cryptography

MDPI

Article

Securing Data Exchange with Elliptic Curve Cryptography: A Novel Hash-Based Method for Message Mapping and Integrity Assurance

Younes Lahraoui [1], Saiida Lazaar [1], Youssef Amal [1] and Abderrahmane Nitaj [2,*]

[1] Mathematics, Computer Science and Applications TEAM, Abdelmalek Essaâdi University, ENSA, Tangier 90000, Morocco; younes.lahraoui@etu.uae.ac.ma (Y.L.); slazaar@uae.ac.ma (S.L.); yamal@uae.ac.ma (Y.A.)

[2] Department of Mathematics, Normandie University, UNICAEN, CNRS, LMNO, 14000 Caen, France

* Correspondence: abderrahmane.nitaj@unicaen.fr

Abstract: To ensure the security of sensitive data, elliptic curve cryptography (ECC) is adopted as an asymmetric method that balances security and efficiency. Nevertheless, embedding messages into elliptic curve (EC) points poses a significant challenge. The intricacies of this process can greatly affect the overall security and efficiency of the cryptosystem, reflecting security vulnerabilities observed in many existing schemes that utilize ElGamal ECC-based encryption. In this paper, we introduce an innovative hash-based technique for securely embedding messages into EC points before encryption. A random parameter and a shared secret point generated through the EC Diffie–Hellman protocol are used to bolster the scheme's security. The security of the proposed method is evaluated against various attack models; moreover, the complexity, and sensitivity of the encryption scheme, as well as its inputs, are analyzed. The randomness assessment of the ciphertext was performed using the NIST statistical test suite. Additionally, we propose a mechanism to ensure the integrity of the message by securely appending a tag to the ciphertext. As a consequence, a comprehensive analysis of our scheme demonstrates its effectiveness in maintaining data security and integrity against various attack models. The algorithm also meets more criteria such as the strict avalanche criterion, linear complexity, and operability.

Keywords: elliptic curve cryptography; message mapping; message integrity; statistical tests; hash function; sensitivity analysis; NIST

check for updates

Citation: Lahraoui, Y.; Lazaar, S.; Amal, Y.; Nitaj, A. Securing Data Exchange with Elliptic Curve Cryptography: A Novel Hash-Based Method for Message Mapping and Integrity Assurance. *Cryptography* **2024**, *8*, 23. https://doi.org/10.3390/cryptography8020023

Academic Editors: Carlo Blundo and Josef Pieprzyk

Received: 30 March 2024
Revised: 23 May 2024
Accepted: 30 May 2024
Published: 2 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

During the first half of 2023, SonicWall Capture Labs reported a 37% year-to-date increase in IoT malware attacks, reaching 77.9 million, as per the SonicWall 2023 report [1]. IBM's 2023 report highlighted an average data breach cost of USD 3.78 million across sectors, with critical infrastructure incurring an additional USD 1.26 million [2]. These alarming figures emphasize the urgent need to secure sensitive data. Asymmetric cryptographic approaches, particularly elliptic curve cryptography (ECC), play a pivotal role in balancing heightened security with manageable complexity. They also help address the computational demands associated with their impact on battery-powered IoT devices [3].

Within the realm of cryptography, ECC is highly versatile, excelling in key exchange through the elliptic curve Diffie–Hellman protocol (ECDHP), which enables entities to securely establish a shared secret. ECC is widely used in digital signatures via the elliptic curve digital signature algorithm (ECDSA) protocol, ensuring the integrity of transmitted data. For encryption, numerous recent research papers have introduced innovative ECC encryption schemes, often building on foundational methods such as ElGamal ECC-based encryption [4] and the elliptic curve integrated encryption scheme (ECIES) [5]. The latter operates as a hybrid system, utilizing ECC to generate a shared key through ECDHP and,

subsequently, encrypt the message using a symmetric encryption scheme. This makes it efficient in terms of computational complexity while facing certain drawbacks highlighted in [6,7].

In this paper, we use ElGamal ECC-based encryption as the foundation for introducing an enhanced version. However, a significant challenge arises during the process of mapping messages into elliptic curve (EC) points. If not carefully addressed, this challenge can compromise the overall security and the efficiency of the cryptosystem. Existing schemes utilizing ElGamal ECC-based encryption have notably revealed vulnerabilities in this context, as evidenced by a flaw observed when encrypting the same block several times within a single encryption process, as discussed in [8]. Such a flaw could potentially lead to a successful chosen-plaintext attack (CPA) and secret key leakage. This underscores the pressing need for innovative solutions to fortify the resilience of the system against such vulnerabilities.

