IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 1, 2019, accepted November 28, 2019, date of publication January 2, 2020, date of current version January 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963317

A Combined Single Trace Attack on Global Shuffling Long Integer Multiplication and Its Novel Countermeasure

SANGYUB LEE[1], SUNG MIN CHO[2], HEESEOK KIM[3], AND SEOKHIE HONG[1]
[1]Graduate School of Information Security, Institute of Cyber Security and Privacy (ICSP), Korea University, Seoul 02841, South Korea
[2]Crypt & Tech, Seoul 02841, South Korea
[3]Department of Cyber Security, College of Science and Technology, Korea University, Sejong 30019, South Korea

Corresponding author: Seokhie Hong (shhong@korea.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korea Government (MSIT) under Grant 2017-0-00520, and in part by the Development of SCR-Friendly Symmetric Key Cryptosystem and Its Application Modes.

ABSTRACT Advanced collision-based single trace attacks which can be applied on simple power analysis resistant scalar multiplications become virtual threat on elliptic curve cryptosystems recently as their practical experimental results are increasingly reported in the literature. Since such attacks are based on detecting collisions of data dependent leakage caused by underlying long integer multiplications, so-called global shuffling countermeasure which breaks such collision correlation by independently randomizing the execution order of unit operations such as single precision multiplication and carry propagation, is considered as promising countermeasure if theoretical randomness of shuffling order is guaranteed. In this paper, we firstly analyze the practical security of the global shuffling long integer multiplications by exhibiting a combined single trace attack on software implementations on an ARM Cortex-M4 microcontroller. Our combined attack consists of a simple power analysis for revealing random permutation vectors which enables later collision-based single trace attack. First we demonstrate how to reveal random permutation vectors for carry propagation process of whole global shuffling long integer multiplications within a single power trace by simple power analysis accompanied with straightforward substitution of power consumption samples. Then we perform collision-based single trace attacks after rearranging the order of subtraces for unit carry propagations based on revealed permutation vectors. Since the vulnerability to simple power analysis is originated from the if-statement for selection of proper entries of the permutation vectors, we propose a novel countermeasure which eliminates such selection with simple addition and modulus operation and also demonstrate practical result achieving regularity in power trace patterns.

INDEX TERMS Cryptography, digital signatures, elliptic curves, public key, side-channel attacks.

I. INTRODUCTION

Elliptic curve cryptosystems (ECC) [1], [2] are widely used until recently because of their advantages providing equivalent security with shorter key length compared with other public key cryptosystems (PKC) such as RSA, DSA, and, DH. Furthermore, with the shorter keys, scalar multiplications which is the main operation of ECCs feature shorter execution time and lower memory requirement than RSA modular exponentiations. Hence ECCs are preferably deployed for PKCs on secure embedded devices.

On the other hand, side-channel analysis attacks [3], [4] which can reveal the secret from implementations of ECCs exploiting their timing, power consumption, electromagnetic emanation, etc., have been researched consistently. Since ECC protocols such as ECDSA or ECDH use an ephemeral secret, resistance against Simple Power Analysis or Timing Attacks [3] which can be performed with a single power trace is essential for secure embedded devices. The core idea of such attacks is based on distinguishing doubling and addition of a scalar multiplication

The associate editor coordinating the review of this manuscript and approving it for publication was Xiangxue Li.

such as double-and-add algorithm [5]. Hence countermeasures which operate regular scalar multiplication like double-and-add-always [6], Montgomery ladder [7], [8], and atomic scalar multiplication [9] deploying unified point addition formulae are proposed.

Nevertheless, advanced single trace attacks which can defeat such countermeasures are also proposed. Walter [10] proposed Big Mac attack which can distinguish doubling and addition operations in a non-regular scalar multiplication from a single trace utilizing Euclidean distance. Inspired by the Walter's work, Clavier *et al.* [11] proposed Horizontal Correlation Analysis (HCA) which can perform on regular scalar multiplications exploiting correlation between intermediate data and power consumption in a single trace. Bauer *et al.* [12] introduced power trace averaging techniques which can defeat countermeasures proposed by Clavier *et al.* [11] against their attack. Recovery of Secret Exponent by Triangular Trace Analysis (ROSETTA) [13] can attack scalar multiplications by determining inner-collisions of a long integer multiplication (LIM) [14] caused by the same input single precision operations. Horizontal Collison Correlation Attack (HCCA) [15] takes advantages of both Big Mac attack and HCA by exploiting collisions originated by the same operand inputted in two LIMs. Hanley *et al.* [16] improved HCCA by detecting collisions between input and output operand of two LIMs. These attacks, except HCA and the attack of Bauer *et al.* [12], can be categorized as collision-based single trace attacks.

On the other hand, practical results of such collision-based single trace attacks are presented later whereas only simulated results of are shown in the original papers besides only the work of Hanley *et al.* [16] exhibited experimental results targeting 192-bit implementations of scalar multiplication on a 32-bit microcontroller and a FPGA. Thereafter, practical results of an improved HCCA exploiting collisions of multiple LIMs in scalar multiplication targeting a 384-bit implementation on a 64-bit architecture is published by Danger *et al.* [17]. Practical experimental results of HCCA and ROSETTA on specific elliptic curves are presented in the works of Das *et al.* [18] and Cho *et al.* [19] targeting 192-bit and 256-bit implementations, respectively.

Countermeasures of advanced single trace attacks are also proposed mainly for securing LIM operations. Clavier *et al.* [11] firstly proposed the method of randomizing two loops for single precision multiplications in LIM operations to exterminate collision characteristics caused by identical manipulation of the same input. Bauer *et al.* [12] enhanced the latter countermeasure and proposed the global shuffling LIM which utilizes incorporated single random permutations for the two loop randomization of single precision multiplications and separate random permutations for carry processing operations. Furthermore, in more recent work [15], this countermeasure is referenced as a possible countermeasure against HCCA. However, its practical effectiveness is not explored in the literature.

Our contribution is threefold. We present the first practical results of a combined single trace attack, which is a combination of a simple power analysis for revealing permutation vectors and collision-based single trace attack with rearranging subtraces, on software implementations of global shuffling LIM, which is known to be secure against advanced collision-based single trace attacks, operated on an ARM Cortex-M4 based STM32F405 microcontroller [20] targeting 128, 192, and 256-bit ECC primitives. We analyze the vulnerability of the algorithm's carry propagation process exploitable by SPA despite theoretically it is intended to give an adversary $(2l-1)!$ complexity of guessing random permutations where l is the word length of input operands of LIM and demonstrate practical result of recovering whole permutation vectors with a single trace by SPA accompanied with straightforward substitution of power consumption samples. Then we successfully mount collision-based single trace attacks with power consumption subtraces of unit carry propagation operations after rearrangement of processing order on the basis of revealed permutation vectors. Secondly, we provide three attack scenarios on which such vulnerability of the global shuffling algorithm is exploited to successfully recover the secret scalar. Since our proposed attack targets the carry propagation process in which only the results of single precision multiplications are manipulated, exploitable collisions in operands of LIM is limited in case of both operands are the same. Nevertheless, still the vulnerability of global shuffling LIM can lead to the recovery of the secret scalar for three cases where particular unified point additions are deployed. Finally, we propose a novel countermeasure against our proposed attack. The vulnerability of global shuffling LIM is caused by the if-statement for selection of proper entries from the permutation vector. Our proposed countermeasure eliminates such selection with a permutation vector rearrangement method utilizing simple addition and modulus operation and achieves regularity in power trace patterns consequently providing security for resistance against SPA. Practical result demonstrating the regularity of power consumption trace acquired from the implementation of our countermeasure is also presented.

This paper is organized as follows. In Section II, we introduce SPA-resistant scalar multiplications and advanced collision-based single trace attacks defeating such countermeasures. And we introduce global shuffling LIM proposed in [12] which is known to defeat advanced collision-based single trace attacks when the algorithm is deployed by the latter scalar multiplications. And in Section III, we analyze the vulnerability of the global shuffling LIM and present attack scenarios for three elliptic curve cases on which recovery of the secret scalar is possible when unified point additions in projective coordinates are deployed. Then we present practical results of our combined single trace attacks which consist revealing random permutation vectors by SPA and detecting input collisions of global shuffling LIMs by collision-based single trace attacks followed by rearrangement of subtraces based on revealed permutation vectors.

