

Mathematical analysis of the computational complexity of integer sub-decomposition algorithm

Ruma Kareem K. Ajeena¹ and Hailiza Kamarulhaili²

^{1,2} School of Mathematical Sciences, Universiti Sains Malaysia, 11800, Penang, Malaysia

E-mail: ruma.usm@gmail.com & hailiza@cs.usm.my

Abstract. In this paper, the computational complexity to compute a scalar multiplication on classes of elliptic curve over a prime field that have efficiently-computable endomorphisms is analyzed mathematically. This scalar multiplication is called integer sub-decomposition (ISD) method, which is based on the GLV method of Gallant, Lambert and Vanstone that was initially proposed in the year 2001. In this work, the mathematical proof of the computational cost of ISD implementation is presented. The computational cost of ISD algorithm has been proven based on the operations counting. Two types of the operations are used, namely, elliptic curve operations represented by elliptic curve point addition A and elliptic curve point doubling D and finite field operations represented by field inversion I , field multiplication M and field squaring S that design the computation of the running time of ISD scalar multiplication.

Key words and phrases: Elliptic curves, scalar multiplication, efficiently-computable endomorphisms, integer sub-decomposition, computational complexity.

1. Introduction

The attractive features of elliptic curves history awarded it studying by mathematicians over a hundred of years to solve a variety of problems. The entry of these curves into cryptography independently by Neal Koblitz [1] and Victor Miller [2] in 1985 who suggested elliptic curve public key cryptosystems. The elliptic curves performance has active importance in the security level as a traditional asymmetric cryptosystem, such as RSA [3],[4]. The fundamental step of elliptic curve cryptosystems is to compute elliptic curve scalar multiplication kP for a point P which has a large prime order n . To accomplish this end, various methods have been innovated, adopting on elliptic curves E over finite fields. A group of methods cleverly employs a distinguished endomorphism $\psi \in \text{End}(E)$ to split a large computation into a sequence of cheaper ones, so that the overall computational cost will be lowered [3].

Recently, Gallant, Lambert and Vanstone [5],[6],[7] used such a technique that, contrary to the previous ones, also applied to special curves defined over large prime fields. Their method uses an efficiently computable endomorphism $\psi \in \text{End}(E)$ to rewrite kP as

$$kP = k_1P + k_2\psi(P), \text{ with } \max\{|k_1|, |k_2|\} = O(\sqrt{n}). \quad (1.1)$$

Their key point is an algorithm, that will be called the GLV method, which inputs integers n and $\lambda \in [1, n-1]$ and produces for any $k \pmod{n}$, two residues k_1 and $k_2 \pmod{n}$ such that

$$k = k_1 + \lambda k_2 \pmod{n}. \quad (1.2)$$



Starting with analyzing the GLV method of Gallant, Lambert and Vanstone, our study uses two fast endomorphisms with minimal polynomials $p_j(X) = X - \lambda_j \pmod{n}$, for $j = 1, 2$ to compute any multiple kP of a point P of order n lying on an elliptic curve. When both values or one of them is not bounded by $\pm\sqrt{n}$, the value k is then decomposed into the values k_1 and k_2 . The sub-decomposition from $k = k_1 + k_2\lambda \pmod{n}$ is shown clearly as follows:

$$k_1 = k_{11} + k_{12}\lambda_1 \pmod{n} \quad \text{and} \quad k_2 = k_{21} + k_{22}\lambda_2 \pmod{n}. \quad (1.3)$$

We calculate, in particular, the integer sub-decomposition (ISD) as follows:

$$\begin{aligned} kP &= k_{11}P + k_{12}[\lambda_1]P + k_{21}P + k_{22}[\lambda_2]P \\ &= k_{11}P + k_{12}\psi_1(P) + k_{21}P + k_{22}\psi_2(P). \end{aligned} \quad (1.4)$$

where $-C\sqrt{n} < k_{11}, k_{12}, k_{21}, k_{22} < C\sqrt{n}$, where $C > 1$ as proven in [9].

The mathematical proofs of the computational complexity to compute a scalar multiplication kP on elliptic curve over a finite field F_p are determined in this paper. This determination depends on the elliptic curve operations and finite field operations that design the computation of the running time of scalar multiplication. The fast computation of addition and doubling on an elliptic curve indicates an efficient scalar multiplication algorithm that is performed based on these sped-up computations. On the other hand, using endomorphism applications can be applied to improve performance in elliptic curve scalar multiplication, such as in the GLV decomposition and ISD sub-decomposition of a scalar k in a scalar multiplication kP .

The elliptic curve model and coordinate model used as well as the algorithm applied for computing a scalar multiplication kP consider as the three principal factors to determine the computational cost of a scalar multiplication kP . The outline of this paper shows: Section 2 gives a summary of the mathematical background to clarify elliptic curve E over prime field and endomorphisms on it. Section 3 reviews the procedure of ISD scalar multiplication method. Section 4 presents the mathematical proof of the computational complexity of ISD algorithm that includes the mathematical proofs of all sub-algorithms which form ISD method through two types of operations, namely, elliptic curve operations and finite field operations. Finally, Section 5 draws the concluding remarks.

