Collision-based Power Analysis of Modular Exponentiation Using Chosen-message Pairs

Naofumi Homma[1], Atsushi Miyamoto[1], Takafumi Aoki[1],
Akashi Satoh[2], and Adi Shamir[3]

[1]Graduate School of Information Sciences, Tohoku University
{homma, miyamoto}@aoki.ecei.tohoku.ac.jp, aoki@ecei.tohoku.ac.jp
[2]National Institute of Advanced Industrial Science and Technology
akashi.satoh@aist.go.jp
[3]Weizmann Institute of Science
adi.shamir@weizmann.ac.il

Abstract. This paper proposes new chosen-message power-analysis attacks against public-key cryptosystems based on modular exponentiation, which use specific input pairs to generate collisions between squaring operations at different locations in the two power traces. Unlike previous attacks of this kind, the new attacks can be applied to all the standard implementations of the exponentiation process: binary (left-to-right and right-to-left), m-ary, and sliding window methods. The SPA countermeasure of inserting dummy multiplications can also be defeated (in some cases) by using the proposed attacks. The effectiveness of the attacks is demonstrated by actual experiments with hardware and software implementations of RSA on an FPGA and the PowerPC processor, respectively. In addition to the new collision generation methods, a high-accuracy waveform matching technique is introduced to detect the collisions even when the recorded signals are noisy and the clock has some jitter.

Keywords: side-channel attacks, power-analysis attacks, RSA, modular exponentiation, waveform matching

1 Introduction

Physical attacks on cryptographic modules using side-channel information are attracting extensive attention. In order to reveal the secret parameters, the power dissipation, the electromagnetic radiation, or the operating times related to internal operations are analyzed. Two of the best known attacks are Simple Power Analysis (SPA) and Differential Power Analysis (DPA) proposed by Kocher et al. [1, 2].

The original concept of side-channel attacks against modular exponentiation [3] is to look for some physical phenomena which differentiates between multiplication and squaring operations. Messerges presented a variety of power-analysis attacks against RSA with some experimental results [4]. However, most of the implementations of modular exponentiation nowadays use the same sequence of instructions to implement multiplications and squarings, and for random inputs, it is very difficult to distinguish between these two operations. In order to cause secret information to leak via the power waveforms, chosen-message attacks that use specific data specialized for a particular cryptographic module were proposed [5–10].

The timing attacks against RSA with Montgomery multiplication [11] and/or CRT algorithm in [5, 6] measures the operating times caused by extra calculations depending on input data. The SPA with adaptively chosen messages [7] can be applied to an RSA implementation using CRT based on Garner's algorithm, in which an extra modular reduction is performed at the end of the operation according to the input data. The DPA using the Hamming weight of an intermediate value [8] was also applied to RSA with CRT. These attacks focused on specific RSA implementations, and thus information about the implementation is indispensable to reveal the secret keys. The first three attacks can be defeated by inserting dummy operations, and the DPA of [8] cannot be applied to implementations using the Montgomery algorithm.

Over the last few years, several researchers have proposed to use a power analysis technique which is a mixture of the simple and the differential approaches. This technique compares two segments of power consumption data (within a single execution or in two different executions) and uses the result to determine whether the values operated on were the same or different. For example, when we perform two multiplications $a \times b$ and $c \times d$, we expect the power consumption curves to be similar when $a = c$ and $b = d$, and different in all other cases. This can give us a simple equality oracle, even though it may be extremely difficult to determine the actual values of a, b, c, and d from the complex waveforms. This is not a standard SPA technique since we do not try to understand the details of each waveform, and it is not a standard DPA since it is not based on the statistical analysis of large collections of power traces. We propose to call such attacks on pairs of waveforms CPA (Comparative Power Analysis).

One of the simplest attacks of this type was proposed by Yen et al [10]. It uses the particular input data of $N - 1$ where N is the modulus, which has the special property that all its powers are either 1 or -1. However, a simple countermeasure is to block the special message $N - 1$, and the attack can only be applied to implementations using a left-to-right binary method.

Another attack of this type is the "doubling attack" of Fouque and Valette [9]. They used the two related input messages X and X^2 to cause collisions between adjacent time frames in the two power waveforms, where squaring operations are performed. Since every message X can be part of such a message pair, it is harder to block potentially harmful messages. As in the case of Yen's method, these attacks can only be applied to the left-to-right binary method, and the authors make this point explicit in the title of their paper: "The Doubling Attack - Why Upwards is Better than Downwards".

In this paper we propose new power-analysis attacks using input pairs which can be successfully applied to all the standard implementations of the exponentiation function, including both left-to-right and right-to-left binary methods, m-ary (window), and sliding window methods. The major new element of these attacks is the observation that an attacker can easily choose pairs of messages that generate collisions between their power traces at arbitrary time frames (which need not be the same or adjacent) even though he does not know the factorization of the modulus and thus cannot extract modular roots. Information about the locations of such non-adjacent collisions in the power traces is then used to identify the bit pattern of the secret exponent. In the proposed attack, the relationship between the two input messages can cope flexibly with the many variants of exponentiation algorithms, including those which were immune to previous attacks.