In Section IV, we propose a countermeasure eliminating SPA-leakage and also present practical result of such implementation. Finally, we conclude this paper in Section V.

II. PRELIMINARIES

Since ECC cryptographic protocols such as ECDSA or ECDH use an ephemeral secret, differential power analysis [4] which requires several or many SCA-leakage traces, for example, power consumption traces, manipulating the same secret is not possible. As result, attacks exploiting a single power trace are researched importantly. SPA or Timing attack is simple and powerful attack which can reveal the secret from a single power trace hence the SPA-resistant property became essential for secure implementation for embedded devices. Thereafter more advanced single trace attacks defeating SPA-resistant implementations are also proposed. In this section, we describe basic idea of SPA on some ECC protocols and the side-channel atomic scalar multiplication with unified point addition which can defeat SPA. And illustrate some advanced collision-based single trace attacks which can be performed in the presence of additional countermeasures defeating DPA-like (advanced) single trace attacks, such as HCA, and the global shuffling LIM countermeasure.

A. SIMPLE POWER ANALYSIS ON ELLIPTIC CURVE CRYPTOGRAPHY AND UNIFIED POINT ADDITION

Most sensitive operation in ECDSA or ECDH protocols is scalar multiplication since a secret scalar k is directly manipulated during calculation of kP where P is a point on an elliptic curve. Simplest algorithm for scalar multiplication is left-to-right binary or double-and-add [5] method. Assume an n-bit scalar k is represented as $k = (k_{n-1}, \ldots, k_0)_2$ and a register R for the result of the algorithm is set as the point at infinity of an elliptic curve, the algorithm scans the most significant bit to the least significant bit of k and if scanned bit is zero it performs a point doubling on R or a point addition $R + P$ followed by a point doubling on Rif the bit is one.

Since double-and-add algorithm operates different sequences depending on the secret bit, an adversary can reveal the key bit if it can determine the difference by inspecting side-channel leakage of an implementation such as timing or patterns of power trace [3]. Hence so-called regular algorithms, that is, double-and-add-always [6], Montgomery ladder [7], [8] and, side-channel atomic scalar multiplication [9] are proposed. Double-and-add-always and Montgomery ladder algorithms operate identical sequence regardless of the key bit value, that is, one doubling and one addition per the key bit. On the other hand, operation sequence of side-channel atomic scalar multiplication, as described in Alg. 1, is similar to double-and-add algorithm. However, doubling operations are replaced by point additions with the same point within R_0 and the process of scanning the key bit is replaced by Step 5 which is regular regardless of the key bit value whereas double-and-add has an if-statement for the process.

To properly implement side-channel atomic scalar multiplication, unified point addition, in which both addition and doubling operation are calculated on the same formula, is deployed for the indistinguishability of two operations. Since Brier and Joye firstly proposed on Weirstrass curve in [21], unified point additions on various curves and point representations are studied [21]–[23].

Algorithm 1 Side-Channel Atomic Scalar Multiplication

Input: $P, k = (k_{n-1}, \ldots, k_0)_2$
Output: kP
1: $R_0 \leftarrow O$; $R_1 \leftarrow P$; $i \leftarrow n-1$
2: $s \leftarrow 0$
3: **while** $(i \geq 0)$ **do**
4: $\quad R_0 \leftarrow R_0 + R_s$
5: $\quad s \leftarrow s \oplus k_i$; $i \leftarrow i - \neg s$
6: **end while**
7: Return R_0

Algorithm 2 Long Integer Multiplication [14]

Input: $x = (x_{l-1}, \ldots, x_0)_{2^w}, y = (y_{l-1}, \ldots, y_0)_{2^w}$
Output: $LIM(x, y) = x \times y$
1: **for** $i = 0$ to $2l-1$ **do**
2: $\quad r_i = 0$
3: **end for**
4: **for** $a = 0$ to $l-1$ **do**
5: $\quad c \leftarrow 0$
6: $\quad$ **for** $b = 0$ to $l-1$ **do**
7: $\quad\quad (uv)_{2^w} \leftarrow (r_{a+b} + x_a \times y_b) + c$
8: $\quad\quad r_{a+b} \leftarrow v$ and $c \leftarrow u$
9: $\quad$ **end for**
10: $\quad r_{a+l} \leftarrow c$
11: **end for**
12: Return r

B. ADVANCED COLLISION-BASED SINGLE TRACE ATTACKS AND COUNTERMEASURES

However, advanced collision-based single trace attacks [13], [15], [16] can be performed despite such regular scalar multiplications are implemented. These attacks are based on determining collision characteristics originated from identical unit operations in field multiplications consisting in regular scalar multiplications and key-dependent existence of such characteristics. We describe collision characteristics of advanced collision-based attacks in the following and we assume that a field multiplication is composed of a long integer multiplication, as shown in Alg. 2, and a following modular reduction.

ROSETTA [13] targets detecting inner collisions of field squaring operations. Assume an adversary can extract side-channel traces of single precision multiplications, that is, $x_a \times y_b$ in Step 7 of Alg. 2, if a field squaring is performed a half of (l^2-l)traces have collision correlation since $x_a \times x_b = x_b \times x_a$ for all $a \neq b$. In the opposite case, for a field multiplication, since $x_a \times y_b \neq x_b \times y_a$ for all $a \neq b$ the collision characteristic does not exist.

HCCA [15] determines collisions in at least one input operand of two (or more) field multiplications. Consider two LIMs consisting in two field multiplications, $LIM(x, y)$ and $LIM(x, z)$, if an adversary can collect two groups of side-channel traces of single precision multiplications, respectively for $x_a \times y_b$ and $x_a \times z_b$, and if correlation of two groups are higher than other cases where none of the inputs have the same value, the collision characteristic is exploitable.

The attack of Hanley *et Al.* [16] exploits collisions of input and output of field multiplications. Consider two field multiplications denoted by $MUL(x, y) = z$ and $MUL(z, w)$, if there is higher correlation in this case compared to non-collision cases, the attack is possible. Contrary to two former attacks, this attack requires considering the different placement of side-channel leakage in time domain, thus they deploy the cross-correlation technique to find points of interest where input-output collision correlation exists.

Countermeasures of those attacks are already proposed. Clavier *et al.* [11] first proposed the idea of randomizing the two loops of Alg. 2, that is, a and b, separately to constrain collision correlations caused by sequential (or predetermined) manipulation of input operands in LIMs. After that, Bauer *et al.* [12] exhibited the attack weakening the effect of the separate randomization and proposed global shuffling LIM, described in Alg. 3. Also in [15], this countermeasure is considered to be an effective countermeasure against HCCA and to the best of our knowledge novel single trace attacks can defeat this countermeasure are not proposed in the literature. We further analyze the global shuffling LIM in the next section.

III. COMBINED SINGLE TRACE ATTACK ON GLOBAL SHUFFLING LONG INTEGER MULTIPLICATION

In this section, we present a combined single trace attack which can be applied on side-channel atomic scalar multiplications with unified point addition deploying global shuffling LIM which is known to be secure against advanced collision-based single trace attacks. First, we analyze a vulnerability of the global shuffling LIM proposed in [12] and present attack scenarios which can lead to the recovery of the secret scalar for three cases, that is, unified addition formulas with projective coordinates for short Weirstrass, Jacobi quartic, and Jacobi intersection curves, where such vulnerability can be exploited by an adversary. Next, we demonstrate by practical experiments that the vulnerability really exists showing distinct patterns with respect to the condition of an if-statement in power consumption traces. Consequently, as revealing whole permutation vectors by simple power analysis with a straightforward substitution of subtraces are possible, then collision-based single trace attacks can be mounted on originally shuffled carry propagation operations after rearranging subtraces based on revealed vectors.

