Collision Attack on XTR and a Countermeasure with a Fixed Pattern ⋆

Dong-Guk Han[1], Tsuyoshi Takagi[1], Tae Hyun Kim[2],
Ho Won Kim[3], and Kyo Il Chung[3]

[1] FUTURE UNIVERSITY-HAKODATE, JAPAN
{christa,takagi}@fun.ac.jp
[2] Center for Information and Security Technologies(CIST),
Korea University, Seoul, KOREA
thkim@cist.korea.ac.kr
[3] Electronics and Telecommunications Research Institute(ETRI), KOREA
{khw,kyoil}@etri.re.kr

Abstract. Public-key cryptosystem (PKC) is one of inevitable key technologies in order to accomplish fruitful security applications in ubiquitous computing systems. The ubiquitous computer only has scarce computational resources (like Smart cards, RFID, Sensor Network), however, so that the light weight PKC is necessary for those miniaturized low-power devices. Recently, XTR is considered as one of good candidates for more energy efficient cryptosystems. Among XTR exponentiation algorithms, the most efficient one is the Improved XTR Single Exponentiation (XTR-ISE) proposed by Stam-Lenstra. Thus among the family of XTR algorithms, XTR-ISE is the most efficient one suitable for ubiquitous computer. Even though the security of such devices against side channel attacks is very dangerous, there are few works on side channel attacks against XTR-ISE. In this paper we propose a new collision attack on XTR-ISE, derived from the structural properties of XTR-ISE. The analysis complexity of the proposed one is about 2^{40} where the key size is 160-bit, which is 55% improvement from the previously best known analysis of Page-Stam. We also propose a novel countermeasure using a fixed pattern which is secure against SPA. We deploy a variant of Euclidean algorithm whose one of the registers is a monotone decreasing function with odd value. From our estimation of the efficiency of the proposed method, XTR exponentiation, computing $Tr(g^n)$ with $Tr(g)$ and n, takes $11.2 \log_2 n$ multiplications in $\mathbf{F}_{p^2}$. In the sense of both efficiency and security the proposed countermeasure is the best one among the previous countermeasures- it is about 30% faster.

Keywords: *Ubiquitous computer, XTR public key system, XTR Exponentiation Algorithms, Side Channel Attacks, Collision Attack*

1 Introduction

We are standing to the beginning of the ubiquitous computing era. It is expected that we can accomplish lucrative applications by effectively synthesizing the ubiquitous computer with cryptography. The ubiquitous computer only has scarce computational resources (like Smart cards, RFID, Sensor Network), so that we have to make an effort to optimize the memory and efficiency of the security system. Our expectation is that secure symmetric encryption will be widely available on the ubiquitous computer of the future, but one of the biggest problems in using secret key algorithms is the protection of the sensitive key material. However, the use of public-key cryptosystem (PKC) facilitates security protocols and has a potential impact on a much wider range of applications.

⋆ This is a "full" version of a paper that will be published in SecUbiq 2005.

Furthermore, only the public key would have to be embedded into the target devices. Currently there are a few implementations on ubiquitous environments with PKC. In ESAS 2004 Gaubatz-Kaps-Sunar showed an implementation of Rabin and Ntru in sensor networks [8]. Recently Watro et al. showed RSA (in the case the encryption key is 3) is feasible to the applications of ubiquitous computer, and remarked that XTR is one of good candidates for light weight cryptosystems in SASN 2004 [21].

However, the applications of ubiquitous computer will be carried into and used in hostile environments and often house sensitive information, for example identity related tokens or financial information, the threat of attack is significant. This threat is magnified by both the potential pay-off and level of anonymity that side channel attacks (SCA) allow [12, 13]. The fact that one can attack a device somewhat remotely via timing and power consumption means that most ubiquitous computing devices need to be aware of similar problems in their operational environments.

In Crypto 2000 Lenstra-Verheul introduced XTR, a cryptosystem using a sub-group of the multiplicative group of $\mathbf{F}_{p^6}$ but with a compact representation based on the trace over $\mathbf{F}_{p^2}$ that allows highly efficient arithmetic [14]. In Crypto 2003, Rubin-Silverberg proposed torus based public key cryptosystem CEILIDH to provide greater efficiency for the same security [17]. Recently, Dijk et al. proposed an optimal communication technique for torus-based cryptosystem [6, 5]. The common main idea of XTR and CEILIDH is to shorten the bandwidth of transmission data. Even though the efficiency of communication of CEILIDH is better than XTR, unfortunately it was shown that CEILIDH seems bound to be inherently slower than XTR [9]. Given the current state of affairs in breaking the discrete logarithm problems over either finite fields or elliptic curves, XTR can compete with elliptic curve cryptosystems (ECC) in terms of both speed and bandwidth. This makes XTR suitable for deployment on similar sorts of constrained devices such as smart-cards, where computational power and storage capacity are both very limited. Among XTR exponentiation algorithms, the most efficient one is proposed in [18], called as Improved XTR Single Exponentiation (XTR-ISE), and that is on average more than 22% faster than the old method. Thus among the family of XTR algorithms, XTR-ISE is the most efficient one suitable for smart-cards, where computational power and memory capacity are both very limited. Even though the security of such devices against side channel attacks is very dangerous, however, there are few works on side channel attacks against XTR-ISE.

In 2004 Chung-Hasan [2] and Page-Stam [16] proposed simple power analysis (SPA) against XTR-ISE and that it was the first try to analyze it with SCA. Chung-Hasan showed it takes 2^{100} tries for an attacker until he/she correctly finds the secret key in XTR-ISE with 160-bits key length. On the other hand, Page-Stam showed it requires 2^{88} tries. It's more nice result than that of Chung-Hasan, but these results are far worse than well-known square-root type algorithms (Baby-Step-Giant-Step or Pollards' Rho methods), i.e., their results are not practically feasible. Page-Stam introduced the indistinguishable arithmetic formula and the exponent splitting method as SPA and DPA countermeasure. It is considered as the most efficient countermeasure against SPA and DPA among the proposed ones in XTR family.

1.1 Contribution of this paper

From the above previous results about XTR-ISE, we are encouraged to start the following two challenges;

(1) **Analysis -** *How can we reduce the complexity of analysis:* We can see that the analysis result of Page-Stam and Chung-Hasan are not practically feasible. In this paper we find a new analysis technique, called as XTR collision attack, derived from the structural properties of XTR-ISE. The complexity of XTR collision attack is about $2^{0.25 \cdot l}$

where l is the length of the key. Thus the complexity of XTR collision attack against XTR-ISE is about 2^{40} where the key length is 160-bit, which is about 55% improvement from the result of Page-Stam [16].

(2) **Countermeasure -** *How can we design a secure countermeasure against SCA:* In the countermeasure of Page-Stam, the indistinguishable arithmetic formula and exponent splitting as a SPA and DPA countermeasure respectively, there are some controversial points.

- Recently Walter showed that the produced unified code for elliptic addition and doubling in order to avoid SPA may still be insecure against SPA if there is sufficient side channel leakage at lower levels [20]. Thus the indistinguishable arithmetic technique is not recommended as a SPA countermeasure.
- If Montgomery arithmetic is used in the proposed indistinguishable arithmetic formula then some extra dummy additions have to added to XTR addition routine to make up for this [16]. However, Yen et al. showed that the safe-error attack is applicable to the dummy method [22].

Thus in this paper we propose a novel countermeasure using a fixed pattern which is secure against SPA. As the behavior of XTR-ISE is based on an adaptation of a Euclidean Algorithm we propose a special Euclidean algorithm such that one parameter is always odd integer and monotone decreasing to construct a fixed pattern of XTR addition **A** and doubling **D**. In deed we generate an XTR *AD* sequence with the fixed pattern such that **ADDADD**...**ADD**. In order to defeat DPA the exponent splitting technique is utilized. From our estimation of the efficiency of the proposed method, XTR exponentiation, computing $Tr(g^n)$ with $Tr(g)$ and n, takes $11.2 \log_2 n$ multiplications in $\mathbf{F}_{p^2}$. In the sense of both efficiency and security the proposed countermeasure is the best one among the previous countermeasures- it is about 30% faster.

The remainder of the paper is structured as follows. After a brief description of XTR public key systems in Section 2, and an introduction into side channel attacks on XTR in Section 3, we will propose a new analysis technique, called as XTR collision attack in Section 4. In Section 5 we explain the proposed countermeasure using a fixed pattern which is secure against SPA.

2 XTR Public Key Cryptosystems

In this section, we review mathematics of XTR including basic parameters and fundamental algorithms to calculate traces of powers [14, 18].

