International Journal of Computer Applications (0975 8887)
Volume 108 - No. 20, December 2014

High Performance Methods of Elliptic Curve Scalar Multiplication

Najlae Falah Hameed Al Saffar
1st: Institute for Mathematical Research
Universiti Putra Malaysia, Malaysia
Faculty of Mathematics and Computer Science,
Kufa University, Iraq

Mohamad Rushdan Md Said
Institute for Mathematical Research
Universiti Putra Malaysia, Malaysia

ABSTRACT

Elliptic curve scalar multiplication is the operation of successively adding a point along an elliptic curve to itself k times. It is used in elliptic curve cryptography (ECC) as a means of producing a trapdoor function. In this paper, algorithms to compute the elliptic curve scalar multiplication using a special form for integers will introduce, and then two types of signed digit representation will use. The signed digit form of the scalar is calculated by many types of algorithms such as binary , non adjacent form and direct recoding. The results indicate that the proposed methods perform better to compute the scalar multiplication on elliptic curves and it is more efficient than the existing methods.

Keywords:

Elliptic Curve Cryptosystem, Elliptic Curve Scalar Multiplication, Signed Digit Representation

1. INTRODUCTION

In the mid of $1980s$, Miller [15] and Koblitz [12] introduced a new and an efficient public key crypyosystem named elliptic curve cryptography which is dependent on the difficulty of the one of the hard mathematical problem (HMP), which is elliptic curve discrete logarithm problem ($ECDLP$). Since there are no known subexponential time algorithms to solve the $ECDLP$, ECC supplies the same level of security with a smaller key size compared with the well known public key cryptosystems based on the discrete logarithm problem (DLP) and the integer factoring problem (IFP) over finite fields such as RSA [22], DSA [13] and $AlGamal$ [6]. Because of this singularity (requires a shorter key sizes are translated to less power and storage requirements, and reduced computing times compared with another public cryptosystems) using ECC is recommended in resource constrained environments, such as embedded devices and smart cards. Meanwhile the standard bodies such as $NIST$, and ISO have adopted ECC as an alternative and efficient public key cryptosystem.
Scalar multiplication (point multiplication) on elliptic curve is the operation of computing an integer multiple of an element in the group of elliptic curve. In other words, it is the calculation of kP for any scalar k and a point P on the elliptic curve. There are multiple investigations on how to make this operation (scalar multiplication on elliptic curve) over prime or binary fields faster as much as possible. In this work elliptic curves which are defined on prime field will be consider. For more details the reader is referred to [8] [1] [14].
Since work began on encryption/ decryption system using ECC, researchers were trying to enhance the efficiency of it. One of these ways is improving the elliptic curve scalar multiplication by reducing the number of operations required to calculate it. This operation kP is exactly the processing of computing

$$Q = kP = \underbrace{P + P + ... + P}_{k\ times}$$

where k is a positive integer called scalar and P, Q are elliptic curve points. Therefore, reducing the number of these additions ($k\ times$) will make the elliptic curve scalar multiplication faster, and then it will make ECC more efficient.
The basic technique to compute kP is based on the binary form of the scalar k which is called Binary Method [11]. This method scans the bits of k and it performance depends on if the bit is 0 or 1. That means, computing kP depends on the type of representation of k. In 1951, Booth [3] proposed a new signed representation for any integer, where the bits not only contains 0, 1 but with -1. This method is named Non Adjacent Form (NAF). Using this form to compute kP made the ECC more efficient. A generalization of the NAF called the Window-w Non Adjacent (denoted $w - NAF$) which is another method used to compute kP also made the ECC more efficient. Many algorithms had been introduced to improve the efficiency of ECC by transferring the scalar k to the NAF of $w - NAF$ such as [16] [19] [10].
In 2010, H. K. Pathak and Manju Sanghi [20] proposed a method to compute the elliptic scalar multiplication kP named as "Direct Recoding". In this method a new singed binary representation is created by bitwise subtraction. Compared with many types of representation of the scalar k the direct recoding method reduces the number of non zero bits . The main contribution of the current work: Firstly, analysing the Direct Recoding method and the methods which were mentioned in [20]. Secondly, proposed new algorithms with high performance using the same technique but with the lowest number of non zero bits comparing with existing algorithms to compute elliptic curve scalar multiplication.

