SPA on MIST Exponentiation Algorithm with Multiple Computational Sequences

Chien-Ning Chen, Jheng-Hong Tu, Sung-Ming Yen

SPA on MIST Exponentiation Algorithm with Multiple Computational Sequences*

Chien-Ning Chen[1], Jheng-Hong Tu[2], and Sung-Ming Yen[2]

[1] Physical Analysis & Cryptographic Engineering (PACE)
Nanyang Technological University, Singapore
chienning@ntu.edu.sg
[2] Laboratory of Cryptography and Information Security (LCIS)
Dept of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan
{jhtu,yensm}@csie.ncu.edu.tw

Abstract. The MIST algorithm is a randomized version of the division chain exponentiation algorithm and is a side-channel countermeasure. When analyzing the MIST algorithm by ordinary simple power analysis (with only one square-multiply sequence obtained), an attacker cannot retrieve the secret exponent due to the ambiguous relationship between the square-multiply sequence and the computation. We point out the MIST algorithm is still vulnerable to simple power analysis observing multiple power consumption traces and propose a practical method with detailed steps to deduce the secret exponent from multiple square-multiply sequences. Further countermeasures such as exponent blinding are required to prevent the analysis proposed in this paper.

Keywords: division chain, exponentiation, MIST algorithm, side-channel analysis, simple power analysis

1 Introduction

Exponentiation evaluation is a fundamental computation in most public-key cryptography. Conventional exponentiation algorithms are vulnerable to simple power analysis (SPA) [3], differential power analysis (DPA) [3] and other side-channel attacks, which break a cryptosystem by analyzing side-channel leakages from their implementation. SPA assumes that an attacker can distinguish power consumption patterns generated by different operations. When analyzing exponentiation computation, if a fast squaring algorithm is adopted, the attacker can distinguish the power consumption pattern of a squaring from that of a multiplication and then obtain a computational sequence composed of squarings and multiplications (abbreviated to "SM sequence"). Since a multiplication indicates

* Some part of this research was done while Chien-Ning Chen was a postdoctoral research fellow at the National Central University. The research of Jheng-Hong Tu and Sung-Ming Yen on this work were supported by the National Science Council of the Republic of China under contract NSC 101-2221-E-008-111-MY2.

a nonzero bit in the exponent, an attacker can deduce the secret exponent or partial information about the secret exponent from the SM sequence.

Many research results of countermeasures against side-channel attacks can be found in the literature. The square-and-multiply-always algorithm [2] is a well-known method against SPA. It performs a dummy multiplication when the corresponding bit of the exponent is zero, i.e., there are always a squaring and a multiplication in each iteration. An attacker cannot retrieve any information about the secret exponent from a regular SM sequence.

The MIST algorithm [4] proposed by Walter is an efficient exponentiation algorithm and also a countermeasure against side-channel attacks. Conventional exponentiation algorithms have exponents in binary and scan one or more bits per iteration. In contrast, the MIST algorithm represents exponents in mixed-radix. Mixed-radix representation means that the radix varies from position to position. An application of mixed-radix is the representation of dates and times, for example, 18429 seconds can be represented as $(0100011111111101)_2$ in binary or as $(5_{24}7_{60}9_{60})$, 5 hours 7 minutes 9 seconds in the mixed-radix clock system. The radixes in the MIST algorithm are randomly selected from small integers, and the radixes $\{2, 3, 5\}$ are used in [4]. Side-channel attacks which average a number of power consumption traces (e.g., DPA) are infeasible because the computation varies among each exponentiation evaluation.

The MIST algorithm is also immune to ordinary SPA due to the ambiguous relationship between an observed SM sequence and the computation. For example, the pattern `SM` (a squaring followed by a multiplication) in a sequence corresponds to either 1_2 (digit 1 with radix 2) or 0_3 (digit 0 with radix 3). While analyzing an SM sequence of the MIST algorithm with the exponent smaller than 2^n, the average number of exponents which may possibly generate this sequence is about $2^{3n/5}$ [6], i.e., there are $2^{3n/5}$ candidates of the exponent. A unique candidate might be deduced from multiple SM sequences, but it is infeasible to find the intersection of sets of $2^{3n/5}$ candidates. Walter claimed analyzing multiple sequences is an open problem, and Okeya [5] also observed that a small portion of the exponent reveals when analyzing multiple sequences. Besides the analysis by Walter, Oswald pointed out the security margin provided in [6] is wrong (might be lower) [8]. Sim *et al.* [9] also analyzed the MIST algorithm with an additional assumption which is similar to the doubling attack [7].

This paper presents a multi-sequence SPA on the MIST algorithm. We assume that an attacker can collect multiple power consumption traces generated by the MIST algorithm with the same exponent and then obtain SM sequences by the same technique used in ordinary SPA. In contrast with finding the intersection of sets of candidates after analyzing all sequences individually, the proposed method simultaneously analyzes multiple sequences and finds candidates satisfying all sequences. The proposed analysis employs the residue class operations and is a practical method to deduce the exponent directly from multiple SM sequences. Our implementation results show that around 25 sequences are sufficient to deduce a 1024-bit exponent in less than two hours by using a modern desktop PC in 2012.

The remaining parts of this paper are organized as follows. We review the MIST algorithm as well as some operations of residue class in Sect. 2. Section 3 describes our analysis, a theoretical attack with one SM sequence, followed by the extended attack exploiting multiple sequences. Section 4 provides the implementation results and also some tricks to reduce the memory requirement. Finally, Sect. 5 concludes this paper.