2.1 XTR Parameters

Let p and q be primes with $p \equiv 2 \bmod 3$ and q dividing $p^2 - p + 1$, and let g be a generator of the order q subgroup of $\mathbf{F}^*_{p^6}$. Suggested lengths to provide adequate levels of security are $\log_2 q \approx 160$ and $\log_2 p \approx 170$. As p is 2 modulo 3, it follows that $(X^3-1)/(X-1) = X^2+X+1$ is irreducible over $\mathbf{F}_p$ and that two roots α and α^p form an optimal normal basis for $\mathbf{F}_{p^2}$ over $\mathbf{F}_p$, i.e., $\mathbf{F}_{p^2} \cong \{x_1\alpha + x_2\alpha^p : x_1, x_2 \in \mathbf{F}_p\}$. Since $\alpha^i = \alpha^{i \bmod 3}$ it follows that

$$\mathbf{F}_{p^2} \cong \{x_1\alpha + x_2\alpha^2 : \alpha^2 + \alpha + 1 = 0 \text{ and } x_1, x_2 \in \mathbf{F}_p\}.$$

For the simplicity, we denote $x = x_1\alpha + x_2\alpha^2$ as (x_1, x_2).

Property 1. As $\alpha^2 = \alpha^p$ it follows that $x^p = x_2\alpha + x_1\alpha^2$, so that p-th powering in $\mathbf{F}_{p^2}$ is free.

For an element $h \in \mathbf{F}_{p^6}^*$ its trace $\mathrm{Tr}(h)$ over $\mathbf{F}_{p^2}$ is defined as a sum of the conjugates over $\mathbf{F}_{p^2}$ of h:

$$\mathrm{Tr}(h) = h + h^{p^2} + h^{p^4} \in \mathbf{F}_{p^2}.$$

2.2 XTR-ElGamal encryption

XTR can be used in any cryptosystem that relies on the discrete logarithm problem. This section contains a description of an application of XTR to ElGamal encryption [14].

Public key data of Alice: p, q, $\mathrm{Tr}(g)$, $\mathrm{Tr}(g^k)$.
Secret key data of Alice: k in $[2, q-3]$.

Encryption Bob can encryption a message $M \in \mathbf{F}_{p^2}$ intended for Alice as follows.

1. Select at random $r \in [2, q-3]$ and compute $\mathrm{Tr}(g^r) \in \mathbf{F}_{p^2}$.
2. Compute $\mathrm{Tr}(g^{rk}) \in \mathbf{F}_{p^2}$ with r and $\mathrm{Tr}(g^k)$. Let $K = \mathrm{Tr}(g^{rn'})$.
3. Compute $E = K \cdot M \in \mathbf{F}_{p^2}$.
4. Send $(Tr(g^r), E)$ to Alice.

Decryption Using her knowledge of secret key k, Alice decrypts the message $(\mathrm{Tr}(g^r), E)$ as follows.

1. Compute $\mathrm{Tr}(g^{kr}) \in \mathbf{F}_{p^2}$ with k and $Tr(g^r)$.
2. Let $K = \mathrm{Tr}(g^{kr})$ and find K^{-1} such that $K \cdot K^{-1} = 1$ in $\mathbf{F}_{p^2}$.
3. Compute $E \cdot K^{-1}$, resulting in message M.

2.3 XTR Exponentiation Algorithm

Throughout this paper, c_n denotes $\mathrm{Tr}(g^n) \in \mathbf{F}_{p^2}$, for some fixed p and g of order q, where q divides $p^2 - p + 1$. Note that $c_0 = 3$ and $c_1 = c$.

An efficient computation of c_n for given p, q and c depends on the recurrence relations

$$c_{u+v} = c_u c_v - c_v^p c_{u-v} + c_{u-2v}, \tag{1}$$

and

$$c_{2u} = c_u^2 - 2c_u^p, \tag{2}$$

which is derived from the equation (1) when $u = v$.

By using above two formula, we define the following two functions called as XTR addition and XTR doubling respectively;

$$A[x, y, z, w] = x \cdot y - y^p \cdot z + w,$$
$$D[x] = x^2 - 2x^p.$$

Computation time of the basic operations: Let Mul denote the computation time of modulo multiplication in $\mathbf{F}_p$ and $x = (x_1, x_2), y = (y_1, y_2), z = (z_1, z_2)$, and $w = (w_1, w_2)$ be elements of $\mathbf{F}_{p^2}$.

The basic operations $D[x]$ and $A[x, y, z, w]$ in XTR can be calculated like this [14].

$$D[x] = (x_2(x_2 - 2x_1 - 2),\ x_1(x_1 - 2x_2 - 2)), \tag{3}$$
$$A[x, y, z, w] = (y_1 \cdot T_1 + y_2 \cdot T_2 + w_1,\ y_1 \cdot T_3 + y_2 \cdot T_4 + w_2), \tag{4}$$

where $T_1 = z_1 - x_2 - z_2, T_2 = x_2 - x_1 + z_2, T_3 = x_1 - x_2 + z_1$, and $T_4 = z_2 - x_1 - z_1$.

Thus the required number of multiplications to compute $D[\cdot]$ and $A[\cdot]$ is as follows;

- $2 \cdot Mul$ is required to compute $D[\cdot]$,
- $4 \cdot Mul$ is required to compute $A[\cdot]$.

Remark 1. As usual, the small number of additions and subtractions is not counted because the computation time of that is negligible [14].

By using above defined notations we introduce Improved XTR exponentiation algorithms proposed by Stam-Lenstra [18]. The goal of these algorithms is to compute c_n for given c_1 and $n \in Z$, i.e. to compute $Tr(g^n)$ with $Tr(g)$ and an integer n.

Improved XTR Single Exponentiation (XTR-ISE) [18]

Input: c_1 and n where $n > 2$
Output: c_n

---

1. Initialization:
   1.1. Let $a = \text{round}(\frac{3-\sqrt{5}}{2}n)$ and $b = n - a$ (where round(x) is the integer closest to x).
   1.2. Let $f = 0$. As long as a and b are both even, replace (a, b) by $(a/2, b/2)$ and f by $f + 1$.
   1.3. Let $i = 1$ and $G_i := (Q_0, Q_1, Q_2, Q_3) = (c_1, c_1, 3, c_1^p)$.
2. As long as $a \neq b$
   2.1. If $b > a$
   - X_1. if $b \leq 4a$, then $(a, b) \leftarrow (b - a, a)$
     $T_0 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$, $T_1 \leftarrow Q_0$,
     $T_2 \leftarrow Q_1$, $T_3 \leftarrow Q_2^p$.
   - X_2. else if b is even, then $(a, b) \leftarrow (a, b/2)$
     $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
     $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$.
   - X_3. else if a is odd, then $(a, b) \leftarrow (a, (b - a)/2)$
     $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
     $T_2 \leftarrow Q_2$, $T_3 \leftarrow D[Q_1]^p$.
   - X_4. else (a is even), then $(a, b) \leftarrow (b, a/2)$
     $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow Q_0$,
     $T_2 \leftarrow Q_3^p$, $T_3 \leftarrow D[Q_2]^p$.

   2.2. Else (if $a > b$)
   - Y_1. if $a \leq 4b$, then $(a, b) \leftarrow (a - b, b)$
     $T_0 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$, $T_1 \leftarrow Q_1$,
     $T_2 \leftarrow Q_0$, $T_3 \leftarrow Q_2$.
   - Y_2. else if a is even, then $(a, b) \leftarrow (b, a/2)$
     $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow Q_0$,
     $T_2 \leftarrow Q_3^p$, $T_3 \leftarrow D[Q_2]^p$.
   - Y_3. else if b is odd, then $(a, b) \leftarrow (b, (a - b)/2)$
     $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
     $T_2 \leftarrow Q_2^p$, $T_3 \leftarrow D[Q_0]^p$.
   - Y_4. else (b is even), then $(a, b) \leftarrow (a, b/2)$
     $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
     $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$

   2.3. $i \leftarrow i + 1$ and set $G_i = (T_0, T_1, T_2, T_3)$.
3. Compute $\tilde{c} = A[Q_0, Q_1, Q_2, Q_3] = c_{u+v}$.
4. Output $\tilde{c}_{2^f}$.
5. If $a = 1$ then return $\tilde{c}_{2^f}$
   else run Improved XTR Single Exponentiation with $c = \tilde{c}_{2^f}$ and $n = a$.

