International Journal of Internet, Broadcasting and Communication Vol.16 No.1 17-28 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.1.17

IJIBC 24-1-3

Montgomery Multiplier with Very Regular Behavior

Yoo-Jin Baek

Associate Professor, Department of Information Security, Woosuk University
yoojin.baek@gmail.com

Abstract

As listed as one of the most important requirements for Post-Quantum Cryptography standardization process by National Institute of Standards and Technology, the resistance to various side-channel attacks is considered very critical in deploying cryptosystems in practice. In fact, cryptosystems can easily be broken by side-channel attacks, even though they are considered to be secure in the mathematical point of view. The timing attack(TA) and the simple power analysis attack(SPA) are such side-channel attack methods which can reveal sensitive information by analyzing the timing behavior or the power consumption pattern of cryptographic operations. Thus, appropriate measures against such attacks must carefully be considered in the early stage of cryptosystem's implementation process. The Montgomery multiplier is a commonly used and classical gadget in implementing big-number-based cryptosystems including RSA and ECC. And, as recently proposed as an alternative of building blocks for implementing post quantum cryptography such as lattice-based cryptography, the big-number multiplier including the Montgomery multiplier still plays a role in modern cryptography. However, in spite of its effectiveness and wide-adoption, the multiplier is known to be vulnerable to TA and SPA. And this paper proposes a new countermeasure for the Montgomery multiplier against TA and SPA. Briefly speaking, the new measure first represents a multiplication operand without 0 digits, so the resulting multiplication operation behaves in a very regular manner. Also, the new algorithm removes the extra final reduction (which is intrinsic to the modular multiplication) to make the resulting multiplier more timing-independent. Consequently, the resulting multiplier operates in constant time so that it totally removes any TA and SPA vulnerabilities. Since the proposed method can process multi bits at a time, implementers can also trade-off the performance with the resource usage to get desirable implementation characteristics.

Keywords: *Side-Channel Attack, Montgomery Multiplier, RSA, Countermeasure*

1. INTRODUCTION

As the threat of the quantum computer to the conventional cryptography grows, the effort of mitigating such a menace is also attracting more interest of the public and the research community. To cope with this phenomenon, National Institute of Standards and Technology (NIST) formally initiated the process of standardizing post-quantum cryptography in 2016. And, as quoted in the following [1]

Manuscript Received: December. 6, 2023 / Revised: January. 3, 2024 / Accepted: January. 12, 2024
Corresponding Author: yoojin.baek@gmail.com
Tel: +82-63-290-1221, Fax: +82-63-290-1518
Associate Professor, Department of Information Security, Woosuk University, Korea

"Another case where security and performance interact is resistance to side-channel attacks. … We further note that optimized implementations that address side-channel attacks (e.g., constant-time implementations) are more meaningful than those which do not."

one of the most important requirements for Post-Quantum Cryptography standardization process is resistance to various side-channel attacks including the timing attack.

Cryptographic attacks can, in general, be classified into two categories, mathematical attacks and side-channel attacks. And, it is well known that, even though being proved to be secure in the point of view of mathematical attacks, cryptographic algorithms can be broken by side-channel attacks [2].

The side-channel attacks take advantage of side-channel information captured in cryptographic computations to retrieve secret materials in devices. And, the side-channel information includes computation time, power consumption pattern, electromagnetic emission and so on. Since the introduction of the timing attack in [3], many papers have considered various side-channel attack methods and the corresponding countermeasures. And this paper mainly concerns about protecting cryptographic systems from the timing attack and the simple power attack.

The timing attack(TA) relies on the fact that the timing information is highly dependent of the input data and the internal operations [3]. Consequently, the attack tries to recover secret information by analyzing the timing profile of cryptographic algorithms' execution. On the other hand, the power analysis attack recovers secret keys from the power consumption pattern of cryptographic devices [2]. Especially, the simple power attack(SPA), one category of the power attack methods, observes one or a few power traces, from which it distinguishes between various cryptographic primitives

The Montgomery multiplier [5] is popularly used for efficiently implementing the RSA cryptosystem. However, as recently proposed as an alternative of building blocks for implementing post quantum cryptography such as lattice-based cryptography [7], the multiplier can also play a role in implementing various post quantum algorithms. And, since the multiplier may be vulnerable to TA and SPA as shown in the subsequent, it should carefully be designed to address adequate countermeasures.

The TA and SPA vulnerabilities of the modular multiplication may come from several leakage sources. First of all, its different behavior in handling certain input digits may be problematic. For example, when the multiply-and-then-reduce strategy is used for computing $ab \bmod N$ for given three integers a, b and N, the multiplication algorithm first represents the operand a into $a = a_i 2^{wi}$ for the fixed window size w. Then, it repeatedly compute $a_i b$ and accumulates the result to a register. Thus, if $a_i = 0$, then no action is taken to compute $a_i b$, which would show a different behavior compared with $a_i \neq 0$. And, this phenomenon can result in some TA/SPA vulnerabilities. Another leakage, which is specific to the Montgomery multiplier, may come from the extra reduction step. That is, the execution of Step 5 of Algorithm 1 in the next section may occur or not, dependent on the input values, which may be investigated by side-channel attackers. To prevent these weaknesses, this paper proposes a new regular Montgomery-type multiplier which has the following properties:

- The new multiplier transforms one of operands into a representation without 0 digits. Thus, the multiplications by the digit 0 are removed during the operation.