2. Mathematical Background

2.1. Elliptic Curves over Prime Fields

Definition 2.1. Let $p \neq 2, 3$. An elliptic curve $E(F_p)$ over F_p , is defined by an equation of the form [8]:

$$E : Y^2 = X^3 + AX + B \pmod{p}, \quad (2.5)$$

where $A, B \in F_p$. The curve E is said to be non-singular if it has no double zeroes, that means the discriminant $D_E = 4A^3 + 27B^2 \neq 0 \pmod{p}$.

2.2. Endomorphisms of Elliptic Curve over Prime fields

Assume that E is an elliptic curve defined over the finite field F_p . The point at infinity is denoted by O_E . The set of F_p -rational points on E forms the group $E(F_p)$. A rational map $\psi : E \rightarrow E$ satisfies $\psi(O_E) = O_E$ dubbed an endomorphism of E . The endomorphism ψ will be defined over F_q where $q = p^n$, if the rational map is defined over F_q . Therefore, clearly, for any $n \geq 1$, ψ is a group homomorphism of $E(F_p)$ and also of $E(F_q)$ [3] and [8].

Definition 2.2. The endomorphism of elliptic curve E defined over F_q is the m -multiplication map $[m] : E \rightarrow E$ defined by

$$P \rightarrow mP \quad (2.6)$$

for each $m \in \mathbb{Z}$. The negation map $[-1] : E \rightarrow E$ defined by $P \rightarrow -P$ is a special case from m -multiplication map [3].

Theorem 2.3. (*Hasse Theorem*). Let E be an elliptic curve over a finite field F_p [3]. Then, the order of $E(F_p)$ satisfies

$$|p + 1 - \#E(F_p)| \leq 2\sqrt{p}. \quad (2.7)$$

Definition 2.4. The rectangle norm [4] of (x, y) is defined by $\max\{|x|, |y|\}$. We denote it by $|(x, y)|$.

3. Integer sub-decomposition (ISD) method

The idea of GLV method is the main source on which the ISD method depends to obtain an efficient scalar multiplication on an ordinary elliptic curve E defined in equation (2.5). This method primarily aims to sub-decompose the values k_1 and k_2 when one or both values are not bounded by $\pm\sqrt{n}$. Decomposing a scalar $k \in [1, n-1]$ in a scalar multiplication kP gives two new scalars k_1 and k_2 . According to GLV idea the values of k_1 and k_2 satisfy the main GLV condition, that is $\max\{|k_1|, |k_2|\} \leq \sqrt{n}$ on the interval $[1, n-1]$. However, generically this decomposition produces also new scalars k_1 and k_2 lie outside the range \sqrt{n} on the same interval $[1, n-1]$. In other words, there are other scalars k_1 and k_2 on the same interval $[1, n-1]$ satisfy $\max\{|k_1|, |k_2|\} > \sqrt{n}$. But with the last condition, the GLV idea to compute kP cannot work. So, this gap considers as a main reason to sub-decompose k_1 and k_2 that lie outside the range \sqrt{n} through our proposed method called Integer sub-decomposition (ISD) that is based on new sub-scalars k_{11}, k_{12} and k_{21}, k_{22} of scalars k_1 and k_2 respectively. The sub-decomposition is expressed by these formulas:

$$k_1 = k_{11} + k_{12}\lambda_1(\text{mod } n) \text{ and } k_2 = k_{21} + k_{22}\lambda_2(\text{mod } n),$$

as shown in Equation (1.3). To accomplish sub-decomposition, one should first find a GLV generator $\{v_1, v_2\}$ by using a GLV generator algorithm in [7] for a given n and λ , where n is a large prime order of elliptic curve point P and λ is a root of the characteristic polynomial of endomorphism ψ of E . Consequently, $k \in [1, n-1]$ is decomposed into k_1 and k_2 . This decomposition can be performed using the balanced length-two representation of a multiplier k algorithm in [3]. Our proposed algorithm can then be used to generate the ISD generators $\{v_3, v_4\}$ and $\{v_5, v_6\}$ such that each component of v_3, v_4 and v_5, v_6 is bounded by \sqrt{n} and relatively prime to each other. These generators can be easily computed by solving the shortest vector problem in a lattice that is involved in using an extended Euclidean algorithm in [5]. k_1 and k_2 can be decomposed again into integers k_{11}, k_{12} and k_{21}, k_{22} which means that the sub-decomposition of k is as follows:

$$k = k_{11} + k_{12}\lambda_1 + k_{21} + k_{22}\lambda_2(\text{mod } n) \quad (3.8)$$

with $-C\sqrt{n} < k_{11}, k_{12}, k_{21}, k_{22} < C\sqrt{n}$ from any ISD generators $\{v_3, v_4\}$ and $\{v_5, v_6\}$. Finally, the scalar multiplication kP can be computed as follows.

