Malaysian Journal of Mathematical Sciences 18(3): 501–514 (2024)
https://doi.org/10.47836/mjms.18.3.03

Malaysian Journal of Mathematical Sciences

Journal homepage: https://mjms.upm.edu.my

Fast Endomorphisms in Integer Sub-Decomposition Method on Secp192k1

Antony, S. N. F. M. A.*[1], Yion, C. H. K.[1], Kamarulhaili, H.[1], Ariffin, M. R. K.[2], and Yunos, F.[2]

[1]*School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM Penang, Malaysia*
[2]*Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia*

E-mail: farwina@usm.my
**Corresponding author*

Received: 17 November 2023
Accepted: 25 March 2024

Abstract

Elliptic curve cryptography involves numerous scalar multiplications, incurring high operational costs. In view of this, fast endomorphism is used to represent scalar multiplications, kP on elliptic curves. In the past, techniques such as Gallant-Lambert-Vanstone (GLV) method and Integer Sub-Decomposition (ISD) method have been proposed to reduce the cost of scalar multiplication on elliptic curves by using fast endomorphism. The GLV method employs a single-layer decomposition, breaking k into k_1 and k_2, while the ISD method uses a bilayer decomposition. The existence of fast endomorphisms which are constructed based on the concept of isogeny increase the computational efficiency of the GLV approach and reduce the operation count on the ISD method. This paper embeds the fast endomorphisms in the scalar multiplications on one of the family of elliptic curves with j-invariant 0, E_0, which is the 192-bit Koblitz curve (Secp192k1). The performance of the ISD method in computing certain scalar multiplications on Secp192k1 in conjunction with fast endomorphisms and other various techniques such as binary representation, NAF representation, w-NAF and sliding windows are computed. The results demonstrated that the ISD method together with fast endomorphism, yields the most promising outcomes. This underscores the advantages of using fast endomorphisms in the ISD method on E_0.

Keywords: elliptic curve scalar multiplication; fast endomorphism; GLV method; ISD method; Secp192k1.

1 Introduction

Cryptography is a platform that provides confidentiality, integrity, authentication and disallows repudiation during communication between two parties in a public network. There are two types of cryptography which are symmetric cryptography and asymmetric cryptography or also known as Public Key Cryptosystems (PKC). Among examples of popular and widespread utilized PKC is the Elliptic Curve Cryptosystem (ECC).

ECC is a cryptosystem which is based on a chosen elliptic curve, E and its security relies on the algebraic structure of the curve since it was first introduced by Miller [18] in 1985 and Koblitz [14] in 1987. E can be categorized according to its characteristic field, $char(K)$. When $char(K) \neq 2, 3$, the elliptic curve is known as the ordinary elliptic curve which has the form of

$$E(K) : y^2 = x^3 + Ax + B,$$

where $A, B \in \mathbb{F}_p$ and $4A^3 + 16B^2 \neq 0$ [19, 21]. One of the algebraic structures of an elliptic curves is its points form a group. This group of points has an order $\#E(\mathbb{F}_p) = nh$, where h is the cofactor with $h \leq 4$ for cryptographic use, and n is the order of the largest prime subgroup. The largest prime subgroup consists of points that form an abelian group, known as the maximal order.

Apart from that, the algebraic structures of an elliptic curve help to classify the family of elliptic curves. Another important algebraic structures of an elliptic curve is its j-invariant, defined by

$$j(E) = 1728 \left(\frac{4A^3}{4A^3 + 27B^2} \right).$$

The j-invariant of elliptic curve helps to distinguish the isomorphism that exists between two curves. Two elliptic curves, E_1 and E_2 have the same algebraic structures whenever $j(E_1) = j(E_2)$ and are isomorphic to each other. One of the type of the family of elliptic curves is E_0, the elliptic curve with j-invariant 0.

The complexity of solving the discrete logarithm problem (DLP) attributes to the security of ECC, which requires finding scalar k such that $Q = kP$ where Q is the public key, P is the parameter that has been agreed upon by both parties and k is the secret key. The parameters P and Q refer to points on an elliptic curve belonging to the prime subgroup of order n, while k is the private key such that $k \in [1, n]$. It is difficult to compute k when P and Q are given. Meanwhile, the process of computing $Q = kP$ is easy and it is called the scalar multiplication of elliptic curves, which is known as the most expensive operation in elliptic curve.

There are a few approaches to compute kP where they forced on improving the numeric expansion of k [4]. One of the approaches is by encoding k into binary form which consists of $\{0, 1\}$. Other than encoding k into a binary representation, k can also be encoded into other forms such as non-adjacent form (NAF) [13]. This approach reduces the computational time as it considers the subtraction of points. The NAF-representation consists of $\{0, 1, -1\}$, where no two non-zero digits are adjacent to each other. Another approach is to represent k using the width-w NAF representation [13]. The width-NAF is an extension of NAF representation which involves processing w digits from the integer k at each step. Additionally, a sliding window technique can be applied to the NAF representation of k. It ensures that the value in the window is always odd, thereby minimizing precomputation for the algorithm. However, the operation cost by using encoding approach will increase when one dealing with a bigger prime field and hence reduces the efficiency of ECC.