Let $G_i := (Q_0, Q_1, Q_2, Q_3)$ be the i-th updated intermediate values of Q_i in Step 2 of XTR-ISE for $i \geq 1$. $G_1 = (Q_0, Q_1, Q_2, Q_3)$ denotes the updated values at the initialization step. Let $G_i \xrightarrow{T_i} G_{i+1}$ denote that G_{i+1} is updated from G_i after state $T_i \in \{X_j, Y_j\}_{1 \leq j \leq 4}$. For example, if XTR-ISE is terminated after $X_1 \to X_1 \to X_4 \to Y_4 \to Y_3 \to X_2 \to Y_1$, then the relation between G_i and T_i is as follows; $G_1 \stackrel{X_1}{\to} G_2 \stackrel{X_1}{\to} G_3 \stackrel{X_4}{\to} G_4 \stackrel{Y_4}{\to} G_5 \stackrel{Y_3}{\to} G_6 \stackrel{X_2}{\to} G_7 \stackrel{Y_1}{\to} G_8$.

XTR-ISE is based on an adaptation of a Euclidean algorithm by Montgomery using Lucas chains. For ease of notation, we will momentarily use ordinary exponentiation in our description instead of the third order XTR recurrence. Given $Tr(g)$ and n, computing $Tr(g^n)$ takes $6.7 \log_2 n$ Mul in XTR-ISE. Note that it is the most efficient one among XTR exponentiation algorithms without pre-computation.

3 Side Channel Attacks

Side channel attacks (SCA) are allowed to access the additional information linked to the operations using the secret key, e.g., timings, power consumptions, etc [12, 13]. This type of attack, which includes Simple Power Analysis (SPA) and Differential Power Analysis (DPA), render cryptographic devices such as smart cards vulnerable.

− In SPA, an attacker just needs to monitor the devices power consumption and identify the parts of the power trace that correspond to the difference of operations using the secret key. This gives trivially the secret key.

− DPA observes many power consumptions and analyzes these information together with statistic tools. An adversary should analyze the information of power consumptions with statistic tools per every target bit, however, he/she does not need re-observe new power consumptions to detect a next target bit because he/she can use the same obtained information.

3.1 Side Channel Attacks on XTR and Their Countermeasures

We summarize the previous related results about side channel attacks on XTR and countermeasures. According to target algorithms the proposed analysis techniques and countermeasures are as follows:

XTR Single Exponentiation Method (XTR-SE)[4]: Several side channel attacks such as DPA, the doubling attack, the refined power analysis, and the zero value attack, were proposed by Han et al. [10, 11]. But, it is secure against SPA without any countermeasures. In ICICS 2004, Ciet-Giraud showed that it can be also analyzed by the transient fault induction attacks [1]. As countermeasures against DPA, the exponent randomization, the exponent splitting, the base randomization, and the field randomization techniques were proposed in [10, 11, 16].

Improved XTR Single Exponentiation Method (XTR-ISE): Chung-Hasan [2] and Page-Stam [16] proposed SPA to XTR-ISE and that it was the first try to analyze it with SCA. Chung-Hasan showed it takes 2^{100} tries for an attacker until he/she correctly finds the secret key in XTR-ISE with 160-bits key length. On the other hand, Page-Stam showed it requires 2^{88} tries. As countermeasures against SPA and DPA, the indistinguishable arithmetic formula and the exponent splitting, were proposed and recommended as an efficient and adequate SPA and DPA countermeasure respectively [16]. It is the most efficient countermeasure against SPA and DPA among the proposed ones.

3.2 Challenges of This Paper

From the above previous results about XTR-ISE, we are encouraged to start the following two challenges;

(1) **Analysis -** *How can we reduce the complexity of analysis:* The analysis result of Page-Stam is more nice than that of Chung-Hasan, but these results are far worse than well-known square-root type algorithms (Baby-Step-Giant-Step or Pollards' Rho methods), i.e., their results are not practically feasible. In this paper we find new analysis technique, called as XTR collision attack, derived from the structural properties of XTR-ISE. The complexity of XTR collision attack against XTR-ISE is about 2^{40} where the key length is 160-bit, which is about 55% improvement from the result of Page-Stam [16]. The description of it is contained Section 4.

[4] This algorithm was proposed by Lenstra-Verheul [14].

(2) **Countermeasure -** *How can we design a secure countermeasure against SCA:* Even though Page-Stam's method, the indistinguishable arithmetic formula and exponent splitting as an SPA and DPA countermeasure respectively, is the most efficient countermeasure against SPA and DPA among the proposed ones, there are some controversial points in their method.

- Recently Walter showed that the produced unified code for elliptic addition and doubling in order to avoid SPA may still be insecure against SPA if there is sufficient side channel leakage at lower levels [20]. Thus the indistinguishable arithmetic technique is not recommended as a SPA countermeasure.
- If Montgomery arithmetic is used in the proposed indistinguishable arithmetic formula then some extra dummy additions have to added to $A[\cdot]$ routine to make up for this [16]. However, Yen et al. showed that the safe-error attack is applicable to the dummy method [22].

Thus in this paper we propose a novel countermeasure using a fixed pattern which is secure against SPA. In order to defeat DPA the exponent splitting technique is utilized. In Section 5 we describe the proposed countermeasure.

4 New Collision Attack on XTR

In this section we find new analysis technique, called as XTR collision attack, derived from the structural properties of XTR-ISE.

4.1 Some Properties of XTR-ISE

In XTR-ISE, Step 2 consists of eight states X_i and Y_i where $1 \leq i \leq 4$. One state is only determined by the condition of a and b. The following properties give us useful information to determine the next state, and their proof can be found in the appendix.

Proposition 1. *In XTR-ISE, the following relations are satisfied;*

1. *X_1 is followed by X_i and only Y_1 where $1 \leq i \leq 4$.*
2. *X_2 is followed by X_i where $1 \leq i \leq 4$.*
3. *X_3 is followed by X_i where $1 \leq i \leq 3$.*
4. *X_4 is followed by only Y_3 or Y_4.*
5. *Y_1 is followed by X_i and only Y_1 where $1 \leq i \leq 4$.*
6. *Y_2 is followed by X_i where $1 \leq i \leq 4$.*
7. *Y_3 is followed by X_i where $1 \leq i \leq 3$.*
8. *Y_4 is followed by only Y_3 or Y_4.*
9. *The last step is either X_1 or Y_1.*

The results of Proposition 1 give some useful information of the relation of executing consecutive two states. For example, Y_4 always follows either Y_3 or by oneself. From Proposition 1 we can see another property,

Property 2. State Y_2 never occurs in the process of XTR-ISE except the first time.

Thus, there is no state which can actually leads to Y_2's precondition. Thus Y_2 can not be occurred in the process of XTR-ISE except an appearance of the first time.

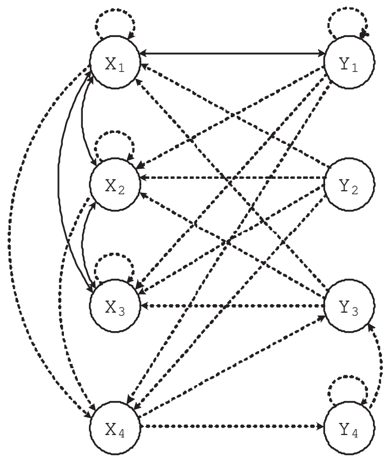

Fig. 1. The finite Markov chain associated with XTR-ISE

4.2 Assumptions and Notations

We first introduce some reasonable assumptions which is used in a new attack.

1. $A[x, y, z, w]$ and $D[x]$ are distinguishable by a single measurement of power consumption, whereas $D[x]^p$ and $D[x]$, and $A[x, y, z, w^p]$ and $A[x, y, z, w]$ are indistinguishable, respectively. Here, $x, y, z, w \in \mathbf{F}_{p^2}$.
2. When $\{A[\cdot], D[\cdot], D[\cdot]\}$ are all operated, e.g. in the case of X_3 in XTR-ISE, we assume these three functions are operated according to the following order $A[\cdot]D[\cdot]D[\cdot]$. In more detail, states X_i and Y_i are updated according to the following orders;
   (a) In X_2 and Y_4: the computation order is $T_2 \to T_0 \to T_3 \to T_1$,
   (b) In X_3 and Y_3: the computation order is $T_1 \to T_0 \to T_3 \to T_2$.
3. If $D[c_u]$ and $D[c_v]$ are computed, the attacker is not able to guess the value of c_u nor c_v but he/she is able to check if $c_u = c_v$.

As the required computing time of $A[\cdot]$ is two times of that of $D[\cdot]$ and p-th powering is free (refer to Section 2) in XTR, the above Assumption 1 is reasonable. Assumption 3 is also reasonable since this kind of computation usually takes many clock cycles and depends greatly on the value of the operand. This kind assumption has been used in a stronger variant and validated by Schramm et al. [19] who are able to distinguish collisions during one DES round computation. It was extended to ECC by Fouque at al [7].

