Chapter

The Graphs for Elliptic Curve Cryptography

Ruma Kareem K. Ajeena

Abstract

The scalar multiplication on elliptic curves defined over finite fields is a core operation in elliptic curve cryptography (ECC). Several different methods are used for computing this operation. One of them, the binary method, is applied depending on the binary representation of the scalar v in a scalar multiplication vP, where P is a point that lies on elliptic curve E defined over a prime field F_p. On the binary method, two methodologies are performed based on the implementation of the binary string bits from the right to the left (RLB) [or from the left to the right (LRB)]. Another method is a nonadjacent form (NAF) which depended on the signed digit representation of a positive integer v. In this chapter, the graphs and subgraphs are employed for the serial computations of elliptic scalar multiplications defined over prime fields. This work proposed using the subgraphs H of the graphs G or the (simple, undirected, directed, connected, bipartite, and other) graphs to represent a scalar v directly. This usage speeds up the computations on the elliptic scalar multiplication algorithms. The computational complexities of the proposed algorithms and previous ones are determined. The comparison results of the computational complexities on all these algorithms are discussed. The experimental results show that the proposed algorithms which are used the sub-graphs H and graphs G need to the less costs for computing vP in compare to previous algorithms which are employed the binary representations or NAF expansion. Thus, the proposed algorithms that use the subgraphs or the graphs to represent the scalars v are more efficient than the original ones.

Keywords: ECC, scalar multiplication, BRL, BLR, NAF, graphs, subgraphs, computational complexity

1. Introduction

The scalar multiplication on elliptic curves defined over finite fields is considered as a central and most time-consuming operation in elliptic curve cryptography (ECC) [1–7]. Different methods are used for computing the scalar multiplication such as the binary method, nonadjacent form, and others [8–15]. The binary method is applied depending on the binary representation of the scalar v in a scalar multiplication vP, where P is a point that lies on elliptic curve E defined over a prime field F_p. On the binary method, two methodologies are performed based on the implementation of the binary string bits from the right to the left (RLB) [or from the left to the right (LRB)], whereas the nonadjacent form (NAF) depends on the signed digit representation of a positive integer v [1].

In this chapter, the computation of the scalar multiplication vP on elliptic curve E defined over a prime field F_p has been done using the (undirected or directed) graph and (undirected or directed) subgraph. These graph and subgraph are used to represent the scalar v in two ways. The first one is the binary representation and the second one is the sign digit representation.

Also, the l-tuple of the elliptic scalar multiplications is computed using the proposed generalized binary methods (GRLB) and (GLRB) and GNAF. The computational complexities of the proposed algorithms and previous ones are determined. The comparison results of the computational complexities on all these algorithms are discussed. Several experimental results showed that the proposed algorithms which are used the graphs G need to the less costs for computing vP in compare to previous algorithms which are employed the binary representations or NAF expansion. Therefore, the proposed algorithms that use the subgraphs or the graphs to represent the scalars v are more efficient than the original ones.

This chapter is organized as follows: Section 2 presents the vector representation of the graph. Section 3 discusses the matrix representation of the graph. Section 4 includes the binary methods of the elliptic scalar multiplication which are the right-to-left binary and left-to-right binary representations. Section 5 explains the non-adjacent form method, whereas Section 6 discusses the graphic binary methods of the elliptic scalar multiplications. Section 7 displays the digraphic NAF method. Section 8 presents the subgraphs for computing the elliptic scalar multiplication. Section 9 determines the computational complexities on the original elliptic scalar multiplication methods. Section 10 shows the computational complexity for serial computing l-tuple of the scalar multiplications. The computational complexity of the graphic elliptic scalar multiplication methods is explained in Section 11. Section 12 illustrates the computational complexity comparison on the serial and graphic computation methods. Finally, Section 13 draws the conclusions.

2. The vector representation of the graph

Suppose G is a graph as shown in **Figure 1**.

A graph G has four vertices and five edges e_1, e_2, e_3, e_4, and e_5. A subgraph H (and any other subgraphs) of G is represented by a 5-tuple.

This means that $E = (e_1, e_2, e_3, e_4, e_5)$ such that

$$e_i = 1, \text{ if } e_i \text{ is in } H,$$

$$e_i = 0, \text{ if } e_i \text{ is not in } H.$$

The subgraphs H_1 and H_2 in **Figure 1** can be represented by (1,0,1,0,1) and (0,1,1,1,0), respectively. Here, there are $2^5 = 32$ possible cases for 5-tuples which

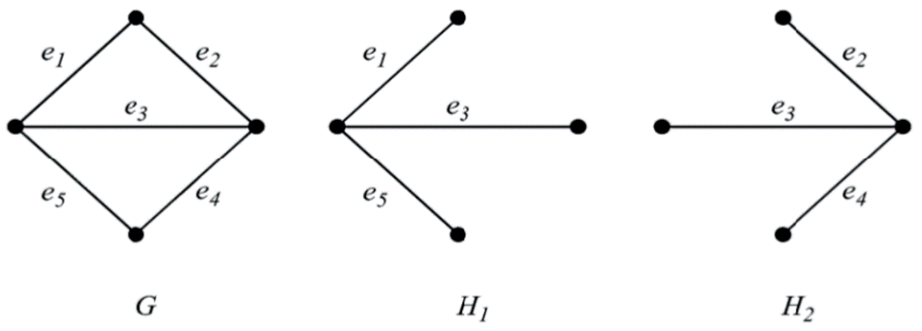

Figure 1.
The subgraphs H_1 and H_2 of the graph G [16].

correspond to 32 subgraphs. Among them are the (0,0,0,0,0) and (1,1,1,1,1) which represent a null graph and a graph G itself, respectively [16].

3. The matrix representation of the graph

Suppose G is any undirected graph that is formed by two finite sets V and E, which are called the vertices and edges, respectively. In other words, $V = \{v_1, v_2, \dots, v_l\}$ and $E = \{e_1, e_2, \dots, e_m\}$. The matrix representation $A(G) = \left(e_{ij}\right)_{l\times m}$ on graph G has been defined by

$$A(G) = \begin{array}{c|c} \begin{array}{c} v_1 \\ v_2 \\ \vdots \\ v_l \end{array} & \begin{bmatrix} e_{1_1} & e_{2_1} & e_{3_1} & \cdots & e_{m_1} \\ e_{1_2} & e_{2_2} & e_{3_2} & \cdots & e_{m_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{1_l} & e_{2_l} & e_{2_l} & \cdots & e_{m_l} \end{bmatrix} \end{array} \tag{1}$$

with l rows corresponding to the l vertices v_i and the m columns corresponding to the m edges e_i. Whereas the incidence matrix of a connected digraph can be defined by $A = \left(e_{ij}\right)_{l\times m}$, where $e_{ij} \in \{0, \mp 1\}$. In other words, if j^{th} edge is incident out of i^{th} vertex, then $e_{ij} = 1$, while $e_{ij} = -1$, if j^{th} edge is incident into i^{th} vertex and if j^{th} edge is neither incident out nor incident into i^{th} vertex, then $e_{ij} = 0$ [16, 17].