Besides encoding k into other representations, endomorphisms can be used to compute kP. An endomorphism is a homomorphism defined by

$$\Phi(P) = \Phi(x, y) = (f_1(x, y), f_2(x, y)) = (x', y') = Q,$$

where $P, Q \in E$. It acts as a catalyst to jump the computation from P to λP, where λ is the value obtained from the endomorphism. However, to compute kP for any scalar k, the endomorphism's approach still needs to be combined with other encoding approaches.

One of the methods which combine these two approaches is the Gallant-Lambert-Vanstone (GLV) method. Instead of computing kP directly, they suggested to compute shorter scalars k_1 and k_2 which are obtained from k by using fast endomorphisms. The computational cost is reduced by 50% as long k_1 and k_2 satisfy the GLV condition where $max\{|k_1|, |k_2|\} \leq \sqrt{n}$ [12, 8]. By satisfying this condition, the bitlength of k is reduced to half [20]. However, not all decomposed scalars k_1, k_2 can satisfy the condition.

To further reduce this computational cost, the GLV method is extended into higher dimensions and applied on a larger field such as F_{p^2} [10]. In 2009, Galbraith et al. [11] proposed a Frobenius endomorphism as their fast endomorphism that applied on F_{p^2}. Their method, known as the Galbraith-Lin-Scott (GLS) approach, reduces the computational time of scalar multiplication by 16% to 30% as compared to scalar multiplication without using fast endomorphism. Based on this idea, Zhou et al. [23] proposed a higher dimension of GLV-GLS approach on a certain family of the elliptic curve, namely elliptic curves with j-invariant 0, E_0. They applied their method on E_0 defined over 128-bit prime p. Their method accelerated the computation by 10.3% as compared to the GLS approach. Later in 2012, Longa and Sica proposed a four-dimensional GLV-GLS approach [17]. In 2013, Bos et al. [7] came up with the idea of eight dimensional GLV-GLS decomposition to further accelerate the computation. In 2018, Kwon et al. [16] implemented the four dimensional GLV-GLS approach into several microcontrollers.

However, all these GLV variants need to satisfy the same GLV condition. In contrast to this GLV condition, Ajeena and Kamarulhaili proposed to further decompose scalars k_1 and k_2 into $k_{1,1}$, $k_{1,2}$, $k_{2,1}$ and $k_{2,2}$ in their Integer Sub-Decomposition (ISD) method by using another two endomorphisms [1, 2, 3]. They choose a random $\lambda \in [1, n-1]$ to define their trivial endomorphism, $\Phi(P) = \lambda P$, instead of using fast endomorphisms [22]. As a result, their method is less efficient especially when a bigger λ is chosen or when an elliptic curve is defined over a large prime field, as λP needs to be computed by an encoding approach.

Our Contribution. Using the algebraic structures of an elliptic curve, this study puts forward fast endomorphisms defined on a family of elliptic curves with j-invariant 0, E_0. These constructed endomorphisms can be applied in any scalar decomposition method, especially the ISD method, to further accelerate the computation. Based on the constructed endomorphisms, we compute the computational time for each defined scalar multiplications by using various encoding approaches and also fast endomorphisms. To highlight the significance of this work, we present the algorithm for the naive ISD method and the algorithm for the ISD method with the defined constructed endomorphisms on Koblitz curves with 192-bit, as stated by Standards for Efficient Cryptography [6]. Then, we compared the computational time taken by the ISD method using various encoding approaches and also by using fast endomorphism. Based on our result, the existence of additional fast endomorphisms accelerates the computation of λP by 99%. Through our computations using the ISD method on Secp192k1, we observe that the existence of fast endomorphisms enhance the computation speed by 15%.

Outline of This Paper. This paper is divided into eight sections. Section 1 gives some introductions to elliptic curve cryptography. This section also discusses some previous works in elliptic curve scalar multiplication. Section 2 explains the encoding process of scalar k into a binary form and NAF representation. This section also discusses various processes of point multiplication by using these two representations as fundamental frameworks. Section 3 explains the mechanism of the GLV and ISD methods where we presented the algorithm for the ISD method. Section 4 gives some descriptions of elliptic curves with j-invariant 0, E_0. Section 5 discusses the implementation of fast endomorphism on Secp192k1. Section 6 gives the analysis of the result which includes the computational time taken by the naive ISD method and the improved ISD method on Secp192k1. Section 7 and Section 8 provide the discussion and conclusion, respectively.

2 Encoding Scalar k

As indicated in Section 1, the scalar k can be encoded into binary form and the scalar multiplication kP can be accomplished by using the Right-to-Left algorithm, which is detailed in the following Algorithm 1:

Algorithm 1 : Right-to-Left algorithm for point multiplication [13].

```
Require: $P \in E(\mathbb{F}_p)$, $k = (k_{(\ell-1)}, \cdots, k_1, k_0)_2$.
Ensure: $kP$.
  1 : $Q \leftarrow \mathcal{O}_E$ for $i \leftarrow 0$ to $\ell - 1$ do
  2 : If $k_i = 1$, then $Q \leftarrow Q + P$.
  3 : Else: $P \leftarrow 2P$.
  4 : Return $Q$.
```

The amount of addition and doubling operations required is determined by the Hamming weight w and the bit length ℓ from the binary form of k.

