Power Analysis to ECC Using Differential Power between Multiplication and Squaring

Toru Akishita[1] and Tsuyoshi Takagi[2]

[1] Sony Corporation, Information Technologies Laboratories, Tokyo, Japan
akishita@pal.arch.sony.co.jp
[2] Future University - Hakodate, Japan
takagi@fun.ac.jp

Abstract. Power analysis is a serious attack to implementation of elliptic curve cryptosystems (ECC) on smart cards. For ECC, many power analysis attacks and countermeasures have been proposed. In this paper, we propose a novel power analysis attack using differential power between modular *multiplication* and modular *squaring*. We show how this difference occurs in CMOS circuits by counting the expectation of signal transition frequency, and present a simulation result on our ECC co-processor. The proposed attack is applicable to two efficient power analysis countermeasures based on unified addition formulae and elliptic curves with Montgomery form.

Keywords: smart cards, elliptic curve cryptosystems, power analysis, DPA, modular multiplication

1 Introduction

Elliptic Curve Cryptosystems (ECC) offer the same level security with much shorter key length than RSA cryptosystems, so that they are suitable for implementing on resource-constraint devices such as smart cards. In recent years, a new class of attacks has been proposed to extract some secret information from a cryptographic device using side channel information (execution time, power consumption, etc.), that are called side channel attacks. Power analysis attacks, the most typical side channel attacks, are real threats to smart cards since the power consumption during cryptographic protocols is relatively dominant in such devices. These attacks include Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [11]. SPA utilizes a power consumption trace during a single execution, whereas DPA requires many power consumption traces and analyzes them with statistical tools.

For ECC, many SPA/DPA attacks and countermeasures have been investigated since Coron generalized power analysis attacks to a scalar multiplication dP [5], where d is a secret scalar and P is a point on an elliptic curve. In 2002, Brier and Joye proposed unified addition formulae that make a point addition and a point doubling indistinguishable on an elliptic curve with Weierstrass form [4]. This indistinguishability guarantees SPA-resistance and enables the

use of efficient addition chains such as Non-Adjacent Form (NAF) and so-called window methods. In the meantime, Montgomery ladder always repeats a point addition and doubling, thus an SPA attacker cannot know any bit information of a secret scalar [16]. Montgomery ladder on an elliptic curve with Montgomery form requires much less costs than that on an elliptic curve with Weierstrass form [15, 16]. One can easily enhance these two SPA countermeasures to be DPA-resistant by combining them to randomized projective representation [5, 17] or randomized curve isomorphism [9].

In this paper, we propose a novel attack to these DPA countermeasures. We firstly describe the difference of power consumption between modular *multiplication* and modular *squaring*. Messerges et al. experimented DPA attacks to modular exponentiation using distinguishability between *multiplication* and *squaring* [13], but there is no investigation about this bias in CMOS circuits. We give detailed descriptions of the bias by estimating the transition probability for each gate in carry-save adder tree, which is a main component of a multiplier, during Montgomery modular multiplication algorithm. We performed a net-list timing simulation of our ECC co-processor and confirmed sharp peaks in the difference between Montgomery modular *multiplication* and Montgomery modular *squaring*.

Secondly, we apply this bias to the above mentioned two SPA/DPA countermeasures. For unified addition formulae, an attacker can distinguish whether the formulae work as a point addition or doubling, and detect a secret scalar. For Montgomery ladder on a Montgomery-form curve, we utilize a "special" point that equalizes both inputs of a certain modular multiplication in a point doubling. The point satisfies $x^2 + (A-4)x + 1 = 0$, and the proposed attack is effective to ECC on any curve that has this point.

The rest of this paper is organized as follows: in section 2 we briefly review power analysis attacks and countermeasures to ECC. Section 3 provides detailed description of the bias between Montgomery modular *multiplication* and Montgomery modular *squaring* together with a simulation result. In section 4, we apply the bias to two power analysis countermeasures. Finally, we conclude in section 5.

2 Elliptic Curve Cryptosystems and Power Analysis

In this section, we introduce power analysis attacks and countermeasures against elliptic curve cryptosystems, including unified addition formulae and elliptic curves with Montgomery form. More details are described in [3, Chapter IV and V].

2.1 Elliptic Curve Cryptosystems

The Weierstrass form of an elliptic curve E_W over a prime field $\mathbb{F}_p$ ($p > 3$) is represented by

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

Input: an n-bit scalar d, a point P
Output: scalar multiplication dP
1. $Q \leftarrow P$
2. For $i = n-2$ downto 0 do:
2.1. $Q \leftarrow \mathrm{ECDBL}(Q)$
2.2. if $d_i = 1$ then
$Q \leftarrow \mathrm{ECADD}(Q, P)$
3. Return(Q)

