An Efficient Countermeasure against Side Channel Attacks for Pairing Computation

Masaaki Shirase[1], Tsuyoshi Takagi[1], and Eiji Okamoto[2]

[1] Future University Hakodate, Japan
[2] University of Tsukuba, Japan

Abstract. Pairing-based cryptosystems have been widely researched, and several efficient hardware implementations of pairings have also been proposed. However, side channel attacks (SCAs) are serious attacks on hardware implementations. Whelan *et al.* pointed out that pairings except the η_T pairing might not be vulnerable against SCAs by setting the secret point to the first parameter [25]. This paper deals with SCAs for the η_T pairing over $\mathbb{F}_{3^n}$. To our knowledge, the randomized-projective-coordinate method has the smallest overhead among all countermeasures against SCAs for the η_T pairing. The cost of that overhead is $3nM$, where M is the cost of a multiplication in $\mathbb{F}_{3^n}$. In this paper, we propose another countermeasure based on random value additions $(x_p + \lambda)$ and $(y_p + \lambda)$, where $P = (x_p, y_p)$ is the input point, and λ is a random value in $\mathbb{F}_{3^n}$. The countermeasure using the random value addition was relatively slow in the case of the scalar multiplication of elliptic curve cryptosystems. However, in the case of the η_T pairing, we can construct an efficient countermeasure due to the form of the function $g_P(x, y) = y_p^3 y - (x_p^3 + x - 1)^2$ for a point $P = (x_p, y_p)$. The overhead of our proposed scheme is just $0.5nM$, which is a reduction of more than 75% compared with the randomized-projective-coordinate method.

Keywords: η_T pairing, Tate pairing, side channel attacks, random value addition.

1 Introduction

Pairings over elliptic curves are functions from two points on an elliptic curve to an element over finite fields that exhibit bilinearity and non-degeneracy. Tate pairing is a popular pairing and the Miller algorithm is the first efficient algorithm for computing Tate pairing [18]. Barreto *et al.* improved the Miller algorithm by denominator elimination, which is called the BKLS algorithm in this paper [1]. Moreover, the η_T pairing [2] and Ate pairing [10] are efficient algorithms for computing the Tate pairing for supersingular curves and ordinary curves, respectively. The Ate pairing is a variation of the BKLS algorithm. However, the η_T pairing requires a special algorithm that arises from the function $g_P(x, y) = y_p^3 y - (x_p^3 + x - 1)^2$ for a point $P = (x_p, y_p)$.

There have been many research studies on software and hardware implementations of pairings. Indeed, pairings have implemented on FPGAs [7,12,23,20,3]

L. Chen, Y. Mu, and W. Susilo (Eds.): ISPEC 2008, LNCS 4991, pp. 290–303, 2008.

and smart cards [22]. On the other hand, side channel attacks (SCAs) reveal secret data on hardware devices by monitoring side channel information such as power consumption and timing [15,16]. Therefore, hardware devices executing cryptographic algorithms with secret data need countermeasures against SCAs.

Countermeasures for pairing devices have recently been investigated [21,19,25] [13,26]. Let $e(P,Q)$ be a pairing for two points $P = (x_p, y_p),\ Q = (x_q, y_q)$ on the underlying elliptic curve. Scott proposed the method of randomizing the intermediate value [21]. This method multiplies a random value in the finite field by a Miller variable. The overhead of this method is $3.5nM$ for the η_T pairing over $\mathbb{F}_{3^n}$ according to our estimation. On the other hand, Coron proposed some countermeasures against SCAs on the scalar multiplication over an elliptic curve cryptosystem (ECC) [6]. Page *et al.* applied Coron's countermeasures to the pairing, and proposed two countermeasures [19]. The first countermeasure is the scalar multiplication and bilinearity method that computes $e(\alpha P, \beta Q)^{1/\alpha\beta}$ for randomized integers α and β instead of $e(P,Q)$, and its overhead is about $18nM$ for the η_T pairing over $\mathbb{F}_{3^n}$. The second countermeasure is the point-blinding method that computes $e(P, Q + R)/e(P, R)$ for randomized point R, and its overhead is about $7.5nM$ for the η_T pairing over $\mathbb{F}_{3^n}$. Kim *et al.* evaluated their efficiency for the η_T pairing over $\mathbb{F}_{2^n}$. Moreover, they proposed the randomized-projective-coordinate method [13]. In this method, settings $X_p \leftarrow \lambda x_p$, $Y_p \leftarrow \lambda y_p$, and $Z_p \leftarrow \lambda$ are performed, and its overhead is $3nM$ for the η_T pairing over $\mathbb{F}_{3^n}$. Whelan *et al.* also considered SCAs against pairings [25,26]. They concluded that pairings using the BKLS algorithm (such as Ate pairing) might not be vulnerable against SCAs by setting the secret point P to the first parameter of $e(P,Q)$. However, a countermeasure is needed for the η_T pairing algorithm due to its symmetric structure.

In this paper, we provide an improved η_T pairing algorithm over characteristic three, which is secure against SCAs. The proposed scheme is based on random value additions $(x_p + \lambda)$ and $(y_p + \lambda)$, where $P = (x_p, y_p)$ is the input point and λ is a random value of $\mathbb{F}_{3^n}^*$. The overhead of the proposed countermeasure is just $0.5nM$ for the η_T pairing over $\mathbb{F}_{3^n}$. This method is similar to the randomized linearly transformed coordinate (RLC) method for ECC [11,17] although RLC is a relatively slow countermeasure against SCAs. However, our method is a very efficient countermeasure for the η_T pairing. That is an interesting result. We noted that changing "$r_0 \leftarrow x_p + x_q + d$" to "$r_0 \leftarrow (x_p + \lambda) + (x_q - \lambda) + d$" using a random element λ at Step 5 in Algorithm 1 described in Section 2 fixes r_0. Although we have to randomize the y-coordinate, the resulting cost is not large. Therefore, randomizations "$x_p \leftarrow x_p + \lambda,\ y_p \leftarrow y_p + \lambda,\ x_q \leftarrow x_q - \lambda$, and $y_q \leftarrow y_q - \lambda$" in Algorithm 1 can be used to derive an efficient countermeasure against SCAs.