We demonstrate the practical effectiveness of the proposed attacks against hardware and software implementations of RSA using a Xilinx FPGA with a PowerPC processor core. In this experiment, a high-accuracy waveform matching technique is introduced to find collisions between squaring patterns that appear at different time frames even when the signal is noisy and the clock has some jitter.

The remainder of this paper is organized as follows: Section 2 presents an overview of modular exponentiation algorithms and describes power-analysis attacks using a chosen-message pair. In Section 3, the new power-analysis attacks using chosen-message pairs against binary and m-ary methods are proposed. Section 4 describes the experimental results using actual RSA hardware and software implementations. Finally, Section 5 contains some concluding remarks.

2 Preliminary and related attacks

2.1 Modular exponentiation algorithms

Modular exponentiation is one of the most important arithmetic operations for public-key cryptography, such as the RSA scheme and the ElGamal encryption scheme, and for the Diffie-Hellman key agreement. Basically, there are two types of efficient exponentiation algorithms: binary methods and m-ary (or window) methods [12, 13].

The binary method performs multiplications and squarings sequentially according to the bit pattern of the exponent. There are two variations of the algorithm. The left-to-right binary method starts at the exponent's MSB and works downward. The right-to-left binary method, on the other hand, starts at the exponent's LSB and works upward. **ALGORITHM 1** shows the left-to-right binary method, where k indicates the bit length of the secret keys. Each multiplication (or squaring) operation requires a large number of clock cycles due to the long operand length depending on the implementation. The binary method is frequently used in smartcards and embedded devices, due to its simplicity and low resource consumption.

The m-ary method processes more than one bit of the exponent in each iteration cycle, in which the exponent uses a representation with base m. **ALGORITHM 2** shows the m-ary method in which the exponent is processed from

ALGORITHM 1
Left-to-right binary method

```
Input:  $X$, $N$,
        $E = (e_{k-1}, ..., e_1, e_0)_2$
Output: $Z = X^E \bmod N$
1 :  $Z := 1$;
2 :  for $i = k-1$ downto $0$
3 :     $Z := Z * Z \bmod N$;
4 :     if $(e_i = 1)$ then
5 :        $Z := Z * X \bmod N$;
6 :     end if
7 :  end for
```

ALGORITHM 2
m-ary method

```
Input:  $X$, $N$,
        $E = (e_{k-1}, ..., e_1, e_0)_{2^m}$,
        for $m \geq 1$.
Output: $Z = X^E \bmod N$
1 :  $g_0 := 1$;
2 :  for $i = 1$ to $2^m - 1$
3 :     $g_i := g_{i-1} * X$;    — $g_i = X^i$
4 :  end for
5 :  $Z := 1$;
6 :  for $i = k-1$ downto $0$
7 :     for $l = 1$ to $m$
8 :        $Z := Z * Z \bmod N$;
9 :     end for
10:     $Z := Z * g_{e_i} \bmod N$;
11:  end for
```

the MSB down to the LSB. The powers $g_i \bmod N$ $(i = 0, 1, 2, ..., 2^m - 1)$ are pre-computed and used in multiplication. The intermediate value Z is raised to the power of 2^m by repeating the squaring operation m times. The m-ary method requires fewer clock cycles but more memory resources compared with the binary methods, and thus is often used for software implementation on processors with large memory resources. The sliding window method is an extension of the m-ary method to reduce the amount of pre-computation by using the presence of zero bits in the exponent.

2.2 SPA using a chosen-message pair against modular exponentiation

The doubling attack [9] uses the two related inputs X and X^2. The secret exponent is revealed by detecting collisions of squaring operations in two power traces. Fig. 1 illustrates an image of the doubling attack against the left-to-right binary method in **ALGORITHM 1** with the secret key exponent of "101001..." The doubling attack can generate a collision between a squaring operation at the $i+1$-th cycle in the power trace of X and a squaring operation at the i-th cycle in that of X^2 only if the corresponding key bit e_i is 0. The collision for squaring is detected by comparing the power traces, and thus we do not have to know the intermediate data being processed. The doubling attack works on modular exponentiation based on left-to-right binary methods including those using the blinding countermeasures shown in [14].

A different attack which uses the message pair X and $-X$ $(= N - X \bmod N)$ was proposed by Yen et al [10]. Fig. 2 illustrates an image of this attack against the left-to-right binary method. When the key bit e_i is 0, a collision between power traces can be observed for the two squaring operations during the same iteration cycle.