4. The binary methods for the elliptic scalar multiplication

Two methods for computing the scalar multiplication vP have been created based on using the binary representation of a scalar v. One of them is called the right-to-left binary (RLB) method, and another one is called left-to-right binary (LRB) method [1, 9, 10]. These methods depend on the basic repeated-square-and multiply methods for exponentiation with additive version. Using the RLB method, the process of v-bits starts from the right to the left, whereas the v-bits processing starts from the left to the right using the LRB method. The RLB and LRB methods are discussed mathematically as follows.

4.1 The right-to-left binary method

Suppose E is an elliptic curve defined over a prime field F_p. The equation of E is given by $E: y^2 = x^3 + ax + b \pmod p$. Let $P = (x, y)$ be a generator point that lies on E which has a (large) prime order n. Choosing v to compute vP can be done from the range [1, $n-1$]. So, it should first write v in a binary representation string $(e_{t-1}, \dots, e_1, e_0)_2$. The starting will be happened with a point Q in E (F_p), (that is, $Q = \infty$). With the i index that takes the values 0, 1, ..., $t-1$, the computation of $Q = Q + P$ can be done if $e_i = 1$. After then, the value $2P$ is computed and plugging $2P$ by P. The processing continues until the last value $t-1$. Therefore, the last computed value of a point Q is the scalar multiplication point vP [1]. The summary of the RLB method can be given in the following algorithm.

Algorithm 4.1 The RLB algorithm

Input: A scalar v in [1, n-1] and a point P in $E(F_p)$.
Output: A scalar multiplication vP.

```
1. Write down a scalar $v$ as a binary string $v = (e_{t-1}, \ldots, e_1, e_0)_2$.
2. $Q = \infty$.
3. For $i = 0,1,\ldots, t-1$ do
    3.1 If $e_i = 1$ then $Q = Q + P$.
    3.2 Compute $P = 2P$.
    3.3 Else compute $P = 2P$.
    3.4 End if
4. End for
5. Return $Q = vP$.
```

4.2 The left-to-right binary method

With the same parameters E, P, n, and v which are used in the RLB method, the computation of vP using the LRB method can be done easily. A scalar v can be written in a binary representation string $(e_{t-1}, \ldots, e_1, e_0)_2$. Let us start with a point Q in $E(F_p)$, where $Q = \infty$. With the i index which takes the values $t-1,\ldots, 1, 0$, then the computation of $2Q$ can be done and plugged into Q. After then, the value $Q = Q + P$ is computed. The processing continues until the last value 0. Therefore, the last computed value of a point Q is the scalar multiplication point vP. The LRB method can be summarized in Algorithm (4.2) [1].

Algorithm 4.2 The LRB algorithm

Input: A scalar v in $[1, n\text{-}1]$ and a point P in $E(F_p)$.
Output: A scalar multiplication vP.

```
1. Write down a scalar $v$ as a binary string $v = (e_{t-1},\ldots, e_1, e_0)_2$.
2. $Q = \infty$.
3. For $i = t-1,\ldots,1, 0$ do
    3.1 Compute $Q = 2Q$.
    3.2 If $e_i = 1$ then $Q = Q + P$.
    3.3 Else go to step (3.4).
    3.4 End if
4. End for
5. Return $Q = vP$.
```

5. The non-adjacent form for the elliptic scalar multiplication

The motivation to use the signed digit representation of a scalar v, in a scalar multiplication vP, is the computation of the subtraction and addition of the points lying on elliptic curve E which has the same efficient. A signed digit representation of v is given by $v = \sum_{i=0}^{l-1} e_i 2^i$, where $e_i \in \{0, \pm 1\}$ will be explained in this section with more details. The signed digit representation forms the nonadjacent form (NAF) [1, 9, 10] which is given in the next algorithm.

Algorithm 5.1 The NAF computation of a positive integer

Input: A positive integer v in [1, n-1].
Output: The expansion NAF (v).

```
1. $i \leftarrow 0$.
2. While $v \geq 1$ do
   2.1 If $v$ is odd then $e_i \leftarrow 2 - (v \bmod 4)$,
         $v \leftarrow v - e_i$ ;
   2.2 Else: $e_i \leftarrow 0$.
   2.3 End if
3. $v \leftarrow v / 2$, $i \leftarrow i + 1$.
4. End while
5. Return $(e_{i-1}, \ldots, e_1, e_0)$.
```

The computation of a scalar multiplication vP by employing the NAF algorithm can be done using the following algorithm:

Algorithm 5.2 The NAF method for computing the scalar multiplication

Input: A positive integer v in [1, n-1] and $P \in E(F_p)$.
Output: A scalar multiplication vP.

```
1. Algorithm (5.1) uses to compute NAF($v$).
2. $Q \leftarrow \infty$.
3. For $i = t - 1, \ldots, 1, 0$ do
   3.1 $Q \leftarrow 2Q$.
   3.2 If $e_i = 1$ then $Q \leftarrow Q + P$.
   3.3 ElseIf $e_i = -1$ then $Q \leftarrow Q - P$.
```

```
        3.4 Else go to step (3.5).
        3.5 End if
    4. End for
    5. Return ($Q = vP$).
```

6. The graphic methods for the elliptic scalar multiplications

This section discusses the generalization on the binary methods and NAF to compute l-tuple of the scalar multiplications on elliptic curve E defined over prime field Fp. This generalization employed the simple undirected and directed graphs.

6.1 The graphic right-to-left binary (GRLB) method

Suppose Ec is an elliptic curve defined over a prime field F_p [1–7]. The equation of Ec is given by

$$Ec : y^2 = x^3 + ax + b \ (\mathrm{mod}\, p). \qquad (2)$$

Let $P = (x, y)$ be a point that lies on Ec which has a (large) prime order r. Let $G(V, E)$ be a simple (or multigraph or others) graph, where V is a vertex set and E is an edge set. The matrix representation $A(G)$ on $G(V, E)$ is defined as given in Eq. (1). Directly from the rows of the matrix $A(G)$, the binary representation strings $\left(e_{(m-1)_l}, \ldots, e_{1_l}, e_{0_l}\right)_2$ are obtained. The starting will happen with an elliptic point Q_1 which belongs to $E(F_p)$, where $Q_1 = \infty$. With the i index which takes the values $0_1, 1_1, \ldots, (m-1)_1$ in the first row of $A(G)$, the computation of $Q_1 = Q_1 + P$ can be done if $e_{i_1} = 1$. After then, the value $2P$ is computed and plugging it by P. The processing on the first row continues until the last value $m - 1$. Therefore, the last computed value of a point Q_1 is the value of the first scalar multiplication point v_1P in l-tuple $\langle vP \rangle$. In similar way, the processing on others rows can be done. The summary of the GRLB method can be given in the following algorithm:

