Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA

Kouichi Itoh[1], Tetsuya Izu[2], and Masahiko Takenaka[1]

[1] FUJITSU LABORATORIES Ltd.,
64, Nishiwaki, Okubo-cho, Akashi, 674-8555, Japan
{kito,takenaka}@flab.fujitsu.co.jp
[2] FUJITSU LABORATORIES Ltd.,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
izu@flab.fujitsu.co.jp

Abstract. The differential power analysis (DPA) is a powerful attack against the implementation of cryptographic schemes on mobile devices. This paper proposes an alternative DPA using the addresses of registers of elliptic curve based cryptosystems (ECC) implemented on smart cards. We call the analysis the *address-bit DPA* in this paper. The analysis was originally investigated by Messerges, Dabbish and Sloan, however it was thought to be of no effect if the intermediate data are randomized. We extend the analysis and show how the extended analysis works against scalar exponentiations even if the implementation is resistant against the data-based DPA. We show experimental results of our analysis of cryptographic schemes OK-ECDH and OK-ECDSA, which are candidates of the CRYPTREC project in Japan, and evidence of their weakness.

Keywords. Differential power analysis (DPA), address-bit DPA, elliptic curve cryptosystems (ECC), scalar exponentiation, OK-ECDH, OK-ECDSA

1 Introduction

The key agreement scheme OK-ECDH [17] and the digital signature scheme OK-ECDSA [18], developed by HITACHI, are candidates of the CRYPTREC project, in which a list of cryptographic schemes suitable for e-government in Japan are being made [6]. OK-ECDH (OK-ECDSA) is based on the discrete logarithm problem over the Montgomery form elliptic curves [15] but the scheme is very similar to the standard ECDH (ECDSA) [8,23]. As we will discuss the scalar exponentiation of OK-ECDH and OK-ECDSA, we refer them as OK-Schemes (OKS) in the following.

The self-evaluation reports of OKS have claimed resistance against side channel attacks [17,18]. The side channel attacks, firstly proposed by Kocher et al. [11,12], are attacks in which an attacker observes side channel information such

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 129–143, 2003.

as computing time and power consumption from a cryptographic device, and attempts to reveal secret information (a secret key) hidden in the device. At the moment, the Simple Power Analysis (SPA) and the Differential Power Analysis (DPA) are typical examples of the side channel attacks. Implementers should pay the most attention to the attacks and take measures against each of them in order to avoid these attacks.

The power consumption changes in accordance with the Hamming weights of data passed in the device. The DPA is classified into two types; the data-bit DPA, in which an attacker reveals dependence of values of data on the difference of power consumption [13,4], and the address-bit DPA, in which an attacker reveals those of addresses of registers [14]. The address-bit DPA is based on the fact that if we load data from various addresses, the power consumption of the device changes in accordance with a difference of Hamming weights of addresses. Thus the address-bit DPA is as noteworthy as the data-bit DPA [14]. It is, however, thought to be of no effect if the intermediate data in the device are randomized.

This paper extends the analysis and shows how the extended address-bit DPA works against scalar exponentiations of elliptic curve based cryptosystems (ECC), even if the algorithm is resistant against the data-bit DPA. We assumed two conditions. One is that the algorithm in the device is known. The other is that the number of registers used in the algorithm is small. These conditions make our address-bit DPA easier. This paper also shows experimental results of our address-bit DPA of OKS. As the (recommended) algorithms for OKS is public and the number of registers is at most 3, the above conditions are satisfied. We show evidence of the weakness of OKS against our address-bit DPA.

The paper is organized as follows: we introduce previous results of DPA in Section 2. The validity of the address-bit DPA is shown in Section 3. Then, in Section 4, we explain how to apply our DPA to OKS together with experimental results.

2 Preliminaries

2.1 Elliptic Curve

In this paper, we discuss elliptic curves over $K = GF(p)$, a finite field with p-elements for a prime p. Let E be an elliptic curve over K and $E(K)$ be a set of points on the curve including the special point $\mathcal{O}$ (the point of infinity). The set $E(K)$ has an additive group structure. A concrete algorithm for computing an addition for given points is found in a textbook ([3], for example). For two points P_1, P_2 on $E(K)$, we denote an operation $P_1 + P_2$ as ECADD (where $P_1 \neq P_2$), and an operation $2 * P_1$ as ECDBL. For a given elliptic curve $E(K)$, a point P on $E(K)$, and an integer d, computing $d * P = P + P + ... + P$ (d times) is called a scalar exponentiation and P, d are called the base point and the exponent, respectively. A scalar exponentiation is computed by a combination of ECADD and ECDBL. An addition chain determines such combination. Let d

be an n-bit integer and consider a binary expression $d = d[n-1] * 2^{n-1} + d[n-2] * 2^{n-2} + ... + d[1] * 2^1 + d[0]$ ($d[n-1] = 1$). A standard binary chain is given in the following algorithm (Algorithm 1).