- The proposed digit transformation method is implemented using simple closed formula, so its implementation does not involve any conditional statements. It is worth noting that, in addition to giving

a resistance to TA and SPA, this property is also helpful in preventing a kind of fault attacks which makes the conditional statements bypassed and then analyses the faulty output to get some meaningful information.

- The digit manipulation of the new multiplier can be performed on-the-fly fashion with negligible performance degradation, which is beneficial in the memory-constrained circumstance.

The proposed method can process multi bits at a time, so implementers can trade-off the performance with the resource usage to get desirable implementation characteristics.

2. PRELIQUISITE

2.1 TIMING ATTACK AND SIMPLE POWER ATTACK

The side-channel attack is a very powerful implementation attack method and deals with side-channel information from cryptographic computations. In fact, real-world cryptographic devices may pour out some unintentional information related to the internal states or operations and if this information can be used to retrieve some secret material in the devices, it is called the 'side-channel information'. The side-channel information includes computation time, power consumption pattern, electromagnetic emission and so on. There are proposed various side-channel attack methods and the corresponding countermeasures in the literature.

The timing attack(TA) tries to recover secret information by analyzing the timing profile in the execution of cryptographic algorithms and highly relies on the fact that the timing information is dependent of the input data and the internal operations [3]. Especially, the attack has been proven to be practical when the authors of [4] successfully applied it to the SSL-based network web server using the CRT-based RSA cryptosystem.

On the other hand, the power attack, one of the most powerful side-channel attack methods, recovers secret keys from the power consumption pattern of cryptographic devices [2]. There are proposed two basic categories of power attack techniques, that is, the simple power analysis attack (SPA) and the differential power analysis attack (DPA). SPA observes one or a few power signals, from which it tries to distinguish between various cryptographic primitives. For example, the modular multiplication and the modular squaring, the basic operations adopted in the RSA cryptosystem, are known to be distinguishable from each other by their power consumption pattern. Different than SPA, DPA collects lots of power consumption data and uses sophisticated statistical tools to get some useful information from these data.

It is known that cryptosystems without countermeasures are highly vulnerable to side-channel attacks even though they are proven to be mathematically secure. Thus, cryptosystems' implementers must carefully design appropriate measures to prevent such side-channel attacks. For example, as countermeasures against DPA, a variety of randomization methods were introduced so far, including the random scalar blinding and the random point blinding applicable to the elliptic curve cryptosystems [8]. On the other hand, the double-and-add always method [8], its right-to-left variant [9], the Montgomery powering ladder [5,10] and various exponent recoding techniques [11,12,13] are proposed for preventing SPA. Note that these SPA countermeasures can also be used to mitigate from TA since the basic mechanisms of performing TA and SPA are very similar.

However, all these methods are only applicable for securing the very high-level operations, say, exponentiation or scalar multiplication. In other words, they don't consider the SPA-and TA-resistance of, for example, the multiplication operation, which is the basic building block for exponentiation. And this paper concerns secure implementation of the modular multiplication against SPA and TA, keeping in mind that SPA-

and TA-resistant multipliers are the essential part of securely implementing the whole exponentiation operation. Especially, since the Montgomery multiplier is commonly used for efficiently implementing cryptosystems based on big-number arithmetic, this paper mainly devotes to securely implementing the Montgomery multiplier against TA and SPA.

Another noteworthy TA and SPA countermeasure is given in [14]. The authors first defined the concept of 'side-channel equivalent'. Briefly speaking, two instructions are side-channel equivalent if they are indistinguishable in the side-channel-point of view. And then, they concluded that, given a group in which cryptographic operations are taking place, if multiplication-like and squaring-like operations are side-channel equivalent, the exponentiation-like operation can be implemented in the TA and SPA secure manner with negligible computational overhead. However, the authors didn't mention how to make multiplication-like and squaring-like operations be side-channel equivalent. Thus, if the proposed countermeasure for the Montgomery multiplier really behaves regularly irrespective of input data, the resulting multiplication and squaring operations can be used as a side-channel equivalent instructions.

2.2 MONTGOMERY MULTIPLIER

Given a modulus N and a public/private exponent pair (e, d), the RSA cryptosystem [15] encrypts a message m by computing $c = m^e \bmod N$ and decrypts a ciphertext c by calculating $m = c^d \bmod N$. Thus, the modular exponentiation is the main operation of RSA (and some big-number-based cryptosystems) and should be implemented in a secure manner to side-channel attacks.