A. EXPLOITABLE VULNERABILITY ANALYSIS OF GLOBAL SHUFFLING LONG INTEGER MULTIPLICATION

The global shuffling long integer multiplication [12] is an enhanced version of two loops randomization LIM proposed by Clavier *et al.* [11]. The core idea of both countermeasures is to impose an $(l!)^2$ complexity on an adversary to perform advanced single trace attacks exploiting single precision multiplications in LIM by randomizing the execution order of them. To this end, the two loops randomization LIM deploys two random permutations in $[0, l-1]$ to randomize index variable a and b in Step 7 in Alg. 2, separately. However, Bauer *et al.* [12] demonstrate the technique of nullifying the effect of randomization for one random permutation by averaging l subtraces. Such attack is possible since an adversary knows the fact that when executing the loop of Step 6-9, there are l single precision operations while the index variable a fixed with some unknown value. Consequently, Bauer *et al.* [12] proposed the method of randomizing both indices simultaneously for operations of single precision multiplications $x_a \times y_b$.

On the other hand, by the result of changing the construction of the loop for operations of single precision multiplications $x_a \times y_b$, carry propagation process needs to be changed properly. In the global shuffling LIM algorithm, as described in Alg. 3, such carry propagations are processed independently as shown in Step 12-21 and to prevent advanced single trace attacks exploiting the correlations caused by the collisions of input operands of LIMs, another random permutation vector in $[1, 2l-1]$ is deployed. Thus theoretically, an adversary attempting advance single trace attacks on carry propagation process must guess the permutation out of $(2l-1)!$ possibilities. Nevertheless, such attacks are still possible because of the existence of if-statement in Step 15, consequently the random permutations can be revealed by SPA.

First, consider Step 12-21 of Alg. 3, when $i = 2l-1$ the if-statement suffices only once where the entry of the permutation vector P is $2l-1$. Hence in the power consumption trace, there will be $2l-2$ identical patterns executing Step 13-15 where the condition of if-statement is *false* and one different and longer pattern caused by additional operations of Step 16-19. If an adversary can identify two different patterns, it can recover the position of entry $2l-1$ in P where there exist only $2l-1$ positions in the vector. Next, for $i = 2l-2$ the if-statement suffices twice, that is, for entries $2l-1$ and $2l-2$, in this case there will be two different and longer patterns are expected to be observed in the power consumption trace. Since the adversary already knows the position of the entry $2l-1$, it can recover the position of entry $2l-2$ simply considering the remained position corresponding to another pattern caused by *true* condition of the if-statement is for the latter entry. Similarly, entire positions of entries which are randomly distributed in P can be recovered step-by-step.

Now assuming that the adversary knows every permutation vectors corresponding to each global shuffling LIM's carry propagation process, it can mount collision-based single trace attacks independently exploiting carry propagation process. Note that the previous attacks mainly exploit the single precision multiplications in LIM. Since our proposed

Algorithm 3 Global Shuffling Long Integer Multiplication [12]

Input: $x = (x_{l-1}, \ldots, x_0)_{2^w}$, $y = (y_{l-1}, \ldots, y_0)_{2^w}$
Output: $LIM(x, y) = x \times y$
1: $\gamma = (\alpha, \beta)$ ←a permutation vector where α, β are random permutations in $[0, l-1]$
2: P ←a random permutation in $[1, 2l-1]$
3: **for** $i = 0$ to $2l-1$ **do**
4: $\quad r_i = c_i = 0$
5: **end for**
6: **for** $h = 0$ to $l^2 - 1$ **do**
7: $\quad (a, b) \leftarrow \gamma_h$
8: $\quad (uv)_{2^w} \leftarrow r_{a+b} + x_a \times y_b$
9: $\quad r_{a+b} \leftarrow v$
10: $\quad c_{a+b+1} \leftarrow c_{a+b+1} + u$
11: **end for**
12: **for** $i = 1$ to $2l-1$ **do**
13: $\quad$ **for** $j = 1$ to $2l-1$ **do**
14: $\quad\quad s \leftarrow P_j$
15: $\quad\quad$ **if** $(s \geq i)$ **then**
16: $\quad\quad\quad (uv)_{2^w} \leftarrow r_s + c_s$
17: $\quad\quad\quad r_s \leftarrow v$
18: $\quad\quad\quad c_{s+1} \leftarrow c_{s+1} + u$
19: $\quad\quad\quad c_s \leftarrow 0$
20: $\quad$ **end for**
21: **end for**
22: Return r

attack targets the carry propagation process in which only the results of single precision multiplications are manipulated, exploitable collisions in operands of LIM are limited in case of both operands are the same. Nevertheless, still this vulnerability of global shuffling LIM can lead to the recovery of the secret scalar for three cases where particular unified point additions are deployed, as described in the next section.

B. APPLICABILITY OF PROPOSED ATTACK ON ELLIPTIC CURVES

Here, we analyze the possibility of attacking side-channel atomic scalar multiplication with unified point addition deploying global shuffling LIM. Assuming an adversary can detect collisions of two operands in two field multiplications, that is, two global shuffling LIMs in this context, the secret scalar k of Alg. 1 can be recovered if such collisions exist only for doubling (or addition) despite the same sequence of operations are executed by the unified point addition. In the following, we demonstrate such possible attack scenarios by analyzing unified point addition formulae for several elliptic curves.

1) SHORT WEIERSTRASS CURVES

A Short Weierstrass elliptic curve E_K defined over a field K satisfies following equation

$$y^2 = x^3 + ax + b \tag{1}$$

Algorithm 4 Unified Point Addition for Short Weierstrass Curves [22]

Input: $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$
Output: $P_1 + P_2 = (X_3, Y_3, Z_3)$
1: $U_1 = X_1 \times Z_2$
2: $U_2 = X_2 \times Z_1$
3: $S_1 = Y_1 \times Z_2$
4: $S_2 = Y_2 \times Z_1$
5: $ZZ = Z_1 \times Z_2$
6: $T = U_1 + U_2$
7: $TT = T^2$
8: $M = S_1 + S_2$
9: $R = TT - U_1 \times U_2 + a \times ZZ^2$
10: $F = ZZ \times M$
11: $L = M \times F$
12: $LL = L^2$
13: $G = (T + L)^2 - TT - LL$
14: $W = 2 \times R^2 - G$
15: $X_3 = 2 \times F \times W$
16: $Y_3 = R \times (G - 2 \times W) - 2 \times LL$
17: $Z_3 = 4 \times F \times F^2$
18: Return (X_3, Y_3, Z_3)

where $(x, y) \in K^2$ in affine coordinates and $a, b \in K$. Given two points on the curve $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ with the projective coordinate representation where $x = X/Z$ and $y = Y/Z$, the unified addition formulas for $P_1 + P_2 = (X_3, Y_3, Z_3)$ are presented as following algorithm [22].

If $P_1 = P_2$, two input operands of field multiplications corresponding to (Step 1, Step 2), and (Step 3, Step 4) are identical, respectively. For the case when $P_1 \neq P_2$, the latter collisions of input operands do not occur.

2) JACOBI QUARTIC CURVES

A Jacobi quartic elliptic curve E_K defined over a field K satisfies following equation

$$y^2 = x^4 + 2ax^2 + 1 \tag{2}$$

where $(x, y) \in K^2$ in affine coordinates and $a \in K$. Given two points on the curve $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$with the projective coordinate representation where $x = X/Z$ and $y = Y/Z$, the unified addition formulas for $P_1 + P_2 = (X_3, Y_3, Z_3)$ are presented as following algorithm [22].

If $P_1 = P_2$, two input operands of field multiplications corresponding to (Step 1, Step 6), (Step 2, Step 7), and (Step 4, Step 9) are identical, respectively. For the case when $P_1 \neq P_2$, the latter collisions of input operands do not occur.

