Available online at http://scik.org

J. Math. Comput. Sci. 8 (2018), No. 1, 130-152

https://doi.org/10.28919/jmcs/3550

ISSN: 1927-5307

A NEW ALGORITHM FOR SIGNED BINARY REPRESENTATION AND APPLICATION IN MOBILE PHONES

N. M. G. AL-SAIDI*, M. A. MAGAMISS, S. F. IBRAHEEM AND A. KH. FARAJ

Department of Applied Sciences, University of Technology, Baghdad, Iraq

Abstract: Cryptographic protocols become a requirement for many software applications and communication. Therefore, understanding of such protocols is more essential in order to improve them mathematically and in software implementations. Elliptic curve cryptography is one of these protocol, which are highly used nowadays especially in mobile devices due to its small key size and good security level. Despite of these advantages, it is considered as slower than other cryptosystems with the same key size and this revert to the most important operation the scalar multiplication. In this work, we aimed to reduce the computation of this operation by proposing an algorithm considered as faster in comparing to other efficient methods. This algorithm is implemented in mobile device to prove its efficiency.

Keywords: elliptic curve scalar multiplication (ECSM); binary method; signed binary representation.

2010 AMS Subject Classification: 68Q25.

1. Introduction

For the computational environment with limited memory such as, mobile devices, wireless sensor nod, etc. The cryptosystem is used as a center part to ensure the security of such wireless

*Corresponding author

E-mail address: nadiamg08@gmail.com

Received October 26, 2017

communication Elliptic curve cryptography was proved to be more secure than many other cryptosystems such as, RSA AlGamal, when they used the same key size. The most important operation in elliptic curve is the scalar multiplication, it is the operation to compute $F = rG$, where F, G are points on the elliptic curve and r is any positive integer. The use of binary expansion to represent r such that $r=\sum_{s=0}^{n-1} 2^s$ is considered as the most common way. Here, r_s is an element of a finite digit set D_k. In literature, several methods are proposed to achieve a perfect representation that helps to enhance the performance of ECC and speeding up the scalar multiplication operation. Some of them are based on reducing of the Hamming weight, such as, the recording method []. Some other are based on window representation, which are considered as a generalization of the recoding methods, for example, nonadjacent form (NAF) and Window NAF [9–11], mutual opposite form (MOF) and Window MOF [12], and Window fractional methods [13,14].

The complexity of the scalar multiplication operation is also improved by introducing fast composite methods [2,15] as well as, using of coordinates systems instead of affine coordinates [16,17]. It also speeds up through precomputations techniques [18–22]. Some other different approach was through utilizing of customized hardware [23–25]. The representation of r is not only limited to binary expansion, it was also represented in large radix. Another method for representing r is known as double base number system (DBNS) [], it is used to improve the performance of scalar multiplication. Combining software together is also helps to improve the scalar operation of EC, this was obvious from the efficiency of EC single scalar multiplication when combining of the optimal methods to solve EC problems. The operations on EC are two types, one is basic and the other is composite as shown in Table 1 [1,5–8].

EC operation type	Operation name	Math representation	Denoted by
Basic	Point addition	Q+P	ADD
	Doubling	2P	DBL
composite	Double and add	2P+Q	DBL and ADD
	Tripling	3P	TRP
	Triple and add	3P+Q	TADD
	Quadrupling	4P	QPL
	Quadruple and add	4P+Q	QADD

In this paper, we aimed to enhance the EC scalar multiplication by proposing of an efficient algorithm for this purpose, and compare it with other binary representation methods to prove its efficiency.

2. Preliminaries

This section presents an overview of the material used in this work, for more details and some background in cryptography; we refer the reader to see [22-24].

Definition 2.1: Let *F* be any field. Based on Weierstras's equation, the elliptic curve *E* over *F* is defined by:

$$+a_1\ xy+a_3\ y=x^3+a_2x^2\ +a_4x+a_6\qquad y^2 \tag{1}$$

Where, the coefficients $a_i \in F$ and the discriminant of *E is* $\Delta \neq 0$.

The discriminant Δ of *E* given in (1) is defined by;

$$=-\ d_2{}^2\ d_8\ -8d_2{}^3\ -27\ d_2{}^2\ +9\ d_2d_4d_6\Delta \tag{2}$$

where,

$$d_2\ =\ a_1{}^2\ +4\ a_2$$

$$d_4 = 2a_4 + a_1 a_3$$

$$d_6 = a_3^2 + 4 a_6$$

$$d_8 = a_1^2 a_8 + 4 a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2$$

Let L be an arbitrary extension of F, the set of all points on E is given by:

$$E(L) = \{(x, y) \in L \times L: y^2 + a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{0_\infty\}.$$

where, 0_∞ is called the point at infinity.