In general, the modular exponentiation is implemented by applying modular multiplications and modular squarings iteratively. And the modular multiplication (and the modular squaring as well) can be realized with the school-book method which is basically a composition of a multi-precision integer multiplication and a modular reduction. Also, even though the classical modular multiplication is believed to be more adequate for implementing a single modular multiplication operation, it is also known that the Montgomery multiplier is a very effective gadget for implementing the modular exponentiation since it substitutes the expensive modular reduction by a cheap shift operation [16].

Given two (big) integers a, b and a modulus N, the Montgomery multiplier computes $abR^{-1} \bmod N$, where R is a specifically chosen constant which is greater than N and generally has the form of a power of 2 [5]. The following algorithm describes the exemplary realization of the 2^w-ary Montgomery multiplier [16].

Algorithm 1 (2^w-ary Montgomery Multiplier)
Input (N, a, b, w, k, R) with

- N, an odd modulus
- w, the fixed window size
- k, a positive integer satisfying $N < 2^{wk}$
- $R = 2^{wk}$
- a and b, two positive integers with $a, b < N$ (in the algorithm below, a is expressed as $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \ldots, 2^w - 1\}$ for all i)

Output $abR^{-1} \bmod N$

1. $b_0 \leftarrow b \bmod 2^w$
2. $I \leftarrow -N^{-1} \bmod 2^w$
3. $S \leftarrow 0$

4. For $i = 0$ to $k - 1$
   a. $u \leftarrow (s_0 + a_i b_0) I \bmod 2^w$ for $s_0 = S \bmod 2^w$
   b. $S \leftarrow (S + a_i b + uN)/2^w$
5. If $S \geq N$, then $S \leftarrow S - N$.
6. Return S

Even though the Montgomery multiplier shows a good performance feature, it may also have some security-related pitfalls. In particular, in the view of the security against TA and SPA, the following two points should be considered as a potential leakage source. The first one is about Step 4.b of Algorithm 1. Clearly, the step will behave differently according to the values of a_i and u. More precisely, if they are both nonzero, then there must occur two multi-precision integer additions by $a_i b$ and uN followed by a division by 2^w (which can be realized by a right-shift operation). On the other hand, if a_i or u is zero, then there is occurring at most one multi-precision integer addition followed by a division, which is distinguishable for the previous case by TA or SPA attackers. The second point comes from Step 5 of Algorithm 1. That is, the step contains a conditional final subtraction $S - N$ and the triggering condition is obviously dependent on the input values a, b (for a fixed N). Consequently, both points may be investigated by TA or SPA attackers, so must be removed in the implementation. And, this paper tries to eliminate these two weaknesses to get a highly regular multiplier.

3. REGULAR MONTGOMERY MULTIPLIER

As noted in the previous section, the different behavior in processing digits of the a-operand in Algorithm 1 may lead to TA and SPA vulnerabilities. And, to prevent such undesirable phenomenon, we begin with converting all digits of a into non-zero digits so that a is represented without 0-digit. Thus, assuming that, for a fixed window size $w(\geq 1)$, a is represented by $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \dots, 2^w - 1\}$, we change the representation into $a = \sum_{i=0}^{k} a'_i 2^{wi}$ with $a'_i \in \{\pm 1, \pm 2, \dots, \pm 2^{w-1}, \pm 2^w\}$ for $i = 0, 1, \dots, k-1$ and $a'_k \in \{0, 1\}$. The following points are worthy of noting for understanding the new representation:

- The new digit set $\{\pm 1, \pm 2, \dots, \pm 2^{w-1}, \pm 2^w\}$ for $a'_i, 0 \leq i \leq k-1$ does not contain 0 while it includes negative values.
- For $0 \leq i \leq k-1$, $0 < a'_i \leq 2^{w-2}$ or $a'_i = 2^w$. Thus, the absolute value of new digits except of a'_k cannot take the values between 2^{w-1} and 2^w.
- The new representation is incomplete in the sense that the new most significant digit a'_k can take 0 and this problem will be fixed later.

The basic procedure of converting the representation is as follows: from the least significant digit to the most significant digit, if we encounter a 0-digit, the digit is converted to -2^w and the next digit is added by 1 to adjust the whole value. And, this is repeated until the last digit. More precisely, letting $c_0 = 0$, the carry bit c_{i+1} and the new digit a'_i for $0 \leq i \leq k-1$ are inductively defined as: for $x_i = a_i + c_i$,

$$(c_{i+1}, a'_i) = \begin{cases} (1, -2^w) & \text{if } x_i = 0 \\ (0, x_i) & \text{if } 0 < x_i \leq 2^{w-1} \\ (1, x_i - 2^w) & \text{if } 2^{w-1} < x_i < 2^w \\ (0, 2^w) & \text{if } x_i = 2^w \end{cases} \quad (1)$$

Also, a'_k is defined to be c_k.

Lemma 1 When (1) is applied to $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \dots, 2^w - 1\}$, the followings hold.

