Power Analysis Attacks and Countermeasures on η_T Pairing over Binary Fields

Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Howon Kim, and Jongin Lim

Since many efficient algorithms for implementing pairings have been proposed such as η_T pairing and the Ate pairing, pairings could be used in constraint devices such as smart cards. However, the secure implementation of pairings has not been thoroughly investigated. In this paper, we investigate the security of η_T pairing over binary fields in the context of side-channel attacks. We propose efficient and secure η_T pairing algorithms using randomized projective coordinate systems for computing the pairing.

Keywords: Pairing-based cryptosystems, side-channel attacks, power analysis, randomized projective coordinate systems, η_T pairing.

Manuscript received Mar. 19, 2007; revised Dec. 05, 2007.

This work was supported partly by the MIC (Ministry of Information and Communication), Rep. of Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2008-(C1090-0801-0025)) and by the IT R&D program of MIC/IITA, Rep. of Korea [2005-S088-04, Development of Security technology for Secure RFID/USN Service].

Tae Hyun Kim (email: thkim@cist.korea.ac.kr) and Jongin Lim (email: jilim@korea.ac.kr) are with the Center for Information Security Technologies (CIST), Korea University, Seoul, Rep. of Korea.

Tsuyoshi Takagi (email: takagi@fun.ac.jp) is with the School of Systems Information Science, Future University-Hakodate, Japan.

Dong-Guk Han (phone: + 82 42 860 1784, email: christa@etri.re.kr) and Howon Kim (email: khw@etri.re.kr) are with Information Security Research Division, ETRI, Daejeon, Rep. of Korea.

I. Introduction

Since pairings have new and useful cryptographic properties such as bilinearity and non-degeneracy, there has been a great deal of recent interest and active research into them in the field of cryptography. Recently, many cryptographic schemes based on Tate pairing and Weil pairing have been proposed. For example, identity-based encryption schemes [1], [2], identity-based signature schemes [3]-[5] short signature scheme [6], and identity-based authenticated key agreement scheme [7].

To facilitate the practical application of pairing-based schemes, much work has focused on the development of efficient and easy computations of pairings on elliptic curves. Barreto and others [8] and Galbraith and others [9] provided the fast computation of the Tate pairing on supersingular elliptic curves defined over finite fields of characteristics two and three. Duursma and Lee [10] gave a closed formula in the case of characteristic three, and Kwon [11] extended it to supersingular curves in characteristic two. Barreto and others [12] proposed η_T pairing, which is a general technique for the efficient computation of pairings on supersingular Abelian varieties.

Recently, such methods for computing pairings have been implemented in software and hardware for application in constrained devices, such as smartcards [13]-[17]. In the implementation of cryptosystems or protocols on such devices, we should consider both efficiency and security. If cryptosystems are not carefully implemented on constrained devices, they could be insecure against side-channel attacks (SCAs). Therefore, it is important to consider the secure implementation of pairing-based cryptosystems against SCAs.

Pairing-based schemes can be divided into two types according to whether or not the input of a pairing is secret [18].

For example, identity-based signature schemes, such as the short signature scheme proposed by Boneh and Franklin, require secret information as an input (such as, the secret scalar) of the elliptic curve scalar multiplication, not of the pairing. SCAs and countermeasures on scalar multiplications have been thoroughly studied. However, identity-based encryption schemes, such as the Boneh-Franklin encryption scheme [1] use secret information as an input of the pairing. In this case, there are only a few studies of SCAs on these pairings [19]-[22]. In [19], Page and Vercauteren described SCAs against the Duursma-Lee algorithm and the BKLS algorithm. They focused SCAs on a multiplication operation over characteristic three. In [21], Scott suggested countermeasures to provide resistance to more sophisticated simple power analysis (SPA) attacks and differential power analysis (DPA) attacks. Very recently, Whelan and Scott discussed the security of practical pairings (Tate, η_T, and Ate pairing), which include the case of characteristic two, using correlation power analysis (CPA) [22]. In particular, they investigated all field operations used in the η_T pairing over characteristic two.

However, in this paper, we describe the difference between finite field multiplications used in the η_T pairings over characteristic two and three. Then, we concretely examine the security of the η_T pairing on supersingular curves over F_{2^m} against timing attacks (TAs), SPA attacks and DPA attacks, which are different from Whelan-Scott attacks.

In general, to speed up elliptic curve point addition and doubling, projective coordinate systems are used instead of the affine coordinate system because the affine coordinate system requires a modular inversion operation, which is computationally expensive. In [23], Izu and Takagi showed that the Tate pairing on general elliptic curves over prime fields F_{p^m} is efficiently computed using projective coordinate systems. In [24], Chatterjee and others considered the use of Jacobian coordinates for Tate pairing computation in general characteristics. In [25], Hess and others proposed Ate pairing, which is an extension of η_T pairing of supersingular curves to general curves over prime fields F_{p^m}, and examined efficiency in the projective coordinate systems. To provide protection against SCAs, Coron [26] used randomized projective coordinates. In this paper, to resist SCAs, we propose explicit algorithms using the randomness of projective coordinate systems in the η_T pairing for a curve in characteristic two.

This paper is organized as follows. In the next section, we review several methods for the efficient computation of Tate pairing. Section III describes side-channel attacks on the η_T pairing of supersingular curves in characteristic two. Section IV presents countermeasures to prevent the attacks described in Section III. Section V compares the proposed countermeasure with previous methods. Finally, we conclude in Section VI.

II. Tate Pairing

Let E be an elliptic curve over a finite field F_q. The number of points in $E(F_q)$, called the order of the elliptic curve group, is denoted by $\#E(F_q)$. Let l be a positive integer coprime to q, which divides the order of the elliptic curve $E(F_q)$, that is, $l \mid \#E(F_q)$. Then, E has embedding degree or security parameter k with respect to l if k is the smallest positive integer such that the subgroup order l divides (q^k-1) but does not divide (q^i-1) for any $1<i<k$.

A divisor D on E is a formal sum of the points P on the curve

$$D = \sum n_P(P),$$

where $n_P \in \mathbb{Z}$, and $n_P = 0$ for all but finitely many $P \in E$. The support of a divisor D is the set of points such that $n_P \neq 0$.

Let $f : E(F_{q^k}) \rightarrow F_{q^k}$ be a rational function on E. Given a nonzero rational function f, the divisor of f is defined as $(f) = \sum \mathrm{ord}_P(f)(P)$, where $\mathrm{ord}_P(f)$ is the order of the zero or pole of f at P (if f has no zero or pole at P , then $\mathrm{ord}_P(f) = 0$). A divisor D is called principal if $D = (f)$ for some rational function f. It is well known that a divisor $D = \sum n_P(P)$ is principal if, and only if, $\sum n_P = 0$ and $\sum [n_P]P = O$, where O denotes the point at infinity [27]. Two divisors D_1 and D_2 are equivalent (denoted by $D_1 \sim D_2$) if the difference D_1-D_2 is a principal divisor. Let $P \in E[l]$ and D be a divisor equivalent to $(P) - (O)$. Then nD is principal, and thus, there is a rational function $f_{l,P}$ such that $(f_{l,P}) = nD = l(P) - l(O)$. The Tate pairing of order l is defined as follows.

Definition 1. The Tate pairing is a map

$$e_l : E(F_q)[l] \times E(F_{q^k})/lE(F_{q^k}) \rightarrow \mu_l$$

$$e_l(P,Q) = f_{l,P}(D_Q)^{(q^k-1)/l},$$

where $f_{l,P}$ is a rational function such that $(f_{l,P}) = l(P) - l(O)$, D_Q is equivalent to $(Q) - (O)$ such that D_Q and $(f_{l,P})$ have disjoint supports, and μ_l is the subgroup of l-th roots of unity of $F^*_{q^k}$.

The first efficient method for computing such a rational function was proposed by Miller [27]. This algorithm is based on the binary method for elliptic curve scalar multiplication combined with an evaluation of the tangent lines used in the elliptic curve addition process. In the original Miller algorithm, a denominator in the step of an evaluation of the tangent lines should be computed. Barreto and others [8] demonstrated a method to speed up the computation by eliminating the denominator. For supersingular elliptic curves, they used a specific endomorphism $\psi : E(F_q)[l] \rightarrow E[l]$ such that the image is linearly independent of $E(F_q)[l]$. Such a map is called a distortion map [28]. Therefore, the modified Tate pairing of P and Q on $E(F_q)$ of order l is defined as follows.

