An Efficient DPA Countermeasure for the Eta_T Pairing Algorithm over $GF(2^n)$ Based on Random Value Addition

Seog Chung Seo, Dong-Guk Han, and Seokhie Hong

This paper presents an efficient differential power analysis (DPA) countermeasure for the Eta_T pairing algorithm over $GF(2^n)$. The proposed algorithm is based on a random value addition (RVA) mechanism. An RVA-based DPA countermeasure for the Eta_T pairing computation over $GF(3^n)$ was proposed in 2008. This paper examines the security of this RVA-based DPA countermeasure and defines the design principles for making the countermeasure more secure. Finally, the paper proposes an efficient RVA-based DPA countermeasure for the secure computation of the Eta_T pairing over $GF(2^n)$. The proposed countermeasure not only overcomes the security flaws in the previous RVA-based method but also exhibits the enhanced performance. Actually, on the 8-bit ATmega128L and 16-bit MSP430 processors, the proposed method can achieve almost 39% and 43% of performance improvements, respectively, compared with the best-known countermeasure.

Keywords: Eta_T pairing computation, differential power analysis, efficient countermeasure.

I. Introduction

Pairings over elliptic curves are functions that map two points over the underlying curves into an element over extended finite fields and are characterized by bilinearity and non-degeneracy. Recently, many ID-based encryption/signature/authentication key agreement schemes have been developed by using pairing characteristics. However, because pairing computations require higher overhead than elliptic curve operations, researchers have developed efficient pairing algorithms that can be practically applied to ID-based protocols. Early studies involved optimizing the computation of the Tate pairing.

Barreto and others [1] and Galbraith and others [2] proposed denominator elimination techniques for the original Miller loop and the efficient elliptic curve tripling formula and pairing arithmetic over $GF(3^n)$. Duursma and Lee presented an efficient closed formula for the Tate pairing over a field of characteristic three by combining elliptic curve and divisor arithmetic [3]. Similarly, Kwon presented a closed formula for the Tate pairing over a binary field [4]. Barreto and others generalized the results of [3], [4] as the *Eta* pairing on supersingular curves with characteristics two and three. Furthermore, they presented an Eta_T pairing that shortens the main loop of the Eta_T pairing by half on supersingular curves in characteristic two [5]. Hess and others extended the result of [5] in a more generic way as the Ate pairing on non-supersingular curves [6].

Side-channel attacks (SCAs) make use of some unintentionally leaked side-channel information related to secret data or intermediate results arising from cryptographic operations for finding the secret data [7]. Page and others [8], Scott [9], Whelan [10], Kim and others [11], and Choi and

Manuscript received Oct. 19, 2010; revised Feb. 8, 2011; accepted Feb. 24, 2011.

This work was supported by the Second Brain Korea 21 Project. This work was supported by the IT R&D program of MKE, Rep. of Korea [Development of Privacy Enhancing Cryptography on Ubiquitous Computing Environment].

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0026354).

Seog Chung Seo (phone: +82 10 2086 9621, email: ssc2007@korea.ac.kr) and Seokhie Hong (corresponding author, email: shhong@ korea.ac.kr) are with the Center for Information Security Technologies (CIST), Korea University, Seoul, Rep. of Korea.

Dong-Guk Han (email: christa@kookmin.ac.kr) is with the Department of Mathematics, Kookmin University, Seoul, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.11.0110.0597

others [12] examined SCAs on pairing algorithms. In contrast to SCAs on ECCs, SCAs on pairing algorithms attempt to determine the value of the point used for the private key because the scalar used in the Miller loop is no longer secret information. For more information on the scenarios of DPA/CPA [13] on the Eta_T pairing algorithm, the reader is referred to [10], [11]. Previous research has proposed three types of DPA countermeasures. The first type uses the bilinearity of pairings [8]. The second type, the most widely researched type, multiplies intermediate variables by random values during pairing computations [9]-[12]. The third type, proposed by Shirase [14], adds some random values, that is, random value addition (RVA), to intermediate variables suspicious of being targets of DPA to secure the computation of Eta_T pairing over $GF(3^n)$. Even though Shirase's method is very efficient, it has some security flaws. The details about three types of countermeasure are described in section II.

This paper contributes to the literature by providing the following:

i) An analysis of the security flaws in Shirase's DPA countermeasure in the context of its implementation,

ii) A presentation of an efficient RVA-based DPA countermeasure for the Eta_T pairing over $GF(2^n)$,

iii) A comparison of the proposed method with previously proposed algorithms with regard to both security and performance.

The rest of this paper is organized as follows. Section II provides a literature review about the Eta_T pairing over $GF(2^n)$, weaknesses of the Eta_T pairing with respect to DPA, and previously proposed DPA countermeasures. Section III evaluates the security of Shirase's RVA-based countermeasure in the context of its implementation. Section IV discusses some guidelines for making the RVA-based countermeasures more secure and then presents a new more efficient RVA-based countermeasure obeying the established guidelines. Section V verifies the security of the proposed method through practical CPA experiments and compares the proposed countermeasure with previous countermeasures in terms of computational efficiency, and section VI concludes this paper.

II. Related Work

We now examine the Eta_T pairing over $GF(2^n)$ and its previously proposed DPA countermeasures.

1. Overview of the Eta_T Pairing over $GF(2^n)$

Let E be a supersingular elliptic curve over $GF(2^n)$:

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

Let l be the largest prime with $l \mid \#E_b(GF(2^n))$ and let the embedding degree of this curve be 4, the smallest positive integer k that ensures $l \mid 2^{kn}-1$. Then, we have

$$N=\#E_b(GF(2^n))=\begin{cases}2^n+1+(-1)^b 2^{(n+1)/2}, & \text{if } n\equiv 1, 7 \pmod 8,\\ 2^n+1+(-1)^b 2^{(n+1)/2}, & \text{if } n\equiv 3, 5 \pmod 8.\end{cases}$$

By choosing $T=2^n-N$ and a prime l, Barreto and others [5] proposed the Eta_T pairing which is a type of mapping yielding

$$\eta_T: E_b(GF(2^n))[l]\times E_b(GF(2^n))[l]\rightarrow GF(2^{4n})^*.$$

Then, $\eta_T(P,Q) = f_{T,P}(\psi(Q))$ if $T < 0$ we replace P with $-P$ and T with $-T$ where $f_{T,P}$ is a rational function satisfying $(f_{T,P}) = T(P)-T(O)$. Here ψ, a distortion map that converts an element in $E_b(GF(2^n))[l]$ into $E_b(GF(2^{4n}))[l]$, is defined as $\psi(x,y) = (x+s^2, y+sx+t)$, where $s^2+s+1=0$, $t^2+t+s=0$, and $s, t \in GF(2^{4n})$.

Thus, we can represent an element in $GF(2^{4n})$ as an extension of $GF(2^n)$ by using the basis $(1, s, t, st)$.

Here $f_{T,P}^M$, including the final powering, is given by [5]

$$f_{T,P}(\psi(Q))^M = (l(\psi(Q))\prod_{i=0}^{\frac{n-1}{2}} g_{[2^i P]}(\psi(Q))^{2^{\frac{n-1}{2}-i}})^M,$$

where $M=(2^{2n}-1)(2^n\pm 2^{(n+1)/2}+1)$, where g_V is a rational function satisfying $g_V = 2(V)+([-2]V)-3(O)$.