Algorithm 6.1 The GRLB method

Input: A graph $G(V, E)$, $P \in E(F_p)$, l and m, where l and m are the order and size of a graph G, respectively.
Output: The m-tuple of the scalar multiplications $\langle vP \rangle = \langle v_1P, \ldots, v_lP \rangle$.

```
1. Write down the matrix representation $A(G)$ of the graph $G(V, E)$.
2. Directly determine the binary representation strings $v_j = \left(e_{(m-1)_j}, \ldots, e_{1_j}, e_{0j}\right)_2$
   from $A(G)$.
3. For $j = 1, 2, \ldots, l$.
4.     $Q_j \leftarrow \infty$.
5.     For $i = 0_j : (m-1)_j$ do
```

```
        5.1 If $e_{i_j} = 1$ then $Q_j = Q_j + P$.
        5.2 Else go to step (6).
        5.3 End if
    6.  Compute $P \leftarrow 2P$.
    7.  End for
    8. Return ($Q_j = v_j P$).
    9. End for
  10. Return ($\langle Q \rangle = \langle vP \rangle = \langle v_1 P, v_2 P, \dots, v_l P \rangle$).
```

6.2 The implementation results on the GRLB method

With different kinds of graphs which are given in **Figure 2**, the matrix representations of the graphs have been computed by $A(G_a)$, $A(G_b)$, $A(G_c)$, and $A(G_d)$, respectively.

$$A(G_a) = \left| \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \right. . \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad A(G_b) = \left| \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \right. . \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A(G_c) = \left| \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \right. . \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \text{and } A(G_d) = \left| \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \right. . \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The l-tuple computations of the scalar multiplications that correspond to these graphs are shown in **Table 1**.

6.3 The graphic left-to-right binary method

With the same parameters p, E, P, G, and V which are used in the GRLB method, the computations of l-tuple $\langle vP \rangle$ using the GLRB method can be done easily. The scalars $v_1, \dots, v_n$ can be written in the binary representation strings $\left(e_{(m-1)_j}, \dots, e_{1j}, e_{0_j}\right)_2$, for j = 1, 2, …, l, directly from the matrix representation $A(G)$ of G. Let us start with a point Q_1 in $E(F_p)$, where $Q_1 = \infty$. With the i index which takes the values $(m-1)_1, \dots, 1_1, 0_1$, then the computation of $2Q_1$ can be done and plugged into Q_1. After then, the value $Q_1 = Q_1 + P$ is computed. The processing continues until the last value 0_1. Therefore, the last computed value of a point Q_1 is the first scalar multiplication point in an l-tuple $\langle vP \rangle$. Similarly, the processing on others rows can be computed. The GLRB method can be summarized in Algorithm (6.2).

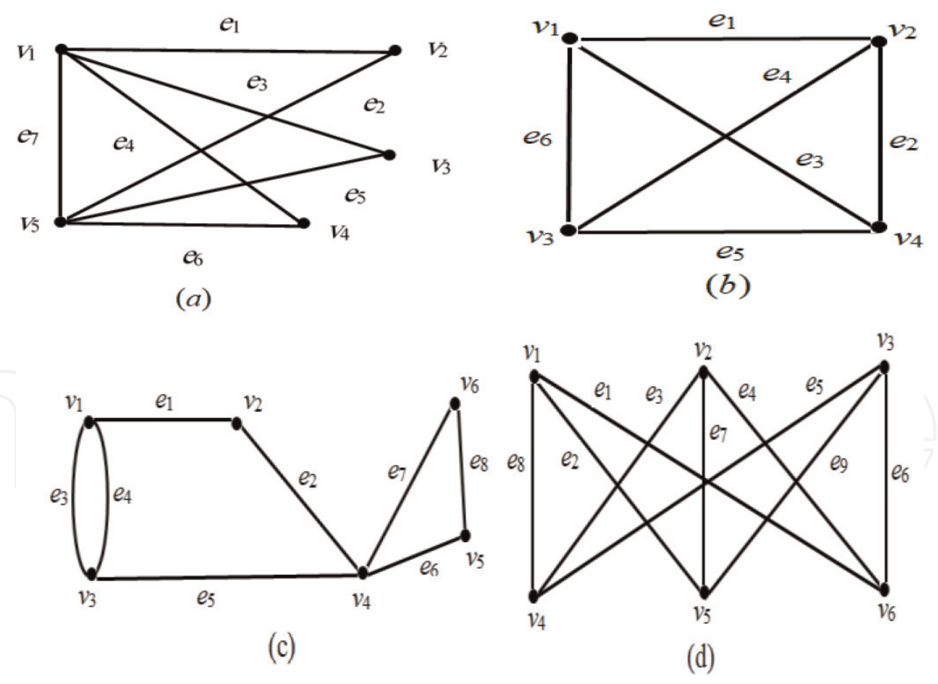

Figure 2.
Different kinds of graphs [16].

P	**E (a,b)**	**N**	**Generator point**	**G**	**G (l,m)**	**⟨vP⟩**
101	E (10,2)	109	P = (68,14)	G_a	G_a (5,7)	$\langle v_1P, v_2P, v_3P, v_4P, v_5P\rangle = \langle(14,19),(91,66),(44,68),(5,51),(93,4)\rangle$
61	E (4,1)	67	P = (24,14)	G_b	G_b (4, 6)	$\langle v_1P, v_2P, v_3P, v_4P\rangle = \langle(0,60),(4,52),(43,21),(0,1)\rangle$
191	E (7,2)	193	P = (41,91)	G_c	G_c (6, 8)	$\langle v_1P, v_2P, v_3P, v_4P, v_5P, v_6P\rangle = \langle(24,137),(41,100),(43,113),(18,109),(16,114),(105,86)\rangle$
449	E (2,2)	467	P = (50,27)	G_d	G_d (6, 9)	$\langle v_1P, v_2P, v_3P, v_4P, v_5P, v_6P\rangle = \langle(93,281),(405,104),(96,20),(266,382),(236,399),(31,391)\rangle$

Table 1.
The experimental results of the l*-tuple of the scalar multiplications that correspond to the graphs* G_a, G_b, G_c, *and* G_d.