The expression of the Weierstrass equation given in (1) for elliptic curve over the prime field F_p can be simplified as follows:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \qquad (3)$$

where $a, b \in R$ and $\Delta = 4a^3 + 27b^2 \neq 0$ [19].

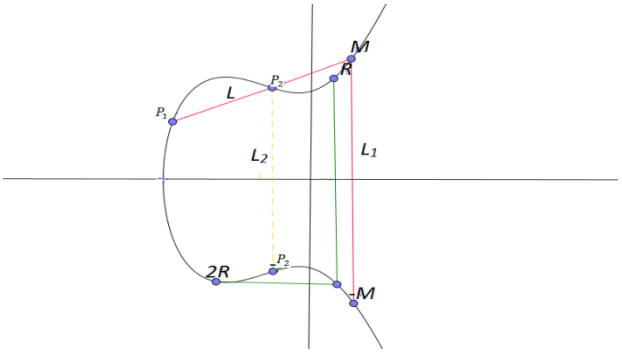

Figure 1: Elliptic Curve Addition

2.2 Addition of two points over *EC* algebraically

If we have two different points $P_1 = (x_1, y_1)$, and $P_2 = (x_2, y_2) \in E(F_p)$. Then its addition operation defined as $P_1 + P_2 = (x_1, y_1) + (x_2, y_2) = (x_3, y_3) = P_3 \in E(F_p)$ is computed as follows:

$$P_1 + P_2 = \begin{cases} O\infty \ \ if\ x_1 = x_2\ y_1 = -y_2 \\ (x_3, y_3) \quad if\ otherwise \end{cases} \tag{4}$$

where,

$$x_3 = \lambda^2 - x_1 - x_2 \tag{5}$$

$$= \lambda\,(x_1 - x_3) - y_1\ y_3 \tag{6}$$

and λ is the slope of the line L that connecting the points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ such that:

$$\lambda = [(y_2 - y_1)/(x_2 - x_1)]\ mod\ p \tag{7}$$

If we have two similar points $P_1 = (x_1, y_1) \in E(F_p)$ and itself, then its addition operation with itself is defined as $P_1 + P_1 = (x_1, y_1) + (x_1, y_1) = (x_3, y_3) = 2P_1 \in E(F_p)$. It is computed as follows:

$$x_3 = \lambda^2 - x_1 - x_1 \tag{8}$$

$$= \lambda\,(x_1 - x_3) - y_1 y_3 \tag{9}$$

where

$$= \frac{3x_1^2 + a}{2y_1}\ .\lambda \tag{10}$$

The inverse of the point $P_1 = (x_1, y_1) \in E(F_p)$ is defined as $-P_1 = (x_1, -y_1) \in E(F_p)$.

Definition 2.3: The hamming weight of the scalar *k* is the number of non-zero bits in the signed binary representation, it is denoted by *h* (*k*).

Definition 2.3: A signed binary representation of a scalar *k* with the base *b* is denoted by $(k)_b$, such that *k*=$k_{l-1}k_{l-2}\ \dots\ k_0$ with $|k_i| < b$ for i =0, 1, 2, …, $l-1$, and *k*= $\sum_0^{l-1} k_i\ b^i$.

Definition 2.4: The length of a singed binary representation is denoted by $l(k)$, which represents the number of bits in this representation.

3. The Previous work

The binary representation length of the scalar *k* and its number of 1's are controlling the efficiency and the cost of the *ECSM*. There has been extensive research toward achieving of an efficient representation for *k*. From literature, some of the existing well-known methods are presenting in this section.

3.1 Binary method:

The simplest method to represent an integer in (0, 1) bits is called the binary representation. It is defined as $(k_{l-1}\ k_{l-2}\ \dots\ k_0)_2$ where $k_i \in (0, 1)$, i =0, 1, 2 …, $l-1$. Thus any integer can be expressed as *k* = $\sum_0^{l-1} k_i\ 2^i$. Let P_1 be a point on an elliptic curve *E*, then the scalar multiplication defined as $kP_1 = \sum_0^{l-1} k_i\ 2^i\ P_1 = P_2$, where P_2 is another point on *EC*. The computation of the scalar multiplication for binary method is showing in Algorithm 1:

Algorithm 1: Binary Method for Elliptic Curve Scalar Multiplication

Input:k=$(k_{l-1},\ k_{l-2},\ldots,\ k_0)_2,\ P_1 \in E(F_p)$

Output: $Q = kP$

1.$Q = P$

2. for $i = l -$ 1 down to 0 do

2.1. If $k_i = 1$ then $Q = Q + P$

2.2. $Q = 2P$

3. Return Q.

The bits of *k* can be scans either from left to right or right to left in this method. The number of operations for execution this algorithm is determined the running time. The elliptic curve adding (*ECADD*) and doubling (*ECDBL*) operations are performed if $k_i = 1$, otherwise, only elliptic curve doubling (*ECDBL*) operation is performed. The average of getting the number of ones in the binary expansion of the scalar *k* is $l -$ 1.

