PAPER

Secure Elliptic Curve Exponentiation against RPA, ZRA, DPA, and SPA***

Hideyo MAMIYA[†*], *Nonmember*, **Atsuko MIYAJI**[†a)], *Member*, *and* **Hiroaki MORIMOTO**[†**], *Nonmember*

SUMMARY In the execution on a smart card, side channel attacks such as the simple power analysis (SPA) and the differential power analysis (DPA) have become serious threat. Side channel attacks monitor the side channel information such as power consumption and even exploit the leakage information related to power consumption to reveal bits of a secret key d although d is hidden inside a smart card. Almost public key cryptosystems including RSA, DLP-based cryptosystems, and elliptic curve cryptosystems execute an exponentiation algorithm with a secret-key exponent, and they thus suffer from both SPA and DPA. In the case of elliptic curve cryptosystems, DPA is improved to the refined power analysis (RPA), which exploits a special point with a zero value and reveals a secret key. RPA is further generalized to zero-value register attack (ZRA). Both RPA and ZRA utilize a special feature of elliptic curves that happens to have a special point or a register used in addition and doubling formulae with a zero value and that the power consumption of 0 is distinguishable from that of a non-zero element. To make the matters worse, some previous efficient countermeasures to DPA are neither resistant to RPA nor ZRA. This paper focuses on elegant countermeasures of elliptic curve exponentiations against RPA, ZRA, DPA and SPA. Our novel countermeasure is easily generalized to be more efficient algorithm with a pre-computed table.

key words: *elliptic curve exponentiation, ZPA, RPA, DPA, SPA*

1. Introduction

1.1 Elliptic Curve Cryptosystems

Koblitz [20] and Miller [26] proposed a method by which public key cryptosystems can be constructed on the group of points of an elliptic curve over a finite field. If elliptic curve cryptosystems satisfy both MOV-conditions [25] and FR-conditions [9], and avoid p-divisible elliptic curves over $\mathbb{F}_{p^r}$ [2], [35], [36], then the only known attacks are the Pollard $\rho-$method [32] and the Pohlig-Hellman method [31]. Hence with current knowledge, we can construct elliptic curve cryptosystems over a smaller definition field than the discrete-logarithm-problem (DLP)-based cryptosystems like the ElGamal cryptosystems [11] or the DSA [10] and RSA cryptosystems [33]. Elliptic curve cryptosystems with a 160-bit key are thus believed to have the same security as both the ElGamal cryptosystems and RSA with a 1,024-bit key. This is why elliptic curve cryptosystems have been attractive in smart card applications, whose memory storage and CPU power are very limited. Elliptic curve cryptosystems execute an exponentiation algorithm of dP for a secret key d and a publicly known P as a cryptographic primitive. Thus, the efficiency of elliptic curve cryptosystems on a smart card depends on the implementation of exponentiation.

1.2 Overview of RPA and ZRA

Side channel attacks, first introduced in [21], [22], monitor power consumption and even exploit the leakage information related to power consumption to reveal bits of a secret key d although d is hidden inside a smart card. There are two types of power analysis, the simple power analysis (SPA) and the differential power analysis (DPA). SPA makes use of such an instruction performed during an exponentiation algorithm that depends on the data being processed. DPA uses correlation between power consumption and specific key-dependent bits. The address-bit DPA (ADPA) [14], which is one of DPA, uses the leaked information from the address bus and can be applied on such algorithms that fix the address bus during execution. It is a serious issue that the implementation should be resistant to SPA and DPA, and many countermeasures have been proposed in [4], [5], [15], [19], [22], [27], [28], [30]. We may note here that almost public key cryptosystems including RSA and DLP-based cryptosystems also execute an exponentiation algorithm with a secret-key exponent, and, thus, they also suffer from both SPA and DPA in the same way as elliptic curve cryptosystems. However, in the case of elliptic curve cryptosystems, DPA is further specialized to the refined power analysis (RPA) by [12], which exploits a special point with a zero value and reveals a secret key. An elliptic curve happens to have a special point $(0, y)$ or $(x, 0)$, which can be controlled by an adversary because the order of basepoint is usually known. RPA utilizes such a feature that the power consumption of 0 is distinguishable from that of an non-zero element. Although elliptic curve cryptosystems are vulnerable to RPA, RPA are not applied to RSA or DLP-based cryptosystems because they don't have such a special zero element. Furthermore, RPA is generalized to zero-value register attack (ZRA) by [3]. ZRA utilizes a special feature of elliptic curves that addition and doubling formulae need a lot of each different operations stored in auxiliary registers, one of which happens to become 0. Not all elliptic

Manuscript received September 28, 2005.
Manuscript revised February 17, 2006.
Final manuscript received May 10, 2006.

[†]The authors are with Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.
[*]Presently, with Hitachi System and Services, Ltd.
[**]Presently, with the Japan Self-Defense Forces.

[***]A preliminary version was presented at ISEC 2004-03 and CHES 2004.

a) E-mail: miyaji@jaist.ac.jp

DOI: 10.1093/ietfec/e89–a.8.2207

curves are vulnerable against RPA or ZRA, but some curves in [34] are vulnerable against theses attacks. To make the matters worse, some previous efficient countermeasures of the randomized-projective-coordinate method (RPC) [8] or the randomized-curve method (RC) [19] are neither resistant to RPA nor ZRA.

