DPA Countermeasures by Improving the Window Method

Kouichi Itoh, Jun Yajima, Masahiko Takenaka, and Naoya Torii

Fujitsu Laboratories Ltd., 64 Nishiwaki, Ohkubo-cho, Akashi 674-8555 Japan.
{kito, jyajima, takenaka}@flab.fujitsu.co.jp, torii.naoya@jp.fujitsu.com

Abstract. We propose three differential power analysis (DPA) countermeasures for securing the public key cryptosystems. All countermeasures are based on the window method, and can be used in both RSA and elliptic curve cryptosystems (ECC). By using the optimal countermeasure, performance penalty is small. In comparison with k-ary method, computation time of our countermeasure is only 105% in 1024-bit RSA and 119% in 160-bit ECC.

1 Introduction

Differential power analysis (DPA, [4]), proposed by Kocher et al., is an attack that enables extraction of a secret key stored in a cryptographic device, such as smartcard. In this attack, an attacker monitors the power consumption of the cryptographic devices, then statistically analyzes the collected power signal data to extract the secret key. This attack can be used against both secret and public key cryptosystems.

Currently, DPA is known as a big threat against the smartcard security, and the necessity of countermeasure for protecting the cryptographic device from the DPA attack is described in many standards. For example, the countermeasure against this attack is commented in FIPS 140-2, which is the US standard of the cryptographic module. Moreover, the requirement for DPA protection is included in the protection profile (PP), which is a list of smartcard security requirements based on the ISO 15408.

Various DPA countermeasures have been already proposed. Data randomizing is a well-known DPA countermeasure, in which the intermediate data is randomly transformed inside the cryptographic device. By using this technique, statistical analysis method on DPA attack is disabled, because the intermediate data on the encryption is unpredictable to the attacker. DPA countermeasure using data randomizing technique can be used both in secret and public key cryptosystems. We focus on the countermeasure for public key cryptosystems in the rest of this paper.

Previous DPA countermeasures for public key cryptosystems are described in [1] [2] [3] [5] [7] [8] [9] and [10]. Some of these countermeasures have demerits in comparison with the straight-forward implementation. Roughly speaking, these demerits are divided into 3 types. The first demerit is that, the countermeasures involve a performance penalty. Especially, an exponent splitting technique

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 303–317, 2003.

described in [1] requires two times of the computation as that of the straightforward implementation. The second demerit is that, the countermeasures are not always available in both RSA and elliptic curve cryptosystems (ECC). That is, countermeasures described in [2] [3] [9] uses the technique by randomizing the representation of projective coordinates, which are available only in ECC, not in RSA. The third demerit is that, some countermeasures require the additional parameters. For example, a countermeasure described in [2] requires the parameter $\phi(n)$ in RSA where n is a public modulo, or order of the base point in ECC. This makes it hard to match the I/F of the cryptographic engine with and without countermeasure, that is, a vulnerable engine can not easily be replaced to a secure one.

In this paper, we propose three novel DPA countermeasures for securing the public key cryptosystems. All countermeasures are based on the window method, which is an efficient algorithm for computing the public key cryptosystems. By using our countermeasures, all of the above demerits are avoided. In the first countermeasure, we use the novel idea of 'overlapping window method'. In the second countermeasure, we use the window method in which the pre-computed table data is randomized. In the third countermeasure, we use a hybrid technique of the first and second countermeasure. We call these countermeasures overlapping window method (O-WM), randomized table window method (RT-WM) and hybrid randomizing window method (HR-WM) respectively. These have different characteristics, but have common three merits: (i) encryption operation is fast, (ii) available in both RSA and ECC, and (iii) additional parameter is unnecessary. Merit (i) is that, the performance penalty of our countermeasure is small. In comparison with the k-ary method, computation time of our countermeasure is 119% in ECC and 105% in RSA. Merit (ii) dues to that our countermeasures are base on the window method, and we show that two of our countermeasures have high security against DPA attacks in both RSA and ECC. By merit (iii), I/F of a cryptographic engine with and without DPA countermeasures can be the same. That is, a vulnerable engine is easily replaced to a secure one.

We describe the previous data randomizing technique in section 2, our countermeasures in section 3, security evaluation in section 4, DPA experiment result in section 5 and performance comparison in section 6.

2 Data Randomizing Techniques

Data randomizing (blinding) is a well-known DPA countermeasure described in [1] [2] [3] [5] [7] [8] [9] and [10]. These techniques make the intermediate data on the encryption operation unpredictable to the attacker by randomly transforming the data. Hence, an intermediate encryption data at a moment is one of the possible values of the randomized data. We call the number of the possible values of randomized data at a moment 'NRD'. It is easy to see that as NRD is larger, DPA attack is harder, so that the security of the data randomizing technique can be evaluated by NRD.

Previous data randomizing techniques that can be used for both ECC and RSA are described in [1] [2] [5] [7] [8] and [10]. Among these countermeasures, we

take up three typical methods in which NRD is clearly evaluated. These three countermeasures are shown in following (A), (B) and (C).

(A) exponent blinding ([2]) : Instead of a secret key d, $d' = d + r \times \phi$ is used, where r is randomly given and ϕ is an order. NRD is the number of possible values of r.
(B) calculation randomizing ([7]) : Randomly choose the bit position of a secret key d, then first perform a binary-method from the chosen bit to MSB, and second, perform a binary-method from the chosen bit to LSB by using the fist calculation result. NRD is $log_2 d$.
(C) exponent splitting ([1]) : From a secret key d, generate random numbers d_1 and d_2 that satisfy $d = d_1 + d_2$, then calculate $a^{d_1} \times a^{d_2} \pmod n = a^d \pmod n$. NRD is d.