Table 1. Binary method

Input: an n-bit scalar d, a point P
Output: scalar multiplication dP
1. $Q[0] \leftarrow P$
2. For $i = n-2$ downto 0 do:
2.1. $Q[0] \leftarrow \mathrm{ECDBL}(Q[0])$
2.2. $Q[1] \leftarrow \mathrm{ECADD}(Q[0], P)$
2.3. $Q[0] \leftarrow Q[d_i]$
3. Return($Q[0]$)

Table 2. Double-and-add-always method

The set of all points on E_W and a point of infinity $\mathcal{O}$ forms an additive group, where $\mathcal{O}$ is a neutral element. Let $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ be points on E_W. The addition $P_2 = (x_2, y_2) = P_0 + P_1$ is defined in different formulae depending on whether $P_0 = P_1$ or not as following: $x_2 = \lambda^2 - x_0 - x_1$, $y_2 = \lambda(x_0 - x_2) - y_0$, where $\lambda = (y_1 - y_0)/(x_1 - x_0)$ for $P_0 \neq \pm P_1$, and $\lambda = (3{x_0}^2 + a)/(2y_0)$ for $P_0 = P_1$. We call $P_0 + P_1$ ($P_0 \neq \pm P_1$) an elliptic curve addition (ECADD) and $P_0 + P_1$ ($P_0 = P_1$), namely $2P_0$, an elliptic curve doubling (ECDBL). In practice, both ECADD and ECDBL are implemented in Jacobian coordinates by $x = X/Z^2, y = Y/Z^3$ because an inversion is much more expensive than any other arithmetic (addition, subtraction, multiplication) over $\mathbb{F}_p$. In this case, both are also implemented with different formulae.

In order to construct Elliptic Curve Cryptosystems (ECC) we need to compute a scalar multiplication: computing a point $dP = \underbrace{P + \cdots + P}_{d}$ given a scalar d and a point P. On the other hand, the security of ECC is based on the hardness of Elliptic Curve Discrete Logarithm Problem (ECDLP): computing d given P and dP. Therefore, in most ECC protocols, d is used as a secret key; P and dP are made to be public. The basic method to compute a scalar multiplication is called as the binary method. Let $d = (d_{n-1} \cdots d_1 d_0)_2$ be a binary representation of d. The binary method is shown in Table 1.

2.2 Power Analysis Attacks and Countermeasures

Power analysis attacks are serious on resource-constraint devices such as smart cards. An attacker can successfully reveal some secret information by observing the power consumption on a device during cryptographic protocols. Simple Power Analysis (SPA) and Differential Power Analysis (DPA) are typical ones; SPA requires a power consumption trace during only a single execution, whereas DPA utilizes many power consumption traces with statistical analysis [11]. These attacks utilize a correlation between secret information and power consumption, and are also applicable to ECC.

The binary method shown in Table 1 is vulnerable to SPA. It computes ECADD only when $d_i = 1$, although ECDBL is always computed regardless of d_i. ECADD and ECDBL are different operations as described above, and thus an attacker can easily distinguish ECDBL and ECADD by observing a power

consumption trace and detect secret bit d_i. Many SPA countermeasures against ECC have been proposed, and they are principally divided into two types as follows.

(S1) repeating regular pattern
(S2) unifying ECADD and ECDBL

(S1) includes the double-and-add-always method in Table 2 [5], which repeats ECDBL and ECADD by appending dummy ECADD to the binary method, and Montgomery ladder [15], which is discussed in section 2.4. (S2) includes Hessian curves [10], Jacobi curves [12], and unified addition formulae [4], which is shown in section 2.3.

The resistance against SPA doesn't always guarantee the resistance against DPA because a power consumption trace depends on not only a type of operations, namely ECADD or ECDBL, but also intermediate values. A DPA attacker collects many power consumption traces during the scalar multiplication and guesses a bit of the secret scalar by analyzing correlation between these traces and intermediate values [5]. In order to resist DPA, intermediate values must be randomized. There are three standard randomization methods as follows.

(D1) blinding scalar [11, 5]
(D2) randomized projective representation [5, 17]
(D3) randomized curve isomorphism [9]

These three DPA countermeasures together with the SPA countermeasure (S1) or (S2) enables SPA/DPA resistance. (D1), however, requires more additional costs than (D2) and (D3), so that the combination of either (S1) or (S2) and either (D2) or (D3) is thought to be an optimal SPA/DPA countermeasure.

In 2003, Goubin presented a new power analysis attack called Refined Power Analysis (RPA) [6]. This attack utilizes a "special" point $(x, 0)$ or $(0, y)$ that can be fully randomized by neither (D2) nor (D3). In the addition, an attacker can pick up the point only in a few power consumption traces because a power trace in processing this point is distinctive [2]. RPA with a point $(x, 0)$ can be easily discarded by multiplying co-factor on the underlying curve to an input point P since the order of $(x, 0)$ is 2 [18]. On the other side, RPA with a point $(0, y)$ cannot be discarded because $(0, y)$ has large order. Therefore, RPA is effective on a curve that has a point $(0, y)$.