The remainder of this paper is organized as follows. In Section 2, we explain pairings and computational cost of some computations used in this paper. In Section 3, we briefly review SCAs against hardware implementations of the η_T pairing. Then, we describe existing countermeasures against SCAs for the η_T

pairing. In Section 4, we propose an efficient countermeasure based on the random value addition. In Section 5, we conclude this paper.

2 η_T Pairing and Computational Cost

In this section, we first explain the η_T pairing, which is one of the most efficient pairings proposed by Barreto *et al.* [2]. Second, we discuss the costs of the multiplication in $\mathbb{F}_{3^{6n}}$ and the scalar multiplication on the supersingular elliptic curve over $\mathbb{F}_{3^n}$. These cost estimations are applied to the comparison among various countermeasures against SCAs for the η_T pairing.

2.1 η_T Pairing

Let $\mathbb{F}_{3^n}$ be an extension field of extension degree n over $\mathbb{F}_3$. Let E^b be a supersingular elliptic curve defined by

$$E^b : y^2 = x^3 - x + b \text{ with } b \in \{1, -1\}. \tag{1}$$

All supersingular curves are isomorphic to this curve. All points in E^b with the point at infinity $\mathcal{O}$ are denoted by

$$E^b(\mathbb{F}_{3^n}) = \{(x, y) \in \mathbb{F}_{3^n} \times \mathbb{F}_{3^n} : y^2 = x^3 - x + b\} \cup \{\mathcal{O}\}.$$

Then, $E^b(\mathbb{F}_{3^n})$ forms a group, and the summation $P + Q \in E^b(\mathbb{F}_{3^n})$ for any $P, Q \in E^b(\mathbb{F}_{3^n})$ is computed by an explicit formula [9,24]. For $P \in E^b(\mathbb{F}_{3^n})$ and an integer $m(\neq 0)$, the operation

$$mP = P + P + \cdots + P \text{ (summation of } m \text{ terms)}$$

is called the *scalar multiplication*.

The extension degree n satisfies $gcd(n, 6) = 1$ because n is chosen to be a prime [2]. Then, n satisfies $n \equiv 1, 5, 7, 11 \pmod{12}$.

Denote the number of elements in the set S by $\#S$. Then, we know that

$$\#E^b(\mathbb{F}_{3^n}) = 3^n + 1 + b'3^{(n+1)/2},$$

where b' is defined by the following equation:

$$b' = \begin{cases} b & \text{if } n \equiv 1,\ 11 \pmod{12}, \\ -b & \text{if } n \equiv 5,\ 7 \pmod{12}. \end{cases}$$

The distortion map ψ is defined by

$$\psi(x, y) = (\rho - x, y\sigma) \tag{2}$$

with $\sigma^2 = -1$ and $\rho^3 = \rho + b$.

Let $l_{3P',b'P}$ be a function of a line going through $3P'$ and $b'P$, and let

$$g_R(x, y) = y_r^3 y - (x_r^3 + x - b)^2$$

Algorithm 1: Computation of η_T pairing for $n \equiv 1 \pmod{12}$

input: $P = (x_p, y_p), Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})$
output: $(\eta_T(P,Q)^{3^{(n+1)/2}})^W \in \mathbb{F}^*_{3^{6n}}$

1. **if** $b' = 1$ **then** $y_p \leftarrow -y_p$
2. $R_0 \leftarrow -y_p(x_p + x_q + b) + y_q\sigma + y_p\rho$
3. $d \leftarrow b$
4. **for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
5. $\quad r_0 \leftarrow x_p + x_q + d$
6. $\quad R_1 \leftarrow -r_0^2 + y_p y_q \sigma - r_0\rho - \rho^2$
7. $\quad R_0 \leftarrow R_0 R_1$
8. $\quad y_p \leftarrow -y_p$
9. $\quad x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9$
10. $\quad R_0 \leftarrow R_0^3$
11. $\quad d \leftarrow d - b \pmod 3$
12. **end for**
13. **return** R_0^W

be a function whose divisor is $(g_R) = 3(R) + (-3R) - 4(\mathcal{O})$ for $R = (x_r, y_r)$. The η_T pairing,

$$\eta_T : E^b(\mathbb{F}_{3^n}) \times E^b(\mathbb{F}_{3^n}) \to \mathbb{F}^*_{3^{6n}}$$

is defined by

$$\eta_T(P,Q) = l_{3P',b'P}(\psi(Q)) \prod_{j=0}^{(n-1)/2} g_{3^{-j}P'}(\psi(Q))^{3^j}, \tag{3}$$

where $P, Q \in E^b(\mathbb{F}_{3^n})$ and $P' = 3^{(n-1)/2}P$. Note that g_R only has such a simple form for supersingular elliptic curve $E^b(\mathbb{F}_{3^n})$. To obtain the bilinearity of the η_T pairing, we need a powering by

$$W = (3^{3n} - 1)(3^n + 1)(3^n + 1 - b'(3^{(n+1)})) \,(= (3^{6n} - 1)/\#E^b(\mathbb{F}_{3^n})\,). \tag{4}$$

Then, $\eta_T(aP, Q)^W = \eta_T(P, aQ)^W = (\eta_T(P,Q)^W)^a$ holds for any non-zero integer a. This powering by W is called the *final exponentiation*.

