Efficiently Secure ECC Scalar Multiplication Methods against Power Analysis Attacks on Resource Constrained Devices

TURKI F. ALSOMANI
Computer Engineering Department
Umm Al-Qura University
P.O. Box: 715, Makkah 21955
SAUDI ARABIA
tfsomani@uqu.edu.sa

HILAL HOUSSAIN
Islamic Research and Training Institute
Islamic Development Bank
Jeddah
SAUDI ARABIA
hilal.hussein@gmail.com

Abstract: - Power analysis attacks are serious security threats to most cryptographic implementations, where these attacks may reveal the secret keys by exploiting leaked power consumption from running cryptographic devices. Most of the research efforts and proposed countermeasures against these attacks result in additional computational cost for hardware implementation. This paper presents efficient scalar multiplication methods, which is secure against the simple power analysis attacks. The main idea of the proposed method is to delay the elliptic curve point addition operation of the scalar multiplication using buffering technique. By such delay, the correlation between the key and the power consumption is eliminated. In addition, when combined with randomization techniques, the second method may also protect against differential power analysis attacks. Accordingly, the proposed methods have been implemented on an Altera Cyclone III EP3C80F780C7 FPGA and the results show that the proposed methods exhibit a time advantage over related works. Compared to other methods, the proposed methods can achieve up t o 50% time improvement for accomplishing one scalar multiplication with 173-bit, 191-bit, and 230-bit.

Key-Words: - Elliptic Curve Cryptosystems, Simple Power Analysis attacks, Differential Power Analysis Attacks, and Scalar Multiplication.

1 Introduction

Elliptic Curve Cryptosystems (ECC), which was originally proposed by Niel Koblitz and Victor Miller in 1985 [1, 2] is a serious alternative to RSA [3] with much shorter key size [4]. ECC is considered to be ideal for implementation on r esource constrained devices such a wi reless sensor networks (WSNs) (e.g., [5 - 7]), smart cards, mobile phones, radio frequency identity (RFID), etc.

Scalar multiplication (SM), kP, is the basic operation for ECC. Computing kP can be done with the straightforward *double-and-add* method (also called binary method) [4], as described in Algorithm (1), based on the binary expression of $k = (k_{m-1},\ldots,k_0)$ where k_{m-1} is the most significant bit of k.

Algorithm 1: The straightforward *double-and-add* method (left-to-right version)

Inputs: *P: Base Point, k: Secret key*
Outputs: kP.

1: $Q \leftarrow P$
2: for i = m-2 down to 0 do
  2.1 $Q \leftarrow 2Q$
  2.2 if $k_i = 1$ then $Q \leftarrow Q + P$
3: end for
Return Q

Several SM methods have been proposed in the literature [4]. Furthermore, such SM methods are disposed to the power analysis attacks (PAA), such as simple power analysis (SPA) and differential power analysis (DPA) attacks, which was introduced by Paul Kocher in 1999 [9, 10], that exploit leaked power consumption from running cryptographic devices such as WSNs, smart cards, mobile phones, RFIDs etc., to reveal the secret keys. R ecently P.

Dyka [11] in 2013 discussed the security of WSNs against Side Channel Attacks (SCA), and various solutions were proposed to secure SM against SCA [43, 44].

This paper presents an efficient scalar multiplication method, which is secure against the simple power analysis (SPA). The main idea of the proposed method is to delay the elliptic curve point addition (PADD) operation of the scalar multiplication using buffering technique; i.e., some points are stored into buffer and the PADD operations are performed in later stages. By such delay, the correlation between the key and the power consumption is eliminated. In addition, when combined with randomization techniques, the proposed method may protect against some DPA attacks.

The rest of the paper is organized as follows. In Section 2, the background of ECC, and the PAA are presented. In Section 3, we described the proposed SM method, followed by an example. In section 4, we conducted a performance and a security analysis of the proposed SM method. Section 5 concludes the presented paper.

2 Background

2.1 Elliptic Curve Cryptography

ECC, which was originally proposed by Niel Koblitz and Victor Miller in 1985 [1, 2] is a serious alternative to RSA [3] with much shorter key size. For instance, as recommended by NIST for ECC and RSA key length, ECC-160 provides comparable security to RSA-1024 and ECC-224 provides comparable security to RSA-2048 [4]. To date, no significant breakthroughs have been made in determining weaknesses in the ECC algorithm, which is based on the discrete logarithm problem over points on an elliptic curve. The advantage of ECC is being recognized recently where it is being incorporated in many standards. ECC is considered to be ideal for implementation on resource constrained devices such a WSN (e.g., [5 - 7]), smart cards, mobile phones, RFID, etc.

Computing $P + Q$ is called PADD if $P \neq Q$ and is called elliptic curve point doubling (PDBL) if $P = Q$. Elliptic Curve Point subtraction (PSUB) is a useful operation in some algorithms. This operation can be performed with the PADD formula using the additive inverse of the point to be subtracted. For example, the point subtraction $P - Q$ can be computed using the PADD operation where: $P - Q = P + (-Q)$. The additive inverse of a point $P = (x, y)$ is the point $(x, x + y)$ for curves defined over the $GF(2^m)$ fields.