1.3 Our Contributions

This paper focuses on countermeasures against both RPA and ZRA, which are also resistant to both SPA and DPA. Our countermeasure makes use of a random-initial-point method (RIP): choose a random point R, compute $dP + R$, subtract R, and get dP. By using a random initial point at each execution of exponentiation, any point or any register used in addition formulae changes at each execution. Thus, it is resistant to DPA, RPA, and ZRA because an attacker cannot control a point P itself as he needs. In order to be secure against SPA, we have to compute $dP + R$ in such a way that it does not have any branch instruction dependent on the data being processed. The simple way would be to compute $dP + R$ from LSB in the add-and-double-always algorithm [16], which is called LRIP in this paper: change an initial value O to R in the binary algorithm from LSB for the binary representation of $d = (d_{n-1}, \cdots, d_0)_2$,

$$dP+R=R+d_0P+d_12P+d_22(2P)+\cdots+d_{n-1}2(2^{n-2})P,$$

which is combined with the add-and-double-always algorithm. However, the computation of $dP + R$ from MSB (see Algorithm 1) is not straightforward: if we change an initial value O to R in the binary algorithm from MSB, then it computes $2^{n-1}R + dP$, and we thus have to subtract $2^{n-1}R$ to get dP. It needs more work than LRIP.

Our remarkable idea lies in the computation algorithm of $dP + R$ that uses the binary algorithm from MSB and not LSB and is resistant to SPA. The binary algorithm from MSB has an advantage over that from LSB in that it is more easily generalized to a sophisticated algorithm with a pre-computed table like the window algorithm [24] or the extended binary algorithm [38]. In this paper, we first show the basic SPA-resistant algorithm of $dP + R$ that uses the binary representation from MSB. This is called BRIP in this paper. Next we apply the extended binary algorithm [38] and present more efficient SPA-resistant algorithm of $dP+R$ with a pre-computed table. This is called EBRIP in this paper. EBRIP is a rather flexible algorithm that can reduce the total computation amount by increasing the size of a pre-computed table. BRIP can get dP in the computation of approximately 24.0 M in each bit, where M shows the computation amount of 1 modular multiplication on the definition field. EBRIP can get dP in the computation of approximately 12.9 M in each bit with using a pre-computed table of 16 points.

Let us compare our algorithms with other countermeasures to SPA, DPA, RPA. A countermeasure to RPA [37] is not a universal countermeasure, gives each different method to each type of elliptic curves, and do not consider ZRA-resistance. The exponent-splitting algorithm (ES) in [4], [5] is the universal countermeasure, which splits an exponent and computes $dP = rP + (d - r)P = \lfloor d/r \rfloor rP + (d \bmod r)P$ by using a random number r. ES computes dP by the same cost as the add-and-double-always algorithm with an extra point for computation but would require another challenging work to be faster algorithm with some pre-computed-table technique. Compared with ES, the computation amount of BRIP is the same as that of ES, where ES has the 1 extra point to BRIP for computation. The computation amount of EBRIP can be reduced to only 54% of that of ES, where EBRIP has the 1 extra point to ES for computation. Another universal countermeasure is a randomized window algorithm [28], which starts with the window algorithm secure against SPA and enhances the security to DPA by RPC. Therefore, it requires at least 2 random points (or RPC to at least 3 points) for each computation in the least window size of $w = 2$, and thus the performance is rather worse than our EBRIP. Compared with our BRIP, the randomized window algorithm can not work without at least 4 additional points for computation.

1.4 Organization

This paper is organized as follows. Section 2 summarizes some facts of elliptic curves such as coordinate systems and reviews power analysis of SPA, DPA, RPA, and ZRA together with some known countermeasures. Section 3 presents our new countermeasures, BRIP and EBRIP. Section 4 compares our strategy with the previous RPA-, ZRA-, and SPA-resistant countermeasure. Appendix presents algorithms that strengthen our algorithms against ADPA.

2. Preliminary

This section summarizes some facts of elliptic curves such as coordinate systems and reviews power analysis of SPA, DPA, RPA, and ZRA together with some known countermeasures.

2.1 Elliptic Curve

Let $\mathbb{F}_p$ be a finite field, where $p > 3$ is a prime. The Weierstrass form of an elliptic curve over $\mathbb{F}_p$ is described as

$$E/\mathbb{F}_p : y^2 = x^3 + ax + b \ \ (a, b \in \mathbb{F}_p, 4a^3 + 27b^2 \neq 0).$$

The set of all points $P = (x, y)$ satisfying E, together with the point of infinity O, is denoted by $E(\mathbb{F}_p)$, which forms an abelian group. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on $E(\mathbb{F}_p)$ and $P_3 = P_1 + P_2 = (x_3, y_3)$ be the sum. Then the addition formulae in affine coordinate are given as follows [7].

- **Addition formulae in affine coordinate ($P_1 \neq \pm P_2$)**

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1,$$

where $\lambda = (y_2 - y_1)/(x_2 - x_1)$.

• **Doubling formulae in affine coordinate** ($P_1 = P_2$)

$$x_3 = \lambda^2 - 2x_1, \quad y_3 = \lambda(x_1 - x_3) - y_1,$$

where $\lambda = (3x_1^2 + a)/(2y_1)$.

Let us denote the computation time of an addition (resp. a doubling) in the affine coordinate by $t(\mathcal{A} + \mathcal{A})$ (resp. $t(2\mathcal{A})$) and represent multiplication (resp. inverse, resp. squaring) in $\mathbb{F}_p$ by M (resp. I, resp. S). Then we see that $t(\mathcal{A} + \mathcal{A}) = I + 2M + S$ and $t(2\mathcal{A}) = I + 2M + 2S$. Both addition and doubling formulae need one inversion over $\mathbb{F}_p$, which is much more expensive than multiplication over $\mathbb{F}_p$. Therefore, we transform affine coordinate(x, y) into other coordinates, where the inversion is free. We give the addition and doubling formulae with Jacobian coordinate, which are widely used.

In the Jacobian coordinates [7], we set $x = X/Z^2$ and $y = Y/Z^3$, giving the equation

$$E_{\mathcal{J}} : Y^2 = X^3 + aXZ^4 + bZ^6.$$

Then, two points (X, Y, Z) and (r^2X, r^3Y, rZ) for some $r \in \mathbb{F}_p^*$ are recognized as the same point. The point at infinity is represented with $(1, 1, 0)$. Let $P_1 = (X_1, Y_1, Z_1), P_2 = (X_2, Y_2, Z_2)$, and $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$. The doubling and addition formulae can be represented as follows.