Next, we explain a relationship between the η_T pairing and Tate pairing[1] $e : E^b(\mathbb{F}_{3^n}) \times E^b(\mathbb{F}_{3^n}) \to \mathbb{F}^*_{3^{6n}}$, which is often used in practice. Then, there is a relationship between the η_T pairing and Tate pairing,

$$(\eta_T(P,Q)^W)^{3T^2} = e(P,Q)^Z,$$

where T and Z are integers defined by

$$T = 3^{(n+1)/2} + b',\ Z = -b'3^{(n+3)/2}.$$

Eq. (3) provides Algorithm 1 for computing the η_T pairing over $\mathbb{F}_{3^n}$, which is the no-cube-root version proposed by Beuchat *et al.* [3].

[1] More precisely e is the *modified Tate pairing*.

2.2 Cost of Some Computations

In this section, we describe computational costs of a multiplication in $\mathbb{F}_{3^{6n}}$, Algorithm 1, and a point addition and a scalar multiplication of a point on $E^b(\mathbb{F}_{3^n})$. These descriptions are required to compare the cost of our countermeasure with other existing countermeasures.

Some notations about the computational costs in $\mathbb{F}_{3^n}$ are defined as follows: M, C, I, A, and As are computational costs of a multiplication, a cubing, an inversion, an addition/subtraction, and some additions/subtractions in $\mathbb{F}_{3^n}$, respectively. We know that the cost of cubing in $\mathbb{F}_{3^{6n}}$ is $6M + As$.

We use the basis $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ for the extension field $\mathbb{F}_{3^{6n}}$, where σ and ρ are defined for the distortion map (Eq. (2)). For simplicity, $a_0 + a_1\sigma + a_2\rho + a_3\sigma\rho + a_4\rho^2 + a_5\sigma\rho^2 \in \mathbb{F}_{3^{6n}}$ is represented as $(a_0, a_1, a_2, a_3, a_4, a_5)$.

We have the following computational costs of some multiplications with special constant coefficients in $\mathbb{F}_{3^{6n}}$. Note that Step 7 in Algorithm 1 is computed by (ii) (not (i)).

Property 1. Multiplications in $\mathbb{F}_{3^{6n}}$ are computed by the Karatsuba method [14] with the following costs:

(i) $18M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, b_3, b_4, b_5)$ [12],
(ii) $13M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, -1, 0)$ [8],
(iii) $15M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, b_4, 0)$.

Proof. Refer to Appendix A for the proof of (iii). □

The cost of Algorithm 1 except the final exponentiation (Step 13) is estimated using Property 1. Note that a multiplication of Step 7, R_0R_1, costs $13M + As$ according to Property 1. Thus, Steps 2, 5, 6, 7, 8, 9, and 10 cost M, As, $2M$, $13M + As$, As, $4C$, and $6C + As$, respectively. Therefore, the total cost of Algorithm 1 except the final exponentiation is $M + ((n+1)/2) \cdot (As + 2M + 13M + As + As + 4C + 6C + As) = (7.5n + 8.5)M + (5n + 5)C + As$.

A scalar multiplication on $E^b(\mathbb{F}_{3^n})$ is efficiently performed with the tripling-and-addition method because a computation of tripling the point is very efficient. Indeed, a tripling point and a point addition using a projective coordinate system cost $M + 6C$ and $12M + 4C$, respectively. Then, a computation of scalar multiplication of mP costs

$$(9 \log_3 m)M + (8.7 \log_3 m)C + 2I \tag{5}$$

on average because $\log_3 m$ point triplings and $2/3 \cdot \log_3 m$ point additions on average and two inversions for restoring a point to the affine coordinate are required [9].

3 Previous Countermeasures against SCAs

In this section, we review previously known SCAs on pairing and countermeasures against them.

SCAs try to reveal the secret information on hardware devices by selectively inputting data into the device and monitoring the side channel information such as power consumption and timing while the device executes a cryptographic algorithm [15,16].

Scott first pointed out that SCAs can be used against pairing devices when P is public and Q is secret for the input of pairing $e(P, Q)$ [21]. For example, such a situation appears in Boneh and Franklin's identity-based encryption [4]. Moreover, Scott proposed two countermeasures, namely the *scalar multiplication and bilinearity method* and the *method of randomizing the intermediate value*. Page *et al.* [19] showed that SCAs can reveal secret data by monitoring a multiplication of the secret data by public data. They also proposed two countermeasures against these attacks, the *improved scalar multiplication and bilinearity method* and the *point-blinding method*. Kim *et al.* [13] showed that a computation of $\alpha \cdot (\beta + \gamma)$ can also be a subject of differential power analysis (DPA) attacks, where α and β are secret, and γ is public. They proposed a countermeasure, the *randomized-projective-coordinate method*, and that was improved by Choi *et al.* [5].

Whelan *et al.* [25] explained that a multiplication of secret data by public data, and squaring and square-root computations of secret data became subjects of SCAs if we use the pairing over finite fields of characteristic two. Moreover, they discuss that one input point Q of pairing $e(P, Q)$ is fixed (never changed) during computation of the BKLS algorithm. Thus, one might resist SCAs by setting the secret data to the updating point P. On the contrary, one needs a countermeasure in the case of the η_T pairing due to its symmetric structure.

Let $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ be points input into Algorithm 1. If P is a secret point, the target operations of SCA are $y_p(x_p + x_q + b)$ in Step 2, r_0^2 in Step 6, $y_p y_q$ in Step 6, or $R_0 R_1$ in Step 7. Moreover, if Q is a secret point, then x_q^9 or y_q^9 in Step 9 could also be the target of the attack.

In the remainder of this section, we explain details of the above countermeasures and estimate their costs when using the η_T pairing over $\mathbb{F}_{3^n}$.