$$\begin{aligned} kP &= k_{11}P + k_{12}[\lambda_1]P + k_{21}P + k_{22}[\lambda_2]P \\ &= k_{11}P + k_{12}\psi_1(P) + k_{21}P + k_{22}\psi_2(P). \end{aligned}$$

as shown in Equation (1.4). This formula demonstrates our proposal which includes computing two endomorphisms, namely, $\psi_1(P) = \lambda_1P$ and $\psi_2(P) = \lambda_2P$, where $P \in E(F_q)$, $\lambda_1, \lambda_2 \in [1, n-1]$ and $\lambda_1 \neq \pm\lambda_2$. The implementation results for computing ISD scalar multiplication kP are provided using Algorithm (1). This algorithm first finds the shortest integer vectors v_1 and v_2 using GLV algorithm (1) in [7] that consists from extended Euclidean algorithm (EEA) in the first part or necessary condition in the second part. These vectors consider as the reduced integer lattice points. The next step of ISD algorithm is to implement balanced length-two representation of a multiplier algorithm (3.74) in [3] to decompose k into k_1 and k_2

depending on the coordinates of the vectors v_1 and v_2 and computing $c_1 = \lfloor -t_{m+2}k/n \rfloor$ and $c_2 = \lfloor t_{m+1}k/n \rfloor$ with $v_1 = (r_{m+1}, -t_{m+1})$ and $v_2 = (r_{m+2}, -t_{m+2})$ are the shortest vector that help us to compute $k_1 = k - c_1r_{m+1} - c_2r_{m+2}$ and $k_2 = -c_1(-t_{m+1}) - c_2(-t_{m+2})$.

Algorithm 1 Computing ISD Scalar Multiplication Through Interleaving Based on wNAF Expansions.

Input: The integers p, n, λ, w_j with $j = 1, 2, 3, 4$ and a point $P = (x, y)$ on E .

Output: kP .

Computation stage:

Run GLV generator Algorithm (1) that is given in [7] to find the generator $\{v_1, v_2\}$ such that $v_1 \leftarrow (r_{m+1}, -t_{m+1})$ and $v_2 \leftarrow (r_m, -t_m)$ or $v_2 \leftarrow (r_{m+2}, -t_{m+2})$.

Choose $k \in [1, n-1]$.

Run balanced length-two representation of a multiplier algorithm (3.74) that given in [3] to decompose k into k_1 and k_2 .

if $\max\{|k_1|, |k_2|\} > \sqrt{n}$ **then**

Set $\lambda_1, \lambda_2 \in [1, n-1]$

Precomputation stage:

The precomputation of the endomorphism $\psi_1(P) = \lambda_1 P$ and $\psi_2(P) = \lambda_2 P$.

Computation stage:

Run ISD generators Algorithm (1) given in [9], [10] with input $(n, \lambda_1, \lambda_2)$ to find $\{v_3, v_4\}$ and $\{v_5, v_6\}$ such that $v_3 \leftarrow (r_{(m+1)_1}, -t_{(m+1)_1})$, $v_4 \leftarrow (r_{m_1}, -t_{m_1})$ or $v_4 \leftarrow (r_{(m+2)_1}, -t_{(m+2)_1})$ and $v_5 \leftarrow (r_{(m+1)_2}, -t_{(m+1)_2})$, $v_6 \leftarrow (r_{m_2}, -t_{m_2})$ or $v_6 \leftarrow (r_{(m+2)_2}, -t_{(m+2)_2})$, respectively.

Use Algorithm (2) in [9], [10] to sub-decompose k_1 and k_2 into $k_1 \equiv k_{11} + k_{12}\lambda_1 \pmod{n}$ and $k_2 \equiv k_{21} + k_{22}\lambda_2 \pmod{n}$, respectively.

return k_{11}, k_{12} and k_{21}, k_{22} .

Use generalized computing wNAF Algorithm (1) or (2) in [11] to compute $w_j \text{NAF}$ expansions for j from 1 to 4 of integers k_{11}, k_{12}, k_{21} and k_{22} .

Use extended interleaving Algorithm (3) or (4) in [11] to compute $kP = k_{11}P +_E k_{12}\psi_1(P) + k_{21}P +_E k_{22}\psi_2(P)$.

else

Stop and go to choose another value of k .

end if

4. Mathematical analysis of the computational complexity ISD method

The computational cost of the ISD method is discussed in this section mathematically based on presenting the proof of the computational costs of all sub-algorithms that form the ISD algorithm.

4.1. Computational complexity for GLV generator algorithm

Generating the GLV generator $\{v_1, v_2\}$, where v_1 and v_2 are linearly independent vectors, can be performed through the GLV generator algorithm (1) provided in [7]. GLV generator algorithm consists of two parts, namely, the extended Euclidean algorithm (EEA) shown in [5] to generate shortest vectors in two dimensions, $v_1 = (a_1, b_1)$ and $v_2 = (a_2, b_2)$ or the second part that is the necessary condition part (NCP). The following sections discuss the computational complexity for the said parts.