Algorithm 6.2 The GLRB method

Input: A graph $G(V,E)$, $P \in E(F_p)$, l and m.
Output: The l-tuple of the scalar multiplications $\langle vP\rangle = \langle v_1P, \dots, v_lP\rangle$.

```
1. Write down the matrix representation $A(G)$ of the graph $G(V,E)$.
2. Directly determine the binary representation strings $v_j = \left(e_{(m-1)_j}, \dots, e_{1_j}, e_{0_j}\right)_2$,
   for $j$=1,2,..., $l$ from $A(G)$.
3. For $j = 1, 2, \dots, l$.
4.     $Q_j \leftarrow \infty$.
5.     For $i = (m-1)_j : 0_j$ do
```

```
            5.1 Compute $Q_j = 2Q_j$.
            5.2 If $e_{i_j} = 1$ then $Q_j = Q_j + P$.
            5.3 Else go to Step (5.4).
            5.4 End if
    6.    End for
    7.    Return $(Q_j = v_j P)$.
8. End for
9. Return $(\langle Q \rangle = \langle vP \rangle = \langle v_1P, v_2P, \ldots, v_lP \rangle)$.
```

7. The digraphic NAF for the elliptic scalar multiplication

The signed digit representation of an l -tuple $\langle v \rangle$ of scalars v_j, which are used to compute an l-tuple $\langle vP \rangle$ of the scalar multiplications v_jP, can be represented directly from the digraphs. The signed digit representations of v_j are given by $v_j = \sum_{i=0}^{l-1} e_{i_j} 2^{i_j}$, where $e_{i_j} \in \{0, \pm 1\}$. The signed digit representations form the generalized nonadjacent form (GNAF). These representations are computed using the following algorithm:

Algorithm 7.1 The GNAF computation of an l-tuple of the positive integers

Input: An l-tuple of positive integers v_j.
Output:$\langle NAF_S(v) \rangle = \langle NAF_S(v_1), NAF_S(v_2), \ldots, NAF_S(v_l) \rangle$.

```
1. Determine $v_j$, $j$ = 1, 2, ...,$l$ and $\left(e_{1j}, e_{2j}, \ldots, e_{m_j}\right)$in any digraph $G$.
2. For $j$ = 1, 2, ..., $l$.
3.     For $i$ = 1,...,$m$.
4.         If $v_s$ is an incident out of $v_t$, where $s, t \in j$
5.             then $e_{i_j} = 1$.
6.         Elseif $v_s$ is an incident into $v_t$
7.             then $e_{ij} = -1$.
8.         Else there is no edge between $v_s$ and $v_t$.
9.             then $e_{i_j} = 0$.
10.        End if
11.    End For
```

```
12.    Return $\left(e_{1j}, e_{2j}, \dots, e_{m_j}\right)$.

13. End For

14. Return $NAF(v_j) = \left(e_{m_j}, \dots, e_{2j}, e_{1j}\right)$.
```

In **Figure 3**, the digraph G has the vertices v_j for $j = 1, 2, 3, 4$ and edges e_m for $m = 1, 2, \dots, 7$.

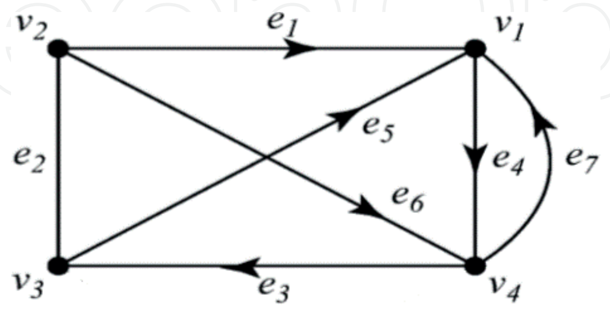

Figure 3.
The digraph has the vertices v_j *for* $j = 1, 2, 3, 4$ *and edges* e_m *for* $m = 1, 2, \dots, 7$.

The incidence matrix of G that is given in **Figure 3** is

$$A = \left|\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\right. \begin{bmatrix} -1 & 0 & 0 & 1 & -1 & 0 & -1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix}.$$

So, the NAF representations of 4-tuple $\langle v_1, v_2, v_3, v_4 \rangle$ are

$$\langle (-1,0,0,1,-1,0,-1), (1,1,0,0,0,1,0), (0,-1,-1,0,1,0,0), (0,0,1,-1,0,-1,1) \rangle.$$

The GNAF method for l-tuple of the scalar multiplications can be performed using Algorithm (7.2).

Algorithm 7.2 The GNAF method for computing l-tuple of the scalar multiplication

Input: The l-tuple of positive integers v_j and $P \in E(F_p)$.
Output: The l-tuple of the scalar multiplications $\langle vP \rangle$.

```
1. Algorithm (7.1) uses to compute GNAF($v$).

2. $Q_j \leftarrow \infty$.

3. For $j = 1, 2, \dots, l$

4. For $i = t - 1, \dots, 1, 0$

    4.1 $Q_j \leftarrow 2Q_j$.
```

```
        4.2 If $e_{i_j} = 1$ then $Q_j \leftarrow Q_j + P$.
        4.3 Elseif $e_{i_j} = -1$ then $Q_j \leftarrow Q_j - P$.
        4.4 Else go to step (4.5).
        4.5 End if
    5. End for
    6. End for
    7. Return $\left\langle Q_j = v_j P \right\rangle$.
```

Using Algorithm (7.2), the final result of 4-tuple of the scalar multiplications is given by

$$\langle v_1P, v_2P, v_3P, v_4P\rangle = \langle(28,32),(46,63),(25,90),(82,15)\rangle.$$

8. The subgraphs for the elliptic scalar multiplication

8.1 The binary representations

Suppose G is a graph and H_i, for $i = 1, 2, 3$ are subgraphs as shown in **Figure 4**.
Next algorithm can be applied for determining the binary representation of any subgraph from a given graph.

Algorithm 8.1 The graphic binary representation of a subgraph from a given graph

Input: A graph $G(V, E)$, where $V = (v_1, v_2, \ldots, v_l)$ and $E = (e_1, e_2, \ldots, e_m)$.
Output: The $\mathrm{BR}_{\mathrm{subgraph}}(v)$.

```
 1. Determine $(v_1, v_2, \ldots, v_k)$ and $(e_1, e_2, \ldots, e_m)$ in any subgraph $H$ of $G$.
 2. $i \leftarrow 0$.
 3. For $j = 0: k$, where $k \le l$.
 4.     If there is an edge between $v_s$ and $v_t$, where $s, t \in j$
 5.         then $e_i = 1$.
 6.     Else there is no edge between $v_s$ and $v_t$.
 7.         then $e_i = 0$.
 8.     End if
 9. $i \leftarrow i + 1$.
10. Return $\mathrm{BR}_{\mathrm{subgraph}} = (e_{m-1}, \ldots, e_1, e_0)_2$.
```