2. PRELIMINARIES

In this section, a brief review of the materials which is used in the current work will be introduced. An elliptic curve E over an arbitrary field F denoted by $E(F)$ is given by the Weierstrass equation [23] as follows:

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

where a_1, a_2, a_3, a_4, $a_6 \in F$, and $\Delta \neq 0$, where Δ denoted to the discriminant of E.
The set of points on E is the set of solutions in F to the equation (1), together with a special point named point at infinity O_∞ .
Over the prime field F_p, with $p > 3$ the equation (1) simplifies as follows:

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

where $a,\, b \in F_p$ and $\Delta = 4a^3 + 27b^2 \neq 0$.
Over the binary field F_{2^m}, equation (1) can be simplified to:

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

where $a,\, b \in F_{2^m}$ and $\Delta = b \neq 0$.
Over the field of real number R, the elliptic curve is defined on equation (2) but with $a,\, b \in R$ and $\Delta = 4a^3 + 27b^2 \neq 0$.

Theorem: Let $P, Q \in E$, L the line connecting P and Q (tangent line to E if $P = Q$), and M the third point of intersection L with E. let L' be the line connecting M and O_∞. Then the point $P + Q$ is the third point on E such that L' intersects E at M, O_∞ and $P + Q$.
The set $E(F)$ of rational points on an EC E defined over a field F forms an abelian additive group. The additively operation defined by the tangent and secant law. **Figure** 1 is illustrated this operation geometrically [24] [4] on special EC over the real field, as an example if the target is compute $P + Q$ for P and Q are points on E, then draw a line though P and Q which intersects with the E at the third point M on E, the intersection between the vertical line and the E is $P + Q$.

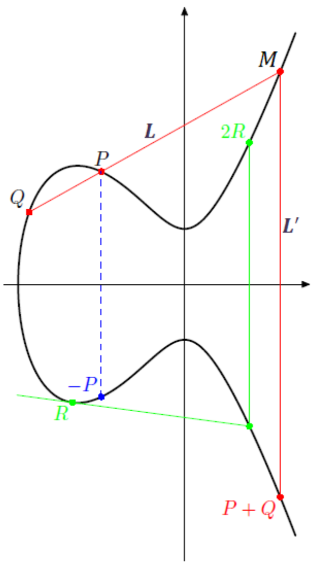

Fig. 1. Elliptic Curve Addition

The focus of this work will be with EC E defined over field of prime number F_p which is denoted by $E(F_p)$ given by equation (2).

Theorem: Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ be points in $E(F_p)$. Then $P_3 = P_1 + P_2 = (x_3, y_3)$ in $E(F_p)$ is computed by

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

where

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{3x_1^2+a}{2y_1} & \text{if } P_1 = P_2 \\ \frac{y_2-y_1}{x_2-x_1} & \text{if otherwise.} \end{cases}$$

That means, the doubling operation requires some steps. So, for each doubling of $P \in E(F_p)$ the above procedure is performed to obtain $2P$, $2(2P)$, $2(2(2P))$, ... This operation is considered as elliptic curve scalar multiplication kP, where k a secret key in the ECC, usually with very long bits.
Consider an operation $P_1 + P_2$ as $ECADD$, and an operation $2P$ with $P \in E(F_p)$ as $ECDBL$. Furthermore, details can be found in [4] [2] [24].
The $ECDLP$ is to compute the scalar k for a given point on EC say P and kP, and this is known to be very difficult, which is similar to the DLP on the finite field. That is why the elliptic curve scalar multiplication is the central operation in ECC. The main idea is to get the shortest representation of the scalar k in order to reach the most efficient way to compute the elliptic curve scalar multiplication. The following definitions needed to illustrate this item.

Definition: A signed digit representation of an integer k to the base b (denoted by $(k)_b$ is an ordered sequence of integers $k_0\, k_1\, k_2 \ldots k_r$ with $|k_i| < b$ for $i = 0, 1, \ldots, r$, such that $k = \sum_{i=1}^{r} k_i b^i$.
Signed digit representation is not unique, for example,

$$\begin{aligned} 18 &= (11\bar{1}\bar{1}0)_2 \\ &= (10010)_2 \\ &= (1\bar{1}0010)_2 \end{aligned}$$

where $\bar{1} = -1$.
In 2007, Ebeid and Hasan [5] proposed an algorithm to generate all possible signed digit representation of any integer k.
One of the types of representation of the number can be generated using the following proposition:

Proposition: For any positive integer k there exists s such that $2^s \leqslant k < 2^{s+1}$.
Proof: Consider the statement $A(k)$ given by

$$2^s \leqslant k < 2^{s+1}. \qquad (4)$$

Mathematical induction is used to show that $A(k)$ is true for any $k \geqslant 1$.