Notations: For simplicity, $A[x, y, z, w]$ and $D[x]$ are referred to as $\mathbf{A}$ and $\mathbf{D}$, respectively. Let $S[c_1, n]$ be an AD sequence when the inputs are c_1 and n in XTR-ISE, i.e. $\mathbf{A}$ and $\mathbf{D}$ are elements of $S[c_1, n]$, which are written with time-increasing from left to right. Due to the above Assumption 1, $\mathbf{A}$ and $\mathbf{D}$ also denote $A[x, y, z, w^p]$ and $D[x]^p$, respectively.

As described in Section 2.3 $G_i = (Q_0, Q_1, Q_2, Q_3)$ is the i-th updated intermediate values of $\{Q_j\}_{0 \le j \le 3}$ in Step 2 of XTR-ISE for $i \ge 1$. For $G_i \xrightarrow{T_i} G_{i+1}$ where $T_i \in \{X_j, Y_j\}_{1 \le j \le 4}$, if T_i is one of $\{X_2, X_3, X_4, Y_3, Y_4\}$, then $\mathbf{DD}$ is computed. We denote these two $\mathbf{DD}$ as $\mathbf{D_i^1 D_i^2}$ ($\mathbf{D_i^1 D_i^2}$ are carried along for expository purposely only).

For example, with input c_1 and n, the utilized operations are

$$G_1 \overset{X_1}{\to} G_2 \overset{X_4}{\to} G_3 \overset{Y_3}{\to} G_4 \overset{X_2}{\to} G_5 \overset{X_1}{\to} G_6 \overset{Y_1}{\to} G_7 \ldots,$$

then the AD sequence of it is $S[c_1, n] = \mathbf{AD}_2^1\mathbf{D}_2^2\mathbf{AD}_3^1\mathbf{D}_3^2\mathbf{AD}_4^1\mathbf{D}_4^2\mathbf{AA} \cdots$.

4.3 Attacker's Goal

In XTR-ISE, Step 2 consists of eight states X_i and Y_i where $1 \leq i \leq 4$. One state is only determined by the condition of two integers a and b. However, if an attacker can decide the states executed during the computation then the secret key can be easily reconstructed from the recovered state. Note that from the properties of XTR-ISE described in Section 4.1, we do not consider state Y_2 in this attack any more.

Under Assumption 1, an attacker is able to distinguish **A**, **DD**, and **ADD**. With this information he/she can categorize seven states of XTR-ISE into the following three groups;
- **A** is corresponding to X_1 or Y_1,
- **DD** is corresponding to X_4,
- **ADD** is corresponding to X_2, X_3, Y_3 or Y_4.

However, there are some ambiguity decisions such as (1) X_1 and Y_1 are not distinguished, (2) if **ADD** is observed in AD sequence then there are two possibilities; **ADD** and **A**|**DD**. Using a brute force search technique, one might test around 6 candidates; i.e. **ADD** is corresponding to one of $\{X_2, X_3, Y_3, Y_4, X_1|X_4, Y_1|X_4\}$.

Thus the attacker needs to check the possible candidates until he/she has found the correct one, so in order to improve the efficiency of the attack we want to minimize the number of candidates.

4.4 Analysis based on the Finite Markov Chain

First we consider the following three types of AD sequences;

– **ADD**|**DD**.
– $\overbrace{\mathbf{ADD}|\mathbf{ADD}|\ldots|\mathbf{ADD}}^{m-times}$, briefly it is denoted as $\{\mathbf{ADD}\}^{\mathbf{m}}$.
– $\mathbf{ADD}|\overbrace{\mathbf{A}\ldots\mathbf{A}}^{m-times}|\mathbf{ADD}$, denoted as $\mathbf{ADD}|\{\mathbf{A}\}^{\mathbf{m}}|\mathbf{ADD}$.

When ADD|DD is observed in AD sequence we can decide $\underbrace{\mathbf{ADD}}_{X_2}|\underbrace{\mathbf{DD}}_{X_4}$. Because the last two **DD** originates from X_4 and the possible preconditions of X_4 are X_1, X_2, and Y_1. Thus **ADD** implies X_2

When $\{\mathbf{ADD}\}^{\mathbf{m}}$ is observed in AD sequence there are 6^m possible combinations from $\{X_2, X_3, Y_3, Y_4, X_1|X_4, Y_1|X_4\}$. However, if we consider the finite markov chain (Fig. 1) then we can reduce the possible number of combinations such as 15, 39, 102, and 243 combinations when m is $2, 3, 4$, and 5, respectively. If $m \geq 4$ then the number of all possible combinations from the finite markov chain is

$$\#\left[\{\mathbf{ADD}\}^{\mathbf{m}}\right] = (39m + 48) \cdot 2^{m-5}. \tag{5}$$

The proof of it is contained in Appendix.

When $\mathbf{ADD}|\{\mathbf{A}\}^{\mathbf{m}}|\mathbf{ADD}$ is observed in AD sequence there are $6 \cdot 2^m \cdot 6 = 9 \cdot 2^{m+2}$ combinations of XTR states. However, if we consider the finite markov chain (Fig. 1) then the only possible combinations of XTR states are as follows;

$$\underbrace{\begin{bmatrix} X_2 \\ X_3 \\ Y_3 \end{bmatrix}}_{\mathbf{ADD}} \times \underbrace{[X_1] \times \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}^{m-1}}_{\{\mathbf{A}\}^{\mathbf{m}}} \times \underbrace{\begin{bmatrix} X_2 \\ X_3 \\ X_1X_4 \\ Y_1X_4 \end{bmatrix}}_{\mathbf{ADD}} \tag{6}$$

Thus the number of possible combinations is $3 \cdot 2^{m+1}$. Furthermore, we propose the following decision rule derived from Proposition 1.

Property 3. If **AADDAA** is observed in AD sequence then we can decide $\mathbf{A}|\underbrace{\mathbf{ADD}}_{\mathbf{X_2} \text{ or } \mathbf{X_3}}|\underbrace{\mathbf{A}}_{\mathbf{X_1}}|\mathbf{A}$.

Proof. **ADD** can be one of $\{X_2, X_3, Y_3, Y_4, X_1|X_4, Y_1|X_4\}$. As Y_3 and Y_4's precondition is one of $\{X_4, Y_4\}$, Y_3 and Y_4 are discarded in candidates. As $X_1|X_4$ and $Y_1|X_4$ are followed by X_3 or Y_4 these two are also discarded. Thus **ADD** implies X_2 or X_3. As Y_1 can not be followed by X_2 or X_3, **A** placed after **ADD** is X_1. □

4.5 XTR Collision Attack

At the previous section, the number of possible combinations for $\{\mathbf{ADD}\}^{\mathbf{m}}$ and $\mathbf{ADD}|\{\mathbf{A}\}^{\mathbf{m}}|\mathbf{ADD}$ is decreased by using the finite markov chain of XTR-ISE. In this section, in order to reduce the search space from the finite markov chain we introduce a new attack mainly based on the above assumptions, especially Assumption 3, described in 4.2.

Key Observation: If we focus on **D** operation, we notice that some of them manipulate the same operand. We consider two AD sequences $S[c_1, n]$ and $S[c_2, n]$.

In the case of $\{\mathbf{ADD}\}^{\mathbf{m}}$: For simplicity, we assume $m = 2$, i.e. **ADDADD** is considered. Note that there are 15 combinations of states.

$$S[c_1, n] = \ldots \mathbf{AD_i^1 D_i^2 AD_j^1 D_j^2} \ldots$$
$$S[c_2, n] = \ldots \mathbf{AD_i^1 D_i^2 AD_j^1 D_j^2} \ldots$$

Depending on the combination type, we can observe the following results;

CASE_I: $\mathbf{D_j^1}$ of $S[c_1, n]$ is same to $\mathbf{D_i^1}$ of $S[c_2, n]$,
CASE_II: $\mathbf{D_j^2}$ of $S[c_1, n]$ is same to $\mathbf{D_i^1}$ of $S[c_2, n]$.

According to the above observation, the 15 combination pairs are categorized as
CASE_I : $(X_2, X_2), (X_2, X_3), (X_3, X_2), (X_3, X_3), (X_1, X_4, Y_4), (Y_1, X_4, Y_4), (Y_3, X_2),$ $(Y_3, X_3), (Y_4, Y_4), (X_2, X_1, X_4), (X_3, X_1, X_4), (Y_3, X_1, X_4),$
CASE_II : $(X_1, X_4, Y_3), (Y_1, X_4, Y_3), (Y_4, Y_3)$.