In this paper, we introduce a novel hash-based technique designed to securely embed messages into EC points before encryption, addressing the aforementioned challenge. To augment the security infrastructure, the incorporation of a random parameter and a shared secret point generated through the ECDHP is proposed, in a manner that prevents unauthorized entities from conducting the message mapping process, distinguishing it from all existing message mapping algorithms. The choice of ECDH and ElGamal encryption, coupled with our proposed mapping method, is motivated by their well-established security features, computational efficiency, and adherence to established cryptographic standards. This combination ensures robust data protection within our proposed cryptosystem, as recommended in [9] and highlighted in [10]. Our approach undergoes rigorous evaluation against various attack models, with a comprehensive analysis covering the complexity and sensitivity of the encryption scheme and its inputs. The randomness assessment of the ciphertext is conducted using the NIST statistical test suite [11], ensuring a robust measure of unpredictability. Additionally, we put forth a mechanism aimed at ensuring the integrity of transmitted messages. This involves combining the message mapping scheme and a hash function to generate a tag, which is then securely appended to the ciphertext, enhancing the overall reliability of the communication process.

The findings of our study demonstrate the effectiveness of the proposed scheme in preserving data security and integrity across various attack models. Notably, it provides security against chosen-ciphertext attacks (CCAs), chosen-plaintext attacks (CPAs), malleability, and replay attacks (RAs), as emphasized in Section 5.2. Moreover, the algorithm successfully meets the strict avalanche criterion, showcasing a remarkable change tolerance of 50% even with minor input variations. We have meticulously verified the linear complexity and the operational robustness of the algorithm, reinforcing its standing as a good solution in the realm of ECC-based encryption.

The rest of this paper is organized as follows. In Section 2, we review existing literature, introduce relevant materials, and discuss related works. In Section 3, we propose the new method. In Section 4, we study the performance of the new method. In Section 5, we analyze its security and sensitivity assessment. We provide a step-by-step example of the implementation in Section 6. We conclude the paper in Section 7.

2. Background and Literature Review

In this section, we delve into definitions and key protocols of elliptic curve cryptography, specifically exploring the elliptic curve cryptography over prime fields, elliptic curve Diffie–Hellman protocol, and ElGamal ECC-based encryption. Additionally, we investigate essential guidelines for secure message mapping. Finally, we highlight related works that contribute to shaping our understanding of advancements in ECC encryption, with a specific focus on schemes utilizing ElGamal encryption. Throughout the remainder of this paper, we denote the sender as (s) and the receiver as (r), which represent two communicating parties.

2.1. Elliptic Curve over Prime Fields

Let $p > 3$ be a prime number, and $\mathbb{F}_p$ be the prime field of p elements. An elliptic curve E over $\mathbb{F}_p$ is the set of points (x, y) that satisfy the Weierstrass equation [12]:

$$y^2 \equiv x^3 + ax + b \pmod{p}, \tag{1}$$

where $a, b \in \mathbb{F}_p$ with $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. Let $E(\mathbb{F}_p)$ denote the following set:

$$E(\mathbb{F}_p) = (x, y) \mid y^2 \equiv x^3 + ax + b \pmod{p} \cup \mathcal{O}, \tag{2}$$

where $\mathcal{O}$ represents the point at infinity. A group structure can be defined over the set $E(\mathbb{F}_p)$ together with the addition law described below.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points of $E(\mathbb{F}_p)$. The addition is defined by the following:

$$P_1 + P_2 = \begin{cases} \mathcal{O} & \text{if } \mathrm{P}_1 = -\mathrm{P}_2 \\ P_3 & \text{if } \mathrm{P}_1 \neq -\mathrm{P}_2 \end{cases} \tag{3}$$

Algebraically, the coordinates of $-P_2$ are $(x_2, -y_2)$, and $P_3 = (x_3, y_3)$ is defined by the following:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 & \pmod{p} \\ y_3 \equiv (x_1 - x_3)\lambda - y_1 & \pmod{p}, \end{cases} \tag{4}$$

with

$$\lambda \equiv \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & \pmod{p} \text{ if } \mathrm{P}_1 \neq \mathrm{P}_2 \\ (3x_1^2 + a)(2y_1)^{-1} & \pmod{p} \text{ if } \mathrm{P}_1 = \mathrm{P}_2 \end{cases} \tag{5}$$

A scalar multiplication of an integer k and a point $P \in E(\mathbb{F}_p)$ yields another point $Q \in E(\mathbb{F}_p)$ defined by

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

The elliptic curve discrete logarithm problem (ECDLP) involves finding the integer $k \in \mathbb{N}$, such that $Q = kP$, given points Q and P on the curve. Additionally, $\#P$ denotes the order of the point P, which is the smallest positive integer n, such that $nP = \mathcal{O}$.