In the above formula,

i) The point doubling formula is given by

$$[2^i]P=(x_P^{(2i)}+i, y_P^{(2i)}+ix_P^{(2i)}+\varepsilon_i),$$

where $x_P^{(2i)}$ denotes $x_P^{(2i)} = x_P^{2^{2i'}}$ with $i = i' \pmod n$ and $\varepsilon_i = 0$ if $i \equiv 0,1 \pmod 4$ $\varepsilon_i = 1$ otherwise.

ii) $g_V(x, y)$ defined over $E(GF(2^{4n}))[l]$ is the rational function having divisor $2(V)+([-2]V)-3(O)$, where $V=(x_V, y_V)$ in $E(GF(2^n))[l]$, (x, y) in $E(GF(2^{4n}))[l]$. This is given by

$$g_V(x, y) =(x_V^2+1)(x_V+x)+y_V+y = x(x_V^2+1)+y_V^2+y+b.$$

iii) l_V is the equation of the line passing $2^{(n+1)/2}V$ and εV with $\varepsilon = (-1)^{b+\varepsilon_{(n+1)/2}}$. This is given by

$$l_V(x,y) = y+y_V+b+\varepsilon_{(n+1)/2}+(x_V+(n-1)/2)(x+x_V).$$

The Eta_T pairing, like the Tate pairing, preserves the bilinearity property, such as $\eta_T(aP, bQ) = \eta_T(aP, bQ) = \eta_T(P, Q)^{ab}$, for any P, Q in $E(GF(2^n))$ and any integers a, b.

Algorithm 1 depicts the Eta_T pairing algorithm over $GF(2^n)$. In the algorithm, $\boldsymbol{M}$ and $\boldsymbol{S}$ denote the computational overhead of the multiplication and squaring over $GF(2^n)$. The total cost of algorithm 1 without the final powering is $(3.5n+3.5)\boldsymbol{M}+(4n-1)\boldsymbol{S}$ by omitting the computation of steps 8 and 9 when i is zero.

2. Weaknesses of the Eta_T Pairing over $GF(2^n)$ against DPA

As described in the introduction, DPA on pairing algorithms

Algorithm 1. Eta_T pairing algorithm over $GF(2^n)$ without a DPA countermeasure.

1. Input: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$
2. Output: $C = \eta_T(P,Q)$
3. $C \leftarrow 1$.
4. $x_P \leftarrow x_P^2+1$, $y_P \leftarrow y_P^2+1$. // (2***S***)
5. $u \leftarrow y_Q+b+1$, $v \leftarrow x_Q+1$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
6. for $i \leftarrow 0$ to $(n-1)/2$ do
7. $A \leftarrow (y_P+\theta+u)+(x_P+v+1)s+t$.
8. $C \leftarrow C^2$. // (4***S***)
9. $C \leftarrow C \cdot A$. // (6***M***)
10. if $i < (n-1)/2$ then
11. $x_P \leftarrow x_P^4$, $y_P \leftarrow y_P^4$. // (4***S***)
12. $u \leftarrow u+v+1$, $v \leftarrow v+1$.
13. $\theta \leftarrow x_P \cdot v$. // (1***M***)
14. end if
15. end for
16. $A \leftarrow A+(x_P^2+v+1)+s$. // (1***S***)
17. $C \leftarrow C \cdot A$. // (6***M***)
18. $C \leftarrow C^M$, where $M=(2^{2n}-1)(2^n+2^{(n+1)/2}+1)$.
19. return C.

targets finite field arithmetic such as multiplication and addition because the sequence of the Miller loop is not secret information any more but public information, unlike with elliptic curve cryptosystems.

The aim of DPA on pairing algorithms is to find either the x or y coordinate of the fixed private point. Attackers need to be able to guess the actual values of intermediate results to launch a DPA attack. For example, if there are computations such as $(k \circ p)$ ($\circ$ is a binary operator), where the k is a fixed and unknown value and the p is a known and controllable value to the attacker, the attackers can determine the value of k by observing the peaks in their DPA simulation. For example, in the DPA simulation, attackers divide the power traces into two sets and average each set as S_0, S_1 according to the results of $(k \circ p)$ such as LSB (0 or 1). If the guessed k is correct, then one can observe a significant peak (Fig. 1) in the differential trace, $|S_1-S_0|$. Otherwise, there is no peak in the $|S_1-S_0|$. For more details, refer to [7]. In other words, if the attacker guesses the least significant bit (LSB) in the result of (k xor p) (the LSB of p is known to the attacker as 1 (resp. 0)) as 1 and there are some distinguishable peaks in their DPA simulation, then the LSB of k is 0 (resp. 1). For a detailed DPA/CPA scenario for attacking Eta_T pairing algorithms, the reader is referred to [10] and [11]. Thus, the target points of DPA in algorithm 1 can be summarized as

i) (Fixed secret value) + (variable public value)
- In step 7, (x_P+v+1).
- In step 7, $(y_P+\theta+u)$ if (y_P+u) is first computed instead of computing $(y_P+\theta)$ or $(\theta+u)$.
- In step 16, (x_P^2+v+1).

ii) (Fixed secret value)*(variable public value)
- In step 5, $\theta \leftarrow x_P \cdot v$.
- In step 13, $\theta \leftarrow x_P \cdot v$.

Thus, the above weaknesses need to be properly addressed to protect the Eta_T pairings from DPA attacks.

3. DPA Countermeasures for the Eta_T Pairing over $GF(2^n)$

Three types of DPA countermeasure have been proposed for the Eta_T pairing algorithm.

A. Countermeasures Using Bilinearity

Page and others used the bilinearity property to randomize private data such as $e(P, Q) = e(sP, tQ)^{1/st}$, where s and t are random values [8]. Further, they blinded the public input point by computing $e(P, Q) = e(P, Q+R) \cdot e(P, R)^{-1}$, where R is a random point.

B. Countermeasures Based on RPC

Kim and others used randomized projective coordinates [11] that had been proposed by Coron to protect elliptic curve cryptography against DPA [15]. They converted the input point $Q = (x_Q, y_Q)$ into $(X_Q, Y_Q, Z_Q)=(\lambda x_Q, \lambda y_Q, \lambda)$, where λ is a random value in $GF(2^n)$. Then, they presented following explicit formula for computing the step 7 of algorithm 1 securely (Actually, they derived the above secure explicit formula for the Eta_T pairing algorithm using square root computations from [5]).

$$A \leftarrow \frac{1}{Z_Q}\Big\{(x_P(Z_Q\sqrt{x_P}+X_Q)+Z_Q(\sqrt{y_P}+\sqrt{x_P})+Y_Q) + (Z_Q x_P + X_Q)s + (Z_Q)t\Big\}.$$

In the above explicit formula, not only the coordinates of the input point Q are randomized but also those of the private point P are masked with randomized intermediate values. Since the computation of the step 7 is masked with multiplicative random value Z_Q, attackers cannot guess the actual value for launching DPA. However, the number of multiplications in the step 7 increases from one to four.

Choi and others extended the finding of Kim and others by simplifying the explicit formula by using the expression of the Weierstrass equation E_b: $Y^2+Y=X^3+X+b$ [12]. They proposed an RPC friendly Eta_T pairing computation algorithm and applied their countermeasure to Eta_T pairing algorithm using square root computations. They converted the explicit formula (the counterpart of the step 7 of algorithm 1) from

$$A \leftarrow wx_Q + y_Q + \left((n+1)/2 \cdot w + y_P + b + \varepsilon_{(n-1)/2}\right) + (w+x_Q)s + t$$

to

$$A \leftarrow w\overline{X_Q}+\overline{Y_Q}+\overline{Z_Q}((n+1)/2 \cdot w+y_P+b+\varepsilon_{(n-1)/2}) +(\overline{Z_Q}w+\overline{X_Q})s+\overline{Z_Q}t,$$

where $\overline{X_Q}=x_Q\lambda$, $\overline{Y_Q}=y_Q\lambda$, $\overline{Z_Q}=\lambda$, $w=x_P+(m+1)/2$, a random λ in $GF(2^n)$. Because this explicit formula for computing the step 7 securely requires three multiplications, the formula requires one less multiplication than Kim's method. For more details, refer to [11], [12].

