Vol. 45 No. 8 IPSJ Journal Aug. 2004

Regular Paper

A Practical Countermeasure against Address-bit Differential Power Analysis

Kouichi Itoh,† Tetsuya Izu† and Masahiko Takenaka†

The differential power analysis (DPA) is a realistic attack in which an adversary reveals the secret key hidden in a smart card. In 1999, Messerges, et al. proposed the address-bit DPA, a variant of DPA, against DES. Then, in 2002, Itoh, et al. extended the attack to the exponentiation-based public key cryptosystems. In this paper, we propose a practical countermeasure against the address-bit DPA applicable to the exponentiation part in RSA or ECC with and without pre-computed tables. Proposed countermeasure has almost no overhead for the protection, namely the processing speed is no slower than that without the countermeasure. Since the countermeasure resists only the address-bit DPA, other countermeasures should be combined in order to resist all DPAs. We list major DPA-countermeasures and discuss the performance including the processing speed and the security level when combined. As a result of the comparison, our proposed countermeasure will be included in practical solutions.

1. Introduction

Smart cards are becoming a new infrastructure of the coming IT society because of their plenty attractive applications such as identification cards, telephone cards and electronic tickets. However, the power analysis is a realistic threat [16),17)], in which an adversary reveals the secret key hidden in a smart card by using the power consumption of the device. The attack will be successful if there is a correlation between the power consumption and the secret key. The simple power analysis (SPA) and the differential power analysis (DPA) are typical examples. Implementers of cryptographic algorithms should take countermeasures against these attacks.

In 1999, Messerges, et al. proposed a new variant of DPA against common key cryptosystems, the *address-bit DPA* (ADPA) (from now on, we call the previous DPAs as the *data-bit DPA* (DDPA)), which analyzes a correlation between the secret key and addresses of registers [22)]. Then, in 2002, Itoh, et al. extended the attack to the exponentiation-based public key cryptosystems including the famous RSA and Elliptic Curve Cryptosystems (ECC) [8)]. Their result implies that implementers should consider correlations of the secret key to not only data of registers but also addresses of registers. Itoh, et al. also gave several countermeasures against the attack, but those countermeasures require at least twice processing time than without them.

In this paper, we propose a practical countermeasure against ADPA applicable to the exponentiation-based public-key cryptosystems by randomizing addresses of registers. Proposed countermeasure does not change the scalar to be exponentiated; an overhead is very small and the processing speed is as fast as before, namely, a scheme resistant against the data-bit DPA can be easily enhanced to that against the address-bit DPA with almost no penalty. The conversion can be applied to not only binary methods but to some window-based methods. We also discuss the security of our countermeasure from both theoretical and experimental approaches.

Since our countermeasure is dedicated to resist ADPA, other DPA-countermeasures should be combined in order to resist all DPAs. However, finding a good combination, which satisfies a certain security level and requires a compromisable processing speed, is a hard task for implementers. In this paper, we list some countermeasures and discuss their performance including the processing time and the security level. As a criteria for the security, we adopted the Attenuation Ration (AR) proposed by Itoh, et al. [13)] Consequently, our proposed method (combined with other DPA-countermeasures)

† FUJITSU LABORATORIES LTD.

A part of the paper was published in the proceedings of the 5th International Workshop for Cryptographic Hardware and Embedded Systems (CHES 2003) [10)].

provides a good practical solution for resisting the power analysis.

In the followings, we mainly deal with scalar exponentiations in Elliptic Curve Cryptosystems (ECC), however, most algorithms including our countermeasure can be applied to other exponentiation-based cryptosystems. The rest of this paper is organized as follows: In Section 2, we give a brief overview of the power analysis and countermeasures. Then Section 3 describes our proposed countermeasure with experimental results. Combinations of countermeasures are discussed in Section 4.

2. Preliminaries

In this section, we give a brief introduction of Elliptic Curve Cryptosystems (ECC) and the power analysis against them. However, most attacks can be applied to other exponentiation-based cryptosystems.

2.1 Elliptic Curve Cryptosystems

Elliptic Curve Cryptosystems (ECC) are among the standard technology in the area of cryptography [7),25),33)], because they only require smaller keys compared to existing other cryptosystems. This feature of ECC is quite suitable for implementing on smart cards.

Let K be a finite field. An elliptic curve over K can be represented by the Weierstrass equation

$$E(K) = \{(x, y) \in K \times K \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}, \qquad (1)$$

for $a_i \in K$. The special point $\mathcal{O}$ is called the point of infinity. An elliptic curve $E(K)$ has an additive group structure, in which the point of infinity acts a neutral element. We call $P_1 + P_2$ ($P_1 \neq P_2$) the elliptic curve addition (ECADD) and $P_1 + P_2$ ($P_1 = P_2$), that is $2P_1$, the elliptic curve doubling (ECDBL). Let d be an integer and P be a point on the elliptic curve $E(K)$. A *scalar exponentiation* is to compute the point $dP = P + \cdots + P$ ($d - 1$ additions). A dominant computation of all encryption/decryption and signature generation/verification algorithms of ECC is the computation of dP for a secret integer d and a public base point P.

Let $d = d_{n-1}2^{n-1} + \cdots + d_12^1 + d_0$ be a binary expression of d with $d_{n-1} = 1$. Then the binary method (from the most significant bit (MSB)) for a scalar exponentiation is in Algorithm 1. Another algorithm from the least significant bit (LSB) is constructed similarly.

Alg. 1. Binary method

```
INPUT: d[], P
OUTPUT: dP
1:  T[0] = P
2:  for i=n-2 downto 0 {
3:    T[0] = ECDBL(T[0])
4:    if(d[i]==1){
5:      T[0] = ECADD(T[0],P)
6:    }
7:  }
8:  return T[0]
```

Alg. 2. Add-and-double-always method

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

2.2 Power Analysis

The power analysis is a powerful attack on smart cards with cryptographic algorithms. An adversary reveals the secret key hidden in a smart card by using power traces of the observed power consumption of the device. The following *simple power analysis* (SPA) [16)] and the differential power analysis (DPA) [17)] are typical examples.