3.2 Mutual Opposite Form (*MOF*)

In 2004, Mutual opposite form (*MOF*) proposed by Okeya et al. [20], is introduced to reduce the hamming weight by converting the scalar to signed binary representation. The integer with length l of the binary representation by *MOF* is at most of the length $(l + 1)$ and has a unique representation. *MOF* satisfies the following properties:

a- The signs are different for every two adjacent non zero bits

b- The values 1and -1 are the first and the last non-zero bit in *MOF* representation, respectively.

c- Every positive integer can be expressed by a unique *MOF*.

d- Sliding window method with width *w* can be executed in *MOF*, but in other methods such as binary complementary, it cannot be executed with the sliding.

To represent the integer *k* in *MOF* we follow:

2k =	k_{l-1}	k_{l-2}	⋯	k_{r-1}	⋯	k_1	k_0	
Θk =		k_{l-1}	⋯	k_r	⋯	k_2	k_1	k_0
MOF(*k*)	k_{l-1}	k_{l-2}	⋯	k_{r-1} -	⋯	k_1	k_0	$-k_0$
		$-k_{l-1}$		k_r		$-k_2$	$-k_1$	

Where, Θ refers to bitwise subtraction. This calculation can be computed in two directions; left to right (*L2R*) or right to left (*R2L*), as presented in algorithm 2 and 3 respectively.

The *L2R MOF* representation is presented in Algorithm 2:

Algorithm 2: *L2R MOF* to find *ECSM*

Input: $k = (d_l\ d_{l-1} \dots d_1 d_0)_{MOF(k)}, P \in E(F_p)$

Output: $Q = kP$

1. Set Q=0
2. For $i = l -$ 1 down to 0 do
3. $Q = 2P$
4. if $d_i = 1$ then $Q = Q + P$
5. if $d_i = -1$ then $Q = Q - P$
6. Return Q.

Example 1: Let *k*=7, Table 2 presents the steps for converting the scalar *k* into *L2R MOF* representation.

Table 2: Binary representation and *MOF* for *k*=7

Iteration value (*i*)	Binary representation	*MOF*(7)
3		1
2	1	10
1	11	100
0	111	$100\bar{1}$

3.3 Complementary Recoding Method (*CRM*)

Firstly, we find the binary representation of a scalar *k*= $(k_{l-1}\ k_{l-2}\ \dots\ k_0)_2$. Utilizing complementary method [45], to change the binary string into a signed binary string, the procedure is given in the following:

$$k = \sum_0^{l-1} k_i\ 2^i = (1000\dots0)_{(l+1)bits} - \bar{k} - 1 \quad (4)$$

where $\bar{k} = \bar{k}_{i-1}\bar{k}_{i-2}\dots\bar{k}_0$ and $\bar{k}_i$= 1 when k_i = 0 for i=0, 1, 2… i -1.

Algorithm 3: Complementary Recoding Method (CRM)

Input: $k = (k_{l-1}\dots k_1 k_0)_2$

Output: $k = (d_l\ d_{l-1}\dots d_1 d_0)_{CRM(k)}$

1. v= 2^n
2. r=$\bar{k}$
3. *CRM* (*k*)= v-r-1
4. Return $(d_l\ d_{l-1}\dots d_1 d_0)_{CRM(k)}$.

Example 2: Let us take an integer $k = (7967)_{10} = (1111100011111)_2$ is the binary representation with 13 bits.

The complement of the binary representation $\bar{k} = (0000011100000)_2$.

Thus, using equation (4) to get the signed binary representation for a scalar $k = (10000000000000)_2 - (0000011100000)_2 - 1$

$= (100000\text{-}1\text{-}1\text{-}10000\text{-}1) = 2^{13} - 2^7 - 2^6 - 2^5 - 2^0$.

The binary representation has 10 hamming weights and the complementary recoding has 5 hamming weights which saves 5 point addition operation in scalar multiplication.