First, it will be confirmed that the $A(1)$ is true. Since there exists an integer $s = 0$ such that

$$2^0 \leqslant 1 < 2. \qquad (5)$$

Thus, $A(1)$ is true.

Next, assume that $A(k)$ is true for some $k \geqslant 1$. So, assume that

$$2^s \leqslant k < 2^{s+1}. \quad (6)$$

Finally, to prove that $A(k+1)$ is true. Since $2^s \leqslant k$ then $2^s + 1 \leqslant k + 1$. And, since $2^s \leqslant 2^s + 1$, then $2^s \leqslant k + 1$. Now, to prove the right side of $A(k+1)$, there are two cases for $k \geqslant 2$:

Either $k = 2^s$, then $k + 1 = 2^s + 1 < 2^{s+1}$.

Or $k < 2^s$, then $k + 1 < 2^s + 1 < 2^{s+1}$.

Thus, $A(k+1)$ is true.

By induction, $A(k)$ proved that it is true for any positive integer k. □

Definition: The hamming weight of an integer k (denoted by $h(k)$) is the number of 1s in the signed digit representation.

Definition: The length of the expression $(k)_b$ (denoted by $l(k)$) is the number of its digits.

3. ELLIPTIC CURVE SCALAR MULTIPLICATION

Elliptic curve cryptographic schemes require calculations for

$$Q = kP = \underbrace{P + P + ... + P}_{k\ times}$$

where k is a positive integer called scalar and P, Q are elliptic curve points. This operation is known as elliptic curve scalar multiplication. The simplest way to perform the elliptic curve scalar multiplication kP is the binary algorithms, which is the analogue of the square and multiply process for fast modular exponentiations [24].
Elliptic curve scalar multiplication is involved in elliptic curve digital signature algorithm ($ECDSA$) [9] and many others protocols. Implementing such protocols on embedded devices requires particular care from both the efficiency and the security points of view.
In this section the common methods for performing scalar multiplication on an elliptic curve will discuss . These methods are to represent the scalar k in different ways to compute the main operation in ECC which is kP elliptic scalar multiplication.

3.1 Binary Method

The basic technique for elliptic scalar multiplication is the $ECADD$ and $ECDBL$. It is based on the binary method of the coefficient k. The integer k is represented as a signed digit representation or as $k = k_{l-1}2^{n-1} + k_{l-2}2^{l-2} + ... + k_0$ where $k_{l-1} = 1$ and $k_i \in \{0, 1\}$, $i = 0, 1, 2, ..., l-1$. That is $k = \sum_{i=0}^{l-1} k_i 2^i$, where $k_i \in \{0, 1\}$. This method scans the bits of the bits of k from the left to right, if the bit is 1 then perform a $ECDBL$ and $ECADD$, otherwise, the $ECDBL$ will perform (the first bit is always 1 which is use as initialization). This method is called binary method. The process of this method to compute kP is given in the following **Algorithm 1**.

Algorithm 1: Binary Method for Elliptic Curve Scalar Multiplication

Input:$(k)_{10} = (k_{l-1}...k_1k_0)_2$, $P \in E(F_p)$
Output:$Q = kP$
1. $Q = P$
2. For $i = l-1$ down to 0 do
2.1. $Q = 2Q$
2.2. If $k_i = 1$ then $Q = Q + P$
3. Return Q

For example, let us assume that k is equal to $(109)_{10}$, so in the binary representation k is equal to $(1101101)_2$. The $109P$ for $P \in E(F_p)$ is compute as follows:

$$\begin{array}{llcc} e_6 & 1 & P & initialization \\ e_5 & 1 & 2P+P & doubling\ and\ addition \\ e_4 & 0 & 2(2P+P) & doubling \\ e_3 & 1 & 2(2(2P+P))+P & doubling\ and\ addition \\ e_2 & 1 & 2(2(2(2P+P))+P)+P & doubling\ and\ addition \\ e_1 & 0 & 2(2(2(2(2P+P))+P)+P) & doubling \\ e_0 & 1 & \underbrace{2(2(2(2(2(2P+P))+P)+P))+P} & doubling\ and\ addtion \\ & & \Downarrow & \\ & & 109P & \end{array}$$

The cost of elliptic curve scalar multiplication using binary method depends on the $l(k)$and the $h(k)$ in the representation of k. If $(k)_{10} = (k_{l-1}\,k_{l-2}\,...\,k_1\,k_0)_2$, then the number of $ECDBL$ is $l-1$ and the number of $ECADD$ is one less than the $h(k)$. In an average, the binary method requires $l-1$ $ECDBL$ and $\frac{l-1}{2}$ $ECADD$. For example, the cost of computation of $109P$ in the above example requires $(6ECDBL + 4ECADD)$.
As a conclusion, whenever the bit is 1, two elliptic curve arithmetic operations doubling and addition will be made and if it is 0, only one operation, doubling is required. Thus, if the number of $h(k)$ in the scalar representation reduced , then accelerate this computation will accrue.

