Retrieving Lost Efficiency of Scalar Multiplications for Resisting against Side-Channel Attacks

Keke Wu

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences and The Chinese University of Hong Kong, Shenzhen, China
Graduate University of Chinese Academy of Sciences, Beijing, China
kk.wu@siat.ac.cn

Huiyun Li, Fengqi Yu

Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences and The Chinese University of Hong Kong, Shenzhen, China
{hy.li, fq.yu}@siat.ac.cn

Abstract—At the elliptic curve cryptosystems (ECC) implementation stage, a major concern is securing scalar multiplications against so-called side-channel attacks (SCA). Existing solutions reach the goal by inserting dummy operations (typically increase 33% computational costs) based on commonly-used binary method, which largely increases the computational costs and prohibits the deployment of ECC in computation resource-restricted devices. In this paper, we for the first time propose a secure scalar multiplication method that does not penalise the computational cost compared to binary method. We partition the bit string of the scalar in half and extracting the common substring from the two parts based on bit-wise logical operations, so as to save the number of point additions required for the computation of the common substring. Computational results demonstrate the proposed method remains approximately the same computational cost as binary method. The side-channel experiments prove that the proposed method is secure against SSCA. Also, we use the randomization technique to secure our method against differential SCA (DSCA).

Keywords-Elliptic curve cryptosystems (ECC); side-channel attacks (SCA); scalar multiplication; binary method; Montgomery method

I. INTRODUCTION

Smart cards are small, portable, tamper-resistant devices providing users with convenient storage and processing capability. Because of their unique form factor, smart cards are proposed for use in a wide variety of consumer electronics in credit cards, SIM cards and set-top boxes etc. For many of these proposed applications, cryptographic services would be required. Elliptic curve cryptosystems (ECC) were independently introduced by Miller [1] and Koblitz [2] in the 1980s, which attracted increasing attention in recent years due to the requirement of shorter private key length with the same security level, in comparison with other public-key cryptosystems. A shorter private key length means reduced power consumption, computing effort and storage requirement, factors that are fundamental in portable devices. Thus, ECC's unique properties make them especially well suited to the cryptographic services in resource-restricted devices, like the smart cards.

The scalar multiplication is the dominant operation in ECC. The most commonly-used algorithm for computing scalar multiplication is binary method [3]: given an integer d and an elliptic curve (EC) point P, a scalar multiplication dP is computed by a series of addition operations (ADDs) and doubling operations (DBLs) of the point P, we will refer to them as EC-operations in this paper, depending upon the binary bit sequence of d, where d is the private key of ECC. However, such implementations of ECC are vulnerable to the widely known side-channel attacks (SCA) that were first proposed in [4, 5]. By monitoring some side-channel information (e.g., the power consumption) from the binary implementation, it is possible to deduce the inner working of an (unprotected) scalar multiplication method and thereby to recover the secret key.

To counteract these attacks, many secure scalar multiplication methods against SSCA have been proposed in literatures [6-12]. However, almost all of them involve increased computational costs (typically increase 33% computational costs) by inserting dummy point or field operations.

In this paper, we for the first time propose a secure scalar multiplication method that does not penalise the computational cost. We first partition the bit string of scalar d in half, and extract the common substrings by means of deducing the two substrings based on bit-wise logical operations. Then we compute the ADD once for every common bit. The computation is thus saved, so as to reduce the computational costs compared to the existing solutions. Moreover, we use randomization

doi:10.4304/jcp.5.12.1878-1884

technique of key to secure our method against differential side-channel attacks (DSCA).

The rest of the paper is organized as follows. In Section 2, we present the background of ECC, the side-channel attacks. In Section 3, we then propose the efficient and secure scalar multiplication method. In Section 4, we propose the differential SCA secure version of our method. In Section 5, we use side-channel experiments to prove that the proposed method is secure against SSCA. Finally, we conclude in Section 6.

II. BACKGROUND

In this section, we briefly discuss the fundamentals in ECC and summarize the current state of affairs in securing ECC against side-channel attacks (SCA).

A. Elliptic Curve Preliminaries

An elliptic curve (EC) is the set of points (x, y) which are solutions of a bivariate cubic equation over a finite field F (see [13]). It can be expressed as the form of Weierstrass equation:

$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\,, \qquad (1)$$

where $a_i \in F$, defines an EC over F.

