Two Operands of Multipliers in Side-Channel Attack*

Takeshi Sugawara, Daisuke Suzuki, and Minoru Saeki

Mitsubishi Electric Corporation
sugawara.takeshi@bp.mitsubishielectric.co.jp

Abstract. The single-shot collision attack on RSA proposed by Hanley et al. is studied focusing on the difference between two operands of multipliers. There are two consequences. Firstly, designing order of operands can be a cost-effective countermeasure. We show a concrete example in which operand order determines success and failure of the attack. Secondly, countermeasures can be ineffective if the asymmetric leakage is considered. In addition to the main results, the attack by Hanley et al. is extended using the signal-processing technique of the big mac attack. An experimental result to successfully analyze an FPGA implementation of RSA with the multiply-always method is also presented.

Keywords: RSA, Side-Channel Attack, Collision Attack, Montgomery Multiplication

1 Introduction

Side-channel attacks use unintentional information leakage from secure chips to compromise their security. New attacks and countermeasures have been studied for years since the first attack was discovered in 90s [1].

Side-channel attacks are divided into multiple- and single-shot attacks depending on the number of traces used. The first side-channel attack of RSA presented by Kocher et al. is a single-shot attack [1]. A typical modular exponentiation algorithm makes conditional branch between multiplication and squaring depending on a bit of the secret exponent. Kocher et al. showed that the multiplication and squaring are distinguishable by analyzing a power trace, and thus the secret exponent can be revealed.

Conditional branch is easily exploitable and thus should be removed. The multiply-always method in Alg. 1 is a well-known method to implement RSA without data-dependent branch. Even after data-dependent branch is removed, there is residual side-channel leakage correlated to the operands of the multiplication and squaring [4]. The residual leakage is weak [5], but the emerging new attacks can exploit it. Recently, successful single-shot attacks on FPGA implementations were reported [6] [7] [8]. The successful single-shot attacks have

* This is a draft version of the paper submitted to COSADE 2015. The final publication will be available at http://www.springer.com/.

a large impact in designing secure implementations. That is because suppressing the residual leakage requires a lot of effort. Hanley et al. suggest that a multiplier-level countermeasure [9] [10] [11] is needed for the suppression. In this paper, leakage from multipliers is studied. In contrast to the previous works on leakage from multipliers [9] [12] [13], asymmetry between operands of multipliers is focused. The contributions of this paper are summarized as follows.

A1 The single-shot collision attack on RSA proposed by Hanley et al. is studied focusing on the difference between two operands of multipliers. The asymmetry is reasoned by the Booth recoding and operand scanning.

A2 It is shown that designing order of operands can be a cost-effective countermeasure.

A3 It is shown that some countermeasure become ineffective when the asymmetric leakage is considered.

In addition to the above main results, there are two additional contributions.

B1 The single-shot attack by Hanley et al. [8] is extended using the technique of the big mac attack [12].

B2 An experimental result to successfully analyze an FPGA implementation of RSA with the multiply-always method is presented.

The paper is organized as follows. The conventional internal collision attacks are reviewed in Sect. 2, followed by the proposed extension of the attack by Hanley et al. Difference of operands of multipliers is discussed in Sect. 3. The experimental results are shown in Sect. 4. In the section, the attack in Sect. 2 is applied to an FPGA implementation with various operand orders. The experimental result are discussed in Sect. 5. Sect. 6 is a concluding remark.

2 Single-Shot Collision Attack

Firstly, conventional attacks are briefly reviewed. Then, the two most relevant attacks namely (i) the multiple-shot attack by Witteman et al. [14] and (ii) the single-shot attack by Hanley et al. [8] are described in detail. Finally, the proposed extension of the attack by Hanley et al. is described.

2.1 Conventional Single-Shot Attacks

Simple Power Analysis (SPA) [1] As described in the introduction, Kocher et al. proposed the first single-shot attack on RSA. The binary method used for modular exponentiation is targeted. In the binary method, there is a branch between square and multiplication depending a bit of the secret exponent. Kocher et al. showed that the path taken in the branch can be distinguished by analyzing a power trace.

Big Mac Attack (BMA) [12] Walter proposed BMA to attack another modular exponentiation algorithm called the window method [12]. The idea is to compare two segments of a power trace in order to find collision. In addition, a sophisticated signal-processing technique is introduced to improve the

Algorithm 1 Multiply-Always Method with Left-To-Right Scanning

Input: Message M, Modulo N, Secret exponent $d = (d_{t-1}, \cdots, d_0)_2$
Output: Ciphertext M^d
1: $R_0 \leftarrow 1$
2: **for** $j = t - 1$ downto 0 **do**
3: $\quad \overline{d_j} \leftarrow 1 - d_j$
4: $\quad R_0 \leftarrow R_0^2 \mod N$
5: $\quad R_{\overline{d_j}} \leftarrow R_0 \times M \mod N$
6: **end for**
7: Return R_0

performance of the comparison. Firstly, the segment is split into multiple sub-traces. Then the sub-traces are averaged together to make a processed segment. Signal-to-noise ratio (SNR) is improved by the processing. Finally the processed segments are compared. The feasibility of the attack is proved with simulation [12]. However, no practical result has been reported as described in [15].