3.2 Non Adjacent Form (NAF) Method

The hamming weight of the scalar k can be reduced with a signed representation that uses the numbers 0 and ±1. Among various signed representation, NAF is a canonical representation with less number of hamming weight for integer k. The NAF representation of k has been proposed in 1951, by Booth [3] (some time the searchers called it Addition-subtraction method according to its process). And after 10 years Rietweisner [21] has been proved that every integer could be uniquely represented in this form.
Nowadays, The NAF of a scalar k denoted by $NAF(k)$ becomes the subject of various investigations in different contexts such as elliptic curve scalar multiplication. The property of this representation is that, of any two consecutive digits, at most one is non zero, Moreover, the length of $NAF(k)$, denoted by $l(NAF(k))$ is at most 1 more bit than its binary representation. This means, fewer point additions and therefore more efficiency when needed to compute the scalar multiplication on EC.
Now, because of the group law of elliptic curve group, it is observed that the inverse of $P = (x, y) \in E(F_p)$ is $-P = (x, -y) \in E(F_p)$. Therefore, computing inverse of any point on elliptic curve is free and very fast in terms of computational time. That is, in the process of computing the kP, and the minus is come across, subtraction of P is performed during this computation, furthermore, it costs the same amount of $ECADD$ in the total operation.
In the example of signed digit representation, it mentioned that there is no unique singed digit representation for any integer k. To get this uniqueness has to add some conditions on the representation, this condition it will be that there are no adjacent non zeros (using NAF).

For example, the number 7 can have several signed-digit representations:

$$(0111)_2 = 4 + 2 + 1 = 7$$
$$(10\bar{1}1)_2 = 8 - 2 + 1 = 7$$
$$(1\bar{1}11)_2 = 8 - 4 + 2 + 1 = 7$$
$$(100\bar{1})_{NAF(7)} = 8 - 1 = 7$$

But only the last representation is NAF.

Definition: A NAF of a positive integer k is a singed digit representation of k to the base $b = 2$, such that $k_i\, k_{i+1} = 0$ for $i \geq 0$. The $NAF(k)$ is written $(k_{l-1} \ldots k_1\ k_0)_{NAF(k)}$.

The reader can refer to Gordon [7] for the proofs of existence and uniqueness of $NAF(k)$ for any integer k. Muir and Stinson [17] in their paper have proved that the hamming weight of the $NAF(k)$ is minimal among all signed digit representations of k. Fortunately, the number of bits in the $NAF(k)$ is at most one more than the number of bits in the binary form of k. **Algorithm 2** is for the conversion of a scalar k into NAF.

Algorithm 2: Computing NAF of a Scalar k

Input: A scalar $(k)_{10}$
Output: $N = (k_{l-1} \ldots k_1 k_0)_{NAF(k)}$
$i = 1$; $c = k$
While $c > 0$
If c odd
$N(i) = 2 - (c\, mod\, 4)$
$c = c - N(i)$
Else
$N(i) = 0$
End if
$c = \frac{c}{2}$;$i = i + 1$
End while
Return N

Performance of **Algorithm 2** can be summarized in the following steps:

(1) If k is an even integer, 0 have to take, and continue with $\frac{k}{2}$.
(2) If $k \equiv 1(mod4)$, 1 have to take, and continue with $\frac{k-1}{2}$ which is even integer that guarantees a 0 in the next step.
(3) If $k \equiv 3 \equiv -1(mod4)$, take -1, and continue with $\frac{k+1}{2}$ which is even integer that guarantees a 0 in the next step.

This measure produces zero after each non zero digit, which means this signed-digit representation must has low hamming weight.
For example, the mechanism to compute $NAF(27)$, according to **Algorithm 2**, is shown in **Table 1**.