C. Countermeasure Based on RVA

A key weakness of RPC-based countermeasures is their relatively high overhead. Thus, Shirase and others [14] proposed an RVA-based DPA countermeasure that uses some random values to mask intermediate variables of the Miller loop, which are targets of DPA. Their countermeasure uses the fact that the cost of additions over $GF(p^n)$ (p=2 or 3) is much lower than that of multiplications. Algorithm 2 is Shirase's Eta_T pairing algorithm over $GF(3^n)$ (We have slightly modified its notation). Even if the countermeasure is very efficient, it has some security flaws: the method is secure at the moment of the computation, but the masking effect disappears after the computation. For example, when computing step 11 of algorithm 2, x_P and x_Q are masked with random λ and $-\lambda$, respectively. However, after completing step 11, the effect of masking is canceled out because of $0=\lambda+(-\lambda)$. Thus, the unmasked real value $r_0=(x_P+x_Q+b)$ is moved to a certain memory location through the bus. Because attackers can obtain power traces that are related to this unmasked value moving through the bus, they can launch DPA attacks. Furthermore, when r_0 is loaded from the memory to register through the bus for computing step 12, the power consumption related to (x_P+x_Q+b) is revealed to attackers. A similar security problem appears when the result of step 12 is moved to the memory. In the next section, we analyze the security flaws in Shirase's method in greater detail.

Algorithm 2. Shirase's Eta_T pairing algorithm over $GF(3^n)$ for $n \equiv 1 \pmod{12}$.

1. Input: $P=(x_P, y_P)$, $Q=(x_Q,y_Q) \in E_b(GF(3^n))$
2. Output: $\eta_T(P,Q)^{3^{(n+1)/2}} \in GF(3^{6n})$
3. if $b == 1$ then
4. $y_P \leftarrow -y_P$.
5. end if
6. $Y_P \leftarrow \lambda' y_P$, $Y_Q \leftarrow \lambda' y_Q$. // ($\lambda'$ is a random value in $GF(3^n)$)
7. $x_P \leftarrow x_P+\lambda$, $y_P \leftarrow y_P+\lambda$, $x_Q \leftarrow x_Q-\lambda$, $y_Q \leftarrow y_Q-\lambda$, $\lambda'' \leftarrow \lambda^2$. ($\lambda$ is a random value in $GF(3^n)$, $\lambda=\lambda'$ is possible).
8. $C \leftarrow -Y_P(x_P+x_Q+b)+Y_Q\sigma+Y_P\rho$.
9. $d \leftarrow b$.
10. for from $i \leftarrow 0$ to $(n-1)/2$ do
11. $r_0 \leftarrow x_P+x_Q+d$.
12. $A \leftarrow -(r_0+\lambda)(r_0-\lambda)-\lambda''+(y_Py_Q+\lambda(y_P-y_Q-\lambda))\sigma-r_0\rho-\rho^2$.
13. $C \leftarrow C \cdot A$.
14. $x_P \leftarrow x_P-\lambda+\lambda^9$, $y_P \leftarrow -y_P+\lambda+\lambda^9$.
15. $x_Q \leftarrow x_Q^9$, $y_Q \leftarrow y_Q^9$, $\lambda \leftarrow \lambda^9$, $\lambda'' \leftarrow \lambda''^9$.
16. $C \leftarrow C^3$.
17. $d \leftarrow d-b \pmod 3$.
18. end for
19. $C \leftarrow C^W$, where $W=(3^{3n}+1)(3^n+1)(3^n-b3^{(n+1)/2}+1)$.
20. return C.

III. Security Analysis of Previous RVA-Based Countermeasure

We now analyze the security flaws in Shirase's method in the context of its implementation in detail. For simplicity, we convert Shirase's method originally proposed for the Eta_T pairing over $GF(3^n)$ into that over $GF(2^n)$ (This is because arithmetic operations in $GF(2^n)$ are simpler to illustrate than those in $GF(3^n)$ in point of assembly code implementation). Algorithm 3 is equipped with the idea of [14]. In contrast to the original algorithm, we set $(x_P, y_P) \leftarrow (x_P+\lambda, y_P+\lambda)$ and $(x_Q, y_Q) \leftarrow (x_Q+\lambda^2, y_Q+\lambda^2)$ instead of $(x_P, y_P) \leftarrow (x_P+\lambda, y_P+\lambda)$ and $(x_Q, y_Q) \leftarrow (x_Q-\lambda, y_Q-\lambda)$ because of the

Algorithm 3. Modified Shirase's Eta_T pairing algorithm for $GF(2^n)$.

1. Input: $P=(x_P, y_P)$, $Q=(x_Q,y_Q)$
2. Output: $C=\eta_T(P,Q)$
3. Prepare random values λ, λ' in $GF(2^n)$.
4. $G \leftarrow \lambda'$.
5. $\lambda_0 \leftarrow \lambda$, $\lambda_1 \leftarrow \lambda^2$, $a \leftarrow 1$. // (1***S***)
6. $w \leftarrow (x_P \cdot \lambda)$. // (1***M***)
7. $x_P \leftarrow x_P+\lambda_0$, $y_P \leftarrow y_P+\lambda_0$, $x_Q \leftarrow x_Q+\lambda_1$, $y_Q \leftarrow y_Q+\lambda_1$.
8. $x_P \leftarrow (x_P^2+1)$, $y_P \leftarrow (y_P^2+1)$, $w \leftarrow w^2$, $\lambda_0 \leftarrow \lambda_1$, $\lambda_1 \leftarrow \lambda_1^2$, $T \leftarrow (w+\lambda_1+x_Q \cdot \lambda_0)$. // (1***M***+4***S***)
9. $u \leftarrow y_Q+b+1$, $v \leftarrow (x_Q+1)$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
10. for $i \leftarrow 0$ to $(n-1)/2$ do
11. $A \leftarrow (\theta+u+T+y_P)+(x_P+v+1)s+t$.
12. $G \leftarrow G^2$. // (4***S***)
13. $G \leftarrow A \cdot G$. // (6***M***)
14. if $i < (n-1)/2$ do
15. $x_P \leftarrow x_P^4$, $y_P \leftarrow y_P^4$, $w \leftarrow w^4$, $v \leftarrow v+\lambda_0+1$, $\lambda_0 \leftarrow \lambda_0^4$, $\lambda_1 \leftarrow \lambda_1^4$, $T \leftarrow (\lambda_0 \cdot a+\lambda_1+w+x_Q \cdot \lambda_0)$, $a \leftarrow a+1$. //(1***M***+10***S***)
16. $u \leftarrow u+v+\lambda_0+1$, $v \leftarrow v+\lambda_0$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
17. end if
18. end for
19. $A \leftarrow A+(x_P^2+v+1)+s$. // (1***S***)
20. $C \leftarrow C \cdot A$. // (6***M***)
21. $C \leftarrow C^M$, where $M=(2^{2n}-1)(2^n+2^{(n+1)/2}+1)$.
22. return C.

characteristics of the Eta_T pairing over $GF(2^n)$. We apply the original RVA-based method to algorithm 2. The total cost of algorithm 3 is $(4n+11)\boldsymbol{M}+(7n+3)\boldsymbol{S}$. We now analyze the security flaws in algorithm 3 using the idea from [14]. The following steps, despite a DPA countermeasure, are still vulnerable to DPA.

1. At step 11, (x_P+v+1) if $i=0$.

The actual processing of (x_P+v+1) is $(x_P^2+x_Q+1)=((x_P^2+\lambda^2+1)+(x_Q+\lambda^2+1)+1)$ at step 11. We can implement the above process with the following assembly codes. We assume that the address registers such as X, Y, and Z hold addresses for $x_P'=(x_P^2+\lambda^2+1)$, the $x_Q'=(x_Q+\lambda^2)$ and the result of $(x_P'+x_Q')$.

```
A. Loop:
B.   tst R2
C.   breq Exit
D.   dec R2
E.   ld   R4, X+     // loading (xP^2+λ^2 +1)
F.   ld   R6, Y+     // loading (xQ+ λ^2)
G.   eor R4, R6      // R4 ← (xP^2+xQ+1)
H.   st   Z+, R4     // storing (xP^2+xQ+1)
I.   jmp Loop
J. Exit:
```

In the above codes, x_P' and x_Q' occupy $t=$ ceil(n/*WORDSIZE*) words in the memory because they are elements in $GF(2^n)$. Thus, the above code first loads x_P' and x_Q' from the memory to registers R_4 and R_6 (steps E and F) and then xor R_4 and R_6 (step G); finally, the result of xor (step H) is stored in the memory. This procedure is repeated t times by sequentially incrementing the displacements of memory addresses ($X+$, $Y+$, and $Z+$ automatically increment the displacements of their own addresses). During both step E and F, x_P^2 and x_Q are moved (masked with the random value λ^2) from the memory to the registers, and thus, attackers cannot obtain useful information. However, as a result of step G, λ^2 disappears. Thus, at step H, $(x_P^2+x_Q+1)$, which has no mask value, is stored at the memory via the bus. At this moment, attackers can obtain power traces related to x_P^2. Actually, because a large amount of power is consumed when data are moved over the bus, attackers can engage in DPA attacks by using the gathered power traces.