If characteristic (char) $F \neq 2, 3$, Weierstrass equation can be transformed to: $y^2 = x^3 + ax + b$, with $a, b \in F$. The set of points on an EC, together with a special point O called the *point at infinity* can be equipped with an Abelian group structure by the following addition operation:

Addition formula [13] **for char $F \neq 2, 3$:** Let $P = (x_1, y_1) \neq O$ be a point, the inverse of P is $-P = (x_1, -y_1)$. Let $Q = (x_2, y_2) \neq O$ be a second point with $Q \neq -P$, the sum $P + Q = (x_3, y_3)$ can be calculated as:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases} \quad with\ \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1},\ if\ P \neq Q, \\ \dfrac{3x_1^2 + a}{2y_1},\ if\ P = Q. \end{cases} \qquad (2)$$

The operation of $P + Q$ is called *point addition* (ADD) if $P \neq Q$, and it is called *point doubling* (DBL) if $P = Q$. For addition formulas when char $F = 2$ or 3, please refer to [13]. Obviously, ADD is differed from DBL in formulas.

For ECC, the most commonly-used algorithm for computing scalar multiplication dP is the straightforward binary method [3] based on the binary expansion of $d = (d_k d_{k-1} \ldots d_2 d_1)_2$, where d_k and d_1 is the most significant bit (MSB) and the least significant bit (LSB) of d, respectively, and k is the number of bits of the scalar (usually larger than 256). The procedure is shown in Algorithm 1.

Algorithm 1 (Binary method)
Input: $d = (d_k d_{k-1} \ldots d_2 d_1)_2$, $P \in E$, $(d_k = 1)$. Output: dP.

$Q = O$;
for $i = 1$ to k do /* scan d from LSB to MSB */
{
  if $(d_i = 1)$ then $Q = Q + P$; /* ADD */
  $P = 2P$; /* DBL */
}
return (Q);

On average, Algorithm 1 performs $\{Q + P, 2P\}$ and $\{2P\}$ with the same probability. Thus it takes approximately $k/2$ point additions (ADDs) plus k point doublings (DBLs) to compute dP, denoted as $(k/2)A + kD$.

B. Countermeasures of Scalar Multiplications

Side-channel attacks (SCA) [4, 5] exploit leaked side-channel information, such as power consumption, electromagnetic emanations and running time etc., from running cryptographic devices (e.g. smart cards, mobile phones etc.) to reveal the secret keys, as shown in Fig. 1.

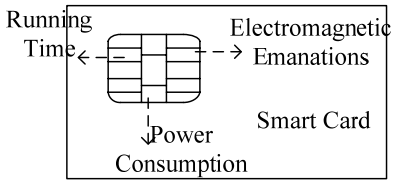


Figure 1. Side-channel attacks on a smart card.

Generally, simple SCA (SSCA) and differential SCA (DSCA) are the two main attack techniques. The goal of SSCA is to learn the secret key d using the information obtained through carefully observing the side-channel leakage from a single measurement trace. For example, binary method consists of an ADD followed by a DBL if the key bit is "1", and a DBL if the key bit is "0". If the side-channel trace pattern of DBL is different from that of ADD, attackers can easily retrieve the secret key from a single side-channel trace. The main concern for ECC is the SSCA.

Since ECC are gaining increasing popularity for implementations on smart cards and other portable devices, great research efforts have been paid to secure ECC method implementations against SSCA [6-12]. These secure methods can be generally classified into the following three groups.

(1) The first group is to make the processing of bits "0" and "1" of multiplier indistinguishable by inserting extra point operations, typically Montgomery method [6] and double-and-add-always method [7], where scalar multiplication operations appear as an ADD followed by a DBL regularly.

(2) The second group is to make ADD and DBL operations indistinguishable by inserting extra field operations to unify the addition formulas [8-10].

(3) The third group is to insert extra field operations and then divide each process into atomic blocks so that it can be expressed as the repetition of instruction blocks

which appear equivalent by SSCA [11, 12].

However, these secure methods increase the computational costs due to the extra point or field operations, i.e. they need higher computational costs for side-channel security. For example, Montgomery method, as shown in the following Algorithm 2, requires $kA + kD$ for dP to make "ADD-DBL" iterations regular, i.e. it expends extra $k/2$ ADDs for security compared to binary method.