We suppose that all above countermeasures provide enough security. That is, NRD of (A), (B) and (C) ranges from $log_2 d$ to d widely, but any countermeasures listed above can attain enough security at current technology, because the size of the spike is reduced to less than 1/100 of the device using a straight-forward implementation.

On remarking the performance, two times modular exponentiation are required in (C), which makes the performance worse. So we compare our countermeasure with (A) and (B).

3 Our Method

3.1 Overview

We propose three DPA countermeasures by improving the window method. First one is overlapping window method (O-WM), second one is randomized table window method (RT-WM), and third one is hybrid randomizing window method (HR-WM). Each countermeasure has unique characteristics from the viewpoint of operation, speed and security. Appropriate countermeasure can be chosen according to the usage of the cryptographic devices (ex. encryption algorithm is RSA or ECC). Details of our countermeasures are described in later sections. In the rest of this paper, we use following notations.

- Our countermeasures can be used in both RSA and ECC, but we unified the description for RSA for simplicity.
- We refer the ECC computation using the affine coordinates as 'ECC-2D', and that using the projective or Jacobian coordinates as 'ECC-3D'.
- d is the secret key, u is represented with $u = log_2 d$, w_i is an index value for the pre-computed table (i.e., w_i is a window) and q is the number of w_i. (i.e., $i = 0, 1, ..., q-1$.)
- $EXP_{WM}(w_0, \ldots, w_s)$ represents the intermediate exponent (or scalar) value when the table look-up operation proceeds from w_0 to w_s, where WM represents the type of window method, that is, k-ary, O-WM, RT-WM, or HR-WM. For example, $EXP_{k-ary}(w_0, \ldots, w_s) = (\ldots(((2^k \times w_0) + w_1) \times 2^k)\ldots) \times 2^k + w_s$. Representations of $EXP_{O-WM}()$, $EXP_{RT-WM}()$ and $EXP_{HR-WM}()$ are described in the later section.

- $DAT_{WM}(a, w_0, \ldots, w_s)$ represents the intermediate data value when the table look-up operation proceeds from w_0 to w_s for an input value a, where WM represents the type of window method. When O-WM or HR-WM is used in ECC-3D, the number of possible values of $DAT_{WM}(a, w_0, \ldots, w_s)$ is much greater than that of $EXP_{WM}(w_0, \ldots, w_s)$, because of the redundant data representation in the projective coordinates. The reason for this is given in section 4-2.
- NRD means the number of possible randomized data values at any given moment when using the data randomizing techniques, as described in section 2.
- AR is attenuation ratio that represents the ratio of the size of the spikes that appears in the differential power trace with and without DPA countermeasure. Detail of AR is described in section 4-1.
- $bit(a, x, \ldots, y)$ represents the concatenation of the bit values of a, which is represented in binary, from the x-th to the y-th bit. ($x = 0, 1, \ldots; y = 0, 1, \ldots; x \geq y$.) The bit values upper than the MSB are regarded as 0. e.g., if $a = 6 = (110)_2$, $bit(a, 0) = 0$, $bit(a, 1) = 1$, and $bit(a, 4, 3, 2, 1) = (0011)_2 = 3$.

3.2 Overlapping Window Method (O-WM)

The characteristic of the O-WM is that, two continuous windows w_i and w_{i+1} 'overlap' each other at the same bit position of d. We show the steps of O-WM in algorithm 1 and an overview in figure 1. In O-WM, w_i and w_{i+1} are allowed to 'overlap' at the same bit position of d (figure 1). By overlapping the w_i, plural possible values of $\{w_0, \ldots, w_{q-1}\}$ are generated for a fixed d. Hence, the intermediate data on the encryption operation will be unpredictable to the attacker by randomly choosing one of these values. We denote h_i as the overlapping bit length between w_i and w_{i+1}. If the table look-up operation in step 18 is finished for $i = s$, $EXP_{O-WM}()$ is represented as $EXP_{O-WM}(w_0, \ldots, w_s) = (\ldots(w_0 \times 2^{k-h_0} + w_1)\ldots)2^{k-h_{s-1}} + w_s$, whose bit length is $s \times k - (h_0 + \ldots + h_{s-1})$ and lowest h_s-bit randomly value (figure 1).

In comparison with the k-ary method, the overhead for table making is the same, but the number of repeating the table look-up operations is larger.

Note 1. For securing against SPA attacks, we recommend to set h_i a fixed value h that satisfy $h \geq k/2$. This tweak provides a protection against the SPA by observing only one time execution of the cryptographic device, because multiplication and square are repeated in constant pattern. h must satisfy $h \geq k/2$ to prevent the bias distribution of w_i.