Definition 2. Let E be a supersingular elliptic curve over a finite field F_q. For P, $Q \in E(F_q)[l]$, the modified Tate pairing with an endomorphism ψ is defined by

$$e_l(P,Q) = f_{l,P}(\psi(Q))^{(q^k-1)/l}.$$

To improve the computation speed of this pairing on curves, we can use $N = hl$ as a multiple of the order of the elliptic curve for some integer h instead of l [9]. Since $(f_{N,P}) = h(f_{l,P}) = (f_{l,P}^h)$ the Tate pairing can be computed by $f_{N,P}(\psi(Q))^M$, where $M = (q^k-1)/N$, $N = hl$, and $f_{N,P}$ is a rational function such that $(f_{N,P}) = N(P) - (N)(O)$. Using this property, Duursma and Lee [10] and Kwon [11] replaced l by $N = hl$, which has low Hamming weight in the cases of characteristics three and two. In characteristic two, that is, $q = 2^m$, the order and the embedding degree of supersingular curves E: $y^2 + y = x^3 + x + b$, where $b \in \mathrm{F}_2$ are $2^m \pm 2^{(m+1)/2} + 1$ and 4, respectively. Thus, we can use $N = 2^{2m} + 1 = (2^m + 2^{(m+1)/2} + 1)(2^m - 2^{(m+1)/2} + 1)$ of Hamming weight 2 in the binary representation. Also, the final exponentiation by $(2^{4m} - 1)/(2^{2m} + 1) = 2^{2m} - 1$ is very simple. It is computed by applying one Frobenius map and one division [10], [11].

The fastest method for computing the Tate pairing on supersingular curves is η_T pairing [12], which includes the algorithms by Duursma and Lee [10] and Kwon [11] as special cases.

Definition 3. For suitable T, the η_T pairing of P and Q on $E(\mathrm{F}_q)$ is defined by

$$\eta_T(P,Q) = f_{T,P}(\psi(Q)).$$

The following relation exists between the η_T pairing and the Tate pairing.

$$\left(\eta_T(P,Q)^M\right)^{aT^{a-1}} = e_N(P,Q)^L,$$

where $T^a + 1 = LN$ for some $a \in \mathbb{N}$ and $L \in \mathbb{Z}$, $T = q + cN$ for some $c \in \mathbb{Z}$, and $M = (q^k - 1)/N$. Thus, the aim of the η_T pairing is to choose values of T which are smaller than N.

We now present an outline of the η_T pairing algorithm. The elliptic curve we are interested in is the supersingular curve E: $y^2 + y = x^3 + x + b$ over F_{2^m}, where $m \equiv 3 \bmod 8$ and $b \in \mathrm{F}_2$. To reduce the number of loops by half, we can choose $N = \#E(\mathrm{F}_q) = 2^m \pm 2^{(m+1)/2} + 1$. Then, $T = q - N = \mp 2^{(m+1)/2} - 1$, $c = -1$, $a = 2$, and $L = 2$. The η_T pairing computes the Miller function $f_{T,P}$, which requires $(m-1)/2$ elliptic curve doublings and one elliptic curve addition. Given a point $P = (x_P, y_P)$, the tangent line in doubling P is given by $g_P(x, y) = (x_P^2 + 1)(x_P + x) + y_P + y$. The function g_P has divisor $2(P) + ([-2]P) - 3(O)$. Hence, by the definition of the η_T pairing, we have

$$f_{T,P}(\psi(Q)) = \left(\prod_{i=0}^{(m-1)/2} \left(g_{[2^i]P}(\psi(Q))\right)^{2^{\frac{(m-1)}{2}-i}}\right) l(\psi(Q)), \quad (1)$$

where the function l comes from the elliptic curve addition of $[2^{(m+1)/2}]P$ with P. By substituting $P' = [2^{(m-1)/2}]P$ and $j = (m-1)/2 - i$, we have

$$f_{T,P}(\psi(Q)) = l(\psi(Q)) \prod_{j=0}^{(m-1)/2} \left(g_{[2^{-j}]P'}(\psi(Q))\right)^{2^j}.$$

Let the extension field $\mathrm{F}_{2^{4m}}$ be represented by the basis $\{1, s, t, st\}$ such that $s^2 + s + 1 = 0$ and $t^2 + t + s = 0$. The distortion map is $\psi(x,y) = (x + s^2, y + sx + t)$. Then, $[2^i]P = \varphi^i(x_P^{(2i)}, y_P^{(2i)})$, where $\varphi(x,y) = (x+1, y+x)$, and $x^{(j)}$ denotes x^{2^j}. With these prerequisites, the connecting line is given by $l(x,y) = y + \lambda(x + x_P) + y_P + \delta$, where $\lambda = x_P$ when $m \equiv 1, 5 \pmod 8$; $\lambda = x_P + 1$ when $m \equiv 3, 7 \pmod 8$; $\delta = b$ when $m \equiv 1, 7 \pmod 8$; and $\delta = b + 1$ when $m \equiv 3, 5 \pmod 8$.

In case of $m \equiv 3 \bmod 8$, $2^{[-j]}P' = \varphi^{(m-1)/2-j}(x_P^{(-1-2j)}, y_P^{(-1-2j)})$. Thus, the function $g_{[2^{-j}]P'}(\psi(Q))^{(j)}$ is given by

$$x_P^{(-j)}(x_Q^{(j)} + x_P^{(-1-j)}) + y_P^{(-1-j)} + x_P^{(-1-j)} + y_Q^{(j)} + (x_P^{(-j)} + x_Q^{(j)})s + t. \quad (2)$$

The concrete algorithm of the η_T pairing on supersingular curves in characteristic two with $m \equiv 3 \bmod 8$ is shown in algorithm 1.

Algorithm 1. $\eta_T(P,Q)$ on the curve E: $y^2 + y = x^3 + x + b$ over F_{2^m} where $m \equiv 3 \bmod 8$ and $b \in \mathrm{F}_2$ [12]

Input: $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.
Output: $\eta_T(P,Q)$.
1. $u = x_P + 1$
2. $f = u(u + x_Q) + y_P + y_Q + b + 1 + (u + x_Q)s + t$
3. For i=0 to $(m - 1)/2$ do
4. $\quad u = x_P, x_P = x_P^{1/2}, y_P = y_P^{1/2}$
5. $\quad g = u(x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t$
6. $\quad f = fg$
7. $\quad x_Q = x_Q^2, y_Q = y_Q^2$
8. Return $f^{(2^{2m}-1)(2^m - 2^{(m+1)/2}+1)}$

However, to obtain the same result as the Tate pairing, it must be further exponentiated to the power of T because of the following relation between the η_T pairing and the Tate pairing:

$$\left(\eta_T(P,Q)^M\right)^{2T} = e_N(P,Q)^2 \Rightarrow \left(\eta_T(P,Q)^M\right)^T = e_N(P,Q).$$

III. Side-Channel Attacks

SCAs have been recognized as serious menaces to constrained devices, such as smartcards. By monitoring computation timing, power consumption, electromagnetic radiation, and so on during cryptographic operations, it is possible to recover the secret information related to the keys inside the device [26], [29]. TAs analyze the time taken to

execute cryptographic algorithms. SPA attacks directly interpret power consumption measurements collected during cryptographic operations. DPA attacks analyze the correlation between the secret and the power consumption and specific key-dependent intermediate values which appear during computation. These attacks use statistical tools and error correction techniques.

In this section we investigate the η_T pairing used in identity-based encryption schemes, such as Boneh-Franklin encryption scheme [1] and the Sakai-Kasahara encryption scheme [2] in the context of SCAs, such as TAs, SPAs, and DPAs.

1. Weak Points in Pairing Computation

In the decryption step of identity-based encryption schemes, a dominant operation is $e(S_{ID},U)$, where S_{ID} is the fixed secret key, and U is a part of a ciphertext. In this case, SCAs may try to extract the secret key from the pairing computation by repeatedly manipulating U. Recently, Page and Vercauteren [19] presented SPA and DPA attacks on a field multiplication step of the pairing computation with the secret value. They showed that there are such field multiplications in the Duursma-Lee algorithm [10] and the BLKS algorithm [8] of characteristic three, that is, the multiplication of y and r where y is an unknown and fixed value related with the y-coordinate of the secret point S_{ID}, and r is a known and variable value related with the ciphertext U. The process of a field multiplication such as the binary method or m-array method is analogous to that of exponentiation on a multiplicative group or scalar multiplication on an additive group since the multiplicand is processed one bit or m bits at a time. Thus, we can easily apply DPA attacks as in [30].

However, the η_T pairing in characteristic two includes the multiplication of the different form $a(b+r)$ compared with the case of characteristic three, where both a and b are unknown and are related to the secret value S_{ID}. In this case, since r is chosen by an attacker and is added to by an unknown value b, we may not simulate or guess an intermediate value related with the secret value. Thus, the Page-Vercauteren attack cannot be directly applied to the η_T pairing over characteristic two, and it seems secure against SCAs. However, in this paper, we will show that the addition and the multiplication of $a(b+r)$ can be vulnerable to DPA attacks and TA or SPA attacks.

A. Assumption

From this section on, we assume that the first input $P = (x_P,y_P)$ of the pairing in algorithm 1 is secret, and the second input $Q = (x_Q,y_Q)$ is public and known to or even chosen by the attacker. Note, however, that the descriptions of the attacks are similar even if P is public and Q is secret. For the computation of $a(b+r)$, we also assume that the addition $(b+r)$ is computed before the multiplication. Then the multiplication $a(b+r)$ is computed. We assume that the multiplication is computed by the right-to-left binary method.