• **Addition formulae in Jacobian coordinate** ($P_1 \neq \pm P_2$)

$$\begin{aligned} X_3 &= -H^3 - 2U_1H^2 + R^2, \\ Y_3 &= -S_1H^3 + R(U_1H^2 - X_3), \\ Z_3 &= Z_1Z_2H, \end{aligned}$$

where $U_1 = X_1Z_2^2$, $U_2 = X_2Z_1^2$, $S_1 = Y_1Z_2^3$, $S_2 = Y_2Z_1^3$, $H = U_2 - U_1$, and $R = S_2 - S_1$.

• **Doubling formulae in Jacobian coordinate** ($P_1 = P_2$)

$$X_3 = T,\ Y_3 = -8Y_1^4 + M(S - T),\ Z_3 = 2Y_1Z_1,$$

where $S = 4X_1Y_1^2$, $M = 3X_1^2 + aZ_1^4$, and $T = -2S + M^2$.

The computation times in the Jacobian coordinate are $t(\mathcal{J} + \mathcal{J}) = 12M + 4S$ and $t(2\mathcal{J}) = 4M + 6S$, where $\mathcal{J}$ means Jacobian coordinates.

Elliptic curve cryptosystems requires the elliptic curve exponentiation of $dP = P + P + \cdots + P$, where $P \in E(\mathbb{F}_p)$ and d is an n-bit integer. The simple method to compute dP is a so-called binary algorithm. Algorithm 1 shows the binary algorithm to compute dP from MSB, where the binary representation of d is $d = (d_{n-1}, \cdots, d_0)$. Average computing complexity of Algorithm 1 is $nD + n/2A$, where A and D denotes the computation amount of addition and doubling, respectively. When we compute dP from LSB, we have to keep another point 2^iP instead of $T_1 = P$ but can apply the iterated doubling formulae in Jacobian coordinate [13], which computes 2^kP for $k \geq 1$ by $4kM + (4k+2)S$. However, the binary algorithm from LSB is not easily generalized to a sophisticated method with a pre-computed table.

Algorithm 1 (`Binary algorithm (MSB)`):
`Input:` d, P
`Output:` dP
1. $T[0] = O, T[1] = P$.
2. `for` $i = n - 2$ `to` 0
   $T[0] = 2T[0]$
   `if` $d_i = 1$ `then` $T[0] = T[0] + T[1]$
3. `output` $T[0]$.

2.2 Power Analysis

There are two types of power analysis, the simple power analysis (SPA) and the differential power analysis (DPA), which are described in [21], [22]. In the case of elliptic curve and also hyper elliptic curve, DPA is further improved to use a special point with a zero value, which is called the Refined Power Analysis (RPA) [12]. RPA is generalized to the Zero-value Register Attack (ZRA) [3]. In this paper, DPA, RPA, and ZRA are called DPA variants generically.

2.2.1 Simple Power Analysis

SPA makes use of such an instruction performed during an exponentiation algorithm that depends on the data being processed. Apparently, Algorithm 1 has a branch instruction conditioned by a secret exponent d, and thus it reveals the secret d. In order to be resistant against SPA, any branch instruction of exponentiation algorithm should be eliminated. There are mainly two types of countermeasures: the fixed procedure method [8] and the indistinguishable method [4]. The fixed procedure method deletes any branch instruction conditioned by a secret exponent d like add-and-double-always algorithm [8] and Montgomery-ladder algorithm [29]. Add-and-double-always algorithm is described in Algorithm 2. The indistinguishable method conceals all branch instructions of exponentiation algorithm by using indistinguishable addition and doubling operations, in which dummy operations are inserted.

Algorithm 2 (`Add-and-double-always algorithm`):
`Input:` d, P
`Output:` dP
1. $T[0] = P$ `and` $T[2] = P$.
2. `for` $i = n - 2$ `to` 0
   $T[0] = 2T[0]$. $T[1] = T[0] + T[2]$.
   `if` $d_i = 0$ `then` $T[0] = T[0]$.
   `else` $T[0] = T[1]$.
3. `output` $T[0]$.

2.2.2 Differential Power Analysis

DPA uses correlation between power consumption and specific key-dependent bits. Algorithm 2 reveals d_{n-2} by computing the correlation between power consumption and any specific bit of the binary representation of $4P$. In order to be resistant against DPA, power consumption should be changed at each new execution of the exponentiation. There

are mainly 3 types of countermeasures, the randomized-projective-coordinate method (RPC) [8], the randomized curve method (RC) [19], and the exponent splitting (ES) [4], [5]. RPC uses the Jacobian or Projective coordinate to randomize a point $P = (x, y)$ into (r^2x, r^3y, r) or (rx, ry, r) for a random number $r \in \mathbb{F}_p^*$, respectively. RC maps an elliptic curve into an isomorphic elliptic curve by using an isomorphism map of (x, y) to (c^2x, c^3y) for $c \in \mathbb{F}_p^*$. ES splits an exponent and computes $dP = rP + (d - r)P$ for a random integer r.

2.2.3 Refined Power Analysis and Zero-Value Point Attack

DPA is specialized to reveal a secret key d by using a special elliptic-curve point with a zero value, which is defined as $(x, 0)$ or $(0, y)$. These special points of $(x, 0)$ and $(0, y)$ can not be randomized by RPC or RC since they still have a zero value such as $(r^2x, 0, r)$ (resp. $(rx, 0, r)$) and $(0, r^3y, r)$ (resp. $(0, ry, r)$) in Jacobian (resp. Projective) coordinate after conversion. A countermeasure to RPA are proposed in [37], but this is not a universal countermeasure, gives each different method to each type of elliptic curves.

RPA is generalized to ZRA by [3], which makes use of any zero-value register in addition formulae. The addition and doubling formulae have a lot of each different operations stored in auxiliary registers, one of which may become zero. ZRA uses the difference in any zero value register of addition and doubling, which is not randomized by RPC or RC.