2.2. Elliptic Curve Diffie–Hellman Protocol

The elliptic curve Diffie–Hellman (ECDH) protocol is a key exchange scheme that enables two communicating parties to establish a shared secret point over an open and insecure channel. Its security is based on the difficulty of the elliptic curve discrete logarithm problem (ECDLP). The protocol operates as follows.

Let G be a generator point of $E(\mathbb{F}_p)$. To agree on a secret point, the sender (s) and receiver (r) randomly select their private keys in $[1, \ldots, \#G]$, denoted as u_s and u_r, and then proceed as described in Algorithm 1.

Algorithm 1 Elliptic curve Diffie–Hellman algorithm

Require: Generator point G of $E(\mathbb{F}_p)$, private key u_s of (s), and private key u_r of (r).
Ensure: The shared key K_{sh}.
1: (s) and (r) compute and exchange their public keys $P_s = u_s G$ and $P_r = u_r G$.
2: (s) computes the point $K_{sh} = u_s P_r$ in $E(\mathbb{F}_p)$.
3: (r) computes the point $K_{sh} = u_r P_s$ in $E(\mathbb{F}_p)$.
4: **return** the shared key K_{sh}.

2.3. Elliptic Curve Analogue of ElGamal Encryption

In ElGamal ECC-based encryption [4], the receiver first publishes his public key $P_r = u_r G$, where G is a generator of $E(\mathbb{F}_p)$ and $u_r \in [1, \ldots, \#G]$. To encrypt and send a message, m, to (r), the sender (s) starts by encoding it into B blocks. Subsequently, these

blocks are mapped into corresponding points $(P_{m_i})_{1 \leq i \leq B}$ in the elliptic curve $E(\mathbb{F}_p)$. The sender then proceeds with the following steps to encrypt each point P_{m_i}.

- Generate a random ephemeral key k_i.
- Compute $C_{i,1} = k_i G$ and $C_{i,2} = P_{m_i} + k_i P_r$.
- Transmit the encrypted ciphertext pairs $(C_{i,1}, C_{i,2})$ to the receiver.

In various research papers, ElGamal ECC encryption is often structured to use identical ephemeral keys, denoted as $k_1 = k_2 = \cdots = k_B$, aiming to mitigate the computational overhead associated with point multiplications, specifically $k_i G$ and $k_i P_r$ computations. Throughout the remainder of this paper, we will distinguish between two variants of ElGamal ECC encryption. The first variant, denoted as ElGamal-v1, involves using pairwise distinct values for $(k_i), 1 \leq i \leq B$. The second variant, labeled as ElGamal-v2, employs equal values for $(k_i), 1 \leq i \leq B$ across all blocks. The benefits of ElGamal-v2 are obvious in its reduced need for both point multiplications and additions, resulting in lower throughput compared to ElGamal-v1, as illustrated in Table 1. Nevertheless, inadequately implemented message mapping in ElGamal-v2 could expose vulnerabilities to various attacks, including CPA and CCA.

Table 1. Comparison between ElGamal-v1 and ElGamal-v2.

	ElGamal-v1	**ElGamal-v2**
Number of Point Additions	B	B
Number of Point Multiplications	$2B$	2
Throughput	High (considering B)	Lower (compared to B)

2.4. Ground Rules of a Valid Message Mapping

As previously illustrated, a mapping process can compromise the overall encryption system's security if not implemented correctly. Here are some ground rules for mapping plaintexts to elliptic curve points. A plaintext message, M, to be mapped to an elliptic curve, $E(\mathbb{F}_p)$, is first subdivided into blocks and converted to an integer in $\mathbb{F}_p$; finally, each block, m, is mapped to a point, $P_m \in E(\mathbb{F}_p)$, using a mapping application, as follows:

$$f : \mathbb{F}_p \rightarrow E(\mathbb{F}_p) \tag{6}$$

satisfying the following properties [13]