3.1 Scalar Multiplication and Bilinearity Method [19]

In the scalar multiplication and bilinearity method, an integer α with $0 \leq \alpha < l$ is selected at random by a device, and another integer $\beta = (\alpha^{-1} \bmod l)$ is computed, where l is the largest prime factor of the order of the points. The device then computes

$$\eta_T(\alpha P, \beta Q)^W,$$

which is equal to $\eta_T(P, Q)^W$ because $\eta_T(\alpha P, \beta Q)^W = (\eta_T(P, Q)^W)^{\alpha\beta} = \eta_T(P, Q)^W$ due to the bilinearity of the η_T pairing. Even if point Q is selected by the attacker, it is changed to βQ by the device, and the attacker cannot know βQ. Therefore, this method provides a secure computation method for the η_T pairing against SCAs. The overhead is two scalar multiplications on the elliptic curve, which is $18nM + 17.4nC + 2I$ on average according to Eq. (5) because α and $\beta \approx 3^n$ and $\log_3 \alpha \approx \log_3 \beta \approx n$. Now, the costs of the computation of β and additions/subtractions is ignored.

3.2 Point-Blinding Method [19]

The point-blinding method computes

$$(\eta_T(P, Q + R) \cdot \eta_T(P, -R))^W,$$

where point R is selected by the device at random, and thus, the attacker cannot control R. Note that this value is equal to $\eta_T(P, Q)^W$ because $(\eta_T(P, Q + R) \cdot \eta_T(P, -R))^W = \eta_T(P, Q + R)^W \cdot \eta_T(P, -R)^W = \eta_T(P, Q)^W \cdot \eta_T(P, R)^W \cdot (\eta_T(P, R)^W)^{-1} = \eta_T(P, Q)^W$ due to the bilinearity of the η_T pairing. The overhead is a point addition on the elliptic curve $(12M + 2I)$ ($2I$ is needed for conversion of coordinates), a computation of the η_T pairing without final exponentiation $((7.5n + 8.5)M + (5n + 5)C)$, and a multiplication in $\mathbb{F}_{3^{6n}}$ $(18M)$. Note that there is no cost for the computation of $-R$ because $-R = (x_r, -y_r)$ for $R = (x_r, y_r)$. Then, the overhead cost is $(7.5n + 38.5)M + (5n + 5)C + 2I$.

3.3 Method of Randomizing Intermediate Value [21]

In the method of randomizing the intermediate value proposed by Scott, randomizations are performed for intermediate values related to Q and R_0 in loops of the pairing algorithm. Then, Steps 2, 5, and 6 in Algorithm 1 are modified as follows:

2. $R_0 \leftarrow -\lambda \cdot y_p(x_p + x_q + b) + \lambda \cdot y_q\sigma + \lambda \cdot y_p\rho$
5. $r_0 \leftarrow \lambda \cdot x_p + \lambda \cdot x_q + \lambda \cdot d$
6. $R_1 \leftarrow -\lambda \cdot r_0^2 + \lambda \cdot y_p y_q\sigma - \lambda \cdot r_0\rho - \lambda \cdot \rho^2$

where λ is a random value in $\mathbb{F}_{3^n}^*$ selected by the device. Denote by Algorithm 1' the modified Algorithm 1. Then, (R_0 in Step 12 of Algorithm 1) $= \lambda'$(R_0 in Step 12 of Algorithm 1') for some $\lambda' \in \mathbb{F}_{3^n}^*$. However, the effect of λ' is removed by the final exponentiation, namely $(\lambda' \cdot \eta_T(P, Q))^W = \eta_T(P, Q)^W$ for W of Eq. (4). Indeed, for $r \in \mathbb{F}_{3^{3n}}^*$ or $r \in \mathbb{F}_{3^n}^*$

$$r^W = 1 \tag{6}$$

is in general satisfied because

$$r^W = r^{(3^{3n}-1)(3^n+1)(3^n+1-b'(3^{(n+1)}))} = 1^{(3^n+1)(3^n+1-b'(3^{(n+1)}))} = 1.$$

Scott recommends to multiply intermediate values by λ not at once but one by one for security. Then, the overheads of Steps 2, 5, and 6 are $3M$, $3M$, and $3M$, respectively. Note that Step 7 also creates an overhead because R_1 has the form of $(b_0, b_1, b_2, 0, b_4, 0)$ not $(b_0, b_1, b_2, 0, -1, 0)$. Then, Step 7 takes $15M$ not $13M$ according to Property 1; namely, the overhead of Step 7 is $2M$. Therefore, the total overhead of this method is $(3.5n + 6.5)M$.

3.4 Randomized-Projective-Coordinate Method

Kim *et al.* proposed a randomized-projective-coordinate method for the η_T pairing algorithm over characteristic two [13].

Algorithm 2: Computation of the η_t pairing with the randomized projective coordinate method for $n \equiv 1 \pmod{12}$

input: $P = (x_p, y_p), Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})$
output: $(\eta_T(P,Q)^{3^{(n+1)/2}})^W \in \mathbb{F}^*_{3^{6n}}$