2. Finite Field Arithmetic

Let $f(x)$ be an irreducible polynomial of degree m over F_2. Assume that an element $a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ of $F_{2^m} \cong F_2[x]/(f(x))$ is represented by the polynomial basis. Let the bit string be $(a_{m-1}a_{m-2}\cdots a_1a_0)$, where $a_i \in F_2$ denotes an element a of F_{2^m}. Addition and multiplication of $a = (a_{m-1}a_{m-2}\cdots a_1a_0)$ and $b = (b_{m-1}b_{m-2}\cdots b_1b_0)$ in F_{2^m} are performed as follows.

- Addition: $a + b = (c_{m-1}c_{m-2}\cdots c_1c_0)$, where $c_i = (a_i + b_i) \bmod 2$.
- Multiplication: $c = ab = (c_{m-1}\cdots c_1c_0)$, where c is computed as a multiplication of polynomials $a(x)$ and $b(x)$ of $F_2[x]$, followed by a reduction by $f(x)$. That is, $c = a(x)b(x) \bmod f(x)$.

The addition of the two elements a and b in F_{2^m} is easily performed by a bitwise XOR operation. A general method of multiplying the two elements a and b of F_{2^m} is done by scanning one bit of the multiplier b at a time. This method is known as the binary method and is based on the following observation:

$$a(x)b(x) = a_{m-1}x^{m-1}b(x) + \cdots + a_1xb(x) + a_0b(x).$$

For efficiency, the irreducible polynomial is selected as a trinomial or a pentanomial. Therefore, we assume that a multiplication of polynomials is performed before modulo reduction. Then, the result is reduced by an irreducible polynomial. Note, however, that the attack is not limited to the previous multiplication and the separate computation of multiplication and reduction. This attack can be extended to other multiplication methods and the interleaving multiplication and reduction method. An algorithm of the binary method is given in algorithm 2.

Algorithm 2. Right-to-left multiplication method
Input: $a(x) = (a_{m-1}a_{m-2} \ldots a_1a_0)$, $b(x) = (b_{m-1}b_{m-2} \ldots b_1b_0)$,
an irreducible polynomial $f(x)$.
Output: $c(x) = a(x)\,b(x) \bmod f(x)$.
1. $C = 0$ and $B = b(x)$
2. For i=0 to m - 1 do
3. if $a_i = 1$ then $C = C + B$
4. $B = Bx$
5. Return $C \bmod f(x)$

3. SPA or Timing Attack

We consider the multiplication $a(x)(b(x)+r(x))$. If $a(x)$ is a multiplier, the addition is performed depending on whether $a_i = 1$.

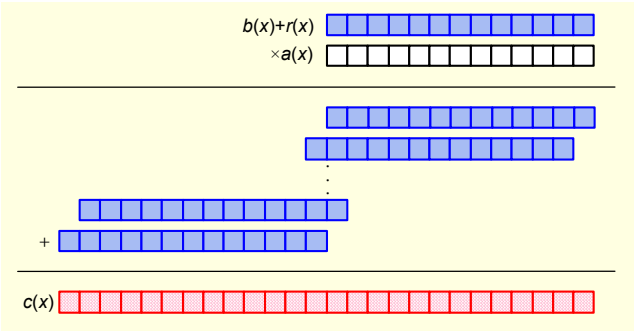

Fig. 1. Right-to-left multiplication of $a(x)(b(x) + r(x))$ in F_{2^m} when $a(x)$ is a multiplier and $b(x) + r(x)$ is a multiplicand. The addition must be performed before the multiplication.

The structure of the right-to-left method for the multiplication of $a(x)(b(x)+r(x))$ is shown in Fig. 1.

Page and Vercauteren [19] first presented an SPA attack against $a(x)r(x)$ in F_{3^m}, where $a(x)$ is secret and $r(x)$ is public. It assumes that this conditional branch is vulnerable to TA or SPA attacks [26], [29]. Thus, we can also recover u of $u(x_P+x_Q)$, which is a part of step 5 of algorithm 1, even though $x_P + x_Q$ is unknown. In this case, u is the x-coordinate of the secret point P. Then, we can find the y-coordinate from the x-coordinate by solving the quadratic equation of elliptic curves.

However, if $b(x)+r(x)$ is a multiplier, then the conditional branch occurs, depending on b_i+r_i. If we can detect an appearance of the conditional branch by a TA or SPA attack, then $b_i \neq r_i$. Otherwise, $b_i = r_i$. Thus, since x_P and x_Q of algorithm 1 correspond to b_i and r_i, we can recover x_P by controlling x_Q in $u(x_P+x_Q)$. Since x_P is the square root of the x-coordinate of the secret point P, we can obtain the x-coordinate by squaring x_P and then obtain the y-coordinate from the x-coordinate of the secret point P.

4. DPA Attacks

In this section, we investigate DPA attacks against the addition $b(x)+r(x)$ and the multiplication of $a(x)$ and $(b(x)+r(x))$ used in the η_T pairing on supersingular curves in characteristic two, where $a(x)$ and $b(x)$ are secret and are related to the secret value x_P, and $r(x)$ is public and known to, or even chosen by, the attacker. To ease the explanation we consider the simplified Hamming weight model for power leakage [31]. In this model, power consumption depends on the Hamming weight of the data being processed. Thus, we can express the power consumption W as

$$W = \varepsilon H + n,$$

where H, ε, and n represent the Hamming weight of the intermediate data, the incremental amount of power for each extra 1 in the Hamming weight, and the noise, respectively. We assume that the average of noise n is zero.

A. Attack on the Addition

The addition of $b(x)+r(x)$ is of the form $(b_{m-1} \oplus r_{m-1})x^{m-1}+(b_{m-2} \oplus r_{m-2})x^{m-2} + \cdots + (b_1 \oplus r_1)x + (b_0 \oplus r_0)$.

Let W be the power consumption associated with the addition operation $b(x)+r(x)$. To recover the i-th bit of $b(x)$, we guess that $b_i=0$ and divide power consumptions into two sets by r_i.

$$S_k = \{ W \mid r_i = k \} \text{with } k \in \{0,1\}$$

Thus, the differential power consumption is

$$\Delta = <S_1 - S_0>.$$

If the guess is correct, then the averages of S_1 and S_0 are, $\varepsilon(M+1)/2$ and $\varepsilon(M-1)/2$, where M is the size of the register. Thus, if $\Delta > 0$, we know that $b_i = 0$; otherwise, the averages of S_1 and S_0 is $\varepsilon(M-1)/2$ and $\varepsilon(M+1)/2$. Thus, if $\Delta < 0$ then $b_i = 1$. There should be a positive signal when $b_i = 0$ and a negative signal when $b_i = 1$.

In summary, since the addition operation (x_P+x_Q) of step 5 in algorithm 1 is vulnerable to the previous analysis, we can recover x_P. From this value, we can obtain the x- and y-coordinates of the secret point P.

B. Attack on the Multiplication

Since DPA attacks use correlation between power consumptions and specific key-dependent intermediate values that appear during computation related with the secret, it is important to examine the intermediate values appearing in algorithm 2 for computing the multiplication of $a(x)$ and $(b(x)+r(x))$. We will treat two cases: one in which $a(x)$ is a multiplier and $b(x)+r(x)$ is a multiplicand and one in which the opposite is true.

Theorem 1. In algorithm 2, if $a(x)$ is a multiplier and $b(x)+r(x)$ is a multiplicand, then the algorithm is vulnerable to the DPA attack.

Proof. In this case, the multiplication is performed as follows:

$$a_{m-1}x^{m-1}(b(x)+r(x)) + \cdots + a_1x(b(x)+r(x)) + a_0(b(x)+r(x)).$$

In describing how to recover $a(x)$, assume the lowest bits $a_{i-1},\cdots, a_0$ of the secret multiplier $a(x)$ are already recovered. We describe how to find the next bit a_i.

In algorithm 2, the intermediate value obtained at the end of i-th step of the loop is

$$a_ix^i(b(x)+r(x)) + \cdots + a_1x(b(x)+r(x)) + a_0(b(x)+r(x)). \quad (3)$$

The i-th bit of (3) is $a_i(b_0+r_0) + \cdots + a_1(b_{i-1}+r_{i-1}) + a_0(b_i+r_i)$. By rearranging, we have

$$(a_ib_0 + \cdots + a_1b_{i-1} + a_0b_i) + (a_ir_0 + \cdots + a_1r_{i-1} + a_0r_i). \quad (4)$$

In this case, we cannot simulate this intermediate value because we do not know and control the value of $a_ib_0 + a_{i-1}b_1 + \cdots + a_0b_i$. However, since the value is fixed at 0 or 1 but

unknown, we can accomplish a DPA attack by only controlling the input value $r(x)$, not the intermediate value.