- Ensuring the operationality: The sender can map a block, m, of the message to a point $P_m = f(m) \in E(\mathbb{F}_p)$, and the receiver can reverse the message mapping process to retrieve $m = f^{-1}(P_m)$.
- Key dependency: Incorporates shared secret keys into the mapping process to make it key-dependent. This enhances security and prevents unauthorized parties from mapping plaintexts to curve points.
- Randomization: Introduce an element of randomness into the mapping process to prevent attackers from predicting the mapping for specific plaintexts, namely, systems with small plaintext spaces. This can be achieved by using a random value or salt in the mapping.
- Resistance to attacks: The mapping process should be designed to resist cryptographic attacks, such as chosen-plaintext attacks (CPA) and chosen-ciphertext attacks (CCAs), and should not mount a weakness that could be exploited by the statistical attack against ciphertext.
- Bandwidth usage: Well-designed message mapping must depress the bandwidth utilization. For example, if each character of a message is mapped separately to a point on the curve, then the encryption will produce a large amount of data when compared with another mapping scheme that embeds a block of the message.

- Complexity: Sometimes, there may be trade-offs between complexity and efficiency. More complex mapping processes may provide stronger security but require more computational resources, and, therefore, more power consumption.

2.5. Related Work

The following algorithms elucidate various techniques for embedding an integer, m, into a point, P_m, on an elliptic curve, $E(\mathbb{F}_p)$.

In [4,14], Koblitz proposes a method that starts by choosing a base parameter, k, such that $(m+1)k < p$. It defines $x = mk + j$, where $j = 0, 1, \ldots$, until finding an integer, y, so that (x, y) fulfills Equation (1). The probability of failure of this method is around $(1/2^k)$.

In [15], Srinath and Chandrasekaran propose a method where the likelihood of finding a mapping point is increased by verifying if $-x^3 - ax + b$ is a quadratic residue modulo p. To retrieve m from the point $P_m = (x, y)$, the receiver calculates $m = \lfloor x/k \rfloor$, the greatest integer less than or equal to x/k. An adapted version of Koblitz's method for binary finite fields is discussed in [16].

In [17], Boneh and Franklin propose a method to encode elements into an elliptic curve via the Weierstrass equation $y^2 \equiv x^3 + b \pmod{q}$ with $q = p^n$, and $p \equiv 2 \pmod{3}$. An element $u \in \mathbb{F}_q$ is mapped to $\left((u^2 - b)^{\frac{2q-1}{3}}, u \right)$.

In [18], Icart presents a method to encode elements into elliptic curves with short equations $y^2 \equiv x^3 + ax + b \pmod{p}$ with $p \equiv 2 \pmod{3}$. An element $u \in \mathbb{F}_p$ is mapped to (x, y) with the following:

$$x = \left(v^2 - b - \frac{1}{27}u^6 \right)^{\frac{2p-1}{3}} + \frac{1}{3}u^2,$$
$$y = ux + v,$$
$$v = \frac{3a - u^4}{6u}.$$

In [19], Vigila et al. introduce a method to encode each character into a distinct point. In this approach, communicating parties agree on a random point $P_m \in E(\mathbb{F}_p)$. To communicate using this setting, the sender multiplies the ASCII value corresponding to each character by the point P_m and encrypts the resulting points using ElGamal-v1. After decryption, the receiver solves the ECDLP to retrieve the ASCII values and reconstructs the original message by converting ASCII values back to characters. This procedure demands substantial resources in terms of power and computation.

In [20,21], the communicating parties agree on a table where each character is associated with a distinct point on the curve. Utilizing this table, the sender encrypts points that correspond to message characters. Once the receiver decrypts the cipher, they can retrieve the original message using the same table. Although [21] employs ElGamal-v1, and [20] utilizes a modified version of ElGamal-v1, both approaches exhibit high complexity and bandwidth usage.

To map a message with n characters, the approach suggested in [22] starts by selecting a generator point, G. Subsequently, each character is associated with the point αG, where α corresponds to the alphabetic order of the character. These points are then organized into a matrix Q with dimensions of $3 \times \frac{(n+q)}{3}$, filling the remaining $q = n \pmod{3}$ positions with points corresponding to white space. Additionally, the matrix Q is multiplied by a public 3×3 non-singular matrix A. The resulting matrix AQ contains the message mapping points, which will be encrypted later using ElGamal-v2. An implementation of this method is provided in [23].

