Modified Versions of Menezes-Vanstone Elliptic Curve Cryptography

Mahmoud H. Hasan[1] and Ibrahim A. Almerhag[2]

[1] College of Information Technology, Alasmarya Islamic University, Zliten – Libya
[2] Faculty of Information Technology, University of Tripoli, Tripoli – Libya
tea_mhs@asmarya.edu.ly

Abstract. Encryption came about as a result of the need for information security. Encryption serves the purpose of secure data storage. It needs to be transmitted, encrypted, then decrypted. Encryption is the process of transforming plain text into cipher text; decryption is the process of obtaining the original message from the cipher text. The key used for both encryption and decryption plays a major role in determining the strength of a cryptosystem. It is the key's size that complicates brute-force attacks. But this also means that the algorithm becomes more sophisticated and demands more computing power. For encryption systems based on elliptic curves, we can employ smaller key sizes to ensure a comparable level of security and improved performance.

This paper compares two modified versions of the Menezes-Vanstone elliptic curve cryptography with the original MVECC algorithm, applying them to different file sizes. Then the time complexity and execution time of the suggested algorithms against MVECC was calculated. The outcomes demonstrated that the suggested algorithms outperformed the original MVECC.

Keywords: Cryptography, Elliptic Curves Cryptography, Discrete Logarithm Problem, Menezes-Vanstone.

1 Introduction

Sensitive data can be securely stored and transmitted over unsecured networks, like the Internet, with the use of encryption, guaranteeing that only the intended recipient can access and read it. For millennia, encryption has been integral to the protection of transactions. The principal aim was to obscure communications so as to hinder intruders from gaining access to or changing sensitive data. Cryptography is the process of using encryption to change plaintext into ciphertext, which is unintelligible. Only an entity possessing the secret key may decrypt the ciphertext and return it to plaintext, making a cryptosystem secure. Since its inception, cryptographic algorithms has been divided into two kinds: symmetric and asymmetric cryptosystems. These terms specify whether the encryption and decryption processes within the cryptosystem utilize a single key or two keys [1].

It is said that the Discrete Logarithm Problem (DLP), which is the hard direction of a one-way function, is also challenging. It has to do with groups' of mathematical structure, which can be defined as a binary operation on a nonempty set G in its most basic form. Let g be an element of group G if g is a generator for G. This means that repeatedly exponentiating g (g*g*g) will provide all of the elements of G.

The group Z*p, a finite and multiplicative group of integers modulo a prime number p, is frequently used in cryptography. These components, provided the following conditions are met, define the DLP. Find the unique integer j in the interval [1, p -1] such that h = g^j mod p, given a generator g of the multiplicative group Z*p, a prime integer p, and another element $h \in Z_p^*$ [1]. While it is feasible to compute the inverse exponentiation operation on g^j mod p, it is impractical to calculate g^j mod p for large numbers. For this reason, the DLP is regarded as a hard issue and is a one-way function.

The Elliptic Curve Discrete Logarithm Problem (ECDLP), which is predicated on the difficulties of computing discrete logs on elliptic curves over finite fields, provides the security of elliptic curve cryptography. EC-ElGamal, Elliptic Curve Digital Signature Algorithm (ECDSA), and Diffie-Hellman cryptosystems are based on the computational complexity of this specific mathematical problem. Analogously to the previously stated discrete logarithm problem, ECDLP is an analogue of DLP. Nonetheless, in the ECDLP, the group of points on an elliptic curve over a finite field assumes the function of the subgroup Z*p. Elliptic curve cryptosystems offer a higher strength-per-key bit than their non-analogue discrete logarithm counterparts because ECDLP has been found to be far tougher than the DLP.

Elliptic Curve Cryptography (ECC) can provide the same level of security as Rivest-Shamir-Adleman (RSA) with much smaller key sizes. For example, a 256-bit ECC key offers the same security as a 3072-bit RSA key [1]. This makes ECC more efficient and better suited for devices with limited resources, such as smartphones, Internet of Things (IoT) devices, and smart cards. The smaller key sizes in ECC lead to lower computational and storage requirements, which translates to lower power consumption. This is especially beneficial for battery-powered devices. Additionally, ECC allows the use of various elliptic curves, each with its own trade-offs in terms of performance, security, and implementation complexity. This flexibility enables the selection of the most suitable curve for a given application.