(i) $0 \le x_i \le 2^w$ for $0 \le i < k$, hence (1) takes into consideration all cases of x_i.

(ii) $x_i = c_{i+1}2^w + a'_i$ for $0 \le i < k$

(iii) $a = c_k 2^{wk} + \sum_{i=0}^{k-1} a'_i 2^{wi} = \sum_{i=0}^{k} a'_i 2^{wi}$

(iv) $a'_i \in \{\pm 1, \pm 2, \dots, \pm 2^{w-1}, \pm 2^w\}$ for $0 \le i < k$ and $a'_k \in \{0,1\}$.

Proof (i) is true since c_i is 0 or 1 and $0 \le a_i < 2^w$. (ii) can be proved case by case. If $x_i = 0$, $(c_{i+1}, a'_i) = (1, -2^w)$ thus $c_{i+1}2^w + a'_i = 0 = x_i$. If $0 < x_i \le 2^{w-1}$, $(c_{i+1}, a'_i) = (0, x_i)$ and $c_{i+1}2^w + a'_i = x_i$. If $2^{w-1} < x_i < 2^w$, $c_{i+1}2^w + a'_i = (1, x_i - 2^w)$ and $c_{i+1}2^w + a'_i = 2^w + (x_i - 2^w) = x_i$. Finally, if $x_i = 2^w$, $c_{i+1}2^w + a'_i = (0, 2^w)$ thus $c_{i+1}2^w + a'_i = 2^w = x_i$. (iii) can be shown by mathematical induction. That is, letting $A_l = \sum_{i=0}^{l} a_i 2^{wi}$ and $B_l = c_{l+1}2^{w(l+1)} + \sum_{i=0}^{l} a'_i 2^{wi}$ for $l = 0, \dots, k-1$, it is obvious that $A_0 = B_0$ since $A_0 = a_0 = x_0 = c_1 2^w + a'_0 = B_0$ by (ii). And, assuming that $A_l = B_i$ for $0 \le l < k-2$, we have $A_{l+1} = a_{l+1}2^{w(l+1)} + A_l = a_{l+1}2^{w(l+1)} + B_l = a_{l+1}2^{w(l+1)} + c_{l+1}2^{w(l+1)} + \sum_{i=0}^{l} a'_i 2^{wi} = (a_{l+1} + c_{l+1})2^{w(l+1)} + \sum_{i=0}^{l} a'_i 2^{wi}$. Now, since $a_{l+1} + c_{l+1} = x_{l+1}$ by definition and $x_{l+1} = c_{i+2}2^w + a'_{l+1}$ by (2), we get $A_{l+1} = (a_{l+1} + c_{l+1})2^{w(l+1)} + \sum_{i=0}^{l} a'_i 2^{wi} = (c_{i+2}2^w + a'_{l+1})2^{w(l+1)} + \sum_{i=0}^{l} a'_i 2^{wi} = c_{l+2}2^{w(l+2)} + \sum_{i=0}^{l+1} a'_i 2^{wi} = B_{l+1}$, which complete the proof of (iii). (iv) is obvious. ■

From Lemma 1, we can eventually conclude that the new representation does not change the original value of a and gives the properties explained above. Table 1 gives an example of how the rule (1) can be applied to the 2^2-ary representation of integers in $\{0,1,\dots,63\}$. In the table, the bar notation stands for the minus value, for example, $\bar{1}$ means -1.

Table 1 New representation in a 2^2-ary fashion of integers in $\{0, 1, \cdots, 63\}$

Original	New	Original	New	Original	New	Original	New
000	$1\bar{4}1\bar{4}$	100	$011\bar{4}$	200	$021\bar{4}$	300	$1\bar{1}1\bar{4}$
001	$01\bar{4}1$	101	$02\bar{4}1$	201	$1\bar{1}\bar{4}1$	301	$04\bar{4}1$
002	$01\bar{4}2$	102	$02\bar{4}2$	202	$1\bar{1}\bar{4}2$	302	$04\bar{4}2$
003	$1\bar{4}1\bar{1}$	103	$011\bar{1}$	203	$021\bar{1}$	303	$1\bar{1}1\bar{1}$
010	$1\bar{4}2\bar{4}$	110	$012\bar{4}$	210	$022\bar{4}$	310	$1\bar{1}2\bar{4}$
011	$1\bar{4}11$	111	0111	211	0211	311	$1\bar{1}11$
012	$1\bar{4}12$	112	0112	212	0212	312	$1\bar{1}12$
013	$1\bar{4}2\bar{1}$	113	$012\bar{1}$	213	$022\bar{1}$	313	$1\bar{1}2\bar{1}$
020	$01\bar{1}\bar{4}$	120	$02\bar{1}\bar{4}$	220	$1\bar{1}\bar{1}\bar{4}$	320	$04\bar{1}\bar{4}$
021	$1\bar{4}21$	121	0121	221	0221	321	$1\bar{1}21$
022	$1\bar{4}22$	122	0122	222	0222	322	$1\bar{1}22$
023	$01\bar{1}\bar{1}$	123	$02\bar{1}\bar{1}$	223	$1\bar{1}\bar{1}\bar{1}$	323	$04\bar{1}\bar{1}$
030	$1\bar{4}4\bar{4}$	130	$014\bar{4}$	230	$024\bar{4}$	330	$1\bar{1}4\bar{4}$
031	$01\bar{1}1$	131	$02\bar{1}1$	231	$1\bar{1}\bar{1}1$	331	$04\bar{1}1$
032	$01\bar{1}2$	132	$02\bar{1}2$	232	$1\bar{1}\bar{1}2$	332	$04\bar{1}2$
033	$1\bar{4}4\bar{1}$	133	$014\bar{1}$	233	$024\bar{1}$	333	$1\bar{1}4\bar{1}$