The mapping algorithm proposed in [13] begins by converting each character of a message block into an 8-bit representation of its corresponding ASCII code. These representations are then concatenated to form a binary string m, which is padded with a fixed number, N, of zeros. The resulting binary string is then converted into an integer,

x. Subsequently, x is incremented by one until finding an integer y, such that (x, y) forms a mapping point of the message block. This method is analogous to Koblitz's message mapping approach with base parameters $k = 2^N$ and $x = m'k + j$, where m' represents the decimal form of m.

The upcoming schemes introduce recent ECC-based encryption techniques designed to enhance data security in IoT-based devices. These schemes incorporate some of the aforementioned message-mapping algorithms.

In [24], Almajed and Almogren introduce an authenticated encryption scheme based on elliptic curve cryptography. In this scheme, the bit string of the first message block is XORed with an initial vector (IV) to yield block B_0. For $i \in \{1, \ldots, NumberOfBlocks\}$, B_i is the result of XORing the bit string of the message block number, i, with B_{i-1}. Each block, B_i, is then padded with 3 bits and mapped to a point on an elliptic curve. An enhanced version of this work is proposed in [8], where each block, B_i, is the result of XORing the bit string of the message block number i with the x-coordinate of B_{i-1}, the corresponding mapping point. The base parameter used in Koblitz's method during message mapping is $k = 16$.

In [25], the authors propose a clustering-based approach that employs fuzzy C-means and enhanced ECC-ElGamal encryption for secure data transmission in wireless sensor networks. The authors employ Koblitz's method to encode messages onto a curve point. However, the paper addresses certain security threats while leaving others unexplored, such as malleability, CCA, and replay attacks.

In [26,27], Reegan and Kabila propose two approaches to integrate DNA sequences for encoding and decoding messages, employing Koblitz's method for mapping messages to curve points and ElGamal-v2 for encryption. While the concept of utilizing random DNA sequences for data encryption is intriguing, its current implementation is neither cost-effective nor practical for most applications. Additionally, both methods fall short of explaining the specific process by which communicating parties reach a consensus on the DNA sequences used during encoding.

In [28], Das and Namasudra propose a novel encryption scheme that utilizes ECC, AES, and Serpent to secure healthcare data in IoT-based healthcare systems. ECC is employed to encrypt a timestamp (TS) using ElGamal ECC-based encryption, but the authors do not clarify the mapping of TS to a point, which can harm the overall security of the system.

Most of the related work schemes are classified as static message mapping schemes [4,13,14]: This occurs when each message character or message block is consistently mapped to the same point under the same public parameter (base parameter k, ASCII code, alphabetic order, etc.). These methods are insecure against CPA and CCA when used with ElGamal-v2, as we will demonstrate in Section 5.2. Notably, those that map each character to a point are also vulnerable to frequency analysis attacks. However, these security flaws can be addressed if the mapping is coupled with ElGamal-v1 [19–21], although there will be trade-offs in terms of security and computational overhead. An alternative mapping approach has emerged in recent schemes that integrate ElGamal-v2 and employ methods such as initial vector (IV) [8,24] or matrix multiplication [22] for mapping. These approaches establish a one-to-many relationship between a message character or block and elliptic curve points, allowing the message to be mapped to different points for each mapping attempt. Nevertheless, an adversary can still map messages of their choice, except in the case of schemes incorporating random DNA sequences [26,27], which face challenges related to DNA sequence agreements and their associated limitations.

3. The Proposed Scheme

In this section, we present an enhanced version of the ECC-based encryption outlined in [29]. First, we consider an elliptic curve $E(\mathbb{F}_p)$ defined by the Weierstrass equation, Equation (1), over the prime field $\mathbb{F}_p$, where G represents its base point. In Table 2, we provide a comprehensive overview of various parameters that will be used throughout the remainder of this paper.

Table 2. Key notations relevant to the proposed scheme.

Symbol	Description
p	Big prime number.
$\mathbb{F}_p$	Prime field $\mathbb{Z}/\mathbb{Z}_p$.
$E(\mathbb{F}_p)$	Elliptic curve defined over $\mathbb{F}_p$.
G	Base point on the elliptic curve.
u_k	Private key of user k.
P_u	Public key of user u.
K_{sh}	Secret key shared between sender (s) and receiver (r).
M	The plaintext message.
l_o	Length of the object o (number of digits or number of characters).
$H(x)$	Hash value of x converted to an integer with $l_p - 2$ digits.
$H^{(n)}(x)$	the n-th nested composition of the hash function H applied to the input x.
B	Number of message blocks.
m_i	The message block number i as an integer.
$lsd_n(x)$	x, after discarding its n least significant digits.
$+$	Employed to represent the addition of points and the addition of numbers.
$x \oplus y$	Aligning binary values of x and y by left-padding with '0's to ensure equal size, then applying bitwise XOR.