2 ELLIPTIC CURVE CRYPTOSYSTEMS

Over a long period, researchers have conducted substantial research on elliptic curves, producing a wealth of literature on the subject. Neal Koblitz and V. S. Miller both submitted suggestions in 1985 urging their use in public-key cryptosystems [2]. They did not develop a brand-new method of cryptography with elliptic curves over finite fields. Rather, they used elliptic curves to incorporate public-key algorithms that already existed, such Diffie-Hellman [3]. The capacity of elliptic curves to create groups by building "elements" and "rules of combining" makes them fascinating. These clusters possess enough common features to build cryptographic algorithms.

The ECDLP is defined as follows: given an elliptic curve E defined over a finite field Fp, a point $P \in E(F_p)$ of order n, and a point $Q \in P$, find the integer $L \in [0, p-1]$ such that $Q = LP$. The integer L is called the discrete logarithm of Q to the base P, denoted by $L = log_p Q$ [4].

Moreover, elliptic curves over a finite field F_p When p is a prime number, the prime field, also known as $F_{(p)}$, is a finite field made up of integers in the interval [0, p - 1] [5]. This field uses modulo p for arithmetic operations, ensuring that all calculation outputs remain within the designated finite space. The formula (1) gives the value of an elliptic curve $E(F_p)$ over a finite field F_p as follows:

$$Y^2 = x^3 + a\,x + b \bmod p \qquad (1)$$

Where $a, b \in Fp$ and the condition $4\,a^3 + 27\,b^2 \neq 0 \bmod p$, together with an imaginary point at infinity O, create a group on the curve [6]. When points within an elliptic curve group are added, another point on the same curve is produced. Similarly, all scalar multiples of the points in the group are also on that elliptic curve.

There are three guidelines for adding points to an elliptic curve group [4]:

- O + O = O
- (x, y) + O = (x, y)
- (x, y) + (x, -y) = O

The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation modulo p (where x and y are numbers in F_p).

Given ECDLP is an elliptic curve E, we consider a primitive element P and another element T. The DLP is finding the integer d, where $1 \leq d \leq \#E$, such that:

P + P + …… + P = dP = T d times, where #E denotes the number of points on the curve.

In cryptosystems, d is the private key which is an integer, while the public key T is a point on the curve with coordinates $T = (x_T, y_T)$. In contrast, in the case of the discrete logarithm problem in Z_p^*, both keys were integers [4].

2.1 Massey-Omura Elliptic Curve Cryptosystem

Massey-Omura cryptosystem is a well-known public-key cryptosystem that relies on discrete logarithms over the finite field Fp. James Massey and Jim K. Omura proposed it in 1982 as a potential enhancement to Shamir's initial three-pass cryptographic technique, which was created “circa 1980” [3]. Massey–Omura cryptographic system is a secure three-pass protocol that enables Alice to transmit a message to Bob without the requirement of exchanging or distributing encryption key [3].

Now let's describe the Massey-Omura protocol by which Alice may send a secret message M to Bob. Let E be an elliptic curve defined over the finite field Fp, N = #E, and there exists a publicly known relationship between the plaintext and some points on the curve, so for any message M the point $P_M \in E$ is known by all the parties, also

known as Plaintext Message (P_M). The embedding system $M \rightarrow P_M$, as well as p, E, and N, are publicly known.

When Alice wants to communicate secretly with Bob, they proceed like this:

Alice chooses a pair of secret numbers: encryption factor (eA) and a decryption factor (dA), such that eA * dA = 1 mod N. Similarly, Bob chooses a pair of secret numbers: (eB, dB) such that eB * dB = 1 mod N.

Now, whenever Alice wanted to send a secret message-point $P_M \in E$ to Bob, she has to proceed as follows:

- Alice sends $e_A\ P_M\ mod\ p$ to Bob,
- Bob sends $e_B\ e_A\ P_M mod\ p$ to Alice,
- Alice sends $d_A\ e_B\ e_A\ P_M\ mod\ p = \ e_B\ P_M$ to Bob.
- Bob calculates $d_B\ e_B\ P_M$value as P_M, allowing him to get the original message point.

2.2 ElGamal Elliptic Curve Cryptosystem

The ElGamal encryption scheme was proposed by Taher ElGamal in 1985 [4]. Similar to other public-key cryptosystems, the renowned ElGamal cryptosystem also has a direct elliptic curve counterpart, which can be defined as follows:

Alice and Bob openly select an elliptic curve E over F_p and a randomly choose a base point $G \in E$. Assume that they are also aware of the quantity of points on E, denoted as $N = \#E$.