3) JACOBI INTERSECTION CURVES

A Jacobi intersection elliptic curve E_K defined over a field K satisfies following equation

$$\begin{cases} s^2 + c^2 = 1 \\ as^2 + d^2 = 1 \end{cases} \tag{3}$$

Algorithm 5 Unified Point Addition for Jacobi Quartic Curves [22]

Input: $P_1 = (X_1, Y_1, Z_1), P_2 = (X_2, Y_2, Z_2)$
Output: $P_1 + P_2 = (X_3, Y_3, Z_3)$
1: $A_2 = X_2^2$
2: $C_2 = Z_2^2$
3: $D_2 = A_2 + C_2$
4: $B_2 = (X_2 + Z_2)^2 - D_2$
5: $E_2 = B_2 + Y_2$
6: $A_1 = X_1^2$
7: $C_1 = Z_1^2$
8: $D_1 = A_1 + C_1$
9: $B_1 = (X_1 + Z_1)^2 - D_1$
10: $E_1 = B_1 + Y_1$
11: $A_1A_2 = A_1 \times A_2$
12: $B_1B_2 = B_1 \times B_2$
13: $C_1C_2 = C_1 \times C_2$
14: $Y_1Y_2 = Y_1 \times Y_2$
15: $F = C_1C_2 + A_1A_2$
16: $G = 2 \times B_1B_2$
17: $X_3 = E_1 \times E_2 - B_1B_2 - Y_1Y_2$
18: $Y_3 = F \times (4 \times Y_1Y_2 + a \times G) + (D_1 \times D_2 - F) \times G$
19: $Z_3 = 2 \times (C_1C_2 - A_1A_2)$
20: Return (X_3, Y_3, Z_3)

Algorithm 6 Unified Point Addition for Jacobi Intersection Curves [23]

Input: $P_1 = (X_1, Y_1, Z_1), P_2 = (X_2, Y_2, Z_2)$
Output: $P_1 + P_2 = (X_3, Y_3, Z_3)$
1: $SC_1 = S_1 \times C_1$
2: $DZ_1 = D_1 \times Z_1$
3: $SC_2 = S_2 \times C_2$
4: $DZ_2 = D_2 \times Z_2$
5: $E = S_1 \times D_2$
6: $F = C_1 \times Z_2$
7: $G = D_1 \times S_2$
8: $H = Z_1 \times C_2$
9: $J = SC_1 \times DZ_2$
10: $K = DZ_1 \times SC_2$
11: $S_3 = (H + F) \times (E + G) - J - K$
12: $C_3 = (H + E) \times (F - G) - J + K$
13: $D_3 = (DZ_1 - a \times SC_1) \times (SC_2 + DZ_2) + a \times J - K$
14: $Z_3 = (H + G)^2 - 2 \times K$
15: Return (X_3, Y_3, Z_3)

where $(s, c, d) \in K^3$ in affine coordinates and $a \in K$. Given two points on the curve $P_1 = (S_1, C_1, D_1, Z_1)$ and $P_2 = (S_2, C_2, D_2, Z_2)$with the projective coordinate representation where $s = S/Z, c = C/Z$, and $d = D/Z$, the unified addition formulas for $P_1 + P_2 = (S_3, C_3, D_3, Z_3)$ are presented as following algorithm [23].

If $P_1 = P_2$, two input operands of field multiplications corresponding to (Step 1, Step 3), (Step 2, Step 4), (Step 5, Step 7), (Step 6, Step 8), and (Step 9, Step 10) are identical, respectively. For the case when $P_1 \neq P_2$, the latter collisions of input operands do not occur.

C. EXPERIMENTAL RESULT

In this section, we evaluate the security of global shuffling LIM against advanced single trace attacks. As analyzed in the previous section, unified point additions for some elliptic curves have collisions in two operands of field multiplications which can be exploitable for recovery of the secret scalar from side-channel atomic scalar multiplications. Such collisions can be detected by an adversary if field multiplications are based on school-book LIMs as proposed in previous works [13], [15], [16]. However, the global shuffling LIM can prevent such detection if it is secure as intended. In the following, we demonstrate that detection of collisions in two operands of global shuffling LIMs is possible by a single trace collision-based attack targeting the carry propagation operations followed by recovery of the random permutation vectors with SPA.

1) EXPERIMENTAL SETUP

We conduct practical experiments of our attack for three power consumption traces consisting of $2n$ ($n = 128, 192, 256$) global shuffling LIMs, where n represents the number of bits of targeted ECC primitive which is operated on ARM Cortex-M4 based STM32F405 microcontroller [20] embedded on ChipWhisperer [24] CW308T-STM32F [25] target board. Note that in real attack situation there will be more than $2n$ LIMs for one scalar multiplication because unified point additions consist of more than two LIMs. For the situation, an adversary should extract power consumption traces of target LIMs. Such extraction can be achieved by finding a LIM trace, we refer it as reference trace, from visual inspection and then locating the rest LIM traces by calculating correlation coefficient of the reference with the whole scalar multiplication point-by-point. For the simplicity of the explanation and due to limited memory of our oscilloscope, we conduct our experiment assuming an adversary exploits one LIM from each unified addition. Alg. 3 is implemented in ARM assembly language. In detail, each power consumption trace has n pair of LIMs where half of them have input collisions of two operands and the rest have no collision, that is, if we represent *(A, B, C, D)* as operands of a pair of LIM operations, $A = C, B = D$ for the collision case and $A \neq C, B \neq D$ for the opposite case where all the inputs are different random values for each pair of LIMs. Of course, permutation vectors γ and P for individual LIM operations are different. Two LIMs consisting a pair are located contiguously. Note that as we target carry propagation process of the global shuffling LIM, we focus on power consumption samples corresponding to $l(2l - 1)$ unit carry propagation operations. Since we conduct experiments for 128, 192, and 256-bit ECC primitives based on 32-bit words, there exist 28, 66, and 90 such unit operations within each LIM, respectively. For all acquisitions of power consumption traces we amplify the signal with CW501 differential probe [26]

TABLE 1. Settings for power trace acquisition.

Bit Length of Target	Operating Frequency	Sampling Rate
128	25MHz	1.25GS/s
192	20MHz	500MS/s
256	50MHz	500MS/s

and capture them with LeCroy HDO6104A oscilloscope [27]. Operating clock frequency of the target device and sampling rate of oscilloscope for each acquisition are listed in Table 1. For simplicity, we illustrate our attack experiment targeting 128-bit implementation hereafter. The attack principle and actual results for the rest of implementations are similar.

2) REVEALING RANDOMIZED VECTORS BY SIMPLE POWER ANALYSIS

To reveal the permutation vector P for each LIM, we firstly identify and extract the power consumption samples of Step 12-21 in Alg. 3 denoted by C_k where $k = \{1, \ldots, 2n\}$ indicates the index of each LIM operation. This process is done by finding a single reference trace of C_k and calculating correlation coefficient point by point with the whole power consumption trace. Since the vector P varies for each LIM, actual patterns of every C_k are not identical. Nevertheless, whereas the value of correlation coefficient peaks ranges from 0.36 to 0.88, all the peaks are distinguishable to sufficiently determine the starting point in time for each C_k. Hence we cut the power consumption samples with the same length of the reference from the starting points and acquire every C_k. Fig. 1 presents all C_k where $k = \{1, \ldots, 256\}$. As shown in Fig. 1, we can determine intervals for each loop of Step 13-20 in Alg. 3 whereas i increases due to the power consumption peaks which are common for all C_k appeared after each loop interval (they are located outside the dotted lines in Fig. 1). Considering the interval where $i = 7$, we can confirm that seven peaks exist since if-statement of Step 15 in Alg. 3 only suffice once, that is, when $s = 7$, whereas s varies from 1 to 7 in random sequence. Hence, by observing the position of each peak among the seven peaks, we can determine the location of entry 7 in each permutation vector P.