Algorithm 1. Binary chain

```
INPUT: d, P
OUTPUT: d * P
 1: Q[0] = P
 2: for i=n-2 down to 0 {
 3:   Q[0] = ECDBL(Q[0])
 4:   if d[i]==1  Q[0] = ECADD(Q[0],P)
 5: }
 6: return Q[0]
```

2.2 Side Channel Attack

Side Channel Attacks (SCA) are proposed by Kocher et al. [11,12], in which an attacker observes side channel information such as computing time and power consumption from a cryptographic device (smart cards), and attempts to reveal secret information (a secret key) hidden in the device without breaking it physically. The attacks are valid if there is dependence between the secret information and the power consumption. At the moment, the Simple Power Analysis (SPA) and the Differential Power Analysis (DPA) are typical examples of side channel attacks. Implementers should pay the most attention to the attacks and take measures against each of them in order to avoid these attacks.

The SPA uses observed side channel information. In Algorithm 1, ECADD is computed only when $d[i] = 1$. An attacker can guess the value of $d[i]$ by checking a pattern of the power consumption and able to reveal the secret key. Coron proposed a countermeasure so called the add-and-double-always method in which ECDBL and ECADD are always computed for all $d[i]$ [4]. Then operations make a fixed pattern of side channel information and the attacker cannot obtain any information by SPA.

The power consumption changes in accordance with a difference of Hamming weights of data. The DPA analyzes these differences from side channel information obtained through a lot of observations. As the Coron's DPA [4] is involved in the Messerges-Dabbish-Sloan's DPA [13], we only introduce the latter one. Messerges-Dabbish-Sloan classified their DPA into three cases depending on the assumption of the attacker. The following explanations are devoted to ECC, but similar attacks are applicable to the famous RSA cryptosystem.

Single-Exponent, Multiple-Data (SEMD): In SEMD, we assume that the attacker knows one exponent d_k, able to measure the traces of power consumption (power traces) for any inputs, but does not know the algorithm. The attacker, who is going to reveal a secret key d_u, first measures the power traces of

the device with d_k on input various random values and obtains an average power trace. Next, he inputs the same values into the device with d_u and obtains an averaged power trace. Then a difference of these two power traces determines the order of ECADD and ECDBL computed in the device, because the trace is 0 only when two exponents operate the same operations at the same time.

Multiple-Exponent, Single-Data (MESD): In MESD, we assume that an attacker can measure the power traces with any exponents, but does not know the algorithm. The attacker measures a power trace with d_u on input a certain value. Suppose the attacker knows $d_u[n-1], \cdots, d_u[i+1]$ of d_u. Then, he/she guesses $d_u[i]$, measures a trace with it on input the same value and obtains a difference of traces. If the guess is correct the difference corresponding $d_u[i]$ is 0 and the attacker convinces the correctness of his/her guess. Thus the secret key d_u is revealed by MESD by repeating the same procedures.

Zero-Exponent, Multiple-Data (ZEMD): In ZEMD, we assume that the attacker knows the algorithm of a scalar exponentiation, knows modules, and able to simulate computations in the device. The attacker measures power traces on input various random values and obtains power traces for each input. Next, the attacker guesses $d_u[i]$ of d_u used in the first module and obtains resulted data of the module for each input by simulations. Then he/she divides the results into two parts depending on their Hamming weights, computes average power traces for each part and obtains a difference of traces. Then there appear spikes in the difference if his guess is correct, and no spikes otherwise. Thus the secret key d_u is revealed by ZEMD by repeating the same procedures.

In the add-and-double-always countermeasure [4], ECDBL and ECADD are computed repeatedly and pattern of the power trace is fixed for any inputs. So SEMD and MESD cannot reveal the secret key. However ZEMD is valid for this countermeasure. One approach to resist ZEMD is to make the simulation impossible by randomizing intermediate values. Coron's Randomized Projective Coordinate (RPC) [4] and Joye-Tymen's randomized isomorphic curve [10] are good examples.

2.3 OK-ECDH and OK-ECDSA

The key agreement scheme OK-ECDH [17] and the digital signature scheme OK-ECDSA [18], developed by HITACHI, are candidates of the CRYPTREC project, in which cryptographic schemes suitable for e-government in Japan are being evaluated [6]. As we will discuss scalar exponentiations of OK-ECDH and OK-ECDSA, we refer them as OK-Schemes (OKS) in this paper. OKS is based on the elliptic curve discrete logarithm problem and the schemes are very similar to standard ECDH and ECDSA [8,23]. An outstanding difference is that all operations of OKS are performed on the Montgomery form elliptic curves [15], which is defined by $By^2 = x^3 + Ax^2 + x \quad A, B \in K,\ B(A^2 - 4) \neq 0$. A special addition formula, which does not use the y-coordinates of points, offers a fast

scalar exponentiation on this curve. OKS uses these techniques as well and hence fast cryptographic operations are possible [20,22]. We show an outline of OK-ECDH in Algorithm 2, for example, where P is a base point and a key pair (d_A, Q_A) satisfies $Q_A = d_A * P$. Roughly speaking, OK-ECDH is obtained by operating ECDH on Montgomery form elliptic curves. See specifications [17,18] for detailed descriptions.