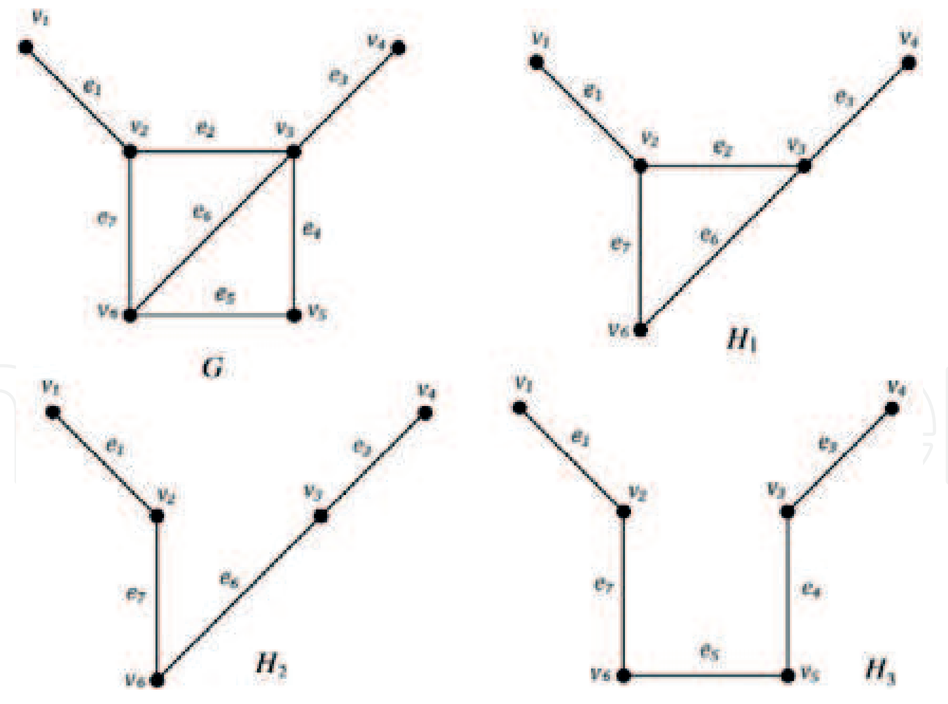

Figure 4.
The subgraphs Hi, for i = 1, 2, 3, for a graph G.

Subgraphs	$(v_1, v_2, \ldots, v_k)$	$(e_1, e_2, \ldots, e_m)$	$\text{BR}_{\text{subgraph}} = (e_{m-1}, \ldots, e_1, e_0)_2$
H_1	$(v_1, v_2, v_3, v_4, v_6)$	$(e_1, e_2, e_3, e_6, e_7)$	(1, 1, 0, 0, 1, 1, 1)
H_2	$(v_1, v_2, v_3, v_4, v_6)$	(e_1, e_3, e_6, e_7)	(1, 1, 0 ,0, 1, 0, 1)
H_3	$(v_1, v_2, v_3, v_4, v_5, v_6)$	$(e_1, e_3, e_4, e_5, e_7)$	(1, 0, 1, 1, 1, 0, 1)

Table 2.
The experimental results of the binary representations of scalars using subgraphs.

p	$E\ (a,b)$	n	Gen Pt P	Subgraph	$\text{BR}_{\text{subgraph}} = (e_{m-1}, \ldots, e_1, e_0)_2$	H_iP
191	E (7,2)	193	P = (41,91)	H_1	(1, 1, 0, 0, 1, 1, 1)	(80,142)
				H_2	(1, 1, 0 ,0 ,1 , 0, 1)	(0,57)
				H_3	(1, 0, 1, 1, 1, 0, 1)	(36,146)

Table 3.
The experimental results for computing of the scalar multiplications based on using the binary representation of the subgraphs.

The small numerical results based on **Figure 4** can be shown in **Table 2**.

On the binary representations which are found directly from the subgraphs, the scalar multiplications H_iP on elliptic curve E defined over a prime field Fp can be computed using Algorithm (4.1) or (4.2). Some experimental results for computing the scalar multiplications based on using the subgraphs to represent the scalars are given in **Table 3**.

9. The signed digit representations

Suppose G is a digraph and H_i, for $i = 1, 2, 3$, are directed subgraphs as shown in **Figure 5**. Algorithm (8.2) can be used to find the signed digit representation of any subgraph from a given graph.

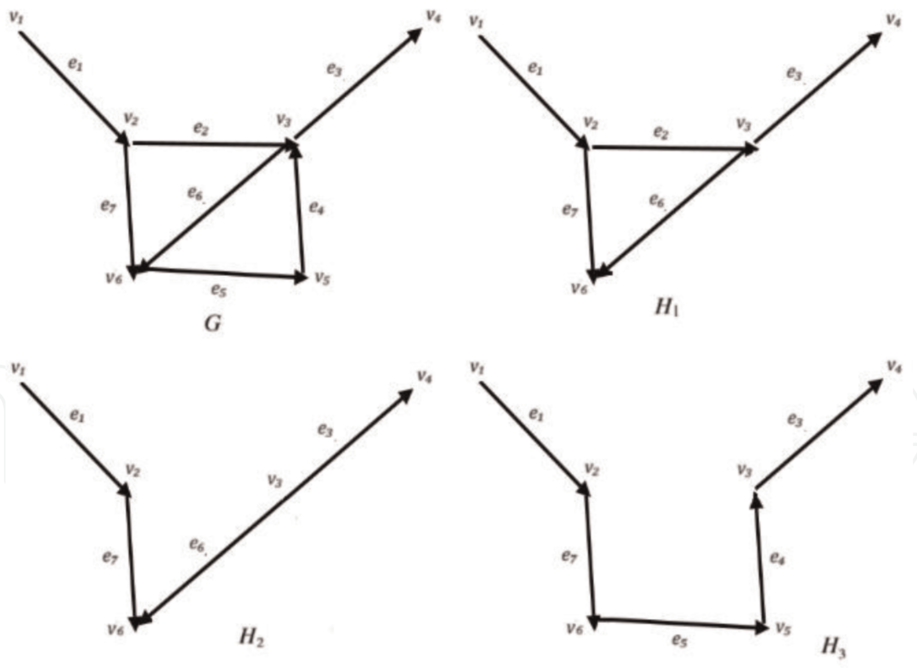

Figure 5.
The directed subgraphs Hi, *for i = 1, 2, 3, 4, for a digraph* G.

Algorithm 8.2 The di-subgraph signed digit representation of the positive integers

Input: A directed graph $G(V, E)$, where $V = (v_1, v_2, \ldots, v_l)$ and $E = (e_1, e_2, \ldots, e_m)$.
Output: The $\mathrm{SDR}_{\mathrm{subgraph}}(v)$.

```
1. Determine $(v_1, v_2, \ldots, v_k)$ and $(e_0, e_1, \ldots, e_{m-1})$ in any subgraph $H$ of $G$.

2. $i \leftarrow 0$.

3. For $j = 0: k$, where $k \quad l$.

4.      If $v_s$ is an incident out of $v_t$, where $s, t \in j$

5.          then $e_i = 1$.

6.      Elseif $v_t$ is an incident into $v_s$

7.          then $e_i = -1$.

8.      Else there is no edge between $v_s$ and $v_t$.

9.          then $e_i = 0$.

10.     End if

11. End for

11. $i \leftarrow i+1$.

12. Return $(e_{m-1}, \ldots, e_1, e_0)$.
```

Subgraphs	*l*-tuple $\langle v \rangle$	*l*-tuple $\langle NAF_S(v) \rangle$
H_1	$\langle v_1, v_2, v_3, v_6 \rangle$	$\langle (1,0,0,0,0,0,0), (-1,1,0,0,0,0,1), (0,-1,1,0,0,1,0), (0,0,0,0,0,-1,-1) \rangle$
H_2	$\langle v_1, v_2, v_3, v_4, v_6 \rangle$	$\langle (1,0,0,0,0,0,0), (-1,0,0,0,0,0,1), (0,0,1,0,0,1,0), (0,0,-1,0,0,0,0), (0,0,0,0,0,-1,-1) \rangle$
H_3	$\langle v_1, v_2, v_3, v_4, v_5, v_6 \rangle$	$\langle (1,0,0,0,0,0,0), (-1,0,0,0,0,0,1), (0,0,1,-1,0,0,0), (0,0,-1,0,0,0,0), (0,0,0,1,-1,0,0), (0,0,0,0,1,0,-1) \rangle$