ES can resist both RPA and ZRA because an attacker cannot handle an elliptic curve point in such a way that any special point with zero value can appear during an execution of exponentiation algorithm.

3. Efficient Countermeasures against SPA and DPA Variants

In this section, we propose a new countermeasure against SPA and all DPA variants.

3.1 Our Basic Countermeasure

Here we show our *Basic SPA-resistant algorithm with RIP*, called BRIP, and then discuss the security and efficiency.

3.1.1 BRIP

Our algorithm uses a random initial point (RIP) R, computes $dP + R$, and subtracts R to get dP. In order to be secure against SPA, we have to compute $dP + R$ in such a way that it does not have any branch instruction dependent on the data being processed. Our remarkable idea lies in a sophisticated combination to compute $dP+R$ from MSB by the same complexity as Algorithm 2: first let 1 represent $1 = (1\overline{11}\cdots\overline{11})_2$ and apply the extended binary algorithm [23] to compute

$$(1\underbrace{\overline{11}\cdots\overline{11}}_{n})_2R + (\underbrace{d_{n-1}d_{n-1}\cdots d_1d_0}_{n})_2P.$$

Algorithm 3 shows our idea in detail. We get dP by computing $dP + R$ and subtracting R. BRIP makes all variables $T[0], T[1]$, and $T[2]$ dependent on a random point R, and thus let all variables of each addition and doubling differ at each execution.

Algorithm 3 (BRIP) :
Input: d, P
Output: dP
1. $T[2]$ =randompoint()
2. $T[0] = -T[2], T[1] = P - T[2]$
3. for $i = n - 1$ to 0
   $T[2] = 2T[2]$
   $T[2] = T[2] + T[d_i]$
4. output $T[2] + T[0]$

3.1.2 Security and Efficiency

We discuss the security, the computation amount, and the memory amount. BRIP lets the power-consumption pattern be fixed regardless of the bit pattern of a secret key d, and thus it is resistant to SPA.

BRIP makes use of a random initial point at each execution and let all variables $T[0], T[1]$, and $T[2]$ be dependent on the random point. Thus, an attacker cannot control a point in such a way that it outputs a special point with a zero-value coordinate or zero-value register. Therefore, if R is chosen randomly by some ways, BRIP can be resistant to DPA, RPA, and ZRA. The performance of BRIP depends on the algorithm of generating a random initial point R. The simplest way is to generate the x-coordinate randomly and compute the corresponding y-coordinate if exists. It should require much work. The cheaper way is to keep one point R_0 and convert R_0 into a randomized point R by RPC [16]. In order to enhance the security of BRIP against ADPA, Algorithm 3 has only to be coded in such a way that it randomizes the address bus as well as the data itself, which will be shown in Appendix.

The computation amount required for Algorithm 3 is $nD + nA$, which is the same as Algorithm 2. The number of variables necessary for computation is only 3.

3.2 Our Generalized Countermeasure

Our basic countermeasure BRIP can be generalized to a faster algorithm with a pre-computed table since BRIP makes use of the binary representation from MSB. We may note that the binary representation from LSB can not be easily generalized to a faster algorithm with a flexible precomputed table. In the following, we will summarize the exponentiation algorithm with a precomputed table and describe two algorithms based on the extended-binary and the window algorithms, which are called EBRIP and WBRIP, respectively. EBRIP is more efficient than WBRIP although

extended-binary algorithm usually does not work on a single exponentiation as efficiently as the window algorithm.

3.2.1 SPA-Resistant Exponentiation Algorithm with a Pre-Computed Table

As for algorithms of using a pre-computed table, there are mainly two algorithms: the window algorithm [24] and the extended binary (simultaneous exponentiation) algorithm [23], [38]. The extended binary algorithm is originally used to compute two exponentiations $aP + bQ$, which is applied to compute one exponentiation as follows [38]. Let $d = \sum_{i=0}^{n-1} d_i 2^i$ and n be even.

1. Divide d into two components of $d = b \parallel a$, where $b = (d_{n-1} \cdots d_{\frac{n}{2}})_2$ and $a = (d_{\frac{n}{2}-1} \cdots d_0)_2$.
2. Compute $Q = 2^{\frac{n}{2}} P$.
3. Set a pre-computed table $\{P, Q, P + Q\}$.
4. Compute $aP + bQ$ in the extended binary algorithm by using the pre-computed table.

The detailed algorithm is shown in Algorithm 4.

Algorithm 4 (2-division extended-binary algorithm):
Input: d, P
Output: dP
1. Set $d = b \parallel a$,
   $b = (b_{\frac{n}{2}-1} \cdots b_0)_2 = (d_{n-1} \cdots d_{\frac{n}{2}})_2$, and
   $a = (a_{\frac{n}{2}-1} \cdots a_0)_2 = (d_{\frac{n}{2}-1} \cdots d_0)_2$.
2. $T[0,1] = P$, $T[1,0] = 2^{\frac{n}{2}} P$,
   $T[1,1] = T[0,1] + T[1,0]$, and $T[2] = O$.
3. for $i = \frac{n}{2} - 1$ to 0
   $T[2] = 2T[2]$.
   if $(b_i, a_i) \neq (0,0)$ then $T[2] = T[2] + T[b_i, a_i]$.
4. output $T[2]$.

Going back to the countermeasure using a pre-computed table, it is necessary for both the extended binary and window algorithms to make power-consumption pattern same in order to be resistant to SPA. In the case of the window algorithm, some SPA-resistant algorithms are proposed in [27], [28], [30]: [27], [30] do not aim at being resistant to DPA and its variants; and [28], called a randomized window method, is resistant to RPA and ZRA by using at least 2 random points in a pre-computed table at each computation, but thus the performance is rather worse than our algorithm. Note that [28] is not resistant to ADPA. In the case of the extended binary algorithm, up to the present, any SPA-resistant algorithm has not been proposed because it usually gives lower performance than the window algorithm.