Algorithm 2. Outline of OK-ECDH

INPUT: Q_V, a (one-time) public key of an entity V
OUTPUT: z, a shared key

1. Generate a one-time key pair (d_U, Q_U) (satisfying $Q_U = d_U * P$).
2. Send Q_U to the entity V
3. Compute a point $Q = d_U * Q_V$ on Montgomery-form elliptic curve.
4. (Option) Compute a point $R = cQ$ and then if $R = \mathcal{O}$ then terminate, where c is the cofactor of the Montgomery-form elliptic curve.
5. If $Q = \mathcal{O}$ then terminate.
6. Transform the x-coordinate x_Q of the point Q to an octet string z.
7. Output shared key z

As OKS uses the special addition formula, it uses an alternative addition chain (Montgomery ladder, Algorithm 3) rather than the standard chain (Algorithm 1). The Montgomery ladder computes ECDBL and ECADD repeatedly, hence the chain is resistant against SPA, SEMD, MESD [19]. Against ZEMD, OKS uses Randomized Projective Coordinates (RPC) in order to resist it [21, 20]. Moreover, developers claimed in their self-evaluation reports that an cryptographic scheme is SCA-resistant, if the secret information and the order of operations are independent and intermediate values are randomized [17,18].

Algorithm 3. Montgomery ladder

INPUT: d, P
OUTPUT: $d * P$

```
1: Q[0] = P, Q[1] = ECDBL(P)
2: for i=n-2 to downto 0 {
3:   Q[2] = ECDBL(Q[d[i]])
4:   Q[1] = ECADD(Q[0],Q[1])
5:   Q[0] = Q[2-d[i]], Q[1] = Q[1+d[i]]
6: }
7: return Q[0]
```

3 Address-Bit DPA

Messerges-Dabbish-Sloan's DPA [13] described in Section 2.2 tries to find dependence of values of data on a difference of power traces. On the other hand, they also proposed alternative DPA which tries to find those of addresses of registers [14]. The latter analysis is based on the fact that if we load same data from different addresses of registers, the power consumption changes in accordance with a difference of Hamming weights of addresses. They show experimental results of basic characteristics of the address-bit DPA and introduced an idea of the attack for DES implementation [14]. However, they did not show a concrete result of the DPA attack experiment, so that the address-bit DPA does not attract much attention and it is thought to be of no effect if the intermediate data are randomized.

In this section, we extend a concept of the address-bit DPA and show how the extended analysis works for elliptic curve based cryptosystems even if the algorithm is resistant against the data-bit DPA.

3.1 Outline

The original address-bit DPA watches a difference of addresses via loading same data from different addresses. The power consumption changes when different data are loaded from different addresses of registers. If the influence of data in a difference of power traces is erased, a secret key can be revealed by watching a difference of addresses. Indeed, the influence of data is erased by averaging the power traces and the averaged power trace only depends on a difference of Hamming weights of register addresses. This is a basic idea of our address-bit DPA. The attack is successful if there is a close dependence between a secret key and addresses of accessed registers. In general, this approach may be hard because a lot of registers are used in the implementation. But in our target of ECC, the number of registers is small and the situation makes our analysis easier than general cases.

3.2 Experimental Results

In order to show the validity of our address-bit DPA, we show experimental results in the following. We have two registers $Q[0], Q[1]$ and given unknown value d (0 or 1). We load 8-bit data $L = 500$ times from $Q[d]$ and try to guess d. Here the data in $Q[d]$ are changed into random values after every load. Such procedure is described as `A = Q[d]` in the algorithmic level. In a low-power device such as smart cards, this procedure is divided into two stages; (1) determining the address of $Q[d]$, and (2) loading data with the address.

Our strategy is as follows. First we observe L power traces for $d = 0$ (a). Next, we observe L power traces for d_b (b) and d_c (c) ($d_b, d_c = 0, 1,\ d_b \neq d_c$). Let $S_{a,i}, S_{b,i}, S_{c,i}$ be the i-th power traces and S_a, S_b, S_c be their averages. Then

the differences of power trace (differential power traces) D_{ab}, D_{ac} are defined by the following:

$$D_{ab} = \frac{1}{L}\sum_{i=1}^{L} S_{a,i} - \frac{1}{L}\sum_{i=1}^{L} S_{b,i} = S_a - S_b,$$
$$D_{ac} = \frac{1}{L}\sum_{i=1}^{L} S_{a,i} - \frac{1}{L}\sum_{i=1}^{L} S_{c,i} = S_a - S_c.$$

Then there should appear spikes in D_{aj} if $d_a \neq d_j$, no spikes in D_{aj} if $d_a = d_j$, where $j \in \{b, c\}$. Our differential power traces D_{ab}, D_{ac} are in Figure 1. Two spikes are found in D_{ac} (arrowed)[1]. From these figures, we are convinced that $d_b = 0$ and $d_c = 1$. In fact, these guesses are correct.

