# On security of XTR public key cryptosystems against Side Channel Attacks *

Dong-Guk Han[1]**, Jongin Lim[1]***, and Kouichi Sakurai[2]

[1] Center for Information and Security Technologies(CIST),
Korea University, Seoul, KOREA
{christa,jilim}@korea.ac.kr
[2] Department of Computer Science and Communication Engineering 6-10-1,
Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan,
sakurai@csce.kyushu-u.ac.jp

**Abstract.** The XTR public key system was introduced at Crypto 2000. Application of XTR in cryptographic protocols leads to substantial savings both in communication and computational overhead without compromising security. It is regarded that XTR is suitable for a variety of environments, including low-end smart cards, and XTR is the excellent alternative to either RSA or ECC. In [LV00a,SL01], authors remarked that XTR single exponentiation (XTR-SE) is less susceptible than usual exponentiation routines to environmental attacks such as timing attacks and Differential Power Analysis (DPA). In this paper, however, we investigate the security of side channel attack (SCA) on XTR. This paper shows that XTR-SE is immune against simple power analysis (SPA) under assumption that the order of the computation of XTR-SE is carefully considered. However we show that XTR-SE is vulnerable to Data-bit DPA (DDPA)[Cor99], Address-bit DPA (ADPA)[IIT02], and doubling attack [FV03]. Moreover, we propose two countermeasures that prevent from DDPA and a countermeasure against ADPA. One of the countermeasures using randomization of the base element proposed to defeat DDPA, i.e., randomization of the base element using field isomorphism, could be used to break doubling attack. Thus if we only deal with SPA, DDPA, ADPA, and doubling attack as the attack algorithm for XTR-SE, XTR-SE should be added following countermeasures: randomization of the base element using field isomorphism (DDPA and doubling attack) + randomized addressing (ADPA). But the proposed countermeasure against doubling attack is very inefficient. So to maintain the advantage of efficiency of XTR a good countermeasure against doubling attack is actually necessary.

**Keywords:** *XTR Public Key Cryptosystem, Side Channel Attacks, SPA, Data-bit DPA, Address-bit DPA, doubling attack*

* This is a "full" version of a paper that will been published in ACISP04.

** This work was done while the first author visits in Kyushu Univ. and was supported by the Korea Science and Engineering Foundation (KOSEF). (M07-2003-000-20123-0)

*** This work was supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

## 1 Introduction

In Crypto 2000 Lenstra and Verheul proposed XTR public key system [LV00a]. It is a novel method that makes use of traces to represent and calculate powers of elements of a subgroup of a finite field. XTR uses the trace over $GF(p^2)$ to represent elements of the order $p^2-p+1$ subgroup of $GF(p^6)^*$, thereby achieving a factor 3 size reduction compared to the traditional representation. The security of XTR relies on the difficulty of solving discrete logarithm related problems in the multiplicative group of a finite field.

In general, it is well known that ECC is suitable for a variety of environments, including low-end smart cards and over-burdened web servers communicating with powerful PC clients. But XTR has some advantages such as its very faster parameter and key selection (much faster than RSA, orders of magnitude faster than ECC), small key sizes (much smaller than RSA, comparable with ECC for current security settings), and speed (overall comparable with ECC for current security settings). Combined with its very easy programmability, this makes XTR an excellent public key system for a very wide variety of environments, ranging from smart cards to web serves.

XTR single exponentiation (XTR-SE) [LV00a,SL01] has a rather unusual property that the two computations involved (i.e., if $m_j = 0$ and $m_j = 1$) are very similar and take the same number of instructions. Thus, the instructions carried out in XTR-SE for the two different cases are very much alike. So, authors remarked that XTR-SE is less susceptible than usual exponentiation routines to environmental attacks such as timing attacks and Differential Power Analysis (DPA).

In this paper, we investigate the security of side channel attack (SCA) on XTR. Especially, we consider following four SCA : simple power analysis (SPA), data-bit differential power analysis (DDPA) proposed by Coron [Cor99], address-bit differential power analysis (ADPA) proposed by Itoh et al. [IIT02], and doubling attack proposed by Fouque et al. [FV03].

This paper shows that XTR-SE is immune against simple power analysis (SPA) under assumption that the order of the computation of XTR-SE is carefully considered. As the instructions performed during XTR-SE does not depend on the secret value being processed the order of computation is flexible. For example, if XTR-SE is implemented as following order $XTRDBL$, $XTRDBL$, and then $XTR\_C_{2n-1}$ if $m_j = 0$, and $XTR\_C_{2n+1}$, $XTRDBL$, and then $XTRDBL$ if $m_j = 1$, then the order of the computation can be easily known to an attacker by SPA. Thus if the order of computation of XTR-SE is not considered XTR-SE could not be any more secure against SPA.

XTR-SE is secure against SPA without any countermeasure if the order of the computation is carefully considered, but we will show that XTR-SE is vulnerable to DDPA, ADPA, and doubling attack.

Moreover, we propose several countermeasures against the proposed attacks. First, we introduce two countermeasures against DDPA as follows: randomization of the base element using field isomorphism and randomization of the private exponent. Also, we propose a countermeasure against ADPA by using random

number. In the case of doubling attack randomization of the base element using field isomorphism which is proposed to defeat DDPA could be used to break doubling attack. Note that the randomization of the private exponent method is not secure against doubling attack [FV03].

Thus if we only deal with SPA, DDPA, ADPA, and doubling attack as the attack algorithm for XTR-SE, XTR-SE should be added following countermeasures: randomization of the base element using field isomorphism (DDPA and doubling attack) + randomized addressing (ADPA).

However, as the proposed countermeasure against doubling attack that is randomization of the base element using field isomorphism is very inefficient, for instance, the cost of XTR-SE with this countermeasure is about 129 times slower than that of XTR-SE without it, the efficiency of XTR-SE with SCA countermeasures could not be comparable with that of ECC or RSA with SCA countermeasures. To maintain the advantage of efficiency of XTR a good countermeasure against doubling attack is actually necessary. Construction of efficient countermeasures against doubling attack is an open question.