However, there is still an issue in the new representation in that the new last digit a'_k can have the value 0, depending on a. And this may make the resulting multiplier with the new representation take a variable processing time so that it may induce another TA and SPA vulnerabilities. This timing variation, however, can be eliminated by appending two more digits to the new representation. That is, for the last carry bit c_k which can have the value 0 or 1, the following two digits a'_{k+1}, a'_k will be appended to the new representation:

$$(a'_{k+1}, a'_k) = \begin{cases} (1,-2) & \text{if } c_k = 0 \\ (1,-1) & \text{if } c_k = 1 \end{cases}. \quad (2)$$

Since $2a'_{k+1} + a'_k = c_k$ in (2), it is emphasized that (a'_{k+1}, a'_k) in (2) should be interpreted as a binary representation. Consequently, the new representation can be summarized as: for $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \dots, 2^w - 1\}$, a is newly represented as $a = a'_{k+1}2^{wk+1} + a'_k 2^{wk} + \sum_{i=0}^{k-1} a'_i 2^{wi} = 2^{wk+1} + a'_k 2^{wk} + \sum_{i=0}^{k-1} a'_i 2^{wi}$ with $a'_i \in \{\pm 1, \pm 2, \dots, \pm 2^{w-1}, \pm 2^w\}$ for $i = 0,1, \dots, k$.

As noted in the previous section, when a cryptographic algorithm is implemented in software or hardware, conditional statements should be avoided as much as possible since it causes data-dependent timing variations or it can be skipped by fault attackers to give rise to meaningful side-channel information. Thus, implementing the new representation conversion with rules (1) and (2) has to keep away from data-dependent conditions as well. In this sense, (2) can be expressed as $a'_k = c_k - 2$ and $a'_{k+1} = 1$ in a closed formula. And, it is preferable to devise a closed formula for (1), hence all the involved variables are implemented without any if-statements. To achieve this purpose, we need the following lemma. (In the subsequent, $\gg$ and $\ll$ stand for the right and left shift operations, respectively.)

Lemma 2 For $f, g, h: \{0, 1, \dots, 2^w\} \to \{0,1\}$ defined by $f(x) = (x + 2^{w+1} - 1) \gg (w + 1)$, $g(x) = (x + 2^{w+1} - 2^{w-1} - 1) \gg (w + 1)$ and $h(x) = (x + 2^w) \gg (w + 1)$, we have

(i) $f(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x \neq 0 \end{cases}$

(ii) $g(x) = \begin{cases} 0 & \text{if } 0 \leq x \leq 2^{w-1} \\ 1 & \text{if } 2^{w-1} < x \leq 2^w \end{cases}$

(iii) $h(x) = \begin{cases} 0 & \text{if } x \neq 2^w \\ 1 & \text{if } x = 2^w \end{cases}$

(iv) $1 - f(x) + g(x) - h(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } 0 < x \leq 2^{w-1} \\ 1 & \text{if } 2^{w-1} < x < 2^w \\ 0 & \text{if } x = 2^w \end{cases}$

Proof The proof is based on the simple observation that, for a non-negative integer x and a positive integer n, $x \gg (w + 1)$ is equal to n if $2^{w+1}n \leq x < 2^{w+1}(n + 1)$. ■

Now, comparing with (iv) of Lemma 2 with (1), we can conclude that c_{i+1} can be re-written as $c_{i+1} = 1 - f(x_i) + g(x_i) - h(x_i)$. Also, considering that a'_i in (1) can be expressed as $a'_i = x_i - (c_{i+1} \ll w)$, the final closed formula of (1) becomes of the form: for $i \geq 0$

$$c_{i+1} = 1 - \big((x_i + 2^{w+1} - 1) \gg (w + 1)\big) + \big((x_i + 2^{w+1} - 2^{w-1} - 1) \gg (w + 1)\big) - \big((x_i + 2^w) \gg (w + 1)\big) \quad (3)$$

$a_i' = x_i - (c_{i+1} \ll w)$.