Assume that Bob desires to transmit a confidential communication, labeled as P_M, to Alice:

- Alice chooses a random integer a, computes: aG mod p and sends it to Bob.
- Bob chooses at random an integer b and computes: bG mod p.
- Bob also computes: (P_M + baG) mod p, then sends the secret encrypted message (bG, P_M + baG) mod p to Alice.
- Alice knows the secret key a. So, she can calculate (abG mod p) to recover is the original plaintext message P_M using equation (2):

$$P_M = (P_M + a(bG)) - b(aG)) \bmod p \quad (2)$$

The eavesdropper can only get P_M if she can solve the ECDLP. In other words, if she has access to the random integer 'a' from aG mod p or 'b' from bG mod p, she can obtain P_M. But it is widely known, there is no an efficient way to find an elliptic curve's discrete logarithms.

3 Menezes Vanstone Elliptic Curve Cryptography and variants

The plaintext message units (P_M) are situated on the elliptic curve E, and presently there is not a good technique for deterministically constructing these points on E, which is a major problem with the Massey-Omura and ElGamal elliptic curve cryptosystems [3].

The method for encrypting a message into a point is the Menezes-Vanstone Elliptic Curve Cryptosystem (MVECC). Where the elliptic curve is used for "masking" in the MVECC version, and ordered pairs of (nonzero) components. They can be any kind of text; and not limited to being points on E. A more effective variation was developed by Menezes and Vanstone, where the elliptic curve is used for "masking" and the pairs of plaintext and ciphertext are allowed to be in F*p× F*p rather than being limited to the elliptic curve. Below is a description of this variety [7].

When Alice wants to send a message to Bob using his public key. Alice and Bob have to decide upon the following conventions, all of which are public:

- p: a large prime number (it must at least be larger than 3).
- *Fp:* a field of size p (p is prime, so it works like modular arithmetic).
- E: an elliptic curve over *Fp* of the form: $y^2 = x^3 + ax + b$; (a, b in *Fp*).
- G: a randomly selected point on E (called the base point) that will generate subgroup.

Each user chooses a random integer d_X which will be his/her own secret key, then computes and publishes the point a_XG.

Suppose Alice wishes to send the message M = (m1, m2) ∈ F*p× F*p to Bob.

Private key: Bob's private key is $d_b \in \#E$, which is randomly selected and only he knows it.

Public Key: Bob's pubic key is calculated as $B = d_bG$. Ideally, it is distributed to the world.

Encryption:

- Alice has secret M, which she splits up into m_1 and m_2,
- calculates $(k_1, k_2) = d_a B$.
- calculates $C_0 = d_a G$. Note that C_0 is a point.
- calculates $C_1 = k_1m_1 \bmod p$.
- calculates $C_2 = k_2m_2 \bmod p$.
- Alice sends encrypted message $C = (C_0, C_1, C_2)$ to Bob.

Decryption:

Bob wants to get back the message M from C.

Bob calculates $d_bC_0 = (k_1, k_2)$ retrieves message M by calculating

- $m_1 = C_1k_1^{-1} \bmod p$.
- $m_2 = C_2k_2^{-1} \bmod p$.
- $M = (m_1, m_2)$.

The MVECC is a significant cryptographic system that offers several advantages. Unlike other cryptosystems, it does not rely on addition operations within the elliptic curve group. Additionally, the message being encrypted does not need to be a point on the elliptic curve [7].

The inverse of the two integers at the critical point must be calculated in order to comply with the MVECC. The plaintext, which is made up of pairs of two numbers, is the same length as the key and the ciphertext. The introduction of more MVECC types resulted from the use of this feature to improve the efficiency of the system. As Hameed et al. stated, the aforementioned variant permits the system to execute the inverse operation one time only [8].

Dawahdeh et al. [9] propose a technique to enhance the MVECC. This improvement significantly decreases the computational time required by the encryption and decryption procedures in comparison to the previous approach. Furthermore, the revised technique exclusively employs addition and subtraction operations that use the hexadecimal ASCII value to encode every alphabet character in the message prior to encryption. This enhances the security and complexity of the algorithm, making it more resilient against adversaries.