3.2.2 Our Extended-Binary-Based Algorithm with RIP

Let us show our extended-binary-based algorithm with RIP, which is called EBRIP for short.

1. Choose a random point R.
2. Let the number of divisions be t.
3. Adjust n to be the least common multiple n' of t and n by setting 0 to MSB of d ($n' < n + t$), where
$$d' = 0 \cdots 0\, d_{n-1} \cdots d_0.$$
4. Divide d' into t components ($\frac{n'}{t} = k$) of $d' = \alpha_{t-1} \parallel \cdots \parallel \alpha_1 \parallel \alpha_0$ and represent 1 in $(k+1)$ bits, those are,
$$\begin{array}{lcccc} \alpha_{t-1} & = & 0 & \cdots & d_{(t-1)k} \\ & \vdots & & & \\ \alpha_1 & = & d_{2k-1} & \cdots & d_k \\ \alpha_0 & = & d_{k-1} & \cdots & d_0 \\ 1 & = & 1\ \overline{1} & \cdots & \overline{1} \end{array}$$
5. Compute $P_i = 2^{ki} P$ for $i = 1$ to $t - 1$. (Set $P_0 = P$).
6. Compute a pre-computed table $T_t = \{\Sigma_{i=0}^{t-1} a_i P_i - R\,(a_i \in \{0,1\})\}$, which consists of 2^t points.
7. Compute $\alpha_0 P_0 + \cdots + \alpha_{t-1} P_{t-1} + 1R$ in the way of the extended binary algorithm. (See Algorithm 4.)

Algorithm 5 shows the case of $t = 2$ and an even n for simplicity.

Algorithm 5 (EBRIP (2 divisions)):
Input: d, P
Output: dP
1. $T[2] =$randompoint().
1. Set $d = b \parallel a$,
   $b = (b_{\frac{n}{2}-1} \cdots b_0)_2 = (d_{n-1} \cdots d_{\frac{n}{2}})_2$, and
   $a = (a_{\frac{n}{2}-1} \cdots a_0)_2 = (d_{\frac{n}{2}-1} \cdots d_0)_2$.
2. $T[0,0] = -R$, $T[0,1] = P - R$,
   $T[1,0] = 2^{\frac{n}{2}} P - R$, and $T[1,1] = 2^{\frac{n}{2}} P + P - R$.
3. for $i = \frac{n}{2} - 1$ to 0
   $T[2] = 2T[2]$.
   $T[2] = T[2] + T[b_i, a_i]$.
4. Output $T[2] + T[0,0]$.

Let us discuss the resistance, computation amount, and memory amount. As for SPA, the power-consumption pattern is not changed for any initial point R and any secret key d thanks to the representation of 1, and EBRIP is thus secure against SPA. Moreover, under the assumption that an initial point R is completely random, EBRIP is secure against DPA, RPA, and ZRA, as we mentioned in Sect. 3.1. In order to enhance the security of EBRIP against ADPA, Algorithm 5 has only to be coded in such a way that it randomizes the address bus as well as the data itself, which will be shown in Appendix.

As for the computation amount, EBRIP consists of these parts: compute base points $P_1, \cdots, P_{t-1}$, a pre-computed table T_t, and the main routine. The computation amount for base points, T_t, or main routine is $\frac{(t-1)n'}{t} D$, $2^t A$, or $\frac{n'}{t} D + \frac{n'}{t} A$, respectively. Thus, the total computation amount is $n'D + \frac{n'}{t} A + 2^t A$. On the other hand, the number of points in T_t is 2^t, which includes a random point R. EBRIP needs one more point of variable to execute. Thus, the necessary memory is $2^t + 1$ in total.

One remarkable point of EBRIP is that the length of representation of 1 is not fixed to n but adjusted to $\lceil \frac{n}{t} \rceil + 1 (<$

n). As a result, EBRIP realizes more efficient computation than the next window-based algorithm.

Remark: The Jacobian coordinate gives the most efficient computation of EBRIP as we will see in Sect. 4. In the use of the Jacobian coordinate, the stage of computation for base points can employ the technique of m-repeated elliptic curve doublings in Sect. 2, which reduces the computation amount for the stage.

3.2.3 Our Window-Based Algorithm with RIP

Our window-based algorithm with RIP is summarized as follow, which is is called WBRIP for short.

1. Choose a random point R.
2. Set the width of window to be w.
3. Adjust n' to be the least common multiple of w and n by setting 0 to MSB ($n \le n' < n + w$) of d, where
$$d' = 0 \cdots 0\, d_{n-1} \cdots d_0.$$
4. Compute $R' = -(2^w - 1)R$.
5. Set a pre-computed table $T_w = \{R', P+R', 2P+R', 3P+R', \cdots, (2^w-2)P+R', (2^w-1)P+R'\}$, where the number of points in T_w is 2^w.
6. Compute $(\underbrace{0 \cdots 0\, d_{n-1} \cdots d_0}_{n'})_2 P + (1\underbrace{\overline{11} \cdots \overline{11}}_{n'})_2 R$ in the way of window algorithm by using T_w.

Let us discuss the security, the computation amount, and the memory amount. Power-consumption pattern is not changed for any random R and any secret key d in the same way as EBRIP and BRIP. This is why WBRIP is resistant to SPA. This means that WBRIP is secure against SPA without any additional modification on the window algorithm seen in [27], [30]. Furthermore, under the assumption that an initial point R is completely random, our algorithm is resistant to DPA, RPA, and ZRA.