2. At step 11, $(\theta+u+T+y_P)$.

The result of $(\theta+u+T+y_P)$ has the form of $(x_P^2(x_Q+1)+y_P^2+x_Q+y_Q+b+1)$. This form does not have any random masking value. Even if this form is more complex to guess the actual value for attackers than the formula in step 11, meaningful information about both x_P^2 and y_P^2 is leaked when it is stored at memory through the data bus.

3. At step 13, $G \leftarrow A \cdot G$.

To compute $A \cdot G$, a multiplication over $GF(2^{4n})$, the value of A has to be loaded into the registers. However, the $A=(\theta+u+T+y_P)+(x_P+v+1)s+t$ is stored without being masked with any random value. Thus, when A is transferred from the memory to registers, the emitted power consumption is related to the actual value of A, particularly $(\theta+u+T+y_P)$ and (x_P+v+1). Thus, attackers launch DPA attacks as described earlier.

The aforementioned security flaws in algorithm 3 originate from the property in which the random masking is eliminated for correct pairing computations before step 13. For example, in RPC-based methods, because A in step 11 has the form of $\lambda \cdot A'$, the result of $A \cdot G$ has the same form as $\lambda \cdot A' \cdot G$. Thus, the use of the random λ does not affect the correct pairing value because the λ becomes one as a result of the final powering. However, for the RVA method, this is no longer valid because the result of $A \cdot G$ does not have a regular form. For this reason, random values at step 11 in algorithm 3 are removed in an automatic manner (x_P+v+1) or in a manual one $(\theta+u+T+y_P)$. The moments of processing, (x_P+v+1) or $(\theta+u+T+y_P)$, are protected against the DPA, but the results become unprotected because the masking values are removed for the correct pairing computation.

IV. Proposed Countermeasure

1. Design Principles of Secure RVA-Based Countermeasure

To make an RVA-based countermeasure valid for secure pairing algorithms, the following moments have to be protected with proper random masking values at the following times:

i) When transferring data from the memory to registers,

ii) When performing arithmetic on the central processing unit,

iii) When storing the results of arithmetic in the memory.

Algorithm 3 obeys only the second principle; it does not satisfy the first and third principles. To satisfy the first and third requirements, the results from step 11 in algorithm 3 have to have suitable random masking values such as $\bar{A}=(a_0+r_0)+(a_1+r_1)s+t$, where $A=a_0+(a_1)s+t$ and r_0, r_1 are random values. That is, in contrast to algorithm 3, which uses symmetric masking values, when masking input points P and Q, asymmetric random values should be applied. The proper use of asymmetric random values guarantees that the results of all computations contain suitable random masking values. However, the computation of $A \cdot G$ can no longer guarantee correct pairing values. Therefore, a newly developed RVA-based method has to not only satisfy the above requirements but also guarantee the computation of correct pairing values.

2. Proposed RVA-Based DPA Countermeasure

In this section, we present a new RVA-based method that obeys the aforementioned requirements and guarantees correct pairing computations.

A. Setting Initial Points

The proposed method makes use of asymmetric masking values instead of symmetric ones. In other words, the proposed method uses three masking values: λ, λ^2, and λ^4 (both λ^2 and λ^4 can be easily computed from λ). We set the input points $P=(x_P, y_P)$ and $Q=(x_Q, y_Q)$ as follows:

$$x_P \leftarrow x_P, \quad y_P \leftarrow y_P+\lambda+\lambda^2, \quad x_Q \leftarrow x_Q+\lambda^2, \quad y_Q \leftarrow y_Q+\lambda+\lambda^4,$$

where λ, λ^2, and λ^4 are random values in $GF(2^n)$ (actually, both λ^2 and λ^4 are computed from the random λ).

At the i-th iteration of the loop, the above equations can be expressed as

$$x_P^{(2i+1)} \leftarrow x_P^{(2i+1)}, \quad y_P^{(2i+1)} \leftarrow y_P^{(2i+1)}+\lambda^{(2i+1)}+\lambda^{(2i+2)},$$
$$x_Q \leftarrow x_Q+\lambda^{(2i+1)}, \quad y_Q \leftarrow y_Q+\lambda^{(2i)}+\lambda^{(2i+2)},$$

where $\lambda^{(2i)}$, $\lambda^{(2i+1)}$, and $\lambda^{(2i+2)}$ are random values in $GF(2^n)$.

Because different masking values are applied to x_P, y_P, x_Q, and y_Q, the masking values are not removed but maintained as a result of any computation sequences. We note that x_P does not have to be masked with a random value because in the computation of $\theta \leftarrow x_P \cdot v$, both x_P and v are unknown. Thus, attackers cannot guess the result of $x_P \cdot v$ because x_Q in v is masked with a random value whenever an Eta_T pairing algorithm is executed. This observation is supported by theorem 5 in [11].

Algorithm 4. Eta_T pairing algorithm over $GF(2^n)$ with the proposed DPA countermeasure.

1. Input: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$
2. Output: $C = \eta_T(P,Q)$
3. Prepare a nonzero random value λ in $GF(2^n)$.
4. $w \leftarrow (x_p \cdot \lambda)$. // (1***M***)
5. $\lambda_0 \leftarrow \lambda$, $\lambda_1 \leftarrow \lambda_0^2$, $\lambda_2 \leftarrow \lambda_1^2$, $\lambda_3 \leftarrow \lambda_1$. // (2***S***)
6. $x_P \leftarrow x_p^2+1$, $y_P \leftarrow (y_P+\lambda_1+\lambda_0)^2+1$, $w \leftarrow w^2$, $x_Q \leftarrow (x_Q+\lambda_1)$, $y_Q \leftarrow (y_Q+\lambda_2+\lambda_0)$. // (3***S***)
7. $u \leftarrow y_Q+b+1$, $v \leftarrow (x_Q+1)$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
8. $G \leftarrow (\theta+u+w+y_P)+(x_P+v+1)s+t$.
9. $G \leftarrow \lambda_0 \cdot G+\lambda_1+(\lambda_1 \cdot \lambda_0)s$. // (3***M***)
10. $x_P \leftarrow x_P^4$, $y_P \leftarrow y_P^4$, $T \leftarrow \lambda_0+\lambda_1$, $\lambda_3 \leftarrow \lambda_1$, $\lambda_0 \leftarrow \lambda_2$, $\lambda_1 \leftarrow \lambda_2^2$, $\lambda_2 \leftarrow \lambda_1^2$, $\lambda_3 \leftarrow \lambda_3+\lambda_1$, $w \leftarrow w^4$, $T \leftarrow T+\lambda_2$. // (8***S***)
11. $u \leftarrow u+v+1+T$, $v \leftarrow v+1+\lambda_3$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
12. for $i \leftarrow 1$ to $(n-1)/2$ do
13. $A \leftarrow (\theta+u+w+y_P)+(x_P+v+1)s+t$.
14. $G \leftarrow G^2$. // (4***S***)
15. $G \leftarrow A \cdot G$. // Call the algorithm 4 with (6***M***+2***S***)
16. if $i < (n-1)/2$ then
17. $x_P \leftarrow x_P^4$, $y_P \leftarrow y_P^4$, $T \leftarrow T^4$, $\lambda_3 \leftarrow \lambda_1$, $\lambda_0 \leftarrow \lambda_2$, $\lambda_1 \leftarrow \lambda_2^2$, $\lambda_2 \leftarrow \lambda_1^2$, $\lambda_3 \leftarrow \lambda_3+\lambda_1$, $w \leftarrow w^4$. // (10***S***)
18. $u \leftarrow u+v+1+T$, $v \leftarrow v+1+\lambda_3$, $\theta \leftarrow x_P \cdot v$. // (1***M***)
19. end if
20. end for
21. $A \leftarrow A+(x_P^2+v+\lambda_1+1)+s$. // (1***S***)
22. $G \leftarrow G \cdot A$. // (6***M***+2***S***)
23. $G \leftarrow G^M$, where $M=(2^{2n}-1)(2^n+2^{(n+1)/2}+1)$.
24. return G.