Algorithm 2 (Montgomery method)
Input: $d = (d_k d_{k-1} \ldots d_2 d_1)_2$, $P \in E$, $(d_k = 1)$. Output: dP.

$Q[0] = P$; $Q[1] = 2P$;
for $i = k - 1$ to 1 do /* scan d from MSB to LSB */
{
$Q[1-d_i] = Q[0] + Q[1]$; /* ADD */
$Q[d_i] = 2Q[d_i]$; /* DBL */
}
return ($Q[0]$);

The more sophisticated DSCA that perform the programs multiple times and manipulate the results with statistical tools are not really a threat to ECC since they are easily avoided by randomizing the inputs [7, 14]. There is no high-order DSCA on ECC so far.

III. The Proposed Efficient and Secure Scalar Multiplication Method

In this section, we first present the simple side-channel analysis (SSCA) model, and then propose a novel method by means of partitioning the scalar d in half, and deducing them based on bit-wise logical operations. The proposed method can be resistant against SSCA and gain more efficiency than six typical secure scalar multiplication methods.

A. Our Side-Channel Models

Simple side-channel attacks (SSCA) attackers try to derive the key more or less directly from one sample. This often requires detailed knowledge about the implementation of the cryptographic algorithm under attack. Generally, the principles of SSCA are: (1) the attackers need to be able to monitor the side-channel leakage of the device under attack; and, (2) in the attacked device, the key must have a significant impact on the side-channel leakage within a trace or very few traces. We define following SSCA model and security model of ECC based on the principles of SSCA and the scalar multiplication implemented rule.

Definition 1. S*SCA model*: SSCA on scalar multiplication implementations exploits: (1) the different patterns between the side-channel features of ADD and DBL; and, (2) the different patterns depend upon the key bits due to conditional branch statements. We refer to this SSCA model as *key-dependent pattern*.

According to the Definition 1, knowledge of how the algorithm is used and implemented facilitates SSCA. Any implementation where the execution path is determined by the key bits has a potential vulnerability.

Definition 2. S*SCA-resistant security model*: The SSCA-resistant security is to remove the dependence between the different side-channel patterns of ADD and DBL and the key bits. We refer to this security model as *key-independent pattern.*

According to the Definition 2, the three group secure methods mentioned above are to remove the key-dependent patterns by means of indistinguishing the processing of bits "1" and bits "0" of multiplier d, indistinguishing the ADD and DBL, and dividing each process into atomic blocks by inserting dummy operations so that it can be expressed as the repetition of instruction blocks which appear equivalent by SSCA, respectively.

B. Our Method Model

In [15], Wu and Chang first introduced a bit string folding technique to improve the generalization of the common-multiplicand multiplications algorithm proposed by Yen and Laih [16]. The idea of the bit string folding technique inspires us to partition the bit string of the elliptic curve cryptographic scalar in half, so that a scalar multiplication can be divided into two sub-scalar multiplications. We herein partition the bit string of the scalar in half and extracting the common substring from the two parts based on bit-wise logical operations, so as to save the number of point additions required for the computation of common substring.

Let B_1 and B_2 denote bit strings. B_1 *AND* B_2 denotes the bit-wise *and logical* operation between the equal-length bit string B_1 and B_2, for example, 0110 *AND* 1010 = 0010. B_1 *XOR* B_2 denotes the bit-wise *exclusive-or logical* operation between the bit string B_1 and B_2, for example, 0110 *XOR* 1010 = 1100.

Let $B_{1\_AND\_2}$ denote the common substring of B_1 and B_2; and let B_{XOR_1} and B_{XOR_2} denote B_1 and B_2 *exclusive-or* the common substring $B_{1\_AND\_2}$, respectively. They can be defined as follows:

$$\begin{cases} B_{1\_AND\_2} = B_1 \; AND \; B_2, \\ B_{XOR\_1} = B_1 \; XOR \; B_{1\_AND\_2}, \\ B_{XOR\_2} = B_2 \; XOR \; B_{1\_AND\_2}. \end{cases} \quad (3)$$

Based on bit-wise logic operations, the equal-length bit string B_1 and B_2 can be represented as the following model:

$$\begin{cases} B_1 = B_{XOR\_1} + B_{1\_AND\_2}, \\ B_2 = B_{XOR\_2} + B_{1\_AND\_2}, \end{cases} \quad (4)$$

where the operation "+" is a normal addition. Based on this model, we develop a novel scalar multiplication algorithm. We partition the scalar d (with bit length of k) in half, i.e.

$$d = (d_2^{k/2} \ldots d_2^e \ldots d_2^1 \,||\, d_1^{k/2} \ldots d_1^e \ldots d_1^1)_2 = (B_2 \,||\, B_1), \quad (5)$$

where '||' is the concatenation operator. If k is odd, we need to add a "0" on the most significant bit, which does not affect the value of scalar d.