Horizontal Correlation Power Analysis (HCPA) [9] HCPA proposed by Clavier et al. is a successor of BMA [9]. In the attack, a single trace is split into many sub-traces in the same manner as BMA. Then, a multiple-shot attack is mounted to the virtual multiple traces. There are experimental results successfully attacking software implementations [9].

Clustering-based attacks [6] Heyszl et al. proposed an attack using the k-means clustering [6]. That is then improved by Perin et al. [7]. In those attacks, segments of traces are classified into two groups using the k-means algorithm. The two groups expectedly correspond to 0 and 1 of secret bits. FPGA implementations are defeated by the attacks [6] [7]. Notably, Perin et al. successfully attacked an FPGA implementation with a multiplier-level countermeasure (the leak resilient arithmetic [11]) by exploiting the remaining first-order leakage.

2.2 Multiple-Shot Internal Collision Attack by Witteman et al. [14]

Witteman et al. proposed a new multiple-shot attack which exploits collision between consecutive operations i.e., internal collision [14]. The attack on the multiply-always method (Alg. 1) is described.

The consecutive multiplication and squaring in Alg. 1 are considered. For clarity they are rewritten as

$$R'_{\overline{d_j}} \leftarrow R_0 \times M \mod N \tag{1}$$

$$R''_0 \leftarrow R'^2_0 \mod N. \tag{2}$$

If $\overline{d_j} = 1$, the memory R_0 is not updated in Eq. (1) and thus $R_0 = R'_0$. Therefore, the multiplication and squaring collide. Alternatively when $\overline{d_j} = 0$, $R_0 \neq R'_0$ and there is no collision. As a result, one bit of the secret exponent is revealed by sensing the collision.

Suppose N different messages are encrypted with the same exponent. The multiplication and squaring traces for the i-the message are denoted by m_x^i and s_x^i, respectively. The subscript x represents time. In order to find the collision, m_x^i and s_x^i are compared. More specifically, the correlation coefficient matrix $C_{x,y}$ is calculated as:

$$C_{x,y} = \mathcal{F}_j[m_x^j, s_y^j]. \tag{3}$$

Here, $\mathcal{F}_j$ is the correlation-coefficient operator defined as follows:

$$\mathcal{F}_j[t^j, s^j] := \frac{1}{N} \sum_{j=0}^{N-1} \frac{(t^j - \mathcal{E}_j[t^j]) \cdot (s^j - \mathcal{E}_j[s^j])}{\sqrt{(\mathcal{E}_j[t^j]^2 - \mathcal{E}_j[(t^j)^2]) \cdot (\mathcal{E}_j[s^j]^2 - \mathcal{E}_j[(s^j)^2])}}, \tag{4}$$

$$\mathcal{E}_j[t^j] := \frac{1}{N} \sum_{j=0}^{N-1} t^j. \tag{5}$$

If there is a collision, $C_{x,y}$ contains a non-zero value. Therefore, the collision can be found by looking at the matrix.

2.3 Single-Shot Collision Attacks by Hanley et al. [8]

Hanley et al. proposed a single-shot attack against various addition-chain algorithms. In the following description, we focus on the one for the multiply-always method.

The attack uses internal collision similarly to the one by Witteman et al. However, the correlation coefficient in Eq. (4) is meaningless when $N = 1$ i.e., under a single-shot attack[1]. Instead, the two time-domain traces m_x^0 and s_x^0 are directly compared. Hanley et al. presented two different ways to measure the similarity: the Euclidean distance and the time-domain correlation coefficient given by $\mathcal{F}_x[m_x^0, s_x^0]$.

Hanley et al. applied the attack to a software implementation and successfully recovered 99 % of the exponent bits. They also applied the attack to an FPGA implementation, however, the attempt was unsuccessful with the one with the multiply-always method. That is explained by the fact that (i) single-shot attacks are susceptible to SNR and (ii) SNR is usually low in FPGAs because of higher parallelism.

The attack uses multiple points of interest. That is the advantage of the attack over the clustering-based attacks [7]. Therefore, the multiply-always method can be defeated even if there is no first-order leakage [7]. In addition, the attack is advantageous to HCPA on the point the known message is not needed. In other words, the attack by Hanley et al. defeats the message-blinding countermeasure.

[1] Clavier et al. proposed another single-shot extension [15]. The purpose of the attack is to distinguish multiplication and squaring. There is a practical result on a software implementation.

2.4 Proposed Extension of the Attack by Hanley et al.

An extension of the attack by Hanley et al. is described. The extension is on measuring the similarity between the traces. The idea is simple: the signal processing of BMA is applied to the traces before comparison [2].