However, for $i = \{2, \ldots, 6\}$ such simple determination is not possible because power consumption peaks in each interval do not present in seven positions. This phenomenon is caused by the fact that additional operations of Step 16-19 in Alg. 3 are performed only when the if-statement is *true* and such condition occurs more than two times for $i = \{2, \ldots, 6\}$ case. To solve this problem, we can reconsider the interval of $i = 7$ in Fig. 1 and analyze different patterns corresponding to the condition of the if-statement of Alg. 3 as represented in Fig. 2. Obviously, if the condition is *true*, longer and distinct pattern of power consumption is observed and this pattern can be used as a reference pattern to locate similar patterns in every C_k by calculating correlation coefficient point by point (Note that these patterns

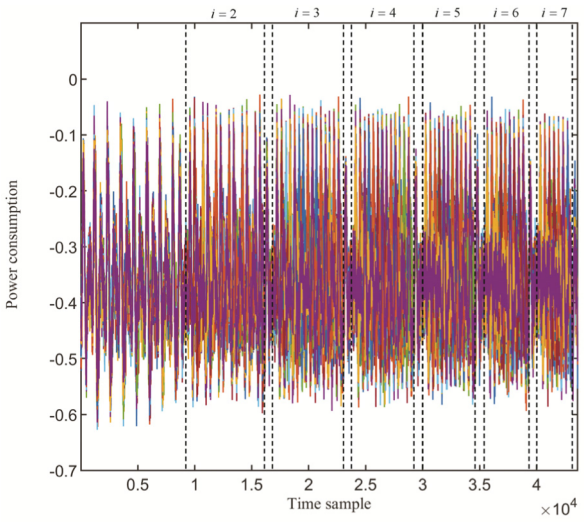


FIGURE 1. Power consumption traces of carry propagation process of global shuffling LIM.

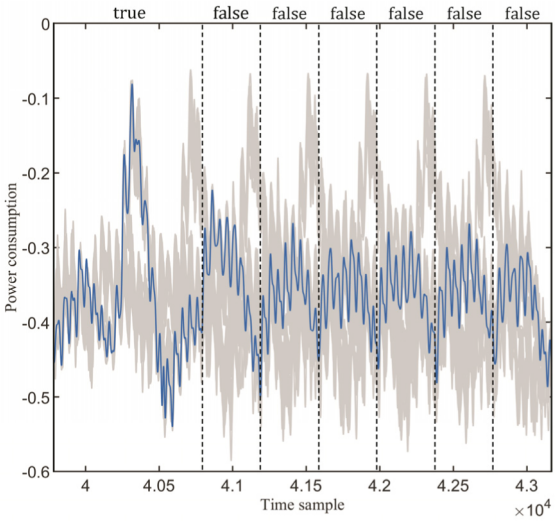


FIGURE 2. Identification of patterns corresponding to the condition of if-statement of Alg. 3 where i = 7. This is an example of the case where entry 7 is located in the first index of permutation vector P (the rest power consumption traces are colored gray to show peak positions).

can be efficiently distinguished by checking whether power consumption exceeds a certain threshold, however we choose this method to locate such patterns accurately and then easily substitute them from the points where correlation peaks exist in the next step). After this process, we can substitute all these particular patterns, which are similar to the reference pattern, with an arbitrary pattern with the same length of a *false* case pattern in Fig. 2 but with different shape. For example, we choose samples located in the center of *true* case pattern in Fig. 2 where relatively higher peak of power consumption is observed. The result of the substitution for all C_k is presented in Fig. 3. As all peaks caused by *true* condition of the if-statement are located in seven positions for each loop of $i = \{2, \ldots, 6\}$, we can now reveal entire entries of the permutation vector P for each C_k by SPA.

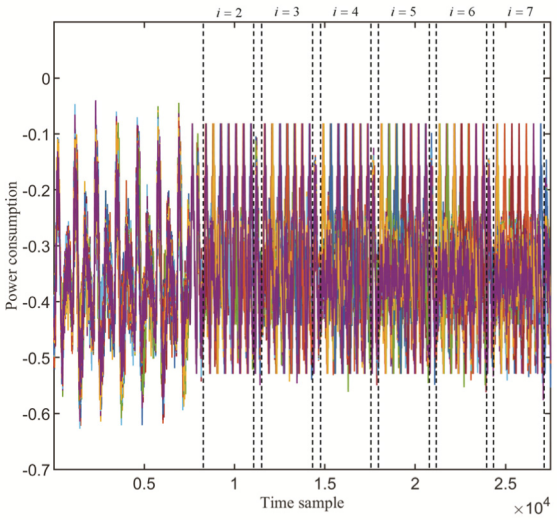


FIGURE 3. Power consumption traces of carry propagation process of global shuffling LIM after substitution of samples to locate seven positions of particular operation for recovering permutation vector P.

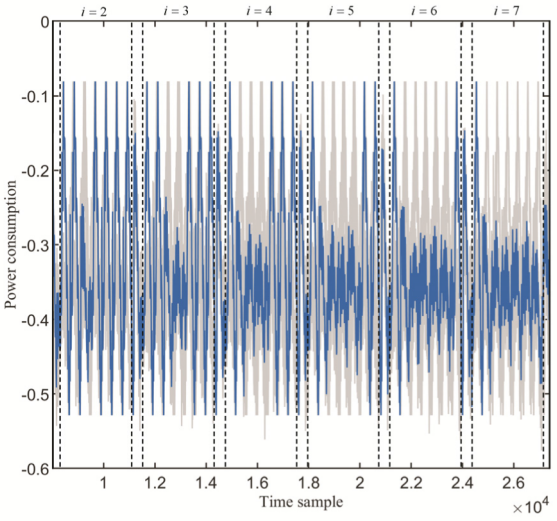


FIGURE 4. An example of recovering permutation vector P with a single power trace.

We illustrate here an example of recovering the permutation vector P for a particular C_k where $k = 255$ in our experiment. Similarly to Fig. 2, we present $\tilde{C}_{255}$ in Fig. 4, where $\tilde{C}_k$ represents a power consumption trace resulted from the previous substitution on each C_k, and the rest $\tilde{C}_k$ where $k = \{1, \ldots, 254, 256\}$ in gray color to reveal the permutation vector P by visual inspection. The recovery of entries of the permutation vector P is performed one by one from the interval of $i = 7$ to $i = 2$. As described previously, a power consumption peak in the interval of $i = 7$ indicates the position of entry 7 in the permutation vector. Hence we can determine that the first entry of P is 7. Next we can observe that for the interval of $i = 6$, there are two peaks in the first position and the seventh position. As we already know that

TABLE 2. Settings of window parameter for compression.

Bit Length of Target	Points per Clock	Window Size
128	50	25
192	25	25
256	10	10

the peak in the first position is caused by the case of $s = 7$, we can infer that the peak in the seventh position is caused by the case of $s = 6$. As a result, the entry 7 and 6 are located in the first and the seventh position in P, respectively. The rest of entries can be recovered similarly and for $i = 1$ case, the entry 1 is assigned to the last remained position. Finally, we can recover the permutation vector as $P = \{7, 3, 1, 2, 4, 5, 6\}$ for $\tilde{C}_{255}$. All permutation vectors for the rest $\tilde{C}_k$ can be recovered by the same manner.

3) RECONSTRUCTING AND POST-PROCESSING POWER CONSUMPTION TRACES

Assuming all permutation vectors are known, we describe preprocesses for performing collision correlation analysis exploiting the recovered vectors. To detect collisions caused by the same intermediate data on carry propagation process of two global shuffling LIMs, we reconsider original power consumption traces C_k hereafter. In the previous subsection, all locations of power consumption samples corresponding to the additional operations of Step 16-19, denoted by $T_{k,m}$ where $m = \{1, \ldots, l(2l-1)\}$, due to *true* condition of the if-statement in Alg. 3 are identified in the process of making each $\tilde{C}_k$ by calculating correlation coefficient point by point with a reference samples of *true* case pattern in Fig. 2. We reconstruct the power consumption traces as $C_k = [T_{k,1} \parallel T_{k,2} \parallel \cdots \parallel T_{k,l(2l-1)}]$ for all $k = \{1, \ldots, 256\}$.

