Improving the Performance of {0,1,3}-NAF Recoding Algorithm for Elliptic Curve Scalar Multiplication

Waleed K. AbdulRaheem[1], Sharifah Bte Md Yasin[2], Nur Izura Binti Udzir[3], Muhammad Rezal bin Kamel Ariffin[4]
Faculty of Computer Science and Information Technology, University Putra Malaysia, Selangor, Malaysia[1, 2, 3]
Institute for Mathematical Research, University Putra Malaysia, Selangor, Malaysia[4]

***Abstract*—Although scalar multiplication is highly fundamental to elliptic curve cryptography (ECC), it is the most time-consuming operation. The performance of such scalar multiplication depends on the performance of its scalar recoding which can be measured in terms of the time and memory consumed, as well as its level of security. This paper focuses on the conversion of binary scalar key representation into {0, 1, 3}-NAF non-adjacent form. Thus, we propose an improved {0, 1, 3}-NAF lookup table and mathematical formula algorithm which improves the performance of {0, 1, 3}-NAF algorithm. This is achieved by reducing the number of rows from 15 rows to 6 rows, and reading two (instead of three) digits to produce one. Furthermore, the improved lookup table reduces the recoding time of the algorithm by over 60% with a significant reduction in memory consumption even with an increase in key size. Specifically, the improved lookup table reduces the memory consumption by as much as 75% for the big key, which shows its higher level of resilience to side channel attacks.**

Keywords*—Elliptic Curve Cryptosystem (ECC); scalar multiplication algorithm; {0, 1, 3}-NAF method; Non-Adjacent Form (NAF)

I. Introduction

Elliptic curves cryptosystem (ECC) was proposed by Neal Koblitz and Victor Miller independently in 1985 to design the public-key cryptographic system [1]. Similar to other public key cryptographic algorithms, elliptic curve cryptosystem deploys a public key and private key. The public key is used for encryption to provide data confidentiality during communication. ECC is implemented in smart card because of its smaller key size and less computational complexity relative to RSA cryptosystem [2]. This makes it attractive and suitable for such applications.

Scalar multiplication is a fundamental and time-consuming operation in ECC [3]. The scalar multiplication involves computing $Q = kP$ where k is an integer and P, Q are points on an elliptic curve. It is performed by repeating point addition/subtraction and point doubling operations. The representation of scalar k plays an important role in improving the performance of this operation. Hamming weight of scalar involves the number of the non-zero digits. As such, it determines the number of the required point addition/subtraction operation. Therefore, hamming weight is one of the performance factor for the scalar multiplication operation. Many researchers have tried to improve the performance of the scalar multiplication by representing k in other forms with minimal hamming weight [4]. However, these works does not improve the hamming weight for the {0,1,3}method, but improving the timing, memory consuming and security for the previous method since it is working on existing lookup table.

In literature, it is proven that reducing the Hamming weight of the scalar k can improve the performance of scalar multiplication [5],[6] and [7]. Additionally, the scalar k can be represented in base 2 or otherwise or by using combination of different bases. In base 2, k is in binary, NAF or w-NAF. In bases other than 2, k can be represented in r-NAF [8] or g-NAF[9]. Examples of combination of different bases include mixed ternary/binary[10], DBNS [11], [12], and mbNAF [13]. Various recoding algorithms used in the literature include complement recoding technique [14], hybrid complementary and 1's complement recoding technique [15].

In the aforementioned methods, the hamming weight and its effect on the performance of the scalar multiplication were well discussed. For example, width w-NAF is more efficient. However, it increases the w value [6], which implies more time and memory is consumed as it requires more operation during pre-computation. It is important to make a trade-off between the performances categories according to the target objective for implementation [16].

The contributions of this paper are as follows: The {0, 1, 3}-NAF method is introduced to convert the binary digit {0, 1} using a proposed lookup table or mathematical formula. The existing lookup table is of size 15 rows and 6 columns and contains special cases, which reads three digits during the recoding to produce one. In this paper, a new lookup table of size 6 rows and 5 columns is proposed to recode the scalar. The proposed lookup table reads two digits to produce one and contains no special cases. The proposed is better than the original in terms of time, memory and security. The remainder of this paper is organized as follows: Section 2 discusses the related work, while Section 3 introduces the {0, 1, 3}-NAF method. The proposed method and the performance analysis are presented in Section 4 and Section 5, respectively. Finally, conclusion and the future works are presented in Section 6.