Let W be the power consumption associated with computation of $a(x)(b(x)+r(x))$. From (4) we guess that $a_i=1$ and divide power consumptions into two sets by $a_i r_0 + a_{i-1} r_1 + \cdots + a_0 r_i$, which is derived from the already known values $a_{i-1},\cdots, a_0$ and the random value r chosen by the attacker.

$$S_k = \{ W \mid r_0 + a_{i-1} r_1 + \cdots + a_0 r_i = k \} \text{ with } k \in \{0,1\}$$

If $a_i b_0 + a_{i-1} b_1 + \cdots + a_0 b_i = 1$ in (4) and the guess is correct, then the average power consumptions of S_0 and S_1 are, respectively, $\varepsilon(M+1)/2$ and $\varepsilon(M-1)/2$, where M is the size of the register. Thus, we have $<S_1 - S_0> = -\varepsilon$, that is, the attacker can recognize a negative peak. Otherwise, if the guess is incorrect, then the average power consumptions of S_0 and S_1 are $\varepsilon M/2$, thus, $<S_1 - S_0> \approx 0$. In the case that $a_i b_0 + a_{i-1} b_1 + \cdots + a_0 b_i = 0$ and the guess is correct, the averages of S_0 and S_1 are, respectively, $\varepsilon(M-1)/2$ and $\varepsilon(M+1)/2$. We have $<S_1 - S_0> = \varepsilon$, that is, the attacker can recognize a positive peak; otherwise, if the guess is incorrect, then $<S_1 - S_0> \approx 0$. The difference between the two cases is only whether the biased signal is positive or negative when the guess is correct. Thus, we compute the differential power consumption by using the absolute value

$$\Delta = <|S_1 - S_0|>.$$

If $\Delta \neq 0$, that is, if we can detect an appreciable peak, then the guess is right ($a_i = 1$); otherwise, the guess is wrong ($a_i = 0$). The remaining bits $a_{m-1},\cdots, a_{i+1}$ are recursively recovered in the same way. Thus, we can recover the multiplier $a(x)$ of algorithm 2. □

Theorem 2. In algorithm 2, if $b(x)+r(x)$ is a multiplier and $a(x)$ is a multiplicand, then the algorithm is vulnerable to the DPA attack.

Proof. The multiplication of $(b(x)+r(x))a(x)$ is performed as follows:

$$(b_{m-1}+r_{m-1})x^{m-1}a(x) + \cdots + (b_1+r_1)xa(x) + (b_0+r_0)a(x).$$

In this case, we also describe how to recover $a(x)$. Assume the lowest bits $a_{i-1},\cdots, a_0$ of $a(x)$ are already recovered. The aim is to find the next bit a_i.

In algorithm 2, the intermediate value obtained at end of i-th step of the loop is

$$(b_i+r_i)x^i a(x) + \cdots + (b_1+r_1)xa(x) + (b_0+r_0)a(x). \quad (5)$$

The i-th bit of (5) is $(b_i+r_i)a_0 + \cdots + (b_1+r_1)a_{i-1} + (b_0+r_0)a_i$. By rearranging, we have

$$(b_i a_0 + \cdots + b_1 a_{i-1} + b_0 a_i) + (r_i a_0 + \cdots + r_1 a_{i-1} + r_0 a_i). \quad (6)$$

The previous equation is equal to (4). Thus, we can recover a_i by the proof of theorem 1. □

In summary, since the multiplication of $u(x_P+x_Q)$ of step 5 in algorithm 1 is vulnerable to the proposed attack, we can recover u, the x-coordinate of the secret point P, even though $x_P + x_Q$ is unknown. Finally, we can obtain the y-coordinate from the x-coordinate by solving the quadratic equation of elliptic curves.

IV. Proposed Countermeasure

The attack described in section III is possible when we can choose and control an input value and simulate the intermediate results. To resist such attacks for elliptic curve cryptosystems (ECCs), Coron [26] proposed three methods to securely compute scalar multiplication $[d]P$, where d is the secret key, and P is public. Let E be an elliptic curve over finite fields F_q. Let $\#E$ be the number of points of the curve. The countermeasures for computing $Q = [d]P$ are the following:

1. Randomization of the private value, that is, $[d]P = [(d+r\#E)]P$ for a random number $r \in F_q$.
2. Blinding the public value, that is, $[d]P = [d](P+R) - [d]R$ for a random point R on $E(F_q)$.
3. Randomizing projective coordinates, that is, $(X; Y; Z) = (rX; rY; rZ)$ for a random value $r \neq 0$ in F_q.

In the context of the above techniques, Page and Vercauteren [19] and Scott [21] proposed some methods for the pairings. First, they used bilinearity to randomize the private data, that is, $e(P,Q) = e([s]P, [t]P)^{1/st}$ where s and t are random variables. Furthermore, the exponentiation to the power $1/st$ can be removed by selecting s and t satisfying $st = 1 \bmod l$, where l is the order of the underlying elliptic curve for the pairing [19]. Second, they presented a method for blinding the input point by the relation $e(P,Q) = e(P,Q+R)e(P,R)^{-1}$ [20].

In [21], Scott introduced a method which multiplies intermediate values appearing during the loop by a random value in F_q. To prevent the DPA attacks described in the previous section, all intermediate variables (not only g) in step 2 and step 5 of algorithm 1 must be multiplied by a random value. Thus, steps 2 and 5 should be respectively changed into $f = u(ru + rx_Q) + ry_P + r(y_Q + b + 1) + (ru + rx_Q)s + rt$ and $g = u(rx_P + rx_Q) + ry_P + ry_Q + rx_P + (ru + rx_Q)s + rt$.

1. Projective Coordinates Randomization

We propose explicit algorithms for the η_T pairing using projective coordinates to resist DPA attacks. To show that it is the fastest method among existing countermeasures, we estimate computational efficiency between the proposed method and previous countermeasures. In [32], they only considered the case of the randomization of public point Q. In this paper, we consider two cases: the randomization of public point P and secret point Q. We will show that the

randomization of Q is faster than that of P as in [32]. We will next describe how to securely compute the η_T pairing using randomized projective coordinates in the case of characteristic two. The following lemma helps to construct a projective coordinate version of the η_T pairing.

Lemma 1. For the order N of a supersingular elliptic curve over F_{2^m} with embedding degree k, one can eliminate all factors lying in F_{2^m} by the power $M = (2^{mk} - 1)/N$.

Proof. Since $F^*_{2^m}$ is a multiplicative subgroup of $F^*_{2^{mk}}$, it follows that $2^m-1|2^{mk}-1$. On the other hand, by the definition of the embedding degree, the order of a supersingular curve with security multiplier $k > 1$ does not divide $2^m - 1$. Therefore, $(2^{mk} -1)/N$ contains a factor $2^m - 1$. □

A *weighted projective* point (X, Y, Z) of an affine point $P=(x,y)$ is given by

$$x = \frac{X}{Z^{\alpha}} \quad \text{and} \quad y = \frac{Y}{Z^{\beta}}.$$

In particular, when $\alpha = \beta = 1$ they are called standard projective coordinates, and when $\alpha = 2$ and $\beta = 3$ they are called Jacobian coordinates.

For projective point $P=(X, Y, Z)$, $[2^i]P = \varphi^i(X^{(2i)}, Y^{(2i)}, Z^{(2i)})$, where $\varphi(X, Y, Z)=(X+Z, Y+X, Z)$.

A. Randomization of P in Weighted Coordinates

First, we consider the randomization of $P = (x_P, y_P)$ in weighted coordinates $(X_P / Z_P^{\alpha}, Y_P / Z_P^{\beta})$.

Theorem 3. For weighted projective coordinates of an affine point $P = (x_P, y_P)$, we can set $x_P = X_P / Z_P^{\alpha}$ and $y_P = Y_P / Z_P^{\beta}$. Then, the function (2), $g_{[2^{-j}]P'}(\psi(Q))^{(j)}$, can be computed with 11 F_{2^m} multiplications as follows. Note that $(Z^{\alpha})^{2^j}$ means $Z^{\alpha(j)}$.

$$g_{[2^{-j}]P'}(\psi(Q))^{(j)} = g_0 + g_1 s + g_2 t,$$

where

$$g_0 = X_P^{(-j)}(x_Q^{(j)} Z_P^{\alpha(-1-j)} + X_P^{(-1-j)}) Z_P^{\beta(-1-j)} + (Y_P^{(-1-j)} Z_P^{\alpha(-1-j)} + X_P^{(-1-j)} Z_P^{\beta(-1-j)} + y_Q^{(j)} Z_P^{\alpha(-1-j)} Z_P^{\beta(-1-j)}) Z_P^{\alpha(-j)},$$

$$g_1 = Z_P^{\alpha(-1-j)} Z_P^{\beta(-1-j)} (X_P^{(-j)} + x_Q^{(j)} Z_P^{\alpha(-j)}),$$

$$g_2 = Z_P^{\alpha(-j)} Z_P^{\alpha(-1-j)} Z_P^{\beta(-1-j)}.$$

Proof. In (2), substituting x_P and y_P by X_P / Z_P^{α} and Y_P / Z_P^{β} and using the preceding notations, we have

$$g_{[2^{-j}]P'}(\psi(Q))^{(j)} = \frac{1}{Z_P^{\alpha(-j)} Z_P^{\alpha(-1-j)} Z_P^{\beta(-1-j)}} g_0 + g_1 s + g_2 t.$$