	randomizing P		randomizing Q
0.	$(X_p, Y_p, Z_p) \leftarrow (\lambda x_p, \lambda y_p, \lambda)$ (λ is a random value in $\mathbb{F}^*_{3^n}$)	0.	$(X_q, Y_q, Z_q) \leftarrow (\lambda x_q, \lambda y_q, \lambda)$ (λ is a random value in $\mathbb{F}^*_{3^n}$)
1.	**if** $b' = 1$ **then** $Y_p \leftarrow -Y_p$	1.	**if** $b' = 1$ **then** $y_p \leftarrow -y_p$
2.	$R_0 \leftarrow -Y_p\,(X_p + Z_p\,(x_q + b)) + Z_p^2\,y_q\sigma + Z_p\,Y_p\rho$	2.	$R_0 \leftarrow -y_p\,(Z_q\,(x_p + b) + X_q) + y_q\sigma + Z_q\,y_p\rho$
3.	$d \leftarrow b$	3.	$d \leftarrow b$
4.	**for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**	4.	**for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
5.	$r_0 \leftarrow X_p + Z_p\,(x_q + d)$	5.	$r_0 \leftarrow Z_q\,(x_q + d) + X_q$
6.	$R_1 \leftarrow -r_0^2 + Z_p\,Y_p\,y_q\sigma - Z_p\,r_0\rho - Z_p^2\,\rho^2$	6.	$R_1 \leftarrow -r_0^2 + Z_q\,y_p\,Y_q\sigma - Z_q\,r_0\rho - Z_q^2\,\rho^2$
7.	$R_0 \leftarrow R_0 R_1$	7.	$R_0 \leftarrow R_0 R_1$
8.	$Y_p \leftarrow -Y_p$	8.	$y_p \leftarrow -y_p$
9.	$x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9$	9.	$X_q \leftarrow X_q^9,\ Y_q \leftarrow Y_q^9,\ Z_q \leftarrow Z_q^9$
10.	$R_0 \leftarrow R_0^3$	10.	$R_0 \leftarrow R_0^3$
11.	$d \leftarrow d - b \pmod 3$	11.	$d \leftarrow d - b \pmod 3$
12.	**end for**	12.	**end for**
13.	**return** R_0^W	13.	**return** R_0^W

This paper gives a characteristic three version of the randomized-projective-coordinate algorithm (Algorithm 2), where the left side uses the projective coordinate for P, and the right side does that for Q. Although $R_0 = \lambda' \eta_T(P,Q)^{3^{(n+1)/2}}$ holds for some $\lambda' \in \mathbb{F}^*_{3^n}$ at Step 12 in both sides of Algorithm 2, the effect of λ' is removed by the final exponentiation as well as the method of randomizing the intermediate value. Note that like the method of randomizing the intermediate value, the cost of Step 7 in each side of Algorithm 2 is $15M$; namely, the overhead is $2M$ for each side. The cost of the left side of Algorithm 2 is $(10.5 + 17.5)M + (5n + 5)C + As$. Then, the overhead is $(3n + 9)M$. The cost of the right side is $(10.5 + 15.5)M + (6n + 6)C$. Then, the overhead is $(3n + 7)M + (n + 1)C$. When $2M > (n + 1)C$, for example, when $C = 0$ using a normal basis of $\mathbb{F}_{3^n}$ over $\mathbb{F}_3$, the left side is more efficient than the right side. In other cases, the right side is more efficient than the left side.

4 Proposed Countermeasure

In this section, we propose an efficient countermeasure against SCAs for the computation of η_T pairing over $\mathbb{F}_{3^n}$. The basic strategy is as follows: (1) The point (x_p, y_p) input into the η_T pairing is randomized by the random value

Algorithm 3: Proposed secure η_T pairing algorithm against SCAs for $n \equiv 1 \pmod{12}$

input: $P = (x_p, y_p), Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})$
output: $(\eta_T(P,Q))^{3^{(n+1)/2}} \in \mathbb{F}^*_{3^{6n}}$

1-1. **if** $b' = 1$ **then** $y_p \leftarrow -y_p$
1-2. $Y_p \leftarrow \lambda' y_p,\ Y_q \leftarrow \lambda' y_q$ (λ' is a random value in $\mathbb{F}^*_{3^n}$)
1-3. $x_p \leftarrow x_p + \lambda,\ y_p \leftarrow y_p + \lambda,\ x_q \leftarrow x_q - \lambda,\ y_q \leftarrow y_q - \lambda,\ \lambda'' \leftarrow \lambda^2$ (λ is a random value in $\mathbb{F}^*_{3^n}$, $\lambda = \lambda'$ is possible)
2. $R_0 \leftarrow -Y_p(x_p + x_q + b) + Y_q\sigma + Y_p\rho$
3. $d \leftarrow b$
4. **for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
5. $r_0 \leftarrow x_p + x_q + d$
6. $R_1 \leftarrow -(r_0 + \lambda)(r_0 - \lambda) - \lambda'' + (y_p y_q + \lambda(y_p - y_q - \lambda))\sigma - r_0\rho - \rho^2$
7. $R_0 \leftarrow R_0 R_1$
8. $x_p \leftarrow x_p - \lambda + \lambda^9,\ y_p \leftarrow -y_p + \lambda + \lambda^9$
9. $x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9,\ \lambda \leftarrow \lambda^9,\ \lambda'' \leftarrow \lambda''^9$
10. $R_0 \leftarrow R_0^3$
11. $d \leftarrow d - b \pmod 3$
12. **end for**
13. **return** R_0^W

additions $(x_p + \lambda)$ and $(y_p + \lambda)$, where λ is a random value in $\mathbb{F}^*_{3^n}$. (2) The effects of the random additions are removed at some steps in the algorithm. The proposed algorithm is represented as Algorithm 3.

4.1 Correctness of Algorithm 3

In the following, we prove that Algorithm 3 outputs the correct value of the η_T pairing over $\mathbb{F}_{3^n}$.

We try to investigate the differences between Algorithms 1 and 3. To distinguish the values in Algorithm 1 from those in Algorithm 3, each variable is denoted by suffix "$_1$" or "$_3$", respectively. For example, x_{p1} is denoted by x_p in Algorithm 1. Without of generality we can assume $\lambda' = 1$ due to Eq. (6). Then, we can prove the following lemma.

Lemma 1. *Suppose $\lambda' = 1$, then we have the following relationships*

$$x_{p3} = x_{p1} + \lambda^{9^i}, y_{p3} = y_{p1} + \lambda^{9^i}, x_{q3} = x_{q1} - \lambda^{9^i},\ \textit{and}\ y_{q3} = y_{q1} - \lambda^{9^i} \qquad (7)$$

at the beginning of the i-th iteration of Algorithms 1 and 3.