II. Related Works

In literature, recoding algorithm is used to change the representation of k to another form without changing the magnitude of the scalar. There are two types of recoding algorithm [17]: left-to-right (L2R) and right-to-left (R2L). L2R recoding is done by scanning digit of k from the most significant bit (MSB) and the latter is by scanning digit k from Least Significant Bit (LSB). L2R recoding saves memory and

is mostly preferred for memory constrained devices [18]. It depends on the number of rows of the lookup table and number of required digits read while recoding.

However, the performance of recoding algorithms depends on the hardware system implementation and memory storage [16] and [19]. Efficient recoding must have recoding rules that are efficient, simple, and consumes less memory [20]. An optimal recoding strategy must provide a trade-off between high nonzero density and low memory consumption [6]. Selection of radix or digit set for a scalar must also satisfy the characteristics of the scalar multiplication algorithm or implementation technology. According to [21], proper selection of radix and digit set for the scalar can promote an increase of the frequency of useful digits such as zero and a reduction in the total number of nonzero digits to represent a number.

Reitwiesner (1960) proposed a R2L with non-adjacent form (NAF) recoding which converts a binary number {0,1} into NAF with digit {-1,0,1}-NAF [22] as shown in Algorithm 1. A non-adjacent form means that there is no consecutive non-zero digit in the scalar k. In {-1, 0, 1}-NAF recoding, a binary number of form $K = (K_{m-1},\dots,K_0)_2$ with $K_i \in \{0, 1\}$ converted into a canonical form $Y = (Y_m, Y_{m-1},\dots,Y_0)$ with $Y_m \in$ {-1, 0, -1} using Algorithm 1. The average hamming weight of NAF is $n/3$.

Algorithm 1: R2L NAF Recoding

Input: $X = (X_{m-1},..,X_0)_2$

Output: $Y = (Y_m, Y_{m-2},\dots,Y_0)_{NAF}$

1 $C_0 \leftarrow 0; X_{m+1} \leftarrow 0; Y_m \leftarrow 0$

2 For i from 0 to m do

3 $C_{i+1} \leftarrow \lfloor$(Ci + Xi + Xi-1)/2$\rfloor$

4 Yi $\leftarrow$ Ci + Yi + 2.Ci+1

5 Return Y= (Ym,Ym-2, ..., Y0)NAF

Example 1: Convert the binary number $k = (11111)_2$ into NAF method using the Algorithm 1.

Solution: $(11111)_2 = (10000 - 1)_{\text{NAF}}$

It is worthy of note that the hamming weight (number of non-zeroes) reduced from 5 into 2.

Joye and Yen [23] proposed an optimal L2R recoding algorithm for the binary number, The recoding however does not have NAF property as shown in Algorithm 2. They also use the lookup table to convert the binary to {-1, 0, 1} form as shown in Table I.

TABLE I. L2R SIGNED-DIGIT RECODING (X = 0 OR 1)

bi	X_i	X_{i-1}	X_{i-2}	b_{i-1}	Y_i
0	0	0	x	0	0
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	x	0	1
1	0	1	x	1	-1
1	1	0	0	0	-1
1	1	0	1	1	0
1	1	1	x	1	0

Algorithm 2: L2R Signed Digit Recoding

Input: $X = (X_{m-1},\dots,X_0)_2$

Output: $Y = (Y_m, Y_{m-1},\dots,Y_0)_2$

1 $b_m \leftarrow 0; X_m \leftarrow 0; X_{-1} \leftarrow 0; X_{-2} \leftarrow 0$

2 For i from m down to 0 do

3 $b_{i-1} \leftarrow \lfloor(b_m + X_{i-1} + X_{i-2})/2\rfloor$

4 $Y_i \leftarrow -2b_i + X_i + b_{i-1}$

5 Return Y

Note that in Example 1, using Algorithm 1, 2 or the lookup Table I will give the same result (10000-1), but there is a considered difference in the time and memory consumed.

Rezai et.al [24] proposed an L2R recoding algorithm while deploying Markov chain to measure the hamming weight. They identified that their L2R method has a hamming weight of 3n/13.