Since $Z_P^{\alpha(-j)}$, $Z_P^{\alpha(-1-j)}$, and $Z_P^{\beta(-1-j)}$ are in F_{2^m}, the denominator becomes 1 after the final exponentiation from lemma 1. Thus, elimination of $1/Z_P^{\alpha(-j)} Z_P^{\alpha(-1-j)} Z_P^{\beta(-1-j)}$ does not effect the result, so this term can be eliminated. Then, g_0, g_1, and g_2 require 8, 2, and 1 multiplication(s) in F_{2^m}, respectively. □

Corollary 1. For $x_P = X_P / Z_P$ and $y_P = Y_P / Z_P$, $g_{[2^{-j}]P'}(\psi(Q))^{(j)}$ can be computed with 7 F_{2^m} multiplications as

$$g_{[2^{-j}]P'}(\psi(Q))^{(j)} = g_0 + g_1 s + g_2 t,$$

where

$$g_0 = X_P^{(-j)}(x_Q^{(j)} Z_P^{(-1-j)} + X_P^{(-1-j)}) + (Y_P^{(-1-j)} + X_P^{(-1-j)} + y_Q^{(j)} Z_P^{(-1-j)}) Z_P^{(-j)},$$

$$g_1 = Z_P^{(-1-j)}(X_P^{(-j)} + x_Q^{(j)} Z_P^{(-j)}),$$

$$g_2 = Z_P^{(-j)} Z_P^{(-1-j)}.$$

Proof. In theorem 3, if $\alpha = \beta = 1$, then the equation of theorem 3 is simplified. With the preceding notations,

$$g_{[2^{-j}]P'}(\psi(Q))^{(j)} = \frac{1}{Z_P^{(-j)} Z_P^{(-1-j)}} g_0 + g_1 s + g_2 t.$$

Since $Z_P^{(-j)}$ and $Z_P^{(-1-j)}$ are in F_{2^m}, the denominator becomes 1 after the final exponentiation from lemma 1. Thus, elimination of $1/Z_P^{(-j)} Z_P^{(-1-j)}$ does not effect the result, so this term can be eliminated. Thus, g_0, g_1, and g_2 require 4, 2, and 1 multiplication(s) in F_{2^m}, respectively. □

We illustrate an explicit algorithm of corollary 1 in algorithm 3.

Algorithm 3. $\eta_T(P,Q)$ on the elliptic curve E: $Y^2Z + YZ^2 = X^3 + XZ^2 + bZ^3$ over F_{2^m} where $m \equiv 3 \bmod 8$ and $b \in F_2$ via randomized projective coordinates of P

Input: $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.
Output: $\eta_T(P,Q)$.
1. $(X_P, Y_P, Z_P) = (\lambda x_P, \lambda y_P, \lambda)$ where λ is a random element of F_{2^m}
2. $u = X_P + Z_P$
3. $f = u(u+Z_P x_Q) + (Y_P+(y_Q + b + 1)Z_P)Z_P + Z_P(u+x_Q Z_P)s + Z_P^2 t$
4. For i=0 to $(m - 1)/2$ do
5. $u = X_P$, $v = Z_P$, $X_P = X_P^{1/2}$, $Y_P = Y_P^{1/2}$, $Z_P = Z_P^{1/2}$
6. $g = u(X_P + x_Q Z_P) + (Y_P + y_Q Z_P + X_P)v + (u + x_Q v)s + v Z_P t$
7. $f = fg$
8. $x_Q = x_Q^2$, $y_Q = y_Q^2$
9. Return $f^{(2^{2m}-1)(2^m - 2^{(m+1)/2}+1)}$

B. Randomization of Q in Weighted Coordinates

Next, we consider the randomization of $Q=(x_Q, y_Q)$ in weighted coordinates $(X_Q / Z_Q^{\alpha}, Y_Q / Z_Q^{\beta})$.

For weighted projective coordinates of an affine point $Q = (x_Q, y_Q)$, we can set $x_Q = X_Q / Z_Q^{\alpha}$ and $y_Q = Y_Q / Z_Q^{\beta}$. Then, the function (2), $g_{[2^{-j}]P}(\psi(Q))^{(j)}$ can be computed with 8 F_{2^m} multiplications as

$$g_{[2^{-j}]P}(\psi(Q))^{(j)} = g_0 + g_1 s + g_2 t,$$

where

$$g_0 = x_P^{(-j)}(x_P^{(-1-j)} Z_Q^{\alpha(j)} + X_Q^{(j)}) Z_Q^{\beta(j)} + ((y_P^{(-1-j)} + x_P^{(-1-j)}) Z_Q^{\alpha(j)} + Y_Q^{(j)}) Z_Q^{\beta(j)},$$

$$g_1 = (x_P^{(-j)} + X_Q^{(j)} Z_Q^{\alpha(j)}) Z_Q^{\beta(j)},$$

$$g_2 = Z_Q^{\alpha(j)} Z_Q^{\beta(j)}.$$

Proof. In (2), substituting x_Q and y_Q by X_Q / Z_Q^{α} and Y_Q / Z_Q^{β} and using the preceding notations, we have

$$g_{[2^{-j}]P}(\psi(Q))^{(j)} = \frac{1}{Z_Q^{\alpha(j)} Z_Q^{\beta(j)}} g_0 + g_1 s + g_2 t.$$

As in theorem 3, since $Z_Q^{\alpha(j)}$ and $Z_Q^{\beta(j)}$ are in F_{2^m}, the denominator becomes 1 after the final exponentiation. Thus, g_0, g_1, and g_2 require 5, 2, and 1 multiplication(s) in F_{2^m}, respectively. □

Corollary 2. For $x_Q = X_Q / Z_Q$ and $y_Q = Y_Q / Z_Q$, $g_{[2^{-j}]P}(\psi(Q))^{(j)}$ can be computed with 4 F_{2^m} multiplications as

$$g_{[2^{-j}]P}(\psi(Q))^{(j)} = g_0 + g_1 s + g_2 t,$$

where

$$g_0 = x_P^{(-j)}(x_P^{(-1-j)} Z_Q^{(j)} + X_Q^{(j)}) + (y_P^{(-1-j)} + x_P^{(-1-j)}) Z_Q^{(j)} + Y_Q^{(j)},$$

$$g_1 = x_P^{(-j)} + X_Q^{(j)} Z_Q^{(j)},$$

$$g_2 = Z_Q^{(j)}.$$

Proof. In theorem 4, if $\alpha = \beta = 1$, then the equation of theorem 4 is simplified. With the preceding notations,

$$g_{[2^{-j}]P}(\psi(Q))^{(j)} = \frac{1}{Z_Q^{(j)}} g_0 + g_1 s + g_2 t.$$

Since $Z_Q^{(j)}$ is in F_{2^m}, the denominator becomes 1 after the final exponentiation. Thus, g_0 and g_1 require 3 and 1 multiplication(s) in F_{2^m}, respectively. □

We illustrate an explicit algorithm of corollary 2 in algorithm 4.

Algorithm 4. $\eta_T(P,Q)$ on the elliptic curve E: $Y^2Z + YZ^2 = X^3 + XZ^2 + bZ^3$ over F_{2^m} where $m \equiv 3 \bmod 8$ and $b \in F_2$ via randomized projective coordinates of Q

Input: $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.

Output: $\eta_T(P,Q)$.

1. $(X_Q, Y_Q, Z_Q) = (\lambda x_Q, \lambda y_Q, \lambda)$ where λ is a random element of F_{2^m}
2. $u = x_P + 1$
3. $f = u(uZ_Q + X_Q) + Z_Q(x_P + b + 1) + Y_Q + (uZ_Q + X_Q)s + Z_Q t$
4. For i=0 to $(m - 1)/2$ do
5. $u = x_P$, $x_P = x_P^{1/2}$, $y_P = y_P^{1/2}$
6. $g = u(Z_Q x_P + X_Q) + (y_P + x_P)Z_Q + Y_Q + (uZ_Q + X_Q)Z_P s + Z_Q t$
7. $f = fg$
8. $X_Q = X_Q^2$, $Y_Q = Y_Q^2$, $Z_Q = Z_Q^2$
9. Return $f^{(2^{2m}-1)(2^m - 2^{(m+1)/2}+1)}$

2. Security Analysis

In this section, we discuss the security of the proposed algorithms against SPA, DPA, and the refined power analysis (RPA).

A. Security against SPA

The SPA of section III.3 retrieves the operation sequence by detecting the conditional branch depending on the secret parameter. Thus, a simple way to avoid SPAs is for the multiplier to use public values known to or even chosen by attackers and for the multiplicand to use secret values.