We extended RPA to Zero-value Point Attack (ZPA) using other "special" points [2]. We pointed out that, even if a point has no zero coordinate, intermediate values in addition or doubling formulae might become zero. ZPA using a point addition is actually difficult for a large scalar d, but ZPA using a point doubling is as effective as RPA if the point that causes zero-value in a point doubling exists on the underlying curve. Therefore, RPA and ZPA using a point doubling may oblige not only (D2) or (D3) but also (D1), or another countermeasure such as randomized initial point countermeasure [7, 14], which leads to extra costs or memories.

2.3 Unified Addition Formulae

Brier and Joye proposed "unified addition formulae" for an elliptic curve addition (ECADD) and an elliptic curve doubling (ECDBL) on a Weierstrass-form elliptic curve as an SPA countermeasure [4]. In their formulae for affine coordinates, a denominator becomes no longer zero in ECDBL as follows.

Unified Addition Formulae $P_2 = P_0 + P_1$

$$x_2 = \left(\frac{{x_1}^2 + x_1x_0 + {x_0}^2 + a}{y_1 + y_0}\right)^2 - x_0 - x_1$$
$$y_2 = \left(\frac{{x_1}^2 + x_1x_0 + {x_0}^2 + a}{y_1 + y_0}\right)(x_0 - x_2) - y_0$$

Izu and Takagi proposed the exceptional procedure attack that forces a denominator to become zero for ECADD by inputting two points (x_0, y_0) and (x_1, y_1), which satisfy $y_0 + y_1 = 0$, but it seems difficult to find a couple of such points for a large scalar [8].

An SPA attacker cannot distinguish whether the unified formulae work as ECADD or ECDBL, thus she knows only a hamming weight of a secret scalar d even if the binary method is used. Moreover, an efficient addition chain such as Non-Adjacent Form (NAF) or window-based methods leads great efficiency. The unified addition formulae for projective coordinates were also proposed. The combination with a DPA countermeasure (D2) or (D3) enables the compatibility of efficiency and SPA/DPA resistance.

Remark 1. In [21], Walter proposed an SPA attack to unified addition formulae based on the existence of a final subtraction in Montgomery modular multiplication. This attack, however, is easily eliminated by computing a final subtraction in any case.

2.4 Elliptic Curve with Montgomery-Form

The Montgomery form of an elliptic curve E_M over $\mathbb{F}_p$, $(p > 3)$ was proposed by Montgomery to speed up integer factorization [15], and represented by

$$E_M : By^2 = x^3 + Ax^2 + x \quad (A, B \in \mathbb{F}_p,\ (A^2 - 4)B \neq 0).$$

About 40% of elliptic curves with Weierstrass form are transformed into Montgomery form, and the order of a Montgomery-form elliptic curve is always divisible by 4 [16]. On a Montgomery-form elliptic curve E_M, x-coordinate of the addition of two points can be computed without y-coordinate if x-coordinate of the difference of these points is known. Affine coordinate x is transformed into projective coordinates $(X : Z)$ by $x = X/Z$. Let $P_0 = (X_0 : Z_0)$ and $P_1 = (X_1 : Z_1)$ be points on E_M. In the following we describe Montgomery addition formulae $P_2 = (X_2 : Z_2) = P_0 + P_1$, where $P_0 \neq \pm P_1$ and $P' = (X' : Z') = P_0 - P_1$, and Montgomery doubling formulae $P_2 = (X_2 : Z_2) = 2P_0$.

Input: an n-bit scalar d, a base point P
Output: scalar multiplication dP
1. $Q[0] \leftarrow P,\ Q[1] \leftarrow 2P$
2. For $i = n-2$ downto 0 do:
2.1. $Q[1-d_i] \leftarrow \text{mECADD}([Q[0], Q[1])$
2.2. $Q[d_i] \leftarrow \text{mECDBL}([Q[d_i])$
3. Return($Q[0]$)

Table 3. Montgomery ladder

Montgomery Addition Formulae (mECADD) $P_2 = P_0 + P_1$ $(P_1 \neq \pm P_0)$

$$X_2 = Z'((X_0 - Z_0)(X_1 + Z_1) + (X_0 + Z_0)(X_1 - Z_1))^2$$
$$Z_2 = X'((X_0 - Z_0)(X_1 + Z_1) - (X_0 + Z_0)(X_1 - Z_1))^2$$

Montgomery Doubling Formulae (mECDBL) $P_2 = 2P_0$

$$4X_0Z_0 = (X_0 + Z_0)^2 - (X_0 - Z_0)^2$$
$$X_2 = (X_0 + Z_0)^2(X_0 - Z_0)^2$$
$$Z_2 = (4X_0Z_0)((X_0 - Z_0)^2 + ((A+2)/4)(4X_0Z_0))$$

A scalar multiplication dP can be computed by so-called Montgomery ladder in Table 3. Montgomery ladder always repeats mECADD and mECDBL whether $d_i = 0$ or 1. Therefore, an SPA attacker cannot guess any bit information of a secret scalar d [16]. Montgomery ladder on an elliptic curve with Weierstrass form was also proposed, but the costs of a point addition and a point doubling on a Weierstrass-form curve are much larger than mECADD and mECDBL. One can enhance Montgomery ladder to be DPA-resistant by applying randomized projective representation [17]. In the addition, RPA and ZPA are easily eliminated by checking whether $4P$ is a point at infinity or not for a input point P because the order of a point $(0, y)$ for RPA, $(-1, y)$ and $(1, y)$ for ZPA are 2, 4 and 4, respectively.