Note that we hope this first step towards side channel attack on XTR public key cryptosystems will be a motivating starting point for further research.

## 2 XTR public key system

### 2.1 Preliminaries

In this section we review some of the results from [LV00a,SL01].

For constructing XTR, primes $p$ and $q$ must satisfy following conditions :

- $p$ is prime such that $p$ $(mod\ 3)$ is a primitive element in $Z_3$.
- $\Phi_6(p)$ has a prime factor $q$ of which the size is more than 160 bits.

Note that $\Phi_6(X)$ is $6-th$ cyclotomic polynomial. The first above condition guarantees $GF(p^2)$ has an optimal normal basis of type I [[M93], Theorem 5.2]. And the subgroup with order $q$ cannot be embedded in the multiplicative group of any true subfield of $GF(p^6)$ by the second condition [[L97], Lemma 2.4].

Let $g$ be a generator of the order $q$ subgroup of $GF(p^6)^*$.

**Definition 1.** *The trace $Tr(h)$ over $GF(p^2)$ of $h \in GF(p^6)$ is the sum of the conjugates over $GF(p^2)$ of $h$, i.e.,*

$$Tr(h) = h + h^{p^2} + h^{p^4}.$$

In XTR elements of $< g >$ are represented by their trace over $GF(p^2)$. It was shown that actual representation of the elements of $< g >$ and other elements of $GF(p^6)$ can be avoided. Thus, there is no need to represent elements of $GF(p^6)$, however, representation of $GF(p^2)$ is needed. That is to say XTR uses $GF(p^2)$ arithmetic to achieve $GF(p^6)$ security, without requiring explicit construction of $GF(p^6)$.

XTR has two main advantages compared to ordinary representation of elements of $< g >$:

- It is shorter, since $Tr(h) \in GF(p^2)$, whereas representing an element of $< g >$ requires in general an element of $GF(p^6)$, i.e., three times more bits.
- It allows faster arithmetic, because given $Tr(g)$ and $n$ the value $Tr(g^n)$ can be computed substantially faster than $g^n$ can be computed given $g$ and $n$.

Throughout this paper, $c_n$ denotes $Tr(g^n) \in GF(p^2)$, for some fixed $p$ and $g$ of order $q$, where $q$ divides $\Phi_6(p)$. Note that in [LV00a,LV00b,LV01] it is shown how $p, q$, and $c_1$ can be found quickly.

**Lemma 1 ([SL01]).** *Let $x, y, z \in GF(p^2)$ with $p \equiv 2 \bmod 3$.*

*i. Computing $x^p$ is free.*
*ii. Computing $x^2$ takes two multiplications in $GF(p)$.*
*iii. Computing $xy$ costs the same as two a half multiplications in $GF(p)$.*
*iv. Computing $xz - yz^p$ costs the same as three multiplications in $GF(p)$.*

Efficient computation of $c_n$ given $p, q$ and $c_1$ is based on the following facts.

**Corollary 1 ([LV00a,SL01]).** *Let $c, c_{n-1}, c_n$ and $c_{n+1}$ be given.*

*i. $c = c_1$.*
*ii. $c_{-n} = c_{np} = c_n^p$ for $n \in Z$.*
*iii. $c_n \in GF(p^2)$ for $n \in Z$.*
*iv. $c_{2n} = c_n^2 - 2c_n^p$ takes two multiplications in $GF(p)$.*
*v. $c_{n+2} = c * c_{n+1} - c^p * c_n + c_{n-1}$ takes three multiplications in $GF(p)$.*
*vi. $c_{2n-1} = c_{n-1} * c_n - c^p * c_n^p + c_{n+1}^p$ takes three multiplications in $GF(p)$.*
*vii. $c_{2n+1} = c_n * c_{n+1} - c * c_n^p + c_{n-1}^p$ takes three multiplications in $GF(p)$.*

Let $S_n = (c_{n-1}, c_n, c_{n+1}) \in GF(p^2)^3$; thus $S_1 = (3, c_1, c_1^2 - 2c_1^p)$. The triple $S_{2n-1} = (c_{2(n-1)}, c_{2n-1}, c_{2n})$ can be computed from $S_n$ and $c_1$ by applying Corollary 1 *iv* twice to compute $c_{2(n-1)}$ and $c_{2n}$ based on $c_{(n-1)}$ and $c_n$, respectively, and by applying Corollary 1 *vi* to compute $c_{2n-1}$ based on $S_n = (c_{n-1}, c_n, c_{n+1})$ and $c_1$. This takes seven multiplications in $GF(p)$. The triple $S_{2n+1}$ can be computed in a similar manner from $S_n$ and $c_1$ at the cost of seven multiplications in $GF(p)$ using Corollary 1 *vii*.

### 2.2 XTR Single Exponentiation

In XTR, the algorithm to compute $Tr(g^n)$ given $Tr(g)$ and $n \in Z$ is needed like the algorithm to compute $g^n$ in public key system based on discrete logarithm problem. We call this algorithm as XTR single exponentiation (XTR-SE). XTR-SE is as follows.

**XTR Single Exponentiation** (cf.[LV00a], Algorithm 2.3.7)

**Input :** $c$ and $n$

**Output :** $S_n = (c_{n-1}, c_n, c_{n+1})$

- If $n < 0$, apply this algorithm to $-n$ and $c$, and apply Corollary 1 *ii* to the resulting value.
- If $n = 0$, then $S_0 = (c^p, 3, c)$ (cf. Corollary 1 *ii*).
- If $n = 1$, then $S_1 = (3, c, c^2 - 2c^p)$ (cf. Corollary 1 *iv*).
- If $n = 2$, use Corollary 1 *v* and $S_1$ to compute $c_3$ and thereby $S_2$.
- Otherwise, to compute $S_n$ for $n > 2$ define $\overline{S}_i = S_{2i+1}$ and let $\overline{m} = n$. If $\overline{m}$ is even, then replace $\overline{m}$ by $\overline{m} - 1$. Let $\overline{m} = 2m + 1$, $k = 1$, and compute $\overline{S}_k = S_3$ using Corollary 1 *iv* and $S_2$.

