^n} with the Karatsuba method. Combining the loop unrolling technique and the proposed multiplication method, we can achieve an efficient algorithm for computing the η_T pairing.

The remainder of this paper is organized as follows. In section II, we provide a loop unrolling algorithm for computing the η_T pairing over F_{2^n}. In section III, we describe polynomial multiplication. In section IV, we propose a multiplication method in $F_{2^{4n}}$ for the η_T pairing and estimate the cost. In section V, we apply the proposed method to an algorithm for computing the η_T pairing. Last, in section VI, we conclude this paper.

II. η_T Pairing in Characteristic Two

A pairing e used in cryptography such as the η_T pairing, the Tate pairing, and the Ate pairing [5] are forms of bilinear mapping,

$$e : G_1 \times G_2 \to F_{p^{kn}},$$

where G_1 is a group based on $E(F_{p^n})$, G_2 is a group based on $E(F_{p^n})$ or $E(F_{p^{kn}})$, E is an elliptic curve defined over F_{p^n}, and k denotes the embedding degree. Table 1 shows the relationships among pairings, finite fields, and embedding degrees.

In this paper, we deal with the η_T pairing defined on supersingular elliptic curve

$$E_b : Y^2 + Y = X^3 + X + b, \ b \in \{0,1\},$$

Table 1. Embedding degree k for pairings.

Pairing	η_T or Tate		Tate or Ate
Field	F_{3^n}	F_{3^n}	F_{p^n}
k	4	6	2, 6, 12, or more

over F_{2^n}, where n is a large enough prime for security of the η_T pairing. Then, n is an odd integer. The η_T pairing is a bilinear mapping

$$\eta_T : E^b(F_{2^n}) \times E^b(F_{2^n}) \to F_{2^{4n}}^*,$$

and is defined as $\eta_T(P,Q) = f_{2^n,P}(\psi(Q))^W$ for $P, Q \in E^b(F_{2^n})$, where $f_{2^n,P}$ is a function such that its divisor satisfies

$$\mathrm{div}(f_{2^n,p}) = 2^n(p) - ([2^n]p) - (2^n - 1)\mathrm{div}(\infty).$$

Here, ψ is the distortion map defined as $\psi : (x, y) \to (x + s^2, y + sx + t)$, where $s^2 = s+1$ and $t^2 = t+s$, and W is an integer defined as $W = (2^{2n} - 1)(2^n + 1 - \varepsilon 2^{(n+1)/2})$, where

$$\varepsilon = \begin{cases} -1, & \text{if } n \equiv 1, 7 \pmod 8 \text{ and } b = 1 \\ & \text{or } n \equiv 3, 5 \pmod 8 \text{ and } b = 0, \\ 1, & \text{otherwise.} \end{cases}$$

Algorithm 1. Original algorithm for η_T pairing [6].

Input: $P = (\alpha, \beta), Q = (\gamma, \beta) \in E^b(F_{2^n})$

Output: $\eta_T(P,Q) \in F_{2^{4n}}^*$

1: $w \leftarrow \alpha + \left[\frac{n-1}{2}\right]$

2: $F \leftarrow w \cdot (\gamma + \alpha + 1) + \delta + (\beta + b + \varepsilon_{(n+1)/2}) + (w + \gamma)s + t$
(cost: M)

3: for i=0 to (n-1)/2 do

4: $w \leftarrow \alpha + \left[\frac{n+1}{2}\right]$

5: $F \leftarrow w \cdot \left(\gamma + \left[\frac{n+1}{2}\right]\right) + \delta + (\beta + b + \varepsilon_{(n-1)/2}) + (w + \gamma)s + t$
(G is st-sparse, cost: M)

6: $G \leftarrow F \rtimes G$ (cost: 6M)

7: if i<(n-1)/2 then

8: $\alpha \leftarrow \sqrt{\alpha}, \beta \leftarrow \sqrt{\beta}, \gamma \leftarrow \gamma^2, \delta \leftarrow \delta^2$ (cost: $2R + 2S$)

9: end if

10: end for

11: return F^W (cost: F_E)

Algorithm 1 is efficient for computing the η_T pairing [6]. In this algorithm, elements in $F_{2^{4n}}$ are represented with a basis $\{1, s, t, st\}$. This basis is called the st-basis, and we denote by $(a_0, a_1, a_2, a_3)_{st}$ the element $a_0 + a_1 s + a_2 t + a_3 st$ in $F_{2^{4n}}$ for $a_0, a_1, a_2, a_3 \in F_{2^n}$. We also use another basis of $F_{2^{4n}}$, namely, the z-basis $\{1, z, z^2, z^3\}$ defined by the irreducible polynomial $h(z) = z^4 + z + 1$ over F_{2^n}. We denote by $(b_0, b_1, b_2, b_3)_z$ the element $b_0 + b_1 z + b_2 z^2 + b_3 \in F_{2^{4n}}$. We can transform between

the *st*-basis and the *z*-basis virtually for free. Indeed, we have the following conversion algorithm for $(a_0, a_1, a_2, a_3)_{st}=(b_0, b_1, b_2, b_3)_z$:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}, \quad \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}. \tag{1}$$

The notations used in algorithm 1 are defined as follows:

$$\varepsilon_m = \begin{cases} 0, & \text{if } m \equiv 0,1 \pmod 4 \\ 1, & \text{otherwise,} \end{cases}$$
$$[m] = m \mod 2,$$

where n is the extension degree of F_{2^n} over F_2, and m is an integer. Powering by W at step 11 in algorithm 1 is called the final exponentiation, which can be efficiently computed [7].

We define symbols M, S, R, and F as follows:

M= the cost of a multiplication in F_{2^n},
S= the cost of a squaring in F_{2^n},
R= the cost of a square root in F_{2^n},
F_E= the cost of the final exponentiation.

Note that G at step 5 is the form of $(g_0, g_1, 1, 0)_{st}$ for $g_0, g_1 \in F_{2^n}$, where $g_0=w\cdot(\gamma+[(n+1)/2])+\delta+(\beta+b+\varepsilon_{(n-1)/2})$ and $g_1=w+\gamma$. Such an element is called *st*-sparse in this paper. Note that the *st*-sparse element $(g_0, g_1, 1, 0)_{st}$ is converted to $(g_0, g_1+1, g_1, 0)_z$. We then call this element *z*-sparse. Note that there is no difference between the cost of multiplications using the *st*-basis and the cost of multiplications using the *z*-basis due to (1). Now we distinguish among three multiplications in $F_{2^{4n}}$:

(a) random element by random element: $\times$
(b) random element by (st, z)-sparse element: $\rtimes$
(c) (st, z)-sparse element by (st, z)-sparse element: $\bowtie$

where the random element is the form of $(g_0, g_1, g_2, g_3)_{st}$ (or $(g_0, g_1, g_2, g_3)_{st}$) for $g_0, g_1, g_2, g_3 \in F_{2^n}$. We use "$\cdot$" for multiplication of elements in F_{2^n} and polynomials with coefficients in F_{2^n}.