Table 4.
The experimental results for sign digit representing l*-tuple of the scalars using the subgraphs.*

P	*P*	Directed subgraphs	*l*-tuple $\langle v \rangle$	$\langle vP \rangle$
191	$P = (41,91)$	H_1	$\langle v_1, v_2, v_3, v_6 \rangle$	$\langle (133,91), (171,71), (132,144), (16,77) \rangle$
		H_2	$\langle v_1, v_2, v_3, v_4, v_6 \rangle$	$\langle (133,91), (17,91), (177,186), (177,5), (16,77) \rangle$
		H_3	$\langle v_1, v_2, v_3, v_4, v_5, v_6 \rangle$	$\langle (133,91), (17,91), (49,23), (177,5)\ (79,97), (105,86) \rangle$

Table 5.
The experimental results for computing l*-tuple of the scalar multiplications based on using the subgraphs.*

The computational results based on **Figure 5** and using Algorithm (8.2) are given in **Table 4**. With the signed digit representations which are given in **Table 4**, the l-tuple of the scalar multiplications on elliptic curve E defined over a prime field F_p can be computed. Some experimental results for computing the l-tuple of the scalar multiplications based on using the directed subgraphs to represent the scalars are given in **Table 5**.

10. The computational complexity on the elliptic scalar multiplication methods

This chapter discusses the problems of the computational complexities which are determined depending on the account operations. These operations are the elliptic curve operations, namely, the addition A and doubling D on the points which lie on elliptic curve E defined over a prime field F_p. Also, the finite field operations which are field inversion I, field multiplication M and a field squaring S. The computational complexity problems are determined first of the original binary methods and NAF for computing the scalar multiplications on E. The computational complexities of the proposed methods which are dependent on the graphs and subgraphs are determined as well.

10.1 The computational complexity of the binary methods

Let $\#E\ (F_p) = n$, where n is prime number and it is the nearest number to prime p. A point P in E(Fp) which has order n. Suppose v is a scalar such as v is a randomly selected integer from the interval $[1, n-1]$. The binary representation of v is denoted $(e_{m-1} \ldots e_2.e_1.e_0)_2$ where $m \approx t = \log_2 p$.

The computational complexity of Algorithm (4.1) or (4.2) is roughly $t/2$ point additions and t point doublings, which is denoted by

$$\frac{t}{2}A + tD, \tag{3}$$

in addition to the time of binary representation which is approximately t/2d and t/2S, where d and S are normal addition and squaring. Using Lemmas (6.1) and (6.2) in [18, 19], the points addition A and doubling D can be re-expressed by 1I + 2 M + 1S and 1I + 2 M + 2S, respectively. In other words, the computational complexity of Algorithm (4.1) or (4.2) is expressed in terms of field operations by.

$$3tS + 3tM + 1.5tI + 0.5td. \tag{4}$$

Several computational complexity results to compute a scalar multiplication by applying the binary method are given in **Table 6**.

10.2 The computational complexity of the NAF

With same the multiplier v which belongs to the interval $[1,n-1]$, the computational complexity to compute a scalar multiplication vP using the NAF is given by

$$D + \frac{t}{3}A + tD = \frac{t}{3}A + (t+1)D. \tag{5}$$

In Eq. (5), D in the first term is the cost of NAF to represent a positive integer v, $t/3A$ + tD is the cost of computing a scalar multiplication vP using NAF method, and t is the length of the NAF string. In other words, the running time of Algorithm (5.1) is expressed in terms of field operations by

$$t/3(1I + 2M + 1S) + (t+1)(1I + 2M + 2S) = ((t/3) + t + 1)I + ((2/3)t + 2t + 2)M + ((t/3) + 2t + 2)S. \tag{6}$$

P	*E* (*a*,*b*)	*n*	Gen. pt. *P*	*vP*	Bin. representation	Comp. complexity
101	*E* (10,2)	109	(68,14)	93*P*	(1, 0, 1, 1, 1, 0, 1)	21S + 21 M + 10.5I + 3.5d
61	*E* (4,1)	67	(24,14)	23*P*	(1, 0, 1, 1, 1)	15S + 15 M + 7.5I + 2.5d
113	E (12,4)	103	(52,41)	39*P*	(1, 0, 0, 1, 1, 1)	18S + 18 M + 9I + 4.5d
149	E (13,1)	167	(32,133)	13*P*	(1, 1, 0, 1)	12S + 12 M + 6I + 2d
1031	E (15,7)	1061	(217,808)	281*P*	(1, 0, 0, 0, 1, 1, 0, 0, 1)	27S + 27 M + 13.5I + 4.5d

Table 6.
The experimental results of the computational complexity for the scalar multiplications using the binary method.

P	*E* (*a*,*b*)	*N*	Gen. pt. *P*	*vP*	NAF. rep.	Comp. complexity
101	*E* (10,2)	109	(68,14)	93*P*	(1, 0, −1, 0, 0, −1, 0, 1)	11.6I + 23.3 M + 20.6S
61	*E* (4,1)	67	(24,14)	23*P*	(1, 0, −1, 0, 0, −1)	9I + 18 M + 16S
113	E (12,4)	103	(52,41)	39*P*	(1, 0, −1, −1, 0, 0, −1)	10.3I + 20.6 M + 23S
149	E (13,1)	167	(32,133)	13*P*	(1, 0, 0, −1, −1)	7.6I + 15.3 M + 13.6S
1031	E (15,7)	1061	(217,808)	281*P*	(1, 0, 0, 1, 0, 0, −1, −1, −1)	13I + 26 M + 23S

Table 7.
The experimental results of the computational complexity for the scalar multiplications using the NAF method.

Some numerical results of the computational complexity to compute a scalar multiplication using the NAF method are given in **Table 7**.

11. The computational complexity for serial computing l-tuple of the scalar multiplications

11.1 The computational complexity of the serial GBR

On l-tuple of the scalar multiplications $\langle vP\rangle = \langle v_1P, v_2P, \ldots, v_lP\rangle$, the computations of v_1P, v_2P, ..., v_lP without using the graphs or subgraphs can be done serially. So, the computational cost of these computations using the binary representations of v_1, v_2, ..., v_l is given by

$$\frac{t}{2}lA + tlD + 0.5tld. \tag{7}$$

In other words, the running time can be expressed in terms of field operations by

$$3tlS + 3tlM + 1.5tlI + 0.5tld. \tag{8}$$

Table 8 displays some small experimental results for computational complexities for serial computations of l-tuples $\langle vP\rangle$using the generalized binary method.