2 Preliminary and Background

2.1 MIST Exponentiation Algorithm

In 1998, Walter proposed an algorithm – exponentiation using division chain [1] which is a predecessor of the MIST exponentiation algorithm. In both algorithms, an exponentiation x^e is decomposed into $x^{(e \bmod d)} \times \left(x^d\right)^{\lfloor \frac{e}{d} \rfloor}$ where d is a small positive integer. The two small exponentiations $x^{(e \bmod d)}$ and x^d are computed by using a computational sequence $\{x, x^2, \cdots, x^d\}$ in which the exponents $\{1, 2, \cdots, d\}$ form an addition sequence containing $(e \bmod d)$ and d, i.e., $x^{(e \bmod d)}$ can be obtained during the evaluation of x^d without any cost. By repeating the above procedure, we have

$$\begin{aligned} x^e &= A_i \times {x_i}^{q_i} = (A_i \times {x_i}^{r_i}) \times \left({x_i}^{d_i}\right)^{\lfloor \frac{q_i}{d_i} \rfloor} \\ &= A_{i+1} \times {x_{i+1}}^{q_{i+1}} = \cdots \end{aligned} \tag{1}$$

with the base number $x_{i+1} = {x_i}^{d_i}$ initialized by $x_1 = x$; the accumulator $A_{i+1} = A_i \times {x_i}^{r_i}$ initialized by $A_1 = 1$; the quotient $q_{i+1} = \lfloor \frac{q_i}{d_i} \rfloor$ initialized by $q_1 = e$; and the remainder $r_i = q_i \bmod d_i$. The quotient q_i is the *remaining exponent* in the i-th iteration of the computation, and we have $q_i = \left((r_{\mathtt{MSB}})_{d_{\mathtt{MSB}}} \cdots (r_i)_{d_i}\right)$ in mixed-radix representation. The divisors d_i are small integers. The MIST algorithm in [4] selects d_i from $\{2, 3, 5\}$ randomly and is a randomized version of its predecessor (exponentiation using division chain).

```
INPUT: x and e
OUTPUT: x^e
01 A = 1, X = x, Q = e
02 while (Q > 0)
03   Randomly choose d from {2,3,5}
04   r = Q mod d
05   Compute A = A × X^r and X = X^d together†
06   Q = ⌊Q/d⌋
07 return A
```

† Referring to Fig.2 for the detailed computational sequence

Fig. 1. The MIST exponentiation algorithm

Figure 1 is the sketch of the MIST algorithm, and Fig. 2 provides the detailed computation of the MIST algorithm for various divisor-remainder pairs (abbreviated to "DR pair"). In the column of register sequence, `X` and `A` are the two variables used in the algorithm in Fig. 1, and `T` is the temporary variable. The computation `XX→T` denotes that the result of the multiplication (squaring) `X` × `X` is stored in `T`. The addition sequence and the square-multiply pattern (abbreviated to "SM pattern") generated by the computation are also provided.

DR pair (d_i, r_i)	Addition sequence	Register sequence	SM pattern
$(2, 0)$	12	XX→X	S
$(2, 1)$	12	XX→T, XA→A, (T→X)	SM
$(3, 0)$	123	XX→T, XT→X	
$(3, 1)$	123	XX→T, XA→A, XT→X	SMM
$(3, 2)$	123	XX→T, TA→A, XT→X	
$(5, 0)$	1235	XX→T, XT→X, XT→X	
$(5, 1)$	1235	XX→T, XA→A, XT→X, XT→X	SMMM
$(5, 2)$	1235	XX→T, TA→A, XT→X, XT→X	
$(5, 3)$	1235	XX→T, XT→X, XA→A, XT→X	
$(5, 4)$	1245	XX→T, TT→T, TA→A, XT→X	SSMM

Fig. 2. Computation in the MIST algorithm

Since the MIST algorithm selects the divisors d_i randomly, it is naturally immune to power analysis which averages over a number of power consumption traces, e.g., differential power analysis. Simple power analysis is the other category of power analysis which assumes the power consumption trace of multiplications is distinguishable from that of squarings. However, an attacker cannot uniquely determine the secret exponent by analyzing one SM sequence of the MIST algorithm due to the ambiguous relationship between an observed SM pattern and the exact computation. As shown in Fig. 2, each of the patterns `SM`, `SMM`, `SMMM` corresponds to two or three DR pairs, and the pattern `SSMM` corresponds to either the DR pair $(5, 4)$ or a squaring (`S`) followed by the pattern `SMM`. In [6], Walter pointed out that on average $2^{3n/5}$ candidates will remain for an n-bit exponent after analyzing one SM sequence.

2.2 Operations of Residue Class

The proposed analysis represents the candidates of the exponent as a collection of residue classes, which is much more efficient than enumerating and storing all candidates. In this paper, a residue class $\langle M, N\rangle$ denotes a set consisting of nonnegative integers congruent to the nonnegative integer N modulo the positive integer M $(0 \leq N < M)$, i.e., $\langle M, N\rangle = \{kM + N | k \in \mathbf{Z}, k \geq 0\}$. The following operations are used in the proposed analysis.

The first operation is *split* which splits a residue class $\langle M, N\rangle$ into ε subsets by expanding the modulus M to εM. Since $kM+N = \lfloor \frac{k}{\varepsilon} \rfloor \varepsilon M + (k \bmod \varepsilon)M + N$, $\langle M, N\rangle$ can be split into the collection of distinct subsets

$$\langle M, N\rangle = \bigcup_{\delta=0}^{\varepsilon-1} \langle \varepsilon M, \delta M + N\rangle ,$$

and $split(\langle M, N\rangle, \varepsilon, \delta) = \langle \varepsilon M, \delta M + N\rangle$ is defined as the δ-th subset of $\langle M, N\rangle$.