Long-integer multiplication $A \times B$ is considered. A and B are composed of s words. They are denoted by $A = \{a_{s-1}, \cdots, a_0\}$ and $B = \{b_{s-1}, \cdots, b_0\}$ where a_j and b_i are words. The long-integer multiplication comprises generation of partial products $a_j \times b_i$. The leakage of $a_j \times b_i$ is denoted by $l(j, i)$.

The trace $l(j, i)$ is processed before comparison. Fig. 1. illustrates the process. Firstly, $l(j, i)$ is compressed into s-dimensional vectors $l_a(j)$ and $l_b(i)$. They are defined as:

$$l_a(j) = \frac{1}{s} \sum_{i=0}^{s-1} l(j, i), \tag{6}$$

$$l_b(i) = \frac{1}{s} \sum_{j=0}^{s-1} l(j, i). \tag{7}$$

$l_a(j)$ and $l_b(i)$ are called the compressed vectors. By the compression, the effect of one operand is removed thereby SNR of another operand is improved. The compressed vectors $l_a(j)$ and $l_b(i)$ correlate to a_j and b_i, respectively.

Finally, the compressed vectors from multiplication and squaring traces are compared in the same manner as the original attack. The measured traces of multiplication and squaring are denoted by $l(j, i)$ and $l'(j, i)$, respectively. The corresponding compressed vectors are denoted by $l_a(j)$, $l'_a(j)$, $l_b(i)$, and $l'_b(i)$. If the time-domain correlation coefficient is used for measurement, they are expressed as:

$$\mathcal{F}_j[l_a(j), l'_a(j)] \in [-1, 1], \tag{8}$$

$$\mathcal{F}_i[l_b(i), l'_b(i)] \in [-1, 1]. \tag{9}$$

The correlation coefficients become high if there is collision.

In order to conduct the attack, the attacker needs to get $l(j, i)$ from a raw trace in the same manner as BMA and HCPA. Even if the prior knowledge is unavailable, the attacker can possibly reverse-engineer the points of $l(j, i)$ by analyzing the correlation matrix in Eq. (3). That is because the patterns on the matrix reflect the underlying long-integer multiplication algorithm. That is explained in Sect. 4.2 with experimental results. Note that in order to get a meaningful correlation matrix, the exponent blinding should be disabled. Such a requirement is satisfied in two cases. Firstly, the attacker with an open sample can possibly profile the device while disabling the countermeasure. Secondly, the same co-processor for modular exponentiation may be used for another purpose without the exponent blinding. One such example is signature verification in which no secret is involved.

[2] The method can be thought as a missing variant with "regular algorithm + unknown message" in the categorization by Bauer et al. [10].

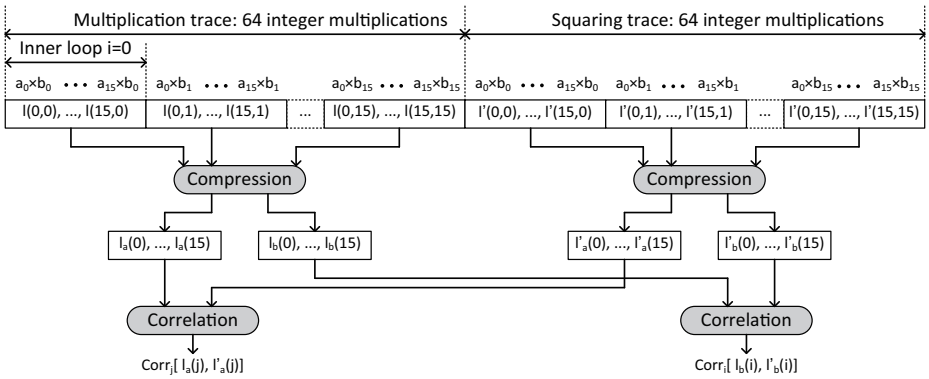


Fig. 1. Distinguisher

3 Asymmetric Leakage

Difference between two operands of multipliers is discussed in (i) integer multiplier and (ii) long-integer multiplication (LIM) levels.

3.1 Asymmetry at Integer Multiplier Level

In the paper of BMA, Walter showed that two operands of a simple multiplier are symmetric in terms of side-channel leakage [12]. However, sophisticated multipliers can be asymmetric as described below.

The Booth recoding is a common technique for partial product generation (PPG)[3] [17]. The technique enables to reduce the total number of partial products thereby improving the performance of integer multiplication. Fig. 2 shows a circuit for generating one partial product using the radix-4 Booth recoding. Firstly, the multiplicand A is expanded to $\{\overline{2A},\ \overline{A},\ 0,\ A,\ 2A\}$. The expansion is efficiently implemented using shifts and NOT gates. Then, one out of the five candidates is selected at the 5:1 selector. The selector output is the partial product. The selector is controlled by a 3-bit chunk of the multiplier namely $\{x_{i+1}, x_i, x_{i-1}\}$.

The circuit in Fig. 2 is asymmetric between operands. Therefore, asymmetric leakage is expected. Leakage from a 32-bit integer multiplier with the radix-4 Booth recoding is simulated. The multiplier is synthesized and post-synthesis logic simulation is conducted. While the logic simulation, the number of signal-transition events i.e., toggles is measured.