3.3 Randomized Table Window Method (RT-WM)

The characteristic of the RT-WM is that, pre-computed table data is randomized. We show the steps of RT-WM in algorithm 2 and an overview in figure 2. In step 5, pre-computed table data is generated by $tab[i] = a^{i \times 2^b + r} \pmod{n}$, where i is an index value and r is a b-bit random value. If step 19 is finished

Algorithm 1. Overlapping window method (O-WM)

1: /* pre-computed table data making */
2: for $(i = 0; i < 2^k; i++)$ $tab[i] = a^i \pmod n$;
3: /* window w_i and overlapping length h_i making */
4: /* generate random number q and $0 < h_0, h_1, \ldots, h_{q-2} < k$
5: which satisfy $q \times k + (h_0 + h_1 + \ldots h_s) = u$.
6: For securing against SPA, h_i are recommended to be
7: the fixed value $h \geq k/2$. */
8: $(h_i, q) = \text{GenRandom}()$; $u' = u - k$; $dt_{q-1} = bit(d, u-1, \ldots u')$;
9: for $(i = 0; i < q-1; i++) = \{$
10: $w_i = (\text{Random number}, \max(0,\ dt_i - 2^{h_i} + 1) \leq w_i \leq dt_i)$;
11: $dt_{i+1} = (dt_i - w_i) \times 2^{k-h_i} + bit(d, u'-1, \ldots, u'-(k-h_i))$;
12: $u' = u' - (k - h_i)$;
13: }
14: $w_{q-1} = dt_{q-1}$;
15: /* modular exponent process */
16: $v = tab[w_0]; i = 1$;
17: while $(i < q)$ {
18: $v = v^{2^{k-h_i}} \pmod n$; $v = v \times tab[w_i] \pmod n$; $i = i + 1$;
19: }
20: Return(v);

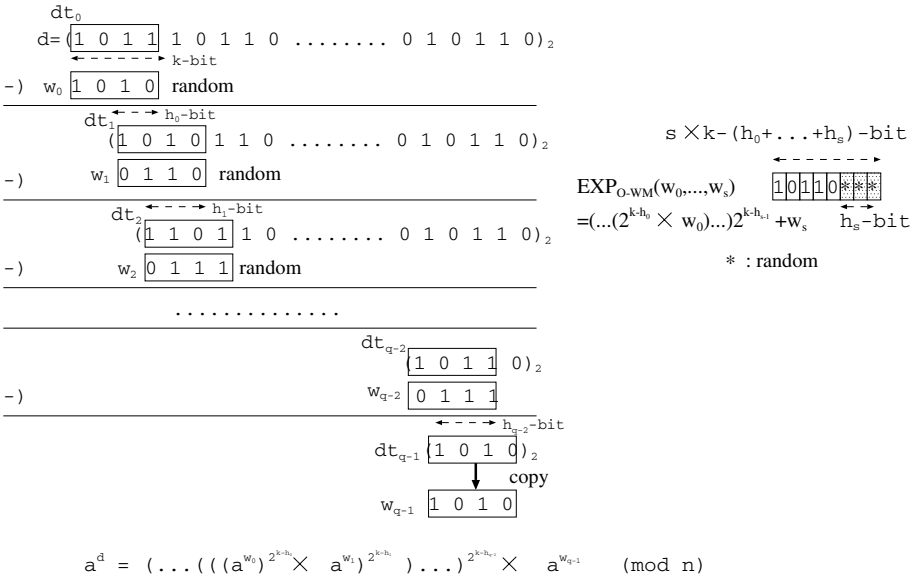

Fig. 1. Overview of O-WM and $EXP_{O-WM}(w_0, \ldots, w_s)$

Algorithm 2. RT-WM (Randomized Table Window Method)

1: /* Generate random number */
2: r = (b-bit random number);
3: /* pre-computed table data making */
4: $t = a^r \pmod n$
5: for $(i = 0; i < 2; i++)$ tab[i] $= a^{i \times 2^b} \times t \pmod n$
6: /* window making phase */
7: dw = d; q=0;
8: while $(dw \geq r \times 2^{k \times q})$ {
9: $dw = dw - r \times 2^{k \times q}; q = q + 1$
10: }
11: for $(i = 0; i < q; i++)$ {
12: $w_i = bit(dw, b + (q - i) \times k - 1, \ldots, b + (q - i - 1) \times k)$;
13: }
14: $d_m = dw \pmod{2^b}$
15: /* modular exponent process */
16: if $(q == 0)$ Return$(a^{d_m} \pmod n)$;
17: $v = w_0$; $i = 1$;
18: while $(i < q)$ {
19: $v = v^{2^k} \pmod n; v = v \times tab[w_i] \pmod n; i = i + 1$;
20: }
21: /* normalization */
22: $v = v \times a^{d_m} \pmod n$;
23: Return(v);

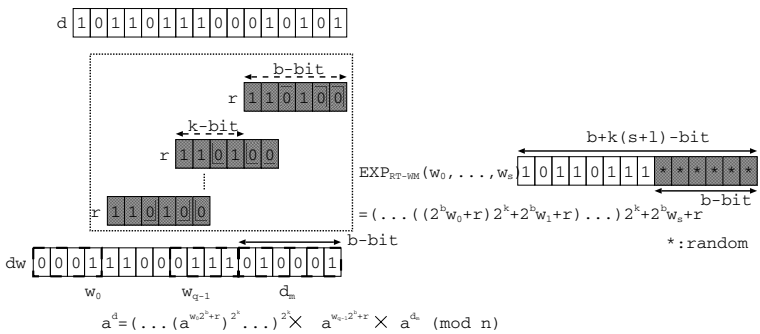

Fig. 2. Overview of RT-WM and $EXP_{RT-WM}(w_0, \ldots, w_s)$

for $i = s$, the $EXP_{RT-WM}()$ is represented as $EXP_{RT-WM}(w_0, \ldots, w_s) = (\ldots((w_0 \times 2^b + r) \times 2^k + w_1 \times 2^b + r) \times 2^k \ldots) \times 2^k + w_s \times 2^b + r$, whose lowest b-bit data is random value (figure 2). To obtain the final result $a^d \pmod n$, the randomized data must be 'normalized' at the end of the operation. This normalization step is $v = v \times a^{d_m} \pmod n$ in step 22, where d_m is b-bit value generated in step 14.