The second operation is *integer division*, i.e., dividing all numbers in a residue class $\langle M, N\rangle$ by a given divisor d and obtaining the integer quotients. Suppose $\varepsilon M = \mathrm{lcm}(M, d)$, the least common multiple of M and d. Since $\lfloor \frac{kM+N}{d} \rfloor = \lfloor \frac{(k'\varepsilon+\delta)M+N}{d} \rfloor = k'(\frac{\varepsilon M}{d}) + \lfloor \frac{\delta M+N}{d} \rfloor$ where $k' = \lfloor \frac{k}{\varepsilon} \rfloor$ and $\delta = k \bmod \varepsilon$, we have

$$\langle M, N\rangle \div d = \left\{ \lfloor \frac{kM+N}{d} \rfloor \middle| k = 1, 2, \cdots \right\} = \bigcup_{i=0}^{\varepsilon-1} \left\langle \frac{\varepsilon M}{d}, \lfloor \frac{iM+N}{d} \rfloor \right\rangle .$$

For example, $\langle 10, 7\rangle$ represents integers $\{7, 17, 27, \cdots\}$. When dividing by 6, $\langle 10, 7\rangle$ will be split to a collection $\langle 30, 7\rangle \cup \langle 30, 17\rangle \cup \langle 30, 27\rangle$, and then we have $\langle 10, 7\rangle \div 6 = \langle 5, 1\rangle \cup \langle 5, 2\rangle \cup \langle 5, 4\rangle$ which represents integers $\{1, 2, 4, 6, 7, 9, \cdots\}$.

The last two operations are to find *union* and *intersection*[3] of two residue classes. Before finding union or intersection, the moduli of the two sets should be expanded to their least common multiple by using the first operation. For example, $\langle 2, 1\rangle = \langle 6, 1\rangle \cup \langle 6, 3\rangle \cup \langle 6, 5\rangle$ and $\langle 3, 2\rangle = \langle 6, 2\rangle \cup \langle 6, 5\rangle$, and we have $\langle 2, 1\rangle \cup \langle 3, 2\rangle = \langle 6, 1\rangle \cup \langle 6, 2\rangle \cup \langle 6, 3\rangle \cup \langle 6, 5\rangle$ as well as $\langle 2, 1\rangle \cap \langle 3, 2\rangle = \langle 6, 5\rangle$.

3 Analysis to MIST by Using Residue Class

The MIST algorithm with a given exponent can generate various SM sequences due to its randomness. These SM sequences are the potential SM sequences associated with the given exponent, and one of them will be generated during each computation. On the other hand, an SM sequence generated by the MIST algorithm is associated with more than one exponents due to the ambiguous relationship between the SM pattern and the computation. The MIST algorithm with these exponents may possibly generate this SM sequence. When an SM sequence is observed by SPA, all these exponents associated with this observed SM sequence are candidates of the secret exponent, and we say these candidates satisfy this SM sequence. The proposed analysis will find the candidates simultaneously satisfying multiple SM sequences.

The observed SM sequences should be decomposed into small blocks before the analysis. Each block contains one SM pattern listed in Fig. 2 and corresponds to one iteration of the MIST algorithm. However, the pattern `SSMM` can

[3] The *intersection* operation is similar to solving the simultaneous congruences in the Chinese remainder theorem. However, in the proposed analysis, the moduli may not be relatively prime.

be interpreted as either an atomic pattern or two patterns `S-SMM`. It should be identified as a special block prior to other SM patterns. The proposed analysis deals with one block per iteration from the first to the last block of each SM sequence. Since there will be some special blocks `SSMM`, the block number in the proposed analysis is different from the iteration number in the MIST algorithm.

In the first step, the proposed analysis will find candidates of the exponent simultaneously satisfying the first block of every observed SM sequence.[4] These candidates can be divided into one or more residue classes, and each residue class is a candidate set. After analyzing the first i blocks of the observed SM sequences and obtaining the candidate sets satisfying the first i blocks, we inspect the subsets of each candidate set and then obtain the candidate sets satisfying the first $(i+1)$ blocks. After analyzing from the first to the last block of the observed SM sequences, we obtain the candidates of the exponent satisfying whole blocks of the observed SM sequences.

The candidates of the exponent in the proposed analysis are organized into one or more candidate sets (represented by using residue class) instead of enumerating all of them. A candidate set will contain more than one candidates when its modulus M is smaller than the upper limit of the exponent of the cryptosystem. The number of candidate sets does not indicate how much exhaustive search required, for example, we can use only one residue class $\langle 1, 0\rangle$ to represent all exponents smaller than the upper limit.

The details of the analysis is provided in the following subsections. The theoretical analysis with one SM sequence is introduced in Sect. 3.1, and the practical analysis of exploiting multiple SM sequences is presented in Sect. 3.2.

3.1 Single-sequence Analysis to MIST

Referring to the equation (1), the computation of the MIST algorithm is controlled by the DR pairs (d_i, r_i). The potential DR pairs associated with the block of SM patterns `S`, `SM`, `SMM`, `SMMM` are listed in Fig. 2, and the DR pairs associated with the special block `SSMM` are $(5, 4)$, $(6, 2)$, $(6, 4)$, and $(10, 0)$.[5] The variable for the potential DR pair as well as some other variables is defined below.

Definition 1. *Let* $\mathrm{DR}_i{}^{(\gamma)} = (d_i{}^{(\gamma)}, r_i{}^{(\gamma)})$ *be the* γ*-th potential DR pair associated with the* i*-th block of the SM sequence.*

Definition 2. *Let* $e_i{}^{(\alpha)} = \left\langle M_i{}^{(\alpha)}, N_i{}^{(\alpha)} \right\rangle$ *be the* α*-th candidate set of the exponent satisfying the first* $(i-1)$ *blocks of the SM sequence.*

[4] A candidate of the exponent satisfying the first i blocks of an observed SM sequence means there is at least one potential SM sequence associated with this candidate, of which the first i blocks are identical to the first i blocks of the observed SM sequence.

[5] The DR pair $(5, 4)$ is associated with the atomic pattern `SSMM`. In contrast, when a block `SSMM` is composed of two patterns `S-SMM`, the remaining exponent will satisfy $q_i \bmod 2 = 0$ as well as one of the three equations $\frac{q_i}{2} \bmod 3 = 1$, $\frac{q_i}{2} \bmod 3 = 2$, and $\frac{q_i}{2} \bmod 5 = 0$. The DR pairs associated with `S-SMM` are $(6, 2)$, $(6, 4)$, and $(10, 0)$.

