Security Analysis of XTR Exponentiation Algorithms Against Simple Power Analysis Attack*

Jaewook Chung and Anwar Hasan
j4chung@uwaterloo.ca
ahasan@secure.uwaterloo.ca

Center for Applied Cryptographic Research
University of Waterloo
Ontario, Canada

Abstract. A security analysis of XTR exponentiation algorithms against simple power analysis attack is presented. Under very reasonable assumptions, we prove that there exists a one-to-one correspondence between power trace and XTR operation sequence. With this result and our observations on the behavior of the simultaneous XTR double exponentiation, we show how simple power analysis attack helps reduce the search space for two input exponents. Our experimental results show that it takes $U^{1.25}$ tries for determining both exponents where $U = \max(a, b)$ and a, b are the input exponents. Moreover we show that it takes $U^{0.625}$ tries for an adversary until he/she correctly finds the secret key used in two XTR single exponentiation algorithms presented in [16]. We also point out a calculation error in [14] and discuss the effectiveness of the Markov method in the attack described here.

Keywords: XTR, side channel attack, simple power analysis, Markov model

1 Introduction

Due to the index calculus method, traditional cryptosystems based on subgroup discrete logarithm problem have a large representation size, which is usually at least 1024 bits, even though they are based on a subgroup whose order is only about 2^{160}. Such a long representation size renders traditional cryptosystems disadvantageous both in bandwidth and computational efficiency.

In 2000, XTR cryptosystem was proposed by Lenstra and Verheul [9]. XTR stands for 'EC-STR' which is an abbreviation for 'Efficient and Compact Subgroup Trace Representation'. The security of XTR is based on traditional subgroup discrete logarithm problem in prime order $q|p^2 - p + 1$ subgroup of $GF(p^6)$, where $p^6 \approx 2^{1024}$. It uses trace representation over $GF(p^2)$ which is 1/3 of the traditional representation. Such compact representation allows efficient computations. Recently strong evidence that 1/3 is the best compression ratio achievable was shown in [1].

Not only is XTR very easy to implement, but it is also claimed to be as fast as elliptic curve cryptosystem (ECC) and significantly faster than RSA. ECC is based on very hard mathematics and its parameter selection is not simple. RSA is the most popular cryptosystem and it is easily

* This work was part of the comprehensive exam proposal of Jaewook Chung. Jaewook Chung passed the comprehensive exam on January 21, 2002.

understood, but it is not so efficient both in bandwidth and computational efficiency. XTR does not suffer any of such disadvantages found in ECC and RSA, but it uses twice longer public key size than ECC (with point compression). Hence XTR is a good compromise between ECC and RSA.

Since the work of Kocher et al. [7, 8], side channel attacks have become the most devastating attack on many implementations of cryptosystems. No matter how hard the underlying mathematical problem is, if not implemented properly, cryptosystems often succumb to side channel attacks. Recently there have been many articles on the side channel attacks on various cryptosystems on various hardware [7, 8, 11, 12]. Also there have been many countermeasures to side channel attacks [2, 4, 6, 15, 17, 5]. However, to the best of our knowledge, there have been no results reported on XTR cryptosystem.

In this report, we analyze the impact of side channel attack on XTR cryptosystem. In specific, we analyze the consequences of side channel attack on the simultaneous XTR double exponentiation algorithm using very reasonable assumptions. We first show that it is possible to reconstruct the sequence of XTR operations by observing power trace during exponentiations. Then we show how to guess the correct exponent pair used in simultaneous XTR double exponentiation algorithm using the algorithm's behavior. We also show why the Markov model [14] is not so useful in the attack described here. We extend our result to XTR single exponentiation algorithms.

2 Identifying Elementary XTR Operations

2.1 Assumptions

The side channel attack described in this report starts from the following very simple assumptions.

1. It is possible to distinguish long integer multiplications and modular reductions (or Montgomery reductions) using side channel information.
2. An adversary can somehow verify whether any given integer is the correct key or not.

The first assumption is clearly reasonable because long integer multiplication and modular reduction are significantly different algorithms. Note that this assumption is still reasonable, even if the Montgomery arithmetic is used. Since postponing (or combining) reductions is recommended for efficient implementation of $\mathrm{GF}(p^2)$ arithmetic [16], long integer multiplication algorithm should be used instead of the Montgomery multiplication algorithm (Algorithm 14.36 in [10]). Reduction should be done by the Montgomery reduction algorithm (Algorithm 14.32 in [10]). Note that the Montgomery reduction should be distinguishable from long integer multiplication since the former algorithm uses an additional single precision multiplication (Step 2.1 of Algorithm 14.32 in [10]) for each loop.

2.2 Elementary XTR Operations

In XTR exponentiation algorithms, operations such as c_{u+v}, c_{2u}, c_{3u}, c_{u+2}, c_{2u-1} and c_{2u+1} are required (See Fact 2.3 in [16]). However we observe that only 3 operations among them are sufficient to implement XTR exponentiations.

Definition 1. *Let $w,\, x,\, y,\, z \in GF(p^2)$. The elementary XTR operations are:*

1. $A(w,\, x,\, y,\, z) = w \cdot x - x^p \cdot y + z$.
2. $D(x) = x^2 - 2x^p$.
3. $T(x) = x^3 - 3x^{p+1} + 3$.