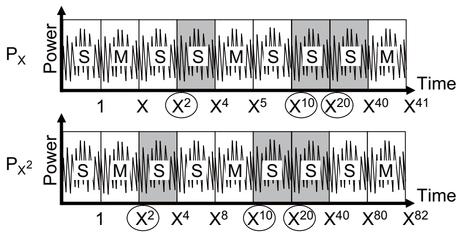

P_X
Power
S M S S M S S S M
Time
1 X X^2 X^4 X^5 X^10 X^20 X^40 X^41
P_X^2
Power
S M S S M S S S M
Time
1 X^2 X^4 X^8 X^10 X^20 X^40 X^80 X^82

Fig. 1. Doubling attack. [9]

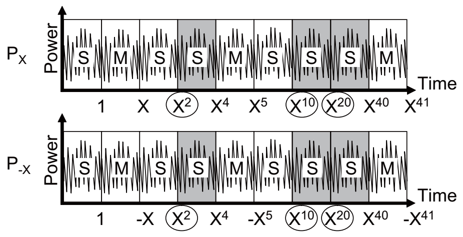

P_X
Power
S M S S M S S S M
Time
1 X X^2 X^4 X^5 X^10 X^20 X^40 X^41
P_-X
Power
S M S S M S S S M
Time
1 -X X^2 X^4 -X^5 X^10 X^20 X^40 -X^41

Fig. 2. Yen's attack. [10]

Both attacks exploit the fact that the values which are squared depend on the bits of the secret exponent. As mentioned in [9], it is hard to apply the attack to exponentiation algorithms such as right-to-left algorithms and window methods that perform squaring operations independently of the secret exponent.

3 The New Attacks

The above two attacks generate collisions of squaring operations at the adjacent or the same time frames in two power traces. In contrast, the proposed attacks generate a collision between two power traces at two arbitrary time frames by using two input messages with a more flexible relationship. One input gives a power trace including an unknown (multiplication or square) operation depending on a target key bit to be estimated, which is called a target operation. The other input gives a power trace including a square operation, the input of which can be determined by the known sub-key bits, referred to as the reference operation. The partial traces for the target and reference operations are called target and reference waveforms, respectively. The collision between the target and reference waveforms is used to estimate the target key bit.

Our attacks provide direct and backward estimations of the key exponent using the collision. The direct estimation simply compares the target (squaring or multiplication) operation with the reference (squaring) operation to identify the target operation corresponding to the key bit. The backward estimation identifies the target operation by comparing a squaring operation following the target operation with the reference operation. Unlike all the previous techniques, these new estimation techniques can be applied to all the standard exponentiation techniques (including both left-to-right and right-to-left binary methods, m-ary methods and the sliding window methods).

The simple trick we use in order to generate a collision at any pair of locations in two power traces is to find a solution for any equation of the form $Y^\alpha = Z^\beta \bmod N$, where α and β are given constants. Note that the attacker does not know the factorization of N and thus cannot solve this equation by extracting modular roots. However, he can choose an arbitrary value R and compute $Y = R^\beta \bmod N$ and $Z = R^\alpha \bmod N$, which is clearly a solution for the equation. This method is also applicable for CRT implementation that uses the prime

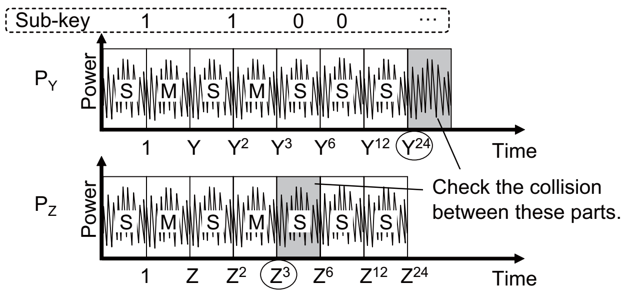


Fig. 3. Attack on the binary method (direct estimation).

factors p and q of N as the moduli since the message pair Y and Z satisfies $Y^\alpha = Z^\beta \bmod p$ and $Y^\alpha = Z^\beta \bmod q$.

3.1 Attack on binary methods

First, the direct estimation of the binary method shown in **ALGORITHM 1** is described. Suppose that the sub-key bits $E^{(j)}$ $(= e_{k-1}, ..., e_{k-j})$ of the secret exponent E have already been obtained. In order to estimate the next key bit $e_{k-(j+1)}$, a message pair is used, which causes a collision between the target and reference operations performed at different time frames. If a collision is observed, the target operation is a squaring (i.e., $e_{k-(j+1)} = 0$). If no collision is observed, then the operation is a multiplication (i.e., $e_{k-(j+1)} = 1$). Once $e_{k-(j+1)}$ is obtained, the remaining bits $e_{k-(j+2)}, ..., e_0$ are sequentially computed in the same manner.

The message pair Y and Z is given as $Y^\alpha = Z^\beta (Y \neq Z)$, where the α and β satisfy

$$\alpha = 2E^{(j)}, \tag{1}$$

$$\beta = \left\lfloor \frac{\alpha}{2^t} \right\rfloor \ (0 \le t \le j), \tag{2}$$

respectively. Here, Y^α is the input for the target operation performed by $e_{k-(j+1)}$, and Z^β is the input for the reference operation. If $e_{k-(j+1)} = 0$, the operation of Y^α is the same as that of Z^β. In contrast, if $e_{k-(j+1)} = 1$, the operation of Y^α is a multiplication, and is different from that of Z^β. As a result, the bit $e_{k-(j+1)}$ is obtained by comparing the target waveforms of Y^α and the reference waveform of Z^β.