Next we investigate the computation amount of WBRIP. WBRIP consists of three parts: compute an intermediate point R', a pre-computed table T_w, and main routine. The computation amount of R', a pre-computed table T_w, or main routine is $wD + A$, $(2^w - 1)A$, or $\frac{n'}{w}A + n'D$, respectively. Therefore, the total computation amount is $n'D + \frac{n'}{w}A + 2^wA + wD$, where $n' < n + w$. It is not as efficient as EBRIP since the length of representation of 1 is fixed to $n' + 1$. If we reduce the length of representation of 1 to n such as $(1\underbrace{\overline{1} \cdots \overline{1}}_{n-1})_2$, then T_w requires other points and thus the size of T_w becomes larger. Regarding as the memory amount of WBRIP, the number of points in T_w is 2^w, which includes a random point R. Additional one variable is necessary for computation. Thus, the necessary memory is $2^w + 1$ points in total.

As a result, compared with EBRIP, WBRIP needs more computation amount with the same memory amount.

Remark: The SPA-resistant-window algorithm [30] can be applied to WBRIP instead of the ordinary window algorithm. That is, convert d to SPA-resistant-window representation d' (n' bits), and compute $d'P + R'$ in a way of WBRIP. The SPA-resistant-window algorithm works in the density of the non-zero bit $\frac{1}{w}$ with the pre-computed table $T = \{\pm P, \pm 3P, \cdots, \pm(2^w - 1)P\}$, which can be constructed by just having positive 2^{w-1} points. However, applying it to WBRIP, the pre-computed table becomes $T'_w = \{R' + P, R' - P, \cdots, R' + (2^w - 1)P, R' - (2^w - 1)P\}$, which requires us to have 2^w points. Then the size of the pre-computed table $|T'_w|$ is the same as that of the pre-computed table $|T_w|$ in WBRIP. Therefore, it does not give any advantage over WBRIP based on the ordinary window method but rather gives an additional work of converting d to d'.

3.3 Further Discussion

An SPA in the chosen-message-attack scenario[†] was proposed in [39]. An attacker sends a point P with order 2 in Algorithm 2 and observes the power consumption. Then, there are only two possible points of $T[0]$ for each i, those are O and P for $d_i = 0$ and 1, respectively. Thus, an attacker can guess each d_i. The same discussion holds in the case of BRIP and EBRIP. However, from the realistic point of view, their attack cannot be applied on an ordinary elliptic curve cryptosystems. Because a prime-order elliptic curve is usually recommended and a basepoint with prime order is usually allowed even if an elliptic curve is not prime order. Furthermore, it can be easily avoided by checking $2P \neq O$ before computing dP.

4. Comparison

From the point of view of computation and memory amount, we compare our countermeasures BRIP and EBRIP with the previous countermeasures to SPA, DPA, and RPA, those are ES [5], randomized window algorithm [28], LRIP [16], and randomized linearly-transformed coordinates (RLC) [16]. There exists another countermeasure [17] that is a combination of RIP and the Montgomery ladder. However, we omit it from comparisons since it computes only the x-coordinate of dP and requires an additional work to recover y-coordinate of dP.

Table 1 shows the comparison in two types of computation amount: the computation amount in the case of 160-bit definition field and the average computation amount in each bit. The average computation amount is computed by $\frac{\texttt{the computation amount over } \mathbb{F}_p}{|\mathbb{F}_p|}$, and, thus, it does not depend on the size of definition field. Here, M, S, or RPC represents the computation amount of modular multiplication, modular square, or RPC on the definition field, respectively. We also assume that $S = 0.8M$ as usual. The Jacobian coordinate is used to compute the total number of modular multiplications for all cases except RLC because it

[†]Their attack is applicable to only ElGamal decryption, a classical RSA signature, and RSA decryption.

Table 1 Comparison of countermeasures.

	memory amount	computation amount†		computation amount
	(#points, #scalar)	#D + #A	#M + #S	in each bit
ES [5]	(4, 2)	160D+160A	$2560M + 1600S\,(3840M)$	$16M + 10S\,(24.0M)$
strengthened window [28]	(5, 0)	161 D + 82A + 3RPC	$1640M + 1297S\,(2678M)$	$10.3M + 8.1S\,(16.7M)$
LRIP [16]	(4, 0)	160D+160A	$2560M + 1282S\,(3596M)$	$16M + 8S\,(22.4M)$
RLC [16]	(3, 0)	160D+160A	$2720M + 1282S\,(3745M)$	$17M + 8S\,(23.4M)$
BRIP	(3, 0)	160D+160A	$2560M + 1600S\,(3840M)$	$16M + 10S\,(24.0M)$
EBRIP($t = 2$)	(5, 0)	160D+84A	$1648M + 1138S\,(2558M)$	$10.3M + 7.1S\,(16.0M)$
EBRIP($t = 3$)	(9, 0)	162D+62A	$1392M + 1006S\,(2197M)$	$8.7M + 6.3S\,(13.7M)$
EBRIP($t = 4$)	(17, 0)	160D+56A	$1312M + 946S\,(2069M)$	$8.2M + 5.9S\,(12.9M)$
EBRIP($t = 5$)	(33, 0)	160D+64A	$1408M + 962S\,(2178M)$	$8.8M + 6.0S\,(13.6M)$

† This shows the computation amount in the case of 160-bit definition field.

is the most efficient. RLC uses addition formulae slightly modified from the Jacobian coordinate [16]. EBRIP, RLC and LRIP can make use of the technique of m-repeated elliptic curve doublings in Sect. 2.

BRIP can compute dP in the computation amount of $160D + 160A$ with 3 points. The computation amount in each bit is $24.0M$, which is the same as that of ES. EBRIP with $t = 2$ can compute dP in the computation amount of $160D + 84A$ with 5 points. The computation amount in each bit is $16.0M$, which is reduced to only 66% of ES. Compared with [28], EBRIP with $t = 2$ is reduced to 96% while using the same memory amount. EBRIP with $t = 4$ can execute dP in the computation amount of $160D + 56A$ with 17 points. In this case, the computation amount in each bit is $12.9M$, which is reduced to only 54% of ES. Note that $t = 4$ is the fastest when the size of definition filed $\mathbb{F}_p$ is 160.