It is straightforward to verify that all non-trivial trace operations used in XTR cryptosystems are derived from elementary XTR operations as shown below. Note that computing the p-th power operation does not involve any computation, *i.e.*, $x^p = x_2\alpha + x_1\alpha^2$ where $x = x_1\alpha + x_2\alpha^2$, $x_1,\, x_2 \in GF(p)$, $p \bmod 3 \equiv 2$ and $\alpha^2 + \alpha + 1 = 0$.

$$\begin{array}{ll} c_{u+v} = A(c_u,\, c_v,\, c_{u-v},\, c_{u-2v}) \ , & c_{2u+1} = A(c_{u+1},\, c_u,\, c_1,\, c_{u-1}^p) \ , \\ c_{u+2} = A(c_{u+1},\, c_1,\, c_u,\, c_{u-1}) \ , & c_{2u-1} = A(c_{u-1},\, c_u,\, c_1^p,\, c_{u+1}^p) \ , \\ c_{2u} = D(c_u) \ , & c_{3u} = T(c_u) \ . \end{array} \tag{1}$$

2.3 Elementary XTR Operation Sequence

Let $m(\cdot)$ denote a long integer multiplication and $r(\cdot)$ a modular reduction by p (or a Montgomery reduction). According to the improved $GF(p^2)$ arithmetic shown in [16], $GF(p^2)$ operations are implemented as follows.

$$\begin{aligned} M(x,\, y) &= x \cdot y = r(t-u)\alpha + r(s-u)\alpha^2 \ , \\ &\quad s = m(x_1,\, y_1) \ , \quad t = m(x_2,\, y_2) \ , \\ &\quad u = m(x_1 + x_2,\, y_1 + y_2) - s - t \ , \\ S(x) &= x^2 = r(m(x_2,\, x_2 - 2x_1))\alpha + r(m(x_1,\, x_1 - 2x_2))\alpha^2 \ , \\ X(x,\, y,\, z) &= x \cdot z - y \cdot z^p \\ &= r(m(z_1,\, y_1 - x_2 - y_2) + m(z_2,\, x_2 - x_1 + y_2))\alpha \\ &\quad + r(m(z_1,\, x_1 - x_2 + y_1) + m(z_2,\, y_2 - x_1 - y_1))\alpha^2 \ , \end{aligned} \tag{2}$$

where $x = x_1\alpha + x_2\alpha^2$, $y = y_1\alpha + y_2\alpha^2$ and $z = z_1\alpha + z_2\alpha^2$ are in $GF(p^2)$, and $\alpha^2 + \alpha + 1 = 0$. Note that only long integer multiplications and modular reductions (or Montgomery reductions) are required for all 3 computations in $GF(p^2)$ as shown below. The integer operation sequence for each $GF(p^2)$ operation may vary depending on implementations. The following shows all possible integer operation sequences for all $GF(p^2)$ operations.

- M : *mmmrr*
- S : *mrmr* or *mmrr*
- X : *mmrmmr*, *mmmmrr* or *mmmrmr*

We easily observe that, in any case, $GF(p^2)$ operations can be clearly identified if an adversary can distinguish integer multiplications and modular reductions (or Montgomery reductions). Now we re-write elementary XTR operations in terms of M, S and X as follows.

$$\begin{aligned} A(w,\, x,\, y,\, z) &= X(w,\, y,\, x) + z \ , \\ D(x) &= S(x) - 2x^p \ , \\ T(x) &= M(x,\, S(x) - 3x^p) + 3 \ . \end{aligned} \tag{3}$$

Since the operations M, S and X are distinguishable, it is straightforward to see that the operations A, D and T are also distinguishable.

3 Features of XTR Exponentiation Algorithms

3.1 XTR Exponentiation Algorithms

Algorithms 1 through 4 are the XTR exponentiation algorithms that were proposed in [16, 9]. We describe them here for our reference in the later sections.

Algorithm 1. XTR-SINGLE-EXP1 $(c,\, v)$

```
INPUT: $c \in GF(p^2)$ and $v = \sum_{i=0}^{r-1} v_i 2^i \in \mathbb{Z}_{\geq 0}$, where $v_{r-1} = 1$.
OUTPUT: $(c_{2v},\, c_{2v+1},\, c_{2v+2})$.
 1. $(c_{y-1},\, c_y,\, c_{y+1}) \leftarrow (3,\, c,\, D(c))$.
 2. For $i = r-1$ down to 0,
    2.1 If $v_i = 0$,
          $(c_{y-1},\, c_y,\, c_{y+1}) \leftarrow (D(c_{y-1}),\, A(c_{y-1},\, c_y,\, c^p,\, c^p_{y+1}),\, D(c_y))$.
    2.2 If $v_i = 1$,
          $(c_{y-1},\, c_y,\, c_{y+1}) \leftarrow (D(c_y),\, A(c_{y+1},\, c_y,\, c,\, c^p_{y-1}),\, D(c_{y+1}))$.
 3. Return $(c_{y-1},\, c_y,\, c_{y+1})$.
```

Algorithm 2. XTR-SINGLE-EXP2 $(c,\, n)$

```
INPUT: $c \in GF(p^2)$ $n = \sum_{i=0}^{r} n_i 2^i \in \mathbb{Z}_{\geq 0}$, where $n_r = 1$.
OUTPUT: $(c_{n-1},\, c_n,\, c_{n+1})$.
 1. If $n_0 = 0$,
    1.1 $v \leftarrow n/2 - 1$.
    1.2 $(c_{2v},\, c_{2v+1},\, c_{2v+2})$ = XTR-SINGLE-EXP1 $(c, v)$.
    1.3 $(c_{n-1},\, c_n,\, c_{n+1}) = (c_{2v+1},\, c_{2v+2},\, A(c_{2v+2},\, c_1,\, c_{2v+1},\, c_{2v}))$.
 2. Else,
    2.1 $v \leftarrow (n-1)/2$.
    2.2 $(c_{n-1},\, c_n,\, c_{n+1})$ = XTR-SINGLE-EXP1 $(c, v)$.
 3. Return $(c_{n-1},\, c_n,\, c_{n+1})$.
```