Theorem 1. For x_P, y_P, x_Q, and y_Q, the computation of $\bar{A} \leftarrow (y_P+\theta+u+w)+(x_P+v+1)s+t$ satisfies the design principles for securing an RVA-based method if $\lambda^{(2i)}$, $\lambda^{(2i+1)}$, and $\lambda^{(2i+2)}$ are uniformly distributed random values and they are statistically independent from x_P, y_P, x_Q, and y_Q.

Proof. To prove that the proposed method satisfies the design principles, we have to show that $\bar{A} \leftarrow (y_P+\theta+u+w)+(x_P+v+1)s+t$ maintains the masking values after processing. Thus, we show that $\bar{A}=(y_P+\theta+u+w)+(x_P+v+1)s+t$ has the form $\bar{A}=(a_0+\lambda)+(a_1+\lambda^2)s+t$ if the original A is represented as $a_0+(a_1)s+t$.

Here, $\bar{A}=(y_P+\theta+u+w)+(x_P+v+1)s+t$ has four forms according to the loop counter i ($i \equiv 0, 1, 2,$ and $3 \bmod 4$). With respect to the forms $x_P^{(2i+1)} \leftarrow x_P^{(2i+1)}$, $y_P^{(2i+1)} \leftarrow (y_P^{(2i+1)}+\lambda^{(2i+1)}+\lambda^{(2i+2)})$, $x_Q \leftarrow (x_Q+\lambda^{(2i+1)})$, $y_Q \leftarrow (y_Q+\lambda^{(2i)}+\lambda^{(2i+2)})$, $w^{(2i+1)} \leftarrow (x_P \cdot \lambda)^{(2i+1)}$ ($\lambda^{(2i)}$ is a uniformly distributed random value, and $\lambda^{(2i+1)}$ and $\lambda^{(2i+2)}$ are computed from this. $\lambda^{(2i+1)}$ and $\lambda^{(2i+2)}$ are also uniformly distributed since the result of squaring over $GF(2^n)$ is a permutation. In addition, they are statistically independent from input data $x_P^{(2i+1)}$, $y_P^{(2i+1)}$, x_Q, and y_Q), $\bar{A}$ has the following four forms: [1]

i) Case $i \equiv 0 \bmod 4$.
$(x_P^{(2i+1)}x_Q+x_P^{(2i+1)}+x_Q+y_P^{(2i+1)}+y_Q+w^{(2i+1)}+b+1)$
$+(x_P^{(2i+1)}+x_Q+1)s+t$,

ii) Case $i \equiv 1 \bmod 4$.
$(x_P^{(2i+1)}x_Q+y_P^{(2i+1)}+y_Q+w^{(2i+1)}+b)+(x_P^{(2i+1)}+x_Q)s+t$,

iii) Case $i \equiv 2 \bmod 4$.
$(x_P^{(2i+1)}x_Q+x_P^{(2i+1)}+x_Q+y_P^{(2i+1)}+y_Q+w^{(2i+1)}+b)$
$+(x_P^{(2i+1)}+x_Q+1)s+t$,

iv) Case $i \equiv 3 \bmod 4$.
$(x_P^{(2i+1)}x_Q+y_P^{(2i+1)}+y_Q+w^{(2i+1)}+b+1)+(x_P^{(2i+1)}+x_Q)s+t$.

If i is zero, then the first formula is $(x_P^2 \cdot x_Q+x_P^2+y_P^2+x_Q+y_Q+\lambda+b+1)+(x_P^2+x_Q+\lambda^2+1)s+t$, whereas its original formula is $(x_P^2 \cdot x_Q+x_P^2+y_P^2+x_Q+y_Q+b+1)+(x_P^2+x_Q+1)s+t$. If i is k ($k \equiv 0 \pmod 4$), then the formula is $(x_P^{(2k+1)}x_Q+x_P^{(2k+1)}+y_P^{(2k+1)}+x_Q+y_Q+\lambda^{(2k)}+b+1)+(x_P^{(2k+1)}+x_Q+\lambda^{(2k+1)}+1)s+t$,

1) $x_P^{(2i+1)}x_Q$ in the following four forms leaks information when the value of either $x_P^{(2i+1)}$ or maksed value x_Q is zero according to [16]. However, because $x_P^{(2i+1)}$ is fixed and secret value and the random value $\lambda^{(2i+1)}$ for masking x_Q is not controllable by attackers, it is impossible to launch DPA against our method even if some information may leaks when the value of either $x_P^{(2i+1)}$ or x_Q is zero.

whereas its original form is $(x_P^{(2k+1)}\cdot x_Q+x_P^{(2k+1)}+y_P^{(2k+1)}+x_Q+y_Q+b+1)+(x_P^{(2k+1)}+x_Q+1)s+t$.

Finally, in the case of k+4, it is $(x_P^{(2k+9)}\cdot x_Q+x_P^{(2k+9)}+y_P^{(2k+9)}+x_Q+y_Q+\lambda^{(2k+8)}+b+1)+(x_P^{(2k+9)}+x_Q+\lambda^{(2k+9)}+1)s+t$, whereas its original formula is $(x_P^{(2k+9)}\cdot x_Q+x_P^{(2k+9)}+y_P^{(2k+9)}+x_Q+y_Q+b+1)+(x_P^{(2k+9)}+x_Q+1)s+t$. Thus, it is clear from the induction steps that the first formula for the case of $i \equiv 0 \pmod 4$ has a regular form composed of masking values. We can prove the remaining formulas in the similar manner.

We now prove that whereas the order of computations affects the security of algorithm 3, it does not affect that of the proposed method. Of the above formulas, the first one can be expressed as the addition of the random variables:

$$(\underbrace{x_P^{(2i+1)}x_Q}_{D}+\underbrace{x_P^{(2i+1)}}_{E}+\underbrace{x_Q}_{A}+\underbrace{y_P^{(2i+1)}}_{B}+\underbrace{y_Q}_{C}+\underbrace{w^{(2i+1)}}_{F}+b+1)$$
$$+(x_P^{(2i+1)}+x_Q+1)s+t.$$

We now show that changing the sequence of A, B, and C in the above formula does not lead to the elimination of masking values. Note that D, E, and F have no effect on the presence of masking values because E does not contain any masking value and both D and F have a form in which they are multiplied by masking values. We can consider following three sequences:

$(A+B)+C=(x_Q+\lambda^{(2i+1)}+y_P^{(2i+1)}+\lambda^{(2i+1)}+\lambda^{(2i+2)})+y_Q+\lambda^{(2i)}+\lambda^{(2i+2)}$,
$(B+C)+A=(y_P^{(2i+1)}+\lambda^{(2i+1)}+\lambda^{(2i+2)}+y_Q+\lambda^{(2i)}+\lambda^{(2i+2)})+x_Q+\lambda^{(2i+1)}$,
$(A+C)+B=(x_Q+\lambda^{(2i+1)}+y_Q+\lambda^{(2i)}+\lambda^{(2i+2)})+y_P^{(2i+1)}+\lambda^{(2i+1)}+\lambda^{(2i+2)}$.