B. Security against DPA

Since DPAs exploit the correlation between intermediate results and secret parameters, the target of a DPA is intermediate data. The problem of the original η_T pairing is that the secret values are fixed values even though attackers do not know or predict the values. To overcome this weakness, we change the secret value or the public value using randomized projective coordinates at each execution.

In algorithm 3, the secret values are blinded by a random element λ in F_{2^m} at each new execution of the algorithm, so the grouping depending on the secret value in DPA is meaningless. More precisely, we check the intermediate values that appear during the process of algorithm 3. In step 3, the computation in relation to the secret values is as follows:

i. $(X_P + Z_P)(X_P + Z_P + x_Q Z_P) = (x_P\lambda + \lambda)(x_P\lambda + \lambda + x_Q\lambda)$
ii. $(Y_P + (y_Q + b + 1)Z_P)Z_P = (y_P\lambda + (y_Q + b + 1)\lambda)\lambda$
iii. $Z_P(X_P + Z_P + x_Q Z_P) = \lambda(x_P\lambda + \lambda + x_Q\lambda)$

In step 6 of the first loop, that is, the case of $i = 0$, the computation in relation to the secret values is as follows:

iv. $X_P(X_P^{1/2} + x_Q Z_P^{1/2}) = x_P\lambda(x_P^{1/2}\lambda^{1/2} + x_Q\lambda^{1/2})$
v. $(Y_P^{1/2}+y_Q Z_P^{1/2}+X_P^{1/2})Z_P=(x_P^{1/2}\lambda^{1/2}+y_Q\lambda^{1/2}+x_P^{1/2}\lambda^{1/2})\lambda$
vi. $(X_P + x_Q Z_P)Z_P^{1/2} = (x_P\lambda + x_Q\lambda)\lambda^{1/2}$

In the preceding six computations, all intermediate variables are blinded by the random element λ. Therefore, algorithm 3 is secure against DPA.

In algorithm 4 the public values known to or chosen by the attacker are randomized and are blinded by a random element λ in F_{2^m} at each new execution of the algorithm. Therefore, it is not possible for the attacker to predict any specific bit of the intermediate value. More precisely, we check the intermediate values that appear during the process of algorithm 4. In step 3, the computation in relation to the secret values is as follows:

i. $(x_P + 1)((x_P + 1)Z_Q + X_Q) = (x_P + 1)((x_P + 1)\lambda + x_Q\lambda)$
ii. $Y_Q + (x_P + b + 1)Z_Q = y_Q\lambda + (x_P + b + 1)\lambda$
iii. $(x_P + 1)Z_Q + X_Q = (x_P + 1)\lambda + x_Q\lambda$

In step 6 of the first loop, that is, the case of $i = 0$, the computation in relation to the secret values is as follows:

vi. $x_P(Z_Q x_P^{1/2} + X_Q) = x_P(x_P^{1/2}\lambda + x_Q\lambda)$
v. $(y_P + x_P)Z_Q + Y_Q = (y_P + x_P)\lambda + y_Q\lambda$
vi. $x_P Z_Q + X_Q = x_P\lambda + X_Q\lambda$

In the preceding six computations, intermediate variables except for the first computation and the fourth computation variables are blinded by the random element λ. However, one variable of the first and fourth computations is not blinded. The computation of the first and fourth computations is formulated as

$$a(x)(b(x)\lambda(x) + r(x)\lambda(x)),$$

where $a(x)$ and $b(x)$ are secret values, $r(x)$ is a public value known to or even chosen by attackers, and $\lambda(x)$ is a random element in F_{2^m}.

Theorem 5. With the above notation, the computation of $a(x)(b(x)\lambda(x) + r(x)\lambda(x))$ is secure against the DPA attack if $\lambda(x)$ is uniformly distributed on $[0, 2^{m-1}]$.

Proof. Without loss of generality, we assume that $a(x)$ is a multiplier and $(b(x)\lambda(x) + r(x)\lambda(x))$ is a multiplicand. To ease the description, we assume that $b(x)\lambda(x)$ and $r(x)\lambda(x)$ are the results of only polynomial multiplication. However, if the reduction is performed, then the attack could be more difficult.

Similar to the proof of theorem 1, the intermediate value obtained at the end of the i-th step of the loop in the multiplication algorithm, that is, algorithm 2, is

$$a_i x^i(b(x)\lambda(x) + r(x)\lambda(x)) + \cdots + a_0(b(x)\lambda(x) + r(x)\lambda(x)).$$

The i-th bit of the above value is

$a_i(b_0\lambda_0 + r_0\lambda_0) + a_{i-1}(b_0\lambda_1 + b_1\lambda_0 + r_0\lambda_1 + r_1\lambda_0) + \cdots + a_0(b_0\lambda_i + \cdots + b_i\lambda_0 + r_0\lambda_i \cdots + r_i\lambda_0)$.

By rearranging, we have

$$[(a_i b_0 + \cdots + a_0 b_i) + (a_i r_0 + \cdots + a_0 r_i)]\lambda_0 + [(a_{i-1}b_0 + \cdots + a_0 b_{i-1}) + (a_{i-1}r_0 + \cdots + a_0 r_{i-1})]\lambda_1 + \cdots + (a_0 b_0 + a_0 r_0)\lambda_i. \quad (7)$$

The probability of correctly guessing for (7) is $1/2^i$ if $\lambda_0, \cdots, \lambda_i$ are uniformly distributed, even if the guess for a_i is correct, as in the proof of theorem 1. Moreover, if $b(x)\lambda(x)$ and $r(x)\lambda(x)$ are the results of the reduction, then all bits $\lambda_{m-1}, \cdots, \lambda_0$ of $\lambda(x)$ could be connected with the target intermediate value. In this case, the probability of correctly guessing for the target intermediate value is roughly $1/2^m$ if $\lambda(x)$ is uniformly distributed on $[0, 2^{m-1}]$. Thus, the intermediate values appear to be random. The grouping depending on $r(x)$ known to or chosen by the attacker, as in the proof of theorem 1, is meaningless.

The case in which $(b(x)\lambda(x) + r(x)\lambda(x))$ is a multiplier and $a(x)$ is a multiplicand has the same intermediate value as the case in which $a(x)$ is a multiplier and $(b(x)\lambda(x) + r(x)\lambda(x))$ is a multiplicand. Therefore, the computation is secure against the DPA attack. □

By theorem 5, the first computation and the fourth computation are also secure against DPA attack. Thus, we can conclude that algorithms 3 and 4 are secure against DPA attack.

C. Security against RPA

In ECC, the randomized projective coordinate is vulnerable to RPA [33] and zero-value point attack (ZVPA) [34]. The RPA detects whether a "special point," such as $(x,0)$ or $(0,y)$ appears during a scalar multiplication. The appearance of a special point depends on the secret scalar. For example, if we use the base point $P = (d^{-1} \bmod \#E)(x,0)$ for some secret integer d, then the RPA can successfully detect the appearance of the special point $(x,0)$ during the scalar multiplication dP when the guess for the secret scalar is correct.

In the field multiplication, we have to find a special element in F_{2^m} having properties similar to those of the special point. Then, we use $b = a^{-1}\gamma$ for some secret value a, where γ is the special element. However, when the secret value a is randomized by a random element λ, that is, $a' = a\lambda$, the intermediate value during the multiplication of a' and b is $a\lambda a^{-1}\gamma = \lambda\gamma$. We cannot detect the appearance of the special element γ. On the other hand, when the selected value is randomized by a random element λ, that is, $b' = a^{-1}\gamma\lambda$, the intermediate value during the multiplication of a and b' is $aa^{-1}\gamma\lambda = \gamma\lambda$. Therefore, the RPA-like attack does not defeat the multiplication with randomized inputs.

V. Efficiency Comparison

We first estimate the computational cost of the proposed

algorithms. From now on, we denote the computation time of a squaring, a multiplication and an inversion in $F_{2^{km}}$ as S_k, M_k, and I_k. In particular, we denote the computation time of a squaring, a multiplication and an inversion in F_{2^m} as S, M, and I, that is, $S_1 = S$, $M_1 = M$, and $I_1 = I$. Moreover, we denote the computation time of a square root as S because the method described in [35] for computing square roots in F_{2^m} is as fast as squaring. Because addition in F_{2^m} is relatively inexpensive compared to multiplication and inversion, we can ignore the cost of field additions [36].