Algorithm 3. XTR-DOUBLE-EXP $(a,\, b,\, c_k,\, c_l,\, c_{k-l},\, c_{k-2l})$

```
INPUT: $a$, $b$, $c_k$, $c_l$, $c_{k-l}$ and $c_{k-2l}$, where $0 < a,\, b < q$.
OUTPUT: $c_{bk+al}$.
 1. $d = b$, $e = a$, $c_u = c_k$, $c_v = c_l$, $c_{u-v} = c_{k-l}$, $c_{u-2v} = c_{k-2l}$, $f_2 = 0$ and $f_3 = 0$.
 2. If both $d$ and $e$ are even, replace $(d, e)$ by $(d/2, e/2)$ and $f_2$ by $f_2 + 1$.
 3. If both $d$ and $e$ are divisible by 3, replace $(d, e)$ by $(d/3, e/3)$ and $f_3$ by $f_3 + 1$.
 4. While $d \neq e$ update $(d, e, c_u, c_v, c_{u-v}, c_{u-2v})$ according to Table 1
 5. Compute $c_{u+v} = A(c_u,\, c_v,\, c_{u-v},\, c_{u-2v})$
 6. Compute $c_{d(u+v)}$ using XTR-SINGLE-EXP2 $(c_{u+v},\, d)$ or XTR-SINGLE-EXP3 $(c_{u+v},\, d)$.
 7. Compute $c_{2^{f_2} d(u+v)} = D^{f_2}(c_{d(u+v)})$
 8. Return $c_{bk+al} = c_{3^{f_3} 2^{f_2} d(u+v)} = T^{f_3}(c_{2^{f_2} d(u+v)})$
```

Algorithm 4. XTR-SINGLE-EXP3 (c, u)

```
INPUT: $c \in GF(p^2)$ and $0 < u < q$.
OUTPUT: $c_u$.
```

Table 1. Update Table for Step 4 in Algorithm 3

Notation	Condition	Update $(d, e, c_u, c_v, c_{u-v}, c_{u-2v})$	XTR Seq.	# Muls.
If $d > e$				
S_0	*i.* If $d \le 4e$	$(e, d-e, \mathbf{c_{u+v}}, c_u, c_v, c_{v-u})$	A	3
S_1	*ii.* Else if d is even	$(\frac{d}{2}, e, \mathbf{c_{2u}}, c_v, \mathbf{c_{2u-v}}, \mathbf{c_{2(u-v)}})$	ADD	7
S_2	*iii.* Else if e is odd	$(\frac{d-e}{2}, e, \mathbf{c_{2u}}, \mathbf{c_{u+v}}, c_{u-v}, \mathbf{c_{-2v}})$	ADD	7
S_3	*v.* Else (e is even)	$(\frac{e}{2}, d, \mathbf{c_{2v}}, c_u, c_{2v-u}, \mathbf{c_{2(v-u)}})$	DD	4
If $d < e$				
S_4	*i.* If $e \le 4d$	$(d, e-d, \mathbf{c_{u+v}}, c_v, c_u, c_{u-v})$	A	3
S_5	*ii.* Else if e is even	$(\frac{e}{2}, d, \mathbf{c_{2v}}, c_u, c_{2v-u}, \mathbf{c_{2(v-u)}})$	DD	4
S_6	*iii.* Else if d is odd	$(\frac{e-d}{2}, d, \mathbf{c_{2v}}, \mathbf{c_{u+v}}, c_{v-u}, \mathbf{c_{-2u}})$	ADD	7
S_7	*vi.* Else (d is even)	$(\frac{d}{2}, e, \mathbf{c_{2u}}, c_v, \mathbf{c_{2u-v}}, \mathbf{c_{2(u-v)}})$	ADD	7

```
1. $a = \lfloor \frac{3-\sqrt{5}}{2} u \rfloor$ and $b = u - a$.
2. $c_{a+b}$ = XTR-DOUBLE-EXP ($a$, $b$, $c$, $c$, 3, $c^p$).
3. Return $c_u = c_{a+b}$.
```

Remark 1. Table 1 lists only the mandatory sub-steps of Algorithm 3, since optional steps [16] are not considered in our analysis. The first column of Table 1 lists the short notation for each sub-step. The second last column of Table 1 lists the corresponding elementary XTR operation sequence for each sub-step. Note that the elementary XTR operation sequences for sub-steps S_1, S_2, S_6 and S_7 may vary depending on implementations. However, for security reason, it is better to implement the way we show in Table 1. In our setting, sub-steps S_1, S_2, S_6, S_7, S_0S_3, S_0S_5, S_4S_3 and S_4S_5 are indistinguishable.

3.2 Required Information to Recover Input Exponents a and b for Algorithm 3

Theorem 1. *Suppose that an adversary knows sub-step sequence along with f_2, f_3 and d value at step 6 in Algorithm 3. Then he/she can recover two input exponents a and b.*

Proof. According to Table 1, in each sub-step S_i, (d, e)-pair is linearly transformed by matrix T_i, where transformation matrices T_i's are defined as follows.

$$T_0 = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}, \quad T_1 = \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1/2 & -1/2 \\ 0 & 1 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 0 & 1/2 \\ 1 & 0 \end{bmatrix},$$
$$T_4 = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \quad T_5 = \begin{bmatrix} 0 & 1/2 \\ 1 & 0 \end{bmatrix}, \quad T_6 = \begin{bmatrix} -1/2 & 1/2 \\ 1 & 0 \end{bmatrix}, \quad T_7 = \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4}$$

Let us define two more matrices, F_2 and F_3,

$$F_2 = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}, \quad F_3 = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix}. \tag{5}$$

Suppose the known sub-step sequence is $S_{l_1} S_{l_1} \ldots S_{l_n}$, where $l_i \in [0, 7]$. Then it is easy to see that,

$$\begin{bmatrix} d \\ d \end{bmatrix} = T_{l_n} T_{l_{n-1}} \cdots T_{l_1} \cdot F_3^{f_3} \cdot F_2^{f_2} \cdot \begin{bmatrix} b \\ a \end{bmatrix} \quad . \tag{6}$$

Since the matrices T_i's for $i = 0 \ldots 7$, F_2 and F_3 are all invertible, the unique solution $(b, a)^T$ exists. $\square$

3.3 Observations

In our work, we have made some important observations on the statistical behavior of Algorithm 3, the simultaneous XTR double exponentiation algorithm. These observations are based on the ideas proposed in [18, 14] to attack MIST and ECC.