As noted before, another TA and SPA leakage source of Algorithm 1 may occur at Step 4.b, which adds a multiple of the modulus N to S depending on the value u. That is, if $u \neq 0$, then the step must add to S a non-zero multiple of N, while, if $u = 0$, no addition occurs, which can be identified by TA and SPA attackers. And, this irregular behavior can be avoided by converting u into a new value u' in a similar manner as in a_i. However, there is a major difference between conversions of a_i and u: a_i should be converted in the manner that the original value a remains unchanged after the conversion, while the constraint does not apply to u since the addition by a different multiple of N in Step 4.b do not influence to the output of Algorithm 1. Keeping this in mind, the basic principle of converting u is that u is converted to 2^w if $u = 0$ and to $u - 2^w$ if $2^{w-1} < u < 2^w$. More precisely, u' is computed as

$$u' = \begin{cases} 2^w & \text{if } u = 0 \\ u & \text{if } 0 < u \leq 2^{w-1} \\ u - 2^w & if\ 2^{w-1} < u < 2^w \end{cases}. \quad (4)$$

And, similarly as for (c_{i+1}, a_i'), (4) can be turned into a closed formula as

$$u' = u + ((1 - ((u + 2^w - 1) \gg w)) \ll w) - (((u + 2^{w-1} - 1) \gg w) \ll w). \quad (5)$$

Summing up all the discussions above, the following algorithm can be obtained.

Algorithm 2 (New Montgomery Multiplier)

Input (N, a, b, w, k, R) with

- N, an odd modulus
- w, the fixed window size
- k, a positive integer satisfying $N < 2^{wk}$
- $R = 2^{wk}$
- a and b, two positive integers with $a, b < N$ (in the algorithm below, a is expressed as $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \ldots, 2^w - 1\}$ for all i)

Output $abR^{-1} \bmod N$

1. $b_0 \leftarrow b \bmod 2^w$
2. $I \leftarrow -N^{-1} \bmod 2^w$
3. $S \leftarrow 0$
4. $c \leftarrow 0$
5. For $i = 0$ to $k - 1$
   a. $x \leftarrow a_i + c$
   b. $c \leftarrow 1 - \left((x + 2^{w+1} - 1) \gg (w + 1)\right) + \left((x + 2^{w+1} - 2^{w-1} - 1) \gg (w + 1)\right) - \left((x + 2^w) \gg (w + 1)\right)$
   c. $x \leftarrow x - (c \ll w)$
   d. $u \leftarrow (s_0 + xb_0)I \bmod 2^w$ for $s_0 = S \bmod 2^w$
   e. $u \leftarrow u + \left(\left(1 - \left((u + 2^w - 1) \gg w\right)\right) \ll w\right) - \left(\left((u + 2^{w-1} - 1) \gg w\right) \ll w\right)$

f. $S \leftarrow (S + xb + uN) \gg w$
6. $x \leftarrow c - 2$
7. $u \leftarrow (s_0 + xb_0)I \bmod 2$ for $s_0 = S \bmod 2$
8. $u \leftarrow 2 - u$
9. $S \leftarrow (S + xb + uN) \gg 1$
10. $u \leftarrow (s_0 + b_0)I \bmod 2$ for $s_0 = S \bmod 2$
11. $u \leftarrow 2 - u$
12. $S \leftarrow (S + b + uN) \gg 1$
13. If $S \geq N$, then $S \leftarrow S - N$
14. Return S

Here are some explanations about Algorithm 2. In the algorithm, Step 5.b and Step 5.c are direct applications of the equations (3), while Step 5.e comes from (5). Also, Step 6 and Step 8 (and Step 11) are derived from (2), noting that, for $w = 1$, $u + ((1 - ((u + 2^w - 1))w)) \ll w) - (((u + 2^{w-1} - 1))w) \ll w)$ is exactly equal to $2 - u$.

Even though its highly regular behavior, Algorithm 2 still has a leakage source. That is, the subtraction by N in Step 13 may occur or not, depending on input values a and b. Hence, it is desirable to remove the step for preventing TA and SPA. And, this can be achieved with the help of the following lemma.

Lemma 3 If $|b| < 8N$ in Algorithm 2, the intermediate result S after Step 12 satisfies that $|S| < 8N$.
Proof The proof follows the same approach in [17] and [18]. Let S' denote the value of S after Step 5. Then, we first claim that $|S'| < 2N + 2|b|$. For the proof, put the initial and the resulting values of S in the loop with index i of Step 5 to $S_{I,i}$ and $S_{O,i}$, respectively. Then, Step 5.f can be re-written as $S_{O,i} = (S_{I,i} + xb + uN) \gg w = \frac{S_{I,i}+xb+uN}{2^w}$ and the followings also hold:

① $S_{I,0}$=0
② $S_{I,i+1} = S_{O,i}$ for $0 \leq i \leq k - 2$
③ For $0 \leq i \leq k - 1$, $S_{O,i}$ is equal to $(S_{I,i} + xb + uN) \gg w = \frac{S_{I,i}+xb+uN}{2^w}$ for some $|x|, |u| \leq 2^w$.