Let $m = \sum_{j=0}^{l} m_j 2^j$ with $m_j \in \{0, 1\}$ and $m_l = 1$. For $j = l-1, l-2, \ldots, 0$ in succession do the following:

- If $m_j = 0$ then use
  $\overline{S}_k = (c_{2k}, c_{2k+1}, c_{2k+2})$ to compute $\overline{S}_{2k} = (c_{4k}, c_{4k+1}, c_{4k+2})$.
  (using Corollary 1 *iv* for $c_{4k}$ and $c_{4k+2}$ and Corollary 1 *vi* for $c_{4k+1}$)
- If $m_j = 1$ then use
  $\overline{S}_k = (c_{2k}, c_{2k+1}, c_{2k+2})$ to compute $\overline{S}_{2k+1} = (c_{4k+2}, c_{4k+3}, c_{4k+4})$.
- Replace $k$ by $2k + m_j$.

After this iteration $k = m$ and $S_{\overline{m}} = \overline{S}_m$. If $n$ is even use

$$S_{\overline{m}} = (c_{\overline{m}-1}, c_{\overline{m}}, c_{\overline{m}+1}) \text{ to compute } S_{\overline{m}+1} = (c_{\overline{m}}, c_{\overline{m}+1}, c_{\overline{m}+2})$$

(using Corollary 1 *v*) and replace $\overline{m}$ by $\overline{m} + 1$. As a result $S_n = S_{\overline{m}}$.

**Theorem 1 ([SL01], cf. Section 2.4).** *Given the representation $Tr(g) \in GF(p^2)$, the representation $Tr(g^n) \in GF(p^2)$ can be computed in $7 \log_2 n$ multiplications in $GF(p)$, for any integer $n$.*

In above algorithm, the trace $c_1$ of $g$ in $S_1 = (c_0, c_1, c_2)$ can be replaced by the trace $c_t$ of the $t-$th power $g^t$ of $g$: with $\widetilde{c}_1 = c_t$, $\widetilde{S}_1 = (\widetilde{c}_0, \widetilde{c}_1, \widetilde{c}_2) = (3, c_t, c_{2t})$, and by the above theorem, the triple $\widetilde{S}_v = (\widetilde{c}_{v-1}, \widetilde{c}_v, \widetilde{c}_{v+1}) = (c_{(v-1)t}, c_{vt}, c_{(v+1)t})$ can be computed in $7 \log_2 v$ multiplications in $GF(p)$, for any integer $v < q$.

The only difference between the two different cases in XTR-SE is the application of Corollary 1 *vi* if $m_j = 0$ and of Corollary 1 *vii* if $m_j = 1$. But, the two computations involved are very similar and take the same number of instructions. Thus in [[LV00a], Remark 2.3.9] they claimed that XTR-SE is much less susceptible than exponentiation routines to environmental attacks such as timing attacks and Differential Power Analysis.

**Notation :** Define following three functions:

$$\begin{aligned} XTRDBL(c_n) &:= c_{2n}, \\ XTR\_C_{n+2}(c_{n-1}, c_n, c_{n+1}, c) &:= c_{n+2}, \\ XTR\_C_{2n-1}(c_{n-1}, c_n, c_{n+1}, c) &:= c_{2n-1}, \\ XTR\_C_{2n+1}(c_{n-1}, c_n, c_{n+1}, c) &:= c_{2n+1}. \end{aligned}$$

Note that $XTRDBL$, $XTR\_C_{n+2}$, $XTR\_C_{2n-1}$, and $XTR\_C_{2n+1}$ are defined by Corollary 1 *iv*, *v*, *vi*, and *vii*, respectively.

When $n > 2$, above XTR-SE could be simplified as following Table 1.

**Table 1.** XTR Single Exponentiation Algorithm (XTR-SE).

| |
|---|
| INPUT : $c$ and $n$ where $n > 2$ |
| OUTPUT : $S_n = (c_{n-1}, c_n, c_{n+1})$ |
| 1. Compute initial values: |
| 1.1. $C[3] \leftarrow c$, $C[0] \leftarrow XTRDBL(C[3])$, |
| $C[1] \leftarrow XTR\_C_{2n+1}(3, C[3], C[0], C[3])$, |
| and $C[2] \leftarrow XTRDBL(C[0])$. |
| 1.2. If $n$ is even, $n$ replace $n-1$. |
| Let $n = 2m+1$ and $m = \sum_{j=0}^{l} m_j 2^j$ with $m_j \in \{0,1\}$ and $m_l = 1$. |
| 2. for $j = l-1$ downto 0 |
| 2.1. $T[1] \leftarrow XTRDBL(C[m_j])$ |
| 2.2. $T[2] \leftarrow XTRDBL(C[1+m_j])$ |
| 2.3. if $(m_j = 0)$ then |
| $T[3] \leftarrow XTR\_C_{2n-1}(C[0], C[1], C[2], C[3])$ |
| if $(m_j = 1)$ then |
| $T[3] \leftarrow XTR\_C_{2n+1}(C[0], C[1], C[2], C[3])$ |
| 2.4. $C[0] \leftarrow T[1]$ |
| 2.5. $C[1] \leftarrow T[3]$ |
| 2.6. $C[2] \leftarrow T[2]$ |
| 3. If $n$ is odd then |
| return $(C[0], C[1], C[2])$, |
| else $C[0] \leftarrow XTR\_C_{n+2}(C[0], C[1], C[2], C[3])$ |
| return $(C[1], C[2], C[0])$. |

### 2.3 Toy example

Let $n = 181$. Then $m = 90 = 2^6+2^4+2^3+2$, i.e, $(m_6, m_5, m_4, m_3, m_2, m_1, m_0) = (1, 0, 1, 1, 0, 1, 0)$. Given $c$ and $n$, $S_{181}$ could be computed as following Table 2.

## 3 Side Channel Attacks on XTR-SE

In 1998, Kocher described in a technical draft [KJJ98] Simple Power Analysis (SPA) and Differential Power Analysis (DPA) on DES. SPA only uses a single observed information, while DPA uses a lot of observed information together with statistic tools.