3 Differential Power between Multiplication and Squaring

Here we show that there exists the difference of power consumption between modular *multiplication* and modular *squaring* in CMOS circuits. We estimate the transition probability of each signal in some full adders, and present a result of net-list timing simulation in order to confirm this difference.

3.1 Montgomery Modular Multiplication

We assume the following standard smart card environment. The embedded CPU on a smart card, typically an 8 or 16 bit CPU, has only so poor computing

Input: $M = (M_4 \cdots M_0)_b,\ X = (X_4 \cdots X_0)_b,\ Y = (Y_4 \cdots Y_0)_b,$
$b = 2^{32},\ R = b^5,\ \gcd(m, b) = 1,\ m' = M_0^{-1} \bmod b$
Output: $XYR^{-1} \bmod M$

1. $A \leftarrow 0$ $((A = (A_5 \cdots A_0)_b)$
2. For i from 0 to 4 do:
   2.1. $temp \leftarrow 0$
   2.2. For j from 0 to 4 do:
   $\{temp, A_j\} \leftarrow X_j Y_i + A_j + temp$
   2.3. $A_5 \leftarrow temp,\ temp \leftarrow 0,\ u_i \leftarrow A_0 m' \bmod b$
   2.4. For j from 0 to 5 do:
   $\{temp, A_j\} \leftarrow M_j u_i + A_j + temp$
   2.5. $A \leftarrow A/b$
3. If $A \geq M$ then $A \leftarrow A - M$
4. Return(A)

Table 4. 160-bit Montgomery modular multiplication using a 32-bit multiplier

power that we usually equip a co-processor for implementing ECC. An addition, subtraction, multiplication and inversion over a base field $\mathbb{F}_p$ are implemented in an ECC co-processor to compute elliptic curve operations such as a point addition, point doubling and scalar multiplication. The modular multiplication is the most frequently used among those modular operations.

Recall that Montgomery modular multiplication algorithm is a standard algorithm for computing modular multiplication over general prime fields. In this paper we analyze a 160-bit Montgomery modular multiplication with 32-bit word size is shown in Table 4, which is a standard size in current implementation of ECC. In this algorithm there are three 32-bit multiplications computed by a 32-bit multiplier, namely $X_j Y_i$ in step 2.2, $A_0 m'$ in step 2.3, and $M_j u_i$ in step 2.4.

We will later show that signal transition probability in a 32-bit multiplier during computing $X_j Y_i$ is biased if the inputs of Montgomery modular multiplication, X and Y, satisfy $X = Y$.

3.2 Structure of a Multiplier

In this section we deal with a 6-bit multiplier instead of a 32-bit multiplier because of space limitation. All observations, however, are applicable to a 32-bit multiplier.

In general, a multiplier consists of three stages: Partial Product Generator (PPG), Partial Product Accumulator (PPA) and Final Stage Adder (FSA). The PPG stage generates partial products from multiplicand and multiplier in parallel. The PPA stage then performs multi-operand addition for all partial products and produces their sum in carry-save form. Finally, the carry-save form is converted to the binary output at the FSA stage.

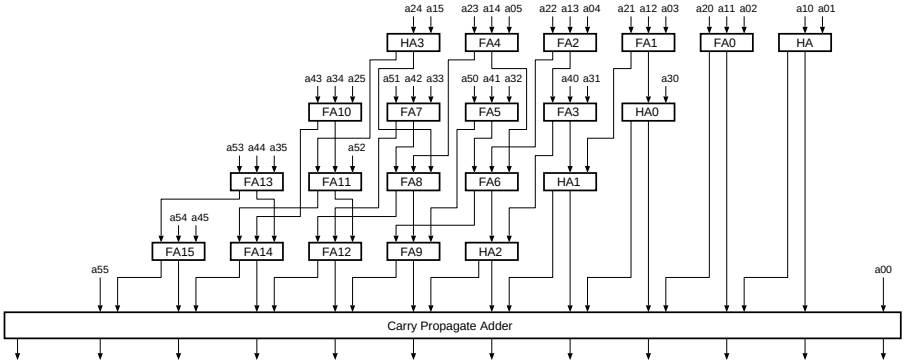

Fig. 1. 6-bit multiplier with Wallace tree

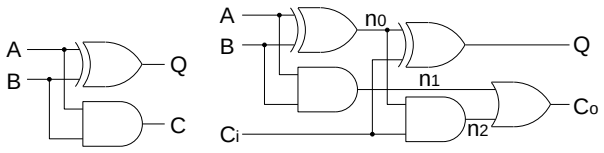

Fig. 2. Half Adder (HA) and Full Adder (FA)