In comparison with the k-ary method, the number of repeating table look-up operations are the same, but the overhead for the computation of table making and normalization are larger.

3.4 Hybrid Randomizing Window Method (HR-WM)

HR-WM is a combination technique of O-WM and RT-WM. We show the steps in algorithm 3 and an overview in figure 3. Pre-computed table data generation

Algorithm 3. HR-WM(Hybrid Randomizing Window Method)

1: /* Generate random number */
2: r = (b-bit random number);
3: /* pre-computed table data making */
4: $t = a^r \pmod n$
5: for $(i = 0; i < 2^k; i++)$ $tab[i] = a^r \pmod n$;
6: /* window making phase */
7: $dw = d; q = 0$;
8: while $(dw \geq r \times 2^{k \times q})$ {
9: $dw = dw - r \times 2^{k \times q}; q = q + 1$;
10 }
11: $d_m = dw \pmod{2^b}$; $dw = dw/2^b$;
12: $u' = (k-h)^{(q-1)}$; $dt_0 = bit(dw, u' + k - 1, .., u')$;
13: for $(i = 0; i < q - 1; i++)$ {
14: $w_i = (Randomnumber, max(0, dt_i - 2^h + 1) \leq w_i \leq dt_i)$;
15: $dt_{i+1} = (dt_i - w_i)2^{(k-h)} + bit(dw, u' - 1, \ldots, u' - (k-h))$;
16: $u' = u' - (k-h)$;
17: }
18: $w_{q-1} = dt_{q-1}$;
19: /* modular exponent process */
20: if $(q == 0)$ Return$(a^{d_m} \pmod n)$;
21: $v = w_0$; $i = 1$;
22: while $(i < q)$ {
23: $v = v^{(2^{(k-h)})} \pmod n$; $v = v \times tab[w_i] \pmod n$; $i = i + 1$;
10 }
24: /* normalization */
25: $v = v \times a^{d_m} \pmod n$;
26: Return (v);

is the same as that in RT-WM, and w_i is generated by combination operation of RT-WM and O-WM (steps.2-18). In HR-WM, the overlapping length h is a fixed value. If step 23 is finished for $i = s$, the $EXP_{HR-WM}()$ is represented as $EXP_{HR-WM}(w_0, \ldots, w_s) = (\ldots(w_0 \times 2^b + r) \times 2^{k-h} \ldots) \times 2^{k-h} + w_s \times 2^b + r$, whose lowest $(b+h)$-bit data is random value (figure 3).

Similar to RT-WM, overhead the computation of table making and normalization are larger in proportion to b and h. But when HR-WM is used, security

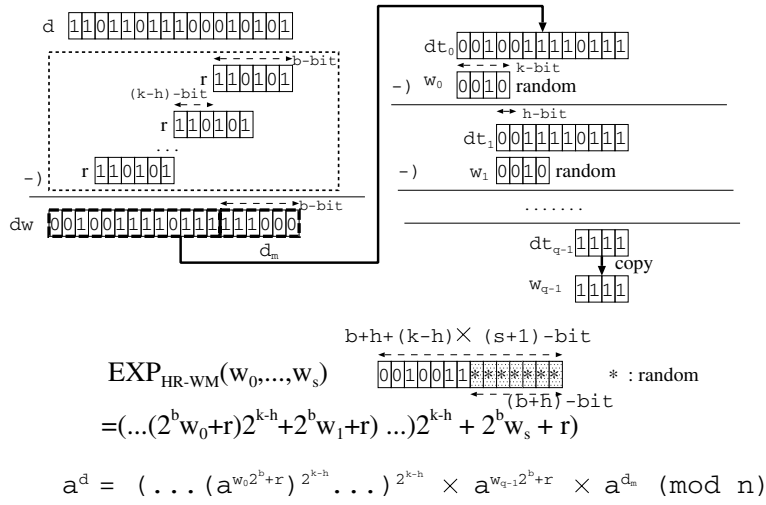

Fig. 3. Overview of HR-WM and $EXP_{HR-WM}(w_0, \ldots, w_s)$

is also strengthened by both effect of O-WM and RT-WM. So, these parameters can be smaller than that in O-WM and RT-WM for attaining the same security level. By setting parameters b and h, an optimal balance between performance and security can be chosen. In HR-WM, h can be small value such as $h = 1$, because h is not limited to $h \geq k/2$.

4 Security Evaluation against DPA

In this section, we evaluate the security of our countermeasures. At first, we describe a basic idea of the security evaluation, then discuss the security of each countermeasure. We finally show that all of our countermeasures have high security in ECC-3D, and RT-WM and HR-WM have high security in RSA and ECC-2D.

4.1 Basic Idea

Before explaining the basic idea of the security evaluation, we explain the DPA attack against the k-ary method. In this attack, the attacker repeats monitoring the power consumption of a cryptographic device N times when inputting plaintext $a_0, \ldots, a_{N-1}$. We denote these monitored data $V(a_i, t)$ where t is time.