Ghadi et al. [10] proposed a new technique, which is based on the Quadratic Bezier Curve (QBC), to enhance the security and efficiency of the Menezes-Vanstone Elliptic Curve Cryptosystem. Also, they utilized the ASCII value to convert the text into numbers. This was achieved by taking every two characters in the message, separating them as a point, and then converting them into ASCII. The proposed method succeeds in achieving a higher level of security than the original MVECC method used in all of the NIST tests. This modification enhanced the MVECC, resulting in a higher level of security. Additionally, the mathematical complexity of the proposed method surpasses that of the original method.

The authors, Heru et al. [11] perform text, image, audio, and video data security using a modified version of the Menezes-Vanstone elliptic curve cryptography algorithm. The study implements MVECC in a program for text, image, audio, and video file encryption and decryption, as well as providing performance data for both the program and alternative approaches for comparison. The result shows that the encrypted file size using the proposed method is smaller than in previous studies.

A novel discrete chaotic map (2D-TFCDM) was introduced in [12], that incorporates a discrete fractional difference. In addition, the authors use the Menezes-Vanstone elliptic curve cryptosystem to generate secret keys. Moreover, the bifurcation diagrams, the plot of the largest Lyapunov exponent, and the phase portraits illustrate the chaotic behaviors of the proposed map. The researchers used the discrete fractional map to encrypt color images. The findings unequivocally demonstrated the superiority of the proposed algorithm over its counterparts.

In the elliptic curve encryption and decryption schemes, the plaintext becomes twice as long as the ciphertext. This characteristic can be exploited to enhance the system's efficiency. Menezes and Vanstone introduce the ElGamal analog variant, this implementation of ECC eliminates the need to embed the plaintext unit in the curve, allowing it to be represented as any two integer numbers in the field and either accepted or rejected in the curve equation. This is the major difference from the other ECC, especially the ElGamal cryptosystem, since there is a need to embed the plaintext unit in the curve by representing the plaintext unit as an acceptable point in the curve. However, the key

point is: the Menezes-Vanstone algorithm must compute the inverses of each number. But the inverse operation is an expensive operation.

4 PROPOSED METHODS OF MVECC

Looking back at the encryption function in the Menezes-Vanstone algorithm, it computes two numbers (i.e., the ciphertext unit) from four numbers; two of these are the plaintext, and the other two numbers are the key points. Furthermore, it needs four multiplication operations, two for each process (encryption and decryption) plus two inverse operations for decryption.

In this paper, we aim to avoid the inverse operation, similar to the MVECC algorithm's attempt to circumvent the inverse operation in the decryption process. The following subsections will introduce the modifications made to MVECC algorithm that utilize addition and subtraction operations only for both encryption and decryption processes. This approach differs from the MVECC algorithm, which employs multiplication and inverse during encryption/decryption processes, resulting in longer processing times than subtraction and addition operations.

Each of the proposed cryptographic methods that modify the MVECC algorithm will be presented in a separate subsection, showing the main functions of the method including: key generation, encryption process, decryption process, a proof of correctness, and an example illustrating how it works. The two modified methods of encryption and decryption will use text files as input data.

4.1 Algorithm I

Suppose Alice wants to send a message $M \in F^*p \times F^*p$ to Bob.

Key generation:
Bob's private key is: $db \in \#E$, only he knows it, and it is a randomly selected.
Bob's Public key: is calculated as $B = db\ G$, ideally it is sent to Alice.

Encryption:
Alice has message $M = (m_1, m_2) \in F^*p \times F^*p$, then she calculates:
$(k_1, k_2) = da\ B$.
$C_0 = da\ G$. Note that C0 is a point.
$C_1 = k_2 m_2 - m_1 \bmod p$.
$C_2 = k_1(m_1 - k_2\ m_2) + m_1 \bmod p$.
Alice sends encrypted message $C = (C_0, C_1, C_2)$ to Bob.

Decryption:
Bob wants to get back the message M from C.

Bob finds db $C_0 = (k_1, k_2)$, and retrieves message M by calculating:
$m_1 = C_2 + k_1 C_1 \bmod p$.
$m_2 = (C_1 + m_1)\, k_2^{-1} \bmod p$.
Thus, Bob recovers the message $M = (m_1, m_2)$.