Proof. We prove the lemma by induction. We can easily see that Eq. (7) is satisfied for $i = 0$ at Step 1-3 of Algorithm 3. Next, suppose that at the i-th iteration Eq. (7) is correct. In Algorithm 1, x_{p1}, y_{p1}, x_{q1}, and y_{q1} are updated as

$x_{p1}, -y_{p1}, x_{q1}^9$, and y_{q1}^9, respectively. At Step 8 of the i-th iteration in Algorithm 3, x_{p3} is updated as

$$x_{p3} \leftarrow x_{p3} - \lambda^{9^i} + \lambda^{9^{i+1}} = (x_{p1} + \lambda^{9^i}) - \lambda^{9^i} + \lambda^{9^{i+1}} = x_{p1} + \lambda^{9^{i+1}},$$

and y_{p3} is updated as

$$y_{p3} \leftarrow -y_{p3} + \lambda^{9^i} + \lambda^{9^{i+1}} = -(y_{p1} + \lambda^{9^i}) + \lambda^{9^i} + \lambda^{9^{i+1}} = -y_{p1} + \lambda^{9^{i+1}}.$$

At Step 9 of the i-th iteration in Algorithm 3, x_{q3} and y_{q3} are updated as

$$x_{q3} \leftarrow x_{q3}^9 = (x_{q1} - \lambda^{9^i})^9 = x_{q1}^9 - \lambda^{9^{i+1}}, \text{ and } y_{q3} \leftarrow y_{q3}^9 = (y_{q1} - \lambda^{9^i})^9 = y_{q1}^9 - \lambda^{9^{i+1}},$$

respectively. Therefore, Eq. (7) is satisfied at the beginning of every $(i+1)$-th iteration. □

To prove the correctness of Algorithm 3, showing that

$$r_{03} = r_{01} \text{ and } R_{13} = R_{11}$$

are satisfied at every i-th iteration under the assumption $\lambda' = 1$ is sufficient. Indeed, at every i-th iteration we have the relationship

$$r_{03} = x_{p3} + x_{q3} + d = (x_{p1} + \lambda^{9^i}) + (x_{q1} - \lambda^{9^i}) + d = x_{p1} + x_{q1} + d = r_{01} \quad (8)$$

and

$$\begin{aligned} R_{13} &= -(r_{03} + \lambda)(r_{03} - \lambda) - \lambda'' + (y_{p3}y_{q3} + \lambda(y_{p3} - y_{q3} - \lambda))\sigma - r_0\rho - \rho^2 \\ &= -r_{03}^2 + \lambda^2 - \lambda'' + (y_{p3}y_{q3} + \lambda(y_{p3} - y_{q3} - \lambda))\sigma - r_0\rho - \rho^2 \\ &= -r_{01}^2 + y_{p1}y_{q1}\sigma - r_{01}\rho - \rho^2 \\ &= R_{11} \end{aligned}$$

from Eqs. (7) and (8) because $\lambda'' = \lambda^2$ holds in every i-th iteration. Therefore, we proved the correctness of Algorithm 3.

4.2 Security

We discuss the security of Algorithm 3 against SCAs. The attacker targets Steps 2, 5, 6, and 7 in Algorithm 3. However, all values are randomized by random values λ and λ'' for Steps 2, 5, and 6. Step 7 is also secure because, although R_1 is not randomized, R_0 is randomized by λ' at Step 1-2. An explanation may be needed for Step 6. Note that

$$-(r_0 + \lambda)(r_0 - \lambda) - \lambda'' = -r_0^2$$

because λ'' is equal to λ^2, as noted above. However, we need the computation of "$-(r_0 + \lambda)(r_0 - \lambda) - \lambda''$" (not r_0^2) because r_0 is unchangeable either with the randomization at Step 1-3 or without it. Therefore, if this process is not performed, then the attacker may guess r_0^2.

Table 1. Comparison of overheads for countermeasure against SCAs

Countermeasure method	Additional Cost
Point Multiplication and Bilinearity Method [19]	$18nM + 17.4nC + 2I$
Point-Blinding Method [19]	$(7.5n + 38.5)M + (5n + 5)C + 2I$
Method of Randomizing the Intermediate Value [21]	$(3.5n + 6.5)M$
Left Side of Algorithm 2*	$(3n + 9)M$
Right Side of Algorithm 2*	$(3n + 7)M + (n + 1)C$
The Proposed Method (Algorithm 3)	$\mathbf{(0.5n + 3.5)M + (3n + 3)C}$

* Algorithm 2 is the characteristic three version of the randomized-projective-coordinate method [13].

4.3 Comparison with Other Methods

We estimate the computational cost of the proposed scheme and compare it with the previously known methods described in Section 3. Now, we suppose that the cost of a squaring is equal to that of a multiplication.

Here, we ignore the costs of additions/subtractions. Step 7 in Algorithm 3 requires $13M$ as well as Algorithm 1 because R_1 in Algorithm 3 has the form of $(b_0, b_1, b_2, 0, -1, 0)$ due to Property 1. Note that this sparse multiplication can be applied to neither Algorithm 2 nor the method of randomizing the intermediate value. Steps 1-2, 2, 6, 7, 8, 9, and 10 in Algorithm 3 cost $3M$, M, $3M$, $13M$, $2C$, $6C$, and $6C$, respectively. Therefore, the total cost of Algorithm 3 except the final exponentiation is $3M + M + ((n + 1)/2) \cdot (3M + 13M + 2C + 6C + 6C) = (8n + 12)M + (7n + 7)C$. Then, the overhead is $(0.5n + 3.5)M + (3n + 3C)$.