In Fig.1, we show the detailed structure of a 6-bit multiplier $x * y$ using simple XORs as PPG and Wallace tree as PPA, where $x = (x_5x_4x_3x_2x_1x_0)_2$ and $y = (y_5y_4y_3y_2y_1y_0)_2$ are binary representations of x and y, respectively. All partial products $a_{ij} = x_i \& y_j$ for $0 \leq i, j \leq 5$ are summed in carry-save form at Wallace tree that is composed of Half Adders (HA) and Full Adders (FA) shown in Fig.2. In the final stage, a carry propagate adder generates a product from 11-bit sum and 9-bit carry.

3.3 Biased Signal Transition Probability in a Multiplier

Power consumption in CMOS circuits depends on the transition probability of signals without power consumption caused by the leakage current, which is determined by the characteristics of the CMOS process. Therefore, regarding to power analysis, we have only to consider transition probability of signals [19, 20]. When the transition probability in two cases is biased, the difference of power consumption occurs.

Here we estimate the signal transition probability of FA0, FA1, FA2, FA4, FA5, FA7, FA10 and FA13, depth-1 full adders of Wallace tree, in Fig.1. The all three inputs A, B, C_i of these full adders consist of partial products a_{ij}. We consider the following two cases about the inputs (x, y) of the 6-bit multiplier:

(i) transition from (s, t) to (s', t),
(ii) transition from (s, t) to (t, t),

signal	transition	TypeN	TypeA1	TypeA2	TypeA3	TypeA4
A	$0 \to 1$	1/8	1/8	1/8	1/4	1/8
	$1 \to 0$	1/8	1/8	1/8	0	1/8
B	$0 \to 1$	1/8	1/8	1/8	1/8	1/4
	$1 \to 0$	1/8	1/8	1/8	1/8	0
C_i	$0 \to 1$	1/8	1/8	1/4	1/8	1/8
	$1 \to 0$	1/8	1/8	0	1/8	1/8
n_0	$0 \to 1$	3/16	3/16	3/16	1/4	1/4
	$1 \to 0$	3/16	3/16	3/16	1/8	1/8
n_1	$0 \to 1$	9/128	9/64	9/64	3/32	3/32
	$1 \to 0$	9/128	3/64	3/64	1/32	1/16
n_2	$0 \to 1$	3/64	3/54	3/64	3/32	3/32
	$1 \to 0$	3/64	3/64	3/64	1/32	1/32
Q	$0 \to 1$	7/32	1/8	1/4	1/4	1/4
	$1 \to 0$	7/32	5/16	3/16	3/16	3/16
C_o	$0 \to 1$	13/128	5/32	21/128	21/128	5/32
	$1 \to 0$	13/128	1/16	9/128	9/128	1/16
Total		2	33/16	133/64	133/64	33/16

Table 5. Signal transition probability of Full Adders (FAs)

where s, s' and t are 6-bit random values, respectively. $a_{ij} = 1$ generally occurs with probability 1/4 by $a_{ij} = x_i \& y_j$.

In case (i), all the eight FAs have the same transition probability of inputs, internal nodes and outputs as TypeN shown in Table 5. Meanwhile, in case (ii) FA4 and FA5 also have the same probability as TypeN, but the other FAs are divided into four types, which have different probability from TypeN in Table 5, as follows.

TypeA1 $A = C_i$ and $B = a_{ii}$ (FA0, FA13)
TypeA2 $B = C_i$ (FA1, FA10)
TypeA3 $C_i = a_{ii}$ (FA2)
TypeA4 $A = a_{ii}$ (FA7)

This biased transition probability results from the following biased state: $a_{ij} = a_{ji}$ and $a_{ii} = 1$ with probability 1/2 when $(x, y) = (t, t)$. The expectation of the total transition frequency for these FAs in case (ii) is larger by 13/32 than that in case (i). Moreover, the biased transition probability of outputs Q, C_o in these FAs influences transition probability of depth-2/3 adders in Wallace tree and the carry propagate adder. Therefore, the power consumption traces of the 6-bit multiplier in case (i) and (ii) differ, which will not depend on the bit-width and structure of a multiplier — the difference will occur, for example, when using a 32-bit multiplier with booth encoder as PPG and 4:2 compressor tree as PPA.