Proof:
$m_1 = C_2 + k_1 C_1 \bmod p$.
$= k_1(m_1 - k_2 m_2) + m_1 + k_1 (k_2 m_2 - m_1) \bmod p$.
$= k_1 m_1 - k_1 k_2 m_2 + m_1 + k_1 k_2 m_2 - k_1 m_1 \bmod p = m_1$.
$m_2 = (C_1 + m_1)\, k_2^{-1} \bmod p$.
$= (k_2 m_2 - m_1 + m_1)\, k_2^{-1} \bmod p = m_2$.

Example:
Let E be an elliptic curve define over Fp where p = 3023 having a = 1, b = 2547 where $(4a^3 + 27b^2) \bmod p = 2027$, and #E = 3083. Therefore, let G = (2237, 2480) be a point on E.

Bob chooses a secret random integer db = 2313. Which is Bob's private key. And his public key is:
B = db G = 2313 (2237, 2480) = (934, 29).

Encryption:
Alice represent the message M = (1111, 2222) = (m1, m2).
Chooses a secret random integer da = 1236
Alice computes:
(k_1, k_2) = da B = 1236 (934, 29) = (2537, 1632),
C_0 = da G = 1236 (2237, 2480) = (1713, 1709)

$C_1 = k_2 m_2 - m_1 \bmod p$
= (2222 * 1632) – 1111 mod 3023 = 616.

$C_2 = k_1(m_1 - m_2 k_2) + m_1 \bmod p$
= 2537 (1111 + 2222 * 1632) + 1111 mod 3023 = 1210.
Alice sends to Bob the following encrypted message
C = ((1713, 1709), 616, 1210).

Decryption:
Bob computes:
db C_0 = 2313 (1713, 1709) = (2537, 1632) = (k_1, k_2),
$m_1 = C_2 + k_1 C_1 \bmod p$.
= 1210 + (2537 * 616) mod 3023 = 1111,
$m_2 = (C_1 + m_1)\, k_2^{-1} \bmod p$.
= (616 + 1111) 1869 mod 3023 = 2222.
Thus, Bob recovers the message M = (1111, 2222).

4.2 Algorithm II

Suppose Alice wants to send a message M ∈ F*p× F*p to Bob.

Key generation:
Bob's private key is: db ∈ #E, only he knows it, and it is a randomly selected.
Bob's Public key: is calculated as B = db G, ideally it is sent to Alice.

Encryption:
Alice has message M = (m1, m2) ∈ F*p× F*p, then she calculates:
(k_1, k_2) = da B.
C_0 = da G. Note that C_0 is a point.
$C_1 = k_1 m_1 – m_2 \bmod p$.
$C_2 = k_2(m_2 – k_1 m_1) + m_1 \bmod p$.
and sends encrypted message $C = (C_0, C_1, C_2)$ to Bob.

Decryption:
Bob wants to get back the message M from C.
Bob calculates db $C_0 = (k_1, k_2)$, retrieves message M by calculating:
$m_1 = C_2 + k_2 C_1 \bmod p$.
$m_2 = k_1 m_1 – C_1 \bmod p$.
Thus, Bob recovers the message $M = (m_1, m_2)$.

Proof:
$m_1 = C_2 + k_2 C_1 \bmod p$.
$= k_2(m_2 – k_1 m_1) + m_1 + k_2(k_1 m_1 – m_2) \bmod p$.
$= k_2 m_2 – k_1 k_2 m_1 + m_1 + k_1 k_2 m_1 – k_2 m_2 \bmod p$.

$m_2 = k_1 m_1 – C_1 \bmod p$.
$= (k_1 m_1 – (k_1 m_1 – m_2)) \bmod p$.

Example:
Consider an elliptic curve E defined over the finite field Fp, where p = 3023. The curve is given by the equation $y^2 = x^3 + 2547x + 1$, and it satisfies the condition $(4a^3 + 27b^2) \bmod p = 2027$. The number of points on the curve, denoted by #E, is equal to 3083. Let G = (2237, 2480) be a point on E.
Bob chooses a secret random integer db = 2313. Which is Bob's private key. And his public key is:
B = dbG = 2313 (2237, 2480) = (934, 29).