Moreover, adding a point P on the elliptic curve E to itself a number of times (k) is known as the scalar product (kP) of point P by the scalar k. SM is the basic operation for ECC. SM in the group of points of an elliptic curve is analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m. Computing kP can be done with the straightforward *double-and-add* method [4], as described in Algorithm (1), based on the binary expression of the multiplier of $k = (k_{m-1},\ldots,k_0)$ where k_{m-1} is the most significant bit of k. The straightforward *double-and-add* method inspects the bits of the scalar multiplier k, if the inspected bit $k_i = 0$, only PDBL is performed. If, however, the inspected bit $k_i = 1$, both PDBL and PADD are performed. This method requires an average of m PDBL + $[m/2]$ PADD. A good survey on SM methods is presented in [4].

2.2 Power Analysis Attacks on ECC

In 1996, Paul Kocher introduced the power analysis procedure; then, in 1999 he introduced the PAA. These attacks have become a major threat against tamper resistant device [8 - 10] [47]. PAA allow adversaries to obtain the secret key, or partial information on it, by observing the power consumption traces of a cryptographic device. This is a serious threat especially to mobile devices such as WSNs, smart cards, mobile phones, RFIDs etc. Thus, implementers need algorithms that are not only efficient, but also PAA-resistant. Two main PAA techniques are the SPA and DPA:

2.2.1 SPA Attack

The main idea of the SPA attacks [10] is to get the secret k using the side-channel leakage information obtained through observing the power consumption from a single measurement trace. For instance, PDBL, in Algorithm (1), is executed for each bit of the scalar k and PADD is executed only if the scalar bit is equal to one. If the power consumption trace pattern of PDBL is different from that of PADD, the side-channel leakage of the implementation reveals the presence of the PADD and thus the value of the scalar bits and attackers can easily retrieve the secret key from a single side-channel trace.

2.2.2 DPA Attack

In DPA attacks [10], the adversary makes use of the obvious variations in the power consumption that are caused by multiple data and operation computations, and use statistical techniques to pry the secret information. This attack uses a two round technique:

(1) data collection and (2) data processing. DPA attack on SM is described in [14]. However, many research efforts have been made to countermeasure the DPA attacks [8][12 - 14][35 - 37][47]. Therefore, more advanced DPA attacks techniques applicable to elliptic curve cryptosystems, such as refined power analysis (RPA) [12], zero power analysis (ZPA) [15], SVPA (Same Values Power Analysis) [17], and doubling attacks (DA) [16] were introduced.

2.2.3 Countermeasures

Since 1996, many research efforts [14] [22 - 30] have been made to secure ECC method implementations, in special the SM, against PAA. The major challenge is to avoid additional computational cost, and to develop relatively fast cryptosystems without compromising security, due to the nature of the resource constrained devices.

There are different strategies to resist SPA attacks. These strategies share the same objective, which is to render the power consumption traces that are caused by the data and operation computations during an SM independent from the secret key. One countermeasure is the *double-and-add-always* algorithm [14], it is highly regular, and it requires no pre-computation or prior recoding. This algorithm requires *m* PDBL and *m* PADD regardless of the value of the scalar multiplier, and two temporary registers are needed to store the results of each iteration (see Algorithm (2)).

Algorithm 2: The *double-and-add-always* Method

Inputs: *P: Base Point, k: Secret key.*
Outputs: *kP.*

1: $R[0] \leftarrow O$.
2: for $i = m$-1 down to 0 do
  2.1: $R[0] \leftarrow 2R[0]$, $R[1] \leftarrow R[0] + P$.
  2.2: $R[0] \leftarrow R[k_i]$.
3: end for
Return $R[0]$.

Another countermeasure is the Montgomery ladder algorithm [25], which for every bit of *k*, computes both a PADD and a P DBL operation, and this algorithm avoids the usage of dummy instructions. Montgomery ladder [25] resists the normal DA. However, it is attacked by the relative DA proposed by S.M. Yen [18]. Moreover, recent studies have shown that processing the bits of multiplier from least to most, as Montgomery ladder does, are vulnerable to certain attacks [16]. Algorithm (3) below outlines this method.

Algorithm 3: Montgomery ladder Method

Inputs: *P: Base Point, k: Secret key.*
Outputs: *kP.*

1: $R[0] \leftarrow P$, $R[1] \leftarrow 2P$
2: for $i = m$ - 2 down to 0 do
  2.1: $R[1 - k_i] \leftarrow R[0] + R[1]$
  2.2: $R[k_i] \leftarrow 2R[k_i]$.
3: end for
Return $R[0]$.

Same as in SPA attacks, there are different approaches and techniques [12 - 14][35 - 37][47] used to resist DPA attacks. In general, the traditional and straightforward approach is by randomizing the intermediate data, thereby rendering the calculation of the hypothetical leakage values rather impossible.

3 Previous scalar multiplication methods against PAA

Ciet and Joye's Method. This method [13] uses the variant of Shamir's double ladder to compute the muli-scalar multiplication $k_1P + k_2Q$. The main difference is to insert a dummy operation in the computation. So, each loop includes one doubling and one addition, and the operation order is a repeated sequence of PDBL and PADD operations in Algorithm 3. Hence one PDBL and one PADD per bit is needed.