Other than that, one can also represent k in NAF-representation as written in the following Algorithm 2:

Algorithm 2 : Computing NAF(k) [13].

```
Require: A positive integer $k$.
Ensure: $k = (k_{(i-1)}, \cdots, k_1, k_0)$.
  1 : $i \leftarrow 0$.
  2 : While $k \geq 1$ do
  2.1 : If $k$ odd, then: $k_i \leftarrow 2 - k \pmod 4$; $k \leftarrow k - k_i$.
  2.2 : Else: $k_i \leftarrow 0$.
  3 : $k \leftarrow \frac{k}{2}$, $i \leftarrow i + 1$.
  4 : Return: $(k_{(i-1)}, \cdots, k_1, k_0)$.
```

The NAF-representation can be used to compute kP. The following Algorithm 3 explains the computation of kP using NAF-representation.

Algorithm 3 : Point multiplication using NAF-representation of positive integer k [13].

```
Require: A positive integer $k$, $P = (x, y)$ in $E(\mathbb{F}_p)$.
Ensure: $Q = kP$.
  1 : Use Algorithm 2 to compute NAF($k$).
  2 : $Q \leftarrow \infty$, $-P = (x, -y)$.
  3 : For $i \leftarrow \ell - 1$ to 0 do
  4 : $Q \leftarrow 2Q$, $-P = (x, -y)$.
  4.1 : If $k_i = 1$, then: $Q \leftarrow Q + P$.
  4.2 : If $k_i = -1$, then: $Q \leftarrow Q - P$.
  5 : Return: $(Q)$.
```

Besides using the NAF-representation, an alternative approach is to represent the integer k using the width-w NAF-representation, as shown in Algorithm 4. It involves processing w digits of the integer k at each step which leading to a reduction in the overall execution time.

Algorithm 4 : Computing width-w NAF(k) [13].

```
Require: A positive integer $k$, width of the window $w$.
Ensure: $k = (k_{(i-1)}, \cdots, k_1, k_0)$.
  1 : $i \leftarrow 0$.
  2 : While $k \geq 1$ do
  2.1 : If $k$ odd, then: $k_i \leftarrow k \bmod 2^w$; $k \leftarrow k - k_i$.
  2.2 : Else: $k_i \leftarrow 0$.
  3 : $k \leftarrow \frac{k}{2}$, $i \leftarrow i + 1$.
  4 : Return: $(k_{(i-1)}, \cdots, k_1, k_0)$.
```

The following Algorithm 5 provides a step-by-step explanation of how to calculate kP using the width-w NAF-representation.

Algorithm 5 : Point multiplication with width-w NAF-representation of positive integer k [13].

```
Require: A positive integer $k$, $P = (x, y)$ in $E(\mathbb{F}_p)$, width of the window $w$.
Ensure: $Q = kP$.
  1 : Use Algorithm 4 to compute width-$w$ NAF($k$).
  2 : Compute $P_j = jP$ for $j \in \{1, 3, \cdots, 2^{w-1} - 1\}$.
  3 : $Q \leftarrow \infty$.
  4 : For $i \leftarrow \ell - 1$ to 0 do
  4.1 : $Q \leftarrow 2Q$.
  4.2 : If $k_i \neq 0$, then: $Q \leftarrow Q - P$.
  4.2.1 : If $k_i > 0$, then: $Q \leftarrow Q + P_{k_i}$.
  4.2.2 : Else: $Q \leftarrow Q - P_{-k_i}$.
  5 : Return: $(Q)$.
```

As mentioned in [13] and Algorithm 4, it is proposed to employ a sliding window technique on the NAF-representation of the integer k. The window is strategically positioned to guarantee that the value in the window always odd. This arrangement reduces the amount of precomputation needed for the algorithm. In consideration of the need for enhanced efficiency and performance in scalar multiplication algorithms, [15] introduces an optimized method for scalar multiplication that utilizes the sliding window technique in conjunction with the 1′s complement approach. The following Algorithm 6 outlines the procedure for the corresponding method.

Algorithm 6 : Optimized sliding window method for point multiplication [15].

```
Require: A positive integer $k$, $P = (x, y)$ in $E(\mathbb{F}_p)$, width of the window $w$.
Ensure: $Q = kP$.
   $1 : k = (k_{(\ell-1)}, \cdots, k_1, k_0)_2$.
   $2 : Q \leftarrow \infty, i = \ell - 1$.
   3 : While $i > 0$ do
   3.1 : If $k_i = 0$, then: $Q \leftarrow 2Q, i \leftarrow i - 1$.
   3.1.1 : Else: $S = max(i - w_1, 0)$.
   3.1.1.1 : While $k_S = 0$ do
   3.1.1.2 : $S = S + 1$.
   3.1.2 : For $h = 1$ to $i - S + 1$ do
   3.1.2.1 : $Q \leftarrow 2Q$.
   3.1.3 : $u = (k_i, \ldots, k_S)_2$ for $k_i = k_S = 1$ and $i - S + 1 \leq w$.
   3.1.4 : $Q \leftarrow Q + [u]\,P$.
   3.1.5 : $i = S - 1$.
   3.1.6 : Return: $(Q)$.
```

3 The Scalar Decomposition Method

As mentioned in Section 1, other than encoding the scalar k into some representations to compute scalar multiplication kP, one can directly use the fast endomorphism to accelerate the computation. In this case, the scalar is decomposed into a few sub-scalars with the help of fast endomorphism. One of the proposed methods was the GLV method. The structure of the GLV method is shown in the following figure.

$$\begin{tikzcd}
 & kP \arrow[dl] \arrow[dr] & \\
k_1P & & k_2\Phi\left(P\right) = k_2\lambda P
\end{tikzcd}$$

Figure 1: GLV method.