Encryption:
Alice represent the message M = (1111, 2222) = (m_1, m_2). Chooses a secret random integer da = 1236

Alice computes:
(k_1, k_2) = da B = 1236 (934, 29) = (2537, 1632),
C_0 = da G = 1236 (2237, 2480) = (1713, 1709),

$C_1 = k_1m_1 – m_2 \bmod p.$
= 1111 * 2537 – 2222 mod 3023 = 1972

$C_2 = k_2(m_2 – k_1 m_1) + m_1 \bmod p.$
= 1632 (2222 + 1111 * 2537) + 1111 mod 3023 = 2302
Alice transmits the encrypted communication to Bob.
C = ((1713, 1709), 1972, 2302).

Decryption:
Bob computes:
db C_0 = 2313 (1713, 1709) = (2537, 1632) = (k_1, k_2),
$m_1 = C_2 + k_2C_1 \bmod p.$
= 2302 + (1632 * 1972) mod 3023 = 1111,
$m_2 = k_1 m_1 – C_1 \bmod p.$
= 1111 * 2537 – 1972 mod 3023 = 2222
Thus, Bob recovers the message M = (1111, 2222).

5 Computational Complexity

The computational complexity of the encryption and decryption functions for the MVECC and the two proposed approaches is computed in this section. The O-notation has been shown to be quite valuable in assisting analysts in categorizing algorithms based on their performance and in directing algorithm designers towards finding the most optimal algorithms for significant problems.

Adding two k-bit numbers needs k Binary calculations [8].

TIM(k-bit + k-bit) = O(s) For the input numbers of size n decimal digits

TIM(n + n) = O(log n), where the number of bits of n equal $\log_2 n$. The multiplication of two k-bits binary integers requires s^2(s*s) bit operation, because it needs s addition operations. That is:

TIM(k-bit * k-bit) = $O(s^2)$ For the input numbers of size n decimal digits

TIM(n * n) = $O(\log n)^2$, the amount of bits in n is equal to the logarithm base 2 of n.

5.1 Menezes-Vanstone Complexity

Let the size of the input message unit be n, in MVECC method the encryption function is :
$C_1 = k_1m_1$
$C_2 = k_2m_2$

then:
$\mathrm{TIM}(C_1) = \mathrm{O}(\log \mathrm{k})^2$ Binary calculations.
$\mathrm{TIM}(C_2) = \mathrm{O}(\log \mathrm{k})^2$ Binary calculations.
The decryption function is:
$m_1 = C_1 \times k_1^{-1} \bmod p$.
$m_2 = C_2 \times k_2^{-1} \bmod p$.

then:
$\mathrm{TIM}(m_1) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{TIM}(k_1^{-1})$.
$\mathrm{TIM}(k_1^{-1}) = \mathrm{O}(\log \mathrm{k})^3$, by utilizing Euclid's approach to a greater degree.
$\mathrm{TIM}(m_1) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})^3$ Binary calculations.
$\mathrm{TIM}(m_2) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})^3$ Binary calculations.

5.2 Algorithm I Complexity

Let the size of the input message unit be n, in algorithm I the encryption function is:
$C_1 = k_2 m_2 - m_1 \bmod p$.
$C_2 = k_1(m_1 - k_2 m_2) + m_1 \bmod p$.

Then:
$\mathrm{TIM}(C_1) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})$ Binary calculations.
$\mathrm{TIM}(C_2) = \mathrm{O}(2(\log \mathrm{k})^2) + \mathrm{O}(2 \log \mathrm{k})$ Binary calculations.

The decryption function is:
$m_1 = C_2 + k_1 C_1 \bmod p$.
$m_2 = (C_1 + m_1) k_2^{-1} \bmod p$.

Then:
$\mathrm{TIM}(m_1) = \mathrm{O}(\log \mathrm{k}) + \mathrm{O}(\log \mathrm{k})^2$ Binary calculations.
$\mathrm{TIM}(m_2) = \mathrm{O}(\log \mathrm{k}) + \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})^3$ Binary calculations.

5.3 Algorithm II Complexity

In Algorithm II the encryption function is defined for an input message unit of size k.
$C_1 = k_1 m_1 - m_2 \bmod p$.
$C_2 = k_2(m_2 - k_1 m_1) + m_1 \bmod p$.

Then:
$\mathrm{TIM}(C_1) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})$ Binary calculations.
$\mathrm{TIM}(C_2) = \mathrm{O}(2(\log \mathrm{k})^2) + \mathrm{O}(2 \log \mathrm{k})$ Binary calculations.
The decryption function is:
$m_1 = C_2 + k_2 C_1 \bmod p$.
$m_2 = k_1 m_1 - C_1 \bmod p$.