According to this model, scalar multiplication dP can

be computed as:

$$\begin{aligned} dP &= (B_2 \,||\, B_1)\cdot P \\ &= 2^{k/2}\cdot(B_2\cdot P)+(B_1\cdot P) \\ &= 2^{k/2}\cdot(B_{XOR\_2}\cdot P+\boxed{B_{1\_AND\_2}\cdot P}) \\ &\quad +(B_{XOR\_1}\cdot P+\boxed{B_{1\_AND\_2}\cdot P}), \end{aligned} \tag{6}$$

where $B_{1\_AND\_2}$ means the common substring of B_1 and B_2. By computing the common part $B_{1\_AND\_2}\cdot P$ once at every iteration, we can save the number of ADDs in a scalar multiplication.

C. The Proposed Method

Based on Eq. (6), an efficient and secure algorithm for computing dP is depicted in Algorithm 3 as follows. The basic idea is to partition the bit string of the scalar d in half and extract the common substring, so as to save the number of ADDs required for the computation of common substring.

Algorithm 3

Input: $d=(d_2^{k/2}...d_2^e...d_2^1 \,||\, d_1^{k/2}...d_1^e...d_1^1)_2$, $P\in E$.
Output: dP.

```
Q[0]=Q[1] = Q[2] = Q[3] = ∞;
for e = 1 to k/2 do                    /* The first loop */
{
    Q[2d_2^e + d_1^e] = Q[2d_2^e + d_1^e] + P ; /* ADD */
    P = 2P;                            /* DBL */
}
Q[1] = Q[1] + Q[3];
Q[2] = Q[2] + Q[3];
for e = 1 to k/2 do                    /* The second loop */
{
    Q[2] = 2Q[2];                      /* DBL */
}
Q[1] = Q[2] + Q[1];
return (Q[1]);
```

Obviously, the expression $2d_2^e+d_1^e$ only has four possible values as follows:

$$2d_2^e+d_1^e=\begin{cases} 0, & when\ d_1^e=0, d_2^e=0, i.e.\ d_{1\_AND\_2}^e=0; \\ 1, & when\ d_1^e=1, d_2^e=0, i.e.\ d_{XOR\_1}^e=1; \\ 2, & when\ d_1^e=0, d_2^e=1, i.e.\ d_{XOR\_2}^e=1; \\ 3, & when\ d_1^e=1, d_2^e=1, i.e.\ d_{1\_AND\_2}^e=1. \end{cases} \tag{7}$$

The results of $B_{XOR\_1}\cdot P$, $B_{XOR\_2}\cdot P$ and $B_{1\_AND\_2}\cdot P$ are contained in $Q[1]$, $Q[2]$ and $Q[3]$ after the execution of the first loop, respectively, i.e. $B_{XOR\_1}\cdot P = Q[1]$, $B_{XOR\_2}\cdot P = Q[2]$, and $B_{1\_AND\_2}\cdot P = Q[3]$. The results of B_1P and B_2P can be derived from $Q[1] = Q[1] + Q[3]$ and $Q[2] = Q[2] + Q[3]$, respectively, i.e. $B_1P = B_{XOR\_1}\cdot P + B_{1\_AND\_2}\cdot P$, and $B_2P = B_{XOR\_2}\cdot P + B_{1\_AND\_2}\cdot P$. The result of $2^{k/2}\cdot B_2P$ is contained in $Q[2]$ after the execution of the second loop. Consequently, the final result of dP can be derived from the last statement $Q[1] = Q[2] + Q[1]$, i.e. $dP = Q[1] = 2^{k/2}\cdot B_2P + B_1P$.