3.4 Non Adjacent Form (*NAF*)

A non-adjacent form (*NAF*) proposed in 1951, by Booth [17]. *NAF* has an optimal traditional a signed binary representation according to the minimum hamming weight as compared as other traditional representations. A scalar *k* can be expressed a *NAF* representation as $k=\sum_0^{l-1} k_i \ 2^i$, where $k_i \in \{0, 1, -1\}$, for every $i = 0, 1, 2 \ldots, l-1$. A *NAF* representation has an important property which is not existing two non-zero bits are adjacent respectively. Furthermore, it is at most greater than binary representation by one digit. Every integer has a unique *NAF* representation and exist proved by Gordon [47]. Algorithm 3 is used to express the integer *k* as assigned binary representation.

Algorithm 3: *NAF* representation of the scalar k

Input: Appositive integer k

Output: $s = (k_{l-1} \ldots k_1 k_0)_{NAF(k)}$

Set $i = 1;\ c = k$

While $c > 0$

If c is odd

$s(i)$=2−(c *mod* 4)

$$c = c - s(i)$$

Else

$$s(i) = 0$$

End if

$$c = \frac{c}{2};\quad i = i + 1$$

End while

Return s.

The average number of hamming weight (non-zero bits) in this representation is equal to its length divided by three ($\frac{l}{3}$).

Example 3: Let us take k = 35 to describe the application of algorithm 5 as shown in Table 3.

Table 3: The result of *NAF* (31)

I	c	$s(i)$	S
1	31	$\bar{1}$	$\bar{1}$
	32		
2	16	0	$0\bar{1}$
3	8	0	$00\bar{1}$
4	4	0	$000\bar{1}$
5	2	0	$0000\bar{1}$
6	1	1	$10000\bar{1}$

The previous table shows the result for *NAF* (31) = ($10000\bar{1}$). Now, computing the binary representation for 31 =$(11111)_2$, so the hamming weight in the binary representation are 5 reduced to 2 in a signed binary representation *NAF*.

Remarks

1. A signed binary representation *NAF* (*k*) for a scalar k has the least hamming weight of any other signed binary representation of k, except if the binary representation of k already has. For examples

   a. 42 = $(101010)_2$

   = $(101010)_{NAF(42)}$

   b. 43 = $(101011)_2$

   = $(10\bar{1}0\bar{1}0\bar{1})_{NAF(43)}$

2. The scalar k is $\frac{2^l}{3} < k < \frac{2^{l+1}}{3}$ if l (NAF (k)) = l.

3.5 Direct Recoding Method (*DRM*)

A new procedure is called Direct Recoding Method [20] for converting the scalar k into a signed binary representation with lowest time compared with traditional methods proposed by Pathak and Sanghi in 2010. In this procedure the output is same as a traditional signed binary representation complementary recoding and *NAF, MOF*. The procedure is below:

$$2^s < k < 2^{s+1} \tag{3.1}$$

$$k = (2^{s+1})_2 - (2^{s+1} - k)_2 \tag{3.2}$$

Bitwise-subtraction used in computation of procedure (3.2) by converting the scalar into a signed binary representation. This method may cause increase by the hamming weight in many numbers as in the following example3.4:

$$2^{3+1} > (9) > 2^3$$

$$2^4 > (9) > 2^3$$

$DRM(9) = (2^4)_2 - (2^4 - 9)_2 = (10000)_2 - (111)_2 = (10\bar{1}\bar{1}\bar{1})_{DRM(9)}$ so the hamming weight in this example are 4 but in binary are 2.

4. The Proposed Algorithm

In this section, a new signed binary representation of integers going to present. We introduce the following special algorithm of signed binary representation in order to achieve a unique representation. After implementing this algorithm for every integer, we found that, its output representation is same as MOF representation.

Algorithm 4: Assigned binary representation of a scalar *k*

Input: A scalar $(k)_{10}$ is appositive integer.

Output: *k*= $\left(d_l\ d_{l-1} \ldots d_1 d_0\right)_{MOF(k)}$.

1. Set *i=1, b1=0;*
2. While *(c>0);*

t=c/2

*b2=c - 2*t*

a (*i*)*=b1-b2*

b1=b2

c=t

i=i+1;

End

a (*i*)*=b1*

3. Return $\left(d_l\ d_{l-1} \ldots d_1 d_0\right)_{MOF(k)}$.

Example 4.1: Initializing the above algorithm for computation the integer *k*=127 to generate the assigned binary representation *k*= (1 0 0 0 0 0 0 -1). The expression of this integer *k*=127 in binary representation is *k*= (1 1 1 1 1 1 1). The number of the hamming weight that generating by the proposed algorithm is reduced to 2, compared with the binary representation that has 7 hamming weight.

4.1 Complexity of the Proposed Algorithm

All traditional algorithms, such as *MOF, CRM, DRM* and *NAF* are compared with our proposed algorithm according to the run time for generating the assigned binary representation. Using numbers of the different sizes (bits) are listed in Table 4 to show the run time in seconds.