Table 1: Computing a $NAF(27)$

i	c	$N(i)$	N
1	27	$\bar{1}$	$(\bar{1})$
	28		
2	14	0	$(0\bar{1})$
3	7	$\bar{1}$	$(\bar{1}0\bar{1})$
	8		
4	4	0	$(0\bar{1}0\bar{1})$
5	2	0	$(00\bar{1}0\bar{1})$
6	1	1	$(100\bar{1}0\bar{1})$
	0		

Remarks:

(1) $NAF(k)$ for a scalar k has fewest non zero digits (hamming weight) of any signed representation of k, unless if the binary representation of k already has. For instance

$$(89)_{10} = (1011001)_2$$
$$= (10\bar{1}0\bar{1}001)_{NAF(89)}.$$

(2) The length of $NAF(k)$ is at most one more bit than its binary representation.
(3) If $l(NAF(k)) = l$, then $\frac{2^l}{3} < k < \frac{2^{l+1}}{3}$.
(4) The average hamming weight of $NAF(k)$ (denoted by $h(NAF(k))$ when $l(NAF(k)) = l$ is $\frac{l}{3}$.

The method for computing the scalar multiplication kP using NAF expression is **Algorithm 3**.

Algorithm 3: NAF Method for Elliptic Curve Scalar Multiplication

Input:$(k)_{10} = (k_{l-1} \ldots k_1 k_0)_{NAF(k)}$, $P \in E(F_p)$
Output:$Q = kP$
1. $Q = P$
2. For $i = l - 1$ down to 0 do
2.1. $Q = 2Q$
2.2. If $k_i = 1$ then $Q = Q + P$
2.3. If $k_i = -1$ then $Q = Q - P$
3. Return Q

According to the **Algorithm 3**, the scalar multiplication using NAF method requires $\frac{l}{3}ECADD$ and $lECDBL$, where the subtraction and addition operation have the same cost in the case of the elliptic curves group.

Example: Let $k = 127$ and P a point on the elliptic curve E. Now, the binary representation of k is $(1111111)_2$, so the cost is exactly equal to $(6ECDBL + 6ECADD)$.
While, the $NAF(127)$ is $(1000000\bar{1})_{NAF(127)}$, so the cost it is equal to $(7ECDBL + 1ECADD)$.

3.3 Mutual Opposite Form (MOF)

In 2004 Okeya et al. [18] introduced an algorithm to compute the elliptic curve scalar multiplication, called mutual opposite form (MOF). He also proved that this form is unique for all positive integers. MOF satisfies the following properties:

(1) Signs of adjacent non zero bits (regardless 0 bits) are opposite.
(2) Most non zero bit and the least non zero bit are 1 and -1 respectively.

The idea of the MOF method is summarized by converting the binary string of the scalar k into signed digit representation by computing

$$mk = 2k - k$$

where $(-)$ stands for a bitwise subtraction. **Algorithm 4** is for the conversion of a scalar k into the MOF.

Algorithm 4: Computing MOF of a Scalar k

Input: A scalar $(k)_{10} = (k_{n-1}\, k_{n-2} \ldots k_1\, k_0)_2$
Output: $MOF(k) = (mk_n\, mk_{n-1} \ldots mk_1\, mk_0)_{MOF(k)}$
Set $mk_n = k_{n-1}$
For $i = n - 1$ down to 1
$mk_i = k_{i-1} - k_i$
$mk_0 = -k_0$
Return N

The conversion of MOF expression can be created from right to left or from left to right.
For example, the mechanism to compute $MOF(9)$ and $MOF(27)$, according to **Algorithm 4**, is shown in **Table 2** and **3**.

Table 2: Computing a $MOF(9)$

i	k_i	mk_i	$MOF(k)$
4		1	(1)
3	1	$0-1=-1$	$(1\bar{1})$
2	0	$0-0=0$	$(1\bar{1}0)$
1	0	$1-0=1$	$(1\bar{1}01)$
0	1	-1	$(1\bar{1}01\bar{1})$

Table 3: Computing a $MOF(27)$

i	k_i	mk_i	$MOF(k)$
5		1	(1)
4	1	$1-1=0$	(10)
3	1	$0-1=-1$	$(10\bar{1})$
2	0	$1-0=1$	$(10\bar{1}1)$
1	1	$1-1=0$	$(10\bar{1}10)$
0	1	-1	$(10\bar{1}10\bar{1})$

According to these examples the MOF dose not work for all integer. In other words, MOF is efficient but not for all scalars, this is due to that for two expressions $MOF(9)$ and $MOF(27)$ the hamming weight are increased from 2 to 4 and from 3 to 4 respectively comparing with $NAF(9) = (1001)_{NAF(9)}$ and $NAF(27)$, while the hamming weight for $(27)_2 = (11011)_2$ is 4 it is equal to the hamming weight of $MOF(27)$.