2.2.1 Simple Power Analysis

The binary method (Algorithm 1) computes ECADD only when $d_i = 1$. Therefore an adversary easily detects this irregular procedure by observing the power consumption, and thus obtains the bit information d_i. This is a basic idea of the *simple power analysis* (SPA) [16)].

In order to resist SPA, Coron proposed the *add-and-double-always* method [4)] (Algorithm 2), in which both ECDBL and ECADD are computed in every bit (in step 3 and step 4). The countermeasure makes a pattern of the power trace fixed, so that the adversary cannot obtain the secret information by SPA. For efficiency, a variant of the add-and-double-always method, the Montgomery ladder [20)], is widely used [2),11),12),15),28),29)]. A sample algorithm of the Montgomery ladder is in Algorithm 3.

2.2.2 Differential Power Analysis

The *differential power analysis* (DPA) reveals the secret key by using statistical analysis of power consumptions [17),21)]. In DPA, an adversary makes an assumption on d_i ($d_i = 0$, for example) and simulates the procedure repeat-

Alg. 3. Montgomery ladder

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

Alg. 4. Add-and-double-always method with RPC

```
INPUT: d[], P
OUTPUT: dP
1:  T[0] = RPC(P)
2:  for i=n-2 downto 0 {
3:    T[0] = ECDBL(T[0])
4:    T[1] = ECADD(T[0],P)
5:    T[0] = T[d[i]]
6:  }
7:  return invRPC(T[0])
```

edly. Then he/she divides the results into two groups depending on the assumption, in order to make a bias of the hamming weights of the internal information between these groups. If the assumption is correct, then a difference of the power consumption of two groups (a *spike*) can be observed in the trace. Because of the principle of the analysis, DPA cannot attack the exponentiations with one-time scalars by nature (signature generation in ECDSA, for example). Coron proposed a methodology of DPA countermeasures by using random numbers [4] in order to make simulations impossible. Most of following countermeasure adopt this methodology.

Earlier DPA (the *data-bit DPA*) only considered a correlation of data of registers to the secret key [17),21)], while newer DPA (the *address-bit DPA*) also considers a correlation of addresses of registers [8),22)].

2.3 Data-bit DPA

The data-bit DPA (DDPA) analyzes a correlation of the secret key to data of registers. For example, after finishing step 5 in Algorithm 2, the data of `T[0]` is equals to that of `T[0]` if $d_i = 0$; otherwise it equals to that of `T[1]` if $d_i = 1$. The randomized projective coordinate (RPC) countermeasure works as follows [4]. Let $P = (X : Y : Z)$ be a base point represented by the projective coordinate. Then $(X : Y : Z)$ equals to $(rX : rY : rZ)$ for all $r \in K^*$ mathematically; but they are different as a bit sequence. RPC converts $(X : Y : Z)$ to $(rX : rY : rZ)$ for a randomly chosen number $r \in K^*$ so that the power consumption is randomized. An example of RPC for the add-and-double-always method is given in Algorithm 4, where a function `RPC` converts a point $(X : Y : Z)$ to a randomized point $(rX : rY : rZ)$ and a function `invRPC` denotes its inverse map.

Joye-Tymen proposed the randomized curve (RC) countermeasure [14], in which the base elliptic curve and the base point on it is transformed to isomorphic curves in a random manner. Two isomorphic curves are same mathematically; but different as a bit sequence. Thus the power consumptions are randomized by RC. A sample algorithm is easily obtained similarly to Algorithm 4 by changing functions `RPC,invRPC` to `RC,invRC`.

Instead of changing the expression of a base point, Coron also proposed the randomized exponent and the randomized base point countermeasures, in which the scalar is randomized. However, Okeya-Sakurai showed the bias of these countermeasures [29]. Messerges, et al. proposed the randomized start point countermeasure, in which a scalar is divided to two parts and computed by different methods [21]. Oswald-Aigner proposed another approach to randomize the scalar [27], but security problems have been pointed out [30),32),35)]. Walter proposed the MIST algorithm [34], which randomizes the intermediate data T (and U) by repeating $T = (d_i \bmod r_i)U + T$, $U = r_iU$ and $d_{i+1} = \lfloor d_i/r_i \rfloor$, where r_i are random numbers and initial values of T, U, d_i are $\mathcal{O}$, P, d respectively.

2.4 Address-bit DPA

The *address-bit DPA* (ADPA) was firstly proposed by Messerges, et al. for common key cryptosystems [22], which analyzes a correlation of the secret-key and addresses of registers. Recently, Itoh, et al. extended the attack to public-key cryptosystems [8]. For example, in step 5 in Alg. 4, a correlation of data are given by $T[0] \leftarrow T[0]$ if $d_i = 0$, and $T[0] \leftarrow T[1]$ if $d_i = 1$; Actually these operations are same, but substituted data are loaded from different registers. A main idea of ADPA is to detect this correlation. Itoh, et al. concluded that only the exponent splitting countermeasure [5], in which the scalar is divided into $d = (d-r)+r$ for a random number r, and the randomized window method [13] are resistant against the attack [8]. But as a drawback, required processing speed become at least twice than that of with-

Alg. 5. 4-bit window method

```
INPUT: d[], P
OUTPUT: dP
 1:  W[0] = 0, W[1] = P
 2:  W[2] = ECDBL(W[1])
 3:  for i=3 upto 15 {
 4:    W[i] = ECADD(W[i-1],W[1])
 5:  }
 6:  T = W[d[n-1,n-4]]
 7:  for i=n-5 downto 0 step -4 {
 8:    T = ECDBL(T), T = ECDBL(T)
 9:    T = ECDBL(T), T = ECDBL(T)
10:    T = ECADD(T,W[d[i,i-3]])
11:  }
12:  return T
```

Alg. 6. 4-bit window method with RPC

```
INPUT: d[], P
OUTPUT: dP
 1:  W[0] = 0, W[1] = RPC(P)
 2:  W[2] = ECDBL(W[1])
 3:  for i=3 upto 15 {
 4:    W[i] = ECADD(W[i-1],W[1])
 5:  }
 6:  T = W[d[n-1,n-4]]
 7:  for i=n-5 downto 0 step -4 {
 8:    T = ECDBL(T), T = ECDBL(T)
 9:    T = ECDBL(T), T = ECDBL(T)
10:    T = ECADD(T,W[d[i,i-3]])
11:  }
12:  return invRPC(T)
```

out countermeasures. Another countermeasure is proposed by May, et al.[24], however, it requires a special hardware using a method called the Random Register Renaming (RRR).