**Table 2.** Compute $S_{181}$ given $c$.

| j | $m_j$ | k | $(C[0], C[1], C[2])$ |
|---|---|---|---|
| 6 | 1 | 1 | $(c_2, c_3, c_4)$ |
| 5 | 0 | 2 | $(c_4, c_5, c_6)$ |
| 4 | 1 | 5 | $(c_{10}, c_{11}, c_{12})$ |
| 3 | 1 | 11 | $(c_{22}, c_{23}, c_{24})$ |
| 2 | 0 | 22 | $(c_{44}, c_{45}, c_{46})$ |
| 1 | 1 | 45 | $(c_{90}, c_{91}, c_{92})$ |
| 0 | 0 | 90 | $(c_{180}, c_{181}, c_{182})$ |

In 1999, Messerges et al. proposed a new powerful attack against the secret key cryptosystems, the address-bit DPA (ADPA), which analyzes a correlation between the secret information and addresses of registers [MDS99]. To distinguish from ADPA, we call general DPA as Data-bit DPA (DDPA).

In 2003, Fouque et al. proposed doubling attack against a classical implementation of the modular exponentiation or scalar multiplication in the ECC that only requires two queries to the device [FV03].

In this section, we investigate the security of side channel attack on XTR, especially SPA, DDPA, ADPA, and doubling attack are considered.

### 3.1 XTR-SE is secure against SPA

The computation of the XTR-SE requires the computations repeatedly that ($XTRDBL$, $XTRDBL$, $XTR\_C_{2n-1}$) or ($XTRDBL$, $XTRDBL$, $XTR\_C_{2n+1}$) from ($C[0]$,$C[1]$,$C[2]$,$C[3]$) depending on the value of each bit $m_j$. As $XTR\_C_{2n-1}$ and $XTR\_C_{2n+1}$ require same multiplications in $GF(p)$, these two operations are indistinguishable from the observation of the power consumption. This means that the instructions performed during XTR-SE does not depend on the secret value being processed. Thus XTR-SE is resistant against SPA.

**Caution :** Since $XTRDBL$s and $XTR\_C_{2n-1}$ (or $XTR\_C_{2n+1}$) are independent, the order of computation is flexible. But the order of the computation is very important. For instance, assume that XTR-SE is implemented as following order $XTRDBL$, $XTRDBL$, and then $XTR\_C_{2n-1}$ if $m_j = 0$, and $XTR\_C_{2n+1}$, $XTRDBL$, and then $XTRDBL$ if $m_j = 1$. Then, the order of the computation can be easily known to an attacker by SPA.
Thus, if the order of the computation of XTR-SE is not considered XTR-SE could not be any more secure against SPA.

*Remark 1.* Similar results could be found in the computation of the scalar multiplication on the Montgomery-form elliptic curves [OS00].

*Remark 2.* In step 3 in XTR-SE, a dummy $XTR\_C_{n+2}$ operation is needed when $n$ is odd. Otherwise the least significant bit of $n$, i.e., $n$ is even or not, could be revealed.

### 3.2 Data-bit DPA against XTR-SE

In this section we describe a DDPA [Cor99] against an implementation of XTR-SE. DDPA on XTR-SE can be performed by noticing that at step $j$ the processed $T[1]$ depends only on the first bits $(m_l, \ldots, m_j)$ of $m$. Now assume that we know how field elements are represented in memory $T[i]$ (or $C[i]$) during computation and select a particular bit of this representation. When $C[i]$ is processed to update $T[1]$, power consumption will be correlated to this specific bit of $C[i]$. No correlation will be observed with an element $C[i]$ not computed inside the card . To update $T[1]$ in XTR-SE $C[0]$ is used when $m_j = 0$ and $C[1]$ is used when $m_j = 1$

Thus it is possible to successively recover the bits of the exponent by guessing which $C[i]$ are computed by the card.

For example, the second most significant bit $m_{l-1}$ of $m$ can be recovered by computing the correlation between power consumption and any specific bit of the binary representation of $c_4$. If $m_{l-1} = 0$, $c_4$ is computed in XTR-SE to update $T[1]$, and power consumption is thus correlated with any specific bit of $c_4$. Otherwise if $m_{l-1} = 1$, $c_4$ is never computed to update $T[1]$, and no correlation will be observed with $c_4$. This gives $m_{l-1}$. The following bits of $m$ can be recursively recovered in the same way.

### 3.3 Address-bit DPA against XTR-SE

The address-bit DPA was originally investigated by Messerges, Dabbish and Sloan [MDS99] and Itoh et al. extended the analysis to elliptic curve based cryptosystems [IIT02].

This paper extends the analysis to XTR-SE. Since XTR-SE has similar structure to the Montgomery form elliptic curves, the analysis technic proposed by Itoh et al. could be applicable to XTR-SE.

ADPA [IIT02] is successful if there is a close dependence between a secret value and addresses of accessed registers. Thus if we could find correlations between address value and secret value then XTR-SE is also vulnerable to ADPA.

Following property shows that there are correlations between address value and secret value $m_j$ in XTR-SE.

*Property 1.* In substep 2.1 and 2.2 in XTR-SE,

- When $m_j = 0$
  - To update $T[1]$ read address $C[0]$.
  - To update $T[2]$ read address $C[1]$.
- When $m_j = 1$
  - To update $T[1]$ read address $C[1]$.
  - To update $T[2]$ read address $C[2]$.

From Property 1, ADPA [IIT02] can be applied to XTR-SE. Thus XTR-SE is not any more secure against ADPA.

### 3.4 Doubling Attack against XTR-SE

In CHES 2003, Fouque et al. proposed the new attack against a classical implementation of the modular exponentiation or scalar multiplication in the ECC that only requires two queries to the device [FV03]. Their attack only works for the `Left-to-Right` implementation.

The main idea of the doubling attack is based on the fact that, even if an attacker could not know which computation is done by the device, he/she could at least detect when the device does twice the same operation. Namely, if the device computes $2 \cdot X$ and $2 \cdot Y$, the attacker could not guess the value of $X$ or $Y$ but he/she could check if $X = Y$.