If the exponent belongs to a candidate set ${e_i}^{(\alpha)}$, the remaining exponent q_i in the i-th iteration will belong to ${e_i}^{(\alpha)} \div ({d_1}^{(\gamma_1)} \cdots {d_{i-1}}^{(\gamma_{i-1})})$ for all enumerations of products $({d_1}^{(\gamma_1)} \cdots {d_{i-1}}^{(\gamma_{i-1})})$. These potential values ${e_i}^{(\alpha)} \div ({d_1}^{(\gamma_1)} \cdots {d_{i-1}}^{(\gamma_{i-1})})$ can be divided into several residue classes, and each one is a candidate set of the remaining exponent.

Definition 3. *Let* ${q_i}^{(\alpha\text{-}\beta)} = \left\langle {m_i}^{(\alpha\text{-}\beta)}, {n_i}^{(\alpha\text{-}\beta)} \right\rangle$ *be the* β*-th candidate set of the remaining exponent in the* i*-th iteration, associated with* ${e_i}^{(\alpha)}$.

The superscript, (γ), (α), $(\alpha\text{-}\beta)$, in variables indicates a specified DR pair or candidate set. We also use $(*)$ to indicate an arbitrary one among these DR pairs or candidate sets. For example, ${q_i}^{(*\text{-}*)}$ is an arbitrary candidate set of q_i, and ${q_i}^{(\alpha\text{-}*)}$ is an arbitrary one associated with ${e_i}^{(\alpha)}$.

The following is an example of the proposed analysis. Suppose the first few operations of an SM sequence are `SSMSMMS`. The initial candidate set of the exponent and that of the remaining exponent associated are

$$ {e_1}^{(1)} = \langle 1, 0 \rangle \text{ and } {q_1}^{(1\text{-}1)} = \langle 1, 0 \rangle . $$

The DR pair associated with the first block `S` is $({d_1}^{(1)}, {r_1}^{(1)}) = (2, 0)$, and we have the expansion factor $\varepsilon_{1,1} = 2$ when finding ${q_1}^{(1\text{-}1)} \div {d_1}^{(1)}$. Since only $split({q_1}^{(1\text{-}1)}, 2, 0) \bigcap \left\langle {d_1}^{(1)}, {r_1}^{(1)} \right\rangle = \langle 2, 0 \rangle$ is nonempty, the candidate sets of the exponent and the associated remaining exponent satisfying the first block are

$$ {e_2}^{(1)} = split({e_1}^{(1)}, 2, 0) = \langle 2, 0 \rangle , {q_2}^{(1\text{-}1)} = split({q_1}^{(1\text{-}1)}, 2, 0) \div {d_1}^{(1)} = \langle 1, 0 \rangle . $$

When analyzing the second block `SM`, the potential DR pairs are $({d_2}^{(1)}, {r_2}^{(1)}) = (2, 1)$ and $({d_2}^{(2)}, {r_2}^{(2)}) = (3, 0)$. We have $\varepsilon_{2,1} = 6$, and ${e_2}^{(1)}$ will be split into six subsets. Four of these subsets will satisfy the first two blocks, and they are

$$ {e_3}^{(1)} = split({e_2}^{(1)}, 6, 0) = \langle 12, 0 \rangle , {q_3}^{(1\text{-}1)} = split({q_2}^{(1\text{-}1)}, 6, 0) \div 3 = \langle 2, 0 \rangle ; $$

$$ {e_3}^{(2)} = split({e_2}^{(1)}, 6, 1) = \langle 12, 2 \rangle , {q_3}^{(2\text{-}1)} = split({q_2}^{(1\text{-}1)}, 6, 1) \div 2 = \langle 3, 0 \rangle ; $$

$$ {e_3}^{(3)} = split({e_2}^{(1)}, 6, 3) = \langle 12, 6 \rangle , \begin{cases} {q_3}^{(3\text{-}1)} = split({q_2}^{(1\text{-}1)}, 6, 3) \div 2 = \langle 3, 1 \rangle \\ {q_3}^{(3\text{-}2)} = split({q_2}^{(1\text{-}1)}, 6, 3) \div 3 = \langle 2, 1 \rangle \end{cases} ; $$

$$ {e_3}^{(4)} = split({e_2}^{(1)}, 6, 5) = \langle 12, 10 \rangle , {q_3}^{(4\text{-}1)} = split({q_2}^{(1\text{-}1)}, 6, 5) \div 2 = \langle 3, 2 \rangle . $$

When analyzing the third block `SMM`, the potential DR pairs are $(3, 1)$, $(3, 2)$, and $(5, 0)$. The sets ${e_3}^{(1)}$, ${e_3}^{(2)}$, ${e_3}^{(3)}$, ${e_3}^{(4)}$ will be split into $\varepsilon_{3,1} = 15$, $\varepsilon_{3,2} = 5$, $\varepsilon_{3,3} = 15$, $\varepsilon_{3,4} = 5$ subsets, and 11, 1, 15, 5 subsets will satisfy the first three blocks, respectively. We only demonstrate how to inspect the subset $split({e_3}^{(2)}, 5, 0)$ of ${e_3}^{(2)}$ and the fourth subset $split({e_3}^{(3)}, 15, 3)$ of ${e_3}^{(3)}$. Since $split({q_3}^{(2\text{-}1)}, 5, 0) = \langle 15, 0 \rangle \subset \langle 5, 0 \rangle$, we have

$$ {e_4}^{(12)} = split({e_3}^{(2)}, 5, 0) = \langle 60, 2 \rangle , {q_4}^{(12\text{-}1)} = split({q_3}^{(2\text{-}1)}, 5, 0) \div 5 = \langle 3, 0 \rangle . $$