2.5 Window-based Method

In a scalar exponentiation, pre-computed tables sometimes deduce the computing time if extra registers are available (the *window-based method*). The most simplest example is in Algorithm 5 with window size 4 just for simplicity (where n is supposed to be a multiple of 4).

Similar to the binary method (Algorithm 4), the window method is also vulnerable to SPA, because ECADD in step 10 is not computed if `W[i,i-3]` $= \mathcal{O}$. Möller proposed a method to construct an addition chain in which `W[]` is never equal to $\mathcal{O}$[18),19)], which assures the SPA-resistance. One can combine RPC or RC countermeasures with Möller's methods (Algorithm 6 for RPC case) in order to resist DDPA. However, Okeya-Sakurai showed the insecureness against the second-order data-bit DPA[31)]. Other approach to resist both DDPA and ADPA is proposed by Itoh, et al.[13)], which randomizes the window to be added in step 10 in Algorithm 5.

3. Proposed Countermeasure

In this section, we propose a practical countermeasure, the *randomized addressing method* (RA), against the address-bit DPA by randomizing addresses of registers used in a scalar exponentiation. An overhead of the countermeasure is small and the effectiveness will be shown by experimental results described in this section.

3.1 Outline

The address-bit DPA is based on the dependency of addresses of registers on the secret key. In other word, addresses of registers are determined by the secret key uniquely, because in Algorithm 1, for example, $T[0] \leftarrow T[0]$ if $d_i = 0$ and $T[0] \leftarrow T[1]$ if $d_i = 1$ so that if d_i changes, then registers will change, too.

A weakness lies on the direct correlation between the secret key and addresses of registers. What we should hide is the correlation (rather than the scalar). So we randomize addresses of registers by a one-time random number $r_{n-1}2^{n-1} + \cdots + r_1 2 + r_0$ $(r_i \in \{0,1\})$. We change all parameters d_i to $d_i \oplus r_i$, where $\oplus$ denotes the XOR operation. Then all addresses of registers are randomized so that the side channel information will be randomized for each scalar exponentiation. This is a basic idea of our proposing countermeasure, the *randomized addressing method* (RA), against ADPA.

The most advantage of our method is the small overhead; the random number is easily generated and required additional operations are just XORs. However, our countermeasure has no DDPA-resistance. We have to combine other countermeasures to resist all DPAs. A DDPA-resistant scheme can be enhanced to an ADPA-scheme with almost no cost. Moreover, our countermeasure can be easily applied to window methods.

3.2 Description of Algorithms

Example algorithms of our countermeasure combined with the add-and-double-always method, the Montgomery ladder and the window method are in Algorithm 7-9. All sample algorithms are resistant against all of SPA, DDPA, and ADPA. The functions `R`,`invR` denote `RPC`,`invRPC` or `RC`,`invR` described in Section 2.3, respectively.

In these algorithms, a random number r is generated at the beginning and stored. In order to reduce the memory, it is possible to generate r_i's on the fly, namely to generate 1-bit random

Alg. 7. Proposed countermeasure (add-and-double-always method)

```
INPUT: d[], P
OUTPUT: dP
T[2] = R(P)
T[r[n-1]] = T[2]
for i=n-2 downto 0 {
  T[r[i+1]] = ECDBL(T[r[i+1]])
  T[1-r[i+1]] =
          ECADD(T[r[i+1]],T[2])
  T[r[i]] = T[d[i]⊕r[i+1]]
}
return invR(T[r[0]])
```

Alg. 8. Proposed countermeasure (Montgomery ladder)

```
INPUT: d[], P
OUTPUT: dP
T[r[n-1]] = R(P)
T[1-r[n-1]] = ECDBL(T[r[n-1]])
for i=n-2 downto 0 {
  T[2] = ECDBL(T[d[i]⊕r[i+1]])
  T[1] = ECADD(T[T[0],T[1])
  T[0] = T[2-(d[i]⊕r[i])]
  T[1] = T[1+(d[i]⊕r[i])]
}
return invR(T[r[0]])
```

Alg. 9. Proposed countermeasure (4-bit window method)

```
INPUT: d[], P
OUTPUT: dP
W[0] = 0, W[1⊕r] = P
W[2⊕r] = ECDBL(W[1⊕r])
for i=3 upto 15 {
  W[i⊕r] = ECADD(W[(i-1)⊕r],W[1⊕r])
}
T = W[d[n-1,n-4]⊕r]
for i=n-5 downto 0 step -4 {
  T = ECDBL(T), T = ECDBL(T)
  T = ECDBL(T), T = ECDBL(T)
  T = ECADD(T,W[d[i,i-3]⊕r])
}
return invR(T)
```

numbers when they are used. However, it may also reduce the security or the processing speed.

Note 1 In the above descriptions, we mainly thought of software implementations. However, even for hardware implementations, ADPA will be successful if indexed registers (e.g., DRAM) are used, because ADPA detects the correlation between indexes and the secret key. In short, ADPA does not depend on implementations 8),9),24). Since RA randomizes destinations of registers, it actually contributes as a countermeasure.

3.3 Security Analysis

Let us discuss the security of our countermeasure. Basically our scheme is designed to combine other countermeasures to totally resist the power analysis; we only consider the security against ADPA. In Alg. 7-9, addresses of registers are determined by $d_i \oplus r_i$. ADPA can distinguish whether two addresses corresponding to d_i and d_j are same or not. If these addresses are same, an adversary can know $d_i \oplus r_i = d_j \oplus r_j$. But the adversary cannot determine $d_i = d_j$ or not, because r_i, r_j are chosen randomly. Conversely, even if $d_i = d_j$, addresses are not always same by r_i and r_j. Thus addresses of registers are randomized and our countermeasure is secure against ADPA.