As shown in Figure 1, k is decomposed into k_1 and k_2 with the help of endomorphism Φ, where $\Phi(P) = \lambda P$. Consider that k has n-bit, then its decomposed scalars will have $\sqrt{n}$-bit. If the decomposed scalars have bit length more than $\sqrt{n}$, the GLV method is inefficient as the computational time using this approach will be greater than computing kP using an encoding approach.

To overcome this problem, the ISD method was proposed. The mechanism is shown in the following figure:

$$\begin{tikzcd}
 & & & kP \arrow[dll] \arrow[drr] & & & \\
 & k_1P \arrow[dl] \arrow[dr] & & & & k_2\Phi\left(P\right) = k_2\lambda P \arrow[dl] \arrow[dr] & \\
k_{1,1}P & & k_{1,2}\Phi_1\left(P\right) = k_{1,2}\lambda_1 P & & k_{2,1}P & & k_{2,2}\Phi_2\left(P\right) = k_{2,2}\lambda_2 P
\end{tikzcd}$$

Figure 2: ISD method.

As shown in Figure 2, the ISD method involves two layers of decomposition. In this method, the scalar k is decomposed into $k_{1,1}, k_{1,2}, k_{2,1}$ and $k_{2,2}$ with the help of three endomorphisms, which are denoted by Φ, Φ_1 and Φ_2. These endomorphisms correspond to λP, $\lambda_1 P$ and $\lambda_2 P$, respectively. However, the ISD method did not use the fast endomorphisms. They used trivial endomorphisms in the form of $\Phi_i(P) = \lambda_i P$, where λ_i was chosen randomly from $[1, n-1]$. This causes their method to consume high computational cost as the scalar multiplication $\lambda_i P$ is computed using an encoding approach. Other than that, the method will be more inefficient if one chooses bigger λ_i. Algorithms 7 and 8 below outline the procedure for the corresponding method. The scalar multiplication using ISD method with naive method (Algorithm 1) is presented by Algorithm 9.

Algorithm 7 : Find the vector V with decomposed scalar components.

```
Require: An order of subgroup $n$, $\lambda$, a scalar $k$, choice.
Ensure: $V = (k_1, k_2)$.
  1 : Use Euclidean Algorithm [13] to find the biggest remainder which is less than $\sqrt{n}$.
  1.1 : $r_{m+1} \leftarrow$ biggest remainder.
  1.2 : $r_m \leftarrow$ remainder before $r_{m+1}$.
  1.3 : $r_{m+2} \leftarrow$ remainder after $r_{m+1}$.
  2 : Use Extended Euclidean Algorithm [13] to find the corresponding $t_m, t_{m+1}, t_{m+2}$.
  2.1 : $v_1 \leftarrow (r_{m+1}, -t_{m+1})$.
  2.2 : $v_2 \leftarrow (r_m, -t_m)$.
  2.3 : $v_3 \leftarrow (r_{m+2}, -t_{m+2})$.
  3 : If $k < 0$, then: $k \equiv k\lambda \pmod{n}$.
  4 : If choice $= 1$, then: $V_1 \leftarrow v_1$ and $V_2 \leftarrow v_3$.
  4.1 : $c_1 \leftarrow \left\lfloor -\frac{t_{m+2}}{n} k \right\rceil, c_2 \leftarrow \left\lfloor \frac{t_{m+1}}{n} k \right\rceil$.
  4.1.1 : Compute $c_1 \cdot V_1$ and $c_2 \cdot V_2$.
  4.2 : Else: $V_1 \leftarrow v_1$ and $V_2 \leftarrow v_2$.
  4.2.1 : $c_1 \leftarrow \left\lfloor -\frac{t_m}{n} k \right\rceil, c_2 \leftarrow \left\lfloor \frac{t_{m+1}}{n} k \right\rceil$.
  5 : $V \leftarrow (k, 0) - (c_1 V_1 + c_2 V_2)$.
  6 : Return: $(V)$.
```