The cost of each multiplication is shown in Table 2. Multiplication (a) is computed by the Karatsuba method, and the computation of (b) is proposed in [1]. We show that (c) can be computed with the cost of $3M$ in the appendix.

Table 2. Multiplication symbols in $F_{2^{4n}}$ and cost.

	a	b	cost
(a) $a\times b$	Random	Random	$9M$
(b) $a\rtimes b$	Random	(st, z)-sparse	6M ([1])
(c) $a\bowtie b$	(st, z)-sparse	(st, z)-sparse	$3M$

Now, we estimate the cost (the number of multiplications) of algorithm 1. The cost of each step is written at the end of each step. For example, step 2 takes M of "$w\cdot(\gamma+a+1)$." Steps 1 and 2 are the initialization, with a cost of M. Steps 3 to 8 are the main loop. Note that the number of iterations of the main loop, except from steps 3 to 6, is $(n+1)/2$, and the number of iterations of steps 3 to 6 is $(n-1)/2$, where n is an odd integer. Then, the cost of the main loop is $(n+1)/2\cdot 7M+(n-1)/2\cdot(2R+2S)$. Step 11 is the final exponentiation, with a cost of F_E. Therefore, the total cost of algorithm 1 is $(3.5n+4.5)M+(n-1)S+(n-1)R+F_E$.

1. Loop Unrolling Technique

We might be able to enhance the speed of the η_T pairing by unrolling the loop. Indeed, the unrolling technique is effective in the case of characteristic three [2], [8].

Algorithm 2 is the loop unrolling algorithm for computing the η_T pairing over F_{2^n} based on algorithm 1. Algorithm 2 merges with algorithm 1 such that "$F\leftarrow F\rtimes G$" at step 6 in the $2i$-th and $(2i+1)$th iterations of algorithm 1 corresponds to "$G\leftarrow G_0\bowtie G_1$" at step 9 and "$F\leftarrow F\times G$" at step 10 in the i-th iteration of algorithm 2.

Note that "$F\leftarrow F\rtimes G$" of algorithm 1 takes $6M$ because F is random and G is sparse. On the other hand, "$G\leftarrow G_0\bowtie G_1$" and "$F\leftarrow F\times G$" of algorithm 2 take $3M$ and $9M$, respectively, because G_0 and G_1 are sparse and F is random. The number of iterations of algorithm 1 is twice that of algorithm 2. The cost of algorithm 2 is the same as that of algorithm 1, that is, $(3.5n+4.5)M+(n-1)M+(n-1)S+F_E$, because $6M+6M=3M+9M$.

Also note that if there was a multiplication method for $F\times G$, namely, random element by random element, with a cost less than $9M$, algorithm 2 would be more efficient than algorithm 1. Our goal is to find such a method.

III. Polynomial Multiplication

Multiplication in a finite field consists of polynomial multiplication and reduction modulo an irreducible polynomial.

Algorithm 2. Loop unrolling algorithm for η_T pairing

Input: $P=(\alpha,\beta), Q=(\gamma,\delta)\in E^b(F_{2^n})$

Output: $\eta_T(P,Q)\in F^*_{2^{4n}}$

1: $w\leftarrow \alpha+\left\lceil\frac{n-1}{2}\right\rceil$

2: $F\leftarrow w\cdot(\gamma+\alpha+1)+\delta+(\beta+b+\varepsilon_{(n+1)/2})+(w+\gamma)s+t$
(cost: M)

3: for i=0 to $\lfloor(n-3)/4\rfloor$ do

4: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

5: $G_0\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+(\beta+b+\varepsilon_{(n-1)/2})+(w+\gamma)s+t$
(G_0 is st-sparse, cost: M)

6: $\alpha\leftarrow\sqrt{\alpha},\beta\leftarrow\sqrt{\beta},\gamma\leftarrow\gamma^2,\delta\leftarrow\delta^2$ (cost: $2R+2S$)

7: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

8: $G_1\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+\delta+(\beta+b+\varepsilon_{(n-1)/2})+(w+\gamma)s+t,$
(G_1 is st-sparse, cost: M)

9: $G\leftarrow G_0\bowtie G_1$ (cost: $3M$)

10: $F\leftarrow F\times G$ (cost: $9M$)

11: $\alpha\leftarrow\sqrt{\alpha},\beta\leftarrow\sqrt{\beta},\gamma\leftarrow\gamma^2,\delta\leftarrow\delta^2$ (cost: $2R+2S$)
(11 is unnecessary if $n\equiv 3 \pmod 4$ for $i=\lfloor(n-3)/4\rfloor$)

12: end for

13: if $i\equiv 1 \pmod 4$ then

14: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

15: $G_0\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+\delta+(\beta+b+\varepsilon_{(n-1)/2})+(w+\gamma)s+t,$
(G_0 is st-sparse, cost: M)

16: $F\leftarrow F\rtimes G$ (cost: $6M$)

17: end if

18: return F^W (cost: F_E)

In this section, we focus on polynomial multiplication.

Let F be a finite field. We deal with three methods for multiplying polynomials $a(z)\cdot b(z)$ with coefficients in F. The first method is based on the Vandermonde matrix (VM) method [9]. The second one is based on the discrete Fourier transform (DFT) method [9]. The third one was proposed by Gorla, Puttmann, and Shokrollahi [3], and we call it the GPS multiplication in accordance with the authors' names. The DFT method is an improvement of the VM method, and the GPS multiplication is an improvement of the DFT method.

Let $a(z)$, $b(z)$ be polynomials of degree k–1with coefficients with F. We consider polynomial multiplications of $c(z)=a(z)\cdot b(z)$ and denote them by

$$\begin{aligned} a(z)&=a_0+a_1z+a_2z^2+\cdots+a_{k-1}z^{k-1},\\ b(z)&=b_0+b_1z+b_2z^2+\cdots+b_{k-1}z^{k-1},\\ c(z)&=c_0+c_1z+c_2z^2+\cdots+c_{2k-2}z^{2k-2}. \end{aligned} \tag{2}$$

Note that $c_0=a_0\cdot b_0$ and $c_{2k-2}=a_{k-1}\cdot b_{k-1}$.