We denote Montgomery modular multiplication with inputs X and Y satisfying $X = Y$, namely Montgomery modular squaring, by SQR and Montgomery modular multiplication with general X and Y by MUL. Let C_S and C_M be power consumption traces during SQR and MUL, respectively. During SQR, the inputs of multiplier transit from (X_{i-1}, X_i) to (X_i, X_i) at $j = i$ in step 2.2, which corresponds to case (ii). On the other hand, during MUL, the inputs of

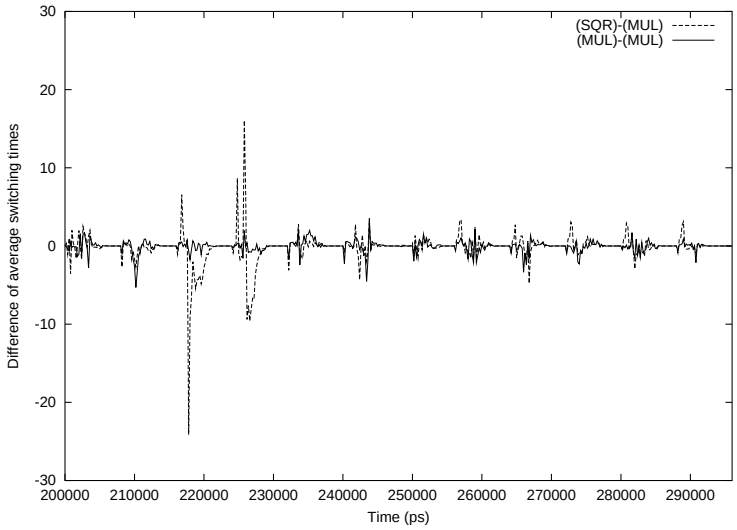

Fig. 3. Simulation result of the differential power between *SQR* and *MUL*

multiplier then transit to from (X_{i-1}, Y_i) to (X_i, Y_i), which corresponds to case (i). Therefore, the difference $\Delta C = C_S - C_M$ will present a peak at $j = i$ in step 2.2. Similarly, ΔC also will show a peak at $j = i + 1$ because the biased transition from biased state to random state occurs in a multiplier during *SQR*.

3.4 Simulation Result

We performed a net-list timing simulation of Montgomery modular multiplication circuits in our ECC co-processor, reported in [1]. We used a 90-nm CMOS standard cell library and then made estimated power consumption traces by counting the number of gate switching times in every 200ps. In Fig.3, the dotted line shows the difference of average switching times between *SQR* and *MUL* for random 10000 inputs at $i = 2$ in step 2.2; the solid line shows the difference between both *MUL* for random 10000 inputs. The dotted line shows sharp peaks at the 3rd and 4th cycles, namely at $j = 2$ and 3.

Remark 2. The power consumption is biased between modular *multiplication* and modular *squaring* for any modular multiplication algorithm because the multiplication of inputs X and Y is required.

4 Application to Elliptic Curve Cryptosystems

We apply the difference of power consumption between *multiplication* and *squaring* to the above-mentioned two DPA countermeasures, namely unified addition formulae and Montgomery ladder on a Montgomery-form curve.

4.1 Attack to Unified Addition Formulae

The distinguishability between modular *multiplication* and modular *squaring* is applicable to an attack to a scalar multiplication dP for a secret scalar d and a point P using unified addition formulae. We notice the modular multiplication x_1x_0, denoted by **MUL1**, in the affine coordinate version of unified addition formulae in section 2.3. The formulae work as ECADD when $x_1 \neq x_0$ and as ECDBL when $x_1 = x_0$. Hence, if an attacker can distinguish whether **MUL1** is a modular *multiplication* or a modular *squaring*, she knows whether the corresponding operation is ECADD or ECDBL and detects bit information of the secret scalar d.

Assume that randomized curve isomorphism is used as a DPA countermeasure. $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$ is transformed to its isomorphic class like $P'_0 = (\lambda^2 x_0, \lambda^3 y_0)$ and $P'_1 = (\lambda^2 x_1, \lambda^3 y_1)$ for a random value $\lambda \in \mathbb{F}_p^*$. In the case, the modular multiplication $(\lambda^2 x_1)(\lambda^2 x_0)$ is computed as **MUL1**. Thus **MUL1** remains a modular *multiplication* when $P'_0 \neq P'_1$ and a modular *squaring* when $P'_0 = P'_1$

In the following, we present the precise algorithm to search the bit of d for a scalar multiplication dP using unified addition formulae. We assume that the scalar multiplication is computed by the binary method. The unified addition formulae is computed $m = l(d) + h(d) - 1$ times during a single scalar multiplication, where $l(d)$ is the bit length of d and $h(d)$ is the hamming weight of d; precisely $h(d) - 1$ times as "A" (ECADD) and $l(d)$ times as "D" (ECDBL).

[Bit search algorithm for unified addition formulae]

1. Measure power consumption traces of dP L times and average them.
2. Extract the average traces C_i $(1 \leq i \leq m)$ when computing **MUL1** during the i-th execution of the formulae.
3. Assume "A" if $\Delta C_i = C_i - C_1$ $(2 \leq i \leq m)$ shows a peak and "D" if not.
4. Regard "DA" as a bit "1" and the remaining "D" as a bit "0".