11.2 The computational complexity of the serial GNAF

The computational complexity for computing l-tuple of the scalar multiplications using GNAF representations in serial way is given by

$$lD + \frac{t}{3}lA + tlD = \frac{t}{3}lA + (t+1)lD. \tag{9}$$

Using the field operations, the formula in Eq. (9) can be rewritten by.

$$((\mathrm{t}/3) + \mathrm{t} + 1)l\mathrm{I} + ((2/3)\mathrm{t} + 2\,\mathrm{t} + 2)l\mathrm{M} + ((\mathrm{t}/3) + 2\,\mathrm{t} + 2)l\mathrm{S}. \tag{10}$$

The computational complexity results for serial computations of l-tuples $\langle vP\rangle$ using the GNAF method are given in **Table 9**.

P	$E\ (a,b)$	n	Gen. pt. P	$\langle vP\rangle$	Comp. complexity
101	E (10,2)	109	(68,14)	$\langle 93P, 25P, 66P\rangle$	63S + 63 M + 31.5I + 10.5d
61	E (4,1)	67	(24,14)	$\langle 23P, 19P, 12P\rangle$	45S + 45 M + 22.5I + 7.5d
113	E (12,4)	103	(52,41)	$\langle 39P, 21P\rangle$	36S + 36 M + 18I + 9d
149	E (13,1)	167	(32,133)	$\langle 13P, 5P\rangle$	24S + 24 M + 12I + 4d
1031	E (15,7)	1061	(217,808)	$\langle 281P, 91P, 63P, 55P\rangle$	108S + 108 M + 54I + 18d

Table 8.
The experimental results for computational complexities for serial computations of l*-tuples* $\langle vP\rangle$ *using the generalized binary method.*

P	*E* (*a*,*b*)	*N*	Gen. pt. *P*	$\langle vP \rangle$	Comp. complexity
101	*E* (10,2)	109	(68,14)	$\langle 93P, 25P, 66P \rangle$	34.8I + 69.9 M + 61.8S
61	*E* (4,1)	67	(24,14)	$\langle 23P, 19P, 12P \rangle$	27I + 54 M + 48S
113	E (12,4)	103	(52,41)	$\langle 39P, 21P \rangle$	20.6I + 41.2 M + 46S
149	E (13,1)	167	(32,133)	$\langle 13P, 5P \rangle$	15.2I + 30.6 M + 27.2S
1031	E (15,7)	1061	(217,808)	$\langle 281P, 91P, 63P, 55P \rangle$	52I + 104 M + 92S

Table 9.
The experimental results of the computational complexities for the serial computations of l-tuples $\langle vP \rangle$ *using the GNAF.*

12. The computational complexity of the graphic elliptic scalar multiplication methods

Suppose $\langle vP \rangle = \langle v_1P, v_2P, \ldots, v_lP \rangle$ is an l-tuple of the scalar multiplications. The graphic computations of v_1P, v_2P, ..., v_lP can be done using the graphs or subgraphs in two ways. One of them is using the graphs directly to find the binary representations of the scalars v_1, v_2, ..., v_l, whereas another one uses the digraphs to represent these scalars. The computational costs of these computations can be discussed as follows.

12.1 The computational complexity of the graphic binary representation (GBR)

Using the graphs to compute l-tuple of the scalar multiplications costs

$$\frac{t}{2}lA + tlD. \tag{11}$$

In terms of field operations, the computational complexity of GBR can be expressed by

$$3tlS + 3tlM + 1.5tlI. \tag{12}$$

Table 10 displays some small experimental results for computational complexities for the graphic representations of l-tuples $\langle vP \rangle$ using the generalized binary method.

P	*E* (*a*,*b*)	*n*	Gen. pt. *P*	$\langle vP \rangle$	C_{GBR} using graphic representations
101	*E* (10,2)	109	(68,14)	$\langle 93P, 25P, 66P \rangle$	63S + 63 M + 31.5I
61	*E* (4,1)	67	(24,14)	$\langle 23P, 19P, 12P \rangle$	45S + 45 M + 22.5I
113	E (12,4)	103	(52,41)	$\langle 39P, 21P \rangle$	36S + 36 M + 18I
149	E (13,1)	167	(32,133)	$\langle 13P, 5P \rangle$	24S + 24 M + 12I
1031	E (15,7)	1061	(217,808)	$\langle 281P, 91P, 63P, 55P \rangle$	108S + 108 M + 54I

Table 10.
The experimental results for computational complexities for graphic computations of l-*tuples* $\langle vP \rangle$ *using the generalized binary method.*

P	$E\,(a,b)$	N	Gen. pt. P	$\langle vP\rangle$	C_{GBR} using graphic representations
101	E (10,2)	109	(68,14)	$\langle 93P, 25P, 66P\rangle$	32I + 64 M + 56S
61	E (4,1)	67	(24,14)	$\langle 23P, 19P, 12P\rangle$	24I + 48 M + 42S
113	E (12,4)	103	(52,41)	$\langle 39P, 21P\rangle$	18.6I + 37.3 M + 32.6S
149	E (13,1)	167	(32,133)	$\langle 13P, 5P\rangle$	13.3I + 26.6 M + 23.3S
1031	E (15,7)	1061	(217,808)	$\langle 281P, 91P, 63P, 55P\rangle$	48I + 96 M + 84S

Table 11.
The experimental results for computational complexities for graphic computations of l-tuples of ⟨vP⟩ using the GNAF method.

12.2 The computational complexity of the digraphic NAF

The computational complexity for computing l-tuple of the scalar multiplications using the digraphs is given by

$$\frac{t}{3}lA + tlD. \tag{13}$$

Eq. (13) can be rewritten using field operations by:

$$((\mathrm{t}/3) + \mathrm{t})l\mathrm{I} + ((2/3)\mathrm{t} + 2\,\mathrm{t})l\mathrm{M} + ((\mathrm{t}/3) + 2\,\mathrm{t})l\mathrm{S}. \tag{14}$$

Several experimental results for computational complexities for digraph representations of l-tuples $\langle vP\rangle$ are given in **Table 11**.

13. Computational complexity comparison on the serial and graphic computations of GBR and GNAF methods

This section discusses first the experimental results of the GBR method that uses serial computations to calculate l-tuple of the scalar multiplications and the GBR method that depends directly on using the graphs. Selecting the scalars v_1, v_2,.... v_l from the interval [1. n − 1] to represent using the GBR method which needs the cost 0.5tld, where t is the length of the string binary representation, l is the length of the tuple and d is a normal addition operation. The final computational cost as given in Eq. (8).

Whereas, the binary representing of the scalars v_1, v_2, ... v_l can be taken directly from graphs or subgraphs without need to extra cost. This saves the 0.5tld operations to compute l-tuple of the scalar multiplications $\langle vP\rangle$. The total cost of the graphic GBR method has been determined previously in Eq. (12). The serial GBR and graphic GBR computational costs for several experimental results are given in **Table 12**. In this table, one can see the serial GBR method with various values of p is more costly compared to the graphic GBR method.