A comparison of the proposed countermeasure with existing countermeasures is shown in Table 1. Note that the extension degree should satisfy $n \geq 97$ for security reasons. We estimate the cost of cubing to be $C = 0.07M$ for the polynomial basis [8,20] and $C = 0$ for the normal basis on hardware. The overhead of the proposed method is $72.58M$ for $C = 0.07M$ and $n = 97$ and that of the left side of Algorithm 2 is $300M$, which is the smallest of any algorithm. Then, the overhead cost is reduced by 76%. When we choose a sufficiently large n, then the overhead of the proposed method and Algorithm 2 becomes $0.5nM$ and $3nM$, respectively, for $C = 0$. In this case, the overhead is reduced by 83%.

5 Conclusion

In this paper, we proposed a variation of the η_T pairing over $\mathbb{F}_{3^n}$ that is secure against SCAs. The randomization technique in the proposed scheme uses the random value additions $x_p + \lambda$ and $y_p + \lambda$ for the input point $P = (x_p, y_p)$ and a random value $\lambda \in \mathbb{F}_{3^n}^*$. Interestingly, the symmetric structure of the η_T pairing provides a simple algebraic equation with a random value of λ in the main loop. Therefore, the proposed scheme has the smallest overhead of the randomization secure against SCAs, which is just $0.5nM$ for the η_T pairing over $\mathbb{F}_{3^n}$. That is reduced by more than 75% compared with the randomized-projective-coordinate method. The method of this paper is applied to the Duursma-Lee algorithm due to its similarity with the η_T pairing algorithm.

Acknowledgments

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

References

1. Barreto, P., Kim, H., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
2. Barreto, P., Galbraith, S., ÓhÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. In: Designs, Codes and Cryptography, vol. 42(3), pp. 239–271. Springer, Heidelberg (2007)
3. Beuchat, J.-L., Shirase, M., Takagi, T., Okamoto, E.: An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation. In: 18th IEEE International Symposium on Computer Arithmetic, ARITH-18, pp. 97–104 (2007)
4. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
5. Choi, D.H., Han, D.-G., Kim, H.W.: Construction of efficient and secure pairing algorithm and its application. Cryptology ePrint Archive, Report 2007/296 (2007)
6. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
7. Grabher, P., Page, D.: Hardware acceleration of the Tate pairing in characteristic three. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 398–411. Springer, Heidelberg (2005)
8. Granger, R., Page, D., Stam, M.: Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three. IEEE Transactions on Computers 54(7), 852–860 (2005)
9. Harrison, K., Page, D., Smart, N.P.: Software implementation of finite fields of characteristic three. LMS JCM 5, 181–193 (2002)
10. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory 52, 4595–4602 (2006)
11. Itoh, K., Izu, T., Takenaka, M.: Efficient countermeasures against power analysis for elliptic curve. In: CARDIS 2004, pp. 99–114 (2004)
12. Kerins, T., Marnane, W., Popovici, E., Barreto, P.: Efficient hardware for the Tate pairing calculation in characteristic three. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 412–426. Springer, Heidelberg (2005)
13. Kim, T.H., Takagi, T., Han, D.-G., Kim, H.W., Lim, J.: gPower analysis attacks and countermeasures on η_T pairing over binary fields. ETRI Journal 30(1), 68–80 (2008)
14. Knuth, D.E.: Seminumerical algorithms. Addison-Wesley, Reading (1981)
15. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
16. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

17. Mamiya, H., Miyaji, A., Morimoto, H.: Secure elliptic curve exponentiation against RPA, ZRA, DPA, and SPA. IEICE Transactions on Fundamentals E90-A(1), 22–28 (2006)
18. Miller, V.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
19. Page, D., Vercauteren, F.: A fault attack on pairing based cryptography. IEEE Transactions on Computers 55(9), 1075–1080 (2006)
20. Ronan, R., ÓhÉigeartaigh, C., Murphy, C., Kerins, T., Barreto, P.: A reconfigurable processor for the cryptographic η_T pairing in characteristic 3. In: Information Technology: New Generations, ITNG 2007, pp. 11–16. IEEE Computer Society, Los Alamitos (2007)
21. Scott, M.: Computing the Tate pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
22. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
23. Shu, C., Kwon, S., Gaj, K.: FPGA accelerated Tate pairing based cryptosystems over binary fields. Cryptology ePrint Archive, Report 2006/179 (2006)
24. Silverman, J.: The arithmetic of elliptic curves. Springer, Heidelberg (1986)
25. Whelan, C., Scott, M.: Side channel analysis of practical pairing implementations: Which path is more secure? In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 81–98. Springer, Heidelberg (2006)
26. Whelan, C., Scott, M.: The importance of the final exponentiation in pairings when considering fault attacks. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 225–246. Springer, Heidelberg (2007)

A Proof of Property 1

Property 1. Multiplications in $\mathbb{F}_{3^{6n}}$ are computed by the Karatsuba method [14] with the following cost:

(i) $18M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, b_3, b_4, b_5)$ [12],
(ii) $13M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, -1, 0)$ [8],
(iii) $15M + As$ for multiplication $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, b_4, 0)$.

Proof. (i) and (ii) are known results. However, we provide proofs of all cases because an explanation of (i) is needed to show (iii), and there may not be an explicit proof of (ii). First, consider a multiplication in $\mathbb{F}_{3^{2n}}$

$$(a_0 + a_1\sigma) \times (b_0 + b_1\sigma) = (a_0b_0 - a_1b_1) + (a_0b_1 + a_1b_0)\sigma.$$

Using the Karatsuba method costs $3M + As$. Indeed, $(a_0b_1 - a_1b_0)$ can be computed as $(a_0b_1 - a_1b_0) = (a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1$. Then, three multiplications, a_0b_0, a_1b_1, and $(a_0 + a_1)(b_0 + b_1)$ are needed to compute $(a_0 + a_1\sigma) \times (b_0 + b_1\sigma)$.