3.4 Direct Recoding Method (DRM)

In 2010, Pathak and Sanghi [20] proposed a new method to present the scalar k in a new form. This method is named direct recoding method (DRM).
DRM has the lowest hamming weight. The idea comes from the fact that for any scalar k, there exist s such that

$$2^s < k < 2^{s+1} \tag{7}$$

So,

$$k = (2^{s+1})_2 - (2^{s+1} - k)_2 \tag{8}$$

In **Section 2** it has proved that for any scalar k, there exist s such that

$$2^s \leq k < 2^{s+1} \tag{9}$$

So, the formula should be as

$$2^s \leq k < 2^{s+1} \tag{10}$$

For example, $2^3 \leqslant 8 < 2^4$. **Table 4** and **5** are examples of the mechanism to create the $DRM(9)$ and $DRM(27)$.

Table 4: Computing a $DRA(9)$

	$2^{3+1} > (9) > 2^3$
$\therefore$	$2^4 > (9) > 2^3$
$\therefore$	$9 = (2^4)_2 - (2^4 - 9)_2$
	$= (16)_2 - (7)_2$
	$= (10000)_2 - (111)_2$
$\therefore$	$DRM(9) = (10\bar{1}\bar{1}\bar{1})_{DRM(9)}$

Table 5: Computing a $DRM(27)$

	$2^{4+1} > (27) > 2^4$
$\therefore$	$2^5 > (27) > 2^4$
$\therefore$	$27 = (2^5)_2 - (2^5 - 27)_2$
	$= (32)_2 - (5)_2$
	$= (100000)_2 - (101)_2$
$\therefore$	$DRM(27) = (100\bar{1}0\bar{1})_{DRM(27)}$

From the result of computing $DRM(9)$ can find that its hamming weight was increased from 2 up to 4 comparing with the hamming weight of $(9)_2$, while in computing $DRM(27)$ it was decreased from 4 to 3 compared with the hamming weight of $(27)_2$. On the other hand, the hamming weight of DRM is the same compared with the hamming weight of $NAF(27)$ but by using only single operation of bitwise subtraction [20]. The reader can find more examples in [20].

4. PROPOSED METHOD

Elliptic curve scalar multiplication is the most prominent computation part of ECC. The proposed methods in this work is for making the elliptic curve scalar multiplication has high performance compared with the existing algorithms. This methods are to create a new form for the scalar k with fewest hamming weight compared with the other methods.
The idea of the proposed method to create new form for the scalar k is as follows: For any scalar k, there exist s such that

$$2^s \leq k < 2^{s+1} \tag{11}$$

So,

$$k = 2^s + (k - 2^s) \tag{12}$$

Then,

$$k = (2^s)_2 + (k - 2^s)_2 \tag{13}$$

Or,

$$k = (2^s)_{NAF(2^s)} + (k - 2^s)_{NAF(k-2^s)} \tag{14}$$

For example, the **Table 6** and **7** illustrate the mechanism of create the new form for $k = 9$ and $k = 27$.

Table 6: Computing the new form for $k = 9$

	$2^4 > (9) \geqslant 2^3$
$\therefore$	$9 = (2^3)_2 + (9 - 2^3)_2$
	$= (8)_2 + (1)_2$
	$= (1000)_2 + (1)_2$
$\therefore$	$9 = (1001)$
Or	
	$9 = (2^3)_{NAF(8)} + (9 - 2^3)_{NAF(1)}$
	$= (8)_{NAF(8)} + (1)_{NAF(1)}$
	$= (1000)_{NAF(1)} + (1)_{NAF(1)}$
$\therefore$	$9 = (1001)$

Table 7: Computing the new form for $k = 27$

	$2^5 > (27) \geqslant 2^4$
$\therefore$	$27 = (2^4)_2 + (27 - 2^4)_2$
	$= (16)_2 + (11)_2$
	$= (10000)_2 + (1011)_2$
$\therefore$	$27 = (11011)$
Or	
	$27 = (2^4)_{NAF(16)} + (27 - 2^4)_{NAF(11)}$
	$= (16)_{NAF(16)} + (11)_{NAF(11)}$
	$= (10000)_{NAF(16)} + (10\bar{1}0\bar{1})_{NAF(11)}$
$\therefore$	$27 = (100\bar{1}0\bar{1})$

From **Table 6** and **7** can see that when represent binary form for the separated part 2^3 and $9 - 2^3$, it got a similar form as the binary form but with least time compared with binary form and the same result when used the NAF with least time. In the other hand, for $k = 27$ when represent binary form for the separated part 2^4 and $27 - 2^4$, it got a similar form same as the binary form but with least time compared with binary form. While when used the NAF to represent the parts, the hamming weight was reduced compared with binary form with least time. Furthermore, the form is the same with the $DRM(27)$. But according to the main aim for this work which is to accelerate the elliptic curve scalar multiplication, the new form is faster than the DRM. This is due to that in DRM they used binary form to represent the separated parts, while in our proposed used the NAF which more efficient than the binary form. **Table 6** is another example to compare all these methods.