1. VM Method

Substituting $z=\alpha$ in (2), we obtain

$$a(\alpha)\cdot b(\alpha)=c(\alpha)=c_0+c_1\alpha+c_2\alpha^2+\cdots+c_{2k-2}\alpha^{2k-2}$$

for any $\alpha\in F$; thus,

$$\underbrace{\begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{2k-2}\\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{2k-2}\\ \vdots & & \ddots & \vdots & \\ 1 & \alpha_{2k-2} & \alpha_{2k-2}^2 & \cdots & \alpha_{2k-2}^{2k-2} \end{pmatrix}}_{A_V}\underbrace{\begin{pmatrix} c_0\\ c_1\\ \vdots\\ c_{2k-2}\end{pmatrix}}_{B_V}=\underbrace{\begin{pmatrix} a(\alpha_0)\cdot b(\alpha_0)\\ a(\alpha_1)\cdot b(\alpha_1)\\ \vdots\\ a(\alpha_{2k-2})\cdot b(\alpha_{2k-2})\end{pmatrix}}_{C_V} \tag{3}$$

holds for $2k$–1 elements $\alpha_0,\alpha_1,\cdots,\alpha_{2k-2}\in F$. The matrix A_V in (3) is called the VM. If this matrix is regular, then we can compute the vector C_V, namely, $c(z)$, by

$$B_V=A_V^{-1}\cdot C_V. \tag{4}$$

2. DFT Method

We assume that F contains the primitive $(2k-1)$th root of unity. Let ξ be this root of unity in F. By setting $\alpha_i=\xi^i$ in (4), we obtain

$$\begin{pmatrix} c_0\\ c_1\\ \vdots\\ c_{2k-2}\end{pmatrix}=\frac{1}{2k-1}\begin{pmatrix} 1 & 1 & \cdots & 1\\ 1 & \xi^{2k-3} & \cdots & \xi\\ \vdots & \vdots & \ddots & \vdots\\ 1 & \xi^2 & \cdots & \xi^{2k-2}\end{pmatrix}\begin{pmatrix} a(1)\cdot b(1)\\ a(\xi)\cdot b(\xi)\\ \vdots\\ a(\xi^{2k-2})\cdot b(\xi^{2k-2})\end{pmatrix}.$$

The computation of $c(z)=a(z)\cdot b(z)$ by the above matrix is called the DFT method.

Here, $c(z)$ is computed with $(2k-1)$ multiplications $(a(1)\cdot b(1), a(\xi)\cdot b(\xi),\cdots,a(\xi^{2k-2})\cdot b(\xi^{2k-2}))$ and one division by $2k$–1 in F under the following assumptions:

(i) The cost of multiplication of ξ^i by each element in F is virtually zero,

(ii) The cost of computations of $a(\xi^i)$ and $b(\xi^j)$ with $0\le i,\ j\le 2k-2$ is virtually zero, and

(iii) $(2k-1)$ is not equal to the characteristic of F.

However, it seems hard to satisfy all three conditions for many finite fields, especially the finite fields for the η_T pairing (see the next sections).

3. GPS Multiplication for $F_{3^{6n}}$ [3]

In the case of characteristic three, the algorithm for computing the η_T pairing includes computations of multiplications in $F_{3^{6n}}$, where n is a prime. Unfortunately, the DFT method cannot be used for multiplications in field $F_{3^{6n}}$ because the 11th (11=2×6-1) primitive root of unity is not contained in F_{3^n}.

The GPS multiplication tries to efficiently compute a multiplication in $F_{3^{6n}}$ by modifying the DFT method in the third extension field $F_{3^{6n}}$ over $F_{3^{2n}}$. Now we consider a polynomial multiplication with coefficients in $F_{3^{2n}}$ of degree 2, that is,

$$\begin{aligned}&(a_0+a_1z+a_2z^2)\cdot(b_0+b_1z+b_2z^2)\\&\quad=c_0+c_1z+c_2z^2+c_3z^3+c_4z^4.\end{aligned}\tag{5}$$

Elements in $F_{3^{2n}}$ are represented as $\{u_0+u_1\sigma : u_0, u_1 \in F_{3^n}\}$, where $\sigma^2=-1$, and σ is the fourth primitive root of unity in $F_{3^{2n}}$. Then we obtain

$$\begin{aligned}&(a_0+a_1\sigma^i+a_2(\sigma^i)^2)\cdot(b_0+b_1\sigma^i+b_2(\sigma^i)^2)-c_4(\sigma^i)^4\\&\quad=c_0+c_1\sigma^i+c_2(\sigma^i)^2+c_3^3(\sigma^i)^3\end{aligned}$$

by setting $z=\sigma^i$ in (5) for i=0, 1, 2, 3. They yield the following matrix equation:

$$\begin{pmatrix}1&1&1&1\\1&\sigma&\sigma^2&\sigma^3\\1&\sigma^2&\sigma^4&\sigma^6\\1&\sigma^3&\sigma^6&\sigma^9\end{pmatrix}\begin{pmatrix}c_0\\c_1\\c_2\\c_3\end{pmatrix}=\begin{pmatrix}a(1)\cdot b(1)-c_4\\a(\sigma)\cdot b(\sigma)-c_4\sigma^4\\a(\sigma^2)\cdot b(\sigma^2)-c_4\sigma^8\\a(\sigma^3)\cdot b(\sigma^3)-c_4\sigma^{12}\end{pmatrix}.\tag{6}$$

Because $\sigma^2=-1$ and c_4=a_2·b_2, (6) can be eventually converted as follows:

$$\begin{pmatrix}c_0\\c_1\\c_2\\c_3\end{pmatrix}=\begin{pmatrix}1&1&1&1\\1&-\sigma&-1&\sigma\\1&-1&1&-1\\1&\sigma&-1&-\sigma\end{pmatrix}=\begin{pmatrix}a(1)\cdot b(1)-a_2\cdot b_2\\a(\sigma)\cdot b(\sigma)-a_2\cdot b_2\\a(\sigma^2)\cdot b(\sigma^2)-a_2\cdot b_2\\a(\sigma^3)\cdot b(\sigma^3)-a_2\cdot b_2\end{pmatrix}.\tag{7}$$

Note that the cost of multiplication of σ by any element in $F_{3^{2n}}$ is virtually zero because $(u_0+u_1\sigma)\cdot\sigma=-u_1+u_0\sigma$, where $u_0+u_1\sigma\in F_{3^{2n}}(u_0,u_1\in F_{3^n})$. Therefore, the cost of computing (7) is 5 multiplications in $F_{3^{2n}}$. A multiplication in $F_{3^{6n}}$ can be computed with 15 multiplications in F_{3^n} by using the Karatsuba method for the multiplication in $F_{3^{2n}}$.

4. Application of DFT Method and GPS Multiplication to $F_{2^{4n}}$

In the algorithm for computing the η_T pairing over F_{2^n}, we need multiplications in $F_{2^{4n}}$. Here, we consider whether we can apply the DFT method and the GPS multiplication to finite field $F_{2^{4n}}$.

There is no 7th (7=2×4-1) primitive root of unity in F_{2^n}, so we cannot apply the DFT method to $F_{2^{4n}}$.