Lee's Method. To resist SPA, Lee improved the simultaneous scalar multiplication in [45]. He changed the value of (ki, mi) when (ki, mi) = (0, 0) to construct another adequate digit pair with at least one non-zero digit. The adjacent pair (ki+1, mi+1) should be modified as wel l. After the transformation, the digit pair (ki,mi) cannot be all zero. Therefore, the modified simultaneous scalar multiplication was proposed by Lee to resist SPA. The cost of Lee's algorithm is one PDBL and one PADD per bit.

Zhang, Chen, Xiao's Algorithm. This algorithm [42] proposes four scalar multiplication algorithms against power analysis. Those algorithms are all based on the highest-weight binary form (HBF) of the scalars and randomization to resist power analysis. Although those four countermeasures have no dummy operations, the efficiency of them is similar to Ciet and Joye's algorithm. They also almost need one PDBL and one PADD per bit. One of these algorithms can be seen as follows in Algorithm5.

Liu, Tan, and Dai's Algorithm. Liu et al. also propose a multi-scalar multiplication to resist SPA in [46]. The difference is that they use a joint sparse form (JSF) to represent a pair of integers and process

two or three JSF columns each time. Although the processed column number may be different, the algorithm always performs 4 PDBL and 2 PADD in each loop. This means that it is not possible for useful information related to the private key to be obtained by the attacker through SPA. The theoretical analysis and simulation results show that this algorithm needs 1.384 PDBLs and 0.692 PADDs per bit.

4 The Proposed Scalar Multiplication Methods

4.1 The Buffer-Based Method

The first proposed scalar multiplication method is presented here. The main idea is to modify the right-to-left (R2L) version of Algorithm 1 method through delaying the computation of PADD operation for bit value of 1 by cashing its corresponding PDBL value in a buffer of fixed size r. Each of the cashed points in the buffer is associated with the location of bit value 1 in the scalar. Once the buffer is full or upon the completion of the scalar's bits inspection, compute a P ADD operation on the cashed points. After all bits of the scalar are full inspected, the resulting PADD operation value is assimilated to produce the scalar product kP. Thus, the PADD operation is delayed, and turned to be independent of the scalar bit value k_i. The pseudocode of the proposed scalar multiplication method is given in Algorithm 4. The inspection of the scalar is performed in Step 2. All points are initially cashed into the buffer (Step 2.1), but the buffer index is only incremented for bit value of 1 (Step 2.4). The PDBL operation is computed for all bits in Step 2.2. In Step 2.3, the computation of PADD operation is performed on a ll cashed points in the buffer. All PADD operations are added in the accumulation point R[0] in the last step of the algorithm to produce the multiplication kP.

For example, for a key of 8-bit length, and value of 186 equivalent to $(10111010)_2$ in binary, and a buffer capacity of 3, PDBL computations (those corresponding to scalar bit value of 1) are cached twice in the buffer of fixed size 3, as follow:

1. In the first iteration, and because the buffer became full, only 2P, 8P, 16P correspond to the scalar bit positions of 1,3,4 respectively, and then
2. In the second iteration, and as the scalar scan is completed, 32P, 128P correspond to the scalar bit positions of 5, and 7 respectively.

Where in each iteration (Step 2.3) a PADD operation is performed on the cached points, so that R[0] values are evaluated as 26P (2P + 8P + 16P) and 186P (26P + 32P + 128P), respectively. Finally, the output value of the scalar multiplication is represented by the final value of R[0], which is equal to 186P. The intermediate results computed by the proposed method are shown in Table 1. Furthermore, the dataflow of the proposed method is depicted in Figure 1.

Algorithm 4: The Buffer-Based method

Inputs: *P: Base Point, k: Secret key, r is capacity limit of buffer*
Outputs: kP.

1: $R[0] \leftarrow O$, $t \leftarrow 1$ /* set buffer index t to 1 */
2: for $i =$ 0 to m-1 do
  2.1: $B[t] \leftarrow P$ /* scan k, store points in buffer */
  2.2: $P \leftarrow 2P$
  2.3: If ($t = r$) or ($i = m$-1), then /* buffer reach its capacity limit or scan k is done */
    2.3.1: for s = 1 to t do
    2.3.1.1: $R[0] \leftarrow R[0] + B[s]$
    2.3.2: $t \leftarrow 1$ /* reset buffer index to 1 */
  2.4 else $t \leftarrow t + k_i$ /* increment t if the bit value of k is 1 */
Return R[0].

4.2 The Randomized Buffer-Based Method

The second method introduces randomization technique to the first method by randomizing the accumulation step of the resulting point from processing the key partitions (see Algorithm 5 and Figure 2).

In Figure 2, the scalar multiplier k is scanned from right to left and for every scalar bit value:

1. Perform a PDBL operation. PDBL operation keeps the significance of the point value at the scalar bit position of the scalar.
2. Write to buffer the updated value of P (result of PDBL operation) The buffer capacity is randomized (greater than zero, and less or equal to the initial random capacity). Index to buffer is directly related to the bit scalar value; i.e., it will only increment for bit value of 1. Therefore, the buffer will only store points corresponding to bit value of 1.
3. Once the buffer is full (i.e. the number of stored points is equal to the capacity of the buffer after applying randomization), the PADD operation is performed on a random number of points stored in the buffer.
4. When the scalar scanning is completed, the PADD operation is performed on t he remaining points in the buffer.