Fig. 3 shows an example of the direct (bit/digit) estimation of **ALGORITHM 1**. Suppose that the attacker already knows the first four bits ($E^{(4)} = 1100_2$). In this condition, α and β are given as $\alpha = 24$ and $\beta = 1, 3, 6, 12$, or 24. In order to estimate the next key bit, a message pair Y and Z, which meets the condition $Y^{24} = Z^3$ (i.e., $\alpha = 24$ and $\beta = 3$) is used. Here, Y^{24} is the input for the target operation, and Z^3 is the input for the reference operation. If $\beta = 24$ ($Y^{24} = Z^{24}$), then $Y = r$ and $Z = -r$. Therefore, this attack is identical to

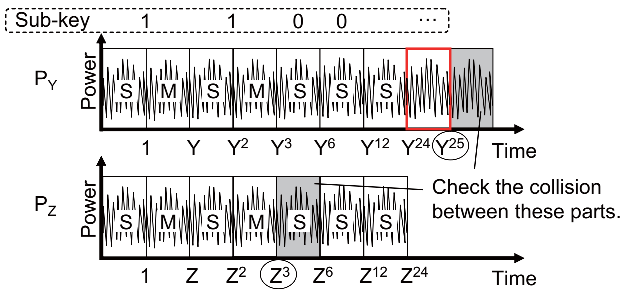


Fig. 4. Attack on the binary method (backward estimation).

Yen's attack [10]. If $\beta = 12$ ($Y^{24} = Z^{12}$), then $Y = r$ and $Z = r^2$, which is identical to the doubling attack [9]. Thus, these attacks are special cases of the present direct estimation.

Now, the backward estimation of **ALGORITHM 1** is explained. To estimate the key bit $e_{k-(j+1)}$, a squaring operation following the target operation for $e_{k-(j+1)}$ is investigated. Unlike the direct estimation, the bit value of $e_{k-(j+1)}$ (0 or 1) is estimated first, and the input message pair is then selected so that the power waveform for the squaring *following* the target operation would match the waveform for the reference operation. Assuming that $e_{k-(j+1)} = 1$, the message pair Y and Z is selected so as to meet the condition $Y^{\alpha+1} = Z^{\beta}$. If the estimation of $e_{k-(j+1)}$ is correct, the operating sequence and data for the squaring of $Y^{\alpha+1}$ are the same as those of Z^{β}, and the two waveforms of the squaring would be identical. In contrast, if the estimation is incorrect, the two square waveforms would be different.

Fig. 4 shows an example of the backward (bit/digit) estimation against the binary method. Assuming that the target key bit is 1, and the message pair is selected to meet the condition $Y^{25} = Z^3$. If the estimation is correct, a multiplication $Y^{24} \times Y$ is performed as the target operation and the result of Y^{25} is fed to the following squaring. Therefore, the same input values Y^{25} and Z^3 $(= Y^{25})$ are used for the squaring operations that generate two power waveforms to be compared. If the target key bit is 0, the target operation is squaring, and the input of the following squaring is Y^{48} $(= Y^{24\times 2})$, which is not equal to Z^3, and thus the two waveforms for the squaring do not match.

As described above, the direct estimation compares the two waveforms generated by the reference (square) and the target (unknown) operations with the same input data to determine the target operation. In contrast, the backward estimation compares the two waveforms generated by square operations to determine the input data to the squaring following the target (unknown) operation. In order to determine the operation or the data using waveform matching, the proposed method controls the relation between the messages Y and Z as Equations (1) and (2).

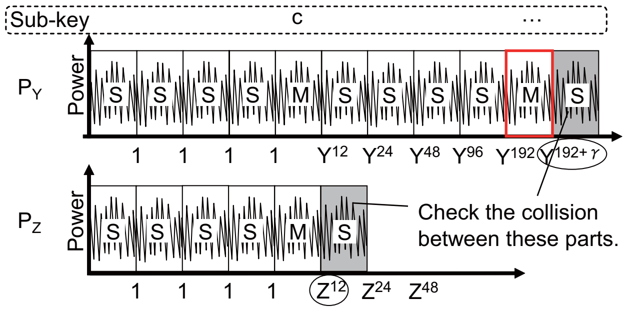

Sub-key
c
…
P_Y
Power
S S S S M S S S S M S
1 1 1 1 Y^12 Y^24 Y^48 Y^96 Y^192 Y^192+γ
P_Z
Power
S S S S M S
1 1 1 1 Z^12 Z^24 Z^48
Check the collision between these parts.

Fig. 5. Attack on the m-ary methods (backward estimation).