3.4 Comparison

We should mention a hardware-based countermeasure proposed by May, et al. 24), which is called the Randomized Register Renaming (RRR). A basic idea of RRR is very similar to ours. However, RRR is assumed to be implemented on a special hardware called the Non-Deterministic Instructions Stream Computer (NDISC), which processes given operations in randomized order, if possible. RRR use physically separated registers from virtual registers described in the code. Consequently RRR on NDISC assures the SPA- and DPA-resistance without taking other countermeasures in the programming level.

Proposed countermeasure does not assume such special hardwares; our countermeasure can be implemented on various processors and can be implemented by only software with very simple program codes. These specifications make our countermeasure very practical. Since our countermeasure only requires physical registers, implementers can easily grasp what is being computed in the program. Even if other security-hole has been discovered, implementers can easily take more countermeasures.

There is a tradeoff between RRR and our countermeasure. RRR resists all DPAs on special hardware, while proposed countermeasure resists only ADPA. In order to resist all DPAs, proposed countermeasure should be combined to other DPA-countermeasures.

3.5 Experimental Results

We performed an experiment for verifying the effect on the security by using our countermeasure. We used the address-bit DPA attack against an implementation of Montgomery ladder using RPC with and without the register

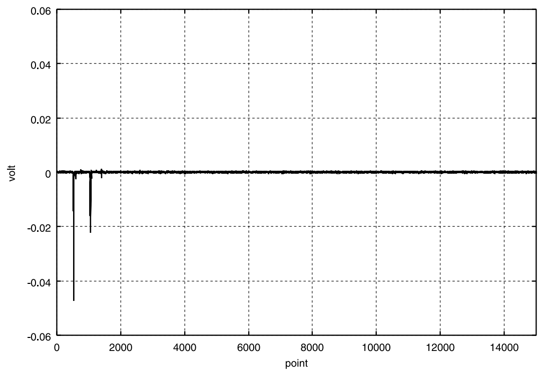

Fig. 1 Differential power traces without register randomization.

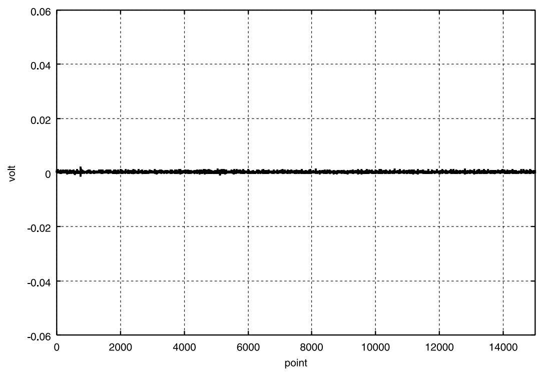

Fig. 2 Differential power traces with register randomization (proposed method).

randomization. In the experiment, the target processor was run at 10 MHz, the sampling ratio was set to 100 MHz, and we made a differential power trace as a difference between means of 10000 traces when loading $Q[\mathrm{addr}[d_a]]$ and means of 10000 traces when loading $Q[\mathrm{addr}[d_b]]$ for $d_a \neq d_b$ where $\mathrm{addr}[d_x](x = a$ or $b)$ represents the address value determined from d_x in each implementation. **Figure 1** shows a differential power trace without register randomization, and **Fig. 2** shows that with register randomization. In Fig. 1, some spikes showing the evidence for $d_a \neq d_b$ are observed, but they are not found in Fig. 2. Hence we confirmed the effect of our countermeasure for protecting against address-bit DPA.

4. Combining Countermeasures

As in the previous section, our countermeasure was dedicated to resist only ADPA. In order to resist all DPAs, other countermeasures should be combined. However, finding a good combination which has a certain security level and compromisable processing speed is a hard task for implementers, for there were no standard criteria to compare these (combined) countermeasures. In this section, we provide some criterion for comparing countermeasures including security level, processing speed, and amount of required registers. Moreover we show that combined countermeasures can be evaluated by these criterion. As a result, our proposed countermeasure can be combined to establish a practical solution for resisting the power analysis in real world.

4.1 Evaluation Technique

We examine an exponentiation in 160-bit ECC from the security level, processing speed, and amount of required registers. In the followings, the binary methods and the Montgomery Ladder are called the *base algorithms* to which countermeasures are applied. We do not discuss the window-based cases nor non-generic methods for space limitation.

4.1.1 Security Level

In order to evaluate the security of a countermeasure against each of SPA, DDPA and ADPA, we adopt a criteria called the *Attenuation Ratio* (AR) proposed by Itoh, et al.[13]. Let denote the ratio against SPA, DDPA and ADPA by AR_S, AR_d and AR_a respectively. Originally, only AR_d was considered[8]. AR_d is between 0 and 1 and is desired to be smaller. If $\mathrm{AR}_d = 0$, an adversary cannot observe spikes by any cost and the countermeasure is secure; if $\mathrm{AR}_d = 1$, the adversary can always observe spikes and the countermeasure is totally insecure. In order to extend AR_d to AR_S and AR_a with inheriting these properties, we define AR_S and AR_a as follows. AR_S takes 0 or 1: $\mathrm{AR}_S = 0$ implies the vulnerability to SPA and $\mathrm{AR}_S = 1$ implies the resistance to SPA. On the other hand, AR_a only take 0 or 1, rather than arbitrary values between 0 and 1. A brief explanation is in the appendix.

Note 2 In the above discussion on AR, we assumed that noises are disappeared during the averaging process. In other words, AR ratios are just theoretical bounds of countermeasures. Different values may be led from real observations.