We can apply the GPS multiplication to $F_{2^{4n}}$ because there is a 2nd (2=2×2-2) primitive root of unity in $F_{2^{2n}}$. Then, the cost of a multiplication in $F_{2^{4n}}$ by the GPS multiplication is equal to 3 (=2×2-1) multiplications in $F_{2^{2n}}$, that is, 9M, where M means the cost of a multiplication in F_{2^n} as defined in section II. However, the Karatsuba method also provides a multiplication method with a cost of 9M. Therefore, the GPS multiplication cannot reduce the cost in the case of characteristic two as it can in the case of characteristic three.

IV. Proposed Multiplication for η_T Pairing

In this section, we present the proposed multiplication method, which is an improvement of the VM method over $F_{2^{4n}}$. The main idea is to deploy the computations of $\beta\cdot F\times G\,(\beta\in F_{2^n}^*)$ instead of $F\times G$ in algorithm 2, where $F, G\in F_{2^{4n}}$. Note that $\beta^W=1$ holds, where powering by W is the final exponentiation of the η_T pairing; thus, we can deal with the relaxed multiplication $\beta\cdot F\times G$ in the following subsection.

1. Main Idea

Here we discuss how to modify the multiplication $F\times G$ at step 10 in algorithm 2, where $F, G\in F_{2^{4n}}$. Let us use the z-basis of $F_{2^{4n}}$ over F_{2^n}, which has a definition polynomial of $h(z)$=z^4+z+1. Let F=$f(z)$ and G=$g(z)$, where $f(z)=f_0+f_1z+f_2z^2+f_3z^3$ and $g(z)=g_0+g_1z+g_2z^2+g_3z^3$, for $f_i, g_j\in F_{2^n}$. To evaluate $F\times G$, we try to compute polynomial

$$e(z)=e_0+e_1z+e_2z^2+e_3z^3+e_4z^4+e_5z^5+e_6z^6,\tag{8}$$

where $e(z)$=$f(z)$·$g(z)$. Then $F\times G$ can be obtained by $F\times G$=$e(z)$ mod $h(z)$.

First, we note that e_0 and e_6 can be easily computed, that is, e_0=f_0·g_0 and e_6=f_3·g_3. Next, we explain how to find the other coefficients e_1, e_2, e_3, e_4, and e_5 using the VM method.

Let $\{1, x, \cdots, x^{n-1}\}$ be a polynomial basis of F_{2^n} over F_2. Substituting z=1, x, x+1, x^2, $(x+1)^2$ for $e(z)$=$f(z)$·$g(z)$, we have the following relationship:

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ x & x^2 & x^3 & x^4 & x^5 \\ x+1 & (x+1)^2 & (x+1)^3 & (x+1)^4 & (x+1)^5 \\ x^2 & x^4 & x^6 & x^8 & x^{10} \\ (x+1)^2 & (x+1)^4 & (x+1)^6 & (x+1)^8 & (x+1)^{10} \end{pmatrix}}_{A_P} \underbrace{\begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{pmatrix}}_{B_P}$$

$$= \underbrace{\begin{pmatrix} f(1)\cdot g(1) + f_0 \cdot g_0 + f_3 \cdot g_3 \\ f(x)\cdot g(x) + f_0 \cdot g_0 + f_3 \cdot g_3 \cdot x^6 \\ f(x+1)\cdot g(x+1) + f_0 \cdot g_0 + f_3 \cdot g_3 \cdot (x+1)^6 \\ f(x^2)\cdot g(x^2) + f_0 \cdot g_0 + f_3 \cdot g_3 \cdot x^{12} \\ f((x+1)^2)\cdot g((x+1)^2) + f_0 \cdot g_0 + f_3 \cdot g_3 \cdot (x+1)^{12} \end{pmatrix}}_{C_P}.$$

Here, the VM A_p is regular; thus, we can compute e_1, e_2, e_3, e_4, and e_5 by $B_p = A_p^{-1} \cdot C_p$. Next, we present an explicit representation of $A_p^{-1} = 1/\beta \cdot (\alpha_{ij})$ for i, j=0,1,2,3,4 where α_{ij} and β are obtained by the following equations[1]:

$\alpha_{00} = x^8+x^6+x^5+x^3$,
$\alpha_{20} = x^6+x^5+x^4+x^3+1$,
$\alpha_{40} = x^2+x+1$,
$\alpha_{11} = x^7+x^6+x^5+x^4+x^3+x$,
$\alpha_{31} = x^3+x$,
$\alpha_{02} = x^7+x^6+x^5+x^4$,
$\alpha_{22} = x^6+x^5+x^4+x^3+x^2+x$,
$\alpha_{42} = x^2+x$,
$\alpha_{13} = x^4+x^3+x^2+1$,
$\alpha_{33} = x^2+1$,
$\alpha_{04} = x^4+x^3$,
$\alpha_{24} = x^2+x+1$,
$\alpha_{44} = 1$,
$\alpha_{10} = x^6+x^5+x^4+x^3+x^2+x$,
$\alpha_{30} = 0$,
$\alpha_{01} = x^7+x^3$,
$\alpha_{21} = x^6+x^5+x^4+x^3+x^2+x$,
$\alpha_{41} = x^2+x$,
$\alpha_{12} = x^7+x^2$,
$\alpha_{32} = x^3+x^2$,
$\alpha_{03} = x^4+x^3+x^2+x$,
$\alpha_{23} = x^2+x+1$,
$\alpha_{43} = 1$,
$\alpha_{14} = x^4+x^3+x$,
$\alpha_{34} = x^2$,
$\beta = x^8+x^6+x^5+x^3$.

Note that these equations are calculated with coefficients in F_2. For example, the (0, 0)-component of $A_p \cdot (\alpha_{ij})$ is computed as

$$\begin{aligned} &\alpha_{00} + \alpha_{10} + \alpha_{20} + \alpha_{30} + \alpha_{40} \\ &= x^8 + 3x^6 + 3x^5 + 2x^4 + 3x^3 + 2x^2 + 2x + 2 \\ &= x^8 + x^6 + x^5 + x^3 \\ &= \beta. \end{aligned}$$

Here, $B_p = 1/\beta \cdot (\alpha_{ij}) \cdot C_p$ is a formula computing $e(z)=f(z)\cdot g(z)$ of (8). However, it still has an inversion of $1/\beta$, which is relatively slow. Recall that multiplying intermediate values in algorithm 2 by any element $r \in F_{2^n}^*$ has no effect on the pairing value due to the final exponentiation because r^w=1. Therefore, dividing by β is not necessary because $1/\beta \in F_{2^n}^*$. Thus, we eventually obtain a new formula computing $e(z)=\beta\cdot f(z)\cdot g(z)$ as follows:

$$\begin{cases} (e_1 e_2 e_3 e_4 e_5)^{\mathrm{T}} = (\alpha_{ij}) \cdot C_p, \\ e_0 = \beta \cdot (f_0 \cdot g_0), \\ e_6 = \beta \cdot (f_3 \cdot g_3). \end{cases} \tag{9}$$

Note that after computation of (9), we need a computation $(e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4 + e_5 z^5 + e_6 z^6) \bmod h(z)$, where $h(z)=z^4+z+1$.