- **Computational Complexity of the Extended Euclidean Algorithm.**

An extended Euclidean algorithm (EEA) is a useful way to compute the reduction lattice in two dimensions to determine short vectors v_i for $i = 1, 2$ for a given n and λ . The computational cost of extended Euclidean algorithm C_{EEA} can be determined by computing the field operations that provide the following result:

Lemma 4.1. *For a prime number n and positive integer λ such that $n \geq \lambda$ and $\gcd(n, \lambda) = 1$, a tuple of Equations $s_i n + t_i \lambda = r_i$ for $i = 0, 1, 2, \dots, m, m+1, m+2, \dots, z-1$ that results from EEA exists. The computational complexity of extended Euclidean algorithm C_{EEA} to determine the tuple of variables r_i, t_i and s_i for $i = 0, 1, 2, \dots, m, m+1, m+2, \dots, z-1$ with $z > m$ is*

$$C_{EEA} = Z(I + 3M), \quad (4.9)$$

where Z is the dimension of the returned variables r_i, t_i and s_i ; M is a field multiplication; and I is a field inversion.

Proof. A reasonable way to determine the computational cost of extended Euclidean algorithm C_{EEA} is by implementing the EEA provided as follows:

```

 $u \leftarrow \lambda, v \leftarrow n.$ 
 $t_1 \leftarrow 1, s_1 \leftarrow 0, t_0 \leftarrow 0, s_0 \leftarrow 1, r_0 \leftarrow n, r_1 \leftarrow \lambda.$ 
While  $u \neq 0$ 
 $q \leftarrow \lfloor v/u \rfloor, r \leftarrow v - qu, t \leftarrow t_0 - qt_1, s \leftarrow s_0 - qs_1.$ 
 $v \leftarrow u, u \leftarrow r, t_0 \leftarrow t_1, t_1 \leftarrow t, s_0 \leftarrow s_1, s_1 \leftarrow s.$ 
Endwhile

```

The implementation of this algorithm for a loop operation shows the running time in field operations that has the cost $1I + 3M$. Repeating the implementation for another loop operation yields another cost $1I + 3M$. Thus, the cost of two loops is

$$1I + 3M + 1I + 3M = 2I + 6M = 2(1I + 3M). \quad (4.10)$$

The third iteration costs

$$1I + 3M + 1I + 3M + 1I + 3M = 3I + 9M = 3(1I + 3M).$$

The iteration of the loops continues until $r = 0$ and also obtains $u = 1$. If the last loop reaches Z times, then the total cost of this algorithm is

$$\underbrace{1I + 3M + \dots + 1I + 3M}_{Z\text{-times}} = ZI + 3ZM = Z(I + 3M) \quad (4.11)$$

□

• Computational Complexity of the Necessary Condition Part of a GLV generator algorithm.

The study of the necessary condition part (NCP) of the algorithm GLV generator provided in algorithm (1) in [7]. Determining the step cost can finally identify the total cost of the NCP of GLV algorithm by

$$ZI + 3ZM + 1I + 2M + 1I + 3M + 4M = Z(I + 3M) + 2I + 9M, \quad (4.12)$$

where M is a field multiplication, I is a field inversion and Z is the demission of sequence of the variables r_i, t_i and s_i for $i = 0, 1, 2, \dots, m, m+1, m+2, \dots, z-1$ with $z > m$.

4.2. Computational Complexity of Decomposing a Scalar k

The following result can be used to show the computational cost of decomposing a scalar k .

Lemma 4.2. *Let E be a prime curve and P be a point on E that has prime order n . The scalar $k \in [1, n-1]$ of scalar multiplication kP decomposes into new scalars k_1 and k_2 such that $k \equiv k_1 + k_2\lambda \pmod{n}$, where λ is a root of characteristic polynomial $p(X) = X - \lambda$ of endomorphism ψ . Thus, the computational cost of this decomposition is*

$$C_{\text{Decomposing } k} = \begin{cases} Z(I + 3M) + 4S + 6M + 2I, & \text{with using EEA,} \\ Z(I + 3M) + 4S + 4I + 15M, & \text{with using NCP.} \end{cases} \quad (4.13)$$

Proof. Let n be a prime order of P which is a point on elliptic curve E . Assume that ψ is a nontrivial endomorphism that has characteristic polynomial $p(X) = X - \lambda \pmod{n}$. The computational cost of decomposition k that was written by sum formula $k \equiv k_1 + k_2\lambda \pmod{n}$ as defined in Equation (1.2) requires the exact computational cost of GLV generator contraction. This contraction is performed by applying the EEA to determine two vectors, v_1 and v_2 . The components of each vector are determined based on a tuple of variables r_i and t_i for $i = 0, 1, 2, \dots, m, m+1, m+2, \dots, z-1$ with $z > m$. The cost to find a tuple of variables r_i and t_i is $Z(I + 3M)$ as shown in Lemma (4.1). Identifying the largest index $r_m \geq \sqrt{n}$ provides a possibility to write $v_1 = (r_{m+1}, -t_{m+1})$ and select v_2 between the shortest vectors $(r_m, -t_m)$ and $(r_{m+2}, -t_{m+2})$. Selecting a vector v_2 can be conducted by verifying the condition defined in Algorithm (1) in [7], Step 3. Computing this condition requires $4S$ operations.