Based on the above explanations, we can see that masking values are maintained at all the times, including during computations and data transfers/storage. Furthermore, the input data are all masked with statistically independent random values $\lambda^{(2i)}$ and $\lambda^{(2i+1)}$, the result of $\bar{A}\leftarrow(y_P+\theta+u+w)+(x_P+v+1)s+t$ is also statistically independent from the input data. Therefore, the proposed method satisfies the previously presented design principles. □

B. Compensation Method

As a result of computing $\bar{A} \leftarrow (y_P+\theta+u+w)+(x_P+v+1)s+t$, $\bar{A}$ has the form of $(a_0+\lambda_0)+(a_1+\lambda_1)s+t$ if we assume that the original form is $a_0+(a_1)s+t$. In the original RVA method, masking values are removed before computing $A\cdot G$ to compute the correct pairing value, which results in security flaws. On the other hand, in the proposed method, the compensation procedure is conducted during the computation of $A\cdot G$, not before. The aim of our compensation method is to make $\bar{A}\cdot G=A\cdot G$. In order words, it is responsible for the computation of $A\cdot G$ by using $\bar{A}\cdot G$ in order to guarantee the correct pairing value. Typically, $H \leftarrow A\cdot G$ can be computed as follows by using the Karatsuba method; it requires 6***M*** in $GF(2^n)$, whereas a naive method uses 8***M*** in $GF(2^n)$.

We propose a Karatsuba-like multiplication algorithm combined with a compensation mechanism. In the following proposed method, we assume that the terms (g_0, g_1, g_2, and g_3) in G are random variables. That is, our method uses g_0, g_1, g_2, and g_3 as a type of random masking values.

$A=(a_0+\lambda_0)+(a_1+\lambda_1)s+t$, $G=(g_0)+(g_1)s+(g_2)t+(g_3)st$.
$h_0 \leftarrow (a_0+g_1)g_0+(a_1+g_0)g_1+g_3$,
$h_1 \leftarrow (a_0+a_1+g_0)(g_0+g_1)+(a_0+g_1)g_0+g_2+g_3+g_0^2$,
$h_2 \leftarrow (a_0+g_2+g_3)g_2+(a_1+g_2)g_3+g_0+g_2+g_2^2$,
$h_3 \leftarrow (a_0+a_1+g_2)(g_2+g_3)+(a_0+g_2+g_3)g_2+g_1+g_3$.

In the proposed method, because the input A has the form of $(a_0+\lambda_0)+(a_1+\lambda_1)s+t$, we need to properly replace masking values (λ_0, λ_1) with (g_0, g_1, g_2, g_3). For example, we compute a_0+g_1 in the computation of h_0 with $(a_0+\lambda_0)+T$, where $T=\lambda_0+g_1$. We incorporate this into our multiplication method, algorithm 4. The proposed multiplication algorithm requires only (6***M***+2***S***) in $GF(2^n)$. Because the squaring operation in $GF(2^n)$ can be efficiently computed with table lookups, our algorithm has little effect on the overall performance of the algorithm.

Algorithm 5. Multiplication algorithm over $GF(2^{4n})$ with correction mechanism.

1. Input: $F=(f_0')+(f_1')s+t=(f_0+\lambda_0)+(f_1+\lambda_1)s+t$, $G=g_0+g_1s+g_2t+g_3st$, λ_0, λ_1.
2. Output: $H=h_0+h_1s+h_2t+h_3st=F\cdot G$.
3. $M\leftarrow G$ ($M=m_0+m_1s+m_2t+m_3st$).
4. $A_0\leftarrow\lambda_0+\lambda_1$, $A_1\leftarrow f_0'+f_1'$, $A_2\leftarrow m_2+m_3$.
5. $T_0\leftarrow\lambda_0+m_1$, $T_0\leftarrow T_0+f_0'$. // Computing h_0
6. $h_0\leftarrow T_0\cdot m_0$, $h_1\leftarrow h_0$. // (1***M***)
7. $T_0\leftarrow m_0+\lambda_1$, $T_0\leftarrow T_0+f_0'$.
8. $T_0\leftarrow T_0\cdot m_1$. // (1***M***)
9. $T_0\leftarrow T_0+m_3$.
10. $h_0\leftarrow h_0+T_0$.
11. $T_0\leftarrow m_0+A_0$, $T_0\leftarrow T_0+A_1$. // Computing h_1
12. $T_1\leftarrow m_0+m_1$.
13. $T_0\leftarrow T_0\cdot T_1$. // (1***M***)
14. $h_1\leftarrow h_1+T_0$, $h_1\leftarrow h_1+A_2$.
15. $T_0\leftarrow m_0^2$. // (1***S***)
16. $h_1\leftarrow h_1+T_0$.
17. $T_0\leftarrow A_2+\lambda_0$, $T_0\leftarrow T_0+f_0'$. // Computing h_2
18. $h_2\leftarrow T_0\cdot m_2$, $h_3\leftarrow h_2$. // (1***M***)
19. $T_0\leftarrow m_2+\lambda_1$, $T_0\leftarrow T_0+f_1'$.
20. $T_0\leftarrow T_0\cdot m_3$. // (1***M***)
21. $h_2\leftarrow h_2+T_0$, $h_2\leftarrow h_2+m_0$, $h_2\leftarrow h_2+m_2$.
22. $T_0\leftarrow m_2^2$. // (1***S***)
23. $h_2\leftarrow h_2+T_0$.
24. $T_0\leftarrow m_2+A_0$, $T_0\leftarrow T_0+A_1$. // Computing h_3
25. $T_0\leftarrow T_0\cdot A_2$. // (1***M***)
26. $h_3\leftarrow h_3+T_0$, $h_3\leftarrow h_3+m_1$, $h_3\leftarrow h_3+m_3$.
27. return H.

Theorem 2. The proposed multiplication algorithm is secure against DPA if g_0, g_1, g_2, and g_3 are uniformly distributed and mutually statistically independent random.

Proof. In our method, only a_0 and a_1 are not random variables, whereas all the others λ_0, λ_1, g_0, g_1, g_2, and g_3 are random variables. Actually, g_0, g_1, g_2, and g_3 are uniformly distributed and mutually statistically independent because they are randomized with multiplicative masking value by step 9 in algorithm 4. In other words, since g_0, g_1, g_2, and g_3 are mutually independent and they are masked with uniformly distributed random value, masked g_0, g_1, g_2, and g_3 are uniformly distributed and mutually statistically independent random variables according to [17] (The way of randomizing G is explained in the next section). Now, we need to ensure that the proposed multiplication method does not leak any information on a_0 and a_1. First, we show that a_0, a_1, or (a_0+a_1) is securely masked with random variables. There are six calculations related to a_0, a_1, or (a_0+a_1): (a_0+g_1), (a_1+g_0), $(a_0+a_1+g_0)$, $(a_0+g_2+g_3)$, (a_1+g_2), and $(a_0+a_1+g_2)$. Thus, we can see that a_0, a_1, and (a_0+a_1) are properly masked with uniformly distributed and mutually statistically independent random masking values. Because other terms except for these calculations are random variables, we do not have to consider their leakage.

We now have to consider the information leaked during updating $(a_0+\lambda_0)$ and $(a_1+\lambda_1)$ to one of (a_0+g_1), (a_1+g_0), $(a_0+a_1+g_0)$, $(a_0+g_2+g_3)$, (a_1+g_2), and $(a_0+a_1+g_2)$.

Our updating procedure is as follows:

(1) $(a_0+g_1) \leftarrow (a_0+\lambda_0)+T$, where $T=(g_1+\lambda_0)$.
(2) $(a_1+g_0) \leftarrow (a_1+\lambda_1)+T$, where $T=(g_0+\lambda_1)$.
(3) $(a_0+a_1+g_0) \leftarrow ((a_0+\lambda_0)+(a_1+\lambda_1))+T$, where $T=(g_0+\lambda_0+\lambda_1)$.
(4) $(a_0+g_2+g_3) \leftarrow (a_0+\lambda_0)+T$, where $T=(g_2+g_3+\lambda_0)$.
(5) $(a_1+g_2) \leftarrow (a_1+\lambda_1)+T$, where $T=(g_2+\lambda_1)$.
(6) $(a_0+a_1+g_2) \leftarrow ((a_0+\lambda_0)+(a_1+\lambda_1))+T$, where $T=(g_2+\lambda_0+\lambda_1)$.