Proposition 1. *The probability of a sub-step occurring in a sub-step sequence is highly dependent on its past sub-step sequence.*

Since there does not appear to be an easy way to analyze the statistical behavior of Algorithm 3, we have performed an experiment to determine the values of $P(S_i|S_j)$ for all combinations of $i, j \in [0, 7]$, where $P(S_i|S_j)$ denotes the probability of the next sub-step S_i assuming the current sub-step S_j. The sub-step transition matrix $B = [b_{(i,j)}]$ shows the results, where $b_{(i,j)} = P(S_j|S_i)$ for $i, j \in [0, 7]$.

$$B = \begin{bmatrix} 0.336 & 0.062 & 0.038 & 0.056 & 0.507 & 0.0 & 0.0 & 0.0 \\ 0.394 & 0.302 & 0.304 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.453 & 0.273 & 0.274 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.503 & 0.497 \\ 0.505 & 0.043 & 0.090 & 0.087 & 0.274 & 0.0 & 0.0 & 0.0 \\ 0.499 & 0.248 & 0.253 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.054 & 0.470 & 0.473 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.503 & 0.497 \end{bmatrix} . \tag{7}$$

Note that more than half of the elements in matrix B are zeros. This implies that some combinations of sub-step sequences never occur in sub-step sequence. Even though there does not appear to be an easy way to prove all the probabilities in matrix B, all the zero occurrences can be easily proven. Note that only the zero occurrences in B are important. It will be shown in Section 5 that the exact values of probability are not important.

We have determined $P(Q_1 Q_2 \cdots Q_n)$ up to $n = 4$ to identify impossible sub-step sequences, and we list in Table 2 all the impossible sub-step sequences that are not observed from matrix B.

We can also determine how frequently each sub-step is executed, by computing the steady state vector of B. Note that we have to ignore the 6-th row and column, since otherwise the matrix B is not irreducible.

Note that the probability $P(S_5)$ is shown to be zero in Table 3, but it is not exactly zero. $P(S_5)$ is extremely small and very close to zero. This is because the sub-step S_5 can appear only

Table 2. Impossible Sub-step Sequences

Impossible Sub-step Sequences	
$S_0S_4S_4S_4$	$S_4S_4S_4S_4$
$S_1S_0S_0$, $S_1S_0S_1$ $S_1S_0S_2$, $S_1S_0S_3$, $S_1S_0S_4S_1$	$S_5S_0S_0$, $S_5S_0S_1$ $S_5S_0S_2$, $S_5S_0S_3$, $S_5S_0S_4S_1$
$S_2S_0S_1$, $S_2S_0S_2$, $S_2S_0S_3$ $S_2S_0S_0S_2$, $S_2S_0S_0S_4$, $S_2S_0S_4S_1$	$S_6S_0S_1$, $S_6S_0S_2$, $S_6S_0S_3$ $S_6S_0S_0S_2$, $S_6S_0S_0S_4$, $S_6S_0S_4S_1$
$S_3S_6S_0S_0$, $S_3S_7S_6S_0$	$S_7S_6S_0S_0$, $S_7S_7S_6S_0$

Table 3. Sub-step Probabilities

Sub-step Probabilities			
$P(S_0) = 0.361$	$P(S_1) = 0.129$	$P(S_2) = 0.132$	$P(S_3) = 0.042$
$P(S_4) = 0.251$	$P(S_5) = 0.000$	$P(S_6) = 0.043$	$P(S_7) = 0.042$

once at the beginning and it does if and only if the input exponents a and b satisfies the condition, $a > 4b$. We observe from the matrix B that no sub-steps can ever be followed by S_5. Hence S_5 must appear at the beginning of sub-step sequence.

4 Side Channel Attack

In this section, we discuss how to obtain the required information as discussed in Section 3.2, to recover a and b using elementary XTR operation sequence.

4.1 Determining f_2 and f_3

Determining f_2 and f_3 is quite simple. In steps 7 and 8, the algorithm uses f_2 times of operation D and f_3 times of T. Hence we only need to count. Alternatively we can determine f_2 and f_3 from steps 2 and 3 by counting the number of bit shifts and divisions by 3, though this seems to be harder.

4.2 Determining Sub-step Sequence in Step 4

Apparently there seems to be no easy way to analyze the expected number of tries until the correct key pair is found. Moreover the fact that there are many impossible sub-step sequences listed in 2 makes things even more complicated. Nevertheless, if we take into account only the impossible sub-step sequences $S_1S_0S_3$, $S_2S_0S_3$, $S_5S_0S_3$, $S_6S_0S_3$ in Table 2, we can easily compute the expected number of possible exponent pairs for a randomly given elementary XTR operation sequence. For example, let S_1 be the current sub-step and the next elementary XTR sequence to be processed is ADD, then there are only 2 possible choices for determining the next sub-step: S_1 or S_2. The sub-step sequence S_0S_3 also leads to ADD but $S_1S_0S_3$ is an impossible case.