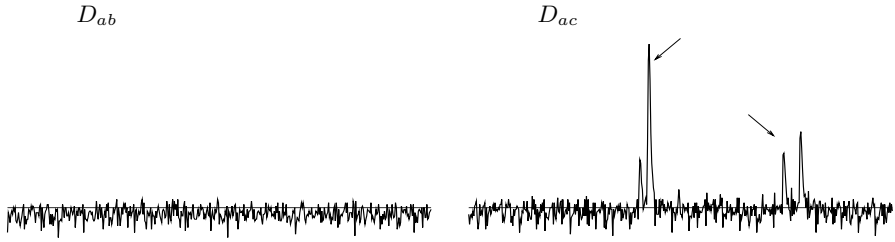

Fig. 1. Differential power traces D_{ab} and D_{ac}

In the experiment, an address $\&Q[d]$ is determined by a secret data d, and the value of the address has an influence on the power trace. Thus we can guess the value of d by using the close dependence between addresses and data. We conclude that the secret key d can be revealed by the address-bit DPA, even if the data are randomized.

Note 1. To be exact, the target of Messerges-Dabbish-Sloan's address-bit DPA [14] is only (2). The spike (1) is detected by the data-bit DPA as well because it occurs through data-loading of addresses of registers.

4 Proposed Address-Bit DPA

This section discusses the address-bit DPA of OK-ECDH and OK-ECDSA based on SEMD and ZEMD. In order to resist the data-bit DPA, OKS uses the Randomized Point Coordinates (RPC) countermeasure. Here intermediate data in the device are varied (randomized) even if the input is unchanged. So MESD

[1] The first spike in D_{ac} corresponds to the addressing (1), and the latter spike corresponds to the loading (2).

cannot be applied to OKS because a 'Single Data' is not available. Moreover there is no need to mention 'MD' for distinguishing SEMD and ZEMD. We just call SE or ZE analysis in the following. Note that in the SE analysis, we assume the algorithm is known. This is stronger condition than those of original SEMD, but there is no problem for our case because the algorithm of OKS is public.

We analyze OKS implemented by either following Implementation 1 or Implementation 2. The algorithms are described in C-like language, where d is an n-bit secret key, $d[i]$ is the i-th bit of d and $Q[0], Q[1], Q[2]$ are registers for intermediate data.

Implementation 1

```
Q[0]=P, Q[1]=ECDBL(P)
for i=n-2 to downto 0 {
  Q[2]=ECDBL(Q[d[i]]) (*11)
  Q[1]=ECADD(Q[0],Q[1])
  Q[0]=Q[2-d[i]], Q[1]=Q[1+d[i]] (*12)
}
return Q[0]
```

Implementation 2

```
Q[0]=P, Q[1]=ECDBL(P)
for i=n-2 to downto 0 {
  Q[2]=ECDBL(Q[d[i]]) (*21)
  Q[1]=ECADD(Q[0],Q[1])
  t=&Q[0], &Q[0]=&Q[2-d[i]], &Q[2-d[i]]=t (*22)
  t=&Q[1], &Q[1]=&Q[1+d[i]], &Q[1+d[i]]=t (*23)
}
return Q[0]
```

Implementation 1 uses $d[i]$ in (*11) and (*12) not to operate different procedures but to load data from different registers. Note that recommended scalar exponentiation algorithm in [17,18] and the algorithm in [21] is equivalent to Implementation 1.

Implementation 2 uses $d[i]$ in (*21), (*22) and (*23). While (*12) in Implementation 1 'copies' the values into $Q[0], Q[1]$, (*22) and (*23) in Implementation 2 'swap' the addresses of registers of $Q[0]$ with $Q[2 - d[i]]$ and $Q[1]$ with $Q[1 + d[i]]$. Note that, in Implementation 1, addresses of registers $Q[0], Q[1], Q[2]$ are unchanged. And in Implementation 2, addresses of registers $Q[0], Q[1], Q[2]$ are changed depending on the value $d[i]$, although operations are same.

Our address-bit DPA uses the differences of addresses of intermediate registers $Q[0], Q[1], Q[2]$. In a scalar exponentiation in OKS (Algorithm 3), ECDBL and ECADD are computed repeatedly and a pattern is fixed independent from

the exponentiation. But inputs are randomly varied because of the RPC. As we showed in Section 3, the influence of randomized data on power consumption can be erased by averaging traces. Then the difference of averaged power traces comes from (*) in Implementation 1 or 2 and a secret key can be revealed.