The scalar multiplication will be the accumulated points of the PADD operation results.

Algorithm 5: The Randomized Buffer-Based method

Inputs: *P: Base Point, k: Secret key, r is capacity limit of buffer*
Outputs: *kP*.

1: $R[0] \leftarrow O$, $t \leftarrow 1$ /* set buffer index t to 1 */
2: for $i =$ 0 to m-1 do
  2.1: $r' \leftarrow$ RNG (0,r); /* Generate random number r', where $0 < r' <$ buffer capacity r.
  2.2: $B[t] \leftarrow P$ /* scan k, store points in buffer */
  2.3: $P \leftarrow 2P$
  2.4: if ($t \leftarrow r'$) then
    2.4.1: $j \leftarrow$ RNG ($\leq r'$); random number generator for a number less than or equal to i
    2.4.2: for $s = j$ to t do
      2.4.2.1: $R[0] \leftarrow R[0]+B[s]$
    2.4.3: $t \leftarrow j$; Reset buffer (to avoid calculating resident points from previous iteration)
  2.5: else $t \leftarrow t + k_i$ /* increment t if the bit value of k is 1 */
  2.6: if $i \leftarrow m$-1 then
    2.6.1: for $s = 1$ to $t - 1$ do
      2.6.1.1: $R[0] \leftarrow R[0]+B[s]$
Return $R[0]$.

5 Performance/Security Analysis and Comparison

5.1 Performance Analysis

In algorithm 4, each loop of Step 2 has one PDBL and a PADD for only the cached points in the buffer, which are corresponding the bit value of 1 of the scalar. Therefore, each bit needs one PBDL and ½ PADD. In order to show the performance of Algorithm 4, a comparison with previous methods is listed in Table 2. This can be improved when NAF encoding is used [47]. Additionally, this method requires no extra dummy computation.

In the proposed Randomized Buffer-based method (Algorithm 5), on the other hand, PADD is performed in later stage and only if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit value k_i. In addition, this proposed method is derived from the binary method (See Algorithm 1); therefore, the performance required by the proposed Buffer-based method is m PDBL and an average of $m/2$ PADD operations, which is equivalent to the performance of the binary method, and it has a better performance in compared to the double-and-add always method. In addition, Buffer-based method requires no extra dummy computation. Similarly, this can be improved to m PDBL and an average of $m/3$ PADD when NAF encoding is used [47].

5.2 Security Analysis

In the proposed method, the PADD operation is delayed by storing points in a buffer, and a PDBL with "write to buffer" is performed for every bit value, and thus the relation between the scalar bit value and point operation is removed. Therefore, this proposed method is robust against SPA attacks since the point operations (PDBL and PADD) are independent of the bit scalar value.

Moreover, the security of this proposed method depends on the provided depth of confusion which is directly proportional to the size of the buffer, i.e., the smaller the buffer is, the easier to guess the number of processed bit "1" during the sequence of PDBL operations, and it will be harder when the buffer is larger.

In the randomized buffer-based method (Algorithm 5), on the other hand, the PADD operation is delayed by storing points in a buffer, and a PDBL with "write to buffer" is performed for every bit value, and thus the relation between the scalar bit value and point operation is removed. In addition, randomization technique is used in number points stored in the buffer, and the number of points processed for PADD in the buffer. However, resistance against all DPA attacks can be achieved by combining two or more of the countermeasures proposed in the literature thus far [48].

5.3 Performance Comparison with other Recent SM Methods

Table 2 compares the number of point operations (PADD and PDBL) needed for the first and second proposed methods to those needed for other recently reported scalar multiplication algorithms with DPA resistant countermeasures. Both of the proposed methods require the same number of point operations, but only the second one will be considered in this comparison as it provides security against DPA attacks.

The five selected countermeasures are the Modular Scalar Randomization (MSR) reported in Section 4 of [13], the Binary Random Initial Point method (BRIP) [41], and the three methods presented in [42]. As it is shown in the comparison (Table 2), the proposed SM methods in this paper outperform the other methods by at least [$m/2$]*PADD

operations, and thus it is efficiently applicable on resource constrained devices.

5.4 Implementation Results and Comparisons

To conduct an appropriate evaluation of the two proposed SM methods, these methods are compared to other two similar methods; One is proposed in [14], which is based on the *double-and-add-always* algorithm (See Algorithm 2), and the other is proposed by [28], and it is based by partitioning the bit string of the scalar into half and extracting the common substring from the two parts based on propositional logic operations.

We implemented the four proposed SM methods over GF(2^{173}), GF(2^{191}), and GF(2^{230}) for different m sizes as recommended by NIST (m ϵ {173, 191, 230}) on an Altera Cyclone III EP3C80F780C7 FPGA, which contains 81,264 Slices. It was to use an identical FPGA chip with these methods to ensure that the power, delay and area comparisons were performed for the same technology and FPGA architecture and resources.

Table 3 lists the implementation results for all four methods in terms of 1 - Delay measured in ms, 2 - Area measured in number of slices, and 3 - Power consumed measured in mW. As expected, the implementation results show that our proposed methods outperformed the other two methods in terms of time delay for all values of m. In addition, very close results for all methods in terms of area, and power consumption, and this quite justifiable as our proposed methods are using slightly extra registers for buffering.