Then:
$\mathrm{TIM}(m_1) = \mathrm{O}(\log \mathrm{k}) + \mathrm{O}(\log \mathrm{k})^2$ Binary calculations.
$\mathrm{TIM}(m_1) = \mathrm{O}(\log \mathrm{k})^2 + \mathrm{O}(\log \mathrm{k})$ Binary calculations.

6 RESULTS AND DISCUSSION

The next section will graphically depict the execution duration for each algorithm to assess its efficiency in encrypting and decrypting data. Processor speed and algorithm complexity influence the time each algorithm takes to complete its operation, as shown in Table 1 below. It is important to note that a shorter execution time indicates a more efficient algorithm.
It is worth mentioning that the size of the decrypted file is equal to the size of the encrypted file. And in implementing the proposed algorithms Java programming language was used, and the tests performed on a laptop running Windows 7 Home Premium 64-bit with an Intel CoreTM i3 2nd generation 2.4 GHz and 4 GB of RAM.

Table 1. Execution times (milliseconds) for different text file sizes.

Algorithm	Process	2KB	4KB	6KB	8KB	16KB	32KB
MVECC	Encrypt	0.66	1.13	1.67	1.98	3.51	6.75
	Decrypt	2.43	4.40	6.24	8.45	16.95	33.20
Algorithm I	Encrypt	0.90	1.30	1.80	2.18	4.14	7.75
	Decrypt	1.63	2.87	4.06	5.37	10.04	20.12
Algorithm II	Encrypt	0.90	1.30	1.80	2.18	4.14	7.75
	Decrypt	0.86	1.24	1.64	2.00	3.66	6.96

6.1 Execution Time of Encryption

Figure 1 displays the execution time needed for encryption, in milliseconds, by each of the proposed methods and the MVECC algorithm. Also, it displays the outcomes obtained by measuring the duration of the encryption process for text files of different sizes. The execution times of the proposed algorithms are about comparable to that of the MVECC method.

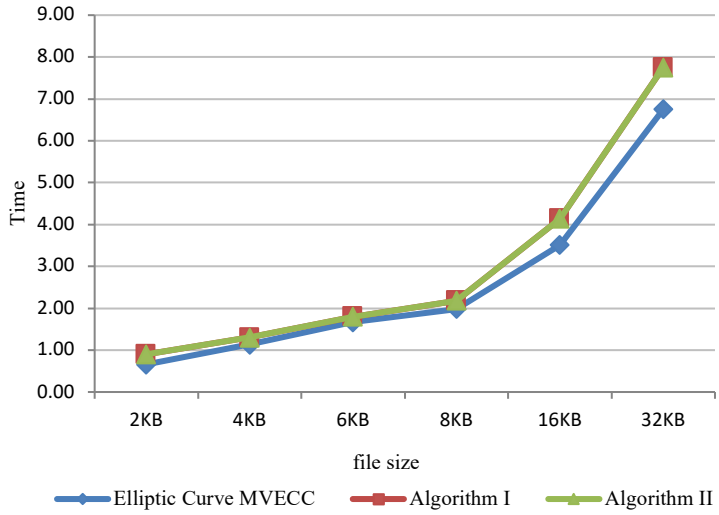

Fig. 1. The encryption execution time (in milliseconds).

6.2 Execution Time of Decryption

Figure 2 displays the execution time in milliseconds for the decryption process of the proposed methods and the MVECC algorithm. And it shows the outcomes obtained from measuring the time it takes to decode text files of different sizes. The results indicate that algorithm I has a shorter execution time than the MVECC algorithm, whereas algorithm II has the shortest execution time among the other two algorithms. Therefore, algorithm II compared to the other algorithms has the highest performance.

Fig. 2. The decryption execution time (in milliseconds).

6.3 Execution Time of Encryption and Decryption

Using the MVECC method and the two suggested techniques, the total execution times (encryption and decryption) in milliseconds was collected. The gathered results represent the total time required to encrypt and decrypt text files of different sizes are shown in Figure 3. In comparison to both algorithm I, II approach and the MVECC, the results show that algorithm II has the least execution time.