III. EXISTING{0, 1, 3}-NAF RECODING ALGORITHM

Yasin [25] proposed a recoding algorithm based on the idea from Reitwiesner's (R2L) [22] and Joye and Yen (L2R) [23]. The algorithm is also an L2R recoding and it converts a binary into a non-adjacent form in base 2 with digit {0, 1, 3} using Table II. The author has been proven that the representation follows the non-adjacent form (NAF) property. Algorithm 3 is used to convert the binary into {0, 1, 3}-NAF

Table II is a lookup table used together with Algorithm 3. Table II consists of 15 rows, and the algorithm starts with scanning three digits L2R. There are also special cases for certain conditions.

Algorithm 3: L2R {0,1,3}-NAF Recoding

Input: $r = (r_{m-1},\dots,r_0)_2$

Output: $r' = (r'_m, r'_{m-1},\dots,r'_0)_{\{0,1,3\}-NAF}$

1 $b_m \leftarrow 0; r_m \leftarrow 0; r_{-1} \leftarrow 0; r_{-2} \leftarrow 0; r'_m \leftarrow 0$

2 For i from m-1 downto 0 do

3 scan two digits r from MSB i.e. r_i and r_{i+1}

4 Compute $b_i \leftarrow \lfloor(b_{i+1} + r_i + r_{i-1})/2\rfloor$

5 Compare $(b_{i+1},r_{i+1},r_i,r_{i-1},b_i)$ with values from lookup table row by row:

If $[(b_{i+1},r_{i+1},r_i,r_{i-1},b_i) \equiv \{$(row1) or (row3) or (row5) or (row6) or (row8) or (row9) or (row10) or (row13) or (row15)$\}]$ then $r'_i = 0$

6 If $[(b_{i+1},r_{i+1},r_i,r_{i-1},b_i) \equiv \{$(row2) or (row4) or (row7)$\}]$ then $r'_i = 1$

7 if $[(b_{i+1},r_{i+1},r_i,r_{i-1},b_i) \equiv \{$(row11) or (row12) or (row14)$\}]$ then $r'_i = 3$

8 return $(r'_m, r'_{m-1},\dots,r'_0)$

Example 2: Convert the number (1101101101) from binary into {0,1,3}-NAF method.

Solution: applying the lookup Table II or Algorithm 3 by reading 3 digits from L2R will give the result $(0300300301)_{\{0,1,3\}\text{-NAF}}$, which reduce the hamming weight from 7 into 4.

TABLE II. L2R {0,1,3}-NAF RECODING (X= 0 OR 1)

No	b_{i+1}	r_{i-1}	r_i	r_{i+1}	Special Case	r'_i	b_i
1	0	0	0	x		0	0
2	0	0	1	0		1	0
3	0	0	1	1	if consecutive #1's is even	0	1
4	0	0	1	1	if consecutive #1's is odd	1	1
5	0	1	0	x		0	0
6	0	1	1	1	If r'_{i+1}= 1 OR 3	0	1
7	1	0	1	0		1	1
8	1	0	1	1		1	1
9	1	1	0	0		0	0
10	1	1	0	1	If r'_{i+1}=1 OR 3	0	1
11	1	1	0	1	If r'_{i+1}=0	3	1
12	1	1	1	0	If r'_{i+1}=0	3	1
13	1	1	1	0	If r'_{i+1}=1	0	1
14	1	1	1	1	If r'_{i+1}=0	3	1
15	1	1	1	1	If r'_{i+1}=1 OR 3	0	1

IV. PROPOSED ALGORITHM

So we proposed Table III which converts a binary into {0,1,3}-NAF with high performance. Table III consists of 6 rows and it is used together with Algorithm 4. The algorithm starts with scanning two digits from R2L

Table III is an improved version of Table II. The table size is reduced from 15 rows to 6 rows. Algorithm 4 is used together with Table III to converts a binary into {0,1,3}-NAF.

Algorithm 4: Improved R2L {0,1,3}-NAF Recoding

Input: $X = (X_{m-1}, \dots, X_0)_2$

Output: $Y = (Y_m, Y_{m-1}, \dots, Y_0)_{\{0,1,3\}-NAF}$

1 $C_0 \leftarrow 0$; $X_m \leftarrow 0$
2 For i from 0 to m do
3 Scan two digit X from LSB (X_{i+1}, X_i)
4 Use lookup table, find Y_i that match (C_i, X_{i+1}, X_i)
5 Use lookup table, find C_{i+1}
6 Return Y

In Algorithm 3, line 4 computes b_i for each iteration. Also, line 5 do comparison of function $f(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (r'_i, b_i)$ with the values in a row in the lookup table. In Algorithm 4, comparison of function$f(C_i, X_{i+1}, X_i) = (Y_i, C_{i+1})$ is done in line 4. It is worthy of note that number of comparison is minimal than the one in Algorithm 3, since size of lookup table for Algorithm 3 is bigger than the size of lookup table used in Algorithm 4.