Two sets of test-vectors are used to drive the circuit. They are $c \times x_i$ and $x_i \times c$ where c and x_i are 32-bit integers. The test-vectors are designed to measure toggles from one operand by fixing another to a constant.

[3] Note that Walter and Samyde noticed that leakage from the Booth recoding is not the one by the Hamming-weight model [13]. However, the difference between operands was not discussed.

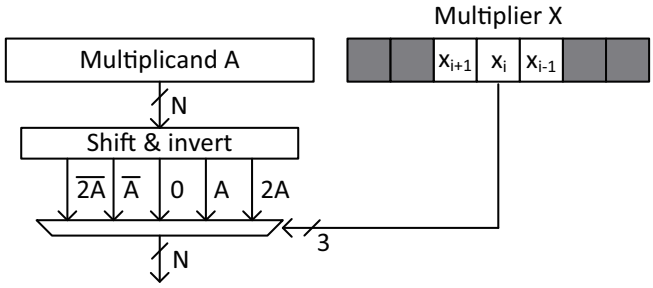


Fig. 2. A circuit for generating a partial product in the radix-4 Booth recoding

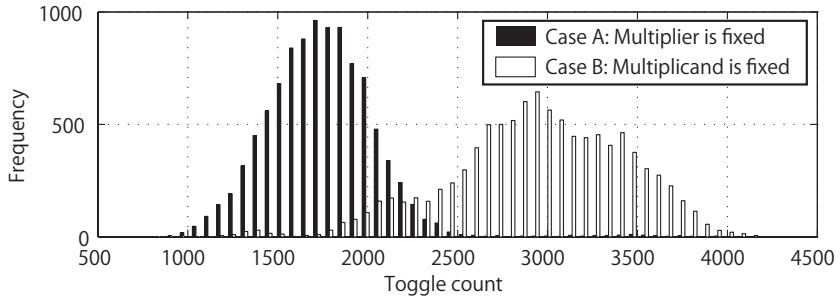


Fig. 3. Toggle counts of a 32-bit multiplier

Histograms of the measured toggle counts are shown in Fig. 3. The black and white bars correspond to the two sets of test-vectors. The two sets show clearly different histograms. As shown in the histograms, more toggles are observed when the multiplicand is fixed. That means the multiplier port makes more toggles. The result is explained by an empirical fact that a selector signal has stronger effect on toggle counts. More specifically, toggles at the 3-bit control signal is amplified to N bits at the selector output.

3.2 Asymmetry at Long-Integer Multiplication Level

Difference of operands at the LIM level is discussed. There are many options at this level. The Montgomery multiplication with the coarsely integrated operand scanning (CIOS) shown in Alg. 2 is considered.

The long integers are represented by $A = \{a_{s-1}, \cdots, a_0\}$ and $B = \{b_{s-1}, \cdots, b_0\}$ where a_j and b_i are words. The core operation is $a_j \times b_i$ at the line 4 of Alg. 2 in which partial products are generated. LIM is commonly implemented with a circuit shown in Fig. 4. The circuit uses a multiply-and-accumulate (MAC) unit. The words a_j and b_i are read from the memory and fed to the MAC unit via temporal registers labeled `regA` and `regB`.

Suppose `regA` and `regB` store the long integers A and B, respectively. Fig. 4 also shows an operation sequence describing the contents of the registers. As

Algorithm 2 Coarsely Integrated Operand Scanning [16]

Input: Word $A = \{a_{s-1}, \cdots, a_0\}$ and $B = \{b_{s-1}, \cdots, b_0\}$
Output: Product t_i for $i \in [0, s-1]$
1: **for** $i = 0$ to $s-1$ **do**
2: $C \leftarrow 0$
3: **for** $j = 0$ to $s-1$ **do**
4: $(C, S) \leftarrow t_j + a_j \times b_i + C$
5: $t_j \leftarrow S$
6: **end for**
7: $(C, S) \leftarrow t_s + C$
8: $t_s \leftarrow S$
9: $t_{s+1} \leftarrow C$
10: $C \leftarrow 0$
11: # Lines for the Montgomery reduction are not displayed for clarity.
12: # See the literature [16] for the complete list.
13: **end for**
14: Return t_j

shown in the table, `regB` is updated less frequently because the stored variables b_i is scanned at the outer loop in Alg. 2. For s-word long integers, `regA` and `regB` are updated s^2 and s times, respectively.

CMOS circuits make data-dependent power consumption when their inputs are changed [2]. As a result, the operand scanned at the inner loop (i.e., A) has stronger leakage. This LIM-level asymmetry is verified through experiments in Sect. 4.