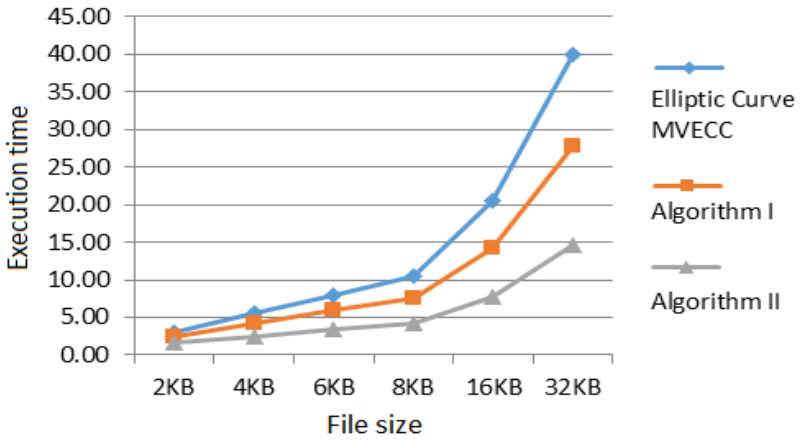

Fig. 3. Total time (in milliseconds) for encryption and decryption.

7 CONCLUSION

In this study, two modifications to the MVECC were suggested. Each demonstrating enhanced performance in terms of rapid data encryption and decryption. The goal was to reduce the amount of time needed by the encryption and decryption processes by optimizing the operations of algorithms. Algorithm I needs to calculate the inverse operation only once for each decryption process, whereas algorithm II does not need to perform any inverse operations. However, the MVECC scheme needs to compute two inverses for each decryption process.

With regard to the encryption and decryption execution times, algorithm II works better than the other two methods. In addition, algorithm II takes fewer Binary calculations than the other two algorithms in terms of time complexity.

There are several promising directions for future research. We will apply modified algorithms to other types of data, such as grayscale images, color images, and videos, to show how well these modified algorithms can encrypt and decrypt these types of files.

8 References

1. W. Stallings, 'Cryptography and Network Security: Principles and Practices.', 5th Edition, 2011.
2. U. Priyatharsan, P. L. Rupasinghe, and I. Murray, 'A new elliptic curve cryptographic system over the finite fields', in Proceedings of the 6th National Conference on Technology and Management: Excel in Research and Build the Nation, NCTM 2017, doi: 10.1109/NCTM.2017.7872847.
3. S. Y. Yan, 'Computational Number Theory and Modern Cryptography.', 2012. doi: 10.1002/9781118188606.
4. B. Preneel, 'Understanding cryptography.', Springer, 2014.
5. Mahmoud H. S. Hasan, 'Image Encryption Based On Multiplicative Ciphers.', Journal of Humanitarian and Applied Sciences, vol. 7, no. 14, pp. 288–297, 2022.
6. Mahmoud H. S.Hasan, 'Image Encryption Based on Elliptic Curve Diffie–Hellman Key Exchange.', Journal of Basic Sciences, vol. 35, no. 2, pp. 37–53, 2022.
7. N. F. Hameed Al-Saffar and M. Rushdan Md Said, 'On the Mathematical Complexity and the Time Implementation of Proposed Variants of Elliptic Curves Cryptosystems.', 2013.
8. S. Y. Yan, 'Number Theory for Computing.', Berlin Springer Berlin, 2010.
9. Z. E. Dawahdeh, S. N. Yaakob, and R. R. Bin Othman, 'A new modification for Menezes-Vanstone elliptic curve cryptosystem.', J Theor Appl Inf Technol, vol. 85, no. 3, 2016.
10. D. M. Ghadi and A. AL-Rammahi, 'Improvement of menezes-vanstone elliptic curve cryptosystem based on quadratic bezier curve technique.', Journal of Computer Science, vol. 16, no. 5, 2020, doi: 10.3844/JCSSP.2020.715.722.
11. E. K. Heru, Faisal, and H. Pranoto, 'File Encryption Application using Menezes-Vanstone Elliptic Curve Cryptography Based on Python.', in Procedia Computer Science, 2023. doi: 10.1016/j.procs.2023.10.569.
12. Z. Liu, T. Xia, and J. Wang, 'Image encryption technique based on new two dimensional fractional-order discrete chaotic map and Menezes-Vanstone elliptic curve cryptosystem.', Chinese Physics B, vol. 27, no. 3, 2018, doi: 10.1088/1674-1056/27/3/030502.