When $2d_2^e+d_1^e=0$, the $Q[0] = Q[0] + P$ are the dummy operations for the regular "ADD-DBL" iterations in the first loop. Fortunately, this lost efficiency can be retrieved by the common $B_{1\_AND\_2}\cdot P$. Based on Algorithm 3, the average number of ADDs and DBLs required to compute dP is

$$(k/2)(A + D) + 2A + (k/2)D + A \approx (k/2)A + kD, \tag{8}$$

which is binary method (i.e. unprotected) remains.

In terms of storage cost, Algorithm 3 requires 5 data registers to store intermediate variables of elliptic curve points and 3 additional data registers compared to binary algorithm. These slight overheads can be allowed.

In Algorithm 3, the scalar multiplication operations consist of two loops. It appears as an ADD followed by a DBL regularly in the first loop and appears as only one DBL regularly in the second loop. Therefore, Algorithm 3 is key-independent executions, which resist against SSCA.

IV. DSCA Consideration

We use randomization technique of key to secure our method (Algorithm 3) against differential side-channel attacks (DSCA) [7]. Let $\#E$ be the number of points of the curve. The computation of $Q = dP$ can be done by the following method:

(1) Select a random number r of size n bits. In practice, we can take $n = 20$ bits.

(2) Compute $s = d + r\cdot \#E$.

(3) Compute the point $Q = sP$. We have $Q=(d+r\cdot\#E)P=dP$ since $\#E\cdot P=\infty$.

Algorithm 4

Input: $d=(d_2^{k/2}...d_2^e...d_2^1 \,||\, d_1^{k/2}...d_1^e...d_1^1)_2$, $P\in E$.
Output: dP.

```
Q[0]=Q[1] = Q[2] = Q[3] = ∞;
s = d + r · #E ;        /* Randomization of key d*/
for e = 1 to k/2 do                    /* The first loop */
{
    Q[2s_2^e + s_1^e] = Q[2s_2^e + s_1^e] + P ;  /* ADD */
    P = 2P;                            /* DBL */
}
Q[1] = Q[1] + Q[3];
Q[2] = Q[2] + Q[3];
for e = 1 to k/2 do                    /* The second loop */
{
    Q[2] = 2Q[2];                      /* DBL */
}
Q[1] = Q[2] + Q[1];
return (Q[1]);
```

This countermeasure makes the DSCA infeasible since the exponent s in $Q = sP$ changes at each new execution of the algorithm. More importantly, this countermeasure also does not penalise the computational cost compared to Algorithm 3.

V. SSCA EXPERIMENTS

In this section, we use power measurement experiments to present the power traces of binary method, Montgomery method, and the proposed method (Algorithm 3). We first analyze power patterns of binary method to illustrate the concept of key-dependent pattern. Then we present that binary method is key-dependent pattern that is vulnerable to simple side-channel attacks (SSCA). To evaluate the side-channel security of the proposed method, we choose Montgomery method from existing secure methods to compare with the proposed method. Thus, we can prove that the proposed method is key-independent pattern that is secure against SSCA.

A. Experimental Setup

Our experimental setup consists of a PC, a power tracer (embedded a resistor), a digital oscilloscope, a USB line, a network cable, a BNC trigger line, a BNC signal line and three smart cards. The three smart cards with 8 bit 10MHz microprocessors are implemented with binary method, Montgomery method and Algorithm 3, respectively, where EC-operations of scalar multiplication methods are implemented in software implementation style except the hardware implementation of modulo multiplication operation in prime field. The sampling frequency of power traces is 781.2K samples per second in our experiments.

The experimental setup is shown in Fig. 2. Experiment procedure is as follows: First, a PC generates random plain texts and instructs the smart card to start scalar multiplication operation through the power tracer. Second, the power tracer, which contains a hardware trigger, will send a trigger signal to instruct a digital oscilloscope to collect the power traces of the smart card during the scalar multiplication operation. Finally, the PC receives the power traces from the oscilloscope along with the plain texts for each scalar multiplication.

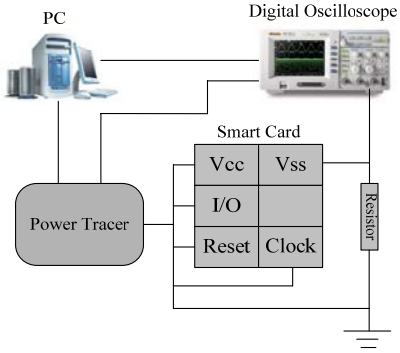


Figure 2. Power measurement setup for smart cards.