2. Cost of Proposed Multiplication Method

The proposed multiplication method consists of computing $e(z)=\beta\cdot f(z)\cdot g(z)$ using (9) and the reduction $e(z)$ mod $h(z)$.

First we deal with the cost of the reduction modulo $h(z)=z^4+z+1$. This modulo is the following linear transform:

$$\begin{aligned} &(e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4 + e_5 z^5 + e_6 z^6) \bmod h(z) \\ &= (e_0 + e_4) + (e_1 + e_4 + e_5) z + (e_2 + e_5 + e_6) z^2 + (e_3 + e_6) z^3. \end{aligned}$$

Hence, the cost of the reduction $e(z)$ mod $f(z)$ is virtually zero. It is then enough to consider the cost of computing (9), that is, $e(z)=\beta\cdot f(z)\cdot g(z)$.

Next, we discuss the cost of computing $e(z)=\beta\cdot f(z)\cdot g(z)$. To estimate the cost of the proposed multiplication method, we have to consider the multiplication with polynomials of very small degrees. Define $\overline{M}$ as

$$\overline{M} = \frac{1}{n-1} M,$$

where M is the cost of a multiplication in F_{2^n} as defined in section II. We assume the following cost for the multiplication with polynomials of very small degrees.

Assumption 1. Let a be a random element in F_{2^n}, and let b be an element of degree d_b in F_{2^n}. Then, the cost of a multiplication $a\cdot b$ in F_{2^n} becomes $d_b \overline{M}$.

Although the cost of multiplication depends on the implementation of a multiplication, assumption 1 is correct when the multiplication is implemented by the shift-addition method [11] or its variations using window methods [4], [11], [12], which are basic multiplication methods. We can explain how assumption 1 holds.

First, we consider a multiplication of polynomials $a(x)\cdot b(x)$ in $F_2[x]$ by the shift-addition method or its variations using window methods. The basic step of their algorithms is to shift $a(x)$ from right to left (or left to right) and perform addition $a(x)+b(x)$ based on each coefficient of $b(x)$. The shift-addition method requires $d_a d_b$ shifts and $0.5 d_a d_b$ additions in F_2, where d_a and d_b are the degree of $a(x)$, and the degree of $b(x)$,

1) We found them using PARI/GP [10].

respectively. The number of shifts and additions in the variations of the shift-addition method using the window methods of width w requires $O(d_b 2^w)$ in a precomputation stage and $O(d_a d_b / w)$ in the main loop. The constant term of O notation depends on the underlying window method. Therefore, the ratio of $a(x)\cdot b(x)$ in $F_2[x]$ of d_b=d_a over $d_b \le d_a$ is equal to d_b / d_a. Then, we have

$$\text{the cost of } a(x)\cdot b(x) = \frac{d_b}{d_a}\mu, \text{ when } d_b \le d_a, \qquad (10)$$

where μ is the cost of a multiplication $a(x)\cdot b(x)$ in $F_2[x]$ of d_b=d_a.

Next, using (10), we show assumption 1 is true. In general, a random element has degree (n–1) and $d_b \le n-1$. Note that M is the same as μ in (10) because the cost of reduction modulo, $h(x)$, is virtually zero. Setting d_a=n–1, we have the cost of $a(x)\cdot b(x)$ becomes $d_b/(n-1)= d_b\mu$=$d_b\overline{M}$. This completes the explanation of assumption 1.

Now, we consider the cost of (9). First, we compute the components of the vector C_p. Then, we compute a matrix multiplication $(a_{ij})\cdot C_p$. Finally, we compute e_0 and e_6.

A. Computation of Components of C_p

To compute the components of the vector C_p, we may compute the following sets:

(i) $\{f(1), f(x), f(x+1), f(x^2), f((x+1)^2)\}$,
(ii) $\{g(1), g(x), g(x+1), g(x^2), g((x+1)^2)\}$,
(iii) $\{f(1)\cdot g(1), f(x)\cdot g(x), f(x+1)\cdot g(x+1), f(x^2)\cdot g(x^2), f((x+1)^2)\cdot g((x+1)^2)\}$,
(iv) $\{f_3 g_3, f_3 g_3 x^6, f_3 g_3 (x+1)^6, f_3 g_3 x^{12}, f_3 g_3 (x+1)^{12}\}$,
(v) $\{f_0 g_0\}$.

We can compute set (i) using algorithm 3, the cost of which is $16\overline{M}$ due to assumption 1. We can also compute set (ii) using algorithm 3. Set (iii) needs 5M. We can compute set (iv) using algorithm 4, which costs $12M+\overline{M}$. Finally, we compute $f_0\cdot g_0$, which costs M. Therefore, we take $7M+44\overline{M}$ to compute the components of C_p.

B. Computation of Matrix Multiplication $(\alpha_{ij})\cdot C_p$

Suppose C_p is represented as $(C_0, C_1, C_2, C_3, C_4)^T$ with $C_j \in F_{2^n}$ for $0 \le j \le 4$. Then, to compute $(\alpha_{ij})\cdot C_p$, we may compute $\alpha_{ij}\cdot C_j$ for $0 \le i, j \le 4$. Here, we consider the computation of set $\{(\alpha_{i0})\cdot C_0 : 0 \le i \le 4\}$. Note that the set is obtained from $x^k C_0$ with $0 \le j \le 8$, and this 8 is the maximum degree of α_{i0}. Therefore, the cost of $(\alpha_{i0})\cdot C_0$ is $8\overline{M}$. Similarly, the costs of computing sets $\{(\alpha_{i1})\cdot C_1\}$, $\{(\alpha_{i2})\cdot C_2\}$, $\{(\alpha_{i3})\cdot C_3\}$, and $\{(\alpha_{i4})\cdot C_4\}$, are $7\overline{M}$, $7\overline{M}$, $4\overline{M}$, and $4\overline{M}$, respectively. Therefore, we take $8\overline{M}+7\overline{M}+7\overline{M}+4\overline{M}+4\overline{M}=30\overline{M}$ to compute $(\alpha_{ij})\cdot C_p$.