Remark: The algorithms of an exponentiation dP are generally classified into two cases: P is pre-determined or not. This paper assumes that P is not pre-determined and given in each execution, which occurs in a decryption. Therefore, a pre-computed table cannot be computed beforehand and is constructed at the same time as the execution of dP. Generally, the more the size of a pre-computed table is, the more the computation amount of construction of a pre-computed table is. This is why the case of $t = 4$ gives the least computation amount. In the case of an exponentiation of a pre-determined point P, which occurs in the signature generation, we do not have to count the computation amount of construction of a pre-computed table. As a result, these two kinds of exponentiation algorithms are evaluated in each different way, and, thus, both usually employ different algorithms. Some concrete algorithms of dP for a pre-determined P, refer to [6].

5. Concluding Remarks

In this paper, we have presented countermeasures of BRIP, EBRIP, and WBRIP that are resistant to RPA, ZRA, DPA, and SPA. Our countermeasure BRIP can get dP in the computation of approximately 24.0 M in each bit with the memory amount of 3 points. The memory size is the smallest and it also can work the fastest among algorithms with the memory amount of 3 points. EBRIP with $t = 4$ can get dP in the computation of approximately 12.9 M in each bit with the memory amount of 17 points, which is the fastest.

Our novel algorithms of BRIP and EBRIP can work in $24.0M$, $16.0M$, $13.7M$, or $12.9M$ in each bit with the memory amount of 3, 5, 9, or 17 points, respectively. Therefore, our results could offer the best algorithm to a wide range of applications: some require the small memory storage at the cost of time or some require the fast implementation at the cost of memory.

Our countermeasure improves the addition-chain itself and not use a specific feature of an elliptic curve such as a coordinate system. Therefore, BRIP and EBRIP can also be generalized to deal with hyper elliptic curve cryptosystem.

Acknowledgements

This work was partly supported by Grant-in-Aid for Scientific Research (B), 17300002.

References

[1] R.M. Avanzi, "On multi-exponentiation in cryptography," Cryptology ePrint Archive, Report 2002/154, http://eprint.iacr.org/2002/154/, 2002.
[2] K. Araki and T. Satoh "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves," Commentarii Math. Univ. St. Pauli., vol.47, pp.81–92, 1998.
[3] T. Akishita and T. Takagi, "Zero-value register attack on elliptic curve cryptosystem," IEICE Trans. Fundamentals, vol.E88-A, no.1, pp.132–139, Jan. 2005.
[4] C. Clavier and M. Joye, "Universal exponentiation algorithm—A first step towards provable SPA-resistance," CHES2001, Lecture Notes in Computer Science, vol.2162, pp.300–308, Springer-Verlag, 2001.
[5] M. Ciet and M. Joye, "(Virtually) Free randomization technique for elliptic curve cryptography," ICICS2003, Lecture Notes in Computer Science, vol.2836, pp.348–359, Springer-Verlag, 2003.
[6] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation," Proc. Information and Communications Security, ICICS'97, Lecture Notes in Computer Science, vol.1334, pp.282–290, Springer-Verlag, 1997.
[7] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," Advances in Cryptology—Proc. ASIACRYPT'98, Lecture Notes in Computer Science, vol.1514, pp.51–65, Springer-Verlag, 1998.
[8] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystem," CHES'99, Lecture Notes in Computer Science, vol.1717, pp.292–302, Springer-Verlag, 1999.
[9] G. Frey and H.G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Mathematics

of Computation, vol.62, pp.865–874, 1994.
[10] "Proposed federal information processing standard for digital signature standard (DSS)," Federal Register, vol.56, no.169, pp.42980–42982, Aug. 1991.
[11] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inf. Theory, vol.IT-31, pp.469–472, 1985.
[12] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," PKC2003, Lecture Notes in Computer Science, vol.2567, pp.199–210, Springer-Verlag, 2003.
[13] K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara, "Fast implementation of public-key cryptography on DSP TMS320C6201," CHES'99, Lecture Notes in Computer Science, vol.1717, pp.61–72, Springer-Verlag, 1999.
[14] K. Itoh, T. Izu, and M. Takenaka, "Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA," CHES2002, Lecture Notes in Computer Science, vol.2523, pp.129–143, Springer-Verlag, 2002.
[15] K. Itoh, T. Izu, and M. Takenaka, "A practical countermeasure against address-bit differential power analysis," CHES2003, Lecture Notes in Computer Science, vol.2779, pp.382–396, Springer-Verlag, 2003.
[16] K. Itoh, T. Izu, and M. Takenaka, "Efficient countermeasures against power analysis for elliptic curve cryptosystems," SCIS2004, 2004 (previous version). The final version is in the Proc. CARDIS 2004.
[17] K. Itoh, T. Izu, and M. Takenaka, "On the randomized initial point countermeasure against power analysis (part III)," SCIS2005, 2D1-1, 2005.
[18] T. Izu and M. Takenaka, "On the randomized initial point countermeasure against power," IEICE Technical Report, ISEC2004-8, 2004.
[19] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptosystem," CHES2001, Lecture Notes in Computer Science, vol.2162, pp.377–390, Springer-Verlag, 2001.
[20] N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, vol.48, pp.203–209, 1987.
[21] C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other system," CRYPTO'96, Lecture Notes in Computer Science, vol.1109, pp.104–113, Springer-Verlag, 1996.
[22] C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," Crypto'99, Lecture Notes in Computer Science, vol.1666, pp.388–397, Springer-Verlag, 1999.
[23] D.E. Knuth, The Art of Computer Programming, vol.2, Seminumerical Algorithms, 2nd ed., Addison-Wesley, 1981.
[24] K. Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method," Advances in Cryptology—Proc. Crypto'92, Lecture Notes in Computer Science, vol.740, pp.345–357, Springer-Verlag, 1993.
[25] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," Proc. 22nd Annual ACM Symposium on the Theory of Computing, pp.80–89, 1991.
[26] V.S. Miller, "Use of elliptic curves in cryptography," Advances in Cryptology—Proc. Crypto'85, Lecture Notes in Computer Science, vol.218, pp.417–426, Springer-Verlag, 1986.
[27] B. Möller, "Securing elliptic curve point multiplication against side-channel attacks," ISC2001, Lecture Notes in Computer Science, vol.2200, pp.324–334, Springer-Verlag, 2001.
[28] B. Möller, "Parallelizable elliptic curve point multiplication method with resistance against side-channel attacks," ISC2002, Lecture Notes in Computer Science, vol.2433, pp.402–413, Springer-Verlag, 2002.
[29] P.L. Montgomery, "Speeding the Pollard and elliptic curve methods for factorization," Mathematics of Computation, vol.48, pp.243–264, 1987.
[30] K. Okeya and T. Takagi, "The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks," CT-RSA2003, Lecture Notes in Computer Science, vol.2612, pp.328–342, Springer-Verlag, 2003.
[31] S.C. Pohlig and M.E. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," IEEE Trans. Inf. Theory, vol.IT-24, pp.106–110, 1978.
[32] J. Pollard, "Monte Carlo methods for index computation (mod p)," Mathematics of Computation, vol.32, pp.918–924, 1978.
[33] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol.21, no.2, pp.120–126, 1978.
[34] Standard for efficient cryptography group, specification of standards for efficient cryptography. Available from: http://www.secg.org
[35] I.A. Semaev, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p," Mathematics of Computation, vol.67, pp.353–356, 1998.
[36] N.P. Smart, "The discrete logarithm problem on elliptic curves of trace one," J. Cryptology, vol.12, pp.193–196, 1999.
[37] N.P. Smart, "An analysis of Goubin's refined power analysis attack," CHES2003, Lecture Notes in Computer Science, vol.2779, pp.281–290, Springer-Verlag, 2003.
[38] J.A. Solinas, "Low-weight binary representation for pairs of integers," Centre for Applied Cryptographic Research, University of Waterloo, Combinatorics and Optimization Research Report CORR 2001-41, 2001.
[39] S.-M. Yen, W.-C. Lien, S.J. Moon, and J.C. Ha, "Power analysis by exploiting chosen message and internal collisions—Vulnerability of checking mechanism for RSA-decryption," Mycrypt 2005, Lecture Notes in Computer Science, vol.3715, pp.183–195, Springer-Verlag, 2005.