4.1 SE Attack

In the SE attack, we assume that an attacker knows one exponent d_k, able to measure the power traces for any inputs, as in SEMD. Moreover we assume that Implementation 1 is used. The attacker measures the power traces with d_k on input various values and obtains an average power trace for d_k. We denote the power trace corresponding to the i-th bit $d_k[i]$ of the j-th measurement as $S_{k,j}[i]$ and an averaged power trace corresponding to $d_k[i]$ as $S_k[i]$. Next he/she measures power traces with unknown exponent d_u on input same values and obtains an average power trace for d_u. We denote the power trace corresponding to the i-th bit $d_u[i]$ of the j-th measurement as $S_{u,j}[i]$ and an averaged power trace corresponding to $d_u[i]$ as $S_u[i]$. Then the differential power trace $D[i]$ is given by the following:

$$D[i] = \frac{1}{L}\sum_{j=1}^{L} S_{k,j}[i] - \frac{1}{L}\sum_{j=1}^{L} S_{u,j}[i] = S_k[i] - S_u[i],$$

where L is the number of measurements. On the other hand, in Implementation 1 of OKS, ECDBL and ECADD are computed in a same manner independent from d_k, d_u. So we expect that $S_{k,j}[i]$ and $S_{u,j}[i]$ are signals generated by completely the same operations. Indeed as $S_k[i], S_u[i]$ are averaged power traces for various data, the influence of data is erased as in Section 3. So we have

$$D[i] \simeq \begin{cases} 0 & \text{if } d_k[i] = d_u[i] \\ \text{nonzero} & \text{if } d_k[i] \neq d_u[i] \end{cases}.$$

Thus a difference of averaged traces determines the values of $d_u[i]$, because the difference is 0 if $d_k[i] = d_u[i]$ and the difference is nonzero if $d_k[i] \neq d_u[i]$. We show an experimental result of the SE attack against OKS in the following. The parameters we used in the experiment are as follows:

$$\begin{array}{lll} p = \texttt{0x200011}, & A = \texttt{0x14c82a}, & B = \texttt{0x11133f}, \\ h = \texttt{0x8019d}, & x = \texttt{0x1b144d}, & y = \texttt{0x1aa97d}, \end{array}$$

where the Montgomery form elliptic curve is defined by $By^2 = x^3 + Ax^2 + x$ over $GF(p)$, the order of the curve is $4h^2$ and a base point is $P = (x, y)$. We measured scalar exponentiations (implemented by Implementation 1) $L = 500$ times with $d_k = 1111\cdots$. We measured power traces corresponding to the most significant 4 bits for each exponent. The differential power trace is shown in Figure 2. There is no spike for $i = n-1, n-3$ and there are spikes for $i = n-2, n-4$. From the figure, we can determine the secret bits as $d_u = 1010\cdots$ and indeed our guess was correct.

[2] The order of the Montgomery form elliptic curve is divisible by 4.

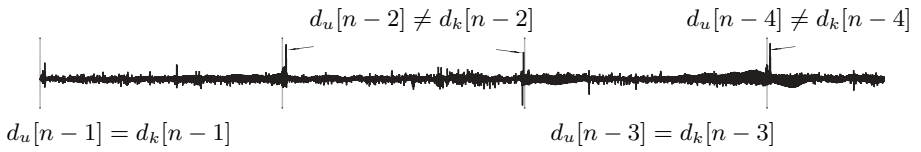

Fig. 2. Differential power traces D_{ab} and D_{ac}

Note 2. We have two spikes for $i = n-2$ in Figure 2, the one corresponds to (*11) and the other to (*12). The second spike for $i = n-4$ is omitted in the figure.

Note 3. In the experiment we used 22-bit parameters, while the bit length of the specification of OKS is 162-bit. This is only because of a simplification, not a theoretical reason. In the above attack, we measured the power traces of successive 4-bit exponents in order to draw comprehensible figures. If we know the timing when spikes appear in the trace, we only need to store data according to the timings for performing the effective analysis. By using this technique, our analysis will be successful with larger parameters.

Remark that, if the parameter is larger than 22-bit, there is no difference except that the interval between the first and the second spikes in Figure 2 becomes longer.

4.2 ZE Attack

In the ZE attack, we assume that an attacker knows the algorithm of a scalar exponentiation (namely, Implementation 1 or 2), and the module, and able to simulate the computation in the device, and able to measure the power traces for any inputs, as in ZEMD. Moreover we assume that Implementation 1 is used. The attack depends on the implementation.