Algorithm 3. Computation of $f(1), f(x), f(x+1), f(x^2), f((x+1)^2)$

Input: $f(x) = f_0 + f_1 z + f_2 z^2 + f_3 z^3 \in F_{2^n}[z]$, $x \in F_{2^n}$ such that x forms the basis of F_{2^n} over F_2
Output: $\{f(1), f(x), f(x+1), f(x^2), f((x+1)^2)\}$
1: $T_0 \leftarrow f_0, T_1 \leftarrow f_1, T_2 \leftarrow f_2, T_3 \leftarrow f_3$
2: $R_0 \leftarrow T_0 + T_1 + T_2 + T_3$
3: $T_1 \leftarrow T_1 \cdot x$ (cost: $\overline{M}$)
4: $T_2 \leftarrow T_2 \cdot x^2$ (cost: $2\overline{M}$)
5: $T_3 \leftarrow T_3 \cdot x^3$ (cost: $3\overline{M}$)
6: $R_1 \leftarrow T_0 + T_1 + T_2 + T_3$
7: $T_1 \leftarrow T_1 \cdot x$ (cost: $\overline{M}$)
8: $T_2 \leftarrow T_2 \cdot x^2$ (cost: $2\overline{M}$)
9: $T_3 \leftarrow T_3 \cdot x^3$ (cost: $3\overline{M}$)
10: $R_3 \leftarrow T_0 + T_1 + T_2 + T_3$
11: $T_0 \leftarrow f_3 \cdot (x^2 + x + 1)$ (cost: $2\overline{M}$)
12: $R_2 \leftarrow R_1 + f_1 + f_2 + T_0$
13: $T_0 \leftarrow T_0 \cdot (x^2 + x + 1)$ (cost: $2\overline{M}$)
14: $R_4 \leftarrow R_3 + f_1 + f_2 + T_0$
15: return $\{R_0, R_1, R_2, R_3, R_4\}$ (total cost: $16\overline{M}$)

C. Computation of e_0 and e_6

We compute $e_0 = \beta\cdot(f_0\cdot g_0)$ and $e_3 = \beta\cdot(f_3\cdot g_3)$. Recall that $f_0\cdot g_0$ and $f_3\cdot g_3$ were already computed when we computed the components of C_p. Thus, the computation of e_0 and e_6 takes $16\overline{M}$ because the degree of β is 8.

D. Total Cost of Proposed Multiplication Method

From sections IV.2.A, IV.2.B, and IV.2.C, we can evaluate (9) with $7M+90\overline{M} = \left(7+90/(n-1)\right)M$. The cost is asymptotically equal to 7M as $n\to\infty$ and is 7.38M in the case of n=239. On the other hand, the previous multiplication using the Karatsuba method requires 9M for any degree n. We summarize these estimations in Table 3.

Table 3. Computation of cost of F←F×G.

	Any n	n = 239	$n \to \infty$
Karatsuba method	9M	9M	9M
Proposed multiplication method	$\left(7+\frac{90}{n-1}\right)M$	7.38M	7M

Algorithm 4. Computation of $f_3 \cdot g_3, \cdots, f_3 \cdot g_3 \cdot (x+1)^{12}$

Input: f_3, g_3, $x \in F_{2^n}$ such that x forms the basis of F_{2^n} over F_2

Output: $\{f_3 g_3, f_3 g_3 x^6, f_3 g_3 (x+1)^6, f_3 g_3 x^{12}, f_3 g_3 (x+1)^{12}\}$

1: $T \leftarrow f_3 \cdot g_3$ (cost: M)
2: $R_0 \leftarrow T_3$, $R_2 \leftarrow T, R_4 \leftarrow T$
3: $T \leftarrow T \cdot x^2$ (cost: $2\overline{M}$)
4: $R_2 \leftarrow R_2 + T$
5: $T \leftarrow T \cdot x^2$ (cost: $2\overline{M}$)
6: $R_2 \leftarrow R_2 + T, R_4 \leftarrow R_4 + T$
7: $T \leftarrow T \cdot x^2$ (cost: $2\overline{M}$)
8: $R_1 \leftarrow T, R_2 \leftarrow R_2 + T$
9: $T \leftarrow T \cdot x^2$ (cost: $2\overline{M}$)
10: $R_4 \leftarrow T + R_4$
11: $T \leftarrow T \cdot x^4$ (cost: $4\overline{M}$)
12: $R_3 \leftarrow T, R_4 \leftarrow R_4 + T$
13: return $\{R_0, R_1, R_2, R_3, R_4,\}$ (total cost: $M+12\overline{M}$)

3. Applying Proposed Multiplication to General Cases

In this subsection, we discuss applying the proposed multiplication to general cases. Recall that the proposed multiplication computes relaxed multiplication $\beta \cdot F \times G$ for $F, G \in F_{2^{4n}}$, where $\beta \in F_{2^n}$ is an element. We easily make the general version of the proposed multiplication in any finite field $F_{p^{nk}}$; thus, C_p in (9) becomes the $(2k-3)$th vector and (α_{ij}) becomes a $(2k-3) \times (2k-3)$ matrix. Define M_{p^n} as the cost of a multiplication in F_{p^n}.

Note that the proposed multiplication is effective in pairing computations because β powers to 1 in the final exponentiation and has no effect on the pairing value. Then, we consider applying the proposed multiplication to other fields in Table 1.

A. Case of (F_{3^n}, k=6)

The F_{3^n} version of the proposed multiplication takes $\left(11+\gamma/(n-1)\right)M_{3^n}$, where γ is a constant number depending on a vector C_p and a matrix (α_{ij}) in (9). Therefore, the $F_{3^{6n}}$ version of the proposed multiplication asymptotically takes $11M_{3^n}$ to compute a relaxed multiplication in $F_{3^{6n}}$. However, if the Vandermonde matrix A_p is not carefully constructed, then γ seems to be very large, and the proposed multiplication incurs a greater cost than the GPS multiplication (see [3] or section III.3) which takes $15M_{3^n}$ in practical cases as n is several hundred. Finding the vector C_p and the matrix (α_{ij}) such that A becomes smallest will be the subject of future work.

B. Case of (F_p, k=*any value*)

In this case components of C_p and (α_{ij}) are elements in F_p, and computation of $(\alpha_{ij}) \cdot C_p$ in (9) in general takes $(2k-3)^2 M_p$. Therefore, the F_p version of the proposed multiplication takes more than $(2k-3)^2 M_p$, which means the Karatsuba method is better than the F_p version of the proposed method.

V. New Efficient Algorithm for η_T Pairing

In this section, we present a new efficient algorithm for computing the η_T pairing using the new multiplication method presented in section IV.

As we described in the last paragraph of section IV.2, if $F \times G$ (random×random) at step 10 in algorithm 2 is efficiently computed, we can obtain an algorithm for computing the η_T pairing. Then, we proposed an efficient multiplication method computing $\beta \cdot F \times G$ instead of $F \times G$ in section IV.4. Recall that β powers to 1 in the final exponentiation, and therefore has no effect on the pairing value. Consequently, we can obtain a new algorithm for computing the η_T pairing, namely, algorithm 5, applying the proposed multiplication method to algorithm 2.