4.1.2 Processing Speed

Processing speed of a scalar exponentiation dP is measured by the numbers of ECADDs denoted N_A and ECDBLs denoted N_D. For the base algorithms, N_D and N_A are given by integers. If a countermeasure requires ECADDs a times than before, we denote it by "$\times a$". We ignore the cost for randomizations or transformations required in the countermeasure because they are relatively small compared to N_A and N_D.

4.1.3 Register

Amount of required registers are measured by the number of points R_P and that of scalars R_s in a scalar exponentiation. We assume all points are represented by the projective coordinate just for simplicity. For the base algorithms, R_P and R_s are given by integers. For countermeasures, R_P and R_s are evaluated as extra points and scalars required in the exponentiation. If a countermeasure requires b extra points than before, R_P is denoted by "$+b$". We ignore the temporary registers for ECADD and ECDBL.

4.2 Countermeasures

In this section, we evaluate these values for each countermeasure. We only deal with countermeasures whose recommended parameters are explicitly given in the original papers, so we exclude RRR and some other countermeasures. We did not deal with the randomized addition-subtraction chain[27] because it is shown to be insecure[30),32),35)].

4.2.1 SPA Countermeasure

The add-and-double-always method (Alg. 2) computes ECDBL and ECADD for every bit and is SPA-resistant ($\mathrm{AR}_S = 0$). But data computed in step 5 is predictable ($\mathrm{AR}_d = 1$) and a correlation of addresses in step 5 is related to d ($\mathrm{AR}_a = 1$). Required ECDBLs are same ($\mathrm{N}_D = \times 1$) but ECADDs are doubled in average ($\mathrm{N}_A = \times 2$). Extra points and scalars are not required ($\mathrm{R}_P = +0,\ \mathrm{R}_s = +0$). Same discussion can be applied to the LSB case.

The Montgomery Ladder (Alg. 3) computes ECDBL and ECADD for every bit and is SPA-resistant ($\mathrm{AR}_S = 0$). But data computed in step 5 and 6 are predictable ($\mathrm{AR}_d = 1$) and a correlation of addresses of registers in step 3,5,6 are related to d ($\mathrm{AR}_a = 1$). Required ECDBLs and ECADDs are 160 ($\mathrm{N}_D = \mathrm{N}_A = 160$). Required registers are $\mathrm{R}_P = 3,\ \mathrm{R}_s = 1$.

4.2.2 DPA Countermeasure by Data Randomization

In the Randomized Projective Coordinate (RPC)[4], a point (X, Y, Z) is randomized to (rX, rY, rZ) by a 160-bit random number r and it resists DDPA ($\mathrm{AR}_d = 2^{-160}$). As it is vulnerable to SPA ($\mathrm{AR}_S = 1$), it must be used with the add-and-double-always method or the Montgomery Ladder. It is also vulnerable to ADPA ($\mathrm{AR}_a = 1$). Processing speed is unchanged ($\mathrm{N}_A = \mathrm{N}_D = \times 1$). An extra point for the randomized point and an extra scalar for random number are required ($\mathrm{R}_P = +1,\ \mathrm{R}_s = +1$).

In the Randomized Curve (RC)[14], a point (X, Y, Z) is randomized to (r^2X, r^3Y, Z) on an isomorphic curve by a 160-bit random number r and it resists DDPA ($\mathrm{AR}_d = 2^{-160}$). But it is vulnerable to SPA and ADPA ($\mathrm{AR}_S = \mathrm{AR}_a = 1$). Processing speed is unchanged ($\mathrm{N}_A = \mathrm{N}_D = \times 1$). Extra scalars for r and coefficients of isomorphic curves are required ($\mathrm{R}_P = +0,\ \mathrm{R}_s = +3$).

In the Randomized Base Point[4], a scalar exponentiation dP is computed by $d(P + R) - dR$ for a random point R. As R is chosen for each exponentiation, the method is DDPA-resistant ($\mathrm{AR}_d = 2^{-160}$). But it is vulnerable to SPA and ADPA ($\mathrm{AR}_S = \mathrm{AR}_a = 1$). The countermeasure requires two exponentiations and an extra ECADD ($\mathrm{N}_D = \times 2,\ \mathrm{N}_A = \times 2 + 1$). Extra registers for R and $P + R$ are required ($\mathrm{R}_P = +2,\ \mathrm{R}_s = +0$).

4.2.3 DPA Countermeasure by Scalar Randomization

In the Randomized Exponent[4], a scalar d is randomized to $d + r\phi$ for a random number r and the order ϕ of the base point P. In the original paper, the length of r is 20-bit and, thus, the countermeasure is DDPA-resistant ($\mathrm{AR}_d = 2^{-20}$) (There is an analysis which claims the 20-bit randomization is not sufficient[29]. Of course, we can relax the condition to 160-bit, however, the processing speed becomes much slower). It is also ADPA-resistant ($\mathrm{AR}_a = 0$), however it is vulnerable to SPA ($\mathrm{AR}_S = 1$). Processing speed becomes slower for the scalar is 20-bit longer ($\mathrm{N}_D = \mathrm{N}_A = \times 180/160 = \times 1.13$). An extra register for r is required ($\mathrm{R}_P = +0,\ \mathrm{R}_s = +1$).

In the Randomized Start Point[21], a start bit is chosen from a 160-bit scalar and an exponentiation is computed from the chosen bit by MSB for upper bits and by LSB for lower bits. However, the effect is rather small ($\mathrm{AR}_d = 1/160 = 2^{-7.3}$). It is also ADPA-resistant ($\mathrm{AR}_a = 0$), however it is vulnerable to SPA ($\mathrm{AR}_S = 1$). There requires no extra process ($\mathrm{N}_D = \mathrm{N}_A = \times 1$) and register ($\mathrm{R}_P = \mathrm{R}_s = +0$).

In the Exponent Splitting[5], a scalar d is divided into r and $d - r$ for a 160-bit random num-

Table 1 Comparison of countermeasures for 160-bit ECC (non-window methods).