Also, the experimental results of the serial GNAF and graphic GNAF methods that are used to calculate l-tuple of the scalar multiplications are discussed in this section. Selecting the scalars v_1, v_2, ... v_l from the interval [1. n − 1] to represent using the GNAF method which needs the 1lI + 2lM + 2lS cost, l is the length of the tuple, M is a field multiplication, S is a field squaring, and I is a field inversion. So, the total computational cost as given in Eq. (10).

The graphic GNAF of the scalars v_1, v_2, ... v_l can be taken directly from graphs. So it can save 1lI + 2lM + 2lS operations for computing l-tuple of the scalar multiplications $\langle vP\rangle$. The total cost of the graphic GNAF method is determined

P	$E\ (a,b)$	N	Gen. pt. P	C_{GBR} using serial computations	C_{GBR} using graphs
101	E (10,2)	109	(68,14)	63S + 63 M + 31.5I + 10.5d	63S + 63 M + 31.5I
61	E (4,1)	67	(24,14)	45S + 45 M + 22.5I + 7.5d	45S + 45 M + 22.5I
113	E (12,4)	103	(52,41)	36S + 36 M + 18I + 9d	36S + 36 M + 18I
149	E (13,1)	167	(32,133)	24S + 24 M + 12I + 4d	24S + 24 M + 12I
1031	E (15,7)	1061	(217,808)	108S + 108 M + 54I + 18d	108S + 108 M + 54I

Table 12.
The computational costs of the serial GBR and graphic GBR with different values of p.

P	$E\ (a,b)$	N	Gen. pt. P	$Cost_{GNFA}$ using serial computations	$Cost_{GNFA}$ using graphs
101	E (10,2)	109	(68,14)	34.8I + 69.9 M + 61.8S	32I + 64 M + 56S
61	E (4,1)	67	(24,14)	27I + 54 M + 48S	24I + 48 M + 42S
113	E (12,4)	103	(52,41)	20.6I + 41.2 M + 46S	18.6I + 37.3 M + 32.6S
149	E (13,1)	167	(32,133)	15.2I + 30.6 M + 27.2S	13.3I + 26.6 M + 23.3S
1031	E (15,7)	1061	(217,808)	52I + 104 M + 92S	48I + 96 M + 84S

Table 13.
The computational costs of the serial GNAF and graphic GNAF with different values of p.

previously in Eq. (14). Several experimental results on the serial GNAF and graphic GNAF computational costs are given in **Table 13**. With various values of p as shown in **Table 13**, it can observe that the graphic GNAF method is less costly than the serial GNAF method.

14. Conclusions

The present chapter was concerned with presenting new graphic elliptic scalar multiplication algorithms for speeding up the computations of the scalar multiplication defined on elliptic curves over a prime field in different ways. These ways employed the undirected graphs and subgraphs to construct the binary representations of the scalars v in the scalar multiplications vP. Also, the sign digit representation of v has been obtained directly from using the digraphs or di-subgraphs. These representations are used to compute one scalar multiplication vP and l-tuple $<vP>$ of the scalar multiplications. The computational complexities of the proposed graphic elliptic scalar multiplication algorithms have been determined. The computational complexity comparison of the proposed algorithms and original ones is discussed based on the elliptic curve and field operations. The experiment results of the computational complexities show that the proposed algorithms are less costly for computing the scalar multiplication or l-tuple of the scalar multiplications than original algorithms which are dependent on the computations of the binary representations or NAF expansions. The new propositions with graphic representations speed up the computations on elliptic scalar multiplication algorithms. Also, it gives the generalized cases with the computations of the l-tuples $<vP>$ using (undirected or directed) graphs or subgraphs. This insight makes the working with graphic elliptic scalar multiplication algorithms more efficient in comparison with the serial original ones.

Author details

Ruma Kareem K. Ajeena
Mathematics Department, University of Babylon, Education College for Pure Sciences, Babil, Iraq

*Address all correspondence to: ruma.usm@gmail.com

References

[1] Hankerson D, Menezes AJ, Vanstone S. Guide to Elliptic Curve Cryptography. New York: Springer; 2004

[2] Hoffstein J et al. An Introduction to Mathematical Cryptography. Vol. 1. New York: Springer; 2008

[3] Oswald E. Introduction to Elliptic Curve Cryptography. Institute for Applied Information Processing and Communication, Graz University Technology; 2002

[4] Gross JL, Yellen J. Graph Theory and Its Applications. Chapman and Hall/CRC; 2005

[5] Lopez J, Dahab R. An Overview of Elliptic Curve Cryptography. 2000

[6] Miller VS. Use of elliptic curves in cryptography. In: Conference on the heory and Application of Cryptographic Techniques. Berlin, Heidelberg: Springer; 1985

[7] Kapoor V, Abraham VS, Singh R. Elliptic curve cryptography. Ubiquity. 2008

[8] Karthikeyan E. Survey of elliptic curve scalar multiplication algorithms. International Journal of Advanced Networking and Applications. 2012;**4**(2)

[9] Brown M et al. Software implementation of the NIST elliptic curves over prime fields. In: Cryptographers' Track at the RSA Conference. Berlin, Heidelberg: Springer; 2001

[10] Dimitrov V, Imbert L, Mishra PK. Efficient and secure elliptic curve point multiplication using double-base chains. In: International Conference on the Theory and Application of Cryptology and Information Security. Berlin, Heidelberg: Springer; 2005

[11] Eisenträger K, Lauter K, Montgomery PL. Fast elliptic curve arithmetic and improved Weil pairing evaluation. In: Cryptographers' Track at the RSA Conference. Berlin, Heidelberg: Springer; 2003

[12] Ciet M et al. Trading inversions for multiplications in elliptic curve cryptography. Designs, Codes and Cryptography. 2006;**39**(2):189-206

[13] Mishra PK, Dimitrov V. Efficient quintuple formulas for elliptic curves and efficient scalar multiplication using multibase number representation. In: International Conference on Information Security. Berlin, Heidelberg: Springer; 2007

[14] Ajeena RKK, Kamarulhaili H. Point multiplication using integer sub-decomposition for elliptic curve cryptography. Applied Mathematics & Information Sciences. 2014;**8**(2):517

[15] Ajeena RKK, Kamarulhaili H. A hybrid approach for elliptic scalar multiplication. AIP Conference Proceedings. Vol. 1660. No. 1. AIP Publishing; 2015

[16] Ray SS. Graph theory with algorithms and its applications. In: Applied Science and Technology. Springer Science & Business Media; 2012

[17] Vasudev C. Graph Theory with Applications. New Age International; 2006

[18] Ajeena RKK, Kamarulhaili Hailiza. The computational complexity of elliptic curve integer sub- decomposition (ISD) method. I: AIP Conference Proceedings. Vol. 1605. No. 1. AIP; 2014

[19] Ajeena RKK. Integer Sub-decomposition (ISD) Method for Elliptic Curve Scalar Multiplication [Diss]. Universiti Sains Malaysia; 2015