In modular arithmetic, lines and curves serve as abstract representations used to describe mathematical relationships or sequences within the modular space, as opposed to the tangible geometric lines in Euclidean geometry. Nevertheless, the intersection of lines with the elliptic curve in modular space, starting from a shared secret point, forms the foundation of our proposed message mapping method. To delineate the entire system, this section is structured into six primary phases. These include key generation, tasked with creating the necessary keys for subsequent phases; message encoding, involving the conversion of messages into integer blocks represented as elements of $\mathbb{F}_p$; message mapping onto elliptic curve points; tag generation and encryption of the mapped points; decryption of the cipher points and tag verification; the reversal of the message mapping; and finally, integer values decoding to restore the original message. A comprehensive overview of the proposed scheme is depicted in Figure 1.

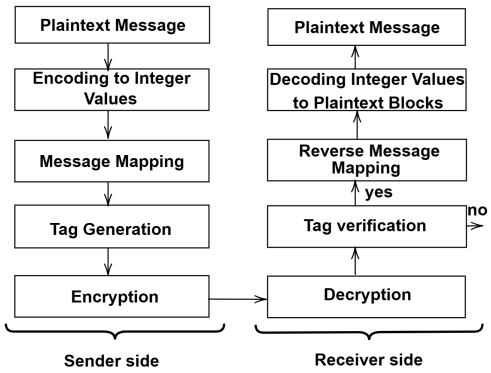


Figure 1. A comprehensive overview of the proposed scheme.

3.1. Key Generation

Key generation is a vital step in establishing secure communication, serving as a cornerstone of modern cryptography. Algorithm 2 illustrates the key generation process using ECC. In this context, the user (s) independently selects a private key and calculates his public key using that private key, while the user (r) does likewise. In the upcoming sections, we will elaborate on how users (s) and (r) use their private and public keys, as well as the parameter *Counter*, for secure communication.

Algorithm 2 Key Generation

Require: An elliptic curve $E(\mathbb{F}_p)$, and a generator G.
Ensure: Two private keys, a shared key, and a Counter.
1: Users (s) and (r) initialize $Counter = 0$
2: User (s) chooses a random private key u_s
3: User (s) calculates the public key $P_s = u_s G$
4: User (r) chooses a random private key u_r
5: User (r) calculates the public key $P_r = u_r G$
6: Publish the resulting points P_s and P_r
7: Both users (s) and (r) use their private key and ECDHP to compute a shared key $K_{sh} = (x_s, y_s)$ as described in Algorithm 1
8: Each user stores his private key, the point K_{sh} and $Counter = 0$

3.2. Message Encoding

In [29], the plaintext M is converted to the integer value $\sum_{i=1}^{l_M} \text{ASCII}(M[i]) \cdot 256^{i-1}$, where $\text{ASCII}(M[i])$ represents the ASCII value of the i-th character of plaintext M. Then, the resulting integer is subdivided into blocks of $l_p - 3$ digits, and each block is mapped to a point on the curve. This process increases the conversion time due to the exponentiation calculations, particularly for larger message sizes. To avoid this issue within this paper, we segment the plaintext into blocks of size $q = \lfloor (l_p - 2)/\log(256) \rfloor$ prior to encoding, this yields $\left\lfloor \frac{l_M}{q} \right\rfloor + \epsilon$ blocks, where ϵ is defined as follows:

$$\epsilon = \begin{cases} 0, & \text{if } q \text{ divides } l_M, \\ 1, & \text{else.} \end{cases}$$

Furthermore, for each encryption attempt, the sender (s) randomly selects an ephemeral key (k), ensuring that k is less than the order of the base point, G. The sender then computes and publicly discloses the value of kG, and subsequently computes and stores the session key $kP_r = (x_k, y_k)$ for further usage.