Table 4: The comparison of run time of *NAF, MOF, CRM* and *DRM*.

Key size bits	$ALgo^{MOF}$	$ALgo^{prop}$	$ALgo^{DRM}$	$ALgo^{CRM}$	$ALgo^{NAF}$
52	0.011138	0.001077	0.001917	0.017754	0.001579
128	0.057325	0.001415	0.003750	0.026085	0.001998
256	0.166952	0.001894	0.004502	0.031501	0.002555
512	0.846201	0.004349	0.012593	0.052072	0.004959
630	1.220566	0.004737	0.013687	0.064883	0.005619
886	1.768649	0.005301	0.014662	0.120793	0.006170
1022	2.203799	0.062964	0.019098	0.135753	0.085153

- $ALgo^{MOF}$ refers to the *MOF* algorithm
- $ALgo^{prop}$ refers to the proposed algorithm
- $ALgo^{DRM}$ refers to the *DRM* algorithm
- $ALgo^{CRM}$ refers to the *CRM* algorithm
- $ALgo^{NAF}$ refers to the *NAF* algorithm

Based on our analysis which is shown in the previous Table 4, the running time computation in the proposed algorithm to generate the assigned binary representation is the least in comparison with the others methods as shown in Figure 3.

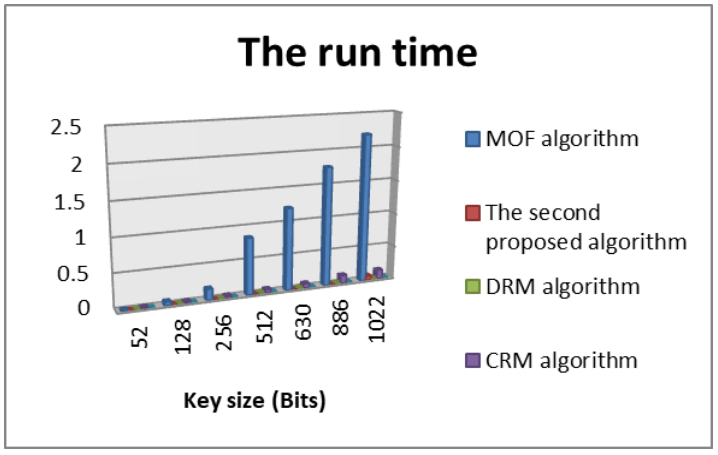

Figure 2: Time ratio of the second proposed algorithm with other algorithms (*NAF, MOF, CRM* and *DRM*).

4.1 Implementation of the Proposed ECC in Mobile Phones with Android *OS*

The *ECC* is an efficient cryptosystem with a simple mathematical conclusion and small key size in comparison to other public key cryptosystem. These facilities granted it advantages to be used for securing data in small devices like mobile phones. The proposed *ECC* that implemented in the previous section is applied on mobile phone with Android Operating System and using Java language programming.

4.2 Implementation and Analysis

In this section, the screenshots of the results obtained from the program are depicted. This program implemented by visual basic.net 2013. Due to successful run of the program, the data can be securely transmitted from one device to another in a wireless transfer, such as Bluetooth, Wi-Fi etc. The program is run on windows platform. This interface is designed to check the number whether it is a prime or not.

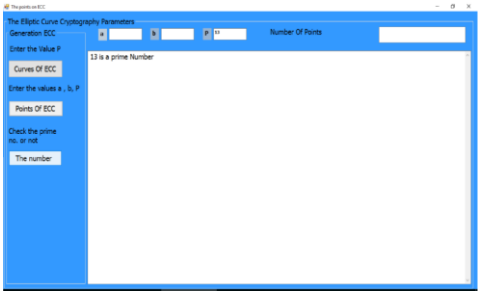

Figure 3: Primary Check.

The second interface is designed to compute all elliptic curves over F_p by given the value of the prime number, which is obtained from first menu as sown in Figure 4. Each output consists of two values (a, b) which are the coefficients of the equation of the elliptic curve. The first value considered as a and the second value considered as b as given in the equation (2.3).