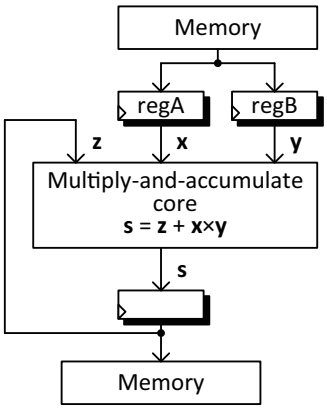
4 Experiments

Traces are captured from an FPGA implementation of the Montgomery multiplication. Firstly, the traces are analyzed using the attack by Witteman et al. Then, the single-shot attack in Sect. 2.4 is applied. The purpose of the experiment is twofold. Firstly, feasibility of the attack in Sect. 2.4 is verified. Secondly, the effects of the asymmetric leakage are examined.

4.1 Setup

A circuit implementing the 1024-bit Montgomery multiplication is examined. The circuit uses the MAC-based architecture in Fig. 4. The MAC unit has a 64-bit integer multiplier and thus the number of words $s = 16 = 1024/64$. The words are scanned with the CIOS method in Alg. 2. The two operands to the integer multiplier can be swapped by an external signal in order to evaluate the asymmetry at the integer-multiplier level. The target circuit is implemented on Virtex-II Pro FPGA on SASEBO [20].

The FPGA is measured by putting a magnetic-field probe on the chip surface [18]. The probe is 0.1 mm in diameter. Traces are captured using an oscilloscope with the bandwidth of 12.5 GHz and the sampling rate of 25.0 GSa/s.



An operation sequence when s=3

Seq. no.	regA	regB
0	a_0	b_0
1	a_1	b_0
2	a_2	b_0
3	a_0	b_1
4	a_1	b_1
5	a_2	b_1
6	a_0	b_2
7	a_1	b_2
8	a_2	b_2

Fig. 4. A common circuit architecture for LIM

Test-vectors are designed to emulate RSA with the multiply-always method (see Alg. 1). Firstly, 1024-bit random numbers x_k, y_k, and z_k are generated for $0 \leq k < 1000$. For each triplet (x_k, y_k, z_k), the Montgomery multiplication is called in five ways as summarized in Tab. 1. Note that the Montgomery multiplication is denoted by $\mathcal{M}(\cdot, \cdot)$ in the table.

The five Montgomery multiplications are denoted by (XY1), (XY2), (XX), (YY), and (ZZ). The Montgomery multiplication $\mathcal{M}(x_k, y_k)$, which corresponds to the multiplication in Alg. 1, is conducted in (XY1) and (XY2). (XY1) and (XY2) are different in the order of operands to the 64-bit integer multiplier. See Tab. 1 for the operands in the LIM and integer-multiplier levels. The remaining Montgomery multiplications namely $\mathcal{M}(x_k, x_k)$, $\mathcal{M}(y_k, y_k)$, and $\mathcal{M}(z_k, z_k)$ correspond to the squaring in Alg. 1.

The traces are examined in pair. Six pairs namely

$$\{(\mathrm{XY1}), (\mathrm{XY2})\} \times \{(\mathrm{XX}), (\mathrm{YY}), (\mathrm{ZZ})\}.$$

are evaluated. The pairs are referred to as (i)-(vi) as summarized in Tab. 2. There are collisions in the pairs (i)-(iv). Colliding operands, both at the LIM and integer-multiplier levels, are also shown in Tab. 2. In (i) and (iii), there is a collision between the LIM-level operands scanned at the inner loop of Alg. 2. On the other hand, the operands scanned at the outer loop collide in (ii) and (iv). At the integer-multiplier level, the multiplicands collide in (i) and (iv). Alternatively the multipliers collide in (ii) and (iii). There is no collision in (v) and (vi).

The pairs are compared under the multiple- and single-shot attacks in the following sections.

Table 1. Test-vectors of the Montgomery multiplication

		LIM level		integer-multiplier level	
Identifier	Operation	Inner loop (a_j)	Outer loop (b_i)	multiplier	multiplicand
(XY1)	$\mathcal{M}(x_k, y_k)$	x_k	y_k	y_k	x_k
(XY2)	$\mathcal{M}(x_k, y_k)$	x_k	y_k	x_k	y_k
(XX)	$\mathcal{M}(x_k, x_k)$	x_k	x_k	x_k	x_k
(YY)	$\mathcal{M}(y_k, y_k)$	y_k	y_k	y_k	y_k
(ZZ)	$\mathcal{M}(z_k, z_k)$	z_k	z_k	z_k	z_k

Table 2. Examined pairs of traces

Identifier	Multiplication m_x^j	Squaring s_y^j	LIM-level collision	integer-multiplier-level collision
(i)	(XY1)	(XX)	inner (a_j)	multiplicand
(ii)	(XY1)	(YY)	outer (b_i)	multiplier
(iii)	(XY2)	(XX)	inner (a_j)	multiplier
(iv)	(XY2)	(YY)	outer (b_i)	multiplicand
(v)	(XY1)	(ZZ)	—	—
(vi)	(XY2)	(ZZ)	—	—

4.2 Multiple-shot leakage using the attack by Witteman et al.

As a preliminary experiment, the pairs of the traces are analyzed using the attack by Witteman et al. The correlation matrices in Eq. (3) are calculated for the pairs (i)-(iv) in Tab. 2. The matrices are visualized as bitmap images in Fig. 5.