The attacker analyzes d by guessing each $w_0, \ldots, w_s$ according to this order. If he already has guessed the correct $w'_0, \ldots, w'_{s-1}$ that satisfy $\{w'_0, \ldots, w'_{s-1}\} = \{w_0, \ldots, w_{s-1}\}$, he guesses $w'_s = w_s$, then calculates the difference power trace $\Delta(t)$ as shown (1), where e $(0 \leq e < (plaintext\ length))$ is a bit position for calculating the differential. If a spike appears in $\Delta(t)$, w'_s turns out to be correct, otherwise it turns to be incorrect.

$$\Delta(t) = \frac{2}{N}\Big(\sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=1} V(a_j,t) - \sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=0} V(a_j,t)\Big) \quad (1)$$

The attacker's possibility for succeeding in the above analysis depends whether the spike will appear or not. Hence, as the size of the spike is smaller, the analysis is harder. If the size of the spike is almost zero, he can't distinguish the correctness of the guessed w'_s. Therefore, the security against DPA can be evaluated by the size of the spike.

Here we describe the basic idea for the evaluation of the size of the spike when our countermeasure is used. In our method, w_s is randomly chosen value, so equation (1) can be transformed to the following equation (2), where $Prob[X]$ represents the probability that equation X holds. (Note that the differential power trace for $Prob[DAT_{WM}(a_j, w_0, \ldots, w_s) \neq DAT_{WM}(a_j, w'_0, .., w'_s)]$ is 0.)

$$\begin{aligned}
\Delta(t) &= \frac{2}{N}\Big(\sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=1} V(a_j,t) - \sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=0} V(a_j,t)\Big) \\
&= Prob[DAT_{WM}(a_j, w'_0, \ldots w'_s) = DAT_{WM}(a_j, w_0, \ldots w_s)] \\
&\quad \times \frac{2}{N}\Big(\sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=1} V(a_j,t) - \sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=0} V(a_j,t)\Big) \\
&\quad + Prob[DAT_{WM}(a_j, w'_0, \ldots w'_s) \neq DAT_{WM}(a_j, w_0, \ldots w_s)] \\
&\quad \times \frac{2}{N}\Big(\sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=1} V(a_j,t) - \sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=0} V(a_j,t)\Big) \\
&= Prob[DAT_{WM}(a_j, w'_0, \ldots w'_s) = DAT_{WM}(a_j, w_0, \ldots w_s)] \\
&\quad \times \frac{2}{N}\Big(\sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=1} V(a_j,t) - \sum_{bit(DAT_{WM}(a_j,w'_0,\ldots w'_s),e)=0} V(a_j,t)\Big) \quad (2)
\end{aligned}$$

From (2), the size of the spike will be smaller in proportion to $Prob[DAT_{WM}(a_j, w_0, \ldots, w_s) = DAT_{WM}(a_j, w'_0, \ldots, w'_s)]$. Therefore, we evaluate the security by maximum value of the probability that $DAT_{WM}(a_j, w_0, \ldots, w_s)$ is computed in the device. We call maximum value of the probability 'attenuation ratio' (AR) in the rest of this paper. (Note that the probability represents the ratio of the size of

spike of (1).) For evaluating AR, we discuss the NRD of $DAT_{WM}(a_j, w_0, \ldots, w_s)$, then evaluate the $DAT_{WM}(a_j, w_0, \ldots, w_s)$ that appears with highest probability.

4.2 O-WM

In O-WM, evaluation of AR differs among RSA, ECC-2D, and ECC-3D. This means that the NRD of $DAT_{O-WM}(a_j, w_0, \ldots, w_s)$ is evaluated using the NRD of $EXP_{O-WM}(w_0, \ldots, w_s)$ for RSA and ECC-2D, and is roughly evaluated using the NRD of the sequence $\{w_0, \ldots, w_s\}$ for ECC-3D. Therefore, we discuss RSA/ECC-2D and ECC-3D separately.

Note 2. The difference between ECC-2D and ECC-3D is due to the difference of the data representation in the projective coordinates. For example, when calculating $7A$ for $A = (X, Y, Z)$ by $(X_1, Y_1, Z_1) = 2((11)_2(X, Y, Z)) + (01)_2(X, Y, Z)$ or $(X_2, Y_2, Z_2) = 2((10)_2(X, Y, Z)) + (11)_2(X, Y, Z)$, these two points represent the same point in affine coordinates, but $X_1 \neq X_2, Y_1 \neq Y_2, Z_1 \neq Z_2$ will hold with high probability.

To see this fact, let us assume that the device computes $B_1 = f_1A + g_1A$ or $B_2 = f_2A + g_2A$ in projective coordinates for point A and scalar values f_1, f_2, g_1 and g_2, when $f_1 + g_1 = f_2 + g_2, f_1, f_2 > g_1, g_2$, and the data representation of f_1A, f_2A are different. Under this assumption, data representation of B_1 and B_2 will be different with probability $1 - 1/p$ where p is the size of the finite field.

In general, NRD of the data representation of $B_z = f_xA + g_yA$ is approximated to $(NRD\ of\ data\ representation\ of\ f_xA) \times (NRD\ of\ data\ representation\ of\ g_yA)$ when the data representation of f_xA are different from each other and $f_x > g_y$ for any x, y. So, $(NRD\ of\ data\ representation\ of\ EXP_{O-WM}(w_0, \ldots, w_s)A)$ is approximated to $(NRD\ of\ data\ representation\ of\ 2^{k-h_{s-1}}\ (EXP_{O-WM}\ (w_0, \ldots, w_{s-1}))A) \times (NRD\ of\ data\ representation\ of\ w_sA)$, which is equal to the NRD of the sequence $\{w_0, \ldots, w_s\}$.