In order to confirm the validity of this categorization, we consider two examples. Assume that two cases (X_2, X_3) and (Y_4, Y_3). Table 1 shows that in the case of (X_2, X_3) CASE_I collision is occurred and CASE_II collision is detected in (Y_4, Y_3). Similarly, other pairs are also categorized into two cases.

With this collision information, an attacker is required to test on average 10.2 ($= \frac{12^2+3^2}{15}$) in order to find the precise combination pair corresponding to the target **ADDADD**. If $m = 3$, i.e. **ADDADDADD**, similarly 39 combination pairs are categorized into $\mathbf{CASE\_(I, I)}, \mathbf{CASE\_(I, II)}$, and $\mathbf{CASE\_(II, I)}$ and the number of elements of each case is 24, 6, and 9, respectively. Here, $\mathbf{CASE\_(I, I)}, \mathbf{CASE\_(I, II)}$, and

Table 1. Some examples for **CASE_I** and **CASE_II**.

Assume $G_i = (c_e, c_f, c_g, c_h)$ with input c_1 in XTR-ISE.

	Step	Input	
		c_1 and n	c_2 and n
CASE_I	G_i	(c_e, c_f, c_g, c_h)	$(c_{2e}, c_{2f}, c_{2g}, c_{2h})$
	$\Downarrow X_2$	$A[\cdot], D_i^1[c_e], D_i^2[c_g]$	$A[\cdot], \underline{D_i^1[c_{2e}]}, D_i^2[c_{2g}]$
	G_{i+1}	$(c_{2e}, c_f, c_{e+g}, c_{2g})$	$(c_{4e}, c_{2f}, c_{2(e+g)}, c_{4g})$
	$\Downarrow X_3$	$A[\cdot], \underline{D_{i+1}^1[c_{2e}]}, D_{i+1}^2[c_f]^p$	$A[\cdot], D_{i+1}^1[c_{4e}], D_{i+1}^2[c_{2f}]^p$
	G_{i+2}	$(c_{4e}, c_{2e+f}, c_{e+g}, c_{2f}^p)$	$(c_{8e}, c_{2(2e+f)}, c_{2(e+g)}, c_{4f}^p)$

	Step	Input	
		c_1 and n	c_2 and n
CASE_II	G_i	(c_e, c_f, c_g, c_h)	$(c_{2e}, c_{2f}, c_{2g}, c_{2h})$
	$\Downarrow Y_4$	$A[\cdot], D_i^1[c_e], D_i^2[c_g]$	$A[\cdot], \underline{D_i^1[c_{2e}]}, D_i^2[c_{2g}]$
	G_{i+1}	$(c_{2e}, c_f, c_{e+g}, c_{2g})$	$(c_{4e}, c_{2f}, c_{2(e+g)}, c_{4g})$
	$\Downarrow Y_3$	$A[\cdot], D_{i+1}^1[c_f], \underline{D_{i+1}^2[c_{2e}]^p}$	$A[\cdot], D_{i+1}^1[c_{2f}], D_{i+1}^2[c_{4e}]^p$
	G_{i+2}	$(c_{2f}, c_{2e+f}, c_{e+g}^p, c_{4e}^p)$	$(c_{4f}, c_{2(2e+f)}, c_{2(e+g)}^p, c_{8e}^p)$

CASE_$(\mathbf{II}, \mathbf{I})$ denote that the first **ADDADD** has Case_I, Case_I, and Case_II collision and the last **ADDADD** has Case_I, Case_II, and Case_I collision. Thus on average 17.77 tests are required to detect the precise combination pair corresponding to $\{\mathbf{ADD}\}^{\mathbf{3}}$.

# of all possible combinations			
m	From Exhaustive Search	From the Finite Markov Chain	From Collision Attack
1	6	6	6
2	36	15	10.2
3	216	39	17.77
4	1296	102	31.59
5	7776	243	45.4
$\vdots$	$\vdots$	$\vdots$	$\vdots$

From the results of the above table we can see that the average number of trial tests with collision information is drastically decreased compared to that of the finite markov chain. For example, if $m = 5$ then the required number of trial tests is only 18% of that of the finite markov chain.

In the case of ADD$|\{\mathbf{A}\}^{\mathbf{m}}|$ADD: Consider

$$S[c_1, n] = \ldots \mathbf{A}\mathbf{D}_{\mathbf{i}}^{\mathbf{1}}\mathbf{D}_{\mathbf{i}}^{\mathbf{2}}\{\mathbf{A}\}^{\mathbf{m}}\mathbf{A}\mathbf{D}_{\mathbf{j}}^{\mathbf{1}}\mathbf{D}_{\mathbf{j}}^{\mathbf{2}} \ldots$$
$$S[c_2, n] = \ldots \mathbf{A}\mathbf{D}_{\mathbf{i}}^{\mathbf{1}}\mathbf{D}_{\mathbf{i}}^{\mathbf{2}}\{\mathbf{A}\}^{\mathbf{m}}\mathbf{A}\mathbf{D}_{\mathbf{j}}^{\mathbf{1}}\mathbf{D}_{\mathbf{j}}^{\mathbf{2}} \ldots$$

Similar to the previous analysis, we can observe the following results depending on the combination type;

CASE_0: There is no relation between **D** operation of $S[c_1, n]$ and $S[c_2, n]$,
CASE_I: $\mathbf{D}_{\mathbf{j}}^{\mathbf{1}}$ of $S[c_1, n]$ is same to $\mathbf{D}_{\mathbf{i}}^{\mathbf{1}}$ of $S[c_1, n]$,
CASE_II: $\mathbf{D}_{\mathbf{j}}^{\mathbf{2}}$ of $S[c_1, n]$ is same to $\mathbf{D}_{\mathbf{i}}^{\mathbf{1}}$ of $S[c_2, n]$.

For example, when $m = 1$

$$\begin{bmatrix} X_1 \\ X_3 \\ Y_3 \end{bmatrix} \times [X_1] \times [Y_1 X_4] \in \mathbf{CASE\_I}, \quad \begin{bmatrix} X_1 \\ X_3 \\ Y_3 \end{bmatrix} \times [X_1] \times \begin{bmatrix} X_1 X_4 \\ X_3 \end{bmatrix} \in \mathbf{CASE\_II}.$$

With this collision information, an attacker is required to test on average 4.5 ($= \frac{3^2+3^2+6^2}{12}$) in order to find the precise combination pair corresponding to the target **ADDAADD**. The results of the following table show the improvement of analysis complexity.

# of all possible combinations			
m	From Exhaustive Search	From the Finite Markov Chain	From Collision Attack
1	72	12	4.5
2	144	24	9.75
3	288	48	28.87
4	576	96	74.43
5	1152	192	169.21
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Implementation Results: From these classifications, we can reduce the search space order required to detect the whole secret value. The graph of results in Fig. 2 shows that the average number of trial XTR exponentiations is roughly given by $2^{0.25 \cdot l}$ where l is the length of the exponents. Thus the complexity of XTR collision attack against XTR-ISE is about 2^{40} where the key length is 160-bit, which is about 55% improvement from the result of Page-Stam [16].

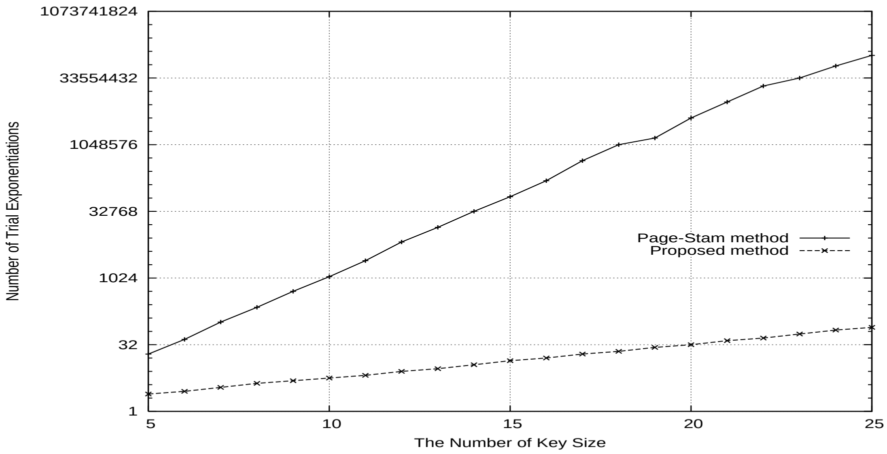

Fig. 2. Comparison of analysis complexity between the proposed attack and Page-Stam's one

5 Proposed Countermeasure

In this section we explain the proposed algorithm. We modify XTR-ISE to be secure against SCA. The main idea is same to that of Okeya-Takagi scheme [15] for elliptic curve cryptosystems. In XTR-ISE there are three different patterns, A, DD, ADD. For example, if X_1, Y_1, and X_4 are consecutively operated then the sequence is "$AAADD$", which is no longer the fixed pattern.