Now, we can use the mathematical induction to prove that $|S_{O,i}| < 2N + 2|b|$ for $0 \leq i \leq k - 1$, which completes the proof of the claim. Actually, for $i = 0$, $S_{O,i} = \frac{xb+uN}{2^w}$ for some x, u with $|x|, |u| \leq 2^w$. Thus, $|S_{O,0}| \leq \frac{|xb|+|uN|}{2^w} \leq |b| + N < 2N + 2|b|$. Next, suppose that $|S_{O,i}| < 2N + 2|b|$ for $0 \leq i \leq k - 2$. Then, since $S_{O,i+1} = \frac{S_{I,i+1}+xb+uN}{2^w} = \frac{S_{O,i}+xb+uN}{2^w}$ for some x, u with $|x|, |u| \leq 2^w$, we have $|S_{O,i+1}| \leq \frac{|S_{O,i}|+|xb|+|uN|}{2^w} < \frac{2N+2|b|+2^w|b|+2^wN}{2^w} \leq 2N + 2|b|$. Now, since $S' = S_{O,k-1}$, we can conclude that $|S'| < 2N + 2|b|$, as claimed. Next, let S_I and S_O be the initial and the resulting values of the register S in Step 9 and S'_I and S'_O be the initial and the resulting values of the register S in Step 12, respectively. Clearly, $|S_I| < 2N + 2|b|$ since $S_I = S_{O,k-1}$. Also, note that the x-value of Step 6 is -2 or -1 according to (2). Thus, since $S'_O = \frac{S'_I+b+u_1N}{2}$ for some u_1 with $|u_1| \leq 2$ and $S_O = \frac{S_I+xb+u_2N}{2}$ for some x and u_2 with $x \in \{-2, -1\}$ and $|u_1| \leq 2$, S'_O is equal to $\frac{S'_I+b+u_1N}{2} = \frac{S_O+b+u_1N}{2} = \frac{S_I+(x+2)b+(u_2+2u_1)N}{4}$ with $x \in$

$\{-2,-1\}$ and $|u_1|,|u_2| \le 2$. Therefore, we have $|S'_O| \le \frac{|S_I|+|b|+6|N|}{4}$. Since $|S_I|$ is already shown to be less than $2N+2|b|$ and $|b|$ is assumed to be less than $8N$, we can derive that $|S'_O| < 8N$, which completes the proof of the lemma. ■

Now, Lemma 3 says that, if the input b of Algorithm 2 satisfies that $|b| < 8N$, then the output satisfies the same bound even though Step 13 is removed. Thus, when a multiplication is executed via Algorithm 2 without Step 13, its result can be reused as an input of another multiplication operation. And, for example, the exponentiation algorithm may get some benefits from this process and the step of subtraction by N can be postponed until the final stage of the exponentiation operation.

Based on all the discussions above, the final regular Montgomery multiplier can be described as:

Algorithm 3 (Regular Montgomery Multiplier)

Input (N, a, b, w, k, R) with

- N, an odd modulus
- w, the fixed window size
- k, a positive integer satisfying $N < 2^{wk}$
- $R = 2^{wk}$
- a and b, two integers with $a < N$ and $|b| < 8N$ (in the algorithm below, a is expressed as $a = \sum_{i=0}^{k-1} a_i 2^{wi}$ with $a_i \in \{0, 1, \dots, 2^w - 1\}$ for all i)

Output $S = abR^{-1} \bmod N$ with $|S| < 8N$

1. $b_0 \leftarrow b \bmod 2^w$
2. $I \leftarrow -N^{-1} \bmod 2^w$
3. $S \leftarrow 0$
4. $c \leftarrow 0$
5. For $i = 0$ to $k-1$
   a. $x \leftarrow a_i + c$
   b. $c \leftarrow 1 - \left((x + 2^{w+1} - 1) \gg (w+1)\right) + \left((x + 2^{w+1} - 2^{w-1} - 1) \gg (w+1)\right) - \left((x + 2^w) \gg (w+1)\right)$
   c. $x \leftarrow x - (c \ll w)$
   d. $u \leftarrow (s_0 + xb_0)I \bmod 2^w$ for $s_0 = S \bmod 2^w$
   e. $u \leftarrow u + \left(\left(1 - \left((u + 2^w - 1) \gg w\right)\right) \ll w\right) - \left(\left((u + 2^{w-1} - 1) \gg w\right) \ll w\right)$
   f. $S \leftarrow (S + xb + uN) \gg w$
6. $x \leftarrow c - 2$
7. $u \leftarrow (s_0 + xb_0)I \bmod 2$ for $s_0 = S \bmod 2$
8. $u \leftarrow 2 - u$
9. $S \leftarrow (S + xb + uN) \gg 1$
10. $u \leftarrow (s_0 + b_0)I \bmod 2$ for $s_0 = S \bmod 2$
11. $u \leftarrow 2 - u$
12. $S \leftarrow (S + b + uN) \gg 1$
13. Return S