Note that we need to use the *z*-basis, not the *st*-basis, to provide the new algorithm for the η_T pairing. However, the conversion between the *st*-basis and the *z*-basis is virtually zero due to (1). In algorithm 5, steps 2 to 18 are executed using the *z*- basis, and the result of step 18 is converted to a value of the *st*-basis. The output of algorithm 5 is then the same as algorithm 2.

Next, we estimate algorithm 5 without the final exponentiation, which can be efficiently computed [7]. Note that the cost of each step of algorithm 5 except step 10 is the same as those of algorithm 2 due to (1). Let m be the number of iterations of the main loop of algorithm 5. Then, the cost of algorithm 5 is calculated as

(the cost of algorithm 5)
=(the cost of algorithm 2)
$+m \cdot$ ((the cost of step 9 in algorithm 5)
−(the cost of step 9 in algorithm 2))
$=((3.5n+4.5)M+(n-1)S+(n-1)R)+m \cdot ((7+90/(n-1))M-9M$

because the cost of algorithm 2 is $(3.5n+4.5)M+(n-1)S+(n-1)R$ as described in section II, where M, S, and R are the cost of a multiplication, a squaring, and a square root in F_{2^n}, respectively. Note $m=(n-1)/4$ if $n \equiv 1 \pmod 4$, and $m=(n+1)/4$ if $n \equiv 3 \pmod 4$. Then, we have the cost of algorithm 5 as $((3n^2+24.5n-27.5)/(n-1))M+(n-1)R+(n-1)S$ if $n \equiv 1 \pmod 4$ and as $((3n^2+23.5n+18.5)/(n-1))M+(n-1)R+(n-1)S$ if $n \equiv 3 \pmod 4$.

Algorithm 5. Proposed algorithm for η_T pairing

Input: $P=(\alpha,\beta), Q=(\gamma,\delta)\in E^b(F_{2^n})$

Output: $\eta_T(P,Q)\in F^*_{2^{4n}}$

1: $w\leftarrow \alpha+\left\lceil\frac{n-1}{2}\right\rceil$

2: $F\leftarrow w\cdot(\gamma+\alpha+1)+\delta+(\beta+b+\varepsilon_{(n+1)/2})$
$+(w+\gamma+1)z+(w+\gamma)z^2$ (cost : M)

3: for i=0 to $\lfloor(n-3)/4\rfloor$ do

4: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

5: $G_0\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+\delta+(\beta+b+\varepsilon_{(n-1)/2})$
$+(w+\gamma+1)z+(w+\gamma)z^2$ (G_0 is z-sparse, cost: M)

6: $\alpha\leftarrow\sqrt{\alpha},\beta\leftarrow\sqrt{\beta},\gamma\leftarrow\gamma^2,\delta\leftarrow\delta^2$ (cost: 2R+2S)

7: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

8: $G_1\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+\delta+(\beta+b+\varepsilon_{(n-1)/2})$
$+(w+\gamma+1)z+(w+\gamma)z^2$ (G_1 is z-sparse, cost: M)

9: $G\leftarrow G_0\bowtie G_1$ (cost: 3M)

10: $F\leftarrow\beta\cdot F\times G$, where $\beta=x^8+x^6+x^5+x^3$
(computed by (9), cost: by Table 3)

11: $\alpha\leftarrow\sqrt{\alpha},\beta\leftarrow\sqrt{\beta},\gamma\leftarrow\gamma^2,\delta\leftarrow\delta^2$ (cost: 2R+2S)
(11 is unnecessary if $n\equiv 3 \pmod 4$ for $i=\lfloor(n-3)/4\rfloor$)

12: end if

13: if $i\equiv 1 \pmod 4$ then

14: $w\leftarrow \alpha+\left\lceil\frac{n+1}{2}\right\rceil$

15: $G_0\leftarrow w\cdot\left(\gamma+\left\lceil\frac{n+1}{2}\right\rceil\right)+\delta+(\beta+b+\varepsilon_{(n-1)/2})$
$+(w+\gamma+1)z+(w+\gamma)z^2$ (G_0 is z-sparse, cost: M)

16: $F\leftarrow F\bowtie G$ (cost : 6M)

17: end if

18: $(f_0, f_1, f_2, f_3)_z\leftarrow F^W$ (cost: F_E)

19: return $(f_0, f_2+f_3, f_1+f_2+f_3, f_3)_{st}$

We aim to estimate the costs of algorithms 1, 2, and 5 using only M to easily compare the proposed algorithm 5 with the conventional algorithms 1 and 2. We then need a relationship among M, S, and R. Suppose that R=0.5M as followed by Kwon [7]. Note that $a^{1/2}=a^{2^{n-2}}$ for any $a\in F_{2^n}$ because $a^{2^n}=a$. Therefore, (n–2)S=R, that is, $S=1/(2n-4)\cdot M$. The comparison is shown in Table 4.

Table 4. Cost comparison between algorithms 1 and 2 and algorithm 5.

	n=239	$n\to\infty$
Algorithms 1 and 2	960.5M	4nM
Proposed algorithm 5	863.2M	3.5nM

Table 5. Comparison of multiplication cost in $F_{2^{4n}}$.

Number of multiplications in $F_{2^{4n}}$	Cost
Without loop unrolling	
(n+1)/2 "sparse by random"	3(n+1)M (Karatsuba)
With loop unrolling	
(n+1)/4 "sparse by sparse"	0.75(n+1)M (Karatsuba)
(n+1)/4 "random by random"	2.25(n+1)M (Karatsuba or GPS) **1.75(*n*+1)*M* (Proposed)**
Total	3 (n+1)M (Karatsuba or GPS) **2.5 (*n*+1)*M* (Proposed)**

Therefore, the cost of computing the η_T pairing is reduced by 11.3% and 14.3% for n=239 and $n\to\infty$, respectively.

VI. Conclusion

The loop unrolling technique and the GPS multiplication are effective in the η_T pairing on supersingular curve in characteristic three. However, they are ineffective in the η_T pairing in characteristic two. In the case of characteristic two, the loop unrolling technique changes (n+1)/2 multiplications of "sparse element by random element" in $F_{2^{4n}}$ in the algorithm for computing the η_T pairing (algorithm 1) to (n+1)/4 "sparse by sparse" multiplications and (n+1)/4 "random by random" multiplications. Note that multiplications of a "sparse by random," a "sparse by sparse," and a "random by random" in $F_{2^{4n}}$ take 6M, 3M, and 9M, respectively, using the Karatsuba method, where M means the cost of a multiplication in F_{2^n}. The GPS multiplication is a method to compute a multiplication of "random by random," and in the case of $F_{2^{4n}}$ the GPS multiplication incurs the same cost as the Karatsuba method. Then, the loop unrolling technique and the GPS multiplication are ineffective in the η_T pairing in characteristic two.