First, consider an example. This example is the same as example described in section 2.3. Then we compare the sequence of operations when XTR-SE is used to compute $S_{180} = (c_{180}, c_{181}, c_{182})$ given $c_1$ and $\widetilde{S}_{180} = (c_{180\cdot 2}, c_{181\cdot 2}, c_{182\cdot 2})$ given $\widetilde{c}_1 = c_2$. Note that these notations are described in section 2.2.

**Table 3.** Compute $S_{181}$ and $\widetilde{S}_{180}$ given $c_1$ and $\widetilde{c}_1 = c_2$, respectively.

| | | | Compute $S_{181}$ given $c_1$ | Compute $\widetilde{S}_{181}$ given $\widetilde{c}_1 = c_2$ |
|---|---|---|---|---|
| $j$ | $m_j$ | $k$ | $(C[0], C[1], C[2])$ | $(C[0], C[1], C[2])$ |
| 6 | 1 | 1 | $(c_2, c_3, c_4)$ | $(\widetilde{c}_2, \widetilde{c}_3, \widetilde{c}_4) = (\lfloor c_{2\cdot 2} \rfloor, c_{3\cdot 2}, c_{4\cdot 2})$ |
| 5 | **0** | 2 | $(\lfloor c_4 \rfloor, c_5, c_6)$ | $(\widetilde{c}_4, \widetilde{c}_5, \widetilde{c}_6) = (c_{4\cdot 2}, c_{5\cdot 2}, c_{6\cdot 2})$ |
| 4 | 1 | 5 | $(c_{10}, c_{11}, c_{12})$ | $(\widetilde{c}_{10}, \widetilde{c}_{11}, \widetilde{c}_{12}) = (c_{10\cdot 2}, c_{11\cdot 2}, c_{12\cdot 2})$ |
| 3 | 1 | 11 | $(c_{22}, c_{23}, c_{24})$ | $(\widetilde{c}_{22}, \widetilde{c}_{23}, \widetilde{c}_{24}) = (\lfloor c_{22\cdot 2} \rfloor, c_{23\cdot 2}, c_{24\cdot 2})$ |
| 2 | **0** | 22 | $(\lfloor c_{44} \rfloor, c_{45}, c_{46})$ | $(\widetilde{c}_{44}, \widetilde{c}_{45}, \widetilde{c}_{46}) = (c_{44\cdot 2}, c_{45\cdot 2}, c_{46\cdot 2})$ |
| 1 | 1 | 45 | $(c_{90}, c_{91}, c_{92})$ | $(\widetilde{c}_{90}, \widetilde{c}_{91}, \widetilde{c}_{92}) = (\lfloor c_{90\cdot 2} \rfloor, c_{91\cdot 2}, c_{92\cdot 2})$ |
| 0 | **0** | 90 | $(\lfloor c_{180} \rfloor, c_{181}, c_{182})$ | $(\widetilde{c}_{180}, \widetilde{c}_{181}, \widetilde{c}_{182}) = (c_{180\cdot 2}, c_{181\cdot 2}, c_{182\cdot 2})$ |

From the table 3, we can see that $XTRDBL$ operation at $j = 5, 2$, and 0 to update $C[0]$ in the computation $S_{181}$ is the same as the $XTRDBL$ operation at $6, 3$, and 1 to update $C[0]$ in the computation $\widetilde{S}_{181}$, respectively.

In XTR-SE, we can easily derive the following property.

*Property 2.* $C[0]$ is updated as $c_{2k\cdot 1}$ (or $c_{2k\cdot v}$) in $S_n$ (or $\widetilde{S}_n$ when $\widetilde{c}_1 = c_v$). If $m_i = 0$ then $k_i = 2 \cdot k_{i-1}$, where $k_i$ denotes the value of $k$ when index $j = i$. Thus if $v = 2$ and $m_i = 0$ then $c_{2k_i\cdot 1} = c_{2k_{i-1}\cdot 2}$. If $m_i = 1$ then $k_i = 2 \cdot k_{i-1} + 1$. Thus if $v = 2$ and $m_i = 1$ then $c_{2k_i\cdot 1} \neq c_{2k_{i-1}\cdot 2}$.

From the above property, $XTRDBL$ operation at rank $j$ to update $C[0]$ in the computation $S_n$ is the same as the $XTRDBL$ operation at rank $j + 1$ to update $C[0]$ in the computation $\widetilde{S}_n$ (when $\widetilde{c}_1 = c_2$) if and only if $m_j = 0$.

Therefore, with only two requests to the device, it is possible to recover all the bits of the secret value.

*Remark 3.* The doubling attack defeats two of the three countermeasures proposed by Coron [Cor99], which are randomization of the private exponent and randomization of the base element.

## 4 Countermeasures against the Proposed Attacks

### 4.1 Countermeasures against Data-bit DPA

Many countermeasures against side channel attacks have been proposed. Okeya et al. classified them into several types such as fixed procedure type, randomized addition chains type, indistinguishable operations type, data randomization type, and so on [OT03]. Especially, to resist against DDPA randomized exponent methods contained in randomized addition chains type and the data randomization type are used.

**Randomization of the Base Element Using Field Isomorphism** To randomize computing objects, we use field isomorphism. As $p \equiv 2 \ mod\ 3$, the zeros $\alpha$ and $\alpha^p$ of the polynomial $(X^3-1)/(X-1) = X^2+X+1$ form an optimal normal basis for $GF(p^2)$ over $GF(p)$. An element $x \in GF(p^2)$ is represented as $x_1\alpha + x_2\alpha^2$ with $x_1, x_2 \in GF(p)$. Namely, $x \in GF(p)[X]/(X^2+X+1) \cong GF(p^2)$.

As we know, there is one and only one finite field $GF(p^2)$ up to isomorphism. So, if we find another quadratic monic irreducible polynomial $X^2 + a_1X + a_0$ over $GF(p)$ then we can construct $GF(p)[X]/(X^2 + a_1X + a_0)$ isomorphic to $GF(p^2)$. Thus we obtain another representation for the element $x$ using the roots of $X^2 + a_1X + a_0$.