Then we applied the integration compression technique [28] to each C_k. The *window size* parameter which represents how many number of samples integrated to one point is selected among factors of *points per clock*, that is, the sampling rate of acquisition of a measuring device divided by the operating clock cycles of a target device. We present selected *window size* parameters which lead to the best success rates of further collision attacks for all three experiments targeting 128/192/256-bit ECC primitives in Table 2.

Finally, we applied the subtraction of mean traces of $T_{k,m}$ in the same manner as represented in [29] to remove the operational dependent leakage and possible noise. Since we target to determine collisions of manipulated data in a pair of global shuffling LIMs, we divide power traces C_k where $k = \{1, \ldots, 256\}$ into two groups. Then for the two groups separately, mean traces of each group are subtracted from each $T_{k,m}$ accordingly.

4) REORDERING POWER CONSUMPTION TRACES

Now to exploit collision characteristic of two global shuffling LIMs caused by input operands similarly to the previous

Algorithm 7 Reordering Subtraces

Input: $C = [T_1 \parallel T_2 \parallel \cdots \parallel T_{l(2l-1)}]$, P
Output: Reordered $C = [\tilde{T}_1 \parallel \tilde{T}_2 \parallel \cdots \parallel \tilde{T}_{l(2l-1)}]$ where position index m of subtrace T_m is rearranged corresponding to entry values of P
1: $offset = 0$
2: **for** $i = 1$ to $2l - 1$ **do**
3: **for** $j = 1$ to $|P|$ **do**
4: $\tilde{T}_{offset+P[j]} \leftarrow T_{offset+j}$
5: **end for**
6: $offset \leftarrow offset + |P|$
7: $P \leftarrow P \backslash \{i\}$ // Omit entry i from vector P
8: **end for**
9: Return $\tilde{C} = [\tilde{T}_1 \parallel \tilde{T}_2 \parallel \cdots \parallel \tilde{T}_{l(2l-1)}]$

attacks performed on non-shuffling LIMs, we reorder the placement of subtraces $T_{k,m}$ in each C_k such that all reordered traces $\tilde{C}_k$ have practically the same order of processing carries to remove the effect of the global shuffling countermeasure vanishing the correlation between two LIMs' power traces by shuffling the manipulating order of each unit carry processing operation, that is, Step 16-19 in Alg. 3, every time the global shuffling LIM is operated. More precisely, as analyzed in the previous subsection, manipulated indices for each iteration of i in Alg. 3 can be denoted by $m_i = \{x | i \le x \le 2l - 1\}$ where the positions of entries for each m_i are shuffled and Alg. 7 rearranges these shuffled positions in ascending order.

5) FINDING POINTS OF INTEREST

To perform further collision-based single trace attack, we find points of interest by calculating correlation coefficient vector with the length of $T_{k,m}$ denoted by l_T. As we have n pairs of LIM traces where each LIM trace C_k is composed of $l(2l-1)T_{k,m}$ traces, we vertically stack all $T_{k,m}$ separately for each group of LIM traces and calculate the correlation coefficient of these two matrices as described in Alg. 8. By performing this process, correlation coefficient peaks caused by the collision characteristic from the same manipulated data for some pair of LIM traces can be identified despite the noise is added in the resulting correlation coefficient vector if there is no collision of input operands for other pair of LIM traces for the same point in time. After choosing samples as points of interest where the correlation coefficient value is equal or greater than some threshold as shown in Fig. 5, we can perform further collision-based single trace attack.

6) RECOVERING SECRET BY COLLISION-BASED SINGLE TRACE ATTACK

We can now determine whether each pair of LIM traces have input collisions or not. As we analyzed in Section III-B, if an adversary can detect whether there is collision of two input operands of global shuffling LIMs in some unified point additions deployed by a side-channel atomic scalar multiplication or not it can recover secret scalar as described in Alg. 10

Algorithm 8 Calculating Correlation Coefficient Trace to Find POIs

Input: $C_k = [T_{k,1} \parallel T_{k,2} \parallel \cdots \parallel T_{k,l(2l-1)}]$
Output: $(1 \times l_T)$ correlation coefficient vector CI
1: **for** $j = 0$ to $n - 1$ **do**
2: **for** $m = 1$ to $l(2l - 1)$ **do**
3: $T_{M1} = [T_{M1}; T_{2\times j,m}]$
4: $T_{M2} = [T_{M2}; T_{2\times j+1,m}]$
5: **end for**
6: **end for**
8: $CI = CorrT(T_{M1}, T_{M2})$
9: Return CI

Algorithm 9 *CorrT* Function

Input: Two matrices $A = (a_{i,j})$ and $B = (b_{i,j})$ of the same size $(1 \le i \le M, 1 \le j \le N)$
Output: $(\rho_1, \rho_2, \ldots, \rho_N)$
1: **for** $i = 1$ to N **do**
2: $X = [a_{1,i}, a_{2,i}, \ldots, a_{M,i}]^T$
3: $Y = [b_{1,i}, b_{2,i}, \ldots, b_{M,i}]^T$
4: $\rho_i = corr(X, Y)$
5: **end for**
6: Return $(\rho_1, \rho_2, \ldots, \rho_N)$

Algorithm 10 Recovering the Secret Scalar With Collision Power Analysis

Input: $C_k = [T_{k,1} \parallel T_{k,2} \parallel \cdots \parallel T_{k,l(2l-1)}]$, CI
Output: Recovered secret scalar $d = (d_{n-1}, \ldots, d_0)_2$
1: Prepare a score vector $\Delta = [\delta_{n-1}, \ldots, \delta_1]$
2: // Select POIs having values larger or equal than arbitrary *threshold* then store indices in a vector $I_P = [i_1, \ldots, i_{n_P}]$
3: **for** $i = 1$ to l_T **do**
4: **if** $(CI(i) \ge threshold)$ **then**
5: $I_P = [I_p, i]$
6: **end for**
7: Reconstruct $\hat{C}_k$ with only POIs extracted $\hat{T}_{k,m} = T_{k,m}(I_P)$
8: **for** $j = 0$ to $n - 1$ **do**
9: $\hat{C}_{2\times j} = [\hat{T}_{2\times j,1} \parallel \hat{T}_{2\times j,2} \parallel \cdots \parallel \hat{T}_{2\times j,l(2l-1)}]$
10: $\hat{C}_{2\times j+1} = [\hat{T}_{2\times j+1,1} \parallel \hat{T}_{2\times j+1,2} \parallel \cdots \parallel \hat{T}_{2\times j+1,l(2l-1)}]$
11: $\delta_j = corr(\hat{C}_{2\times j}, \hat{C}_{2\times j+1})$
12: **end for**
13: **for** $j = 0$ to $n - 1$ **do**
14: **if** $(\delta_j > mean(\Delta))$ **then**
15: $d_j = 0$
16: **else**
17: $d_j = 1$
18: **end for**
19: Return $d = (d_{n-1}, \ldots, d_1)_2$

(this attack can be done by exploiting more than one pair of global shuffling LIMs as long as the collision characteristic of such LIMs exists only in doubling or addition operation of a side-channel atomic scalar multiplication but in our

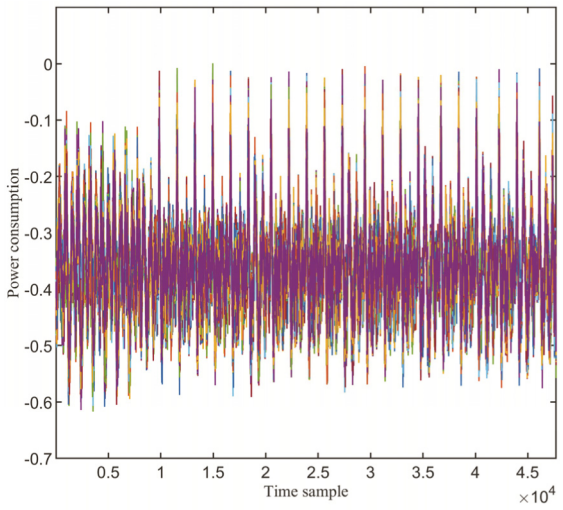


FIGURE 5. Power consumption t races of carry propagation process of proposed global shuffling LIM.