Afterward, the plaintext message blocks are converted into $(m'_i)_{1 \le i \le B}$ integers in $\mathbb{F}_p$, and then encoded into the integer sequences $(m_i)_{1 \le i \le B}$ using $H^{(Counter)}(x_k)$ as the outcome of the nested composition of the hash function H applied to the input x_k and each integer index. Note that $(m_i)_{1 \le i \le B}$ are pairwise-distinct, even if $i \neq j$ exists, such that $m'_i = m'_j$. Figure 2 illustrates the different steps to encode a plaintext message.

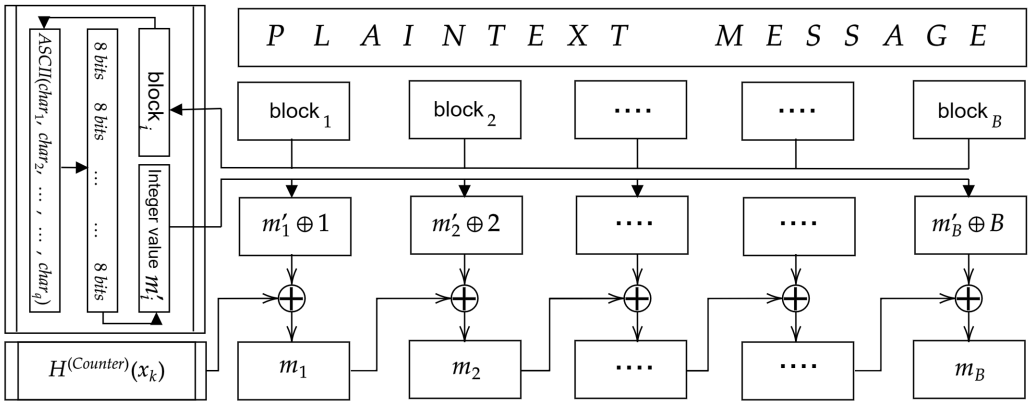


Figure 2. Plaintext message encoding.

3.3. Message Mapping

Message mapping is a necessary process during ElGamal ECC-based encryption, as ECC arithmetic operates solely on points along the curve. It associates the arbitrary

plaintext to some points called mapping points, accomplished by mapping either an individual character or an entire plaintext block to a single point. The proposed message mapping Algorithm 3 involves padding each encoded block, m_i, on both its left and right sides with '1' and '0', respectively. This modified block is subsequently mapped to a point $P_{m_i} = (x_{m_i}, y_{m_i})$, which satisfies the following system of equations:

$$\begin{cases} y^2 \equiv x^3 + ax + b \pmod{p}, \\ y \equiv S_i x + m_i \pmod{p}, \end{cases} \quad \text{and } P_{m_i} \neq \pm K_{sh} \tag{7}$$

where $S_i = (y_s - m_i)x_s^{-1} \pmod{p}$ represents the slope of the line passing through the shared secret point $K_{sh} = (x_s, y_s)$. Furthermore, during the encoding phase, the incorporation of the hash value $H^{(Counter)}(x_k)$ is integral. Here, x_k serves as the random source for the proposed message mapping scheme. However, the apparent randomness observed is only from the adversary's viewpoint. This arises due to x_k being derived as the x-coordinate resulting from the scalar multiplication of the receiver's public key, P_r, and a randomly chosen positive integer, $k < \#G$. The scheme can be executed multiple times, altering both the most significant and least significant digits of m_i. In each iteration, the problem is redefined when seeking a solution (Figure 3).

The task of solving System (7) is addressed by substituting its first equation into the second, resulting in Equation (8)

$$x^3 - S_i^2 x^2 + (a - 2Sm_i)x + b - m_i^2 = 0 \pmod{p}. \tag{8}$$

Since the x-coordinate x_s of K_{sh} is a solution to Equation (8), the problem reduces to the following quadratic equation:

$$x^2 + (x_s - S_i^2)x + (x_s^2 - S_i^2 x_s + a - 2S_i m_i) = 0 \pmod{p}.$$

For the same reason, we have the following:

$$\begin{aligned} (x_s^2 - S_i^2 x_s + a - 2S_i m_i) &= x_s^{-1}(x_s^3 - S_i^2 x_s^2 + a x_s - 2S_i x_s m_i) \pmod{p} \\ &= x_s^{-1}(x_s^3 + a x_s - (S_i x_s + m_i)^2 + m_i^2) \pmod{p} \\ &= x_s^{-1}(y_s^2 - b - y_s^2 + m_i^2) \pmod{p} \\ &= x_s^{-1}(-b + m_i^2) \pmod{p}. \end{aligned}$$