TABLE III. IMPROVED LOOKUP TABLE OF {0,1, 3}-NAF RECODING

No	C_i	X_{i+1}	X_i	Y_i	C_{i+1}
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	1	1	0
4	0	1	1	3	1
5	1	0	1	0	0
6	1	1	1	0	0

In Table III, a new mathematical formula can be introduced to recode the digit without using the lookup table as presented in Algorithm 5.

Algorithm 5: Improved {0,1,3} −NAF Recoding R2L.

Input: $x = (x_{m-1}, \dots, x_0)_2$

Output: $y = (y_m, y_{m-1}, \dots, y_0)_{\{0,1,3\}-NAF}$

1. $C_0 \rightarrow 0, \quad x_m \rightarrow 0$
2. For $i = 0$ *to* m do
3. $y_i = x_i * x_{i+1} + (x_{i+1} - C_i) - C_i$
4. $C_{i+1} = (y_i + x_i + C_i)/2$
5. return $y = (y_m, y_{m-1}, \dots, y_0)_{\{0,1,3\}-NAF}$

In the proposed Algorithm 5, the value of y_i can be calculated using the values of (x_{i+1}, x_i, C_i) mathematically as in step 3, while the value of C_{i+1} can be computed using the values of (y_i, x_i, C_i) mathematically as in step 4.

In general, lookup table is more efficient in terms of time and memory since lookup table contains no mathematical operations such as multiplication and division as in Algorithm 5.

V. PERFORMANCE ANALYSIS

In terms of performance, we will compare between the proposed lookup table and the original lookup table [25] in terms of response time, memory usage and security. We implemented the two tables in JAVA (NetBeans IDE 8.0.2). The conversion from binary expansion to a new {0,1,3}-NAF representation is run successfully.

Table IV shows the time in seconds for different bit sizes of 24, 28, 32, and 36 bits. As the bit sizes decreases, the level of reduction in percentage is also decreases.

It is clear that the proposed lookup table is faster than current lookup table. The conversion processes also consume less time. Fig. 1 shows the reduction time between the two lookup tables.

Fig. 1 shows that our proposed lookup table is more efficient for larger bit size due to its higher reduction percentage.

In terms of the memory performance, the proposed algorithm consumes less memory with higher percentage for large bit key sizes as shown in Table V and Fig. 2.

In Fig. 1 and Fig. 2, the performance achied due the small lookup table size. While recoding, two digits only need to scan so as to produce one digits. Also this can be more efficient with key of big size.

TABLE IV. CONVERSION TIME FROM BINARY TO {0,1,3}-NAF FOR L2R AND MODIFIED R2L {0,1,3}-NAF ALGORITHMS

Size of bits	L2R {0,1,3}-NAF Recoding(Seconds)	Proposed R2L {0,1,3}-NAF Recoding (Seconds)	Reduction Percentage
36	123650	49918	60%
32	6839	2897	58%
28	389	173	56%
24	22	11	50%

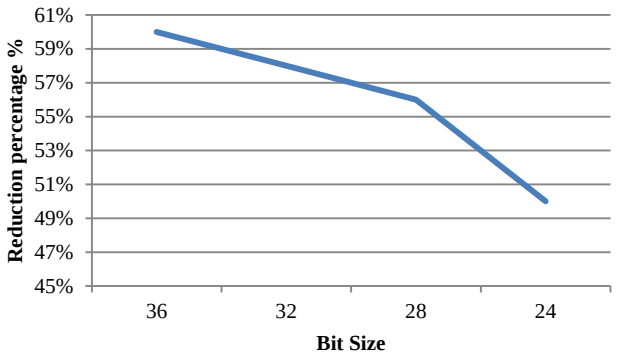

Fig. 1. Reduction Percentage of Time Related to Bit Size for the Proposed Lookup Table.