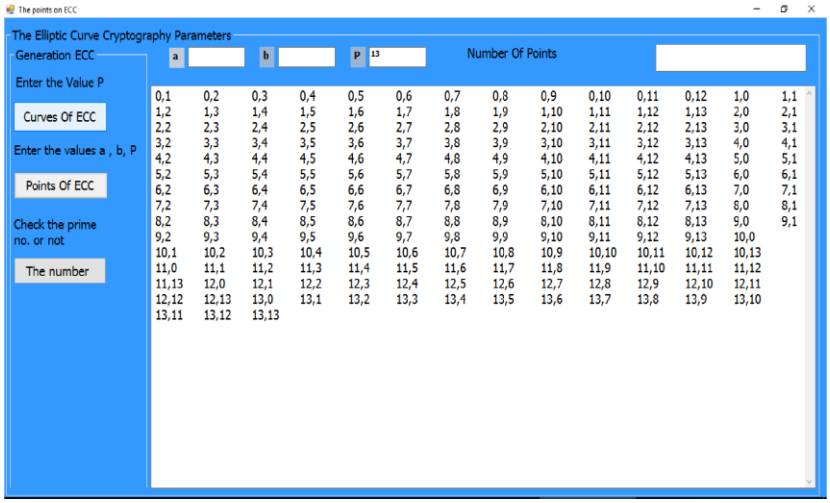

Figure 4: All Possible *EC* for Given Prime *P*.

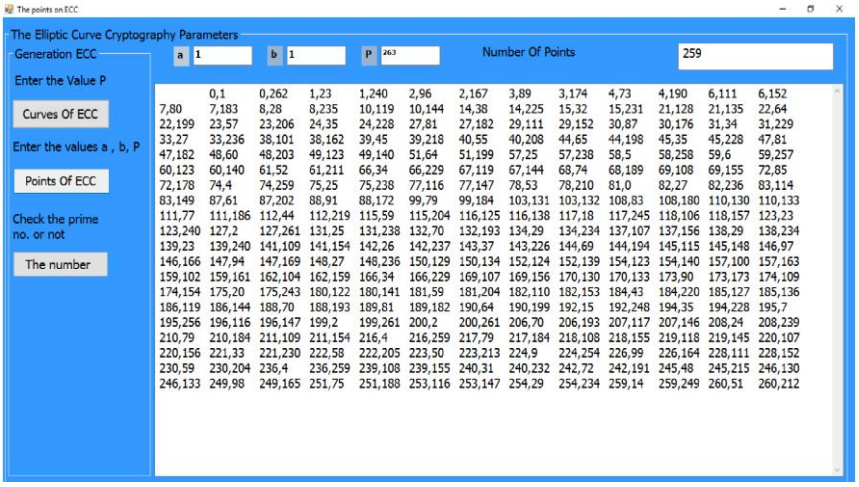

Figure 5: All *EC* Points.

Encryption time: The run time for text files of fixed of size one gaga byte with varying secret key sizes to perform the encryption process based on ElGamal Algorithm 4, as shown in Table 5:

Table 5: The Run Time for Encryption

Key Size (bits)	Run Time(seconds)
5	5.320121
10	7.371525
12	8.721148
15	10.630149
19	12.284151

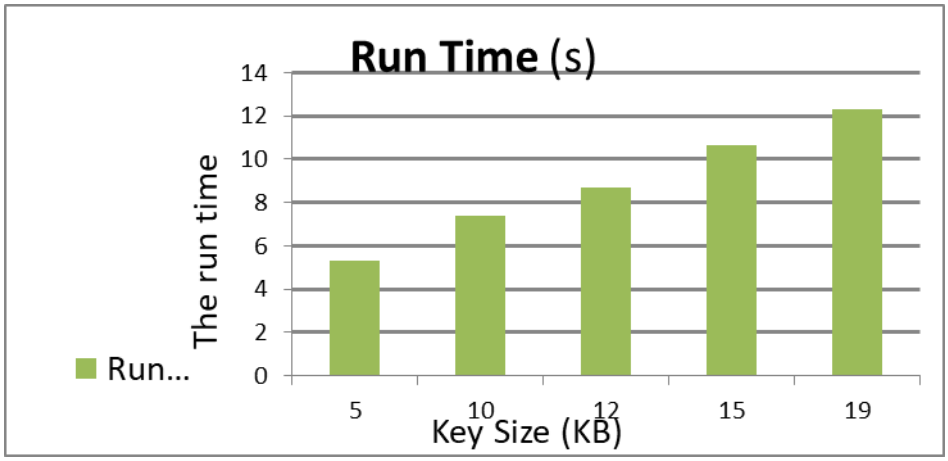

Figure 6: Encryption Time.