Table 8: Computing DRM and the new forms for $k = 686$

	$2^{10} > (686) > 2^9$
$\therefore$	$686 = (2^10)_2 - (2^{10} - 686)_2$
	$= (10000000000)_2 - (101010010)_2$
$\therefore$	$DRM(686) = (10\bar{1}0\bar{1}0\bar{1}00\bar{1}0)_{DRM(686)}$
	$686 = (2^9)_2 + (686 - 2^9)_2$
	$= (512)_2 + (174)_2$
	$= (1000000000)_2 + (10101110)_2$
$\therefore$	$686 = (1010101110)$
Or	
	$686 = (512)_{NAF(512)} + (174)_{NAF(174)}$
	$= (1000000000)_{NAF(512)} + (10\bar{1}0\bar{1}00\bar{1}0)_{NAF(174)}$
$\therefore$	$686 = (110\bar{1}0\bar{1}00\bar{1}0)$

From **Table 8** can find that when used the binary method to represent 512 and 174 got the same form with the binary method for $686 = (1010101110)_2$, but with least time to execute. In the other hand, when used the NAF to represent the separated parts got the same number of hamming weight of $DRM(686)$ which was 5 but the length is least (from 11 to 10). That means the number of operation is reduced as a total, 14 operation in binary and DRM methods to 13 operations in new method with using NAF.

4.1 Complexity of the Proposed Algorithm

In this section the cost of proposed method according to the mentioned methods will discuss with the same number of size, which are used to represent the scalar k to compute the elliptic curve scalar multiplication kP. At the beginning, it is observed that any $ECADD$ requires 2 squaring, 2 multiplication and 1 inversion operation as showed in Section 2. That means, reducing one number of the hamming weight of the signed digit representation of k means saving 5 operation when calculating the elliptic curve scalar multiplication. On the other hand, reducing the length of the signed digit representation of k means reducing the number of operation as a total ($ECADD$ and $ECDBL$). This happens when the proposed method used, but with the NAF on the separated parts.
In order to be able to compare the different elliptic curve scalar multiplication, the number of $ECDBL$ and $ACADD$ counted where every zero digit in the sign digits form of k refer to one $ECDBL$ and one non-zero digit refer to two operations ($ECDBL$ and $ACADD$).
Table 9 and the plots in **Figure 2** shows the number of operations required by the proposed algorithm and the binary, NAF and MOF and DRM algorithms.
Now, the comparison of the complexity of the algorithms which mentioned will give in the following table:

Table 9: Complexity of Computing Elliptic Curve Scalar Multiplication of Various Processors

Size of k	Number of Operations					
	$Algo^1$	$Algo^2$	$Algo^3$	$Algo^4$	$Algo^5$	$Algo^6$
10	14	14	17	14	14	13
16	23	21	25	24	22	21
24	42	29	13	29	42	29
32	59	36	39	37	57	36

* $Algo^1$ refers to the Binary Algorithm
* $Algo^2$ refers to the NAF Algorithm
* $Algo^3$ refers to the MOF Algorithm
* $Algo^4$ refers to the DRM Algorithm
* $Algo^5$ refers to the Proposed Method/ Binary
* $Algo^6$ refers to the Proposed Method/ NAF

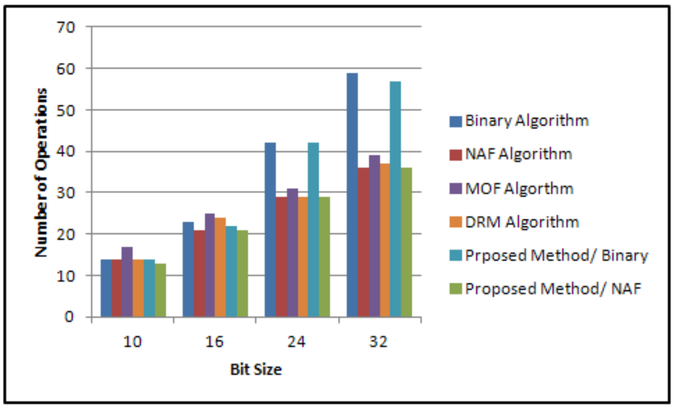

Fig. 2. Complexity of Computing Elliptic Curve Scalar Multiplication of Various Processors