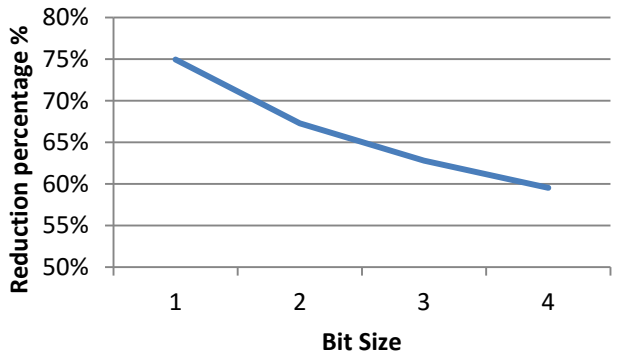

Fig. 2. Reduction Percentage of Time Related to Bit Size for the Proposed Lookup Table.

TABLE V. CONVERSION MEMORY BY KBYTES FROM BINARY TO {0,1,3}-NAF FOR L2R AND MODIFIED R2L {0,1,3}-NAF ALGORITHMS

Size of bits	L2R {0,1,3}-NAF Recoding (Kbytes)	Proposed R2L {0,1,3}-NAF Recoding (Kbytes)	Reduction Percentage
36	75416	18897	75%
32	41837	13691	67%
28	30937	11511	63%
24	25112	10164	60%

So, it is clear that the proposed is better than the original {0, 1, 3}-NAF algorithm.

To achieve better ECC security, a larger bit size is desired which makes the proposed lookup table more efficient in terms of time and the memory usage. It is thus more suitable for implementation in ECC.

In term of security, the original lookup table is vulnerable to side channel attack such as simple power attack SPA and timing attack TA due to its non-constant time execution [26]. The original lookup has two exceptional cases to the count number of 1's in line 4 & 5 in Table II. While using the lookup table, if there is a consecutive 1's is consumes more memory and time while recoding the original lookup table which makes it vulnerable to attacks. For instance, a hacker can guess that there is a consequent 1's at a part of the key [27].

VI. CONCLUSION AND FUTURE WORKS

In this paper, a new lookup table and mathematical formula have been proposed to improve the {0, 1, 3}-NAF method. The proposed method shows improvement in terms of time, memory and security aspects compared to the original {0, 1, 3}-NAF method, since it reduces the lookup table size from 15 rows into 6, and reads two digits during the recoding to produce one instead of three. Time and memory are reduced while recoding execution with a percentage up to 60% and 75% respectively. The performance of the proposed lookup is more efficient while key size is bigger.

We suggest that this scalar recoding is applied in scalar multiplication either using Montgomery Ladder to achieve better security or using τNAF with Koblitz curves for higher efficiency. The digit 3 can be precomputed using different coordinates such as projective and affine over different curves such as binary, Edward and prime curves.

ACKNOWLEDGMENT

This work was supported by Ministry of Higher Education under FRGS Grant no. 5524822.

REFERENCES

[1] K. E. Abdullah and N. H. M. Ali, "Security Improvement in Elliptic Curve Cryptography," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 5, pp. 122–131, 2018.

[2] Z. U. A. Khan and M. Benaissa, "High Speed and Low Latency ECC Processor Implementation over GF (2 m) on FPGA," IEEE Trans. Very Large Scale Integr. Syst., vol. 25, no. 1, p. 165–176., 2017.

[3] N. Thangarasu and A. A. L. Selvakumar, "Improved elliptical curve cryptography and Abelian group theory to resolve linear system problem in sensor-cloud cluster computing," Cluster Comput., pp. 1–10, 2018.

[4] M. M. Ahmad, S. M. Yasin, R. Mahmod, and M. A. Mohamed, "X-Tract Recoding Algorithm for Minimal Hamming Weight Digit Set Conversion," J. Theor. Appl. Inf. Technol., vol. 75, no. 1, pp. 109–114, 2015.

[5] O. Ugus, D. Westhoff, R. Laue, A. Shoufan, and S. A. Huss, "Optimized Implementation of Elliptic Curve Based Additive Homomorphic Encryption for Wireless Sensor Networks," arXiv Prepr. arXiv0903.3900., 2009.

[6] K. Okeya and T. Takagi, "The Width- w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications," pp. 328–343, 2003.

[7] A. Rezai and P. Keshavarzi, "CCS Representation : A new non-adjacent form and its application in ECC," J. Basic Appl. Sci. Res., vol. 2, no. 5, pp. 4577–4586, 2016.