6 Conclusion

Two SM methods are proposed in this paper, and these proposed methods provide security against power analysis attacks on resource constrained devices. These methods eliminate the correlation between the key and the power consumption by delaying the PADD operation of the SM using buffering technique. Efficient performance and high security are both achieved by the two proposed methods. For instance, and in comparison with other recently reported scalar multiplication algorithms with power analysis attacks resistant countermeasures, the performance analysis shows that the proposed methods achieve higher performance, i.e. m*PDBL + [m/2]*PADD.

Furthermore, we implemented the proposed methods together with other two similar methods for GF(2173), GF(2191) and GF(2230) on an Altera Cyclone III EP3C80F780C7 FPGA. The results in terms of the cost measurements (time delay, power and space) and security level prove that the proposed methods are feasibly implementable on resource-constrained devices such as W SNs, RFID mobile devices and smart card technology.

References:

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation,* vol. 48, p. 203–209, 1987.

[2] V. S. M iller, "Use of elliptic curves in cryptography," in *CRYPTO '85: Proceedings of the Advances in cryptology*, New York, NY, USA, 1986.

[3] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM, Vol. 21, No.2, pp. 120-126, DOI: 10.1145/357980.358017, 1978.*

[4] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.

[5] D. Malan, M. Welsh and M. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04), pp. 71–80*, Santa Clara, Calif, USA, 2004.

[6] H. Houssain, M. Badra and T. F. Al Somani, "Hardware Implementations of Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proc. 6th International Conf. on Internet Technology and Secured Transactions (ICITST 2011)*, Abu Dhabi, UAE, pp. 1-6, Dec 2011.

[7] N. Gura, A. Patel, A. S. Wander, H. Eberle and S. Chang Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," *in Cryptographic Hardware and Embedded Systems — CHES 2004,* vol. 3156 of Lecture Notes in Computer Science, pp. 119–132, Springer Verlag, 2004.

[8] J. Fan and I. Verbauwhede, "An updated survey on secure ECC implementations: Attacks, countermeasures and cost," *Cryptogr. Secur. From Theory to Appl.*, pp. 265–282, 2012.

[9] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology, Proc. CRYPTO '96*, N. Koblitz, ed., pp. 104-113, 1996.

[10] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," in *Proc. Adv. Cryptology – CRYPTO'99*, Santa Barbara, CA, 1999, vol. 1666, pp. 388–397.

[11] P. Dyka, Zoya and Langendörfer, "Improving the Security of Wireless Sensor Networks by Protecting the Sensor Nodes against Side Channel Attacks," in *Wireless Networks and Security*, A.-S. Khan, Shafiullah and Khan Pathan, Ed. Springer Berlin Heidelberg, pp. 303–328, 2013.

[12] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," in *Proceedings of PKC 2003, LNCS 2567*, pp. 199-211. Springer Berlin / Heidelberg, 2003.

[13] M. Ciet and M. Joye, "(Virtually) Free Randomization Techniques for Elliptic Curve Cryptography," *in Information and Communications Security (ICICS2006), LNCS 2836,* Springer, 2003, pp. 348–359.

[14] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems – CHES 1999*, Worcester, MA: Springer, 1999, vol. 1717, pp. 292–302.

[15] T. Akishita and T. Takagi, "Zero-value register attack on elliptic curve cryptosystem," IEICE Transactions, 88-A(1):132–139, 2005.

[16] P. Fouque and F. V alette, "The doubling attack– why upwards is better than downwards," in *Proc. CHES'03*, 2003, vol. 2779, pp. 269–280.

[17] C. Murdica, S. G uilley, J.-L. Danger, P. Hoogvorst and D. Naccache, "Same values power analysis using special points on elliptic curves," in *In Proceedings of the Third international conference on Constructive Side-Channel Analysis and Secure Design (COSADE'12), Werner Schindler and Sorin A. Huss (Eds.). Springer-Verlag, Berlin, Heidelberg, 183-198. DOI=10.1007/978-3-642-29912-4_14*.

[18] S. M. Yen, L. C. Ko, S. J. Moon and J. C. Ha, "Relative doubling attack against montgomery ladder," in *Proc. ICISC'05*, 2006, vol. 3935, pp. 117–128.

[19] S. Chari, J. R. Rao and P. Rohatgi, "Template Attacks," *in Cryptographic Hardware and Embedded Systems, CHES, ser. LNCS,* vol. 2523, 2002, pp. 13–28.

[20] P. Fouque, D. R´eal, F. V alette and M. Drissi, "The Carry Leakage on the Randomized Exponent Countermeasure," *in Cryptographic Hardware and Embedded Systems - CHES, ser. LNCS,* vol. 5154. Springer, 2008, pp. 198–213.

[21] L. Batina, J. Hogenboom, N. Mentens, J. Moelans and J. Vliegen, "Side-channel evaluation of FPGA implementations of binary Edwards curves," in *in International Conference on Electronics, Circuits and Systems 2010*, pp. 1255-1258, Athens, Greece, Dec. 12-15, 2010.

[22] B. Möller, "Parallelizable elliptic curve point multiplication method with resistance against side-channel attacks," in *Int. Conf. on Information Security (ISC 2002)*, Sao Paulo, Brazil, 2002, vol. 2433, pp. 402–413.