Since $split({q_3}^{(3\text{-}1)}, 15, 3) = \langle 45, 10\rangle \subset \langle 3, 1\rangle$, $\langle 45, 10\rangle \subset \langle 5, 0\rangle$, and $split({q_3}^{(3\text{-}2)}, 15, 3) = \langle 30, 7\rangle \subset \langle 3, 1\rangle$, we have

$$ {e_4}^{(16)} = split({e_3}^{(3)}, 15, 3) = \langle 180, 42\rangle , \begin{cases} {q_4}^{(16\text{-}1)} = \langle 45, 10\rangle \div 3 = \langle 15, 3\rangle \\ {q_4}^{(16\text{-}2)} = \langle 45, 10\rangle \div 5 = \langle 9, 2\rangle \\ {q_4}^{(16\text{-}3)} = \langle 30, 7\rangle \div 3 = \langle 10, 2\rangle \end{cases} . $$

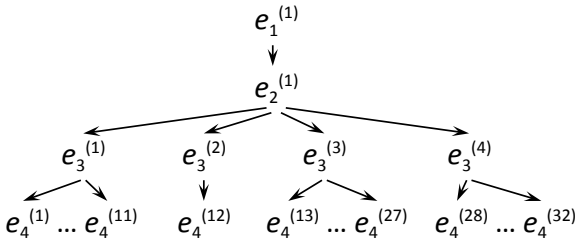

Fig. 3. Candidate sets satisfying the SM sequence `SSMSMMS`

Figure 3 sketches the relationship between the candidate sets in the above example, which forms a tree. Each child node is a subset of its parent node and will satisfy one more block of the SM sequence than its parent node. The proposed analysis is a breadth-first search, starting from the root ${e_1}^{(1)}$. Suppose we have obtained all ${e_i}^{(*)}$ as well as all ${q_i}^{(\alpha\text{-}*)}$ associated with each ${e_i}^{(\alpha)}$. We can find all ${e_{i+1}}^{(*)}$ as well as all ${q_{i+1}}^{(*\text{-}*)}$ by the following steps.

Step 1 Select one ${e_i}^{(*)}$

Step 2 Suppose ${e_i}^{(\alpha)}$ is selected. Find the smallest positive integer $\varepsilon_{i,\alpha}$ satisfying ${d_i}^{(\gamma)} \mid \varepsilon_{i,\alpha} {m_i}^{(\alpha\text{-}\beta)}$ for all γ and β (i.e., for all ${\mathrm{DR}_i}^{(*)}$ and ${q_i}^{(\alpha\text{-}*)}$. Split ${e_i}^{(\alpha)}$ and all ${q_i}^{(\alpha\text{-}*)}$ by the factor $\varepsilon_{i,\alpha}$.

Step 3 Select one subset of ${e_i}^{(\alpha)}$, i.e., select an integer $\delta \in [0, \varepsilon_{i,\alpha} - 1]$ and then compute $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta) = \left\langle \varepsilon_{i,\alpha} {M_i}^{(\alpha)}, \delta {M_i}^{(\alpha)} + {N_i}^{(\alpha)} \right\rangle$.

Step 4 Inspect whether the selected $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ satisfies the first i blocks of the SM sequence (i.e., whether it is a valid ${e_{i+1}}^{(*)}$).
If $\bigcup_{\beta,\gamma} \left(split({q_i}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \bigcap \left\langle {d_i}^{(\gamma)}, {r_i}^{(\gamma)} \right\rangle \right) \neq \emptyset$, then $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ is one ${e_{i+1}}^{(*)}$, and $\bigcup_{\beta,\gamma} \left(\left(split({q_i}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \bigcap \left\langle {d_i}^{(\gamma)}, {r_i}^{(\gamma)} \right\rangle \right) \div {d_i}^{(\gamma)} \right)$ represents all ${q_{i+1}}^{(*\text{-}*)}$ associated with $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$. Otherwise, discard $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$.
Step 5 Select another subset of ${e_i}^{(\alpha)}$ and go back to step 4 until all subsets are processed.

Step 6 Select another ${e_i}^{(*)}$ and go back to step 2 until all ${e_i}^{(*)}$ are processed.

Each ${e_i}^{(*)}$ is split into several subsets in step 2, and whether each subset is a valid ${e_{i+1}}^{(*)}$ is inspected in step 4. Since the remaining exponent should satisfy

$q_i \bmod d_i = r_i$, a subset $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ is one ${e_{i+1}}^{(*)}$ if and only if we can find at least one ${q_i}^{(\alpha\text{-}\beta)}$ (i.e., $\exists\beta$) satisfying $split({q_i}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \subset \bigcup_\gamma \left\langle {d_i}^{(\gamma)}, {r_i}^{(\gamma)} \right\rangle$. When s subset $split({q_i}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \subset \left\langle {d_i}^{(\gamma)}, {r_i}^{(\gamma)} \right\rangle$, $split({q_i}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \div {d_i}^{(\gamma)}$ is one ${q_{i+1}}^{(*\text{-}*)}$ associated with $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$.

The candidate sets ${e_{\mathsf{end}+1}}^{(*)}$ satisfying the whole SM sequence can be obtained by repeating the above steps. In the last few iterations of the analysis, the modulus ${M_i}^{(*)}$ of each ${e_i}^{(*)}$ will be greater than the upper limit of the exponent, and each ${e_i}^{(*)}$ can be simplified by ${e_i}^{(\alpha)} = {N_i}^{(\alpha)}$ because $(\delta {M_i}^{(\alpha)} + {N_i}^{(\alpha)})$ with $\delta \geq 1$ is not a valid exponent. The associated ${q_i}^{(*\text{-}*)}$ can be also simplified by ${q_i}^{(\alpha\text{-}\beta)} = {n_i}^{(\alpha\text{-}\beta)}$. In the last iteration, the DR pair $({d_{\mathsf{end}}}^{(*)}, {r_{\mathsf{end}}}^{(*)})$ should further satisfy ${r_{\mathsf{end}}}^{(*)} \neq 0$ because of $q_{\mathsf{end}} = d_{\mathsf{end}} q_{\mathsf{end}+1} + r_{\mathsf{end}} \neq 0$ and $q_{\mathsf{end}+1} = 0$. After analyzing the whole SM sequence, we have the candidates $\bigcup_\alpha {N_{\mathsf{end}+1}}^{(\alpha)}$ of the exponent.