All algorithms have been implemented on $Intel(R)Core(Tm)2Duo$ with processor $2.99GHz$ and $3.00GB$ of memory using $MATLAB$ version $7.10.0.499$ ($R2010a$). In order to compute the complexity, executed all algorithms with different size bits ($10, 16, 24$ and 32) for the scalar k. The following is result collected from the execution and its corresponding graphical explanation.

The efficiency of the proposed methods is clearly known, from the **Table 9** and **Figure 2**. For instance, elliptic curve scalar multiplication using binary, NAF , MOF and DRM algorithm require 23 , 21 , 25 and 24 respectively with respect to the number of operations, while the proposed algorithm/ Binary and proposed algorithm/ NAF require 22 and 21 operations respectively when k has 16 bits. That was an example that both of the proposed algorithms are efficient than all the mentioned algorithm even when they have the same number of operations but with least time in the execution processing.

5. CONCLUSION

Elliptic curve scalar multiplication is a fundamental operation in elliptic curve cryptosystem. In the recent past, a number of hardware architectures have been proposed in the literature to speed up this operation. In this work, a high performance methods to compute elliptic curve scalar multiplication scheme based on DRM algorithm has been proposed. The computational complexity of the proposed methods have high performance when compared to the other methods . This is due to that the number of operations required for execution is either the same with less time or less when compared to the another mentioned methods.

6. REFERENCES

[1] Ian F. Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography. London Mathematical Society Lecture Note Series 265*. Cambridge University Press, Cambridge, 1999.

[2] P. Smart Nigel Blake Ian F., Seroussi Gadiel. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, Cambridge, 2005.

[3] Andrew D. Booth. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, 1951.

[4] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography. Series on Discrete Mathematics and Its Applications*. Chapman & Hall/CRC, Boca Raton, 2010.

[5] Nevine Ebeid and M. Anwar Hasan. On binary signed digit representations of integers. *Designs, Codes and Cryptography*, 42(1):43–65, 2007.

[6] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[7] Daniel M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27(1):129–146, 1998.

[8] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, New York, 2004.

[9] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm ($ecdsa$). *International Journal of Information Security*, 1(1):36–63, 2001.

[10] Marc Joye and Christophe Tymen. Compact encoding of non-adjacent forms with applications to elliptic curve cryptography. In *Public Key Cryptography, K. Kim, Ed. of Lecture Notes in Computer Science*, volume 1992, pages 353–364. Springer-Verlag, 2001.

[11] Donald E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*. Addison Wesley Longman, Canada, 3rd edition, 1997.

[12] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[13] G. Locke and P. Gallagher. Fips pub 186-3: Digital signature standard (dss). federal information processing standards publication. *National Institute of Standards and Technology*, 2009.

[14] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1st edition, 1996.

[15] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology-CRYPTO'85 Proceedings (Santa Barbara, Calif., 1985)*, volume 218, pages 417–426. Springer-Verlag, 1986.

[16] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *Theoretical Informatics and Applications*, 24:531–543, 1990.

[17] James A. Muir and Douglas R. Stinson. Minimality and other properties of the width-w nonadjacent form. *Mathematics of Computation*, 75(253):369–384, 2006.

[18] Katsuyuki Okeya, Katja Schmidt-Samoa, Christian Spahn, and Tsuyoshi Takagi. Signed binary representations revisited. In *Advances in Cryptology-CRYPTO2004, LNCS 3152*, pages 123–139. Springer-Verlag, 2004.

[19] Katsuyuki Okeya and Tsuyoshi Takagi. The width-$w\,naf$ method provides small memory and fast elliptic scalar multiplications secure against side channel attacks. 2612:328–343, 2003.

[20] H. K. Pathak and Manju Sanghi. Speeding up computation of scalar multiplication in elliptic curve cryptosystem. *International Journal on Computer Science and Engineering*, 2(4):1024–1028, 2010.

[21] George W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[22] Ronald L. Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[23] Joseph H. Silverman. *The Arithmetic of Elliptic Curves. Graduate Texts in Mathematics 106*. Springer, New York, 2nd edition, 2009.

[24] Song Y. Yan. *Number Theory for computing*. Springer-Verlag, Berlin, 2nd edition, 2002.