Furthermore, the cost of decomposing a scalar k depends on computing $c_1 = \lfloor -t_{m+2}k/n \rfloor$ and $c_2 = \lfloor t_{m+1}k/n \rfloor$ that both have a cost of $2M + 2I$. Moreover, determining $k_1 = k - c_1r_{m+1} - c_2r_{m+2}$ and $k_2 = c_1t_{m+1} + c_2t_{m+2}$ costs $2M$ for each one. Therefore, the total computational complexity of decomposing k is

$$C_{\text{Decomposing } k} = \begin{cases} Z(I + 3M) + 4S + 6M + 2I, & \text{with using EEA,} \\ Z(I + 3M) + 4S + 4I + 15M, & \text{with using NCP.} \end{cases}$$

□

4.3. The Computational Complexity of the ISD Generators

In this section, the computational complexity of the ISD generators is provided. This cost is determined based on one of the costs, the cost of the generalization of the extended Euclidean algorithm GEEA because further sub-decomposition in ISD method needs further extension of the EEA that was earlier used in GLV method. For that we have generated the EEA and develop a modified algorithm to perform the sub-decomposition process. On the other hand, cost of the necessary condition part (NCP) of ISD generators algorithm (1) given in [9], [10] can be used to determine the cost of the ISD generators.

- **The Computational Complexity of the ISD Generators Based on the Generalization of the Extended Euclidean Algorithm GEEA.**

Employing the generalization of the extended Euclidean algorithm GEEA to determine two tuples of variables used to form new generators called ISD generators is possible. These computations include identifying linearly independent vectors v_3, v_4 and v_5, v_6 in the $\ker T$, which considers the integer lattice points in two dimensions. The components in each vector must satisfy the conditions represented by the components of each vector being less than \sqrt{n} and the relation between the components in each vector is a relative prime to form ISD

generators $\{v_3, v_4\}$ and $\{v_5, v_6\}$.

When the output vectors v_3, v_4 and v_5, v_6 from implementation of the GEEA cannot satisfy these conditions, applying the NCP shown in the ISD generators Algorithm (1) in [9], [10] to generate these vectors that satisfy these conditions is necessary, especially the vectors v_4 and v_6 . Therefore, identifying the computational cost of ISD generators is conducted by recognizing the cost of the ISD generators algorithm that consist of the GEEA in the first part or computing the NCP in the second part.

The computational cost to compute ISD generators is provided by the following result based on the GEEA.

Theorem 4.3. *Let n be a prime number, the positive integers $\lambda_j \in [1, n-1]$ for $j = 1, 2$ with $\gcd(n, \lambda_j) = 1$ and $\lambda_1 \neq \pm\lambda_2$. The generalization of the extended Euclidean algorithm (GEEA) for (n, λ_1) and (n, λ_2) produces two tuples of variables $r_{i_1}, t_{i_1}, s_{i_1}$ for $i = 0, 1, \dots, m_1, (m+1)_1, (m+2)_1, \dots, (z-1)_1$ and $r_{i_2}, t_{i_2}, s_{i_2}$ for $i = 0, 1, \dots, m_2, (m+1)_2, (m+2)_2, \dots, (z-1)_2$ to find linearly independent vectors v_3, v_4 and v_5, v_6 which generate ISD generators $\{v_3, v_4\}$ and $\{v_5, v_6\}$. Then the computational cost to form ISD generators, when the algorithm uses GEEA that is given in the first part of ISD generators algorithm (1) given in [9], [10], is*

$$C_{ISD \text{ generators}} = \begin{cases} Z(I + 3M) + 4S, & \text{if } Z = Z_1 = Z_2 \text{ when use GEEA,} \\ \max(Z_1, Z_2)(I + 3M) + 4S, & \text{if } Z_1 \neq Z_2 \text{ when use GEEA.} \end{cases} \quad (4.14)$$

where Z is a dimension of two tuples of variables $r_{i_1}, t_{i_1}, s_{i_1}$ which have the same dimension Z_1 and $r_{i_2}, t_{i_2}, s_{i_2}$ that have dimension Z_2 , namely when $Z_1 = Z_2 = Z$, I is a field inversion, M is a field multiplication and S is a field squaring.

Proof. The proof is presented in [12]. □

• The Computational Complexity of the ISD Generators Algorithm Implemented Based on NCP.