RSA/ECC-2D. From figure 1, $EXP_{O-WM}(w_0, \ldots, w_s)$ is $s \times k - (h_0 + \ldots + h_{s-1})$-bit and lowest h_s-bit is randomized. Therefore, the probability (or AR) that some $EXP_{O-WM}(w_0, ..., w_s)$ is used in the device, is represented as (3) for some W_{len} and W_{val}.

$$(Prob[s \times k - (h_0 + \ldots + h_{s-1}) = W_{len}]) \times (Prob[lowest\ h_s bits = W_{val}]) = \alpha(s, W_{len}) \times \beta(s, W_{val}) \quad (3)$$

$\alpha(s, W_{len})$ depends on $h_0, \ldots, h_{s-1}$ and $\beta(s, W_{val})$ depends on h_s and w_{s+1}, which are independent each other. Therefore, the maximum value of (3) is a product of each maximum value.

$\alpha(s, W_{len})$ equals to the maximum value when $h_0 + h_1 + \ldots + h_{s-1} = s \times k/2$ (note that $0 < h_i < k$). It can be calculated directly, or is approximated as a normal distribution by the central limit theorem, if s is large enough. When h_i is fixed value h, $\alpha(s, W_{len}) = 1$.

$\beta(s, W_{val})$ is a probability that the lowest h_s-bit is equal to W_{val}. Taking into account that h_s varies 1 to $k-1$, it is easy to see that varying only LSB can occur for all h_i (Note that 1-bit varying is included in h_i-bit varying.) The probability is represented as $(2^{-1} + ... + 2^{-(k-1)})/(k-1)$, where $(k-1)$ is a number of possible value h_i.

We show the graph of maximum value of AR in figure 4 when $k = 4$ and $1 \leq h_i \leq 3$. This graph can be approximated to $0.15 \times s^{-1/2}$, decreases slowly for s. So this is thought as 'weak' DPA countermeasure, but it can be used to the device whose SNR (signal-to-noise ratio) is small. Detail of the SNR is described in [6].

Here we note that the probability is a mean value. It depends on the partial value of the secret key d that decides the variable range of w_s. (See step 10 in algorithm 1.) As the partial k-bit of d corresponding to w_s is smaller, $\beta(s, W_{val})$ will be larger. (ex. In figure 1, partial k-bit of d corresponding to w_0 is $(1011)_2$, that to w_1 is $(1110)_2$ and that to w_2 is $(1101)_2$.) When h_i is fixed value h, $\beta(s, W_{val}) = 2^{-h}$.

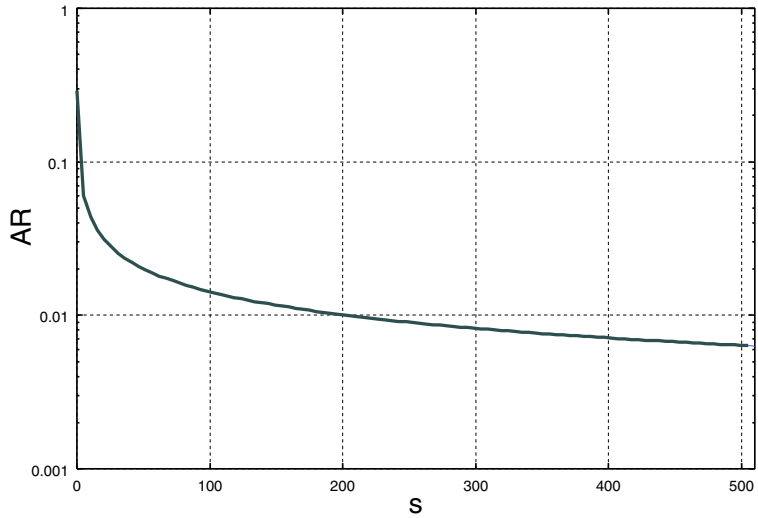

Fig. 4. AR when $k = 4$ and $1 \leq h_i \leq 3$(RSA, ECC-2D)

ECC-3D. Following (4) represents the probability that some sequence $\{w_0, \ldots, w_s\}$ is used in the device.

$$\begin{aligned}(Prob[s \times k - (h_0 + \ldots + h_{s-1}) = W_{len}]) \times 2^{-(h_0+h_1+\ldots+h_s)} \\ = \alpha(s, W_{len}) \times 2^{-(h_0+\ldots+h_s)} \end{aligned} \quad (4)$$

If h_i ranges $0 < h_i < k$, the upper bound of (4) is $2^{-(s+1)}$, and if h_i is a fixed value h, (4) is equal to $2^{-h \times (s+1)}$.

4.3 RT-WM

When RT-WM is used, intermediate encryption data is randomized by the pre-computed table data, which is given by the b-bit random number. So, AR is always equal to 2^{-b}.

4.4 HR-WM

When HR-WM is used, NRD is represented as $(NRD\ when\ using\ O-WM)\times (NRD\ when\ using\ RT-WM)$. If the encryption algorithm is RSA/ECC-2D, NRD is $2^h \times 2^b$. In HR-RM, intermediate data at any given moment is randomly chosen from one of the possible values. Therefore, AR is equal to $2^{-(h+b)}$. In ECC-3D, NRD is $2^{h(s+1)} \times 2^b$, so that AR is equal to $2^{-(h(s+1)+b)}$.