3.2 Attack on m-ary methods.

The backward estimation has no additional advantage over the direct estimation for attacking the conventional binary method. However, the backward estimation is essential when attacking the m-ary method shown in **ALGORITHM 2**. This algorithm always performs a multiplication after raising the intermediate result to the power of 2^m (i.e., m squaring operations). Therefore, the direct estimation, which detects the multiplication performed only if the corresponding key bit is 1, cannot be applied. Suppose that the m-bit sub-keys $E^{(j)} = (e_{k-1}, ..., e_{k-j})_{2^m}$ of the secret exponent E have already been obtained. To estimate the next sub-key $e_{k-(j+1)}$, the waveform of the squaring following the target multiplication is investigated. At the beginning of the attack, the target sub-key $e_{k-(j+1)}$ is assumed as γ $(0 \leq \gamma \leq 2^m - 1)$, and the message pair Y and Z is selected to meet the condition $Y^{\alpha+\gamma} = Z^{\beta}$, where the α and β are given as

$$\alpha = 2^m E^{(j)}, \tag{3}$$

$$\beta = \left\lfloor \frac{\alpha}{2^{mt}} \right\rfloor \ (0 \leq t \leq j), \tag{4}$$

respectively. If the estimation is correct $(e_{k-(j+1)} = \gamma)$, the input data $Y^{\alpha+\gamma}$ to the squaring following the target multiplication is the same as the Z^{β} input in the reference squaring, and thus the waveforms for the two squaring operations would match. Even if the estimation is wrong, the correct sub-key can be obtained after 2^m trials at most.

Fig. 5 shows an example of the attack against the m-ary algorithm of **ALGORITHM 2**, where $m = 4$. When the sub-key $e_{k-1} = 12$ is already known, α and β can be given by $\alpha = 192$ and $\beta = 12$. Assuming that e_{k-2} is γ, a message pair Y and Z is selected to meet the condition $Y^{192+\gamma} = Z^{12}$. If the estimation is correct, the input of the squaring $(Y^{192+\gamma})$ following the target operation is equal to that of the reference squaring (Z^{12}), and these inputs would make identical waveforms. In this case, the correct sub-key e_{k-2} can be estimated with at most $2^4 = 16$ trials.

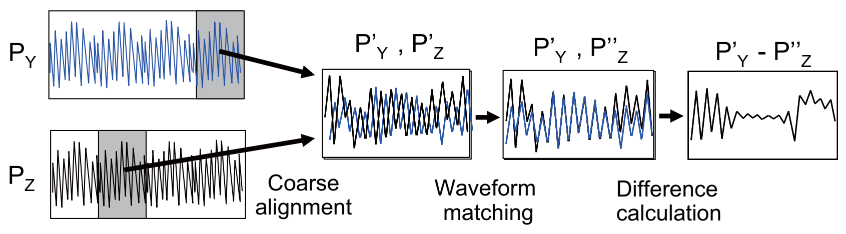


Fig. 6. Identification of operations using waveform matching.

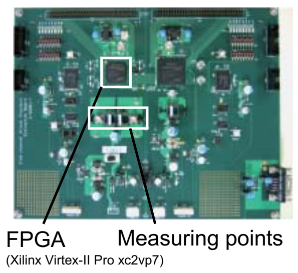


Fig. 7. Evaluation board.

Experimental FPGA board (SASEBO)	
FPGA	Virtex-II Pro xc2vp7
Crystal oscillator	24-MHz
Resistance value	1 Ohm
Power supply voltage	3.3 V
Experimental equipment	
Digital oscilloscope	Agilent MSO6104A
Probe	Coaxial cable (50 Ohm)

Fig. 8. Experimental conditions.

4 Experiments

4.1 Identification of operations by waveform matching

The proposed attacks create collisions between target and reference power waveforms at time frames which can be far apart, whereas previous attacks compare the waveforms at adjacent time frames or at the same time frame, as shown in Figs. 1 and 2. Therefore, a flexible and precise matching technique which can overcome the cumulative effect of clock jitter and noise is crucial for collision detection. In the following, the phase-based waveform matching technique [15], which can match waveform positions with a resolution higher than the sampling resolution, is used. Fig. 6 shows an overview of the identification method. Given two power traces P_Y and P_Z, we first cut out the waveform segments that include the target and reference operations, P'_Y and P'_Z, respectively. The segments can easily be recognized because each multiplication or square operation consumes less power around the boundaries of the operation. The waveform segments are then aligned precisely using the phase-based waveform matching technique. Finally, the difference between the waveforms is calculated to evaluate the equality of the operations or data being processed.

4.2 Experimental results

RSA hardware and software using the Montgomery multiplication algorithm were implemented on the Xilinx FPGA platform Side-channel Attack Standard

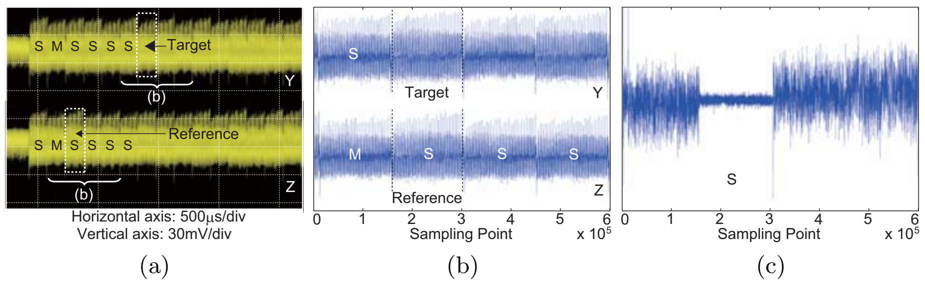


Fig. 9. Results of hardware implementation (target: squaring):(a) power traces of Y and Z, (b) waveform segments, and (c) differential waveform.