Decryption time: The run time for one gaga byte text files with varying secret key sizes to perform the decryption process depending on ElGamal Algorithm 4 as shown in Table 6 and Figure 7:

Table 6: The Run Time for Decryption

Key Size (bits)	Run Time(seconds)
5	18.733718
10	31.405608
12	37.085832
15	39.085793
19	47.748336

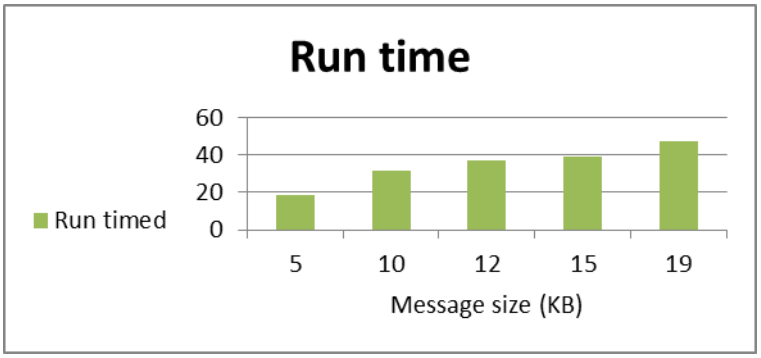

Figure 7: Decryption Time.

4.3 MOH-*ECC* Application

A new application is designed using Java code through Android studio which is a programming language used to build application working in mobile devices. We called this application MOH-*ECC*, it can be installed on any mobile device working under Android operating system. Figure 8 shows screen shoots for the main window of the proposed Android application that used to encrypt and decrypt text message in English language. It has an easy to use interface that shows how to encrypt an input text message and shows the encrypt text, it also contains another button for the decryption process. Also, there is a button used to calculate the processing time for each operation.

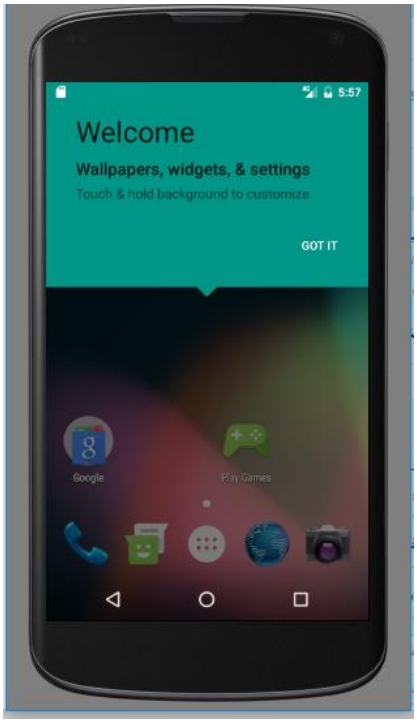

Figure 8: MOH-*ECC* Window.

Figure 9: Implementation time

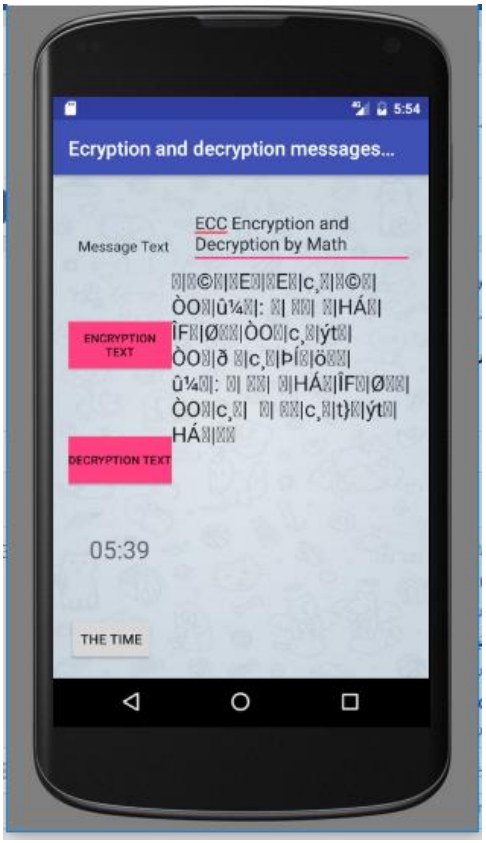

Figure 10: The Original Text and Encryption Text.

Figure 11: The original Text with Encryption and Decryption Text

Conclusions

In *ECC*, the scalar multiplication is the most important operation, while, it is the most expansive operation due to the consuming of time implementation. The speed improvement is a challenge that many researcher try to reach it. The performance of the scalar multiplication is mainly based on the representation of the scalar. In our work, we design and implement a new efficient algorithm called the nested algorithm, which is combined the elliptic curve scalar multiplication and a signed binary representation *MOF* algorithm. The running time of the nested algorithm has been reduced compared with *ECSM* based on *MOF* representation according to the result.