We try to generate a XTR operation sequence that has a fixed pattern such that $|ADD|ADD| \dots |ADD|$. Firstly we describe the proposed scheme as follows:

Fixed Pattern XTR Single Exponentiation (XTR-FSE)

Input: c_1 and n where $n > 2$
Output: c_n

---

1. Initialization:
   1.1. Select a random number a in $[1, n-1]$ and $b = n - a$. If a is even, then let $a \leftarrow a+1$, $b \leftarrow b-1$.
   1.2. Let $Q_0 = c$, $Q_1 = c$, $Q_2 = 3$, and $Q_3 = c^p$.
2. As long as $a \neq b$
   2.1. If $b > a$
   $\mathcal{X}_1$. if b is even, then $(a, b) \leftarrow (a, b/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
   $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$.
   $\mathcal{X}_2$. else (b is odd), then $(a, b) \leftarrow (a, (b-a)/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
   $T_2 \leftarrow Q_2$, $T_3 \leftarrow D[Q_1]^p$.
   2.2. Else (if $a > b$)
   $\mathcal{Y}_1$. if b is odd, then $(a, b) \leftarrow (b, (a-b)/2)$
   $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
   $T_2 \leftarrow Q_2^p$, $T_3 \leftarrow D[Q_0]^p$.
   $\mathcal{Y}_2$. else (b is even), then $(a, b) \leftarrow (a, b/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
   $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$
   2.3. Set $Q_0 \leftarrow T_0$, $Q_1 \leftarrow T_1$, $Q_2 \leftarrow T_2$, $Q_3 \leftarrow T_3$.
3. Compute $\tilde{c} = A[Q_0, Q_1, Q_2, Q_3] = c_{u+v}$.
4. If $a = 1$ then return $\tilde{c}$,
   else goto Step 1. with $c = \tilde{c}$ and $n = a$.

In order to make "*ADD*" fixed pattern, first we eliminate X_1, X_4, Y_1, Y_2 from XTR-ISE and then change the condition of input a and b in the initialization step such that the integer a is always odd. For easy explanation of the properties of XTR-FSE, we define some notations;

Notations: Let $S_i := (\mathcal{A}_i, \mathcal{B}_i)$ be the i-th updated values of a and b in Step 2 of XTR-FSE for $i \geq 0$. $S_0 = (\mathcal{A}_0, \mathcal{B}_0)$ denotes the updated values at the initialization step, i.e. $\mathcal{A}_0 = a$ and $\mathcal{B}_0 = b$. $S_i \xrightarrow{\mathcal{X}_1} S_{i+1}$ denotes S_{i+1} is updated by $\mathcal{X}_1$ from S_i.

We derive some properties of XTR-FSE, and their proofs can be found in the appendix.

Lemma 1. *With input integer n, XTR-FSE has the following properties;*

(1) $\mathcal{A}_i$ *is always odd integer,*
(2) $\mathcal{A}_i$ *is a monotone decreasing function,*
(3) $\mathcal{A}_i = \mathcal{A}_{i-1}$ *and* $\mathcal{B}_i \leq \frac{\mathcal{B}_{i-1}}{2}$ *at* $\mathcal{X}_1, \mathcal{X}_2$, *and* $\mathcal{Y}_2$,
(4) $\mathcal{A}_i + \mathcal{B}_i = \frac{\mathcal{A}_{i-1}+\mathcal{B}_{i-1}}{2}$ *at* $\mathcal{X}_2$ *and* $\mathcal{Y}_1$,
(5) $\mathcal{A}_i + \mathcal{B}_i = t(\mathcal{A}_{i-1} + \mathcal{B}_{i-1})$ *at* $\mathcal{X}_1$ *and* $\mathcal{Y}_2$, *where* $1/2 < t < 1$,
(6) $\mathcal{A}_i + \mathcal{B}_i$ *is a strictly decreasing function,*
(7) $\#\{\mathcal{X}_2, \mathcal{Y}_1\} \leq \log_2 n$.

Proof. The proof of (1),(2),(3),(4), and (5) are immediate and (6) follows from (4) and (5). Suppose $(a, b) = S_0 \overset{W_1}{\to} S_1 \overset{W_2}{\to} S_2 \overset{W_3}{\to} \ldots \overset{W_t}{\to} S_t \ldots$ for $W_i \in \{\mathcal{X}_2, \mathcal{Y}_1\}$. From (4) it follows that $\mathcal{A}_t + \mathcal{B}_t = \left(\frac{1}{2}\right)^t (a+b) = \left(\frac{1}{2}\right)^t n$ for any t. As $\mathcal{A}_i$ and $\mathcal{B}_i$ are positive integers the minimum of $\mathcal{A}_i + \mathcal{B}_i$ is 2, thus $t \leq \log_2(n) - 1$, which proves (7). $\square$

Lemma 2. *Assume* $S_{i-1} \xrightarrow{\mathcal{Y}_1} S_i$.

(1) If $\mathcal{A}_{i-1} > 3\mathcal{B}_{i-1}$ *then*
   a. $\mathcal{A}_i < \mathcal{A}_{i-1}$, $\mathcal{B}_i > \mathcal{B}_{i-1}$, *and* $\mathcal{B}_i < \frac{\mathcal{A}_{i-1}}{2}$,
   b. the next state is $\mathcal{X}_1$ *or* $\mathcal{X}_2$.
(2) Else ($\mathcal{A}_{i-1} \leq 3\mathcal{B}_{i-1}$) *then*

a. $\mathcal{A}_i < \mathcal{A}_{i-1}$ *and* $\mathcal{B}_i \le \mathcal{B}_{i-1}$,
b. if $\mathcal{A}_{i-1} < 3\mathcal{B}_{i-1}$ *then the next state is* $\mathcal{Y}_1$ *or* $\mathcal{Y}_2$*, otherwise goto Step 3.*

The proof of Lemma 2 follows from a straightforward computation. Let $\mathcal{Y}_1^{a>3b}$ and $\mathcal{Y}_1^{a\le 3b}$ be the states such that $\mathcal{Y}_1$ with inputs S_{i-1} such that $\mathcal{A}_{i-1} > 3\mathcal{B}_{i-1}$ and $\mathcal{A}_{i-1} \le 3\mathcal{B}_{i-1}$, respectively.

5.1 Average-case Analysis of Fixed Pattern XTR Single Exponentiation

In this section we will analyze the average number of states of the proposed algorithm. First we prove that the proposed algorithm is polynomial time algorithm, i.e. it is terminated in $O(\log_2 n)$ with the input integer n in XTR-FSE.

XTR-FSE is a Polynomial Time Algorithm: Let $S = S_0S_1S_2\cdots$ be the sequence of S_i updated by one of $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}_1, \mathcal{Y}_2\}$. From Lemma 2, we can separate S into several sub-sequences with $\mathcal{Y}_1^{a>3b}$ like as

$$S = \underbrace{S_0S_1\cdots S_{k_1-1}}_{P_1}|S_{k_1}|\underbrace{S_{k_1+1}\cdots S_{k_2-1}}_{P_2}|S_{k_2}|\cdots|S_{k_{t-1}}|\underbrace{S_{k_{t-1}+1}\cdots S_{k_t-1}}_{P_t}|S_{k_t}|S_{k_t+1}\cdots$$

, where S_{k_i} is updated by state $\mathcal{Y}_1^{a>3b}$. Thus S_j in each separated part P_i is updated by one of $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}_1^{a\le 3b}, \mathcal{Y}_2\}$. Let L_i and M_i be the total number of $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}_2\}$ and $\mathcal{Y}_1^{a\le 3b}$ utilized in part P_i, respectively. Let N be the total number of $\mathcal{Y}_1^{a>3b}$ used in S. From Lemma 1 -(3) and Lemma 2, the maximum values of the updated $\mathcal{A}_i$ and $\mathcal{B}_i$ are described in Table 2.