The first execution of **MUL1** corresponds to a modular *squaring* because the scalar multiplication always computes ECDBL in the beginning. Therefore, if ΔC_i shows a peak, x_1x_0 is a modular *multiplication* and the execution corresponds to ECADD.

Remark 3. The proposed attack is also applicable to the projective coordinate version of unified addition formulae [4].

4.2 Attack to Elliptic Curve with Montgomery-Form

As described in section 2.4, there is no "special" point of small order for RPA and ZPA in Montgomery doubling formulae (mECDBL). Therefore, Montgomery ladder on a Montgomery-form curve together with randomized projective representation is secure against SPA/DPA/RPA/ZPA. $P_0 = (X_0 : Z_0)$ is transformed to its random projective representation like $P_0 = (\lambda X_0 : \lambda Z_0)$ for a random value $\lambda \in \mathbb{F}_p^*$. Here we propose another "special" point that equalizes both inputs of a certain modular multiplication in mECDBL. We notice the modular

multiplication $(4X_0Z_0)((X_0 - Z_0)^2 + ((A+2)/4)(4X_0Z_0))$, denoted by **MUL2**, in mECDBL.

Let $E = 4X_0Z_0$ and $F = (X_0 - Z_0)^2 + ((A+2)/4)(4X_0Z_0)$. **MUL2**, of course, becomes a modular *squaring* when $E = F$. The condition satisfying $E = F$ is that x-coordinate $x_0 = X_0/Z_0$ of P_0 satisfies ${x_0}^2 + (A-4)x_0 + 1 = 0$ by

$$\begin{aligned} E - F &= -({X_0}^2 + (A-4)X_0Z_0 + {Z_0}^2) \\ &= -{Z_0}^2({x_0}^2 + (A-4)x_0 + 1). \end{aligned}$$

Even if projective representation of P_0 is randomized as $P_0 = (\lambda X_0 : \lambda Z_0)$, the condition of $E = F$ still implies ${x_0}^2 + (A-4)x_0 + 1 = 0$ by $-\lambda^2 {Z_0}^2({x_0}^2 + (A-4)x_0 + 1) = 0$.

Let $R = (x_R, y_R)$ of order $\#R$ be the point satisfying ${x_R}^2 + (A-4)x_R + 1 = 0$ and exist on the underlying Montgomery-form curve. If the input point of mECDBL is R, **MUL2** becomes a modular *squaring* despite randomized projective representation. Suppose that a scalar multiplication dP for a secret scalar d and a point P is computed by Montgomery ladder (Table 2) and randomized projective representation, where P can be adaptively chosen by an attacker. Here we assume that she knows $(n-i-1)$ most significant bits $(d_{n-1} \cdots d_{i+1})$ of d. In Table 2, for any given input point P, the points $Q[0]$ and $Q[1]$ obtained at the beginning of the i-th step of the loop are $Q[0] = (\sum_{j=i+1}^{n-1} d_j 2^{j-i-1}) \cdot P$ and $Q[1] = (\sum_{j=i+1}^{n-1} d_j 2^{j-i-1} + 1) \cdot P$. We then have two cases:

- If $d_i = 0$, the input point of mECDBL is $(\sum_{j=i+1}^{n-1} d_j 2^{j-i-1}) \cdot P$.
- If $d_i = 1$, the input point of mECDBL is $(\sum_{j=i+1}^{n-1} d_j 2^{j-i-1} + 1) \cdot P$.

Thus, **MUL2** becomes a modular *squaring* at the i-the step of the loop in the following two cases:

- $d_i = 0$ and $P = [(\sum_{j=i+1}^{n-1} d_j 2^{j-i-1})^{-1} \bmod \#R] \cdot R$,
- $d_i = 1$ and $P = [(\sum_{j=i+1}^{n-1} d_j 2^{j-i-1} + 1)^{-1} \bmod \#R] \cdot R$.

In these cases biased power consumption occurs in **MUL2** compared to a modular *multiplication*.

We present the algorithm to search the bit of a secret scalar d from the most significant bit.

[Bit search algorithm for Montgomery-form curve]

1. Measure power consumption traces for L random input points P and average them by C_t.
2. $i \leftarrow n-2$.
3. Compute $P_0 = [k^{-1} \bmod \#R] \cdot R$ and $P_1 = [(k+1)^{-1} \bmod \#R] \cdot R$ for $k = \sum_{j=i+1}^{n-1} d_j 2^{j-i-1}$.
4. Measure power consumption traces L times for the input point P_0 and average them by C_0.
5. Measure power consumption traces L times for the input point P_1 and average them by C_1.

6. Compute $\Delta C_0 = C_0 - C_t$ and $\Delta C_1 = C_1 - C_t$.
7. Assume that $d_i = 0$ if ΔC_0 during **MUL2** at the i-th step of the loop has larger peaks than ΔC_1 and $d_i = 1$ otherwise.
8. If $i = 0$, terminate; else $i \leftarrow i - 1$ and go to 3.