Because the extension field $F_{2^{4m}}$ is represented by the basis $\{1, s, t, st\}$ such that $t \in F_{2^4}$ satisfies $t^2 + t + s = 0$ and $s \in F_{2^2}$ satisfies $s^2 + s + 1 = 0$, M_4 is performed by $16M$. However, every element in $F_{2^{4m}}$ can be represented by elements in $F_{2^{2m}}$ with the basis $\{1, t\}$ because $F_{2^{4m}}$ is a quadratic extension of $F_{2^{2m}}$. Let $G_0 = g_0 + g_1 s$, $G_1 = g_2 + g_3 s$, $F_0 = f_0 + f_1 s$, and $F_1 = g_2 + g_3 s$. Then g and f are represented as $g = G_0 + G_1 t = g_0 + g_1 s + g_2 t + g_3 st$ and $f = F_0 + F_1 t = f_0 + f_1 s + f_2 t + f_3 st$, respectively. If we apply the Karatsuba multiplication technique, M_4 is performed in $3M_2$ as

$$G_0F_0 + ((G_0 + G_1)(F_0 + F_1) + G_0F_0 + G_1F_1)t + G_1F_1t^2.$$

Because $F_{2^{2m}}$ is also a quadratic extension of F_{2^m}, if we repeatedly apply the Kararsuba technique, M_2 is performed in $3M$. Finally, we can compute $h = fg = (h_0, h_1, h_2, h_3)$ by $9M$.

$$\begin{aligned}
h_0 &= f_0g_0 + f_1g_1 + f_3g_2 + f_2g_3 + f_3g_3 \\
&= f_0g_0 + f_1g_1 + (f_2 + f_3)(g_2 + g_3) + f_2g_2, \\
h_1 &= f_1g_0 + f_0g_1 + f_1g_1 + f_2g_2 + f_3g_2 + f_2g_3 \\
&= (f_0 + f_1)(g_0 + g_1) + f_0g_0 + (f_2 + f_3)(g_2 + g_3) + f_3g_3, \\
h_2 &= f_2g_0 + f_3g_1 + f_0g_2 + f_2g_2 + f_1g_3 + f_3g_3 \\
&= (f_0 + f_2)(g_0 + g_2) + (f_1 + f_3)(g_1 + g_3) + f_0g_0 + f_1g_1, \\
h_3 &= f_3g_0 + f_2g_1 + f_3g_1 + f_1g_2 + f_3g_2 + f_0g_3 + f_1g_3 + f_2g_3 + f_3g_3 \\
&= (f_0 + f_1 + f_2 + f_3)(g_0 + g_1 + g_2 + g_3) + (f_0 + f_2)(g_0 + g_2) \\
&\quad + (f_0 + f_1)(g_0 + g_1) + f_0g_0.
\end{aligned}$$

In algorithm 1, since $g_2 = g_3 = 0$, the preceding formula is simplified as follows and is performed by $6M$ [12]:

$$\begin{aligned}
h_0 &= f_0g_0 + f_1g_1, \\
h_1 &= (f_0 + f_1)(g_0 + g_1) + f_0g_0, \\
h_2 &= (f_0 + f_2)g_0 + (f_1 + f_3)g_1 + f_0g_0 + f_1g_1, \\
h_3 &= (f_0 + f_1 + f_2 + f_3)(g_0 + g_1) + (f_0 + f_2)g_0 + (f_0 + f_1)(g_0 + g_1) \\
&\quad + f_0g_0.
\end{aligned}$$

In the original η_T pairing, the initial step requires $1M$, and each loop requires $7M + 4S$, where $1M$ is used to compute g and $6M$ at step 6 because of the sparse form of g, in addition to two square roots at step 4 and two squarings at step 7. Thus, since the number of iterations is $(m+1)/2$, the total cost of the original η_T pairing is $(7M + 4S)(m+1)/2 + 1M$, that is, $(3.5M + 2S)(m + 1) + 1M$.

However, algorithm 3 requires $7M$ at the initial stage, $2M$ at step 1, $5M$ at step 3, $6M$ at step 6, and $8M$ at step 7. Algorithm 4 requires $5M$ at the initial step, $2M$ at step 1,$3M$ at step 3, $4M$ at step 6, and $8M$ at step 7. In both the proposed algorithms, since $g_3 = 0$, the formula is simplified as

$$\begin{aligned}
h_0 &= f_0g_0 + f_1g_1 + (f_2 + f_3)g_2 + f_2g_2, \\
h_1 &= (f_0 + f_1)(g_0 + g_1) + f_0g_0 + (f_2 + f_3)g_2, \\
h_2 &= (f_0 + f_2)(g_0 + g_2) + (f_1 + f_3)g_1 + f_0g_0 + f_1g_1, \\
h_3 &= (f_0 + f_1 + f_2 + f_3)(g_0 + g_1 + g_2) + (f_0 + f_2)(g_0 + g_2) \\
&\quad + (f_0 + f_1)(g_0 + g_1) + f_0g_0.
\end{aligned}$$

The main loop of both proposed algorithms also requires $5S$. The cost of the main loop of algorithms 3 and 4 is $(14M + 5S)(m+1)/2$ and $(12M + 5S)(m+1)/2$, respectively. Therefore, the total costs of algorithms 3 and 4 are $(14M + 5S)(m+1)/2 + 7M$, that is, $(7M + 2.5S)(m + 1) + 7M$ and $(12M + 5S)(m+1)/2 + 5M$, that is, $(6M + 2.5S)(m + 1) + 5M$, respectively. The total cost of the η_T pairing (except for the final exponentiation) is shown in Table 1.

Thus, we can deduce that algorithm 4 is more efficient than algorithm 3. The additional cost of algorithm 4 over the original η_T pairing, including the initial step, is $(2.5M + 0.5S)(m + 1) + 4M$.

We now compare computational efficiency of the techniques described in section IV. First, the method using bilinearity additionally requires 2 scalar multiplications. In supersingular curves, doubling a point requires $4S$, and adding two distinct points requires $1S + 2M + 1I$ [27]. If we use the binary method for scalar multiplication, the cost of two scalar multiplications is $2m$ elliptic curve doublings and m elliptic curve additions on average. The additional cost of this method is approximately $(2M + 9S + I)m$. Thus, when $I < 0.5M$ this method is more efficient than the proposed method. It is practically infeasible. In general, the ratio of the fastest inversion method to the fastest multiplication is approximately 10 to 1, that is, $I = 10M$ [36]. In this case, the additional cost of this method is approximately $(12M + 9S)m$.

Second, the method of $e(P,Q) = e(P,Q+R)e(P,R)^{-1}$ additionally requires 1 pairing computation, $1M_4$, and $1I_4$. The computational cost of the η_T pairing, that is, algorithm 1, is

Table 1. Total computational cost of the η_T pairing (except for the final exponentiation) on supersingular curves over F_{2^m}.

Proposed algorithms	Computational cost
Algorithm 1 (original η_T pairing)	$(3.5M + 2S)(m + 1) + 1M$
Algorithm 3 (randomization P)	$(7M + 2.5S)(m + 1) + 7M$
Algorithm 4 (randomization Q)	$(6M + 2.5S)(m + 1) + 5M$

approximately $(3.5M + 2S)(m + 1) + 1M$ plus the final exponentiation. In [37], Shirase and others proposed an efficient method for computing the final exponentiation f^W, where $W = (2^{2m} - 1)(2^m - 2^{(m+1)/2} + 1)$. Let f be represented as $\{F_0 + F_1 t : F_0, F_1 \in F_{2^{2m}}\}$. Then, f^W is performed as

$$\left(f^{2^{2m}-1}\right)^{2^m+1}\left(\left(f^{2^{2m}-1}\right)^{2^{(m+1)/2}}\right)^{-1} \quad (8)$$

Since the conjugate of $F_0 + F_1 t$ is $(F_0 + F_1) + F_1 t$, $f^{2^{2m}-1}$ is computed by $5M_2 + I_2$.

$$f^{2^{2m}-1} = \frac{f^{2^{2m}}}{f} = \frac{(F_0 + F_1 t)^{2^{2m}}}{F_0 + F_1 t} = \frac{(F_0 + F_1) + F_1 t}{F_0 + F_1 t}$$
$$= \frac{((F_0 + F_1) + F_1 t)^2}{(F_0 + F_1 t)((F_0 + F_1) + F_1 t)}$$
$$= \frac{F_0^2 + F_1^2(s+1) + F_1^2 t}{F_0^2 + F_0 F_1 + F_1^2 s}.$$

Let $H = f^{2^{2m}-1}$. In (7), $H^{2^m+1} = H^{2^m} \cdot H$ and $H^{2^{(m+1)/2}}$ are computed by M_4 and $(m+1)/2C_4$, respectively. As mentioned above, M_4 requires $9M$. An inversion of $A = A_0 + A_1 t \in F_{2^{4m}}$ (where $A_i \in F_{2^{2m}}$) can be computed with $3M_2 + 1I_2$ as

$$\frac{1}{(A_0 + A_1 t)} = \frac{(A_0 + A_1) + A_1 t}{(A_0 + A_1 t)((A_0 + A_1) + A_1 t)} = \frac{(A_0 + A_1) + A_1 t}{A_0^2 + A_0 A_1 + A_1^2 s}.$$

Similarly, I_2 can be computed with $3M$ and $1I$. Since $M_2 = 3M$ and $I = 10M$, I_4 and I_2 can be computed by $22M$ and $13M$. The computational cost of the final exponentiation is $2M_4 + 5M_2 + (m+1)/2C_4 + I_4 + I_2 = 68M + 2(m+1)C$. Therefore, the additional cost of this method is $(3.5M + 4C)(m + 1) + 69M$.