According to the experiment, it is obvious that, the computational time is reduced in comparison to other *ECSM* methods with the same key size. Also, after applying of the proposed algorithm

to build new EC cryptosystem of this system is efficient and can be used in any mobile device working under Android system. It is easy to install and support both common language in our country (Arabic and English). Using large key sizes, we notice that, the computational time for the proposed cryptosystem is acceptable.

Conflict of Interests

The authors declare that there is no conflict of interests.

REFERENCES:

[1] V.S. Miller, Use of Elliptic Curves in Cryptography, Advances in Cryptology, Proceedings of CRYPTO. 85, pp. 417-426, 1986.

[2] N. Koblitz, Elliptic Curve Cryptosystem, Math. Comput. 48 (1987), 203- 209.

[3] L. Ronald Rivest, Adi Shamir, and Len Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2)(1978), 120–126.

[4] G. Locke and P. Gallagher. Digital Signature Standard (DSS). Federal information processing standards publication. National Institute of Standards and Technology, 2009.

[5] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. IEEE Trans. Inf. Theory, 31(4)(1985), 469–472.

[6] M. Anagreh, A. Samsudin and M.A Omar. Parallel Method for Computing Elliptic Curve Scalar Multiplication Based on MOF. Int. Arab J. Inf. Technol. 11(6)(2014), 521-525.

[7] Y. Dan, X. Zou, Z. Liu, Y. Han, and L. Yi. High-Performance Hardware Architecture of Elliptic Curve Cryptography Processor over GF (2^{163}). J. Zhejiang Univ. Sci. A, 10(2)(2009), 301-310.

[8] A. Osmani. Design and Evaluation of New Intelligent Sensor Placement Algorithm to Improve Converge Problem in Wireless Sensor Networks, J. Basic. Appl. Sci. Res., 2(2)(2012), 1431-1440.

[9] I. Blake, G. Seroussi, and N. Smart. Elliptic Curves in Cryptography. Cambridge University Press, UK, 1999.

[10] E. Al-Daoud. An Improved Implementation of Elliptic Curve Digital Signature by using Sparse Elements. Int. Arab J. Inf. Technol. 1(2)(2004), 203- 208.

[11] T. Al-Somani and M.Ibrahim. Generic-Point Parallel Scalar Multiplication without Precomputation, IEICE Electronics Express, 6(24)(2009), 1732-1736.

[12] B. Ansari and H. Wu, Parallel Scalar Multiplication for Elliptic Curve Cryptosystems, in Proceedings of International Conference on Communications, Circuits and Systems, Ontario, Canada, vol. 1, pp. 71-73, 2005.

[13] P. Balasubramaniam and E. Karthikeyan. Fast Simultaneous Scalar Multiplication, Appl. Math. Comput. 192(2)(2007), 399-404.

[14] X. Huang, P. Shah, and D. Sharma, Minimizing Hamming Weight Based on l's Complement of Binary Numbers over GF(2^m), in Proceedings of the 12th International Conference on Advanced Communication Technology, Piscataway, USA, pp. 1226-1230, 2010.

[15] P. Balasubramaniam, and E. Karthikeyan. Elliptic Curve Scalar Multiplication Algorithm using Complementary Recoding, Appl. Math. Comput., 190(1)(2007), 51-56.

[16] D. A. Booth. A Signed Binary Multiplication Technique. Q. J. Mech. Appl. Math. 4(2)(1951), 236–240.

[17] J.A. Solinas. Efficient arithmetic on Koblitz curves. Designs, Codes Cryptogr. 19(2–3)(2000), 195-249.

[18] K. Koyama, Y. Tsuruoka. Speeding up Elliptic Cryptosystems by using a Signed Binary Window Method. In Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology. Springer-Verlag: California, USA, pp.357–345; 1993

[19] I.F. Blake, G. Seroussi, NP. Smart. Elliptic Curves in Cryptography. Cambridge University Press: Cambridge, UK, 1999.

[20] K. Okeya, K. Samoa, C. Spahn, and T. Takagi. Signed Binary Representations Revisited, in Proceeding of Annual International Cryptology Conference Advances in Cryptology Crypto, Santa Barbara, USA, pp. 123-139, 2004.

[21] H. K. Pathak and Manju Sanghi. Speeding up Computation of Scalar Multiplication in Elliptic Curve Cryptosystem. Int. J. Comput. Sci. Eng. 2(4)(2010), 1024–1028.

[22] S. William. Cryptography and Network Security: Principles and Practice. Pearson-Prentice Hall: NJ, USA, 2006.

[23] H. Darrel, JM. Alfred, V. Scott. Guide to Elliptic Curve Cryptography, Vol. 332. Springer-Verlag: New York, 2003.

[24] M. Stalling. Cryptography and Network Security, Prentice Hill, USA, 2011.