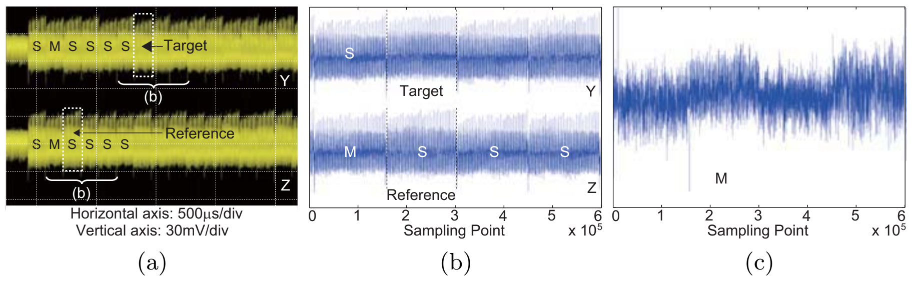


Fig. 10. Results of hardware implementation (target: multiplication):(a) power traces of Y and Z, (b) waveform segments, and (c) differential waveform.

Evaluation BOard (SASEBO) [16] shown in Fig. 7. The RSA hardware with the FPGA's embedded multipliers performs 1,024-bit modular exponentiation using the binary method. On the other hand, the RSA software is executed as a PowerPC processor macro in the FPGA, where both binary and 4-ary methods are applied to a 256-bit exponent due to memory limitations.

The power traces were monitored using an oscilloscope (Agilent MSO 6104A) at 400 Msamples/sec for software and 800 Msamples/sec for hardware as voltage drops caused by the resistor inserted between the FPGA ground pin and the ground plane. Fig. 8 summarizes the experimental conditions.

Figs. 9 and 10 show the experimental results of the direct estimation using power traces generated by the RSA hardware with two different keys. The measured power waveforms in Figs. 9 (a) and 10 (a) are aligned on the reference and target time frames as (b), and then the differential waveforms in (c) are calculated. In order to reduce the noise distortion of the differential waveform, low-pass filtering techniques, as well as phase-based waveform matching, are applied. The result is extremely clean, producing a greatly reduced difference signal when the two squared values are the same. In Figs. 9 and 10, the first four bits of the exponents are the same and are known as "1101", and each 5-th key bit will be identified. As described in the example operation of Fig. 3, a message pair Y and Z that satisfies $Y^{24} = Z^3$ is used for the identification. The amplitude

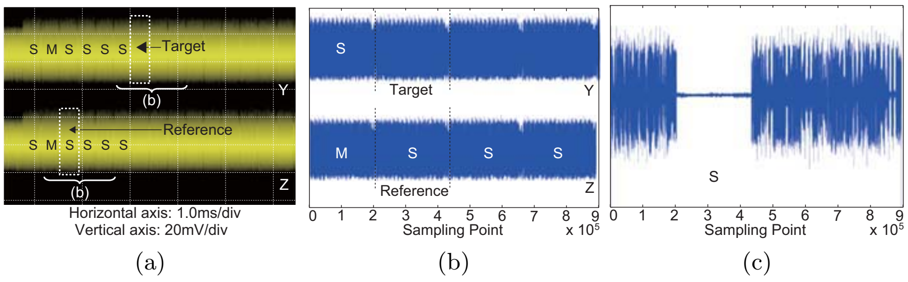


Fig. 11. Results of software implementation (target: squaring):(a) power traces of Y and Z, (b) waveform segments, and (c) differential waveform.

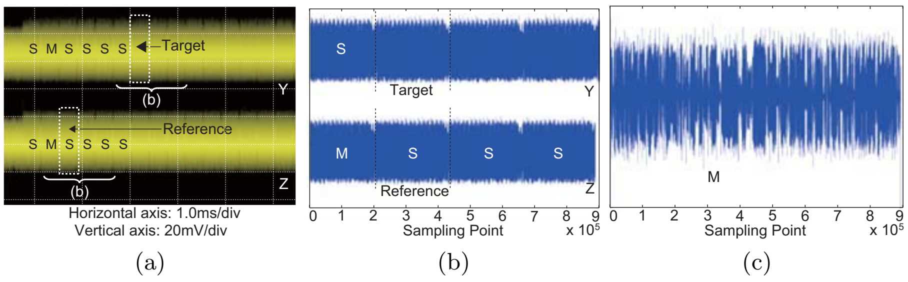
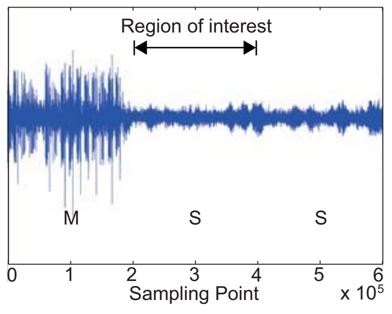