Implementation 1: An attacker measures the power traces on input various random values with unknown secret key d_u and obtains an average trace. Next, the attacker decompose the average trace into modules for each bit $d_u[i]$, where a module is consists of an ECDBL and an ECADD. We note that, as a scalar exponentiation in OKS is processed by computing ECDBL and ECADD repeatedly, this decomposition is done quite easily. Then the attacker computes the differences of average traces. We denote the power trace corresponding to the i-th bit $d_u[i]$ of the j-th measurement as $S_{u,j}[i]$ and an averaged power trace corresponding to $d_u[i]$ as $S_u[i]$. Then the differential power trace $D[a,b]$ of $d_u[a]$ and $d_u[b]$ is given by the following:

$$D[a,b] = \frac{1}{L}\sum_{j=1}^{L} S_{u,j}[a] - \frac{1}{L}\sum_{j=1}^{L} S_{u,j}[b] = S_u[a] - S_u[b],$$

where L is the number of the measurements. On the other hand, in Implementation 1 of OKS, ECDBL and ECADD are computed in a same manner independent from d_u, we expect that $S_u[a]$ and $S_u[b]$ are traces generated by completely the same operations. Indeed, as $S_u[a]$ and $S_u[b]$ are averaged power traces of random data, the influence of data is erased here. Then we have

$$D[i] \simeq \begin{cases} 0 & \text{if } d_u[a] = d_u[b] \\ \text{nonzero} & \text{if } d_u[a] \neq d_u[b] \end{cases}.$$

A difference of traces determines the values of $d_u[i]$, because the difference is 0 if $d_u[a] = d_u[b]$ and the difference is nonzero if $d_u[a] \neq d_u[b]$. Thus a secret key d_u is revealed by the ZE against Implementation 1.

We show an experimental result of the ZE attack against Implementation 1 in the following. We used the same parameters as in the previous experiment. We computed scalar exponentiations (implemented by Implementation 1) $L = 500$ times with an unknown exponent d_u and measured power traces $S_{u,j}[i]$ corresponding to the most significant 4 bits for each exponent. Then we calculated an average trace $S_u[i]$, decomposed it into $S_u[0], \cdots, S_u[3]$[3], and obtained DPA bias signals which is shown in Figure 3. There are no spike in $D[0, 2]$, while there are two spikes in $D[0, 1]$ and $D[0, 3]$ in the figure. From these results, we can determine the secret bits are $d_u = 1010\cdots$, because the most significant bit of d_u is 1. Indeed our guess was correct.

Note 4. We have two spikes in $D[0, 1], D[0, 3]$, the one is corresponding to (*11) and the other is to (*12) as in the SE attack.

Implementation 2: As in the attack against Implementation 1, the attacker decomposes the average trace into modules for each bit $d_u[i]$ of a secret key d_u, where a module is consists of an ECDBL and an ECADD. Here in Implementation 2, the addresses of registers $Q[0], Q[1], Q[2]$ are varied depending on the exponent. That is, the differences of average power trace of modules do not have any sense. So we analyze a transition of addresses. Let (a,b,c) denote addresses of $Q[0], Q[1], Q[2]$. Then the transition is as in Figure 4.

Suppose the current addresses are (a,b,c) which corresponds to $d_u[i]$. First, we consider the recovery of addresses (Case 1). From the transition, the addresses (a,b,c) comes back to (a,b,c), if the exponent is one of

$$R = \{00, 111, 0101, 1010, 011011, 101101, 110110\}.$$

So if the exponent is $r \in R$, the DPA bias signal $D[i, i + |r|] = 0$ in all of (*21),(*22) and (*23), where $|r|$ is a bit length of r, and one of them is nonzero otherwise. Note that if an exponent is 000, the address of $Q[0]$ transits a $\rightarrow$ c $\rightarrow$ a and we have (c, b, a) after the procedure.

Next, let us consider the transition of addresses in (*21) (Case 2). Here the address of $Q[0]$ is used if $d_u[i] = 0$, and the address of $Q[1]$ is used if $d_u[i] = 1$.

[3] It is described $0, \cdots, 3$ for easiness to explain though these suffixes should be precisely described $n - 1, \cdots, n - 4$.