We give all the detailed calculations in Table 4. The third column of Table 4 lists $P(q|S_i)$'s, the probabilities of observing elementary XTR operation sequence q after sub-step S_i, for $i = 0, \ldots 7$ and $q = A(A)$ (an A which is not directly followed by DD), ADD (which comes from a single sub-step or two consecutive sub-steps) and DD. The fourth column lists the number of possible choices for the next sub-step given a current sub-step S_i and one of $A(A)$, ADD or DD, the next elementary XTR operation sub-sequence.

Table 4. Expected Number of Choices at a Sub-step

S_i	q	$P(q\|S_i)$	# Choices	$E(\#\text{ Choices}\|S_i)$	$E(\#\text{ Choices}\|S_i) \cdot P(S_i)$
S_0	$A(A)$	0.780	2		
	ADD	0.163	4	2.269	0.819
	DD	0.057	1		
S_1	$A(A)$	0.394	1	1.606	0.207
	ADD	0.606	2		
S_2	$A(A)$	0.450	1	1.550	0.205
	ADD	0.550	2		
S_3	ADD	1.000	2	2.000	0.084
S_4	$A(A)$	0.700	2		
	ADD	0.212	4	2.337	0.587
	DD	0.088	1		
S_5	$A(A)$	0.501	1	1.499	0.000
	ADD	0.499	2		
S_6	$A(A)$	0.054	1	1.946	0.084
	ADD	0.946	2		
S_7	ADD	1.000	2	2.000	0.084
Expected # of choices for each sub-step					2.069

Using the fact that there are $1.356 \log_2 U$, where $U = \max(a,\, b)$, sub-steps during the execution of Algorithm 3, we conclude that the expected size of search space is $2.069^{1.356 \log_2 U} = U^{1.422}$ for a randomly given sub-step sequence. If we utilize more impossible cases from Table 2, the search space will be reduced significantly.

To validate our calculation, we have experimentally determined the average number of tries required for a randomly given XTR operation sequence. Figure 1 shows the results.

The slope of the line, for which only 4 impossible cases are considered, is determined to be about 1.418 showing that our calculation is correct. The slope of the other line, for which we used all impossible cases in Table 2, is determined to be about 1.25. Even though the range of n is limited to 18 only, the graph looks straight enough to believe that it takes about $U^{1.25}$ tries until the correct key pair is found.

4.3 Determining d at Step 6

Let us denote d at step 6 with g. Suppose an adversary somehow knows that $g = 1$, then he/she will just ignore step 6 since no computation occurs there. However if $g \neq 1$ he/she has to determine

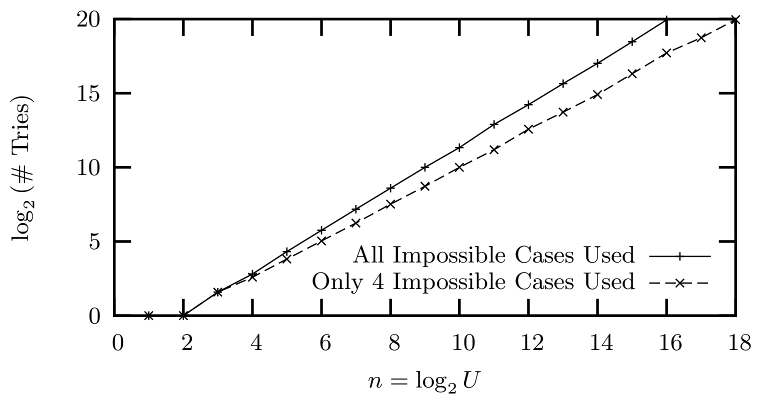

Fig. 1. The Average Number of Tries Required and the Size of Search Space

the exact value of it by attacking Algorithm 2 or Algorithm 4. This could be an obstacle especially if g is a large value. However we have found that g is very small in most cases. In fact, we have rigorously calculated that $g = 1$ occurs about 91.2% of the time and the expected value is only 34.14 assuming 160-bit input exponents. See Appendix A for more details.

Suppose that Algorithm 2 is used at step 6. It is easy enough to see from the description of Algorithm 2 that an adversary can determine the bit-length and the least significant bit of g. Note that to make Algorithm 1 resistant against side channel attack, the elementary XTR operation sequences for the two cases in step 2 should be identical [9]. The bit-length of g can be obtained by counting the number of repetitions of ADD, DAD or DDA during the execution of Algorithm 1. If Algorithm 4 is used at step 6, since Algorithm 4 makes a recursive call to Algorithm 3, an adversary has to attack Algorithm 3 recursively.

4.4 Determining Boundaries Between Steps

We now discuss whether it is possible to determine the exact boundaries between steps in Algorithm 3 by observing an elementary XTR operation sequence.

- **When d = 1 at step 6**

  If $d = 1$, no elementary XTR operation is performed at step 6. The elementary XTR operation sequence from step 5 to the end of the algorithm will then look like:

  $$A\underbrace{DD\cdots D}_{f_2}\underbrace{TT\cdots T}_{f_3}.$$

  Since operation T can only occur at step 8, step 8 can be clearly identified. Whether $f_2 = 0$ or not, one needs to look for the last A. Then it must be from step 5. Operation D's between A and T are from step 7. Since $d = 1$ happens about 91.2% of the time, step boundaries can be found easily in most cases.