In other words, we first configure the random value T that randomizes a_0 and a_1 as a result of $(a_0+\lambda_0)+T$, $(a_1+\lambda_1)+T$, or $((a_0+\lambda_0)+(a_1+\lambda_1))+T$ while removing the presence of λ_0 or λ_1. Thus, our multiplication method does not leak any information on a_0 and a_1. □

C. Randomizing G

In the proposed compensation process, we assume that elements (g_0, g_1, g_2, and g_3) in G are uniformly distributed and statistically independent random values. In this section, we describe how G can be randomized for the compensation method. We modify the first iteration process in the loop to randomize G:

(1) $A \leftarrow (a_0+\lambda)+(a_1+\lambda^2)s+t$,

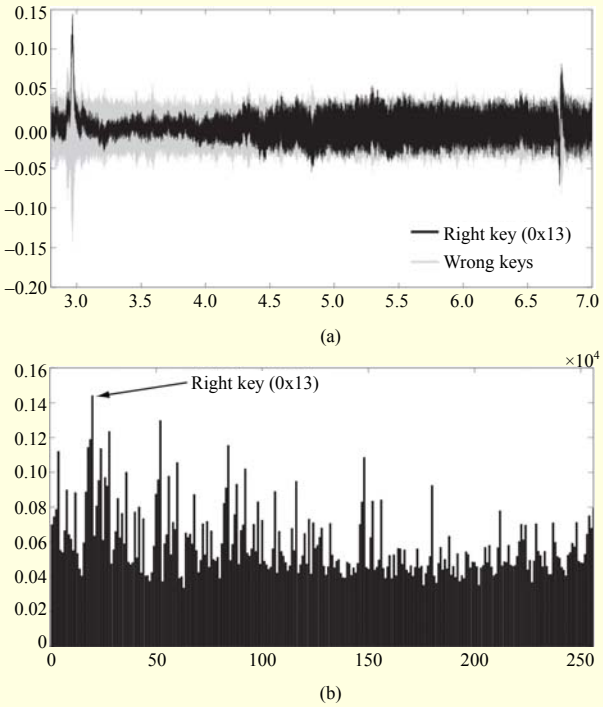

Fig. 1. CPA against (x_P+v+1) in step 7 of algorithm 1: a) correlation traces and b) maximum correlations for each key guess.

(2) $\bar{A} \leftarrow \lambda A+\lambda^2+\lambda^3 s = ((\lambda a_0+\lambda^2)+(\lambda a_1+\lambda^3)s+\lambda t)+\lambda^2+\lambda^3 s$
$= \lambda a_0+\lambda a_1 s+\lambda t$,
(3) $G \leftarrow (\bar{A})^2 = (a_0^2\lambda^2+a_1^2\lambda^2)+(a_1^2\lambda^2+\lambda^2)s+\lambda^2 t$
$= \lambda^2((a_0^2+a_1^2)+(a_1^2+1)s+t)$.

Steps 1 and 3 are exactly the same as the process for the first loop iteration (which requires additional $3\boldsymbol{M}+1\boldsymbol{S}$ to randomize G). We can now use the value of G value as random variables because its form is $G=\lambda^2((a_0^2+a_1^2)+(a_1^2+1)s+t)$. Algorithm 4 is the proposed Eta_T pairing algorithm equipped with the newly designed RVA-based DPA countermeasure. Its computational cost is $(3.5n+7.5)\boldsymbol{M}+(8n-2)\boldsymbol{S}$. The process from step 4 to step 6 is responsible for generating random masking values and updating initial points. Step 9 converts the form of G from $((g_0+\lambda)+(g_1+\lambda^2)s+t)$ to $((\lambda \cdot g_0)+(\lambda \cdot g_1)s+\lambda t)$. Through steps 9 and 10, algorithm 4 uses the terms of G as random variables in step 15 (algorithm 5). Through the proposed method, the final pairing value before computing final powering is randomized with certain multiplicative masking value. Through the final powering step, this masking value becomes 1. Thus, our algorithm guarantees the correct Eta_T pairing computation.

V. Comparisons and Experimental Results

In this section, we compare the performance and security of our method with those of previously proposed countermeasures.

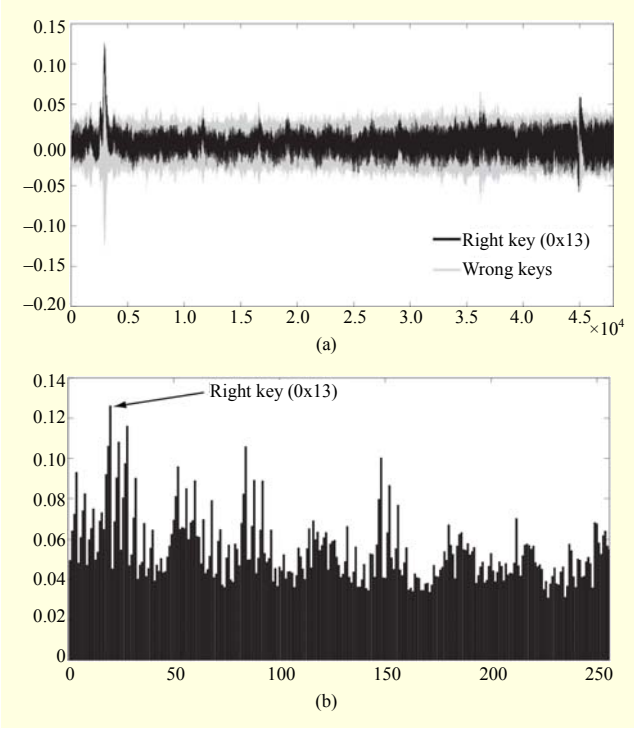

Fig. 2. CPA against (x_P+v+1) in step 11 of algorithm 3.

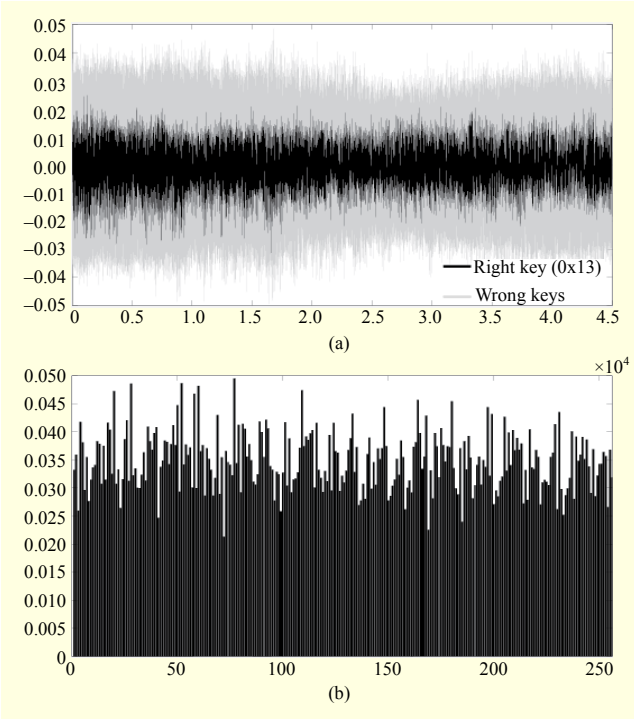

Fig. 3. CPA against (x_P+v+1) in step 13 of algorithm 4.

1. Security of Pairing Computation Algorithms

We conducted a first-order power analysis of three algorithms: the original Eta_T pairing algorithm (algorithm 1), Shirase's method (algorithm 3), and the proposed method (algorithm 4).[2)] This experiment was conducted not only to verify the security of the proposed method but also to demonstrate that both algorithm 1 and algorithm 3 are vulnerable to power analysis attacks in a practical manner, even though we have demonstrated their vulnerability theoretically.

To simplify the security evaluation, we targeted the process of (x_P+v+1) in step 7 of algorithm 1, step 11 of algorithm, and step 13 of algorithm 4. We executed a CPA attack for the result of (x_P+v+1) when it was transferred from the register to the memory. We considered x_P as the fixed key and guessed its value from the least significant byte (LSB) to most significant byte.