B. Insecure Binary Method

SSCA are made easier for unprotected scalar multiplication algorithms because the sequence of field operations of ADD has a different side-channel pattern than that of DBL. We do power experiment of binary method implementation (as shown in Fig. 3): the peaks in ADD trace are 3 clusters, but the uniform distribution peaks in DBL trace, and the running time of ADD is obviously longer than DBL. Therefore, the power trace satisfies the first condition of Definition 1.

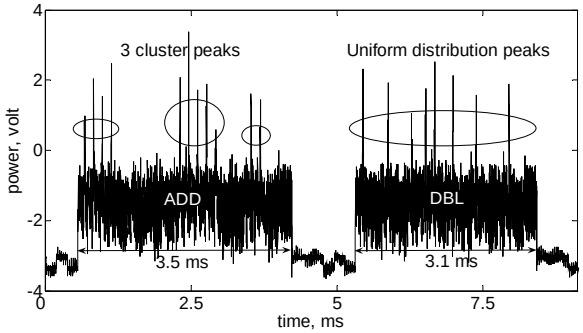


Figure 3. The different power patterns between an ADD and a DBL.

We cut one section power trace randomly from the sampled power trace as the following Fig. 4. We note that the conditional branch "if statement" in binary method and the different power patterns of EC-operations depend on the key bits. Hence, the power trace also satisfies the second condition of Definition 1. Thus, according to Definition 1, recovering the key is feasible in this naive implementation of binary method. Fig. 4 shows that key bit is "1" when "ADD" operation is followed by one "DBL" operation in one iteration between two dashed lines, whereas key bit is "0" when only one "DBL" operation in one iteration.

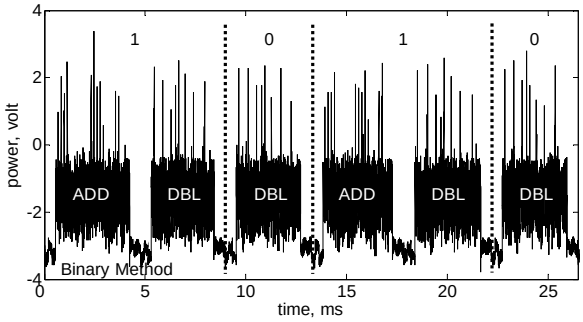


Figure 4. Key-dependent pattern of binary method.

C. Secure Montgomery Method and the Proposed Method

Side-channel analysis is performed on the proposed method and Montgomery method to prove the security of the proposed method. As mentioned above, Montgomery method and the proposed method both consist of regular operation iterations, which are key-independent executions. Experimental results are shown in Fig. 5. The upper subplot shows the regular "ADD-DBL" power trace iterations of Montgomery method, the lower subplot shows the regular "ADD-DBL" power trace iterations and the regular "DBL" power trace iterations of the proposed method. Hence, the experiment results prove that the proposed method is as secure as the Montgomery method against SSCA, while achieves more than 33% efficiency.

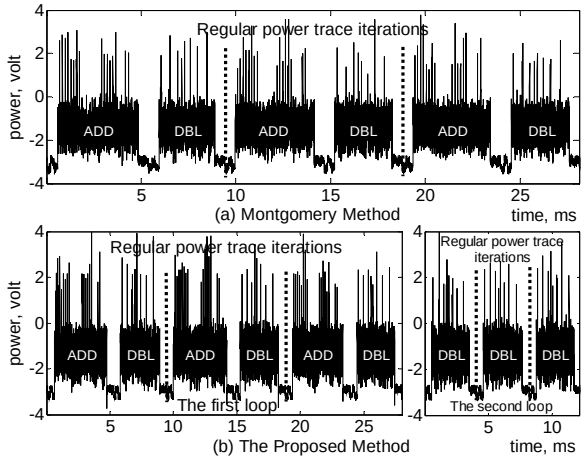


Figure 5. Key-independent pattern of Montgomery method and the proposed method.

Therefore, the conditional statements that lead to the key-dependent pattern in scalar multiplication methods pose serious threats to the security of ECC implementations. Substituting the conditional statements with other operational approaches shall be an important criterion at design time for ECC implementations.