Computing the ISD generator algorithms (1) given in [9], [10] can be implemented with the necessary condition defined in the second part of this algorithm. The computational cost of ISD generators can be determined as follows.

$$C_{NCP} = Z(I + 3M) + 2I + 9M \quad (4.15)$$

when two tuples of the variables r_{i_j}, t_{i_j} and s_{i_j} for $j = 1, 2$ $i = 0, 1, 2, \dots, m_j, (m+1)_j, (m+2)_j, \dots, (z-1)_j$, with $z_j > m_j$ have same dimension $Z_1 = Z_2$, M is a field multiplication, and I is a field inversion. Otherwise, when these two tuples of the variables have different dimensions Z_1 and Z_2 such that $Z_1 \neq Z_2$, then the computational cost of the NCP is

$$C_{NCP} = \max(Z_1, Z_2)(I + 3M) + 2I + 9M. \quad (4.16)$$

Thus, from Equations (4.15) and (4.16), the final computational complexity of the ISD generators can be rewritten by formula

$$C_{ISD \text{ generators}} = \begin{cases} Z(I + 3M) + 2I + 9M + 4S, & \text{if } Z = Z_1 = Z_2, \text{ when use NCP} \\ \max(Z_1, Z_2)(I + 3M) + 2I \\ + 9M + 4S, & \text{if } Z_1 \neq Z_2, \text{ when use NCP.} \end{cases} \quad (4.17)$$

where $4S$, in Equation (4.17) is the cost to determine the shorter vectors v_4 and v_6 .

The computational complexity of ISD generators algorithm can be rewritten by

$$C_{ISD \text{ generators}} = \begin{cases} Z(I + 3M) + 4S, & \text{if } Z = Z_1 = Z_2 \text{ and use GEEA,} \\ \max(Z_1, Z_2)(I + 3M) + 4S, & \text{if } Z_1 \neq Z_2 \text{ and use GEEA,} \\ Z(I + 3M) + 2I + 9M + 4S, & \text{if } Z = Z_1 = Z_2 \text{ and use NCP,} \\ \max(Z_1, Z_2)(I + 3M) + 2I + 9M + 4S, & \text{if } Z_1 \neq Z_2 \text{ and use NCP.} \end{cases} \quad (4.18)$$

4.4. The Computational Complexity of Sub-Decomposition of Scalars k_1 and k_2

The computational complexity to sub-decompose scalars k_1 and k_2 is given by the following result.

Theorem 4.4. *Let E be a prime curve and P be a point on E which has a prime order n . The scalar $k \in [1, n - 1]$ of a scalar multiplication kP was decomposed into k_1 and k_2 with $\max(|k_1|, |k_2|) > \sqrt{n}$. The ISD sub-decomposition decomposes k_1 and k_2 into new sub-scalars k_{11}, k_{12}, k_{21} and k_{22} such that $k \equiv k_{11} + k_{12}\lambda_1 + k_{21} + k_{22}\lambda_2 \pmod{n}$ with $\max(|k_{11}|, |k_{12}|) < C\sqrt{n}$ and with $\max(|k_{21}|, |k_{22}|) < C\sqrt{n}$. Then the computational complexity to find k_{11}, k_{12}, k_{21} and k_{22} is*

$$C_{Sub-Decomposition \ k_1 \text{ and } k_2} = \begin{cases} Z(I + 3M) + 4S + 6M + 2I, & \text{if } Z_1 = Z_2, \\ \max(Z_1, Z_2)(I + 3M) + 4S + 6M + 2I, & \text{if } Z_1 \neq Z_2, \end{cases} \quad (4.19)$$

when the generation of ISD generators uses GEEA. Or

$$C_{Sub-Decomposition \ k_1 \text{ and } k_2} = \begin{cases} Z(I + 3M) + 4S + 4I + 15M, & \text{if } Z_1 = Z_2, \\ \max(Z_1, Z_2)(I + 3M) + 4S + 4I + 15M, & \text{if } Z_1 \neq Z_2, \end{cases} \quad (4.20)$$

when the generation of ISD generators uses NCP.

Proof. The proof is presented in [12]. □

4.5. The Computational Complexity for Parallel Pre-computation of Two Efficient Computable Endomorphisms

The computational complexity to pre-compute two endomorphisms ψ_1 and ψ_2 which are defined by $\psi_1(P) = \lambda_1 P$ and $\psi_2(P) = \lambda_2 P$ that act on the subgroup $\langle P \rangle$ generated by a point P that lies on E is given by the following result:

Theorem 4.5. *Suppose ψ_1 and ψ_2 are two nontrivial separable endomorphisms on a prime curve E which are defined by $\psi_1(P) = \lambda_1 P$ and $\psi_2(P) = \lambda_2 P$ for an elliptic curve point P and $\lambda_j \in [1, n - 1]$ is a root of its characteristic polynomial $p_j(X) = X - \lambda_j$, $j = 1, 2$ and $\lambda_1 \neq \pm \lambda_2$. Then the cost to compute such endomorphisms, $C_{\psi_1(P)}$ and $C_{\psi_2(P)}$, is:*

$$C_{\psi_1(P) \ \& \ \psi_2(P)} = (\max(\lambda_1, \lambda_2) - 1)I + 2(\max(\lambda_1, \lambda_2) - 1)M + (\max(\lambda_1, \lambda_2) - 1)S. \quad (4.21)$$