[8] T. Takagi, S. Yen, and B. Wu, "Radix- r Non-adjacent Form," Springer-Verlag Berlin Heidelb., pp. 99–100, 2004.

[9] M. Joye and S. Yen, "New Minimal Modified Radix- r Representation with Applications to Smart Cards," in International Workshop on Public Key Cryptography, 2002, pp. 375–383.

[10] M. Joye, "Trading Inversions for Multiplications in Elliptic," Des. codes Cryptogr., pp. 189–206, 2006.

[11] V. Dimitrov, L. Imbert, and P. K. Mishra, "The double-base number system and its application to elliptic curve cryptography," Math. Comput., vol. 77, no. 262, pp. 1075–1104, 2008.

[12] C. Doche and L. Habsieger, "A Tree-Based Approach for Computing Double-Base Chains A Tree-Based Approach for Computing Double-Base Chains," Australas. Conf. Inf. Secur. Priv. (pp. 433-446). Springer, Berlin, Heidelberg., no. June 2008, 2016.

[13] P. Longa and C. Gebotys, "Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Multiplication . Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Mult," IACR Cryptol. ePrint Arch., no. February, 2015.

[14] P. Balasubramaniam and E. Karthikeyan, “Elliptic curve scalar multiplication algorithm using complementary recoding,” Appl. Math. Comput., vol. 190, pp. 51–56, 2007.

[15] X. Huang, P. G. Shah, and D. Sharma, “Minimizing Hamming Weight Based on 1 ’ s Complement of Binary Numbers Over GF (2 m),” in In Advanced Communication Technology (ICACT), 2010 The 12th International Conference on (Vol. 2, pp. 1226-1230). IEEE., 2010, pp. 1226–1230.

[16] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, “Comparison of ECC and RSA algorithm in resource constrained devices,” 2013 Int. Conf. IT Converg. Secur. ICITCS 2013, pp. 0–2, 2013.

[17] H. Cohen, G. Frey, and R. Avanzi, Handbook of Elliptic and Hyperelliptic Curve Cryptography. 2006.

[18] M. Khabbazian, T. A. Gulliver, S. Member, and V. K. Bhargava, “A New Minimal Average Weight Representation for Left-to-Right Point Multiplication Methods,” IEEE Trans. Comput. 54(11), 1454-1459., pp. 1–7, 2005.

[19] W. K. A. Abdulraheem, “Comparative Analysis of the Performance for Cloud Computing Hypervisors with Encrypted Algorithms,” 2014.

[20] D. F. Aranha and K. Karabina, “Efficient Software Implementation of Laddering Algorithms Over Binary Elliptic Curves Efficient software implementation of laddering algorithms over binary elliptic curves,” Int. Conf. Secur. Privacy, Appl. Cryptogr. Eng. (pp. 74-92). Springer, Cham, no. December, 2017.

[21] E. Guerrini, L. Imbert, and T. Winterhalter, “Randomized Mixed-Radix Scalar Multiplication,” IEEE Trans. Comput., vol. 67, no. 3, pp. 418–431, 2018.

[22] G. W. Reitwiesnert, “The Determination of Carry Propagation Length for Binary Addition *,” IRE Trans. Electron. Comput. (1), 35-38., vol. 0, pp. 35–38, 1960.

[23] M. Joye and S. Yen, “Optimal Left-to-right Binary Signed-Digit Recoding,” vol. 49, no. 7, pp. 1–8, 2000.

[24] A. Rezai and P. Keshavarzi, “A New Left-to-Right Scalar Multiplication Algorithm Using a New Recoding A New Left-to-Right Scalar Multiplication Algorithm Using a New Recoding Technique,” Int. J. Secur. its Appl., vol. 8, no. 3, pp. 31–38, 2015.

[25] S. M. Yasin, “New signed-digit {0, 1, 3}-NAF scalar multiplication algorithm for elliptic curve over binary field.,” 2011.

[26] N. Tuveri, S. ul Hassan, C. P. Garcia, and B. B. Brumley, “Side-Channel Analysis of SM2: A Late-Stage Featurization Case Study,” in Proceedings of the 34th Annual Computer Security Applications Conference on - ACSAC ’18, 2018, pp. 147–160.

[27] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, “State-of-the-art of secure ECC implementations : a survey on known side-channel attacks and countermeasures,” in In Hardware-Oriented Security and Trust (HOST), IEEE International Symposium on, 2010, pp. 76–87.