VI. Conclusions

A novel scalar multiplication method based on scalar-partitioning is presented. The method achieves both efficiency and security. The main advantage of the proposed method is that the overall computational cost of the resulting algorithm is broadly the same as that of the most commonly-used (i.e. unprotected) binary implementation. Compared with the existing secure solutions, the proposed efficient and secure solution makes the ECC more suitable for smart card applications. Moreover, the differential side-channel attacks are also considered for the proposed method.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 60901052), and the Basic Research Project of Shenzhen (Grant No. JC200903160412A).

References

[1] V. S. Miller, "Use of elliptic curves in cryptography," In: H. C. Williams, editor, Advances in Cryptology - CRYPTO'85, LNCS 218, Springer-Verlag, pp. 417-426, 1986.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, Jan. 1987.

[3] D. E. Knuth, *The art of computer programming volume 2: seminumerical algorithms* (3nd edn). Beijing: Tsinghua University Press. 461-465, 2002.

[4] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," Proceedings of Advances in Cryptology - CRYPTO'96, LNCS 1109, Springer-Verlag, 104-113, 1996.

[5] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," Proceedings of Advances in Cryptology – CRYPTO'99, LNCS 1666, Springer-Verlag, pp. 388-397, 1999.

[6] P. L. Montgomery, Speeding the pollard and elliptic curve algorithms for Factorizations. *Mathematics of Computation*, 48(177): 243-264, 1987.

[7] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," Proceedings of Cryptographic Hardware and Embedded Systems – CHES'99, LNCS 1717, Springer-Verlag, 292-302, 1999.

[8] E. Brier, and M. Joye, "Weierstrass elliptic curves and side-channel attacks," Proceedings of Public Key Cryptography 2002, LNCS 2274, Springer-Verlag, 335-345, 2002.

[9] M. Joye, and J. J. Quisquater, "Hessian elliptic curves and side-channel attacks," Proceedings of Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS 2162, Springer-Verlag, 402-410, 2001.

[10] O. Billet, and M. Joye, "The Jacobi model of an elliptic curve and side-channel analysis," Proceedings of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes 2003, LNCS 2643, Springer-Verlag, 34-42, 2003.

[11] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity," *Transactions on Computers*, 53(6): 760-768, 2004.

[12] P. K. Mishra, "Pipelined computation of scalar multiplication in elliptic curve cryptosystems," IEEE *Transactions on Computers*, 55(8): 1000–1010, 2006.

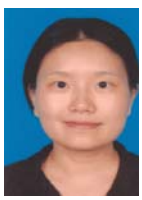
[13] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, London: Springer, 76-86. 2004.

[14] M. Joye, and C. Tymen, "Protections against differential analysis for elliptic curve cryptography: an algebraic approach," Proceedings of Cryptographic hardware and embedded systems - CHES LNCS 2162, Springer-Verlag, 377-390, 2001.

[15] T. C. Wu, Y. S. Chang, "Improved generalisation common -multiplicand multiplications algorithm of Yen and Laih," *IEEE Electronics Letters* (31): 1738–1739, 1995.

[16] S. M. Yen, C. S. Laih, "Common-multiplicand multiplication and its applications to public key cryptography," *IEEE Electronics Letters* (29): 1583–1584, 1993.

Ke-Ke Wu was born in Jiangxi, China, in 1980. He received the B.S. and M.S. degrees in computer science and technology from Jiangxi Normal University in 2003 and 2007, respectively. He is now a Ph.D student in Institute of Computing Technology, Chinese Academy of Sciences. His research interests include information security and side channel technology.



Hui-Yun Li was born in Sichuan, China, in 1977. She received the M.Eng. degree in Electronic Engineering from Nanyang Technological University in 2001, and the Ph.D degree from University of Cambridge in 2006. She is now with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences/The Chinese University of Hong Kong. Her research interests include cryptographic device design and evaluation.

Feng-Qi Yu was born in Harbin, China, in 1962. He received the Ph.D. degree in Integrated Circuits and Systems Lab (ICSL) at UCLA. In 2006, he joined Shenzhen Institute of Advanced Technology (SIAT) as a full professor and director of integrated electronics department. Before joining SIAT, he worked at Rockwell Science Center (USA), Intel (USA), Teradyne (USA), Valence Semiconductor (USA). His R&D interests include CMOS RF integrated circuit design, CMOS sensor design, wireless sensor networks, RFID, and wireless communications.