- **$\mathbf{d \neq 1}$ and Algorithm 2 is used for step 6**
  Suppose step 2.1 and step 2.2 in Algorithm 1 leave elementary XTR operation sequence ADD (DAD and DDA are also possible but they all lead to the same result). Then the elementary XTR operation sequence from step 5 to the end of the algorithm will look like:

$$AD\underbrace{ADDADD\cdots ADD}_{\lfloor \log_2(d) \rfloor}\underbrace{DD\cdots D}_{f_2}\underbrace{TT\cdots T}_{f_3}.$$

  An adversary can clearly determine that $d \neq 1$, if he/she observes the sequence AD followed by A. Note that D operation always occurs two consecutive times in all other steps except for step 7. Note also that the operation A from step 1.3 of Algorithm 2 never occurs, since d at step 6 never can be a multiple of 2 (see Appendix A for details). Therefore there is no difficulty in determining boundaries between steps.
- **$\mathbf{d \neq 1}$ and Algorithm 4 is used for step 6**
  In this case, there appears to be no easy way to locate the boundaries between steps and an exhaustive search is needed for the value of d by trying from the smallest possible one. Note that the adversary only has to try values that are not multiple of 2 or 3 and larger d has lower probability.

Therefore we conclude that boundaries between steps are exactly identified if Algorithm 2 is used. If Algorithm 4 is used for step 6, it is not clear how to determine the boundaries. In that case, exhaustive search has to be done. Note that this does not seriously harm the effectiveness of the attack. Using the fact that $d = 1$ occurs 91.2% of the time (See Appendix A) and that the probability significantly decreases as d increases, it is expected that an attacker should guess the exact d by trying only a few values.

5 Is the Markov Method Useful?

In [14], Oswald shows how the Markov chain method can help enhancing simple power analysis attacks on elliptic curve point multiplication algorithms. However such a method is not very useful for the attack described in this report. Here we give the reasons.

In [14], it was possible to partition an elliptic curve operation sequence into several partitions of small lengths such that only 3 bit-patterns are possible for each partition. Using this weakness, Oswald could prune significant number of spurious keys. However, apparently there is no such a weakness found in Algorithm 3.

In fact, we have found that the Markov method itself may not be so effective in practice as it is claimed. According to [14], an elliptic curve operation sequence resulted from the point multiplication algorithm [13] are broken into length-l partitions, where for each partition only 3 bit patterns are possible. Hence, an adversary has to try all $3^{3n/2l}$ keys in the worst case, where n is the bit length of key. According to [14], one of the 3 bit patterns occurs with probability 1/2 and the others 1/4 each, However, it has been incorrectly reported in [14] that the expected number of tries is $2^{3n/2l}$ when an adversary takes into account the probabilities for the 3 bit patterns. This incorrect calculation gives us an impression that the Markov method, especially its 'Key Testing Phase', brings about significant *exponential* improvement. However, we show this is not true.

Suppose that there are k partitions. We assume that an adversary always tries from the highest probable keys to the least probable one. When $k = 1$, the expected number of tries in this case is $1 = 1 \cdot 1/2 + 2 \cdot 1/4$ (If the first two tries fail, the third one must be the correct key). Now, suppose that there are two partitions, *i.e.*, $k = 2$. Note carefully that an adversary can only test complete keys but not partial key bits, since it is assumed that the adversary only knows a plaintext/ciphertext pair. The expected number of tries in this case is calculated as $3.875 = 1 \cdot (1/2)^2 + (2+3+4+5) \cdot (1/2)(1/4) + (6+7+8) \cdot (1/4)^2$.

Since it is very hard to generalize this calculation for any k, we used a computer program to do this calculation up to $k = 50$ partitions and Figure 2 shows the result.

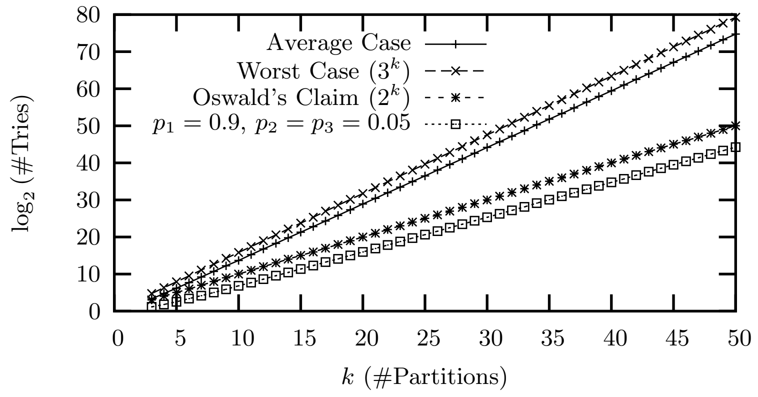

Fig. 2. Number of Tries

We easily see from Figure 2 that the difference between the average and the worst cases is not very significant and our calculation is far from that of [14]. For $l = 16$ and $n = 163$, the number of partitions is 15.28. In such a case, the expected number of tries is about 2^{22} as shown in Figure 2. However in [14], it has been incorrectly reported that the expected number of tries is $2^{15.28}$.

Nevertheless, Figure 2 shows an evidence that the Markov method does result in a significant exponential improvement for an extreme case where one pattern occurs with the probability 0.9, and the others 0.05. In such a case, as seen from Figure 2 the Markov method significantly saves the number of tries. From this result, we deduce that the Markov method is useful when the probabilities are significantly 'skewed'.

The Markov method may not be useful for the attack described in this report, as the probabilities for sub-steps (see Table 3 and the matrix B in section 3.3) do not appear to be skewed enough. Moreover it would be infeasible to sort all the exponential number of spurious keys according to their probabilities.

6 Extension to Single Exponentiation Algorithms

Our cryptanalysis result can be directly applied to the XTR single exponentiation algorithm with precomputation (Algorithm 4.2 in [16]) and Algorithm 4. Note that the side channel attack is meaningful only if it is applied to single exponentiation algorithms, but not double exponentiation algorithms. Since double exponentiation algorithms are usually used in signature verification, all the input values to a double exponentiation algorithm are public information. However, Algorithm 3 is used for the efficient implementation of single XTR exponentiation algorithms [16]. Hence the side channel attack on Algorithm 3 is meaningful.