Table 2. The maximum values of the updated $\mathcal{A}_i$ and $\mathcal{B}_i$

Sequence S	Updated $\mathcal{A}_i$ and $\mathcal{B}_i$	
	$\mathcal{A}_i$	$\mathcal{B}_i$
Initialization (S_0)	$\mathcal{A}_0 = a$	$\mathcal{B}_0 = b$
P_1	$\max(\mathcal{A}_i) \le a$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_1}b$
S_{k_1}	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_1}b$	$\max(\mathcal{B}_i) \le \frac{1}{2}a$
P_2	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_1}b$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_2+1}a$
S_{k_2}	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_2+1}a$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_1+1}b$
P_3	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_2+1}a$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_1+L_3+1}b$
S_{k_3}	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_1+L_3+1}b$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_2+2}a$
P_4	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_1+L_3+1}b$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_2+L_4+2}a$
S_{k_4}	$\max(\mathcal{A}_i) \le (\frac{1}{2})^{L_2+L_4+2}a$	$\max(\mathcal{B}_i) \le (\frac{1}{2})^{L_1+L_3+2}b$
$\vdots$	$\vdots$	$\vdots$

As $\max(\mathcal{A}_i)$ and $\max(\mathcal{B}_i)$ are positive integers, $\sum_{i=1} L_{2i} \le \log_2 a$ and $\sum_{i=1} L_{2i-1} \le \log_2 b$. Thus $\sum_{i=1} L_i \le 2\log_2 n$ because $a, b < n$. From the result of Lemma 1 -(7), $N + \sum_{i=1} M_i \le \log_2 n$. Thus

$$\sum_{i=1}(L_i + M_i) + N \le 3\log_2 n,$$

so that we have proved:

Proposition 2. *For a given integer* n*, the proposed algorithm takes at most* $3\log_2 n$ *iterations in Step 2. This implies that it is a polynomial time algorithm.*

Heuristic Estimation of XTR-FSE: From now on, we would like to investigate a heuristic estimation result of the proposed algorithm. From Proposition 2 we can assume that the sequence $S = S_0S_1\cdots S_m$ where $m \le 3\log_2 n$. Note that the last $S_m = (1, 1)$.

Assume Pr[$\mathcal{X}_1$] =Pr[$\mathcal{X}_2$] =Pr[$\mathcal{Y}_1$] =Pr[$\mathcal{Y}_2$] $= m/4$. From the result of (4)-(6) in Lemma 1, we can derive the following equation:

$$\begin{aligned}\mathcal{A}_m + \mathcal{B}_m = 2 &= \left(\frac{1}{2}\right)^{m/2} \cdot (t)^{m/2} \cdot (\mathcal{A}_0 + \mathcal{B}_0) \\ &= \left(\frac{1}{2}\right)^{m/2} \cdot (t)^{m/2} \cdot n\end{aligned}$$

where $1/2 < t < 1$.

Let $t = 3/4$. Then the iteration number m is asymptotically logarithmic,

$$m \approx 1.41 \log_2 n.$$

In order to confirm our estimation result above we show simulation results. We randomly generate one million integers n with bit length $160, 200, 500, 1000, 5000, 10000$, respectively. Table 3 demonstrates that our estimation result matches the simulation results (SR) for the large input integer n.

Table 3. Experiment for the proposed algorithm

Size of n	160 bits	200 bits	500 bits	1000 bits	5000 bits	10000 bits
# of iterations	222	279	703	1409	7057	14117
SR	$1.39 \log_2 n$	$1.39 \log_2 n$	$1.40 \log_2 n$	$1.40 \log_2 n$	$1.41 \log_2 n$	$1.41 \log_2 n$

5.2 Security Analysis

In this section we discuss the security of the proposed scheme against SPA and DPA.

SPA: As we mentioned previous section, the proposed method compute XTR single exponentiation through the fixed pattern $|ADD|ADD|\ldots|ADD|$. The attacker could distinguish XTR operations $D[\cdot]$ and $A[\cdot]$ in XTR-FSE by measurement of the power consumption, but he/she obtains only the identical ADD sequence for any input c and n. Therefore, he/she cannot detect the secret scalar n by using SPA.

DPA: The use of scalar randomization such as exponent splitting [3] prevents against DPA. Note that the idea of splitting the data was already abstracted in [4] as a general countermeasure against DPA. The proposed method is using exponent splitting technique as a DPA countermeasure, i.e. we split the input integer n into two parts by picking a random $a \in [1, n-1]$ and rewriting the integer n as $a + (n-a)$. Thus XTR-FSE is secure against DPA.

5.3 Comparison of empirical performance and type of countermeasure

In this section we compare the computational cost and the type of countermeasures between the proposed countermeasure and the previous ones.

The compared three methods use the exponent splitting method as DPA countermeasure. But the utilized SPA countermeasure is different each others. The countermeasure of ICICS'04 is based on XTR-SE. Their method does not require SPA countermeasure because XTR-SE has the fixed operations **ADD**. On the other hand, the countermeasure of SAC'04 and the proposed method is based on XTR-ISE, which does not has fixed operations. In order to solve this problem Page-Stam proposed the indistinguishable arithmetic with dummy operation sometimes, but the security of indistinguishable arithmetic [20] and the dummy method [22] are recently very controversial. From the result of Table 4 our proposed countermeasure is the best one in XTR in the sense of both efficiency and security.

Table 4. Comparison of empirical performance and type of countermeasure

	Efficiency	Type of Countermeasure	
	Compute $Tr(g^n)$	SPA	DPA
ICICS'04 [10]	$16 \log_2(n)$	Fixed Pattern + No Dummy Operation	Exponent Splitting
SAC'04 [16]	$8.5 \log_2(n)$	Indistinguishable Assumption + Dummy Operation	Exponent Splitting
Proposed Method	$11.2 \log_2(n)$	Fixed Pattern + No Dummy Operation	Exponent Splitting

Acknowledgements

Dong-Guk Han was supported by the Korea Research Foundation Grant. (KRF-2005-214-C00016) and Tae Hyun Kim was supported in part by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

References

1. M. Ciet and C. Giraud, *Transient Fault Induction Attacks on XTR*, Information and Communications Security (ICICS 2004), LNCS 3269, (2004), 440-451.
2. J. Chung and A. Hasan, *Security Analysis of XTR Exponentiation Algorithms against Simple Power Analysis Attack*, Preprint of CACR, Univ. of Waterloo, CACR 2004-05.
3. C. Clavier and M. Joye, *Univeral Exponentiation Algorithm A First Step towards Provable SPA-Resistance*, Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 300-308.
4. S. Chari, C.S. Jutla, J.R. Rao, and P. Rohatgi, *Towards sound approaches to counteract power-analysis attacks*, Advances in Cryptology - CRYPTO '99, LNCS1666, (1999), 398-412.
5. Dijk, M.v., Granger, R., Page, D., Rubin, K., Silverberg, A., Stam, M., Woodruff, D., *Practical Cryptography in High Dimensional Tori*, to be appeared in Eurocrypt 2005.
6. Dijk, M.v., Woodruff, D., *Asymptotically Optimal Communication for Torus-Based Cryptography*, Advances in Cryptology - CRYPTO '04, LNCS 3152, (2004), 157-178.
7. P.-A. Fouque and F. Valette, *The Doubling Attack Why Upwards is better than Downwards*, Workshop on Cryptographic Hardware and Embedded Systems 2003 (CHES 2003), LNCS 2779, (2003), 269-280.
8. G. Gaubatz, J.-P. Kaps, and B. Sunar, *Public Key Cryptography in Sensor Networks-Revisited*, 1st European Workshop on Security in Ad-Hoc and Sensor Networks, (ESAS 2004), LNCS3313, (2004), 2-18.
9. R. Granger, D. Page, and M. Stam, *A Comparison of CEILIDH and XTR*, Algorithmic Number Theory, (ANTS 2004), LNCS 3076, (2004), 235-249.
10. D.-G. Han, T. Izu, J. Lim, and K. Sakurai, *Modified Power-Analysis Attacks on XTR and An Efficient Countermeasure*, Information and Communications Security (ICICS 2004), LNCS 3269, (2004), 305-317.
11. D.-G. Han, J. Lim, and K. Sakurai, *On security of XTR public key cryptosystems against Side Channel Attacks*, The 9th Australasian Conference in Information Security and Privacy, (ACISP 2004), LNCS 3108, (2004), 454-465.
12. Kocher, C., *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology - CRYPTO '96, LNCS 1109, (1996), 104-113.
13. Kocher, C., Jaffe, J., Jun, B., *Differential Power Analysis*, Advances in Cryptology - CRYPTO '99, LNCS1666, (1999), 388-397.
14. A.K. Lenstra and E.R. Verheul, *The XTR public key system*, Advances in Cryptology - CRYPTO '00, LNCS1880, (2000), 1-19.