The field isomorphism method is described as follows:

- **Goal** : Compute $S_n$ from given $c$ and $n$.
  Note that $c$ is represented as the element of $GF(p)[X]/(X^2+X+1)$.
- Step 1 : Choose randomly a quadratic monic irreducible polynomial $X^2 + a_1X + a_0$ over $GF(p)$.
  Let $\phi$ denote an isomorphism from $GF(p)[X]/(X^2 + X + 1)$ to $GF(p)[X]/(X^2 + a_1X + a_0)$.
- Step 2 : Represent $c$ as an element $\phi(c) \in GF(p)[X]/(X^2 + a_1X + a_0)$. Let $c' := \phi(c)$. Note that in this case basis conversion is needed.
- Step 3 : Compute $S'_n :=$XTR-SE$(c', n) = (c'_{n-1}, c'_n, c'_{n+1})$.
- Step 4 : Go back to the original representation by representing $S'_n$ as an element

$$S_n = (\phi^{-1}(c'_{n-1}), \phi^{-1}(c'_n), \phi^{-1}(c'_{n+1})) \in (GF(p)[X]/(X^2+X+1))^3.$$

**Efficiency of the countermeasure :** The efficiency of the countermeasure depends on the choice of the irreducible polynomial $X^2 + a_1X + a_0$ and basis. To speed up XTR-SE $x^2, xy, x^p, xz - yz^p$ for $x, y, z \in GF(p^2)$ should be efficient because these operations play an important role in XTR-SE. Table 4 shows the

efficiency of basic operations in XTR and $S_n$ for XTR [SL01] using optimal normal basis type I and XTR using random quadratic monic irreducible polynomial with polynomial basis. Note that the numbers in Table 4 denote the required number of multiplications in $GF(p)$.

**Table 4.** The costs of the basic operations in XTR.

| | $x^p$ | $x^2$ | $x \cdot y$ | $xz - yz^p$ | Basis | Cost of $S_n$ |
|---|---|---|---|---|---|---|
| XTR [SL01] | free | 2 | 2.5 | 3 | Optimal normal basis type I | $7 \log_2 n$ |
| XTR | $1.3 \log_2 p$ | 4.6 | 6 | $12+1.3\log_2 p$ | Polynomial basis | $(21.2 + 5.2 \log_2 p) \cdot \log_2 n$ |

*Remark 4.* In step 2 and 4, basis conversions are needed. In general, the cost of basis conversion is not negligible because square root calculation is required in the case of XTR. For instance, in step 2, to represent $c \in GF(p)[X]/(X^2+X+1)$ with normal basis as an element of $GF(p)[X]/(X^2+a_1X+a_0)$ with polynomial basis a root of $X^2+X+1$ should be represented with respect to normal basis of $GF(p)[X]/(X^2+a_1X+a_0)$. At that case, square root calculation is required. If two field bases which are changed between themselves are fixed and a root of $X^2+X+1$ is represented with respect to basis of $GF(p)[X]/(X^2+a_1X+a_0)$ then we could use the novel basis conversion method proposed by Kaliski and Yin [KY98]. But their method can not be directly applicable to the countermeasure using field isomorphism because in step 1 a quadratic irreducible polynomial $X^2+a_1X+a_0$ is randomly chosen. So, whenever a quadratic irreducible polynomial is randomly chosen we should do square root calculation to represent a root of $X^2+X+1$ with respect to basis of the randomly chosen field $GF(p)[X]/(X^2+a_1X+a_0)$.

*Remark 5.* In Table 4 we assume that squaring takes 80% of the complexity taken for multiplication in $GF(p)$. The result of the second low in the above table could be changed depending on the choice of basis and multiplication (squaring) method. However, if the basic operations such as $x^2, xy, x^p, xz - yz^p$ are overlooked in the construction of $GF(p^2)$ there is not any more advantage of speed on XTR. For example, when $p$ is 170-bit prime the cost $S_n$ of second low is 129 times slower than that of the first low in the above table.

*Remark 6.* The field isomorphism method is originally proposed by Joye and Tymen to protect against DDPA for elliptic curve cryptography [JT01].

**Randomization of the Private Exponent** The randomized exponent methods [Cor99] is well known countermeasure against DDPA.

The computation of $S_n$ =XTR-SE$(c, n)$ is done by the following algorithm:

1. Select a random number $r$.
2. Compute $n' = n + r \cdot q$.

3. Compute $S_n = \text{XTR-SE}(c, n')$. Note that
   $\text{XTR-SE}(c, n') = (c_{n'-1}, c_{n'}, c_{n'+1}) = (Tr(g^{n'-1}), Tr(g^{n'}), Tr(g^{n'+1})) \overset{(*)}{=} (Tr(g^{n-1}), Tr(g^n), Tr(g^{n+1})) = \text{XTR-SE}(c, n)$, where $c = Tr(g)$. $(*)$ : as order of $g$ is prime $q$.

**Efficiency of the countermeasure :** The effective key length may increase depending on the bit length of $r \cdot q$. In general, the recommended bit length of $r$ is over 20 bits. Thus required computing time become at least 1.2 times than that of without countermeasure.

### 4.2 Countermeasure against Address-bit DPA

Address-bit DPA is based on the relation between a secret value and addresses of the accessed registers. In order to resist ADPA, this relation should be hidden.

We use random number $r_{l-1}2^{l-1} + \cdots + r_1 2 + r_0$ where $r_i \in \{0, 1\}$. Define $[a]_3$ denote remainder of $a$ modulo 3 for any $a \in Z$. For example, $[5]_3 = 2$.

**Goal :** Remove the relations described in Property 1.
If we could obtain following relations in XTR-SE, ADPA is infeasible to XTR-SE.

- When $m_j = 0$ and $r_j = 0$,
  - To update $T[1]$ read address $C[0]$.
  - To update $T[2]$ read address $C[1]$.
- When $m_j = 0$ and $r_j = 1$,
  - To update $T[1]$ read address $C[1]$.
  - To update $T[2]$ read address $C[2]$.
- When $m_j = 1$ and $r_j = 0$,
  - To update $T[1]$ read address $C[1]$.
  - To update $T[2]$ read address $C[2]$.
- When $m_j = 1$ and $r_j = 1$,
  - To update $T[1]$ read address $C[0]$.
  - To update $T[2]$ read address $C[1]$.