– Algorithm 4.2 in [16]
  Suppose the input exponent U is n-bit long. Then the exponent is split into two $n/2$-bit integers. These two integers are the input exponents to Algorithm 3. Therefore our cryptanalysis is expected to take $\sqrt{U}^{1.25} = U^{0.625}$ tries until the correct exponent is found.
– Algorithm 4
  Suppose the input exponent U is n bits long. Exponent is split into two integers a and b at step 1, but they have almost the same bit length as U. However according to Proposition 5.2 in [16], for the first $\log_{\Phi} \sqrt{U}$ iterations, where $\Phi = (3 - \sqrt{5})/2$, only S_0 is executed and the values of d and e reduce to about $\sqrt{U}$ after first $\log_{\Phi} \sqrt{U}$ iterations. This means one only needs to attack the later half of the algorithm. Therefore it takes about $U^{0.625}$ tries until the correct exponent is found.

7 Conclusions

In this report, we have analyzed the impact of a side channel attack on XTR exponentiation algorithms under a reasonable assumption that an adversary has the ability to distinguish between multiplication and modular reduction (or Montgomery reduction). We have shown that to find the correct exponent pair (a, b) for Algorithm 3 an adversary is expected to make about $U^{1.25}$ tries on average where $U = \max(a, b)$. It immediately follows that an adversary is expected to make $U^{0.625}$ tries on average until he/she finds the correct input exponent to Algorithm 4.2 in [16] and Algorithm 4. We also noted that the Markov model presented in [14] may not be useful in the attack described in this report.

Note that $U^{0.625}$ is too large to make the cryptanalysis feasible. The attack shown here is far worse than well-known square-root type algorithms (Baby-Step-Giant-Step or Pollard's Rho algorithms). However this result is obtained under a very simple assumption. More sophisticated attacks like differential side channel attacks may be able to extract more information from side channel leakages, because the XTR exponentiation algorithms are not randomized.

A Two Proofs on d at Step 6

In this appendix, let us denote d at step 6 of Algorithm 3 with g and (d, e) after step 3 with (d', e'), as we did before. Then it is not too difficult to see that $g = \gcd(d', e')$. In the following subsections, we rigorously calculate $Pr(\gcd(d', e') = 1)$ and the expected value of $\gcd(d', e')$. In our computations, we assume that $1 \leq d', e' \leq n$, unless otherwise specified.

A.1 Probability of $g = 1$

Suppose that we pick two integers a and b independently and randomly from interval $[1,\, n]$. Then, for large n, it is easily seen that the probability that p divides $\gcd(a,\, b)$ is,

$$P(p|\gcd(a,\, b)) = P(p|a)P(p|b) = \left(\frac{\lfloor n/p \rfloor}{n}\right)^2 \approx \frac{1}{p^2} \ , \tag{8}$$

where $p \in [1,\, n]$. We define p_i to be the i-th prime number, *i.e.*, $p_1 = 2$, $p_2 = 3$, *etc.* Then,

$$P(g = 1) = \prod_{i=3}^{s} P(p_i \nmid \gcd(a,\, b)) \approx \prod_{i=3}^{s} \left(1 - \frac{1}{p_i^2}\right) \ , \tag{9}$$

where p_s is the largest prime number in $[1,\, n]$. Note that the product in (9) begins from $i = 3$, since there is no common factor of 2 or 3 between d' and e'.

It is well-known that the probability of choosing two co-primes from infinite interval is $[\zeta(2)]^{-1} = 6/\pi^2$, where $\zeta(z)$ is the Riemann zeta function. For large n,

$$P(\gcd(a,\, b) = 1) \approx \prod_{i=1}^{s} \left(1 - \frac{1}{p_i^2}\right) \approx \frac{6}{\pi^2} \ . \tag{10}$$

Therefore,

$$\begin{aligned} P(g = 1) &\approx \frac{P(\gcd(a,\, b) = 1)}{(1 - 1/p_1^2)(1 - 1/p_2^2)} \\ &\approx \frac{6/\pi^2}{(1 - 1/2^2)(1 - 1/3^2)} \approx \frac{9}{\pi^2} = 0.91189 \ . \end{aligned} \tag{11}$$

A.2 Expected value of d

Let $E(x)$ denote the expected value of x. Then,

$$\begin{aligned} E(g) &= \sum_{x=1}^{n} x \cdot P(\gcd(g) = x) \\ &= \sum_{p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s} \le n} p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s} P(g = p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s}) \ , \end{aligned} \tag{12}$$

where $0 \le k_i \le \lfloor \log_{p_i} n \rfloor$ for $i = 3, \ldots, s$. Since having $p_i^{k_i}$ as a common factor and $p_j^{k_j}$ as a common factor are independent events if $i \ne j$,

$$\begin{aligned} P(g = p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s}) &= \prod_{l=3}^{s} P(p_l^{k_l} | g \cap p_l^{k^l + 1} \nmid g) \\ &\approx \prod_{l=3}^{s} \left(\frac{1}{p_l^2}\right)^{k_l} \cdot \left(1 - \frac{1}{p_l^2}\right) \ . \end{aligned} \tag{13}$$

Substitute (13) in (12) to get,

$$\begin{aligned} E(g) &\approx \sum_{p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s} \le n} \prod_{l=3}^{s} \frac{1}{p_l^{k_l}} \cdot \left(1 - \frac{1}{p_l^2}\right) \\ &\approx \frac{9}{\pi^2} \cdot \sum_{p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s} \le n} \frac{1}{p_3^{k_3} p_4^{k_4} \cdots p_s^{k_s}} \\ &\approx \frac{9}{\pi^2} \cdot Z_n \ , \end{aligned} \tag{14}$$

where,

$$Z_n = \sum_{i=0}^{\lfloor (n-1)/6 \rfloor} \frac{1}{6i+1} + \sum_{i=0}^{\lfloor (n-5)/6 \rfloor} \frac{1}{6i+5} \ . \tag{15}$$