We introduced a relaxed multiplication which computes $\beta\cdot F\times G$ instead of $F\times G$ for $F, G\in F_{2^{4n}}$, where β is an element in F_{2^n}. The β powers to 1 in the final exponentiation

and therefore has no effect on the pairing value. The relaxed multiplication can be computed using the Vandermonde matrix, the inverse of which has coefficients of small degrees. The multiplication in $F_{2^{4n}}$ can be computed by 7 multiplications in F_{2^n} for $n \to \infty$, while the previously fastest Karatsuba method requires 9 multiplications. Consequently, the cost of the η_T pairing computation is reduced by 14.3% as $n \to \infty$.

Appendix. Sparse Multiplication

To describe the cost of (c) in Table 3, we provide the algorithms A1 and A2 for computing a multiplication of (*st, z*)-sparse elements in $F_{2^{4n}}$.

Algorithm A1. Multiplication of *st*-sparse elements in $F_{2^{4n}}$

Input: *st*-sparse $a=(a_0, a_1, 1, 0)_{st}$, $b=(b_0, b_1, 1, 0)_{st} \in F_{2^{4n}}$

Output: $c=(c_0, c_1, c_2, c_3)_{st}:=a \rhd\!\lhd b$

1: $T_0 \leftarrow a_0 \cdot b_0$ (cost: M)
2: $T_1 \leftarrow a_1 \cdot b_1$ (cost: M)
3: $T_2 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$ (cost: M)
4: $R_0 \leftarrow T_0 + T_1$
5: $R_1 \leftarrow T_0 + T_2 + 1$
6: $R_2 \leftarrow a_0 + b_0 + 1$
7: $R_3 \leftarrow a_1 + b_1$
8: return $(R_0, R_1, R_2, R_3)_{st}$

Algorithm A2. Multiplication of *z*-sparse elements in $F_{2^{4n}}$

Input: *z*-sparse $a=(a_0, a_1+1, a_1, 0)_z$, $b=(b_0, b_1+1, b_1, 0)_z \in F_{2^{4n}}$

Output: $c=(c_0, c_1, c_2, c_3)_z:=a \rhd\!\lhd b$

1: $T_0 \leftarrow a_0 \cdot b_0$ (cost: M)
2: $T_1 \leftarrow a_1 \cdot b_1$ (cost: M)
3: $T_2 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$ (cost: M)
4: $R_0 \leftarrow T_0 + T_1$
5: $R_1 \leftarrow T_0 + T_2 + a_0 + b_0$
6: $R_2 \leftarrow T_0 + T_2 + a_1 + b_1 + 1$
7: $R_3 \leftarrow a_1 + b_1$
8: return $(R_0, R_1, R_2, R_3)_z$

References

[1] P. Barreto et al., "Efficient Pairing Computation on Supersingular Abelian Varieties," *Designs, Codes and Cryptography*, vol. 42, no. 3, 2007, pp. 239-271.

[2] R. Granger, D. Page, and M. Stam, "Hardware and Software Normal Basis Arithmetic for Pairing-Based Cryptography in Characteristic Three," *IEEE Transactions on Computers*, vol. 54, no. 7, 2005, pp. 852-860.

[3] E. Gorla, C. Puttmann, and J. Shokrollahi, "Explicit Formulas for Efficient Multiplication in $F_{3^{6m}}$," *SAC 2007*, LNCS 4876, 2007, pp. 163-183.

[4] D.E. Knuth, *Seminumerical Algorithms*, Addison-Wesley, 1981.

[5] F. Hess, N. Smart, and F. Vercauteren, "The Eta Pairing Revisited," *IEEE Transactions on Information Theory*, vol. 52, no. 10, 2006, pp. 4595-4602.

[6] D.H. Choi, D.G. Han, and H.W. Kim, "Construction of Efficient and Secure Pairing Algorithm and Its Application," *Cryptology ePrint Archive*, Report 2007/296, 2007.

[7] S. Kwon, "Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields," *Cryptology ePrint Archive*, Report 2004/303, 2004.

[8] J.-L. Beuchat et al., "Algorithms and Arithmetic Operators for Computing the η_T Pairing in Characteristic Three," *Cryptology ePrint Archive*, Report 2007/417, 2007.

[9] P. Bürgsser, M. Clausen, and M. Shokrollahi, *Algebraic Complexity Theory*, Springer-Verlag, 1997.

[10] PARI/GP, http://pari.math.u-bordeaux.fr/download.html

[11] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[12] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

Masaaki Shirase received the BSc in mathematics from Ibaraki University in 1994, and MIS and DrIS degrees from Japan Advanced Institute of Science and Technology (JAIST) in 2003 and 2006, respectively. He is currently a research associate with the School of System Science Information at Future University-Hakodate. His research interests are algorithms and implementation of cryptography.

Tsuyoshi Takagi received the BSc and MSc degrees in mathematics from Nagoya University in 1993 and 1995, respectively. He engaged in research on network security at NTT Laboratories from 1995 to 2001. He received the Dr.rer.nat degree from Technische Universität Darmstadt in 2001. He was an assistant professor with the Department of Computer Science at Technische Universität Darmstadt until 2005. He is currently a professor with the School of Systems Information Science at Future University-Hakodate. His current research interests are information security and cryptography. Dr. Takagi is a member of International Association for Cryptologic Research (IACR).

Dooho Choi received his BS degree in mathematics from Sungkyunkwan University, Seoul, Korea in 1994, and the MS and PhD degrees in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1996 and 2002, respectively. He has been a senior researcher with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea since Jan. 2002. His research interests include security technologies of RFID and wireless sensor networks. He is an editor of the ITU-T X.1171 (X.nidsec-1).

DongGuk Han received his BS degree in mathematics from Korea University in 1999, and his MS degree in mathematics from Korea University in 2002. He received his PhD of engineering in Information Security from Korea University in 2005. He was a PostDoc with Future University-Hakodate, Japan. After finishing the doctoral course, he was an exchange student with the Department of Computer Science and Communication Engineering, Kyushu University, Japan from April 2004 to March 2005. He was a senior researcher in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea. He is currently working as an assistant professor with the Department of Mathematics of Kookmin University, Seoul, Rep. of Korea. He is a member of KIISC, IEEK, and IACR.

Howon Kim received his BSEE degree from Kyungpook National University, Daegu, Korea, in 1993, and the MS and PhD degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied at the COSY group at the Ruhr-University of Bochum, Germany. He was a senior member of technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He is currently working as an assistant professor with the Department of Computer Engineering, Pusan National University, Busan, Korea. His research interests include RFID technology, sensor networks, information security, and computer architecture. Currently, his main research focus is on mobile RFID technology and sensor networks, public key cryptosystems and their security issues. He is a member of the IEEE, IEEE Computer Society, and IACR.