Algorithm 8 : Find the decomposed scalar $k_{1,1}, k_{1,2}, k_{2,1}, k_{2,2}$.

```
Require: An order of subgroup $n$, a set of $\lambda = \{\lambda_{1,1}, \lambda_{1,2}, \lambda_{2,1}, \lambda_{2,2}\}$, a set of $k = \{k_1, k_2\}$, $\lambda$choice.
Ensure: $k_{1,1}, k_{1,2}, k_{2,1}$ and $k_{2,2}$.
  1 : Use Algorithm 7 to compute $V_1$ and $V_2$.
  1.1 : If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,1}$, then: compute vector $V_1$ using $k_1$ and $\lambda_{1,1}$.
  1.1.1 : Compute vector $V_2$ using $k_2$ and $\lambda_{2,1}$.
  1.2 : Else If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,2}$, then: compute vector $V_1$ using $k_1$ and $\lambda_{1,1}$.
  1.2.1 : Compute vector $V_2$ using $k_2$ and $\lambda_{2,2}$.
  1.3 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,1}$, then: compute vector $V_1$ using $k_1$ and $\lambda_{1,2}$.
  1.3.1 : Compute vector $V_2$ using $k_2$ and $\lambda_{2,1}$.
  1.4 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,2}$, then: compute vector $V_1$ using $k_1$ and $\lambda_{1,2}$.
  1.4.1 : Compute vector $V_2$ using $k_2$ and $\lambda_{2,2}$.
  2 : $(k_{1,1}, k_{1,2}) \leftarrow V_1$ and $(k_{2,1}, k_{2,2}) \leftarrow V_2$.
  3 : Return: $(V_1, V_2)$.
```

Algorithm 9 : Scalar Multiplication with naive ISD method.

```
Require: $E(\mathbb{F}_p)$, $P = (x, y)$ in $E(\mathbb{F}_p)$, $n$, a set of choosen $\lambda = \{\lambda_{1,1}, \lambda_{1,2}, \lambda_{2,1}, \lambda_{2,2}\}$, a set of
  $k = \{k_1, k_2\}$.
Ensure: $Q = kP$.
  1 : Use Algorithm 8 to find $k_{1,1}, k_{1,2}, k_{2,1}, k_{2,2}$.
  2 : Use Algorithm 1 to compute $\Phi(P)$ and scalar multiplication and additions.
  2.1 : If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,1}$, then: $\Phi_1(P) \leftarrow \lambda_{1,1}P$ and $\Phi_2(P) \leftarrow \lambda_{2,1}P$.
  2.2 : Else If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,2}$, then: $\Phi_1(P) \leftarrow \lambda_{1,1}P$ and $\Phi_2(P) \leftarrow \lambda_{2,2}P$.
  2.3 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,1}$, then: $\Phi_1(P) \leftarrow \lambda_{1,2}P$ and $\Phi_2(P) \leftarrow \lambda_{2,1}P$.
  2.4 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,2}$, then: $\Phi_1(P) \leftarrow \lambda_{1,2}P$ and $\Phi_2(P) \leftarrow \lambda_{2,2}P$.
  2.5 : Addition1 $\leftarrow k_{1,1}P + k_{1,2}\Phi_1(P)$.
  2.6 : Addition2 $\leftarrow$ Addition1 $+ k_{2,1}P$.
  2.7 : Addition3 $\leftarrow$ Addition2 $+ k_{2,2}\Phi_2(P)$.
  3 : Return: (Addition3).
```

Note that we can replace the Step 2 in Algorithm 9 with any algorithm discussed in Section 2. The efficiency of each approach, when combined with the ISD method is evaluated in Section 6 for both smaller and larger fields.

4 Family of Elliptic Curves with j-invariant 0

E_0 is a curve defined over a field $K = \mathbb{Q}(\sqrt{-3})$ with discriminant of quadratic field $D = -3$ [9] that has the form of

$$E_0 : y^2 = x^3 + B.$$

The ring of maximal order that exists in this quadratic field consists of algebraic integers which can be written as a linear combination of its basis, $\{1, \delta\}$ where $\delta = \frac{1+\sqrt{-3}}{2}$, the root for $X^2+X+1 = 0$. According to [12, 5], the ring of maximal order defines the ring of first fast endomorphism on E_0, Φ, where $\Phi(P) = (\lambda x, y)$ and λ satisfies $\lambda^2 + \lambda + 1 \equiv 0 \pmod{p}$.

Since the ISD method requires three endomorphisms, where the first endomorphism is similar with the endomorphism used by [12], the other two fast endomorphisms were constructed based on the concept of isogeny so that this method can be employed on E_0 [5]. The following theorem defines the mapping of the other two fast endomorphisms on E_0 as given by [5].

Theorem 4.1. *Let $E_0(\mathbb{F}_p) : y^2 = x^3 + B$ with $p \equiv 1 \pmod 3$. There exist points Q and P in $E_0(\mathbb{F}_p)$ with order 3 and prime order n, respectively. Define the polynomial for the second and third endomorphisms as $\Phi_1^2 + 3 = 0$ and $\Phi_2^2 - 3\Phi_2 + 3 = 0$, respectively. Then, the scalar multiplication acted on point P based on the second and third endomorphisms' mappings are defined by*

$$\Phi_i(x, y) = \left(\frac{x^3 + 4B}{{\epsilon_i}^2 x^2}, y \left[\frac{x^3 - 8B}{\epsilon_i^3 x^3} \right] \right),$$

for $i = 1, 2$, where $\Phi_1 \equiv \epsilon_{i+2} \pmod p$ and $\Phi_2 \equiv \epsilon_{i+4} \pmod p$.