Table 1. Measured AR from DPA experiment (O-WM in RSA $k = 4, 1 \leq h_i \leq 3$)

w_s		w_0	w_3	w_6	w_9	w_{12}
AR (expected)		0.0400	0.104	0.0775	0.064	0.0556
AR (experiment)	spike1	0.0411	0.0863	0.0910	0.0672	0.113
	spike2	0.0414	0.0915	0.109	0.0825	0.120
	spike3	0.0373	0.0887	0.101	0.0653	0.0760
	spike4	0.0398	0.115	0.140	0.0815	0.128
partial 4-bit of d		$(1101)_2$	$(0111)_2$	$(0110)_2$	$(1111)_2$	$(0000)_2$

5 DPA Attack Experiment

For O-WM in RSA case, we have verified effect of the protection against DPA through the experiment. We monitored the power consumption for the RSA encryption by using 4-ary and O-WM in which $k = 4, 1 \leq h_i \leq 3$ and analyzed the secret key. When monitoring the power consumption, we have input 20000 plaintexts and set the sampling ratio 100 MHz. We have analyzed the key by making the difference power trace when guessing w_0, w_3, w_6, w_9 and w_{12}, and confirmed the spike to measure AR. In the analysis, we guessed the sequence $\{w_0, \ldots, w_s\}$ when the size of the spike that appears in (2) is maximum value.

Figure 5 shows the example of the differential power trace, and table 1 shows the expected and measured AR for 4 spikes appeared in the differential power trace. In table 1, partial 4-bit of d corresponding to w_s is also shown. The expected AR is well approximated to the measured AR, and when the partial 4-bit value of d is small, the measured AR is larger than the expected value.

6 Performance Comparison

In table 2, we show the comparison of the performance and security of our countermeasures. The input bit-length of the pre-computed table data is fixed

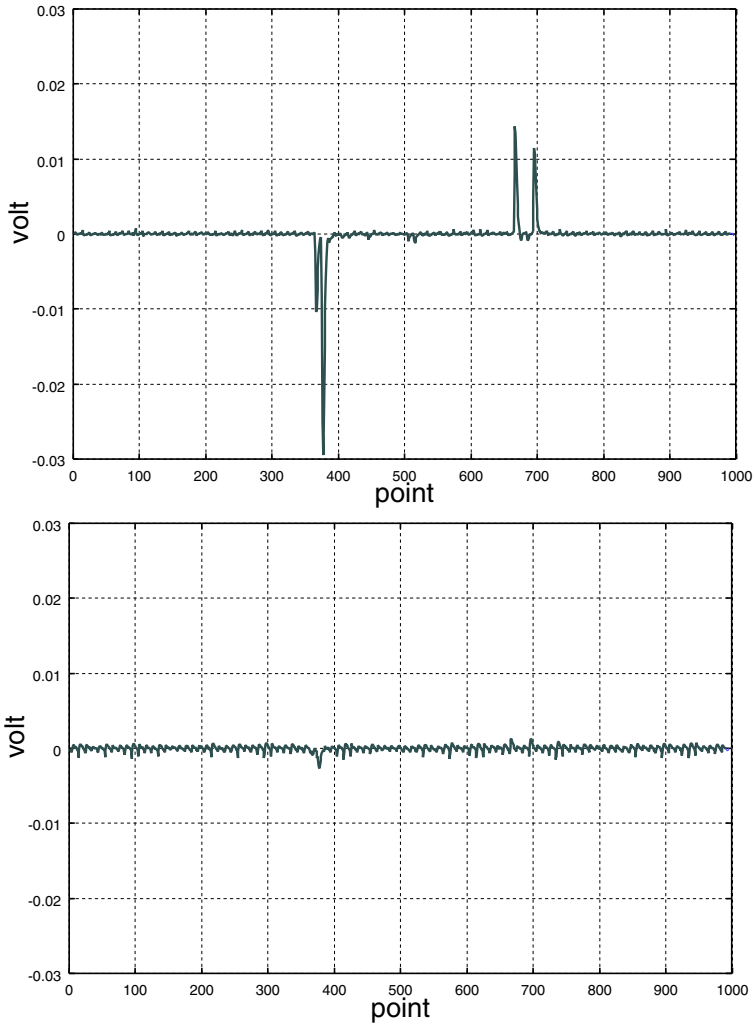

Fig. 5. Differential power traces when guessing w_3 (4-ary:upper, O-WM:lower)

Table 2. Performance comparison among proposed countermeasures

		O-WM	RT-WM	HR-WM
Time	table making	2^k	$b(2+\frac{1}{k'})+2^{k'}+2^k$	$b(2+\frac{1}{k'})+2^{k'}+2^k$
	exponent	$u(S+\frac{M}{k-h})$	$(u-b)(S+\frac{M}{k})$	$(u-b)(S+\frac{M}{k-h})$
	normalization	-	$b(S+\frac{M}{k'})+2^{k'}$	$b(S+\frac{M}{k'})+2^{k'}$
Security (AR)	RSA,ECC-2D	$Cs^{-1/2}(h_i$:random) $2^h(h_i$:fixed)	2^{-b}	$2^{-(h+b)}$
	ECC-3D	$\leq 2^{s+1}$ (h_i:random) $2^{h(s+1)}$ (h_i:fixed)	2^{-b}	$2^{-(h(s+1)+b)}$