The bitmap images show different patterns depending on the colliding operands at the LIM level. There are repeated slash lines on Fig. 5-(i) and -(iii) in which there are collisions at the inner loop. On the other hand, collision at the outer loop makes rectangle patterns as shown in Fig. 5-(ii).

The bitmap images also show the difference caused by the asymmetry at the integer-multiplier level. The multiplier (cf. the multiplicand) shows higher correlation as expected in Sect. 3.1. The slash lines are more clear in Fig 5-(iii) compared to the ones in Fig 5-(i). Similarly, the rectangle patterns are more distinct in Fig 5-(ii).

The bitmap images are intuitive but unsuitable for quantitative comparison. More concrete comparison is conducted in the next section.

As described in Sect. 2.4, the attacker can get the points of interest for $l(j, i)$ by interpreting the bitmap images. The above-mentioned slash-line and rectangle patterns are commonly found in many implementations. The attacker can sample the clock cycles with high correlation for $l(j, i)$.

4.3 Single-shot attack

The pairs of traces are analyzed with the method described in Sect. 2.4. The correlation coefficients are evaluated for the pairs (i)–(vi) in Tab. 2. The pro-

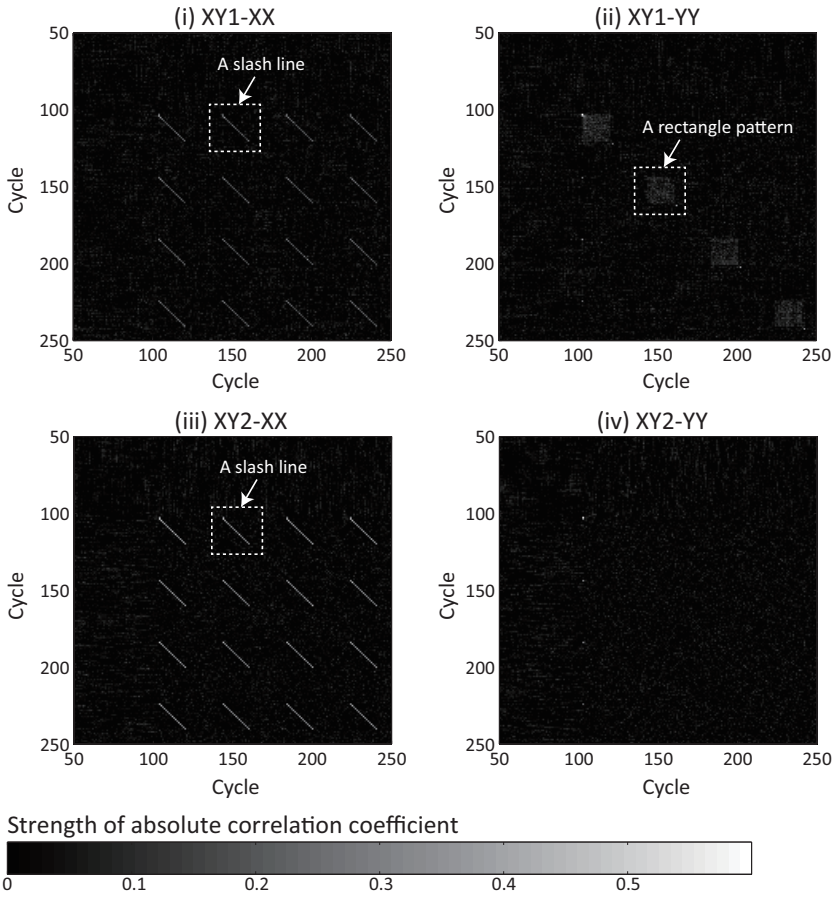


Fig. 5. Correlation coefficient matrices

cessing is chosen considering the patterns on the bitmap images. For the pairs (i) and (iii) with collisions at the inner loop, Eq. (8) is used. Alternatively, Eq. (9) is used for the pairs (ii) and (iv).

The results are shown as histograms of the correlation coefficients in Fig. 6. Fig. 6-(i) to -(iv) correspond to the pairs (i)-(iv) in Tab. 2. The pairs (v) and (vi), that have no collision, are also shown for comparison.