The average power trace C_t is used as a standard one where **MUL2** is a modular *multiplication* at every step of the loop.

5 Conclusion

We presented detailed descriptions of the biased power consumption between Montgomery modular *multiplication* and Montgomery modular *squaring*. However, it must be emphasized that the bias occurs in any modular multiplication algorithm. We applied this bias to unified addition formulae and Montgomery ladder on a Montgomery-form elliptic curve. We should randomize not only a base point but also a secret scalar for these power analysis countermeasures.

References

1. T. Akishita, K. Iizuka, and H. Sato, "Hardware Implementation of Elliptic Curve Cryptosystems for Contactless IC Card", Proceedings of SCIS 2002, 15B-1, pp.1107-1112, 2002 (in Japanese).
2. T. Akishita and T. Takagi, "Zero-Value Point Attack on Elliptic Curve Cryptosystems", *Information Security - ISC 2003*, LNCS 2851, pp.218-233, Springer-Verlag, 2003.
3. I.F. Blake, G. Seroussi, and N.P. Smart, *Advances in Elliptic Curve Cryptography*, Cambridge University Press, 2005.
4. E. Brier and M. Joye, "Weierstrass Elliptic Curve and Side-Channel Attacks", *Public Key Cryptography - PKC 2002*, LNCS 2274, pp.335-345, Springer-Verlag, 2002.
5. J.-S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems", *Cryptographic Hardware and Embedded Systems - CHES '99*, LNCS 1717, pp.292-302, Springer-Verlag, 1999.
6. L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems", *Public Key Cryptography - PKC 2003*, LNCS 2567, pp.199-211, Springer-Verlag, 2003.
7. K. Itoh, T. Izu, and T. Takenaka, "Efficient Countermeasures against Power Analysis for Elliptic Curve Cryptosystems", *Sixth Smart Card Research an Advanced Application IFIP Conference - CARDIS 2004*, pp.99-114, Kluwer, 2004.
8. T. Izu and T. Takagi, "Exceptional Procedure Attack on Elliptic Curve Cryptosystems", *Public Key Cryptography - PKC 2003*, LNCS 2567, pp.224-239, Springer-Verlag, 2003.
9. M. Joye and C. Tymen, "Protection against Differential Analysis for Elliptic Curve Cryptography", *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pp.377-390, Springer-Verlag, 2001.
10. M. Joye and J.-J. Quisquater, "Hessian Elliptic Curves and Side-Channel Attacks", *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pp.402-410, Springer-Verlag, 2001.

11. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", *Advances in Cryptology - CRYPTO '99*, LNCS 1666, pp.388-397, Springer-Verlag, 1999.
12. P.-Y. Liardet and N.P. Smart, "Preventing SPA/DPA in ECC Systems Using the Jacobi Form", *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pp.391-401, Springer-Verlag, 2001.
13. T.S. Messerges, E.A. Dabbish, and R.H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards", *Cryptographic Hardware and Embedded Systems - CHES '99*, LNCS 1717, pp.144-157, Springer-Verlag, 1999.
14. H. Mamiya, A. Miyaji, and H. Morimoto, "Efficient Countermeasure against RPA, DPA, and SPA", *Cryptographic Hardware and Embedded Systems - CHES 2004*, LNCS 3156, pp.343-356, Springer-Verlag, 2004.
15. P.L. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization", *Mathematics of Computation*, vol.48, pp.243-264, 1987.
16. K. Okeya, H. Kurumatani, and K. Sakurai, "Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications", *Public Key Cryptography - PKC 2000*, LNCS 1751, pp.238-257, Springer-Verlag, 2000.
17. K. Okeya, K. Miyazaki, and K. Sakurai, "A Fast Scalar Multiplication Method with Randomized Projective Coordinate on a Montgomery-Form Elliptic Curve Secure against Side Channel Attacks", *Information Security and Cryptology - ICISC 2001*, LNCS 2288, pp.428-439, Springer-Verlag, 2002.
18. N.P. Smart, "An Analysis of Goubin's Refined Power Analysis Attack", *Cryptographic Hardware and Embedded Systems - CHES 2003*, LNCS 2779, pp.281-290, Springer-Verlag, 2003.
19. D. Suzuki, M. Saeki, and T. Ichikawa, "Random Switching Logic: A Countermeasure against DPA based on Transition Probability", IACR Cryptology ePrint Archive 2004/346, 2004. http://eprint.iacr.org/2004/346/
20. D. Suzuki, M. Saeki, and T. Ichikawa, "DPA Leakage Model for CMOS Logic Circuits", *Cryptographic Hardware and Embedded Systems - CHES 2005*, LNCS 3659, pp.366-382, Springer-Verlag, 2005.
21. C.D. Walter, "Simple Power Analysis of Unified Code for ECC Double and Add", *Cryptographic Hardware and Embedded Systems - CHES 2004*, LNCS 3156, pp.191-204, Springer-Verlag, 2004.