Table 3. Performance Comparison with other countermeasures

		O-WM	RT-WM	HR-WM	Coron	Messerges
ECC (160-bit)	Addition	256	279	264	241	214
	AR	$\sim 2^{-6}$(2D) $\sim 2^{-80}$(3D)	2^{-20}	2^{-11}(2D) 2^{-61}(3D)	2^{-20}	$2^{-7.32}$
RSA (1024-bit)	Multiplication	1552	1359	1416	1321	1536
	AR	$\sim 2^{-7.23}$	2^{-20}	2^{-11}	2^{-20}	2^{-10}
Additional parameter		No	No	No	Yes	No

to k-bit among these countermeasures, so that the RAM size of the table data are the same. S represents a computation time of squaring (or doubling), and M represents that of multiplication (or addition). In the 'table making' row, we assumed $S = M$ and showed the performance by the times of the multiplication (or addition). (Note that S and M are omitted in this row.) In the 'O-WM' column, h represents the average value of h_i, and AR is represented when each window method is processed from w_0 to w_s.

In table 3, comparison of our countermeasures with the Coron's (countermeasure (A) in section 2) and Messerges-Dabbish-Sloan's (countermeasure (B) in section 2) countermeasure are shown. In our countermeasures, the parameter are set to $k = 4$ (O-WM, RT-WM, HR-WM), $h = 2$ (O-WM) /1 (HR-WM), $b = 20$(RT-WM) /10 (HR-WM). In Coron's countermeasure, we suppose that the length of the random value is 20-bit, and 4-ary method is used. In Messerges' countermeasure, we supposed that binary-method is used for RSA, and signed-binary method is used for ECC.

We have evaluated the performance of these countermeasures in 1024-bit RSA and 160-bit ECC case, assuming the computation time for squaring and multiplication (doubling and addition) are the same.

6.1 Countermeasure Choice for an Encryption Algorithm

Suitable choice of our countermeasures depends on the encryption scheme and the environment of the device. We categorize them by encryption algorithm as followings.

- RSA/ECC-2D : RT-WM or HR-WM is suitable, O-WM is not recommended. In table 2, HR-WM looks like to be most suitable, but it is because parameters b are different between RT-WM and HR-WM. When parameters b are the same in these two methods, RT-WM is most suitable.
- ECC-3D : All countermeasures are suitable, but the countermeasure can be chosen according to the requirement. When the code size is required to be small, O-WM is suitable, because its computation steps are simple, similar to the k-ary method. Moreover, we recommend to fix h_i for securing against SPA attack. When the encryption speed is significant, suitable countermeasure depends on the bit length of the key. When using the short length key, O-WM is suitable. When using longer length key, HR-WM and RT-WM will be suitable in this order.

7 Conclusion

We proposed three DPA countermeasures based on the window method, O-WM, RT-WM and HR-WM. For O-WM, we assured the effect of the countermeasure through the DPA experiment. When choosing the optimal countermeasure according to the encryption scheme, the computation time ratio to k-ary method is only 105% in RSA and 119% in ECC.. In comparison with the Coron's countermeasure, our countermeasure has the merit that additional parameter is unnecessary. In comparison with the Messerges' countermeasure, encryption speed of our countermeasure is 13% faster in RSA. Except O-WM in which overlapping length is fixed, our countermeasure can protect against SPA by observing only one time execution of the cryptographic device, because square and multiplication are repeated by the constant pattern.

References

1. Christophe Clavier and Marc Joye, "Universal Exponentiation Algorithm – A First Step Towards Provable SPA Resistance", CHES 2001, LNCS 2162, pp. 300–308, Springer-Verlag, 2001.
2. Jean-Sébastien Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems", CHES'99, LNCS 1717, pp. 292–302, Springer-Verlag, 99.
3. Marc Joye and Christophe Tymen, "Protections against Differential Analysis for Elliptic Curve Cryptography", CHES 2001, LNCS 2162, pp. 377–390, Springer-Verlag, 2001.
4. Paul Kocher, Joshua Jaffe, and Benjamin Jun, "Differential Power Analysis", Advances in Cryptography-CRYPTO'99, pp. 388–397.
5. P.-Y. Liardet and N.P. Smart, "Preventing SPA/DPA in ECC Systems Using the Jacobi Form", Cryptographic Hardware and Embedded Systems, CHES 2001, pp. 391–401.
6. Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Investigations of Power Analysis Attacks on Smartcards, ", Proceedings of USENIX Workshop on Smartcard Technology, May 1999, pp. 151–161.
7. Thomas S. Messerges, Ezzy A. Dabbish and Robert H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards.", Cryptographic Hardware and Embedded Systems, CHES'99, LNCS 1717, pp. 144–157.
8. Bodo Moller, "Securing Elliptic Curve Point Multiplication against Side-Channel Attacks", Information Security-ISC 2001, pp. 324–334.
9. K. Okeya, K. Miyazaki and K. Sakurai, "A fast scalar multiplication method with randomized coordinates on a Montgomery-form elliptic curve secure against side channel attacks", ICISC 2001, LNCS 2288, pp. 428–439, Springer-Verlag, 2001.
10. Elisabeth Oswald and Manfred Aigner, "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks", Cryptographic Hardware and Embedded Systems, CHES 2001, pp. 39–50.