All curves that are defined over the quadratic field which is $K = \mathbb{Q}(\sqrt{-3})$ belong to this family where they shared the same algebraic properties. One of the curves which belongs to this family that is commonly used as stated by Standards for Efficient Cryptography 2 (SEC 2) is a Koblitz curve with 192-bit prime field, which is also known as Secp192k1 which is defined by

$$E : y^2 = x^3 + 3,$$

with the certain parameters [6].

5 Implementation of ISD Method on Secp192k1

This paper implemented the fast endomorphisms as described in Section 4 on Secp192k1 as presented in Algorithm 10. It closely resembles to Algorithm 9. The sets $\{\epsilon_1, \epsilon_2\}$, $\{\epsilon_3, \epsilon_4\}$ and $\{\epsilon_5, \epsilon_6\}$ are corresponding to the solution modulo p of the first endomorphism, second endomorphism and third endomorphism respectively.

Algorithm 10 : Scalar multiplication using ISD method with fast endomorphisms on Secp192k1.

```
Require:
  $E(\mathbb{F}_p) : y^2 = x^3 + 3, p = 0xfffffffffffffffffffffffffffffffffffffffeffffee37$,
  $n = 0xfffffffffffffffffffffffe26f2fc170f69466a74defd8d$,
  $P = (0xdb4ff10ec057e9ae26b07d0280b7f4341da5d1b1eae06c7d, 0x9b2f2f6d9c5628a7844163d015be86344082aa88d95e2f9d)$ in Secp192k1, a set of $\lambda = \{\lambda_{1,1}, \lambda_{1,2}, \lambda_{2,1}, \lambda_{2,2}\}$,
  a set of $k = \{k_1, k_2\}$, a set of $\epsilon = \{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6\}$, $\lambda$choice, $\Phi_1(P) = \left(\frac{x^3+12}{{\epsilon_{3,4}}^2 x^2}, y\left[\frac{x^3-24}{\epsilon_{3,4}^3 x^3}\right]\right)$,
  $\Phi_2(P) = \left(\frac{x^3+12}{{\epsilon_{5,6}}^2 x^2}, y\left[\frac{x^3-24}{\epsilon_{5,6}^3 x^3}\right]\right)$.
Ensure: $Q = kP$.
  1 : Use Algorithm 8 to find $k_{1,1}, k_{1,2}, k_{2,1}, k_{2,2}$.
  2 : Use Theorem 4.1 to compute $\Phi(P)$.
  2.1 : If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,1}$, then: compute $\Phi_1(P)$ using $\epsilon_3$ and $\Phi_2(P)$ using $\epsilon_6$.
  2.2 : Else If $\lambda$choice $= \lambda_{1,1}$ and $\lambda_{2,2}$, then: compute $\Phi_1(P)$ using $\epsilon_3$ and $\Phi_2(P)$ using $\epsilon_5$.
  2.3 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,1}$, then: compute $\Phi_1(P)$ using $\epsilon_4$ and $\Phi_2(P)$ using $\epsilon_6$.
  2.4 : Else If $\lambda$choice $= \lambda_{1,2}$ and $\lambda_{2,2}$, then: compute $\Phi_1(P)$ using $\epsilon_4$ and $\Phi_2(P)$ using $\epsilon_5$.
  3 : Use Algorithm 1 to compute scalar multiplication and additions.
  3.1 : Addition1 $\leftarrow k_{1,1}P + k_{1,2}\Phi_1(P)$.
  3.2 : Addition2 $\leftarrow$ Addition1 $+ k_{2,1}P$.
  3.3 : Addition3 $\leftarrow$ Addition2 $+ k_{2,2}\Phi_2(P)$.
  4 : Return: (Addition3).
```

Similarly, in Algorithm 10, Step 3 can be replaced with the most efficient algorithm discussed in Section 2. The comparison on efficiency between the ISD method with fast endomorphism and with other approaches is conducted in Section 6 for both smaller and larger fields.

6 Analysis of the Results

We computed the computational time for the scalar multiplication defined in each fast endormorphism to ensure that our fast endomorphism can accelerate scalar multiplication. We also compared the computational time efficiency among various approaches as stated in Section 2 for the scalar multiplications defined by the fast endormophisms. Then, the computational time taken to perform scalar multiplication by the ISD method on Secp192k1 with various approaches are computed and compared to evaluate the efficiency of fast endomorphisms in the ISD method. All computations were done by using Mathematica software version 12.1.1 on an AMD Ryzen 5730U processor and 8 gigabytes of LPDDR4x RAM. The running time taken for each scalar multiplication defined by all three endomorphisms as stated in Section 4 are shown in the Table 1.

Most notably, point multiplication using window-w NAF-representation with $w = 3$ and $w = 4$ are able to reduce the computational cost on Secp192k1 for certain scalar multiplications as compared to other approaches. The symbol $\%$ in Table 1 represents the percentage of speed-up achieved when comparing the shortest execution time obtained from other approaches to the fast endomorphism method for scalar multiplication mP.