In the real attack, the black and white histograms are not separated. Therefore, the attacker should set a threshold to make a decision. In this experiment, the measured correlation coefficients are split into upper and lower halves. In other words, median is used as a threshold. Finally, the rate of successful decision is calculated. The success rates are (i) 98.3 %, (ii) 93.0 %, (iii) 99.5 %, and (iv) 52.7 %, respectively. The pairs (iii) is the most distinguishable. This is the first successful single-shot collision attack of the multiply-always method

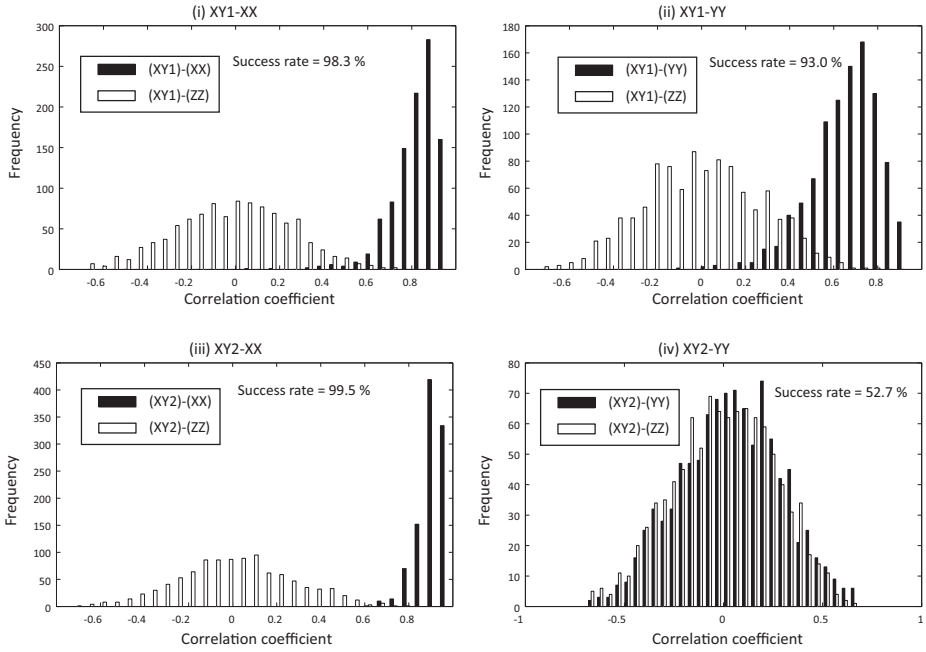


Fig. 6. Histograms of correlation coefficients

on FPGA. In contrast, the attack is unsuccessful in (iv). The result show that operand order has a significant impact on the success rate of the attack.

5 Discussion

The experimental results indicate that the operand order has a considerable impact on side-channel leakage. The results are discussed from countermeasure and attack view points.

5.1 Leak Reduction by Designing Operand Order

The experimental results show that the operand order determines success and failure of the attack. Therefore, the amount of leakage can be reduced by appropriately designing the order of operands. The pair (iv) in Tab. 2 is the best option (see Fig. 6-(iv)). The operand order can be changed at almost no cost. In addition to the cost effectiveness, the proposed method can easily be combined with other conventional countermeasures (e.g., the randomized operand scanning [9] [10]).

The target implementation discussed in the paper, the multiply-always method using the Montgomery multiplication with CIOS, is one of many possible designs. It is worth discussing how the operand order can be designed in other cases.

Firstly, the same idea can be easily extended to many other methods. That is because the cause of asymmetry, the partial-product generation and operand scanning, are essential in long-integer multiplication.

However, there is always an exception. Notable exception is the Finely Integrated Operand Scanning (FIOS) instead of CIOS. In FIOS, the register containing the variable scanned at the outer cannot be kept while the outer loop. That is because another word-wise multiplication, needed for the Montgomery reduction, should be interleaved. As a result, the leakage from the operand scanned at outer loop not necessarily smaller.

Alternatively, the operand order can be determined using the conventional toggle simulation. As shown in Sect. 3.1, the asymmetry at the integer multiplier can be simulated. The LIM-level asymmetry is not simulated in the paper, however, the frequency of register update can be covered by the toggle simulation.

5.2 Attack on Montgomery Powering Ladder

In contrast to the previous result, the asymmetric leakage can make some countermeasures ineffective. Alg. 3 shows the Montgomery powering ladder (MPL). We focus on collisions between inputs [4].

In MPL, there is always collision between consecutive operations:

$$R_{\overline{a}} \leftarrow R_0 \times R_1 \mod N, \tag{10}$$
$$R_a \leftarrow R_a \times R_a \mod N. \tag{11}$$

Therefore, the presence of collision does not leak k_j. However, the colliding operand depends on k_j. If $k_j = a = 0$, the first operands in Eq. (10) and (11) collide. On the other hand, the second operands collide if $k_j = a = 1$. If the attacker can distinguish the collisions at first and second operands, the secret parameter $k_j = a$ is revealed.

Interestingly, if Eq. (10) and Eq. (11) are replaced with the following statements without changing the result of the algorithm, the attack is no longer effective:

$$R_{\overline{a}} \leftarrow R_a \times R_{\overline{a}} \mod N, \tag{12}$$
$$R_a \leftarrow R_a \times R_a \mod N. \tag{13}$$

The algorithm appear in the work by Hanley et al. [8]. Now, collision is always occurred at the first operand. Therefore, the colliding operand becomes independent of k_j. This is another example showing the importance of designing operand order.