Let H_n be the n-th harmonic number, $H_n = \sum_{i=1}^{n} 1/i$ and let H'_n be defined as,

$$H'_n = \sum_{i=1}^{n} \frac{(-1)^{k+1}}{k} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots \ . \tag{16}$$

It is well known that $H'_n = \ln 2 + (-1)^n [H_{(n-1)/2} - H_{n/2}] \approx \ln 2$ for large n. Then it is not too hard to prove that,

$$Z_n \approx \frac{H_n}{2} - \frac{H_{n/3}}{6} + \frac{\ln 2}{3} \ . \tag{17}$$

Since $H_n \approx \ln n + \gamma + 1/2n$, where γ is a Euler-Mascheroni constant,

$$Z_n \approx \frac{\ln(12n^2)}{6} + \frac{\gamma}{3} \ . \tag{18}$$

The numerical value of γ is,

$$\gamma \approx 0.5772156649015328606065120900824024310422\ldots .$$

However, since Z_n is a partial sum of harmonic series, error may accumulate due to the approximation in (8). If we use exact probability $P(p|a) = \lfloor n/p \rfloor / n$, we have better approximation for Z_n, which we denote Z'_n.

$$\begin{aligned} Z'_n &= \sum_{p_3^{k_3} p_4^{k_4} \cdot p_s^{k_s} \le n} \prod_{i=3}^{s} \frac{\lfloor n/p_i \rfloor^{k_i}}{n^{k_i}} \\ &\approx \frac{1}{n} \left(\sum_{i=0}^{\lfloor (n-1)/6 \rfloor} \left\lfloor \frac{n}{6i+1} \right\rfloor + \sum_{i=0}^{\lfloor (n-5)/6 \rfloor} \left\lfloor \frac{n}{6i+5} \right\rfloor \right) \ . \end{aligned} \tag{19}$$

Then the difference between Z_n and Z'_n is,

$$\begin{aligned} E_n &= Z_n - Z'_n \\ &\approx \frac{1}{n}\left(\frac{n \bmod 5}{5} + \frac{n \bmod 7}{7} + \frac{n \bmod 11}{11} + \frac{n \bmod 13}{13} + \cdots\right) . \end{aligned} \tag{20}$$

Due to de la Vallée Poussin [3],

$$\gamma = \lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\left\lfloor \frac{n}{i} \right\rfloor - \frac{n}{i}\right) . \tag{21}$$

Using this fact, it is not too hard to prove that $E_n \approx (1-\gamma)/3$. Therefore,

$$E(g) \approx \frac{9}{\pi^2} \cdot Z'_n = \frac{3}{\pi^2}\left(\frac{\ln(12n^2)}{2} + 2\gamma - 1\right) . \tag{22}$$

Hence, for $n = 2^{160}$, $E(g) \approx 34.135$ only!

References

1. Wieb Bosma, James Hutton, and Eric R. Verheul. Looking beyond xtr. In *Advances in Cryptology - ASIACRYPT 2002*, LNCS 2501, pages 46–63. Springer-Verlag, 2002.
2. J. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems - CHES '99*, LNCS 1717, pages 292–302. Springer-Verlag, 1999.
3. C. J. de la Vallée Poussin. Untitled communication. *Annales de la Soc. Sci. Bruxelles*, 22:84–90, 1898.
4. M. Hasan. Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems. In *Cryptographic Hardware and Embedded Systems - CHES 2000*, LNCS 1965, pages 93–108. Springer-Verlag, 2000.
5. T. Izu and T. Takagi. A fast parallel elliptic curve multiplication resistant against side channel attacks. In *Public Key Cryptography - PKC 2002*, LNCS 2274, pages 280–296. Springer-Verlag, 2002.
6. M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography-an algebraic approach. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pages 377–390. Springer-Verlag, 2001.
7. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology - CRYPTO '96*, LNCS 1109, pages 104–113. Springer-Verlag, 1996.
8. P. Kocher, J. Jeffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99*, LNCS 1666, pages 388–397. Springer-Verlag, 1999.
9. Arjen K. Lenstra and Eric R. Verheul. The XTR public key system. In *Advances in Cryptology - CRYPTO 2000*, LNCS 1880, pages 1–19. Springer-Verlag, 2000.
10. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
11. T. Messerges. Using second-order power analysis to attack DPA resistant software. In *Cryptographic Hardware and Embedded Systems - CHES 2000*, LNCS 1965, pages 238–251. Springer-Verlag, 2000.
12. T. Messerges, E. Dabbish, and R. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51:541–552, 2002.
13. F. Morain and J. Olivos. Speeding up the computation on an elliptic curve using addition-subtraction chains. *RAIRO: R. A. I. R. O. Informatique Theorique et Applications/Theoretical Informatics and Applications*, 24:119–129, 1990.

14. Elisabeth Oswald. Enhancing simple power-analysis attacks on elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, LNCS 2523, pages 82–97. Springer-Verlag, 2002.
15. Elisabeth Oswald and Mangred Aigner. Randomized addition subtractions chains as a countermeasure against power attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pages 39–50. Springer-Verlag, 2001.
16. Martijn Stam and Arjen K. Lenstra. Speeding up XTR. In *Advances in Cryptology - ASIACRYPT 2001*, LNCS 2248, pages 125–143. Springer-Verlag, 2001.
17. C. D. Walter. Sliding windows succumbs to Big Mac attack. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pages 286–299. Springer-Verlag, 2001.
18. C. D. Walter. Some security aspects of the MIST randomized exponentiation algorithm. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, LNCS 2523, pages 276–290. Springer-Verlag, 2002.