[23] L. P.Y. and S. NP, "Preventing SPA/DPA in ECC systems using the Jacobi form," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001)*, Paris, France, 2001, vol. 2162, pp. 391–401.

[24] E. Brier and M. Joye, "Weierstraß elliptic curves and side-channel attacks," in *David Naccache and Pascal Paillier (Eds.), Public Key Cryptography, vol. 2274 of Lecture Notes in Computer Science*, pp. 335–345. Springer, Berlin / Heidelberg, 2002.

[25] P. Montgomery, "Speeding up the Pollard and elliptic curve methods of factorization," *Mathematics of Computation,* vol. 48, no. 177, pp. 243–264, 1987.

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

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

[28] W. Keke, L. Huiun, Z. Dingju and Y. Fengqi, "Efficient Solution to Secure ECC Against Side-channel Attacks," 2011 20 (CJE-3): 471-475.

[29] É. Brier, I. Déchène and M. Joye, "Unified PADDition formulæ for elliptic curve cryptosystems," *In Embedded Cryptographic Hardware: Methodologies & Architectures.,* Nova Science Publishers, 2004.

[30] T. F. A l-Somani and A. A. Amin, "High Performance Elliptic Curve Scalar Multiplication with Resistance against Power Analysis Attacks," *Journal of Applied Sciences,* Volume 8 (24), 2008, pp. 4587-4594.

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

[32] P. Longa, *Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields.,* PhD thesis, School of Information Technology and Engineering, University of Ottawa, 2007.

[33] C. Giraud and V. Verneuil, "Atomicity Improvement for Elliptic Curve Scalar Multiplication," CARDIS 2010: 80441.

[34] D. Bernstein and T. Lange, "Faster Addition and Doubling on Elliptic Curves," *Advances in Cryptology - ASIACRYPT, K. Kurosawa (ed.),* vol. 4833 of LNCS, pp. 29-50, Springer, 2007.

[35] S. Ghosh, D. Mukhopadhyay and D. R. Chowdhury, "Petrel: Power and Timing Attack Resistant Elliptic Curve Scalar Multiplier Based on Programmable GF(p) Arithmetic Unit," *IEEE Trans. on Circuits and Systems 58-I(8)* , : 1798-1812 (2011).

[36] M. Hedabou, P. Pinel and L. Bénéteau, "A comb method to render ECC resistant against Side Channel Attacks," IACR Cryptology ePrint Archive 2004: 342, 2004 .

[37] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography," *In: [cKKNP01] Cryptographic Hardware and Embedded Systems – CHES 2001, Lecture Notes in Computer Science,* Vol. 2162, pp. 377.

[38] D. Naccache, N. P. Smart and J. Stern, "Projective Coordinates Leak," *In: Advances in Cryptology - EuroCrypt 2004, Lecture Notes in Computer Science,* Vol. 3027, pp. 257–267. Springer, Berlin / Heidelberg, 2004.

[39] J. Ha, J. Park, S. Moon and S. Yen, "Provably Secure Countermeasure Resistant to Several Types of Power Attack for ECC," *in Information Security Applications (WISA),* vol. 4867. Springer, 2007, pp. 333–344.

[40] T. Akishita and T. Takagi, "Zero-Value Point Attacks on Elliptic Curve Cryptosystem," vol. 2851, pp. 218–233, 2003.

[41] H. Mamiya, A. Miyaji and H. Morimoto, "Efficient countermeasure against RPA, DPA, and SPA," *Cryptographic Hardware and Embedded Systems - CHES '04, LNCS 3156, Springer-Verlag, pp. 343-356.*

[42] N. Zhang, Z. Chen, and G. Xiao, "Efficient elliptic curve scalar multiplication algorithms resistant to power analysis," *Inf. Sci. (Ny).*, vol. 177, no. 10, pp. 2119–2129.

[43] H. Liu, Y. Zhou, and N. Zhu, "A Novel Elliptic Curve Scalar Multiplication Algorithm against Power Analysis," *Math. Probl. Eng.*, vol. 2013, 2013.

[44] J.-H. Ye, S.-H. Huang, and M.-D. Shieh, "An efficient countermeasure against power attacks for ECC over GF(p)," *2014 IEEE Int. Symp. Circuits Syst.*, no. 1, pp. 814–817, Jun. 2014.

[45] M. Lee, "SPA-resistant simultaneous scalar multiplication," *Computational Science and Its Applications–ICCSA*, vol. 3481, pp. 314–321, 2005.

[46] D. Liu, Z. Tan, and Y. Dai, "New Elliptic Curve Multi-scalar Multiplication Algorithm for a Pair of Integers to Resist SPA", Lecture Notes in Computer Science, vol. 5487, pp. 253–264. Springer, Berlin / Heidelberg, 2009.

[47] F. Morain and J. Olivos, "Speeding up the computations on a n elliptic curve using addition-subtraction chains," *Theoretical Informatics and Applications,* 24, p. 531–543, 1990.

[48] H. Houssain, M. Badra and T. Al-Somani, "Power Analysis Attacks on ECC: A Major Security Threat," *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 3, issue. 6 ,p. 90 - 96, 2012.