We propose following XTR-SE with countermeasure against ADPA.

**Efficiency of the countermeasure :** The proposed countermeasure has almost no overhead for the protection, i.e., the processing speed is no slower than that without the countermeasure.

### 4.3 Countermeasure against Doubling Attack

Since no attack as efficient as the doubling attack is known on the upward double-and-add (square-and-multiply) algorithm from the least to the most significant bit in ECC (RSA), this routine is recommended to the countermeasure against doubling attack.

But there is no upward algorithm in XTR. In the case of XTR, the method of randomization of the base element by using field isomorphism proposed at Section 4.1 could be used to break doubling attack.

XTR-SE with countermeasure against ADPA

INPUT : $c$ and $n$ where $n > 2$
OUTPUT : $S_n$

1. Compute initial values:
   1.1. $C[3] \leftarrow c$,
   $C[[r_{l-1}(1+m_{l-1})]_3] \leftarrow XTRDBL(C[3])$,
   $C[[1+r_{l-1}(1+m_{l-1})]_3] \leftarrow XTR\_C_{2n+1}(3, C[3], C[[1+r_{l-1}(1+m_{l-1})]_3], C[3])$,
   and $C[[2+r_{l-1}(1+m_{l-1})]_3] \leftarrow XTRDBL(C[[r_{l-1}(1+m_{l-1})]_3])$.
   1.2. If $n$ is even, $n$ replace $n-1$.
   Let $n = 2m+1$ and $m = \sum_{j=0}^{l} m_j 2^j$ with $m_j \in \{0,1\}$ and $m_l = 1$.
2. for $j = l-1$ downto 0
   2.1. $T[1] \leftarrow XTRDBL(C[[m_j + r_j(1+m_j)]_3])$
   2.2. $T[2] \leftarrow XTRDBL(C[[1+m_j + r_j(1+m_j)]_3])$
   2.3. if $(m_j = 0)$ then
   $T[3] \leftarrow XTR\_C_{2n-1}(C[[r_j(1+m_j)]_3], C[[1+r_j(1+m_j)]_3], C[[2+r_j(1+m_j)]_3], C[3])$
   if $(m_j = 1)$ then
   $T[3] \leftarrow XTR\_C_{2n+1}(C[[r_j(1+m_j)]_3], C[[1+r_j(1+m_j)]_3], C[[2+r_j(1+m_j)]_3], C[3])$
   If $(j = 0)$ go to step 3.
   2.4. $C[[r_{j-1}(1+m_{j-1})]_3] \leftarrow T[1]$
   2.5. $C[[1+r_{j-1}(1+m_{j-1})]_3] \leftarrow T[3]$
   2.6. $C[[2+r_{j-1}(1+m_{j-1})]_3] \leftarrow T[2]$
3. Compute $C[0] \leftarrow XTR\_C_{n+2}(T[1], T[2], T[3], C[3])$.
   If $n$ is odd then return $(T[1], T[3], T[2])$,
   else return $(T[3], T[2], C[0])$.

*Remark 7.* As previously remarked the countermeasure using field isomorphism is not efficient. Construction of an efficient countermeasure against doubling attack is an open question.

### 4.4 Combining Countermeasures

In this section, we only deal with SPA, DDPA, ADPA, and doubling attack described in the Section 3 as the attack algorithm for XTR-SE. As we saw in the previous section, some all countermeasures only resist specific attacks, for example the randomized exponent method is good countermeasure against DDPA but it could be broken by the doubling attack [FV03]. Thus we should combine them to resist the referred SCA, namely XTR-SE should be added following countermeasures: randomization of the base element using field isomorphism (DDPA+doubling attack) + randomized addressing (ADPA). Note that XTR-SE does not need a countermeasure against SPA.

## 5 Comparison among XTR, ECC, and RSA

In this section we compare XTR to ECC and RSA with countermeasures against SCA, such as SPA, DDPA, ADPA, and doubling attack.

**In the case of ECC :** Itoh et al. [IIT03] recommended the best combination countermeasures against SPA, DDPA, and ADPA from the security level and processing speed: binary method (from MSB or LSB)+ double-and-add-always method (SPA) + randomized projective coordinate (or randomized curve) (DDPA) + randomized addressing (ADPA). To defeat doubling attack upward binary method could be used. Thus in ECC, upward binary method (i.e., from LSB) (doubling attack) + double-and-add-always method (SPA) + randomized projective coordinate (or randomized curve) (DDPA) + randomized addressing (ADPA) are needed to resist against SCA.

Note that above recommended combination of countermeasures is not secure against Goubin's attack [Gou03]. Thus if Goubin's attack [Gou03] is considered ECC needs extra countermeasure such as point blinding or randomization of the private exponent. Goubin's attack [Gou03], however, could not be applicable to XTR and RSA.

**In the case of RSA :** upward binary method (doubling attack) + square-and-multiply-always method (SPA) + randomization of the private exponent (DDPA) + randomized addressing (ADPA) are needed to resist against SCA.

**In the case of XTR :** XTR-SE + randomization of the base element using field isomorphism (DDPA and doubling attack) + randomized addressing (ADPA) are needed to resist against SCA. Note that XTR-SE does not need a countermeasure against SPA.

**Efficiency :** If the side channel attack is not considered, XTR is faster than 170-bit ECC and 1020-bit RSA in the signing (decrypting) step [LV00a].

We roughly compare the efficiency among ECC (170-bit), RSA (1020-bit), and XTR (170-bit) with the countermeasures against SCA. Note that in the case of ECC we also consider Goubin's attack. We ignore the cost for randomization or transformations required in the countermeasures such as randomized projective coordinate and randomized addressing because they are relatively small compared to basic operations, for instance, point addition or 1020-bit multiplication.