Figures 1, 2, and 3 show the results of the 8-bit CPA attack on algorithms 1, 3, and 4, respectively. We obtained the correct value, that is, the LSB of x_P, for algorithms 1 and 3, but we do not observe high correlation peaks for algorithm 4. Actually, the first peak and second peak shown in upper figures of Figs. 1 and 2 are generated at the moments of storing the result of

Table 1. Additional cost of DPA countermeasures for the Eta_T pairing over $GF(2^n)$ (***I***, ***M***, and ***S*** denote finite field inversion, multiplication, and squaring operations, respectively).

	Basis algorithm	Performance overhead
Page [8]	Randomized private value	$(2n)\boldsymbol{M}+(9n)\boldsymbol{S}+(n)\boldsymbol{I}$
Page [8]	Blinding public value	$(3.5n+72.5)\boldsymbol{M}+(4n+4)\boldsymbol{C}$
Scott [9]	Randomized intermediate value	$(3.5n+7.5)\boldsymbol{M}$
Kim [11]	Randomized projective coordinate	$(2.5n+6.5)\boldsymbol{M}+(0.5n+0.5)\boldsymbol{S}$
Choi [12]		$(2n+6)\boldsymbol{M}+(0.5n+0.5)\boldsymbol{S}$
Proposed	Random value addition	$(4)\boldsymbol{M}+(4n-1)\boldsymbol{S}$

Table 2. Additional cost estimates of DPA countermeasures for Eta_T pairing over $GF(2^{239})$.

	8-bit ATmega128 ($\boldsymbol{M}/\boldsymbol{S} \approx 8$)	16-bit MSP430, 32-bit PXA27x ($\boldsymbol{M}/\boldsymbol{S} \approx 11$)
Kim [11]	619***M***	614.9***M***
Choi [12]	499***M***	494.9***M***
Proposed	123.375***M***	90.81***M***

2) Second-order (SO) DPA analysis makes use of two different samples in the power consumption trace corresponding to two different intermediate values computed during the execution of an algorithm [18]. If two different intermediate values use same masking value, the algorithm can be analyzed like ordinary DPA. We expect that algorithm 4 may be vulnerable to SO DPA analysis since it principally uses Boolean masking value (Of course, it applies multiplicative masking value through algorithm 5). Thus, our method can be analyzed by SO type analysis similarly to that symmetric key cryptosystems equipped with Boolean masking method. The aim of our method is just to defend the first-order DPA in the process of *EtaT* pairing over $GF(2^n)$, thus, defending SO type analysis is beyond the scope of this paper.

(x_P+v+1) in the main memory and transferring it from the memory to the register for computing $G \leftarrow A \cdot G$.

2. Performance of Pairing Computation Algorithms

As shown in Table 1, the proposed algorithm does not entail heavy computational overhead; its additional cost was only 4***M***+(4n−1)***S***. Because the squaring operation in $GF(2^n)$ can be computed with efficient table lookups, the proposed algorithm is expected to be superior to existing RPC-based countermeasures in terms of computational efficiency. In the case of n=239, the overhead of the proposed algorithm was (4***M***+955***S***); Kim's overhead was (604***M***+120***S***), and Choi's overhead was (484***M***+120***S***). Using the results for the software implementation in [19] and [20], we can estimate the total overhead for the case of n=239. Table 2 provides estimates of the computational costs for the three countermeasures based on the results of [19], [20]. Because the cost of algorithm 1 was (3.5n+3.5)***M***+(4n−1)***S***, the computational costs of the algorithm on the 8-bit ATmega128 and 16-bit MSP430 processors were (959.375)***M*** and (926.81)***M***, respectively. The ratios of the performance overhead for the proposed algorithm on the 8-bit ATmega128 and 16-bit MSP430 processors were 12.85% and 9.79%, respectively, whereas those for [12] (resp. [11]) were 52.01% (resp. 64.52%) and 53.39% (resp. 66.34%). The results of the experimental studies suggest that the proposed algorithm outperforms the existing DPA countermeasures in terms of securing Eta_T pairing computation over $GF(2^n)$.

VI. Conclusion

This paper proposes an efficient RVA-based DPA countermeasure for securing Eta_T pairing algorithms over $GF(2^n)$. In this paper, we demonstrated the security vulnerability of Shirase's RVA-based countermeasure and propose some guidelines for making RVA-based methods more secure, on which the proposed RVA-based countermeasure is based. The results of the experimental studies indicate that whereas Shirase's method is vulnerable to power analysis attacks, for example, CPA, the proposed algorithm is secure against such attacks. Further, the proposed algorithm outperforms not only existing RPC-based countermeasures but also other types of countermeasures in terms of computational efficiency. As such, the proposed algorithm is expected to be beneficial for embedded devices with limited computing ability, such as sensor motes, mobile computing devices, and smartcards.

References

[1] P.S.L.M. Barreto et al., "Efficient Algorithms for Pairing-Based Cryptosystems," *CRYPTO, LNCS 2442*, 2002, pp. 354-368.

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

[3] 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.

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

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

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

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

[8] D. Page and F. Vercauteren, "Fault and Side-Channel Attacks on Pairing Based Cryptography," *Cryptology ePrint Archive*, Report 2004/283, 2005.

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

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

[11] T.H. Kim et al., "Power Analysis Attacks and Countermeasures on η_T Pairing over Binary Fields," *ETRI J.*, vol. 30, no. 1, 2008, pp. 68-80.

[12] D.H. Choi, D.-G. Han, and H. W. Kim, "Construction of Efficient and Secure Pairing Algorithm and Its Application," *J. Commun. Netw.*, vol. 10, no. 4, 2008, pp. 437-443.

[13] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," *CHES, LNCS 3156*, 2004, pp. 135-152.

[14] M. Shirase, T. Takagi, and E. Okamoto, "An Efficient Countermeasure against Side Channel Attacks for Pairing Computation," *ISPEC, LNCS 4991*, 2008, pp. 290-303.

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

[16] J.D. Golić and C. Tymen, "Multiplicative Masking and Power Analysis of AES," *CHES, LNCS 2523*, 2003, pp. 198-212.

[17] J.D. Golić, "Techniques for Random Masking in Hardware," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 2, 2007, pp. 291-300.

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

[19] L.B. Oliveira et al., "TinyPBC: Pairings for authenticated Identity-Based Non-interactive Key Distribution in Sensor Networks," *Elsevier, Computer Communications,* vol. 34, 2011, pp. 485-493.

[20] S.C. Seo et al., "TinyECCK: Efficient Elliptic Curve Cryptography Implementation over $GF(2^m)$ on 8-Bit Micaz

Mote," *IEICE Trans.Syst.*, vol. 91-D, no. 5, 2010, pp. 1338-1347.

Seog Chung Seo received the BS in information and computer engineering from Ajou University, Suwon, Rep. of Korea, and the MS in information and communications from Gwangju Institute of Science and Technology (GIST), Gwangju, Rep. of Korea, in 2005 and 2007, respectively. Since 2007, he has been working toward the PhD at the Graduate School of Information Management and Security, Korea University, Seoul, Rep. of Korea. His research interests include public-key cryptography and its efficient implementations on various IT devices.

Dong-Guk Han received his BS and MS 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 postdoctoral researcher at Future University, Hakodate, Japan. After finishing the doctoral course, he was an exchange student in the Department of Computer Science and Communication Engineering in Kyushu University in Japan from April 2004 to March 2005. He was a senior researcher in ETRI, Daejeon, Korea. He is currently working as an assistant professor with the Department of Mathematics of Kookmin University, Seoul, Rep. of Korea. He is a member of KIISC, IEEK, and IACR.

Seokhie Hong received his MA and PhD in mathematics from Korea University in 1997 and 2001, respectively. He worked for SECURITY Technologies Inc. from 2000 to 2004. From 2004 to 2005, he worked as a postdoctoral researcher with COSIC at KU Leuven, Belgium. Since 2005, he has been with Korea University, where he is now working in the Graduate School of Information Management and Security. His specialty lies in the area of information security, and his research interests include the design and analysis of symmetric-key cryptosystems, public-key cryptosystems, and forensic systems.