TABLE 3. Success rates of combined single trace attacks.

Bit Length of Target	Number of Bits Revealed	Success Rate
128	123	96.09%
192	186	96.88%
256	251	98.05%

experiment we just consider collisions of a pair of global shuffling LIMs for simplicity). First, we extract POIs selected in the previous subsection in each unit carry propagation subtrace $T_{k,m}$ and reconstruct $\hat{C}_k = [\hat{T}_{k,1}||\hat{T}_{k,2}||\cdots||\hat{T}_{k,l(2l-1)}]$ where the group of extracted samples of POIs are denoted by $\hat{T}_{k,m}$. Then we calculate the correlation coefficient δ_j between two contiguous LIMs $\hat{C}_{2\times j}$ and $\hat{C}_{2\times j+1}$ for all $j = \{0, \ldots, n-1\}$. If the value of δ_j is greater than the mean of all δ_j we consider the bit value of secret scalar d_j is zero else the bit is one. Results of these attacks targeting 128-bit, 192-bit, and 256-bit implementations are presented in Table 3 where each success rate is represented by the number of correctly recovered bits divided by the whole secret bit length n. Despite several bits are recovered incorrectly, we can observe that the success rate increases as n increases as well whereas each success rate of the attack is considerable against the security of a side-channel atomic scalar multiplication deploying specific unified point additions with global shuffling LIMs. This phenomenon can be described by the increased number of unit carry propagations for recovering a bit of secret scalar. Hence the success rates can be improved by exploiting more pair LIMs in one doubling or addition operations in a unified point addition (this can be done by utilizing more than one measuring device as presented in [30]).

Furthermore, for the full recovery of the secret scalar we can consider a method of correcting error bits. If an adversary can determine locations of error bits, the correction could be done by just swapping bit values of corresponding

TABLE 4. Brute-forcing costs for correcting error-bits.

Bit Length of Target	Number of Candidate Locations for Brute-forcing	Bound of Cost
128	25	2^{18}
192	39	2^{23}
256	44	2^{22}

locations. Brute-forcing such locations can be represented by $\sum_{i=1}^{n_e} {}_{n-i}C_i$ where n_e is the number of error bits and assuming the most significant bit of the secret is one. In our experiments, the results are bounded by 2^{29}, 2^{37}, and 2^{35} for attacking 128-bit, 192-bit, and 256-bit implementations, respectively. To decrease these costs, we can select some candidate bit-locations for brute-forcing on which correlation coefficient δ_j values are within an arbitrary distance from the mean of all δ_j. For example, in our experiment targeting 192-bit implementation, the mean value of all δ_j is 0.4915 where the values range from -0.6153 to 0.9614, and by choosing the distance as 0.3035 we have thirty-nine candidate locations including all error-bit-locations. Then we calculate the brute-forcing cost again based on the numbers of candidate locations instead of $(n-1)$ in the combination formula, as results are presented in Table 4.

IV. COUNTERMEASURE

In this section, we propose a novel global shuffling LIM against our attack. As analyzed in Section III-A, the vulnerability of the carry propagation process is caused by the if-statement for the selection of entry s which suffices $s \geq i$ from random permutation vector P whereas i increases from 1 to $2l-1$. Hence, in our proposed countermeasure, as described in Alg. 11, the if-statement is removed. In detail, Step 12-21 in Alg. 3, we refer this as the original process, is substituted by Step 12-32 in Alg. 11. In the following, we describe the core idea for the selection of s without the if-statement considering the value of i in the original process and then present power consumption traces, which have the same patterns for different execution of LIMs, of the practical implementation of our proposed countermeasure.

First, for the case $i = 1$ in Alg. 3, the selection of s is not necessary since all $2l-1$ entries of P are required hence the carry propagation process for this case is represented separately as Step 12-18 in Alg. 11.

Also for the case $i = 2l-1$, the selection is not required since this case means the processing of most significant word of the result register by addition with the last carry. Hence, the process can be simplified as Step 32 in Alg. 11.

Consider the remaining case where $2 \leq i \leq 2l-2$. The number of required entries within a permutation vector P where i increases for the original process is $2l-i$, denoted by q in Step 16 in Alg. 11, since entries with the value smaller than i are not selected. Instead of selecting required entries, our proposed countermeasure deploys a rearrangement method

Algorithm 11 Global Shuffling Long Integer Multiplication With Countermeasure Against Our Attack

Input: $x = (x_{l-1}, \ldots, x_0)_{2^w}, y = (y_{l-1}, \ldots, y_0)_{2^w}$
Output: $LIM(x, y) = x \times y$
1: $\gamma = (\alpha, \beta)$ ←a permutation vector where α, β are random permutations in $[0, l-1]$
2: P ←a random permutation in $[1, 2l-1]$
3: **for** $i = 0$ to $2l-1$ **do**
4: $r_i = c_i = 0$
5: **end for**
6: **for** $h = 0$ to $l^2 - 1$ **do**
7: $(a, b) \leftarrow \gamma_h$
8: $(uv)_{2^w} \leftarrow r_{a+b} + x_a \times y_b$
9: $r_{a+b} \leftarrow v$
10: $c_{a+b+1} \leftarrow c_{a+b+1} + u$
11: **end for**
12: **for** $j = 1$ to $2l-1$ **do**
13: $s \leftarrow P_j$
14: $(uv)_{2^w} \leftarrow r_s + c_s$
15: $r_s \leftarrow v$
16: $c_{s+1} \leftarrow c_{s+1} + u$
17: $c_s \leftarrow 0$
18: **end for**
19: **for** $q = 2l-2$ to 2 **do**
20: $t \leftarrow (q+1) - s$ // s is last used entry of P_j
21: **for** $j = 1$ to q **do**
22: $s \leftarrow (P_j + t) - (q+1)$
23: $s \leftarrow s + N \times (q+1)$ // N is negative flag caused by Step 11
24: $s \leftarrow s + (2l-1) - q$
25: $(uv)_{2^w} \leftarrow r_s + c_s$
26: $r_s \leftarrow v$
27: $c_{s+1} \leftarrow c_{s+1} + u$
28: $c_s \leftarrow 0$
29: $P_j \leftarrow s$
30: **end for**
31: **end for**
32: $r_s \leftarrow r_s + c_s$
33: Return r

which changes the permutation vector to have required entries in the first q positions iteratively, consequently accompanies no selection process.

The core idea can be described as follows. First calculate the difference t between the value of entry in the last position in the original permutation vector and the value $q+1$. Next add t for the all entries then apply mod $(q+1)$, now the entries in the first q positions have the values from *1* to q where still placed in random positions and zero in the last position. Then, since required value of the entries in the original process ranges from i to $2l-1$, add the difference $i - 1 = (2l-1) - q$ for entries in the first q positions in the derived permutation vector. Hence we can use the first q entries from the derived permutation vector for the carry propagation process. To derive the next permutation vector, consider the first q entries of the derived permutation vector as the original permutation vector.

The process of above description is presented as Step 19-31 in Alg. 11. Note that the modulus operation is actually done in Step 22-23 utilizing the negative flag value of the status register of target processor after subtraction with $q+1$. If naively implemented, the modulus operation can cause an SPA-leakage. For example, the modulus operator % in C language can be compiled to a division instruction in assembly language of target processor which has variable execution cycles depending on the input value if an exact instruction for modulus operation do not exist. Considering mod$(q+1)$ operation, the result of subtraction with $q+1$ is negative for the inputs with the value lower than $q+1$ where for such inputs modulus reduction is not necessary hence compensating the result by addition with $q+1$ is required. On the other hand, if the result of subtraction is positive, no compensation is required. Since the negative flag value is set as one for the former case or zero for the latter case, multiplying the value of the negative flag with $q+1$ and then adding to the result perform such compensation and the correct result for the modulus operation can be acquired without SPA-leakage.