From Table 1, it is clear that the existence of fast endomorphisms accelerated the computation for certain scalar multiplications on Secp192k1, by more than 99%. This is due to the number of operations count using the encoding approach escalated for a bigger prime field. However, the number of operations count using fast endomorphism remains unchanged even for a bigger prime field. Since fast endomorphisms require less time complexity than other approaches, this will result in faster computation if fast endomorphisms are applied on any scalar decomposition method as compared to computation without fast endomorphisms on Secp192k1.

Assume that we want to compute $365164723589214823746213573856238746P$. If the ISD method had been used, we have

$$\begin{aligned}
k_{1,1} &= -2575799524310901919045959 7838,\\
k_{1,2} &= 2938909518661970414075852 7421,\\
k_{2,1} &= 6277101735386680763835059093624676943848401272189252500094,\\
k_{2,2} &= -1377154129669410992904817 21943,
\end{aligned}$$

given that

$$\begin{aligned}
\Phi_1(P) &= 3260202861401843664988870687845839771682143121980490210346P,\\
\Phi_2(P) &= 4768652298394262214412330055453552021819633518590667861213P.
\end{aligned}$$

Table 1: Comparative computation times (in seconds) for Secp192k1 using various approaches.

mP	Approaches									
	Algorithm 1	**Algorithm 3**	**Algorithm 5**			**Algorithm 6**			**Fast endomorphism**	**Gain** %
			$w=3$	$w=4$	$w=5$	$w=3$	$w=4$	$w=5$		
150844943699 241854942345 936760771225 013749039661 0177650865P	$0.019785s$	$0.021135s$	0.015474*	$0.016013s$	$0.019794s$	$0.022819s$	$0.022594s$	$0.025387s$	$0.000061s$	99.61
476865229839 426221441233 005545355202 181963351859 0667861211P	$0.019789s$	$0.021748s$	$0.019548s$*	$0.019861s$	$0.022276s$	$0.0296574s$	$0.030188s$	$0.031010s$	$0.000045s$	99.77
301689887398 483709884691 873521542450 027498079322 0355301731P	0.017543	$0.018010s$	$0.015660s$*	$0.015918s$	$0.020137s$	$0.028032s$	$0.024042s$	$0.027251s$	$0.000125s$	99.20
326020286140 184366498887 068784583977 168214312198 0490210346P	$0.020399s$	$0.020529s$	$0.016951s$	$0.016499s$*	$0.018997s$	$0.025854s$	$0.027055s$	$0.026427s$	$0.000117s$	99.29
150844943699 241854942345 936760771225 013749039661 0177650867P	$0.017189s$	$0.019433s$	$0.015421s$*	$0.016281s$	$0.019647s$	$0.023321s$	$0.023673s$	$0.026284s$	$0.000139s$	99.10
476865229839 426221441233 005545355202 181963351859 0667861213P	$0.020748s$	$0.020075s$	$0.021626s$	$0.015923s$*	$0.019475s$	$0.026466s$	$0.026501s$	$0.026432s$	$0.000111s$	99.30

Note: * Shortest time taken.

Subsequently, a comparative analysis of $3651647235892148237462135738562387 46P$ using the ISD method with different approaches is presented in Table 2.

Table 2: Time taken for point multiplication of mP in large fields using ISD method with various approaches.

ISD method with approaches	**Width size**	**Execution time(s)**
Algorithm 1	-	0.094151
Algorithm 3	-	0.093909
Algorithm 5	$w = 3$	0.093010
	$w = 4$	0.093051
	$w = 5$	0.098173
Algorithm 6	$w = 3$	0.109745
	$w = 4$	0.108181
	$w = 5$	0.109780
Fast endomorphism	-	0.079369

According to Table 2, the implementation of the fast endomorphism with the ISD method also effectively decreases the computational time. It is worth noting that the second shortest time recorded is 0.093010 seconds, achieved by Algorithm 5 with a width of 3. This leads to an approximate 15% acceleration in computational time.

7 Discussions

The original ISD method consumes more computational time due to its inefficient endomorphisms especially when dealing with large prime fields. One way to reduce the computational time in the ISD method is by using fast endomorphisms. The existence of all three fast endomorphisms that are defined over $K = \mathbb{Q}(\sqrt{-3})$ accelerates the computation via the ISD method on E_0 on larger prime fields. The comparison for running time between the ISD method with encoding approaches and with fast endomorphism for certain scalar multiplication as shown in Table 2 support our results. Even though the overall running time via the ISD method remains high due to its multilayer decompositions, the existence of fast endomorphisms is able to reduce this cost and hence makes the ISD method with fast endomorphisms more efficient than the ISD method without any fast endomorphisms.

8 Conclusions

The ISD method was proposed to enhance the computational speed and reduce the computational cost of elliptic scalar multiplication when the GLV condition is not satisfied. However, since this method employs endomorphisms which are inefficiently computable, the running time remains to be high especially on larger prime field. The implementation of fast endomorphisms on 192-bits Koblitz curve shows that the presence of fast endomorphisms reduce the computational time taken to compute the scalar multiplications defined by these endomorphism. By employing fast endomorphisms on one of the recommended curves in E_0 which is Secp192k1, the ISD method with fast endomorphisms exhibit the most significant acceleration in scalar multiplication among the various approaches.