The proposed analysis assumes the attacker can distinguish squaring and multiplication in a power consumption trace. However, some operations might not be identified exactly. Ambiguous SM patterns will be isolated and processed by the method similar to that analyzing the special pattern `SSMM`. For example, when an SM sequence is `SSMS?MS`, the fourth to sixth operations `S?M` will be isolated, and they are either `SMM` or `SSM`. The potential DR pairs are $(3, 1)$, $(3, 2)$, $(5, 0)$ for the first case and $(4, 2)$, $(6, 0)$ for the second case.

Some exponentiation algorithms employ uniform computation for both squarings and multiplications to prevent SPA, e.g., the side-channel atomicity [10]. However, the divisions $q_i = \lfloor \frac{q_{i-1}}{d_{i-1}} \rfloor$ in the MIST algorithm still reveal information. If the divisions are evaluated during the computation and the power consumption trace of divisions can be recognized, the SM patterns `S`, `SM`, and `SMM` can be determined by the number of squarings/multiplications between two divisions. In addition, some special chosen messages will cause revelation of computational sequence [11], e.g., $-1 \equiv n - 1 \pmod{n}$ and faulty elliptic curve points of small orders [12]. If the exponent is an odd integer and the input message (base number) is -1, referring to Fig. 2, there are only two types of computation, $1 \times 1 = 1$ and $1 \times (-1) = (-1)$. The computations `XA`→`A` and `TA`→`A` can be identified because `A` is always equal to -1. The analysis based on the observation of divisions or the special chosen message is similar to the proposed analysis.

The proposed analysis is an implementation of the exhaustive search in [6, Section 8]. In the next subsection, we will introduce how to exploit multiple SM sequences simultaneously.

3.2 Multi-sequence Analysis to MIST

The straightforward method to exploit multiple SM sequences is to find the candidates satisfying each SM sequence individually and then find the intersection of these candidates. In contrast, the proposed method directly finds the candidates satisfying all observed SM sequences. Since multiple SM sequences are

handled simultaneously, the additional subscript j is employed to indicate variables associated with the j-th SM sequence. Some variables are redefined below, and the steps 2 and 4 of the analysis in Sec. 3.1 are also modified.

Definition 4. *When analyzing the i-th block of the j-th SM sequence, let $DR_{j,i}{}^{(\gamma)} = (d_{j,i}{}^{(\gamma)}, r_{j,i}{}^{(\gamma)})$ be the γ-th potential DR pair; $q_{j,i}$ be the remaining exponent; $q_{j,i}{}^{(\alpha\text{-}\beta)} = \langle m_{j,i}{}^{(\alpha\text{-}\beta)}, n_{j,i}{}^{(\alpha\text{-}\beta)} \rangle$ be the β-th candidate set of $q_{j,i}$ associated with $e_i{}^{(\alpha)}$.*

Step 2 Suppose $e_i{}^{(\alpha)}$ is selected. Find the smallest positive integer $\varepsilon_{i,\alpha}$ satisfying $d_{j,i}{}^{(\gamma)} | \varepsilon_{i,\alpha} m_{j,i}{}^{(\alpha\text{-}\beta)}$ for all $\mathrm{DR}_{*,i}{}^{(*)}$ and $q_{*,i}{}^{(\alpha\text{-}*)}$ and for all SM sequences (i.e. $\forall j, \gamma, \beta$). Split $e_i{}^{(\alpha)}$ and all $q_{*,i}{}^{(\alpha\text{-}*)}$ by the factor $\varepsilon_{i,\alpha}$.

Step 4 Inspect whether the selected $split(e_i{}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ satisfies the first i blocks of all SM sequences (i.e., whether it is one $e_{i+1}{}^{(*)}$).
If $\bigcup_{\beta,\gamma} \left(split(q_{j,i}{}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \bigcap \left\langle d_{j,i}{}^{(\gamma)}, r_{j,i}{}^{(\gamma)} \right\rangle \right) \neq \emptyset$ for all j (i.e., for all SM sequences), then $split(e_i{}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ is one $e_{i+1}{}^{(*)}$, and
$\bigcup_{\beta,\gamma} \left(\left(split(q_{j,i}{}^{(\alpha\text{-}\beta)}, \varepsilon_{i,\alpha}, \delta) \bigcap \left\langle d_{j,i}{}^{(\gamma)}, r_{j,i}{}^{(\gamma)} \right\rangle \right) \div d_{j,i}{}^{(\gamma)} \right)$ represents all $q_{j,i+1}{}^{(*\text{-}*)}$ associated with $split(e_i{}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ for the j-th SM sequence.
Otherwise, discard $split(e_i{}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$.

Since each $e_i{}^{(*)}$ should satisfy all SM sequences, in the modified step 2, all candidate sets are split with the same factor $\varepsilon_{i,\alpha}$ instead of various factors for each SM sequence. In the modified step 4, $split(e_i{}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$ is one $e_{i+1}{}^{(*)}$ if the candidate set of remaining exponent $q_{j,i+1}{}^{(\alpha\text{-}*)}$ exists for every SM sequence.

According to [6, Theorem 9], the expected number of exponents satisfying a given SM sequence is about $2^{3n/5}$ for an n-bit exponent, i.e., an arbitrary exponent will satisfy a given SM sequence with the probability $2^{-0.4n}$. Three to five SM sequences might be sufficient to obtain a unique candidate of the exponent. In the next section, we will provide some implementation results and discuss the feasibility.