In Scott's method [21], that is, randomization of intermediate variables, the additional cost over the η_T pairing is $3.5M(m + 1) + 4M$. We give a comparison table of the number of operations among existing techniques in Table 2.

Table 2. Additional cost of countermeasures for DPA resistance over the η_T pairing (not over the Tate pairing) on supersingular curves over F_{2^m}.

Countermeasures	Additional Cost
Page-Vercauteren [29] (randomized private value)	$(2M + 9S + I)m$, $(12M + 9S)m$, when I=10M
Page-Vercauteren [29] (blinding public value)	$(3.5M + 4C)(m + 1) + 69M$ (when I=10M)
Scott [34] (randomizing intermediate value)	$3.5M(m + 1) + 4M$
The Proposed Method (algorithm 4)	$(2.5M + 0.5S)(m + 1) + 4M$

VI. Conclusion

In this paper, we investigated the security of the η_T pairing on supersingular curves over characteristic two against timing attack, SPA and DPA. To avoid such attacks we proposed explicit algorithms of the η_T pairing using randomized projective coordinates. We demonstrated that the proposed method is the most efficient countermeasure compared with previous techniques. We further demonstrated that the proposed algorithms are secure against SPA, DPA, and RPA. However, the proposed algorithms could be susceptible to higher-order DPA attacks. Thus, a direction for further research would be to investigate security against higher-order DPA.

References

[1] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," *SIAM J. of Computing*, vol. 32, no. 3, 2003, pp. 586-615.

[2] R. Sakai and M. Kasahara, "ID Based Cryptosystems with Pairing on Elliptic Curve," Cryptography ePrint Archive, Report 2003/054, 2003. http://eprint.iacr.org/2003/054.

[3] J.C. Cha and J.H. Cheon, "An Indentity-Based Signature from Gap Diffie-Hellman Groups," *PKC*, LNCS 2567, 2003, pp. 18-30.

[4] F. Hess, "Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairing," *SAC*, LNCS 2595, 2002, pp. 310-324.

[5] K.G. Paterson, "ID-Based Signature from Pairings on Elliptic Curves," *Electronics Letters*, vol. 38, no. 18, 2002, pp. 1025-1026.

[6] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Asiacrypt 2001*, LNCS 2248, 2002, pp. 514-532.

[7] N.P. Smart, "An Identity Based Authentication Key Agreement Protocol Based on Pairing," *Electronics Letters*, vol. 38, no. 13, 2002, pp. 630-632.

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

[9] S.D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate Pairing," *ANTS V*, LNCS 2369, 2002, pp. 324-337.

[10] I. Duursma and H.-S. Lee, "Tate Pairing Implementation for Hyperelliptic curves $y^2 = x^p - x + d$," *Asiacrypt*, LNCS 2894, 2003, pp. 111-123.

[11] S. Kwon, "Efficient Tate Pairing Computation for Elliptic Curves over Binary Fields," *ACISP*, LNCS 3574, 2005, pp. 134-145.

[12] P.S.L.M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," *Designs, Codes, and Cryptography*, vol. 42, no. 3, 2007, pp. 239 - 271.

[13] G. Bertoni, L. Breveglieri, P. Fragneto, and G. Pelosi, "Parallel Hardware Architectures for the Cryptographic Tate Pairing," *Proc. the Third Int'l Conf. Information Technology: New Generations (ITNG)*, 2006, pp. 186-191.

[14] G.M. Bertoni, L. Chen, P. Fragneto, K.A. Harrison, and G. Pelosi, "Computing Tate Pairing on Smartcards," 2005. http://www.st.com/stonline/products/families/smartcard/ches2005_v4.pdf

[15] Gemplus, "ID based Cryptography and Smartcards," 2005. http://www.gemplus.com/smart/rd/publications/pdf/Joy05iden.pdf.

[16] R. Ronan, C. Ó hÉigeartaigh, C. Murphy, M. Scott, T. Kerins, and W. Marnane, "An Embedded Processor for a Pairing-Based Cryptosystem," *Proc. the Third Int'l Conf. Information Technology: New Generations (ITNG)*, 2006, pp. 192-197.

[17] M. Scott, N. Costigan, and W. Abdulwahab, "Implemeting Cryptographic Pairings on Smartcards," Cryptography ePrint Archive, Report 2006/144, 2006. http://eprint.iacr.org/2006/144.

[18] R. Dutta, R. Barua, and P. Sarkar, "Pairing-Based Cryptographic Protocols: A Survey," Cryptology ePrint Archive, Report 2004/064, 2006. http://eprint.iacr.org/2004/064.

[19] D. Page and F. Vercauteren, "Fault and Side-Channel Attacks on Pairing Based Cryptography," Cryptology ePrint Archive, Report 2004/283, 2004. http://eprint.iacr.org/2004/283.

[20] D. Page and F. Vercauteren, "A Fault Attack on Pairing Based Cryptography," *IEEE Transactions on Computers*, vol. 55, no.9, 2006, pp. 1075-1080.

[21] M. Scott, "Computing the Tate Pairing," *CT-RSA 2005*, LNCS 3376, 2005, pp. 293-304.

[22] C. Whelan and M. Scott, "Side Channel Analysis of Practical Pairing Implementations: Which Path is More Secure?," *VIETCRYPT,* LNCS 4341, 2006, pp. 99-114.

[23] T. Izu and T. Takagi, "Efficient Computations of the Tate Pairing for the Large MOV Degrees," *ICISC 2002*, LNCS 2587, 2003, pp. 283-297.

[24] S. Chatterjee, P. Sarkar, and R. Barua, "Efficient Computation of Tate Pairing in Projective Coordinate over General Characteristic Fields," *ICISC 2004*, LNCS 3506, 2005, pp. 168-181.

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

[26] J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *CHES*, LNCS 1717, 1999, pp. 292-302.

[27] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[28] E. Verheul, "Evidence that XTR is More Secure than Supersingular Elliptic Curve Cryptosystems," *Journal of Cryptology*, vol. 17, no. 4, 2004, pp. 277-296.

[29] C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *CRYPTO*, LNCS 1666, 1999, pp. 388-397.

[30] T. Messerges, E. Dabbish, and R. Sloan, "Power Analysis Attacks of Modula Exponentiation in Smartcards," *CHES*, LNCS 1717, 1999, pp. 144-157.

[31] T.S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," *CHES*, LNCS 1965, 2000, pp. 238-251.

[32] T.H. Kim, T. Takagi, D.-G. Han, H.W. Kim, and J. Lim "Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields," *CANS*, LNCS 4301, 2006, pp. 168-181.

[33] L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems," *PKC*, LNCS 2567, 2003, pp. 199-210.

[34] T. Akishita and T. Takagi, "Zero-Value Point Attacks on Elliptic Curve Cryptosystem," *ISC*, LNCS 2851, 2003, pp.218-233.

[35] K. Fong, D. Hankerson, J. López, and A. Menezes, "Field Inversion and Point Halving Revisited," *IEEE Trans. Computers*, vol. 53, no. 8, Aug. 2004, pp. 1047-1059.

[36] D. Hankerson, J. López Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," *CHES*, LNCS 1965, 2000, pp. 1-24.

[37] M. Shirase, T. Takagi, and E. Okamoto, "Some Efficient Algorithms for the Final Exponentiation of η_T Pairing," *ISPEC 2007*, LNCS 4464, pp. 254-268.

Tae Hyun Kim received the BS degree in mathematics from University of Seoul, Seoul, Rep. of Korea, in 2002 and the MS degree in engineering in information security from Korea University, Seoul, Rep. of Korea in 2004. He was a visiting researcher with the School of Systems Information Science in Future University, Hakodate, Japan, from April to September 2006. He is currently a PhD student with Korea University. He is also a researcher with the Center for Information Security Technologies (CIST).

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

Dong-Guk Han received his BS and MS degrees in mathematics from Korea University in 1999 and 2002, respectively. He received his PhD in engineering in information security from Korea University in 2005. He was a post-doctoral researcher with the Future University, Hakodate, Japan. After completing his doctorate, he was an exchange student with the Department of Computer Science and Communication Engineering with Kyushu University, Japan, from April 2004 to March 2005. He has been a senior researcher with Electronics and Telecommunications Research Institute since June 2006. He is a member of KIISC, IEEK, and IACR.

Howon Kim received his BS degree in electronics engineering from Kyungpook National University, Daegu, Rep. of Korea, in 1993, and the MS and PhD degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Rep. of Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied with the COSY group at Ruhr-University, Bochum, Germany. He is currently a senior member of technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of 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 related security issues. He is a member of the IEEE, IEEE Computer Society, and IACR. He is also an editor of the ISO 24791-6

Jongin Lim received the BS, MS, and DS degrees from Korea University, Seoul, Rep. of Korea, in 1980, 1982, and 1986, respectively. He is currently a professor of the Graduate School of Information Management and Security of Korea University and the Dean of the Center for Information Security Technology (CIST). His current research interests include side channel attacks, cryptanalysis, theory of cryptography, and cyber-law.