Acknowledgement This research was funded by Universiti Sains Malaysia under the short-term grant scheme, project number 304/PMATHS/6315553.

Conflicts of Interest The authors declare no conflict of interest.

References

[1] R. K. K. Ajeena (2021). The soft graphic integer sub-decomposition method for elliptic scalar multiplication. *Journal of Discrete Mathematical Sciences and Cryptography*, *24*(6), 1751–1765. https://doi.org/10.1080/09720529.2021.1885808.

[2] 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. http://dx.doi.org/10.12732/ijpam.v87i1.5.

[3] R. K. K. Ajeena & H. Kamarulhaili (2014). Point multiplication using integer sub-decomposition for elliptic curve cryptography. *Applied Mathematics & Information Sciences*, *8*(2), 517. http://dx.doi.org/10.12785/amis/080209.

[4] N. F. H. Al Saffar & M. R. M. Said (2015). Speeding up the elliptic curve scalar multiplication using the window-w non adjacent form. *Malaysian Journal of Mathematical Sciences*, *9*(1), 91–110.

[5] S. N. F. M. A. Antony & H. Kamarulhaili (2020). Improvement of scalar multiplication on elliptic curve with j-invariant 0. *International Journal of Cryptology Research*, *10*(1), 22–36.

[6] S. Blake-Wilson & M. Qu (1999). *Standards for efficient cryptography 2: Recommended elliptic curve domain parameters*. Certicom Research, Mississauga, Canada.

[7] J. W. Bos, C. Costello, H. Hisil & K. Lauter (2013). High-performance scalar multiplication using 8-dimensional GLV/GLS decomposition. In *Cryptographic Hardware and Embedded Systems – CHES 2013*, pp. 331–348. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40349-1_19.

[8] M. Ciet, T. Lange, F. Sica & J.-J. Quisquater (2003). Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In *Advances in Cryptology – EUROCRYPT 2003*, pp. 388–400. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39200-9_24.

[9] H. Cohen (2013). *A course in computational algebraic number theory*. Springer Science & Business Media, Berlin, Heidelberg, New York. https://doi.org/10.1007/978-3-662-02945-9.

[10] C. Costello & P. Longa (2015). Four $\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the mersenne prime. In *Advances in Cryptology - ASIACRYPT 2015*, pp. 214–235. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-48797-6_10.

[11] S. D. Galbraith, X. Lin & M. Scott (2011). Endomorphisms for faster elliptic curve cryptography on a large class of curves. *Journal of Cryptology*, *24*(3), 446–469. https://doi.org/10.1007/s00145-010-9065-y.

[12] R. P. Gallant, R. J. Lambert & S. A. Vanstone (2001). Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology - CRYPTO 2001*, pp. 190–200. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44647-8_11.

[13] D. Hankerson, S. Vanstone & A. Menezes (2004). *Guide to elliptic curve cryptography*. Springer Professional Computing, New York. https://doi.org/10.1007/b97644.

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

[15] R. K. Kodali & H. S. Budwal (2013). High performance scalar multiplication for ECC. In *2013 International Conference on Computer Communication and Informatics*, pp. 1–4. IEEE. https://doi.org/10.1109/ICCCI.2013.6466286.

[16] J. Kwon, S. C. Seo & S. Hong (2018). Efficient implementations of four-dimensional GLV-GLS scalar multiplication on 8-bit, 16-bit, and 32-bit microcontrollers. *Applied Sciences*, *8*(6), 900. https://doi.org/10.3390/app8060900.

[17] P. Longa & F. Sica (2014). Four-dimensional Gallant–Lambert–Vanstone scalar multiplication. *Journal of Cryptology*, *27*(2), 248–283. https://doi.org/10.1007/s00145-012-9144-3.

[18] V. S. Miller (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology – CRYPTO'85 Proceedings*, pp. 417–426. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39799-X_31.

[19] R. J. S. Mina & J. Bacani (2023). Elliptic curves of type $y^2 = x^3 - 3pqx$ having ranks zero and one. *Malaysian Journal of Mathematical Sciences*, *17*(1), 67–76. https://doi.org/10.47836/mjms.17.1.06.

[20] Y.-H. Park, S. Jeong, C. H. Kim & J. Lim (2002). An alternate decomposition of an integer for faster point multiplication on certain elliptic curves. In *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002 Paris, France, February 12–14, 2002 Proceedings 5*, pp. 323–334. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45664-3_23.

[21] J. H. Silverman (2009). *The arithmetic of elliptic curves* volume 106. Springer, Dordrecht, Heidelberg, London, New York. https://doi.org/10.1007/978-0-387-09494-6.

[22] J. L. Theyab & R. K. K. Ajeena (2022). The 3-dimension integer sub-decomposition method for Edwards curve cryptography. In *AIP Conference Proceedings*, volume 2398 pp. 1–8. AIP Publishing. https://doi.org/10.1063/5.0094171.

[23] Z. Zhou, Z. Hu, M. Xu & W. Song (2010). Efficient 3-dimensional GLV method for faster point multiplication on some GLS elliptic curves. *Information Processing Letters*, *110*(22), 1003–1006. https://doi.org/10.1016/j.ipl.2010.08.014.