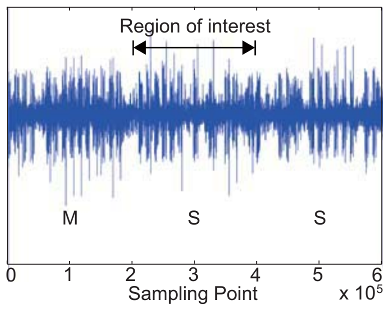
Fig. 12. Results of software implementation (target: multiplication):(a) power traces of Y and Z, (b) waveform segments, and (c) differential waveform.

of the differential waveform in Fig. 9 (c) remains around zero, and thus the target (unknown) and reference (square) operations are the same. As a result, the target operation is squaring, and the 5-th key bit is identified as 0. In contrast, the differential waveform in Fig. 10 (c) indicates that the target and reference operations do not match. Therefore, the target operation is multiplication, and the 5-th key bit is revealed to be 1. Figs. 11 and 12 show the experimental results of the software implementation of RSA with the same algorithm and parameters used in Figs. 9 and 10, respectively. By applying the same matching techniques used for the hardware implementation, the secret key bits (target operations) can be easily identified.

Fig. 13 shows the differential waveforms derived from the backward estimation applied to the RSA software using the 4-ary method, where the known sub-key is 12. As described in Section 3.2 using the example operation of Fig. 5, a message pair Y and Z that meets the condition $Y^{192+\gamma} = Z^{12}$ was executed by the RSA software. The parameter γ denotes the next unknown 4-bit sub-key, and thus all sixteen possible sub-keys 0000 $\sim$ 1111 were tested. Figs. 13 (a) and 13 (b) show the differential waveforms for the correct sub-key ($\gamma = 3$) and for one of the fifteen incorrect sub-keys ($\gamma = 4$), respectively. The correct waveform is easily distinguished from the incorrect waveforms. For additional details, Root Mean Square (RMS) and maximum errors in the differential waveforms



(a)



(b)

Fig. 13. Results of software implementation based on the 4-ary method:(a) differential waveform of correct estimation ($\gamma = 3$), and (b) differential waveform of incorrect estimation ($\gamma = 4$).

Table 1. RMS and maximum errors of differential waveforms

Key guess	0	1	2	**3**	4	5	6	7
RMS error	1.92	2.11	1.98	**1.27**	1.77	2.91	1.75	1.95
Max. error	11.39	11.55	12.05	**4.86**	11.76	12.41	11.70	11.50
Key guess	8	9	a	b	c	d	e	f
RMS error	1.96	1.89	1.74	2.11	1.90	1.82	2.21	2.07
Max. error	11.78	11.52	11.43	12.53	12.29	11.07	12.83	12.55

are shown in Table 1. In addition to visual observation, Table 1 can be used to automate the computation of the correct key bits.

The above results demonstrate that the proposed attacks can defeat both binary and m-ary methods. The m-ary method was not implemented in hardware due to memory limitations. But the proposed attack would defeat RSA hardware with the m-ary method as well as RSA software implementations, judging from the results of RSA hardware with the binary method. In addition to the logical approach, signal processing techniques such as phase-based matching and filtering greatly reduced the noise disturbing the correlation check between the target and reference waveforms. The same squaring operations can then be identified by numerical (RMS and maximum error) evaluation as well as visual observation. Although waveforms are not shown in the present study, the right-to-left binary method under the same condition described above was also defeated by the proposed attacks. Furthermore, the proposed attacks can be adapted to sliding window methods by combining the attacks against the binary and m-ary methods. These results clearly indicate that the proposed attacks are better than the previous attacks, which can only be applied to some of the implementations.

5 Conclusions

In this paper, we proposed new power-analysis attacks using chosen-message pairs against a variety of modular exponentiation algorithms. The message pairs are selected to have an exponential relationship in order to identify the same squaring operations which are performed at different time frames as determined by the bit pattern of the secret exponent. The proposed attacks can be adapted to all the standard exponentiation algorithms such as left-to-right/right-to-left binary methods, m-ary methods, and sliding window methods. Notice that standard message padding techniques such as OAEP provide no protection against our attacks: even though the chosen Y and Z ciphertexts are unlikely to produce validly padded plaintexts, this fact will be discovered only after the modular exponentiations will take place, and thus the attacker can recover the secret exponent even when no plaintexts are provided by the decryption process.