Appendix: ADPA-Resistant BRIP and EBRIP

Algorithms 3 and 5 fix the address bus although they change the data itself at each execution. In order to be resistant to ADPA, both have only to be coded in such a way that they randomize the address bus as well as the data itself. This is generally realized by using the idea of [15]. In our case, concrete algorithms are in [18]. However, their algorithms require two additional variables to randomize the address of variables and so are not suitable for BRIP and EBRIP. Here we present another approach that strengthens BRIP and EBRIP against ADPA. Our method changes not the address of variables but the value itself of variables and thus randomizes the address of variables storing a certain value. As a result, compared with [18], our algorithm can enhance the security with each one additional variable to the original BRIP or EBRIP.

Algorithm 6 (ADPA-resistant BRIP):
Input: d, P
Output: dP
1. $T[3]$ =randompoint()
2. $r = 0$
3. $T[0] = -T[3],\ T[1] = P - T[3],\ T[2] = O$
4. for $i = n - 1$ downto 0 {
   $T[3] = 2T[3]$
   $T[2] = T[0]$
   $T[0] = T[0 \oplus r']$
   $T[1] = T[1 + r']$
   $r = r \oplus r'$
   $T[3] = T[3] + T[d_i \oplus r]$ }
5. output $T[2] + T[r]$

Here, r' is a 1-bit random number of $\{0, 1\}$.

Algorithm 7 (ADPA-resistant EBRIP (2 divisions)):
Input: d, P
Output: dP
1. $T[5] =$ randompoint()
2. $r = 0$
3. Set $d = b||a$
4. $b = (b_{\frac{n}{2}-1} \dots b_0) = (d_{n-1} \dots d_{\frac{n}{2}})$ and
5. $a = (a_{\frac{n}{2}-1} \dots a_0) = (d_{\frac{n}{2}-1} \dots d_0)$
6. $T[0] = -R, T[1] = P - R,\ T[2] = 2^{\frac{n}{2}}P - R,$
7. $T[3] = 2^{\frac{n}{2}}P + P - R,\ T[4] = O$
9. for $i = \frac{n}{2} - 1$ to 0 {
   $T[5] = 2T[5]$
   $T[4] = T[0]$
   $T[0] = T[0 + r']$
   $T[1] = T[1 + r']$
   $T[2] = T[2 + r']$
   $T[3] = T[3 + r']$
   $r = r + r' \pmod 4$
   $T[5] = T[5] + T[(b_i a_i)_2 - r \pmod 4]$ }
10. Output $T[5] + T[4 - r]$

Here, r' is a 1-bit random number of $\{0, 1\}$.

Hideyo Mamiya received his B.Eng. degree from National Defense Academy and M. Info. Sci. degree from Japan Advanced Institute of Science and Technology in 2000 and 2005, respectively.

Atsuko Miyaji received her B.Sc., M.Sc., and Dr. Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997, respectively. She was with Matsushita Electric Industrial Co., LTD from 1990 to 1998, where she engaged in research and development of secure communications. She has been an associate professor at JAIST (Japan Advanced Institute of Science and Technology) since 1998. She has joined the computer science department of University of California, Davis since 2002. Her research interests include the application of projective varieties theory into cryptography and information security. She received the IPSJ Sakai Special Researcher Award in 2002, the Standardization Contribution Award in 2003, and Engineering Sciences Society: Certificate of Appreciation in 2005. She is a member of the International Association for Cryptologic Research, the Information Processing Society of Japan, and the Mathematical Society of Japan.

Hiroaki Morimoto received his B.Sc. degree from Shizuoka University and M. Info. Sci. degree from Japan Advanced Institute of Science and Technology in 2002 and 2004, respectively.