[4] Hanley et al. considered more general cases considering a collision between input and output. However, the input-output collision was very weak in our setup.

Algorithm 3 Montgomery Powering Ladder

Input: Message M, scalar $k = (k_{t-1}, \cdots, k_0)_2$
Output: Ciphertext M^d
1: $R_0 \leftarrow 1$; $R_1 \leftarrow M$
2: **for** $j = t - 1$ downto 0 **do**
3: $a \leftarrow k_j$; $\overline{a} \leftarrow 1 - a$
4: $R_{\overline{a}} \leftarrow R_0 \times R_1 \mod N$
5: $R_a \leftarrow R_a \times R_a \mod N$
6: **end for**
7: Return R_0

6 Conclusion

Two operands of multipliers are asymmetric in terms of side-channel leakage. The reason can be explained by asymmetries at arithmetic-circuit and micro-architecture levels. The leakage can be suppressed by appropriately designing the order of operands. On the other hand, some countermeasure can be defeated if the leakages from first and second operands are distinguishable.

Many problems are remained open. The attack using input-to-output collision is an interesting challenge. Another important open problem is on incomplete exponent recovery. The successful rate more than 99 % is clearly dangerous. The ideal goal is 50.0 %, however, it could possibly be relaxed.

Acknowledgement

The authors would like to thank the anonymous reviewers at COSADE 2015 for their valuable comments. The study was conducted as a part of the CREST Dependable VLSI Systems Project funded by the Japan Science and Technology Agency.

References

1. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", CRYPTO 1999, LNCS, Vol. 1666, pp. 388–397, 1999.
2. S. Mangard, E. Oswald, T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards," Springer-Verlag, 2007.
3. J.-S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," CHES 1999, LNCS, Vol. 1717, pp. 292-302, 1999.
4. F. Amiel, B. Feix, M. Tunstall, C. Whelan, W. P. Marnane, "Distinguishing Multiplications from Squaring Operations," SAC 2008, LNCS Vol. 5381, p. 346-360, 2009.
5. N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, "Collision-based Power Analysis of Modular Exponentiation Using Chosen-message Pairs", CHES 2008, LNCS, Vol. 5154, pp. 100-112, 2008.
6. J. Heyszl, A. Ibing, S. Mangard, F. D. Santis, and G. Sigl, "Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations", Smart Card Research and Advanced Applications LNCS 2014, pp 79-93.

7. G. Perin, L. Imbert, L. Torres, and P. Maurine, " Attacking Randomized Exponentiations Using Unsupervised Learning", In Proc. COSADE 2014.
8. N. Hanley, H. Kim, and M. Tunstall, "Exploiting Collisions in Addition Chain-based Exponentiation Algorithms Using a Single Trace", Cryptography ePrint Archive: Report 2012/485, http://eprint.iacr.org/2012/485
9. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal Correlation Analysis on Exponentiation," In ICICS, pp. 46–64, 2010.
10. A. Bauer, E. Jaulmes, E. Prouff, and J. Wild, "Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations", CT-RSA 2013, LNCS, Vol. 7779, pp. 1-17, 2013.
11. J.-C. Bajard, L. Imbert, P.-Y. Liardet, and Y. Teglia, "Leak Resistant Arithmetic", CHES 2004, LNCS, Vol. 3156, pp. 62-75, 2004.
12. C. D. Walter, "Sliding Windows Succumbs to Big Mac Attack.," CHES 2001, LNCS, Vol. 2162, pp. 286–299, 2001.
13. C. D. Walter and D. Samyde, "Data Dependent Power Use in Multipliers", 17th IEEE Symposium on Computer Arithmetic (ARITH'05).
14. M. F. Witteman, J.G.J. van Woudenberg, F. Menarini, "Defeating RSA Multiply-Always and Message Blinding Countermeasures," CT-RSA 2011, LNCS, Vol. 6558 pp. 77–88, 2011.
15. C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Roussellet, and V. Verneuil, " ROSETTA for Single Trace Analysis Recovery Of Secret Exponent by Triangular Trace Analysis", INDOCRYPT 2012, LNCS, Vol. 7668, pp. 140-155, 2012.
16. C. K. Koç, T. Acar, B. S. Kaliski Jr, "Analyzing and Comparing Montgomery Multiplication Algorithms", Micro, IEEE, 1996.
17. I. Koren, "Computer Arithmetic Algorithms 2nd Ed.", A K Peters/CRC Press, 2001.
18. T. Sugawara, D. Suzuki, M. Saeki, M. Shiozaki, T. Fujino, "On Measurable Side-Channel Leaks Inside ASIC Design Primitives", CHES 2013, LNCS, Vol. 8086, pp. 159-178, 2013.
19. K. Okeya, K. Sakurai, "A Second-order DPA Attack Breaks a Window-method based Countermeasure against Side Channel Attacks," ISC 202. Volume 2433 of LNCS., Springer (2002) 389–401
20. AIST, Side-Channel Attack Standard Evaluation Board, http://www.risec.aist.go.jp/project/sasebo/