The effectiveness of the proposed attacks was demonstrated by experiments on RSA hardware/software implementations with the Montgomery multiplication algorithm. We also introduced signal processing techniques to reduce the expected noise distortion in the waveform comparison process. The proposed attacks derived the secret exponents from both binary methods and m-ary methods independently of the implementation platform. The values of the message pair can be selected arbitrarily. Therefore, the proposed attacks can also be applied to CRT implementations with/without the Montgomery multiplication algorithm, in which the relationship is controllable. In addition, dummy multiplication inserted as an SPA countermeasure for the left-to-right binary method can easily be detected by the new backward estimation technique which compares a squaring waveform following the true or dummy multiplication waveform with the reference waveform.

The right-to-left binary method with the squaring-and-multiply-always technique [17] and the blinding techniques [3] can still be used as effective countermeasures against the proposed attacks. Note however that the blinding techniques for the exponent and the message should be used simultaneously because each one of them separately can be defeated by the proposed attacks. For example, the mask updating technique in [3, 14] is vulnerable to the proposed attacks as suggested in [9]. With regard to m-ary methods, the randomized m-ary methods [18, 19] would also work as countermeasures.

The proposed chosen-message attacks provide a flexible relationship between two input messages and can generate waveform collisions in different time frames. The phase-based waveform matching with filtering technique enables high-accuracy alignment and collision detection between reference and target waveforms in any time frames independently of the algorithms, implementations, and platform. As a whole, the proposed methods and techniques make it possible to apply comparative power-analysis attacks to additional RSA implementations, using a very small number of chosen messages. Further research is being conducted to expand the applicable scope of the attacks even further (e.g., to exponentiation algorithms based on addition chains), and to overcome a variety of possible countermeasures.

References

1. P. Kocher, J. Jaffe, and B. Jun,"Differential power analysis," *CRYPTO 1999, LNCS*, Vol. 1666, pp. 388–397, August 1999.
2. P. Kocher, R. Lee, G. McGraw, and A. Raghunathan,"Security as a new dimension in embedded system design,In *Proc. the 41st annual conference on Design automation*, pp. 753–760. ACM Press, June 2004.
3. P. Kocher,"Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," *CRYPTO 1996, LNCS*, Vol. 1109, pp. 104–113, August 1996.
4. T. S. Messerges, E. A. Dabbish, and Sloan. R. H.,"Power analysis attacks of modular exponentiation in smartcards," *CHES 1999, LNCS*, Vol. 1717, pp. 144–157, August 1999.
5. W. Schindler,"A timing attack against RSA with the Chinese remainder theorem," *CHES 2000, LNCS*, Vol. 1965, pp. 109–124, August 2000.
6. C. D. Walter and S. Thompson,"Distinguishing exponent digits by observing modular subtractions," *CT-RSA 2001, LNCS*, Vol. 2020, pp. 192–207, April 2001.
7. R. Novak,"SPA-based adaptive chosen-ciphertext attack on RSA implementation," *PKC 2002, LNCS*, Vol. 2274, pp. 252–262, February 2002.
8. B. D. Boer, K. Lemke, and G. Wicke,"A DPA attack against the modular reduction within a CRT implementation of RSA," *CHES 2002, LNCS*, Vol. 2523, pp. 228–243, August 2002.
9. A. P. Fouque and F. Valette,"The doubling attack -why upwards is better than downawards," *CHES 2003, LNCS*, Vol. 2779, pp. 269–280, September 2003.
10. S. M. Yen, W. C. Lien, S. J. Moon, and J. C. Ha,"Power analysis by exploiting chosen message and internal collisions - vulnerability of checking mechanism for RSA-decryption.," *Mycrypt 2005, LNCS*, Vol. 3715, pp. 183–195, September 2005.
11. P. L. Montgomery,"Modular multiplication without trial division," *Math. Comp.*, Vol. 44, No. 170, pp. 519–521, 1985.
12. J. A. Menezes, C. P. Oorschot, and A. S. Vanstone,*Handbook of Applied Cryptography*,CRC Press, 1997.
13. C. K. Koc,"High-speed RSA implementation,Technical Report TR201, RSA Laboratories, November 1994.
14. J. S. Coron,"Resistance against differential power analysis for elliptic curve cryptosystems," *CHES 1999, LNCS*, Vol. 1717, pp. 192–302, August 1999.
15. N. Homma, S. Nagashima, Y. Imai, T. Aoki, and A. Satoh,"High-resolution side-channel attack using phase-based waveform matching," *CHES 2006, LNCS*, Vol. 4249, pp. 187–200, May 2006.
16. Side-channel Attack Standard Evaluation Board (SASEBO) http://www.rcis.aist.go.jp/special/SASEBO/.
17. M. Joye,"Highly regular right-to-left algorithms for scalar multiplication," *CHES 2007, LNCS*, Vol. 4727, pp. 135–147, September 2007.
18. C. D. Walter,"MIST: An efficient, randomized exponentiation algorithm for resisting power analysis," *CT-RSA 2002, LNCS*, Vol. 2271, pp. 53–66, April 2002.
19. K. Itoh, J. Yajima, and M. Takenaka,"DPA countermeasures by improving the window method," *CHES 2002, LNCS*, Vol. 2523, pp. 303–317, August 2002.