No.	Method	AR_S	AR_d	AR_a	N_D	N_A	R_P	R_s
1	*binary method (form MSB)*	1	1	1	160	80	1	0
2	*binary method (from LSB)*	1	1	1	160	80	2	0
3	add-and-double-always method	0	1	1	$\times1$	$\times2$	+1	+0
4	*Montgomery ladder*	0	1	1	160	160	3	0
5	Randomized Projective Coordinate	1	2^{-160}	1	$\times1$	$\times1$	+1	+1
6	Randomized Curve	1	2^{-160}	1	$\times1$	$\times1$	+0	+3
7	Randomized Base Point	1	2^{-160}	1	$\times2$	$\times2$	+2	+0
8	Randomized Exponent ($\lvert r\rvert = 20$)	1	2^{-20}	0	$\times1.13$	$\times1.13$	+0	+1
9	Randomized Start Point	1	$2^{-7.3}$	0	$\times1$	$\times1$	+0	+0
10	Exponent Splitting	1	2^{-160}	0	$\times2$	$\times2+1$	+1	+2
11	Randomized Addressing (RA)	1	1	0	$\times1$	$\times1$	+0	+2
1+3+5+11	Best Combination	0	2^{-160}	0	160	160	3	3
1+3+6+11	Best Combination	0	2^{-160}	0	160	160	2	5
4+5+11	Other Solution	0	2^{-160}	0	160	160	4	3
4+6+11	Other Solution	0	2^{-160}	0	160	160	3	5

ber r. As the secret information d is randomized, it is resistant against DDPA and ADPA ($\mathrm{AR}_a = 0$, $\mathrm{AR}_d = 2^{-160}$). However, it is vulnerable to SPA ($\mathrm{AR}_S = 1$). The countermeasure requires two exponentiations and an extra ECADD ($\mathrm{N}_D = \times2$, $\mathrm{N}_A = \times2 + 1$). Extra registers for $(d - r)P$, r, d are required ($\mathrm{R}_P = +1$, $\mathrm{R}_s = +2$).

In the Randomized Addressing (RA) (Alg. 5), proposed in this paper, registers in step 4,5,6 are determined by a random bit r_i. It is ADPA-resistant ($\mathrm{AR}_a = 0$), however, it is vulnerable to SPA and DDPA ($\mathrm{AR}_S = \mathrm{AR}_d = 1$). There requires no extra process ($\mathrm{N}_D = \mathrm{N}_A = \times1$). Extra registers for r and $r \oplus d$ are required ($\mathrm{R}_P = +0$, $\mathrm{R}_s = +2$).

4.3 Combining Countermeasures

As we saw in the previous section, some countermeasures only resist specific attacks. Implementers should combine them to resist all power analysis. In our setting, we can easily evaluate and compare each combined countermeasures (**Table 1**). Here we have three choices from three categories. Firstly, an addition chain is chosen from 1-4 (more precisely, from 1,2,4,1+3 or 2+3). Then secondly, a DPA countermeasure is chosen from 5-10. Finally, if ADPA countermeasure is needed, choose 11. In this way, we can combine countermeasures and evaluate their performance. The security level AR_S, AR_d, AR_a can be evaluated by their product. Similarly, the processing speed is evaluated by their product and amount of registers are evaluated by their sum. For example, if we use the Montgomery Ladder combined with RPC and RA, which is denoted as 4+5+11 in the following Table 1, the security levels, processing speed and amount of registers are given by $\mathrm{AR}_S = 0$, $\mathrm{AR}_d = 2^{-160}$, $\mathrm{AR}_a = 0$, $\mathrm{N}_D = 160$, $\mathrm{N}_A = 160$, $\mathrm{R}_P = 4$, $R_s = 3$.

4.4 Comparison

This section provides a complete comparison of countermeasures (Table 1). The base algorithms are described in *italic*. As in the previous section, the security level can be evaluated for combined countermeasures. By this table, we can easily choose the most suitable countermeasure(s) from the security level, processing speed and amount of required registers.

Form Table 1, we can conclude that combinations 1+3+5+11 and 1+3+6+11 provide the best combination from security level and processing speed. Other possible solutions are 4+5+11 and 4+6+11. In these combinations, more effective addition formula for ECADD and ECDBL are applicable and faster processing speed and fewer amount of registers can be expected 12). In these combinations, our proposed countermeasure are used.

Similar discussion for window-based methods can be established. We do not compare them here for a space limitation.

5. Concluding Remarks

In this paper, we proposed a practical countermeasure against the address-bit DPA applicable to ECC and RSA. Unlike the similar result of RRR, our method can be implemented in software level and thus has more flexibility. A recent result of Han, et al. implies the applicability of our countermeasure to other cryptosystems 6).

In order to resist the power analysis, considering countermeasures against each attack is an important factor for implementers. However, when they are to establish a total security, combinations of some countermeasures are more important. For this purpose, our table (Table 1) will be a great help.