- **ECC :** 87 point additions and 2 point doublings are additionally required in the combination of countermeasures.
  - 85 (= 170/2) point additions are additionally required in double-and-add-always method.
  - 2 point additions and 2 point doublings are additionally required in point blinding (countermeasure against Goubin's attack).

  Thus required computing time become at least 1.39 times than that of scalar multiplication without SCA countermeasures.
- **RSA :** 530 1020-bit multiplications and 20 squarings are additionally required in the combination of countermeasures.
  - 510 1020-bit multiplications are additionally required in square-and-multiply-always method.

- 20 1020-bit multiplications and 20 squarings are additionally required in randomization of the private exponent if the bit length of random number $r$ is 20 bits.

Thus required computing time become at least 1.41 times than that of single exponentiation without SCA countermeasures.

– **XTR :** The efficiency of XTR-SE with SCA countermeasures is at least 129 times slower (without considering the cost of basis conversion) than that of XTR-SE without that.

Thus the comparison of efficiency among XTR, ECC, and RSA with SCA countermeasures may be meaningless if there is no efficient countermeasure against doubling attack.

## 6 Conclusion and Remark

In this paper, we investigate the security of side channel attack on XTR. In [LV00a,SL01] authors remarked that XTR-SE is less susceptible than usual exponentiation routines to environmental attacks such as timing attacks and Differential Power Analysis (DPA).

However we showed that XTR-SE is vulnerable to DDPA [Cor99], ADPA [IIT02], and doubling attack [FV03]. XTR-SE is, however, secure against SPA if the order of computation is carefully considered.

Moreover, We proposed two countermeasures that prevent from DDPA and a countermeasure against ADPA. One of the countermeasures using randomization of the base element proposed to defeat DDPA, i.e., randomization of the base element using field isomorphism, could be used to break doubling attack. If we only deal with SPA, DDPA, ADPA, and doubling attack as the attack algorithm for XTR-SE, XTR-SE should be added following countermeasures: randomization of the base element using field isomorphism (DDPA and doubling attack) + randomized addressing (ADPA).

As the countermeasure against doubling attack is very inefficient, the efficiency of XTR-SE with SCA countermeasures could not be comparable with that of ECC or RSA with SCA countermeasures. To maintain the advantage of efficiency of XTR a good countermeasure against doubling attack is actually necessary. Note that construction of an efficient countermeasure against doubling attack is an open question.

As a final conclusion, we hope that this first step towards side channel attack on XTR will be a motivating starting point for further research.

## References

[Cor99] Coron, J.S., *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*, Cryptographic Hardware and Embedded Systems (CHES'99), LNCS1717, (1999), 292-302.

[CMO98] Cohen, H., Miyaji, A., Ono, T., *Efficient elliptic curve exponentiatiion using mixed coordinates*, Proceedings of Asiacrypt 1998, LNCS1514, (1998), 51-65.
`http://www.ecstr.com`

[FV03] Fouque, P.-A., Valette, F., *The Doubling Attack Why Upwards is better than Downwards*, Workshop on Cryptographic Hardware and Embedded Systems 2003 (CHES 2003), LNCS 2779, (2003), 269-280.

[Gou03] Goubin, L., *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*, Public Key Cryptography, (PKC 2003), LNCS 2567, (2003), 199-211.

[IIT02] Itoh, K., Izu, T., Takenaka, M., *Address-bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA*, Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002), LNCS 2523, (2002), 129-143.

[IIT03] Itoh, K., Izu, T., Takenaka, M., *A Practical Countermeasure against Address-bit Differential Power Analysis*, Workshop on Cryptographic Hardware and Embedded Systems 2003 (CHES 2003), LNCS 2779, (2003), 382-396.

[JT01] Joye, M., Tymen, C., *Protections against differential analysis for elliptic curve cryptography: An algebraic approach*, Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 377-390.

[Koc96] Kocher, C., *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology - CRYPTO '96, LNCS 1109, (1996), 104-113.

[KJJ98] Kocher, P., Jaffe, J., Jun, B., *Introduction to Differential Power Analysis and Related Attacks*, 1998.
`http://www.cryptography.com/dpa/technical.`

[KJJ99] Kocher, C., Jaffe, J., Jun, B., *Differential Power Analysis*, Advances in Cryptology - CRYPTO '99, LNCS1666, (1999), 388-397.

[KY98] Kaliski, B.S., Yin, Y.L., *Storage-Efficient Finite Field Basis Conversion*, Proceedings of SAC 98, LNCS1556 , (1998), 81-93.

[L97] Lenstra, A.K., *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, The 2th Australasian Conference in Information Security and Privacy, (ACISP 1997), LNCS1270, (1997), 127-138.

[LV00a] Lenstra, A.K., Verheul, E.R., *The XTR public key system*, Advances in Cryptology - CRYPTO '00, LNCS1880, (2000), 1-19.
`http://www.ecstr.com`

[LV00b] Lenstra, A.K., Verheul, E.R., *Key improvements to XTR*, Proceedings of Asiacrypt 2000, LNCS1976, (2000), 220-233.
`http://www.ecstr.com`

[LV01] Lenstra, A.K., Verheul, E.R., *Fast irreducibility and subgroup membership testing in XTR*, Public Key Cryptography, (PKC 2001), LNCS 1992, (2001), 73-86.
`http://www.ecstr.com`

[M93] Menezes, A.J., *Applications of Finite Fields*, Waterloo, 1993.

[MDS99] Messerges, T., Dabbish, E., Sloan, R., *Investigations of Power Analysis Attacks on Smartcards*, preprint, USENIX Workshop on Smartcard Technology, 1999.

[OS00] Okeya, K., Sakurai, K., *Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack*, Progress in Cryptology - INDOCRYPT 2000, LNCS1977, (2000), 178-190.

[OT03] Okeya, K., Takagi, T., *The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks*, Topics in Cryptology, The Cryptographers' Track at the RSA Conference 2003 (CT-RSA 2003), LNCS2612, (2003), 328-342.

[SL01] Stam, M., Lenstra, A.K., *Speeding Up XTR*, Proceedings of Asiacrypt 2001, LNCS2248, (2001), 125-143.
`http://www.ecstr.com`