Proof. The proof is presented in [12]. □

4.6. The Computational Complexity for computing the w_j NAF expansions of Positive Integers.

In this section, the computational cost to represent four positive integers a_j for $j = 1, 2, 3, 4$ using w_j NAF expansions is provided by the following result:

Theorem 4.6. *Let w_j for $j = 1, 2, 3, 4$ be width windows. Suppose the w -NAF expansions of positive integers of scalar a_j that form a scalar k of ISD scalar multiplication kP with P is a point on elliptic curve E , are represented. Then the computational complexity for computing w -NAF expansions of a_j simultaneously is*

$$C_{w_jNAF_{a_j}} = 4D + \sum_{j=1}^4 (2^{w_j-2} - 1)A, \quad (4.22)$$

where D and A indicate to point doubling and point addition on elliptic curve E over F_p .

Proof. For the proof, see [12]. □

4.7. The Computational Complexity of the Interleaving Method Based on w_j NAF Expansions.

The computational cost of the interleaving method based on w_j NAF expansions for $j = 1, 2, 3, 4$ is provided by the following result.

Theorem 4.7. *Let w_j , for $j = 1, 2, 3, 4$, be width windows that are positive integers ≥ 2 and a_j be any positive integers. Then the computational cost of the interleaving method that based on w_j NAF expansions is*

$$C_{Interleaving} = \left[4D + \sum_{j=1}^4 (2^{w_j-2} - 1)A \right] + \left[\max_{j=1:4} l_j D + \sum_{j=1}^4 \frac{l_j}{w_j + 1} A \right]. \quad (4.23)$$

where A indicates to elliptic curve point addition, D denotes an elliptic curve point doubling and l_j indicates to the length of $NAF_{w_j}(a_j)$.

Proof. For the proof, see [12]. □

Finally, the computational complexity of ISD method is defined by this result.

Theorem 4.8. *Let E be an elliptic curve over finite field F_p and P be a point on E which has prime order n . Let ψ be a nontrivial endomorphism defines by $\psi_j(P) = \lambda_j P$ where λ_j are roots of characteristic polynomials $p_j(X) = X^2 - \lambda_j X + 1 \pmod{n}$ of ψ_j for $j = 1, 2$. The scalar $k \in [1, n-1]$ of scalar multiplication kP that has ISD sub-decomposition such that $kP = k_{11}P +_E k_{12}\psi_1(P) +_E k_{21}P +_E k_{22}\psi_2(P)$ with $\max\{|k_{11}|, |k_{12}|\} < C\sqrt{n}$ and $\max\{|k_{21}|, |k_{22}|\} < C\sqrt{n}$. Then, generally, the computational complexity of ISD method is. In particular, the computational complexity of ISD method take one of the following costs.*

I. *The computational cost of ISD algorithm through the implementation GEEA to generate ISD generators and with using one interleaving method [11] is*

$$C_{ISD \text{ Method}} = \begin{cases} 2Z(I + 3M) + 8S + 12M + 4I, & \text{if } Z_1 = Z_2 \\ Z(I + 3M) + \max(Z_1, Z_2)(I + 3M) + 8S + 12M + 4I, & \text{if } Z_1 \neq Z_2 \end{cases} \\ + 4D + \sum_{j=1}^4 (2^{w_j-2} - 1)A + \max_{j=1,2,3,4} l_j D \\ + \sum_{j=1}^4 \frac{l_j}{w_j + 1} A. \quad (4.24)$$

II. *The computational cost of ISD algorithm through the implementation NCP to generate ISD generators and with using one interleaving method [11] is*

$$C_{ISD \text{ Method}} = \begin{cases} 2Z(I + 3M) + 8S + 30M + 8I, & \text{if } Z_1 = Z_2 \\ Z(I + 3M) + \max(Z_1, Z_2)(I + 3M) + 8S + 30M + 8I, & \text{if } Z_1 \neq Z_2 \end{cases} \\ + 4D + \sum_{j=1}^4 (2^{w_j-2} - 1)A + \max_{j=1,2,3,4} l_j D \\ + \sum_{j=1}^4 \frac{l_j}{w_j + 1} A. \quad (4.25)$$

III. *The computational cost of ISD algorithm through the implementation GEEA to generate ISD generators and with using two interleaving method [11] is*

$$C_{ISD \text{ Method}} = \begin{cases} 2Z(I + 3M) + 8S + 12M + 4I, & \text{if } Z_1 = Z_2 \\ Z(I + 3M) + \max(Z_1, Z_2)(I + 3M) + 8S + 12M + 4I, & \text{if } Z_1 \neq Z_2 \end{cases} \\ + \max \left[\left(2D + \sum_{j=1}^2 (2^{w_j-2} - 1)A \right) + \left(\max_{j=1:2} l_j D \right. \right. \\ \left. \left. + \sum_{j=1}^2 \frac{l_j}{w_j + 1} A \right), \left(2D + \sum_{j=3}^4 (2^{w_j-2} - 1)A \right) \right. \\ \left. + \left(\max_{j=3:4} l_j D + \sum_{j=3}^4 \frac{l_j}{w_j + 1} A \right) \right]. \quad (4.26)$$