To demonstrate the practical effectiveness of our countermeasure, we implemented 128-bit version of Alg. 11 for the same target and setting from our attack experiment introduced in Section III-C. Fig. 5 presents 256 power consumption traces of our proposed global shuffling LIM in the same manner depicting Fig. 1. It can be observed that there are no differences in patterns of power traces hence revealing the random permutation vectors from SPA is not possible.

V. CONCLUSION

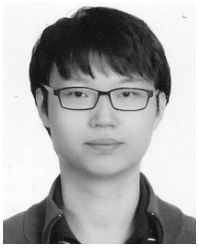
We present the first practical results of a combined single trace attacks on software implementations of global shuffling LIM by analyzing the vulnerability of the algorithm's carry propagation process which is exploitable by SPA and performing collision-based single trace attacks with power consumption subtraces of unit carry propagation operations after revealing permutation vectors. Since the vulnerability of global shuffling LIM is caused by the if-statement for selection of proper entries from the permutation vector, we propose a novel countermeasure which eliminates such selection with a permutation vector rearrangement method utilizing simple addition and modulus operation and achieves regularity in power trace patterns consequently providing security against SPA. We also provide a practical result of implementation our countermeasure.

REFERENCES

[1] R. Balasubramanian, "Elliptic curves and cryptography," in *Proc. Elliptic Curves, Modular Forms Cryptography*, Santa Barbara, CA, USA, 2003, pp. 325–345.
[2] S. S. Vasundhara, , and D. Dr. K. V. Durgaprasad, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 4, no. 3, pp. 308–311, Oct. 2011.

[3] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, 2016, pp. 104–113.
[4] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO*, Santa Barbara, CA, USA, 1999, pp. 388–397.
[5] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, New York, NY, USA: Springer, 2004, pp. 96–97.
[6] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Proc. CHES*, Worcester, MA, USA, 1999, pp. 292–302.
[7] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Math. Comput.*, vol. 48, no. 177, pp. 243–264, Jan. 1987.
[8] M. Joye and S.-M. Yen, "The montgomery powering ladder," in *Proc. CHES*, Redwood Shores, CA, USA, 2003, pp. 291–302.
[9] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 760–768, Jun. 2004.
[10] C. D. Walter, "Sliding windows succumbs to big MAC attack," in *Proc. CHES*, Paris, France, 2001, pp. 286–299.
[11] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Proc. ICICS*, Barcelona, Spain, 2010, pp. 46–61.
[12] A. Bauer, E. Jaulmes, E. Prouff, and J. Wild, "Horizontal and vertical side-channel attacks against secure RSA implementations," in *Proc. CT-RSA*, San Francisco, CA, USA, 2013, pp. 1–17.
[13] C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Roussellet, and V. Verneuil, "ROSETTA for single trace analysis," in *Proc. INDOCRYPT*, Kolkata, India, 2012, pp. 140–155.
[14] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC press, 1996, pp. 595–616.
[15] A. Bauer, E. Jaulmes, E. Prouff, J.-R. Reinhard, and J. Wild, "Horizontal collision correlation attack on elliptic curves," *Cryptogr. Commun.*, vol. 7, no. 1, pp. 91–119, Mar. 2015.
[16] N. Hanley, H. Kim, and M. Tunstall, "Exploiting collisions in addition chain–based exponentiation algorithms using a single trace," in *Proc. CT-RSA*, San Francisco, CA, USA, 2015, pp. 431–448.
[17] J.-L. Danger, S. Guilley, P. Hoogvorst, C. Murdica, and D. Naccache, "Improving the big Mac attack on elliptic curve cryptography," in *The New Codebreakers* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2016, pp. 374–386.
[18] P. Das, D. B. Roy, H. Boyapally, and D. Mukhopadhyay, "Inner collisions in ECC: Vulnerabilities of complete addition formulas for NIST curves," in *Proc. Asian HOST*, Yilan, Taiwan, 2016, pp. 1–6.
[19] S. Cho, S. Jin, and H. Kim, "side–channel vulnerabilities of unified point addition on binary huff curve and its countermeasure," *Appl. Sci.*, vol. 8, no. 10, p. 2002, Oct. 2018, doi: 10.3390/app8102002.
[20] STMicroelectronics. Amsterdam, The Netherlands. *STM32F405/415*. Accessed: Aug. 19, 2019. [Online]. Available: https://www.st.com/en/microcontrollers/stm32f405-415.html
[21] É. Brier and M. Joye, "Weierstraβ elliptic curves and side-channel attacks," in *Proc. PKC*, Paris, France, 2002, pp. 335–345.
[22] D. J. Bernstein. *Explicit-Formulas Database*. Accessed: Aug. 19, 2019. [Online]. Available: www.hyperelliptic.org/EFD
[23] H. Hisil, K. K. H. Wong, G. Carter, and E. Dawson, "Faster group operations on elliptic curves," in *Proc. AISC*, Wellington, New Zealand, 2009, pp. 7–20.
[24] C. O'Flynn and Z. D. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," in *Proc. COSADE*, Paris, France, 2014, pp. 243–260.
[25] NewAE Technology. Halifax, U.K. *CW308T-STM32F*. Accessed: Aug. 19, 2019. [Online]. Available: http://wiki.newae.com/CW308T-STM32F
[26] NewAE Technology. Halifax, U.K. *CW501 Differential Probe*. Accessed: Aug. 19, 2019. [Online]. Available: http://wiki.newae.com/CW501_Differential_Probe
[27] LeCroy. Thousand Oaks, CA, USA. *HDO6104A Teledyne*. Accessed: Aug. 19, 2019. [Online]. Available: http://teledynelecroy.com/oscilloscope/hdo6000a-high-definition-oscilloscopes/hdo6104a,
[28] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Berlin, Germany: Springer, 2007, pp. 82–86.
[29] S. Lee, S. M. Cho, H. Kim, and S. Hong, "A practical collision-based power analysis on RSA prime generation and its countermeasure," *IEEE Access*, vol. 7, pp. 47582–47592, 2019. Accessed on: Aug. 19, 2019, doi: 10.1109/ACCESS.2019.2909113.
[30] I. Diop, Y. Linge, T. Ordas, P.-Y. Liardet, and P. Maurine, "From theory to practice: Horizontal attacks on protected implementations of modular exponentiations," *J. Cryptogr. Eng.*, vol. 9, no. 1, pp. 37–52, Apr. 2019.

SANGYUB LEE received the B.S. degree in telecommunications and the M.E. degree in information security from Korea University, Seoul, South Korea, in 2011 and 2015, respectively, where he is currently pursuing the Ph.D. degree in information security with the Graduate School of Information and Security. His research interests include side-channel attacks, public-key cryptosystems, and embedded system security.



SUNG MIN CHO received the M.A. and Ph.D. degrees in information security from Korea University, Seoul, South Korea, in 2011 and 2019, respectively. He is currently with Crypt & Tech. His specialty includes in the area of information security. His research interests include algorithms for fast implementation and the side-channel attacks of public-key cryptosystems.

HEESEOK KIM received the B.S. degree in mathematics from Yonsei University, Seoul, South Korea, in 2006, and the M.S. and Ph.D. degrees in engineering and information security from Korea University, Seoul, South Korea, in 2008 and 2011, respectively. He was a Postdoctoral Researcher with the University of Bristol, U.K., from 2011 to 2012. From 2013 to 2016, he was a Senior Researcher with the Korea Institute of Science and Technology Information (KISTI). Since 2016, he has been with Korea University. His research interests include side-channel attacks, cryptography, and network security.

SEOKHIE HONG received the M.A. and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1997 and 2001, respectively. He was with Security Technologies Inc., from 2000 to 2004. From 2004 to 2005, he was a Postdoctoral Researcher with COSIC, KU Leuven, Belgium. Since 2005, he has been with Korea University, where he is currently with the Graduate School of Information Security. His specialty includes in the area of information security. His research interests include the design and analysis of symmetric-key cryptosystems, public-key cryptosystems, and forensic systems.