Table 1: Example of the Proposed Buffer-Based Method

Iteration #	bit	$B[t]$	PBDL	$t = r$? (Buffer is full)? ($r = 3$)	$i = m - 1$? (End of Scalar Inspection?)	$R[0]$: the scalar multiplication result (kP)	$t = t +$ bit
1	0	$B[1]= P$; t=1	$P = 2P$	No	No	O	1+0 = 1
	1	$B[1]=2P$; t=1	$P = 4P$	No	No	O	1+1 = 2
	0	$B[2]=4P$; t=2	$P = 8P$	No	No	O	2+0 = 2
	1	$B[2]=8P$; t=2	P=16P	No	No	O	2+1 = 3
	1	$B[3]=16P$; t=3	P=32P	Yes	No	1.1:$R[0] = O + B[1] = 2P$ 1.2:$R[0] = 2P + B[2] = 10P$ 1.3:$R[0] = 10P + B[3] = 26P$	1
2	1	$B[1]=32P$; t=1	P=64P	No	No	26P	1+1=2
	0	$B[2]=64P$; t=2	P=128P	No	No	26P	2+0=2
	1	$B[2]=128P$; t=2	P=256P	No	Yes	2.1:$R[0]$=26P+32P =58P 2.2:$R[0]$=58P+128P=186P	1

Table 2: The Number of Point Operations for the Proposed Methods and Other Recent SM Methods

Countermeasure	Point Operations
MSR [13]	m*PDBL + m*PADD
BRIP [41]	m*PDBL + m*PADD
Algorithm 5 in [42]	[m + 1]*PDBL + [m + 1]*PADD
Algorithm 6 in [42]	m*PDBL + m*PADD
Algorithm 7 in [42]	m*PDBL + m*PADD
The proposed methods here	m*PDBL + [m/2] *PADD

Table 3: Synthesis results of the two proposed SM methods

Cryptoprocessor	m	Clock(MHz)	Delay(ms)	Area (Slices)	Area Usage	Power (mW)
Method in [14]	173	34.11	16.793	22,292	27%	169.70
Method in [28]	173	35.80	10.114	25,977	32%	178.64
First Method	173	41.00	8.831	25,954	32%	173.97
Second Method	173	37.97	9.536	26,155	32%	174.00
Method in [14]	191	33.29	20.959	24,576	30%	177.78
Method in [28]	191	25.63	17.207	28,703	35%	187.53
First Method	191	40.36	10.927	28,625	35%	181.01
Second Method	191	29.34	15.031	29,746	37%	183.14
Method in [14]	230	22.56	44.797	29,539	36%	193.09
Method in [28]	230	22.76	28.063	34,483	42%	199.57
First Method	230	23.05	27.710	35,060	43%	196.77
Second Method	230	22.78	28.038	36,190	45%	197.54