IV. *The computational cost of ISD algorithm through the implementation NCP to generate ISD generators and with using two interleaving method [11] is*

$$C_{ISD \text{ Method}} = \begin{cases} 2Z(I + 3M) + 8S + 30M + 8I, & \text{if } Z_1 = Z_2 \\ Z(I + 3M) + \max(Z_1, Z_2)(I + 3M) + 8S + 30M + 8I, & \text{if } Z_1 \neq Z_2 \end{cases} \\ + \max \left[\left(2D + \sum_{j=1}^2 (2^{w_j-2} - 1)A \right) + \left(\max_{j=1:2} l_j D \right. \right. \\ \left. \left. + \sum_{j=1}^2 \frac{l_j}{w_j + 1} A \right), \left(2D + \sum_{j=3}^4 (2^{w_j-2} - 1)A \right) \right. \\ \left. + \left(\max_{j=3:4} l_j D + \sum_{j=3}^4 \frac{l_j}{w_j + 1} A \right) \right]. \quad (4.27)$$

Proof. The proof is presented in [12]. □

5. Conclusion

This paper presents the computational complexity of the sub-decomposition method (that is, ISD) of a scalar k in a scalar multiplication kP on the elliptic curve E over a prime finite field F_p . The novelty of this work relies on the theoretical part of the computational cost of the ISD method which has been proven mathematically by computing the elliptic curve operations and finite field operations that give the executing time for implementation of this method. Determining the computational cost in the ISD method depends on computing the cost of all sub-algorithms used in this method.

In particular, the type of implemented sub-algorithm determines the cost of this method. For instance, the computational cost of the ISD method was determined by four formulas, namely, the GEEA, NCP, one, and two interleaving methods as shown in Table (1).

Table 1. The cost types of ISD method.

Implementation Type	Cost of ISD Method
GEEA with one interleaving	Type I
NCP with one interleaving	Type II
GEEA with two interleaving	Type III
NCP with two interleaving	Type IV

6. Acknowledgments

The authors would like to express their gratitude to the Fundamental Research Grant (FRGS), 203/PMATHS/6711320, funded by the Ministry of Higher Education and the School of Mathematical Sciences, Universiti Sains Malaysia.

References

- [1] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of computation*, **48**, (1987), pp. 203-209.
- [2] V. Miller, Use of elliptic curves in cryptography, in *Advances in Cryptology-CRYPTO'85 Proceedings*, (1986), pp. 417-426.
- [3] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*, Springer Verlag, USA, (2004).
- [4] F. Sica, M. Ciet, and J.-J. Quisquater, Analysis of the Gallant-Lambert-Vanstone method based on efficient endomorphisms: Elliptic and hyperelliptic curves, in *Selected areas in cryptography*, (2003), pp.21-36.
- [5] R. Gallant, R. Lambert, and L. Vanstone, Faster point multiplication on elliptic curves with efficient endomorphisms, in *Advances in Cryptology-CRYPTO 2001*, (2001), pp.190-201.
- [6] M. Ciet, J.-J. Quisquater, and F. Sica, Preventing differential analysis in GLV elliptic curve scalar multiplication, *Cryptographic Hardware and Embedded Systems-CHES 2002*, (2003), pp. 1-13.
- [7] D. Kim, S. Lim, Integer decomposition for fast scalar multiplication on elliptic curves, in *Selected Areas in Cryptography*, Springer, (2003), pp.13-20.
- [8] L. C. Washington, *Elliptic curves: number theory and cryptography*, Chapman & Hall/CRC, USA, (2008).
- [9] R.K.K. Ajeena, H. Kamarulhaili, 2013. Analysis on The Elliptic Scalar Multiplication Using Integer Sub-Decomposition Method. *International Journal of Pure and Applied Mathematics*, 87(1): 95-114.
- [10] R.K.K. Ajeena, H. Kamarulhaili, 2014. Point Multiplication using Integer Sub-Decomposition for Elliptic Curve Cryptography. *Journal of Applied Mathematics & Information Sciences*, 8(2): 517-525.
- [11] R.K.K. Ajeena, H. Kamarulhaili. Accelerating Integer Sub-Decomposition for Elliptic Scalar Multiplication using w NAF Expansion Method. *Malaysian Journal of Mathematical Sciences*.
- [12] Ruma Kareem K. Ajeena. Integer Sub-Decomposition Method for Elliptic Curve Scalar Multiplication. PhD thesis, November, 2014, School of Mathematical sciences, Universiti Sains Malaysia, Penang, Malaysia.