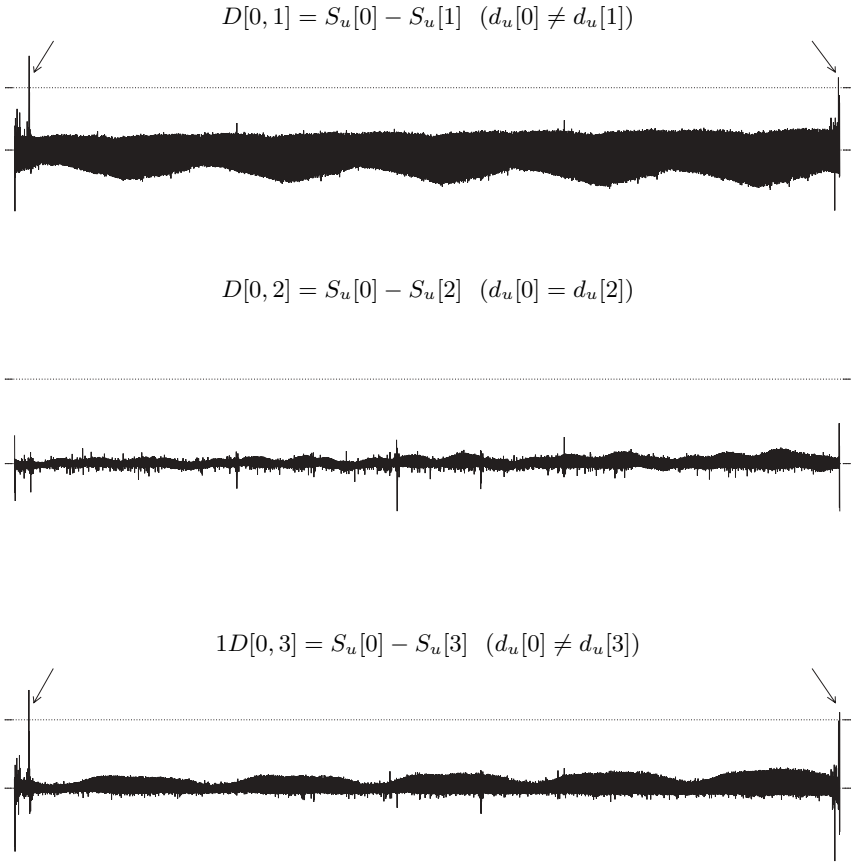

Fig. 3. Differential power traces by the ZE attack against Implementation 1

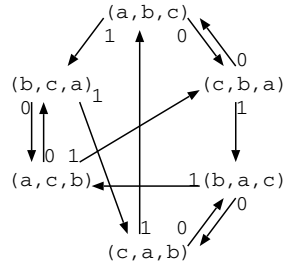

Fig. 4. Address transition of $Q[0], Q[1], Q[2]$ in Implementation 2

Suppose successive bits of d_u are same, then the same addresses are used in (*21) and we have no spikes in the DPA bias signal. Otherwise, namely successive bits are different, we have spikes in the bias signal. For example, suppose an exponent is 1000. Then we have no spikes in the differences $D[i, i+1], D[i, i+3]$, $D[i+1, i+3]$, and have spikes in $D[i, i+2], D[i+1, i+2], D[i+2, i+3]$. Note that if an exponent is 1000, the address of $Q[0]$ transits b → b → a → b and we have (a, c, b) after the procedure.

Let us reveal an secret exponent d_u from $D[i, j]$ by using the above observations. Suppose the initial addresses of $Q[0], Q[1], Q[2]$ are unknown, and we can distinguish $D[i, j]$ being 0 or nonzero in (*21),(*22),(*23). By Case 2, we obtain appeared types of addresses used in (*21), which is described as x,y,z, according to the appeared order. Here we obtain all patterns of x,y,z for 4-bit exponents, which are shown in Table 1. The patterns are sorted lexically.

Table 1. Relative patterns of addresses in (*21)

Exponent	Pattern	Exponent	Pattern	Exponent	Pattern	exponent	pattern
1, 0, 0, 0	x, x, y, x	0, 1, 1, 0	x, y, x, x	0, 1, 0, 1	x, y, y, x	1, 1, 1, 1	x, y, z, x
1, 0, 1, 1	x, x, y, x	0, 0, 0, 0	x, y, x, y	1, 1, 0, 0	x, y, y, x	0, 0, 1, 1	x, y, z, y
1, 0, 1, 0	x, x, y, y	0, 0, 0, 1	x, y, x, z	0, 1, 0, 0	x, y, y, z	0, 0, 1, 0	x, y, z, z
1, 0, 0, 1	x, x, y, z	0, 1, 1, 1	x, y, x, z	1, 1, 0, 1	x, y, y, z	1, 1, 1, 0	x, y, z, z

We have same patterns for (1000,1011), (0111,0101), (1100,0100), (0010,1110), and we cannot distinguish them. But (1000,1011), (0001,0111) can be distinguished by Case 1, because of the exponent includes 000. Thus for 4-bit exponents, 10 patterns out of 16 patterns are uniquely determined. If we make a similar table for 6-bit exponents, they are distinguished uniquely.

5 Concluding Remarks

This paper proposed the address-bit DPA against the elliptic curve based cryptosystems, together with experimental results of OK-ECDH and OK-ECDSA. We showed evidence of the weakness against our address-bit DPA. However, as the attack is targeted to the implementation, it seems to have no relation to the underlying mathematical problem, i.e. the elliptic curve discrete logarithm problem.

The address-bit DPA is based on the dependence between an secret key and address of registers used in the algorithm. That is, the countermeasures by randomizing data, such as [4,10], are not effective on our analysis. The result suggests that an exponent should be randomized as well. The exponent blinding [4,13], in which a scalar d is replaced by $d' = d + r\phi$ (r is a random number, ϕ is the order of the curve), the exponent splitting [5], in which d is split into r and $d - r$, and the overlapped window method [24] are examples. OKS will be secure

against our analysis if such countermeasures has been applied. In any case, the recommended algorithm in the current specification of OKS is vulnerable against our analysis.