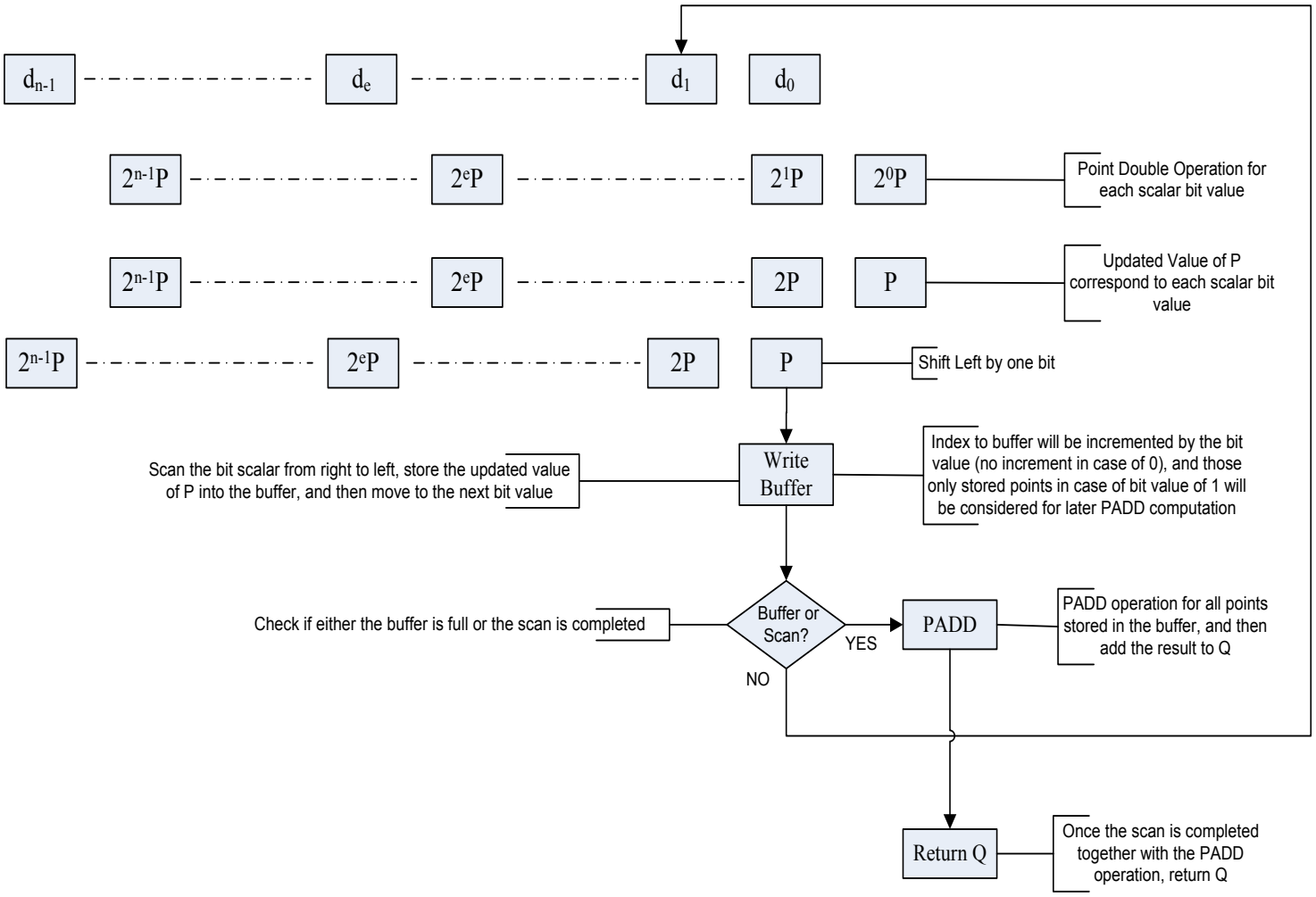

Figure 1: The Dataflow of the Buffer-based Method.

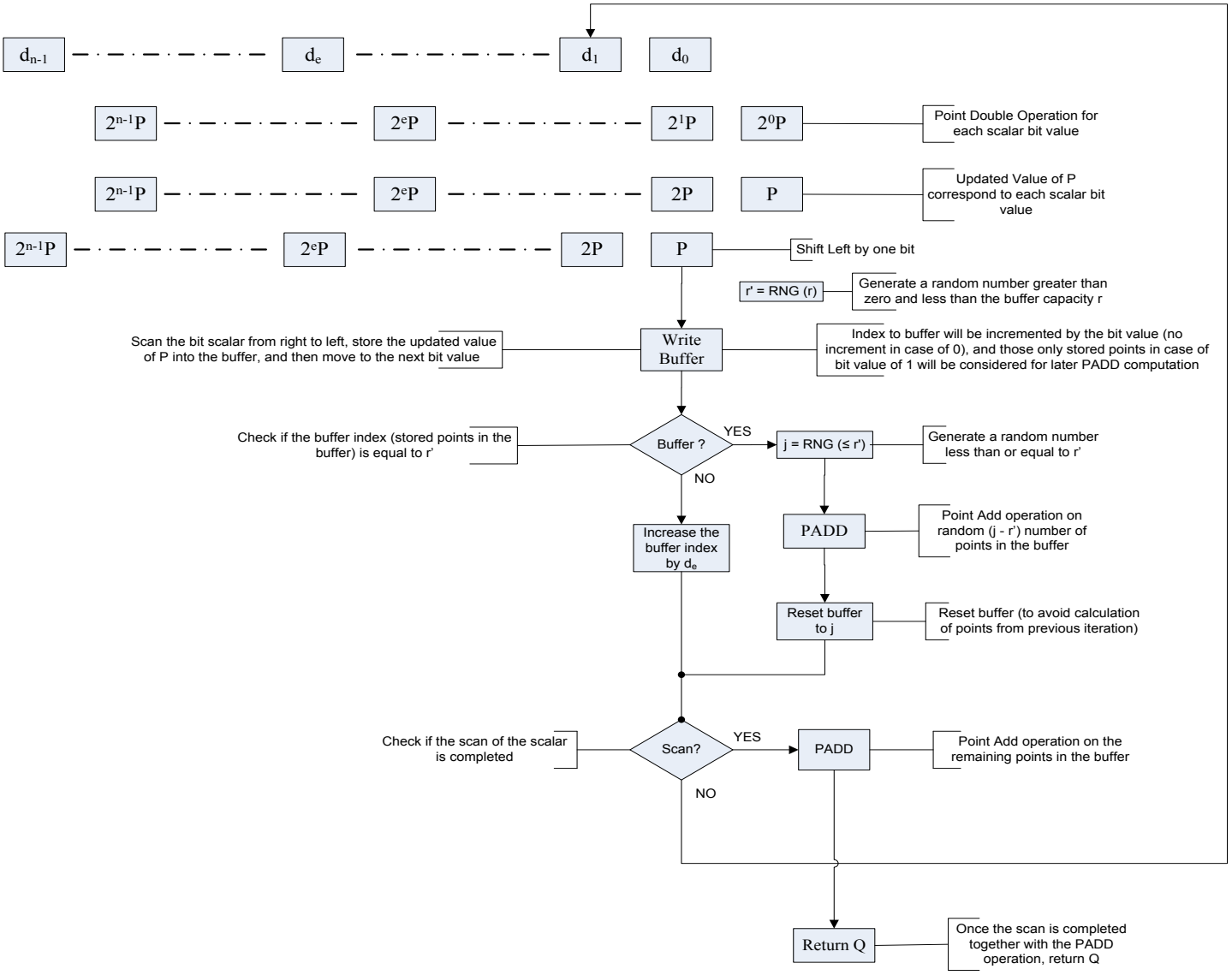

Figure 2: The Dataflow of the Randomized Buffer-based Method.