4 Results and Discussions

The proposed analysis is implemented in C++ and executes on PC with 3.1GHz CPU and 8GB RAM. It uses the *list container* of Standard Template Library to store candidate sets. Each candidate set of exponent $e_i{}^{(\alpha)}$ is associated with one or more candidate sets of remaining exponent $q_{j,i}{}^{(\alpha\text{-}*)}$ for every SM sequence j. For a candidate set, the modulus M in the residue class $\langle M, N \rangle$ is represented as the form $2^a 3^b 5^c$, and the remainder N is represented by 240-base number system. Figure 4 sketches the data structure in our implementation.

Figure 5 provides three results of finding 256-bit exponent by analyzing five SM sequences. Each time, we randomly generate a 256-bit exponent, and then the MIST algorithm executes five times to collect five SM sequences associated with this exponent. The figure illustrates the number of the candidate sets $e_i{}^{(*)}$

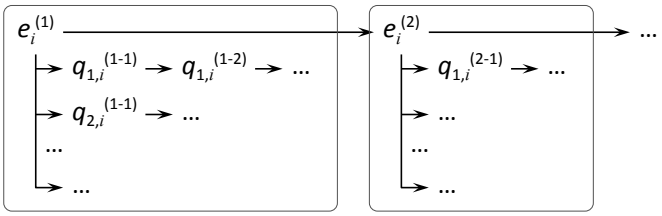

Fig. 4. Data structure for storing candidate sets

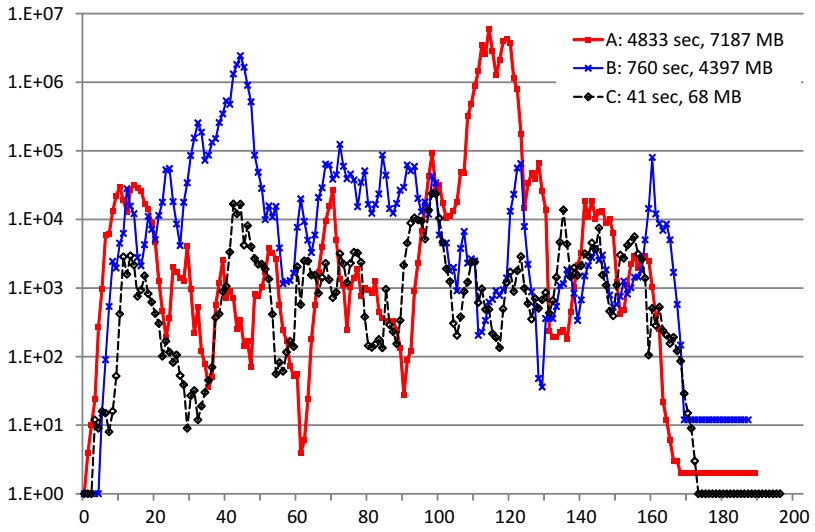

Fig. 5. Results of finding 256-bit exponent with 5 SM sequences

in each iteration, where the x-axis indicates the iteration i, and the logarithmic-scale y-axis indicates the number of ${e_i}^{(*)}$. The execution time (single thread execution) and peak memory usage are also provided. Figure 6 provides results with various numbers of SM sequences. We collect ten SM sequences associated with a 256-bit exponent and then perform analysis with 6/8/10 SM sequences.

The number of candidate sets during the analysis is determined by two factors, the ratio $\mathcal{E}_i$ of expansion in step 2 and the ratio $\mathcal{F}_i$ of filtering in step 4. The product $\mathcal{E}_i \times \mathcal{F}_i$ is the ratio between the number of ${e_i}^{(*)}$ and ${e_{i+1}}^{(*)}$. However, the implementation results show that the number of candidate sets in each iteration is irregular and depends on combination of observed SM sequences. There is no obvious trend of the number, except after nearly the 160th iteration, it decreases continually because expansion is not required when the modulus ${M_i}^{(*)}$ in each ${e_i}^{(*)}$ is greater than the upper limit of the exponent. Exploiting more SM sequences will remove more subsets in step 4, i.e., $\mathcal{F}_i$ gets smaller, but also generate more subsets in step 2, i.e., $\mathcal{E}_i$ gets larger. For example, between the

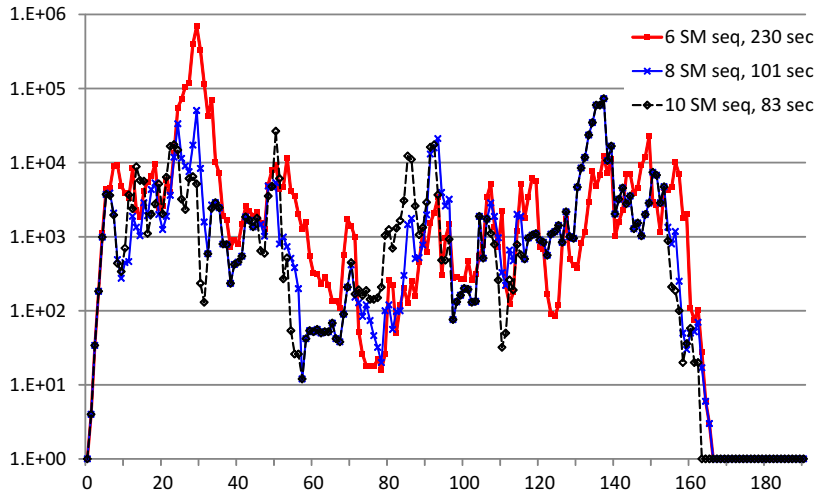

Fig. 6. Results of finding 256-bit exponent with 6/8/10 SM sequences

130th and 140th iteration of Fig. 6, there are more candidate sets in the analysis with 10 SM sequences than with 6 or 8 SM sequences.