Other countermeasure against the address-bit DPA is to use registers with same Hamming weights of addresses. This approach will be successful if power comsumption of the device follows the characteristic of the Hamming weight model. However, in the Linear model and the Quadratic model [1], even if the Hamming weights are same, digit positions are distinguishable and the countermeasure has no effect against our analysis.

Our analysis is expected to be applicable to other scalar exponentiation algorithms in [2,4,7,9], for example. A detailed discussion and experiments will be our future work.

Acknowledgment. The authors would like to thank our colleagues, especially Naoya Torii, for their helpful suggestions. We also thank anonymous referees of this paper for their comments.

References

1. M. Akkar, R. Bevan, P. Dischamp, and D. Moyart, "Power Analysis, What is Now Possible...", Asiacrypt 2000, LNCS 1976, pp. 489–502, Springer-Verlag, 2000.
2. E. Brier, and M. Joye, "Weierstraß Elliptic Curves and Side-Channel Attacks", PKC 2002, LNCS 2274, pp. 335–345, Springer-Verlag, 2002.
3. I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
4. J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystem", CHES'99, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.
5. C. Clavier, and M. Joye, "Universal exponentiation algorithm – A first step towards provable SPA-resistance –", CHES2001, LNCS 2162, pp. 300–308, Springer-Verlag, 2001.
6. Cryptography Research & Evaluation Committees (CRYPTREC), Japan. `http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html`
7. W.Fischer, C.Giraud, E.Knudsen, and J.P.Seifert, "Parallel Scalar Multiplication on General Elliptic Curves over F_p Hedged Against Non-Differential Side-Channel Attacks", Cryptology ePrint Archiver, 2002/007, IACR. Available from `http://www.iacr.org/`
8. IEEE P1363, Standard Specifications for Public-Key Cryptography, 2000.
9. T. Izu, and T. Takagi, "A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks", PKC 2002, LNCS 2274, pp. 280–296, Springer-Verlag, 2002.
10. M. Joye, and C. Tymen, "Protections against differential analysis for elliptic curve cryptography", CHES2001, LNCS 2162, pp. 377–390, Springer-Verlag, 2001.
11. C. Kocher, "Timing attacks on Implementations of Diffie-Hellman, RSA, DSS, and other systems", Crypto'96, LNCS 1109, pp. 104–113, Springer-Verlag, 1996.
12. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis", Crypto'99, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.

13. T.S. Messerges, E.A. Dabbish, and R.H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards", CHES'99, LNCS1717, pp. 144–157, Springer-Verlag, 1999.
14. T.S. Messerges, E.A. Dabbish, and R.H. Sloan, "Investigations of Power Analysis Attacks on Smartcards," preprint, USENIX Workshop on Smartcard Technology, 1999.
15. P. Montgomery, "Speeding the Pollard and elliptic curve methods for factorizations", Mathematics of Computation, vol.48, pp. 243–264, 1987.
16. National Institute of Standards and Technology, Recommended Elliptic Curves for Federal Government Use, in the appendix of FIPS 186-2.
17. Key Agreement Scheme OK-ECDH, HITACHI, 2001. Documents are available from `http://www.sdl.hitachi.co.jp/crypto/ok-ecdh/index.html`
18. Digital Signature Scheme OK-ECDSA, HITACHI, 2001. Documents are available from `http://www.sdl.hitachi.co.jp/crypto/ok-ecdsa/index.html`
19. K. Okeya, H. Kurumatani, and K. Sakurai, "Elliptic curves with the Montgomery form and their cryptographic applications", PKC 2000, LNCS 1751, pp. 446–465, Springer-Verlag, 2000.
20. K. Okeya, K. Miyazaki, and K. Sakurai, "A fast scalar multiplication method with randomized projective coordinates on a Montgomery-form elliptic curve secure against side channel attacks", ICISC 2001, LNCS 2288, pp. 428-439, Springer-Verlag, 2001.
21. K. Okeya, and K. Sakurai, "Power analysis breaks elliptic curve cryptosystem even secure against the timing attack", Indocrypt 2000, LNCS 1977, pp. 178–190, Springer-Verlag, 2000.
22. K. Okeya, and K. Sakurai, "Efficient elliptic curve cryptosystem from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve", CHES 2001, LNCS 2162, pp. 126–141, Springer-Verlag, 2001.
23. Standards for Efficient Cryptography Group (SECG), Specification of Standards for Efficient Cryptography.
24. J. Yajima, K. Itoh, M. Takenaka, and N.Torii, "DPA countermeasure by improving the window method", to appear in the proceeding of CHES 2002.