15. K. Okeya and T. Takagi, *The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks*, CT-RSA 2003, LNCS 2612, (2003), 328-342, 2003.
16. D. Page and M. Stam, *On XTR and Side-Channel Analysis*, Pre-proceedings of SAC 2004, 67-81.
17. K. Rubin and A. Silverberg, *Torus-Based Cryptography*, Advances in Cryptology - CRYPTO '03, LNCS2729, (2003), 349-365.
18. M. Stam and A.K. Lenstra, *Speeding Up XTR*, Proceedings of Asiacrypt 2001, LNCS2248, (2001), 125-143.
19. K. Schramm, T. Wollinger, and C. Paar, *A New Class of Collision Attacks and its Application to DES*, Proceedings of FSE 2003, LNCS2887, (2003), 206-222.
20. C.D. Walter, *Simple Power Analysis of Unified Code for ECC Double and Add*, Workshop on Cryptographic Hardware and Embedded Systems 2004 (CHES 2004), LNCS 3156, (2004), 191-204.
21. R. Watro, D. Kong, S-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus, *TinyPK: Securing Sensor Networks with Public Key Technology*, ACM Workshop on Security of Ad Hoc and Sensor Networks 2004 (SASN 2004), 59-64.
22. S.-M. Yen, S. Kim, S. Lim, and S. Moon, *A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack*, Information Security and Cryptology 2001 (ICISC 2001), LNCS 2288, (2001), 414-427.

A Proof of Proposition 1

We will give a proof of Proposition 1.

1. X_1 has a condition that $a < b \leq 4a$. And the updating rule is $(a', b') \leftarrow (b - a, a)$ where a', b' can be even or odd. Thus $0 < a' \leq 3b'$. If $0 < a' < b'$, then cases X_1 through X_4 will occur because $b' \leq 4a'$ or $b' > 4a'$ can be satisfied depending on given a', b'. If $b' < a' \leq 3b'$, then case Y_1 will occur because $a' > b'$ and $a' \leq 4b'$. So X_1 is followed by X_1 through X_4 or Y_1.
2. X_2 has conditions that $4a < b$ and b is even. And the updating rule is $(a', b') \leftarrow (a, \frac{b}{2})$ where a', b' can be even or odd. $4a < b$ implies $2a' < b'$. If $2a' < b' \leq 4a'$, then cases X_1 will occur. If $b' > 4a'$, then cases X_2 through X_4 will occur. So X_2 is followed by X_1 through X_4.
3. X_3 has conditions that $4a < b$ and b is odd and a is odd. And the updating rule is $(a', b') \leftarrow (a, \frac{b-a}{2})$ where a' is odd. $4a < b$ implies $\frac{3a'}{2} < b'$. If $\frac{3a'}{2} < b' \leq 4a'$, then cases X_1 will occur. If $b' > 4a'$, then cases X_2 through X_4 will occur. As a' is odd, however, X_4 can not be occurred. So X_3 is followed by X_1 through X_3.
4. X_4 has conditions that $4a < b$ and b is odd and a is even. And the updating rule is $(a', b') \leftarrow (b, \frac{a}{2})$ where a' is odd. $4a < b$ implies $8b' < a'$. Since a' is odd and $4b' < a'$, Y_3 and Y_4 will occur. So X_4 is followed by Y_3 and Y_4.
5. Y_1 has a condition that $b < a \leq 4b$. And the updating rule is $(a', b') \leftarrow (a - b, b)$ where a', b' can be even or odd. Thus $0 < a' \leq 3b'$. If $0 < a' < b'$, then cases X_1 through X_4 will occur because $b' \leq 4a'$ or $b' > 4a'$ can be satisfied depending on given a', b'. If $b' < a' \leq 3b'$, then case Y_1 will occur because $a' > b'$ and $a' \leq 4b'$. So Y_1 is followed by X_1 through X_4 or Y_1.
6. Y_2 has conditions that $4b < a$ and a is even. And the updating rule is $(a', b') \leftarrow (b, \frac{a}{2})$ where a', b' can be even or odd. $4b < a$ implies $2a' < b'$. If $2a' < b' \leq 4a'$, then cases X_1 will occur. If $b' > 4a'$, then cases X_2 through X_4 will occur. So Y_2 is followed by X_1 through X_4.
7. Y_3 has conditions that $4b < a$ and a is odd and b is odd. And the updating rule is $(a', b') \leftarrow (b, \frac{a-b}{2})$ where a' is odd. $4b < a$ implies $\frac{3a'}{2} < b'$. If $\frac{3a'}{2} < b' \leq 4a'$, then cases X_1 will occur. If $b' > 4a'$, then cases X_2 through X_4 will occur. As a' is odd, however, X_4 can not be occurred. So Y_3 is followed by X_1 through X_3.

8. Y_4 has conditions that $4b < a$ and a is odd and b is even. And the updating rule is $(a', b') \leftarrow (a, \frac{b}{2})$ where a' is odd. $4b < a$ implies $8b' < a'$. Since a' is odd and $4b' < a'$, Y_3 and Y_4 will occur. So Y_4 is followed by Y_3 and Y_4.
9. In the steps of $\{X_2, X_3, X_4, Y_2, Y_3, Y_4\}$, one of a, b is fourth times bigger than the other. In the case of $4a < b$, X_2, X_3, X_4 are occur. If X_2 occurs, (a', b') is updated by $(a, \frac{b}{2})$. Since $2a' < b'$, $a' = b'$ can not be happened. If X_3 occurs, (a', b') is updated by $(a, \frac{b-a}{2})$. Since $\frac{3a'}{2} < b'$, $a' = b'$ can not be happened. If X_4 occurs, (a', b') is updated by $(b, \frac{a}{2})$. Since $8b' < a'$, $a' = b'$ can not be happened. The cases of $\{Y_2, Y_3, Y_4\}$ are similar with $\{X_2, X_3, X_4\}$. Therefore the last step is either X_1 or Y_1.

□

B Proof of Equation (5)

Let $\mathcal{A} := \begin{bmatrix} X_2 \\ X_3 \\ X_1X_4 \end{bmatrix}$ and $\mathcal{B} := \begin{bmatrix} Y_3 \\ Y_4 \end{bmatrix}$. Then all possible combinations of **ADDADD** are

$$[X_2] \times \mathcal{A}, \quad [X_3] \times \mathcal{A}, \quad [Y_3] \times \mathcal{A}$$

$$[Y_4] \times \mathcal{B}, \qquad [X_1X_4] \times \mathcal{B}, \qquad [Y_1X_4] \times \mathcal{B}$$

If $m = 3$ then all possible combinations of $\{\mathbf{ADD}\}^{\mathbf{3}}$ are

$$[X_2] \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix}, \quad [X_3] \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix}, \quad [Y_3] \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix},$$

$$[Y_4] \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix}, \quad [X_1X_4] \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix}, \quad [Y_1X_4] \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix}.$$

Thus $\#\left[\{\mathbf{ADD}\}^{\mathbf{3}}\right] = 3 * (8 + 5)$.

If $m = 4$ then all possible combinations of $\{\mathbf{ADD}\}^{\mathbf{4}}$ are

$$[X_2] \times \begin{bmatrix} X_2 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_1X_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}, \quad [X_3] \times \begin{bmatrix} X_2 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_1X_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}, \quad [Y_3] \times \begin{bmatrix} X_2 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ X_1X_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}$$

$$[Y_4] \times \begin{bmatrix} Y_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ Y_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}, \quad [X_1X_4] \times \begin{bmatrix} Y_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ Y_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}, \quad [Y_1X_4] \times \begin{bmatrix} Y_3 \times \begin{bmatrix} X_2 \times [\mathcal{A}] \\ X_3 \times [\mathcal{A}] \\ X_1X_4 \times [\mathcal{B}] \end{bmatrix} \\ Y_4 \times \begin{bmatrix} Y_3 \times [\mathcal{A}] \\ Y_4 \times [\mathcal{B}] \end{bmatrix} \end{bmatrix}.$$

Thus $\#\left[\{\mathbf{ADD}\}^{\mathbf{4}}\right] = 3 * (2 * 8 + 5 + 8 + 5)$. From above observations, we can derive the following equation for $m \geq 4$

$$\begin{aligned} \#\left[\{\mathbf{ADD}\}^{\mathbf{m}}\right] &= 3 * \left\{2^{n-4} * (16 + 5) + 2^{n-5} * (n - 4) * (8 + 5) + 2^{n-4} * (8 + 5)\right\} \\ &= (39 * m + 48) * 2^{m-5}. \end{aligned}$$