References

1) Akkar, M., Dischamp, P. and Moyart, D.: Power Analysis, What is Now Possible..., *Asiacrypt 2000*, LNCS 1976, pp.489–502, Springer-Verlag (2000).
2) Brier, E. and Joye, M.: Weierstraß Elliptic Curves and Side-Channel Attacks, *PKC 2002*, LNCS 2274, pp.335–345, Springer-Verlag (2002).
3) Blake, I., Seroussi, G. and Smart, N.: *Elliptic Curves in Cryptography*, Cambridge University Press (1999).
4) Coron, J.: Resistance against differential power analysis for elliptic curve cryptosystem, *CHES'99*, LNCS 1717, pp.292–302, Springer-Verlag (1999).
5) Clavier, C. and Joye, M.: Universal exponentiation algorithm—A first step towards provable SPA-resistance, *CHES 2001*, LNCS 2162, pp.300–308, Springer-Verlag (2001).
6) Han, D.-G., Lim, J. and Sakurai, K.: On Insecurity of the Side Channel Attack on XTR, *Proc. 2004 Symposium on Cryptography and Information Security* (*SCIS2004*), pp.671–676 (2004).
7) IEEE P1363, Standard Specifications for Public-Key Cryptography (2000).
8) Itoh, K., Izu, T. and Takenaka, M.: Address-bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA, *CHES 2002*, LNCS 2523, pp.129–143, Springer-Verlag (2003).
9) Itoh, K., Izu, T. and Takenaka, M.: An Address-bit Differential Power Analysis of the Countermeasure with Montgomery-form Elliptic Curves (in Japanese), to appear, *IPSJ Journal*, Vol.45, No.7 (2004).
10) Itoh, K., Izu, T. and Takenaka, M.: A Practical Countermeasure against Address-bit Differential Power Analysis, *CHES 2003*, LNCS 2779, pp.382–396, Springer-Verlag (2003).
11) Izu, T., Möller, B. and Takagi, T.: Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks, *Indocrypt 2002*, LNCS 2551, pp.296–313, Springer-Verlag (2002).
12) Izu, T. and Takagi, T.: A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks, *PKC 2002*, LNCS 2274, pp.280–296, Springer-Verlag (2002).
13) Itoh, K., Yajima, J., Takenaka, M. and Torii, N.: DPA Countermeasures by Improving the Window Method, *CHES 2002*, LNCS 2523, pp.303–317, Springer-Verlag (2003).
14) Joye, M. and Tymen, C.: Protections against Differential Analysis for Elliptic Curve Cryptography, *CHES 2001*, LNCS 2162, pp.377–390, Springer-Verlag (2001).
15) Joye, M. and Yen, S-M.: The Montgomery Powering Ladder, *CHES 2002*, LNCS 2523, pp.291–302, Springer-Verlag (2003).
16) Kocher, C.: Timing attacks on Implementations of Diffie-Hellman, RSA, DSS, and other systems, *Crypto'96*, LNCS 1109, pp.104–113, Springer-Verlag (1996).
17) Kocher, C., Jaffe, J. and Jun, B.: Differential power analysis, *Crypto'99*, LNCS 1666, pp.388–397, Springer-Verlag (1999).
18) Möller, B.: Securing Elliptic Curve Point Multiplication against Side-Channel Attacks, *ISC 2001*, LNCS 2200, pp.324–334, Springer-Verlag (2001).
19) Möller, B.: Parallelizable Elliptic Curve Point Multiplication Method with Resistance against Side-Channel Attacks, *ISC 2002*, LNCS 2433, pp.402–413, Springer-Verlag (2002).
20) Montgomery, P.: Speeding the Pollard and elliptic curve methods for factorizations, *Math. of Comp*, Vol.48, pp.243–264 (1987).
21) Messerges, T., Dabbish, E. and Sloan, R.: Power Analysis Attacks of Modular Exponentiation in Smartcards, *CHES'99*, LNCS 1717, pp.144–157, Springer-Verlag (1999).
22) Messerges, T., Dabbish, E. and Sloan, R.: Investigations of Power Analysis Attacks on Smartcards, preprint, *USENIX Workshop on Smartcard Technology* (1999).
23) Messerges, T., Dabbish, E. and Sloan, R.: Examining Smart-Card Security under the Threat of Power Analysis Attacks on Smartcards, *IEEE Trans. Comput.*, Vol.51, No.5, pp.541–552 (2002).
24) May, D., Muller, H.L. and Smart, N.P.: Random Register Renaming to Foil DPA, *CHES 2001*, LNCS 2162, pp.28–38, Springer-Verlag (2001).
25) National Institute of Standards and Technology, Recommended Elliptic Curves for Federal Government Use, in the appendix of FIPS 186-2.
26) Oswald, E.: Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems, *CHES 2002*, LNCS 2523, pp.82–97, Springer-Verlag (2003).
27) Oswald, E. and Aigner, M.: Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks, *CHES 2001*, LNCS 2162, pp.39–50, Springer-Verlag (2001).
28) Okeya, K., Kurumatani, H. and Sakurai, K.: Elliptic curves with the Montgomery form and their cryptographic applications, *PKC 2000*, LNCS 1751, pp.446–465, Springer-Verlag

(2000).

29) Okeya, K. and Sakurai, K.: Power analysis breaks elliptic curve cryptosystem even secure against the timing attack, *Indocrypt 2000*, LNCS 1977, pp.178–190, Springer-Verlag (2000).

30) Okeya, K. and Sakurai, K.: On Insecurity of the Side Channel Attack Countermeasure Using Addition-Subtraction Chains under Distinguishability between Addition and Doubling, *ACISP 2002*, LNCS 2384, pp.420–435, Springer-Verlag (2002).

31) Okeya, K. and Sakurai, K.: A Second-Order DPA Attack Breaks a Window-method based Countermeasure against Side Channel Attacks, *ISC 2002*, LNCS 2443, pp.389–401, Springer-Verlag (2002).

32) Okeya, K. and Sakurai, K.: A Multiple Power Analysis Breaks the Advanced Version of the Randomized Addition-Subtraction Chains Countermeasure against Side Channel Attacks, to appear in *Proc. 2003 IEEE Information Theory Workshop.*

33) Standards for Efficient Cryptography Group (SECG), Specification of Standards for Efficient Cryptography.

34) Walter, C.: MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis, *CT-RSA 2002*, LNCS 2271, pp.53–66, Springer-Verlag (2002).

35) Walter, C.: Security Constraints on the Oswald-Aigner Exponentiation Algorithm, Cryptology ePrint Archive, Report 2003/013, 2003. Available from `http://eprint.iacr.org/2003/013/`

Appendix

In this appendix, we briefly discuss the characteristics of AR_a described in Section 4, in which the signal content, not noise content, is evaluated. Merit of this idea is as follows. The possibility of the real-world DPA attack depends on the signal-to-noise ratio (SNR), and its evaluation method is proposed in Refs. 22) and 23). This may lead the exact evaluation, but needs the parameters of the signal and noise, which are peculiar to the target and it is difficult to archive the target-independent evaluation. In opposite, we evaluate the ratio of signal by AR, and if its absolute value is very little, countermeasure is regarded as DPA-resistant. In comparison with SNR, AR is indirect, but it is evaluated by only the characteristic of the countermeasure. Hence, merit of our evaluation is target-independent.