(i) Suppose that $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, b_3, b_4, b_5) = (c_0, c_1, c_2, c_3, c_4, c_5)$. Let $\tilde{a}_{i(i+1)}$, $\tilde{b}_{i(i+1)}$, and $\tilde{c}_{i(i+1)}$ be elements in $\mathbb{F}_{3^{2n}}$ defined by $a_i + a_{i+1}\sigma$, $b_i + b_{i+1}\sigma$,

and $c_i + c_{i+1}\sigma$, respectively, for $i = 0, 2, 4$. For example, $(a_0, a_1, a_2, a_3, a_4, a_5)$ is represented as $\tilde{a}_{01} + \tilde{a}_{23}\rho + \tilde{a}_{45}\rho^2$. Then, $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, b_3, b_4, b_5)$ is computed as follows:

$$\begin{aligned}
\tilde{d}_0 &= \tilde{a}_{01}\tilde{b}_{01}, \\
\tilde{d}_1 &= (\tilde{a}_{01} + \tilde{a}_{23})(\tilde{b}_{01} + \tilde{b}_{23}) - \tilde{a}_{01}\tilde{b}_{01} - \tilde{a}_{23}\tilde{b}_{23}, \\
\tilde{d}_2 &= (\tilde{a}_{01} + \tilde{a}_{45})(\tilde{b}_{01} + \tilde{b}_{45}) + \tilde{a}_{23}\tilde{b}_{23} - \tilde{a}_{01}\tilde{b}_{01} - \tilde{a}_{45}\tilde{b}_{45}, \\
\tilde{d}_3 &= (\tilde{a}_{23} + \tilde{a}_{45})(\tilde{b}_{23} + \tilde{b}_{45}) - \tilde{a}_{23}\tilde{b}_{23} - \tilde{a}_{45}\tilde{b}_{45}, \\
\tilde{d}_4 &= \tilde{a}_{45}\tilde{b}_{45},
\end{aligned} \tag{9}$$

and

$$\begin{aligned}
\tilde{c}_{01} &= \tilde{d}_0 + b\tilde{d}_3, \\
\tilde{c}_{23} &= \tilde{d}_1 + \tilde{d}_3 + b\tilde{d}_4, \\
\tilde{c}_{45} &= \tilde{d}_2 + \tilde{d}_4,
\end{aligned}$$

where $b = 1$ or -1 defined by Eq. (1). Therefore, a multiplication in $\mathbb{F}_{3^{6n}}$ takes 6 multiplications,

$$\tilde{a}_{01}\tilde{b}_{01},\ \tilde{a}_{23}\tilde{b}_{23},\ \tilde{a}_{45}\tilde{b}_{45},\ (\tilde{a}_{01} + \tilde{a}_{23})(\tilde{b}_{01} + \tilde{b}_{23}),\ (\tilde{a}_{01} + \tilde{a}_{45})(\tilde{b}_{01} + \tilde{b}_{45}),$$
$$\text{and } (\tilde{a}_{23} + \tilde{a}_{45})(\tilde{b}_{23} + \tilde{b}_{45}),$$

and some additions/subtractions in $\mathbb{F}_{3^{2n}}$. Then a multiplication in $\mathbb{F}_{3^{6n}}$ takes $18M + As$.

(ii) In this case, $\tilde{b}_{23} = b_2$ and $\tilde{b}_{45} = -1$ in Eq. (9). Each of $\tilde{a}_{01}\tilde{b}_{01}$, $(\tilde{a}_{01}+\tilde{a}_{23})(\tilde{b}_{01}+\tilde{b}_{23})$, and $(\tilde{a}_{01}+\tilde{a}_{45})(\tilde{b}_{01}+\tilde{b}_{45})$ takes $3M+As$, but each of the other multiplications takes a smaller cost. Indeed, $\tilde{a}_{23}\tilde{b}_{23} = a_2b_2 + a_3b_2\sigma$ takes $2M + As$, $\tilde{a}_{45}\tilde{b}_{45} = -a_4 - a_5\sigma$ takes no cost, and $(\tilde{a}_{23} + \tilde{a}_{45})(\tilde{b}_{23} + \tilde{b}_{45}) = (a_2 + a_4)(b_2 - 1) + (a_3 + a_5)(b_2 - 1)\sigma$ takes $2M + As$. Therefore, $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, -1, 0)$ takes $13(= 3 \times 3 + 2 + 0 + 2)M + As$.

(iii) In this case $\tilde{b}_{23} = b_2$ and $\tilde{b}_{45} = b_4$ in Eq. (9). Each of $\tilde{a}_{01}\tilde{b}_{01}$, $(\tilde{a}_{01}+\tilde{a}_{23})(\tilde{b}_{01}+\tilde{b}_{23})$ and $(\tilde{a}_{01} + \tilde{a}_{45})(\tilde{b}_{01} + \tilde{b}_{45})$ takes $3M + As$ as well as (ii). On the other hand, each of the other multiplications takes $2M + As$ because $\tilde{a}_{23}\tilde{b}_{23} = a_2b_2 + a_3b_2\sigma$, $\tilde{a}_{45}\tilde{b}_{45} = a_4b_4 + a_5b_4\sigma$, and $(\tilde{a}_{23} + \tilde{a}_{45})(\tilde{b}_{23} + \tilde{b}_{45}) = (a_2 + a_4)(b_2 + b_4) + (a_3 + a_5)(b_2 + b_4)\sigma$. Therefore, $(a_0, a_1, a_2, a_3, a_4, a_5) \times (b_0, b_1, b_2, 0, b_4, 0)$ takes $15(= 3 \times 3 + 3 \times 2)M + As$. □