The execution time depends on the total number of candidate sets during the analysis, and the peak memory usage depends on the maximum number of candidate sets in an iteration. When the exponent is 1024-bit or longer, the analysis program might spend too much memory and also extreme long time. In order to reduce the number of candidate sets, we can employ supplement SM sequences to inspect candidate sets. Unlike ordinary SM sequences, supplement SM sequences do not allow to split candidate sets of the exponent, i.e., they only reduce the filter factor $\mathcal{F}_i$ but not increase the expansion factor $\mathcal{E}_i$. Suppose the i-th iteration of the analysis with the ordinary SM sequences has finished. For each ${e_i}^{(\alpha)}$, we initialize ${q_{\mathtt{sup},1}}^{(\alpha\text{-}1)} = {e_i}^{(\alpha)}$ and find all ${q_{\mathtt{sup},2}}^{(\alpha\text{-}*)}$, ..., ${q_{\mathtt{sup},i'}}^{(\alpha\text{-}*)}$ relating to a supplement SM sequence until splitting is required (i.e., $\varepsilon_{i'+1,\alpha} \neq 1$ in step 2) to find ${q_{\mathtt{sup},i'+1}}^{(\alpha\text{-}*)}$. A candidate set ${e_i}^{(\alpha)}$ is valid if there is at least one ${q_{\mathtt{sup},i'}}^{(\alpha\text{-}*)}$, i.e., ${e_i}^{(\alpha)}$ satisfies the first $(i-1)$ blocks of the supplement SM sequence. After inspecting all candidate sets ${e_i}^{(*)}$ by a supplement SM sequence, we can filter them again by another supplement SM sequence. Since ${q_{\mathtt{sup},i'}}^{(*\text{-}*)}$ are not stored after analyzing a supplement SM sequence, exploiting supplement SM sequences only slightly increases the memory requirement.

According to our implementation results, five ordinary SM sequences and twenty supplement SM sequences are enough to deduce a 1024-bit exponent. We start to exploit supplement SM sequences when the number of candidate sets is more than $10,000$ and until it is less than $5,000$. These supplement SM sequences are circularly reused during the analysis. In the ten implementation results, the average execution time is 3,353 seconds (between 1,386 seconds and 5,719 seconds), and the average peak memory usage is 415 MB (between 214

MB and 872 MB). Exploiting more supplement SM sequences definitely reduces the number of candidate sets, but it might increase the execution time.

The proposed analysis is a breadth-first search (referring to Fig. 3). The breadth-first search requires a queue to store all nodes of a level, and the peak memory usage is determined by the maximum number of nodes of a level. The memory usage can be reduced by converting the proposed analysis to a depth-first search. When traversing to a child node, we only need to store its parent node and the index of the child node, i.e., storing ${e_i}^{(\alpha)}$ and δ when traversing to $split({e_i}^{(\alpha)}, \varepsilon_{i,\alpha}, \delta)$. The maximum number of nodes stored is equal to the maximum number of blocks of SM sequences, and the execution time is roughly the same as the original method.

5 Conclusions

When designing an exponentiation algorithm with immunity to SPA, removing the explicit relationship between the computational sequence and the secret exponent is a widely used method. However, an attacker can still obtain partial information about the secret exponent from a computational sequence. He might be able to integrate information collected from several computations if the algorithm is nondeterministic, i.e., randomized.

This paper proposes the first practical multi-sequence SPA against the MIST algorithm. Further countermeasures such as exponent blinding are required to prevent the attacker from collecting multiple SM sequences corresponding to the same exponent. The proposed method is an example that a randomized algorithm with immunity to single-sequence SPA might be vulnerable to multi-sequence SPA due to its randomness. Randomized exponentiation algorithms is not an elixir of preventing side-channel attacks.

References

1. Colin D. Walter, "Exponentiation using Division Chains," *IEEE Transactions on Computers*, vol. 47, no. 7, July 1998.
2. Jean-Sébastien Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *Cryptographic Hardware and Embedded Systems (CHES '99)*, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.
3. Paul C. Kocher, Joshua Jaffe and Benjamin Jun, "Differential power analysis," *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
4. Colin D. Walter, "MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis," *Topics in Cryptology – CT-RSA 2002*, LNCS 2271, pp. 53–66, Springer-Verlag, 2002.
5. Katsuyuki Okeya, "A Multiple Power Analysis Attack against Side Channel Attack Countermeasure MIST," (in Japanese) *Technical Report of IEICE*, ISEC2002-104, 53–58, 2002.
6. Colin D. Walter, "Some Security Aspects of the MIST Randomized Exponentiation Algorithm," *Cryptographic Hardware and Embedded Systems – CHES 2002*, LNCS 2523, pp. 276–290, Springer-Verlag, 2003.

7. Pierre-Alain Fouque and Frederic Valette, "The Doubling Attack – Why Upwards Is Better than Downwards," *Cryptographic Hardware and Embedded Systems – CHES 2003*, LNCS 2779, pp. 269–280, Springer-Verlag, 2003.
8. Elisabeth Oswald and Bart Preneel, "A Survey on Passive Side-Channel Attacks and their Countermeasures for the NESSIE Public-Key Cryptosystems," *Public reports of the NESSIE project*, 2003, available at `https://www.cosic.esat.kuleuven.be/nessie/reports/`
9. Sang Gyoo Sim, Dong Jin Park, and Pil Joong Lee, "New Power Analysis on the Ha-Moon Algorithm and the MIST Algorithm," *Information and Communications Security (ICICS 2004)*, LNCS 3269, pp. 291-304, Springer-Verlag, 2004.
10. Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity," *IEEE Transaction on Computers*, vol. 53, no. 6, pp. 760-768, 2004.
11. Jean-Christophe Courrège, Benoit Feix, and Mylène Roussellet, "Simple Power Analysis on Exponentiation Revisited," *Smart Card Research and Advanced Applications – CARDIS 2010*, LNCS 6035, pp. 65-79, Springer, 2010.
12. Sung-Ming Yen, Wei-Chih Lien, and Chien-Ning Chen, "Modified Doubling Attack by Exploiting Chosen Ciphertext of Small Order," *IEICE Transactions*, vol. 94-A, no. 10, pp. 1981-1990, 2011.