Suppose a DPA-adversary inputs data $a_1, \ldots, a_N$ into the target device, and obtains power traces $V(a_1,t), \ldots, V(a_N,t)$, where t denotes the time. Let $\delta_R(V(a_j,t'))$ be an evaluation function (the hamming weight, for example) of $V(a_j,t')$ for an input a_j at time $t=t'$, and GR_0, GR_1 be un-overlapped sets of range of $\delta_R(V(a_j,t'))$. If N is large enough and elements of GR_0 and GR_1 is roughly same, we have a difference

$$\Delta_R(t) = \frac{2}{N}\left(\sum_{j \text{ s.t. } \delta_R(V(a_j,t'))\in GR_0} V(a_j,t) - \sum_{j \text{ s.t. } \delta_R(V(a_j,t'))\in GR_1} V(a_j,t)\right).$$

In DPA, the adversary assumes a base value of data or address, simulates the target computation for time $t=t'$, and computes the evaluation function $\delta_S(V(a_j,t'))$ for each a_j, Let GS_0, GS_1 be un-overlapped sets of range of $\delta_S(V(a_j,t'))$. Then the difference function is given by

$$\Delta_S(t) = \frac{2}{N}\left(\sum_{j \text{ s.t. } \delta_S(V(a_j,t'))\in GS_0} V(a_j,t) - \sum_{j \text{ s.t. } \delta_S(V(a_j,t'))\in GS_1} V(a_j,t)\right).$$

If the assumption is correct, there appears a spike at $\Delta_S(t')$, which implies $\delta_S(V(a_j,t')) = \delta_R(V(a_j,t'))$ and thus $GR_0 = GS_0$, $GR_1 = GS_1$.

When countermeasures are used, $\delta_S(V(a_j,t'))$ is not always equal to $\delta_R(V(a_j,t'))$. Let p be the probability being $\delta_S(V(a_j,t')) = \delta_R(V(a_j,t'))$. Then we have

$$\Delta_S(t) = p\times\Delta_R(t) + (1-p)\times\Delta_F(t),$$

for

$$\Delta_F(t) = \frac{2}{N}\left(\sum_{j \text{ s.t. } \delta_S(V(a_j,t'))\in GS_0} V(a_j,t) - \sum_{j \text{ s.t. } \delta_S(V(a_j,t'))\in GS_1} V(a_j,t)\right)$$

in which $\delta_S(V(a_j,t')) \neq \delta_R(V(a_j,t'))$.

Under above notations, AR is defined as a ratio of $\Delta_R(t)$ and $\Delta_S(t)$ at $t=t'$ by

$$\mathrm{AR} = \left|\frac{\max \Delta_S(t')}{\max \Delta_R(t')}\right| \approx \left|p + (1-p) \times \frac{\max \Delta_F(t')}{\max \Delta_R(t')}\right|. \qquad (2)$$

Note that the spike appears at $t = t'$, and p represents the probability decided by the attacker's guess of an intermediate value of the target device at $t = t'$, which is evaluated with the data entropy of the countermeasure. That is, p is in inverse propotion of the product of the independent random number size of the plural countermeasures. By the definition, AR is between 0 and 1. When it is likely to $\delta_S(V(a_j, t')) = \delta_R(V(a_j, t'))$ for many t's, AR will be near to 1 (weak countermeasures). On the other hand, when $\delta_S(V(a_j, t')) \neq \delta_R(V(a_j, t'))$ for almost every t's, AR will be 0 (strong countermeasures).

Theoretical evaluation of (2) differs between DDPA and ADPA because of the difference of $\Delta_F(t)$. In DDPA, $\Delta_F(t)$ will be 0, since $\delta_R(V(a_j, t')) \neq \delta_S(V(a_j, t'))$ and there is no relation between GS_x and GR_y $(x, y = 0, 1)$. Hence we have $\mathrm{AR}_d = |p|$. On the other hand, in ADPA, possible values of δ_R, δ_S are limited to 2 patterns. Thus $\delta_R \neq \delta_S$ implies that $GS_0 = GR_1$, $GS_1 = GR_0$. Consequently, we have $\Delta_F(t) = -\Delta_R(t)$ and $\mathrm{AR}_a = |2p - 1|$. When no ADPA-countermeasure is used, we have $p = 1$ and $\mathrm{AR}_a = 1$. On the other hand, when ADPA-countermeasures work ideally, the probability p will be $1/2$ and we have $\mathrm{AR}_a = 0$.

(Received December 1, 2003)
(Accepted June 8, 2004)

Kouichi Itoh was born in 1971. He received his B.E. and M.E. degrees from Tokyo Institute of Technology in 1995 and Japan Advanced Institute of Science and Technology in 1997, respectively. In 1997, he joined FUJITSU LABORATORIES LTD., where he has been involved in resecrch and development of implementation of public-key and common-key cryptosystems and side channel attacks. He was awarded SCIS (Symposium on Cryptography and Information Security) paper prize in 2002, and CSS (Computer Security Symposium) paper prize in 2002. He is a member of IEICE.

Tetsuya Izu was born in 1967. He received his B.S. and M.S. degrees from The University of Tokyo in 1992 and Rikkyo University in 1994, respectively. After finishing the doctor course, he joined FUJITSU LABORATORIES LTD. in 1997. He has been involved in resecrch on cryptography and information security. He was a visiting reseracher at the University of Waterloo, Canada, in 2001. He was awarded SCIS (Symposium on Cryptography and Information Security) paper prize in 1999, and CSS (Computer Security Symposium) paper prize in 2002. He is a member of IEICE, IPSJ, JSIAM and IACR.

Masahiko Takenaka was born in 1967. He received his B.E. and M.E. degrees from Osaka University in 1990 and 1992. In 1992, he joined FUJITSU LABORATORIES LTD., where he has been involved in resecrch and development of implementation of public-key and common-key cryptosystems, side channel attacks, and network security. He was awarded CSS (Computer Security Symposium) paper prize in 2002.