Now, the main achievement of Algorithm 3 lies at its regular behavior, which clearly gives resistance to various side-channel attacks. However, it inevitably bears some computational overhead, compared with Algorithm 1. Nevertheless, if the window size w is small enough, the equations (3) and (5) can be implemented using ordinary integer operations, hence their computational cost is negligible, compared with the whole modular multiplication operation. Actually, for real software applications, w is not greater than 32, so the operations in (3) and (5) can be implemented by usual integer arithmetics in the standard C library. Also, in hardware implementations, w is not greater than 16 and so the logic realizing the equations does not cause much overhead of hardware size

4. CONCLUSION

The Montgomery multiplier is popularly used for efficiently implementing some big-number based cryptosystems including RSA. And, this paper proposed a new Montgomery-like multiplier which is expected to behave in a highly regular manner.

In general, the constant-time implementation does not give a full specification of SCA countermeasures. For example, to defeat DPA, it is believed that some randomization techniques should be applied to the implementations. So, in addition to the work in this paper, it will be interesting to devise a randomized version of Montgomery multipliers, which is left for the future research.

REFERENCES

[1] National Institute of Standards and Technology, Submission requirements and evaluation criteria for the Post-Quantum Cryptography standardization process, *http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf*

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Lecture Notes in Computer Science*, Vol. 1666, pp. 388-397, August 1999. DOI: https://doi.org/10.1007/3-540-48405-1_25

[3] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *Lecture Notes in Computer Science*, Vol. 1109, pp. 104-113, August 1996. DOI: https://doi.org/10.1007/3-540-68697-5_9

[4] D. Boneh and D. Brumley, "Remote Timing Attacks Are Practical," *Computer Networks*, Vol. 48, Issue 5, pp. 701-716, August 2005. DOI: https://doi.org/10.1016/j.comnet.2005.01.010

[5] P. Montgomery, "Speeding the Pollard and Elliptic Curve Methods for Factorizations," *Mathematics of Computation*, Vol. 48, No. 177, pp. 243-264, January 1987. DOI: https://doi.org/10.1090/S0025-5718-1987-0866113-7

[6] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," *Lecture Notes in Computer Science*, Vol. 1233, pp. 37-51, May 1997. DOI: https://doi.org/10.1007/3-540-69053-0_4

[7] M.R. Albrecht, C. Hanser, A. Hoeller, T. Pöppelmann, F. Virdia, and A. Wallner, "Implementing RLWE-based Schemes Using an RSA Co-Processor," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, Vol. 2019, Issue 1, pp. 169–208, November 2018. DOI: https://doi.org/10.13154/tches.v2019.i1.169-208

[8] J. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *Lecture Notes in Computer Science*, Vol. 1717, pp. 292-302, September 1999. DOI: https://doi.org/10.1007/3-540-48059-5_25

[9] A. Boscher, R. Naciri, and E. Prouff, "CRT RSA Algorithm Protected Against Fault Attacks," *Lecture Notes in Computer Science*, Vol. 4462, pp. 229-243, May 2007. DOI: https://doi.org/10.1007/978-3-540-72354-7_19

[10] M. Joye, "Highly Regular m-ary Powering Ladders," *Lecture Notes in Computer Science*, Vol. 5867, pp. 350-363, August 1999. DOI: https://doi.org/10.1007/978-3-642-05445-7_22

[11] M. Joye and M. Tunstall, "Exponent Recoding and Regular Exponentiation Algorithms,", *Lecture Notes in Computer Science*, Vol. 5580, pp. 334-349, June 2009. DOI: https://doi.org/10.1007/978-3-642-02384-2_21

[12] B. Möller, "Securing Elliptic Curve Point Multiplication against Side-Channel Attacks," *Lecture Notes in Computer Science*, Vol. 2200, pp. 324-334, October 2001. DOI: https://doi.org/10.1007/3-540-45439-X_22

[13] C. Vuillaume and K. Okeya, "Flexible Exponentiation with Resistance to Side-Channel Attacks," *Lecture Notes in Computer Science*, Vol. 3989, pp. 268-283, June 2006. DOI: https://doi.org/10.1007/11767480_18

[14] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity," *IEEE Transactions on Computers*, Vol. 53, Issue 6, pp. 760-768, June 2004. DOI: https://doi.org/10.1109/TC.2004.13

[15] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, pp. 120-126, February 1978. DOI: https://doi.org/10.1145/359340.359342

[16] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[17] G. Hachez and J.-J. Quisquater, "Montgomery Exponentiation with no Final Subtractions: Improved Results," *Lecture Notes in Computer Science*, Vol. 1965, pp. 293-301, August 2000. DOI: https://doi.org/10.1007/3-540-44499-8_23

[18] C. Walter, "Montgomery Exponentiation Needs No Final Subtractions," *Electronics Letters*, Vol. 35, Issue 21, pp. 1831-1832, October 1999. DOI: https://doi.org/10.1049/el:19991230