Hence, solving System (7) is equivalent to solving the following equation:

$$x^2 + (x_s - S_i^2)x + x_s^{-1}(-b + m_i^2) = 0 \pmod{p}. \tag{9}$$

in $\mathbb{F}_p \setminus \{x_s\}$. According to [30], such equations can be efficiently solved over the prime field $\mathbb{F}_p$ using the algorithm proposed by Berlekamp [31], which has a complexity of $\mathcal{O}(4\log(p))$. However, this equation can be deterministically solved by applying the same relations commonly used to solve second-degree polynomial equations in the real numbers field, while preserving the requirement of the prime field $\mathbb{F}_p$.

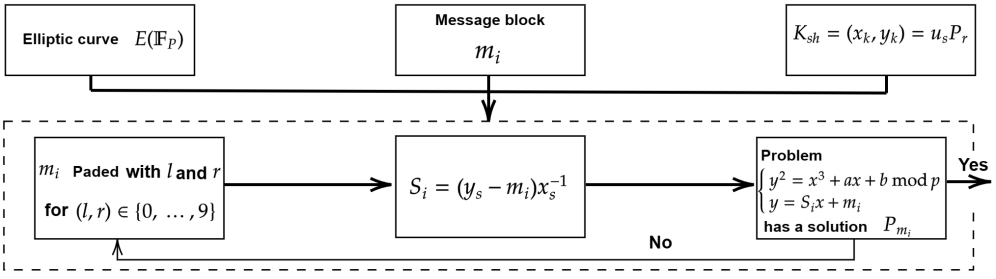


Figure 3. Message mapping process.

Algorithm 3 Message mapping algorithm

Require: An elliptic curve $E(\mathbb{F}_p)$, the shared key $K_{sh} = (x_s, y_s)$, and an integer block m_i.
Ensure: The mapping point $P_{m_i} = (x_{m_i}, y_{m_i})$

```
Pad m_i on the left with '1' and on the right with '0'
for all (l, r) ∈ {0, ..., 9}^2 and l ≠ 0 do
    m_i ← Integer m_i with left digit equals l and right digit equals r
    S_i ← (y_s − m_i) x_s^{-1} (mod p)
    if System (7) has a solution P_{m_i} = (x_{m_i}, y_{m_i}), then
        return P_{m_i} = (x_{m_i}, y_{m_i})
    end if
end for
```

3.4. Encryption

The proposed encryption Algorithm 4 is a variant of ElGamal ECC-based encryption. It plays a crucial role in guaranteeing the confidentiality and integrity of messages during their transmission. In the realm of cryptographic communication, ensuring the security of messages against interception and tampering attacks is of paramount importance. To achieve this, we introduce a tag denoted as C_{m^*}, which is appended to the ciphertext. This tag acts as a safeguard, preserving the integrity of the transmitted data. The process of generating the C_{m^*} tag involves a series of intricate steps. Starting with the computation of m^*, which involves various parameters, including the coordinates of the message's mapping points, the sender's ID_s, and *Counter*; this value is then encoded into a mapping point represented as P_{m^*}. Figure 4 depicts more details about the tag creation. Furthermore, this algorithm addresses the encryption of the mapping point associated with each message block, as shown in Figure 5. In the upcoming security analysis section, we will delve into a comprehensive examination of how these enhancements bolster security and safeguard against vulnerabilities.

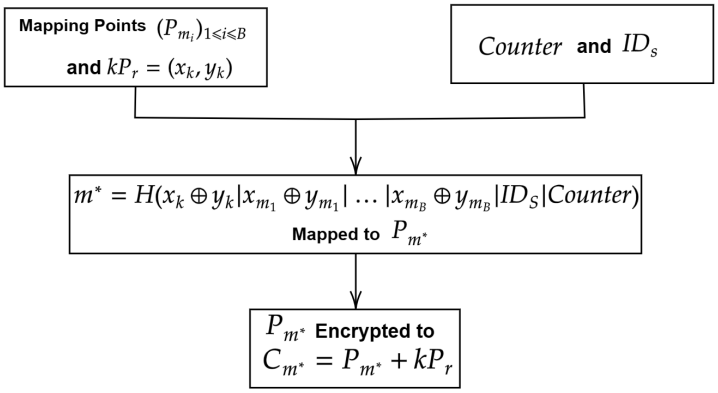


Figure 4. Tag generation procedure.

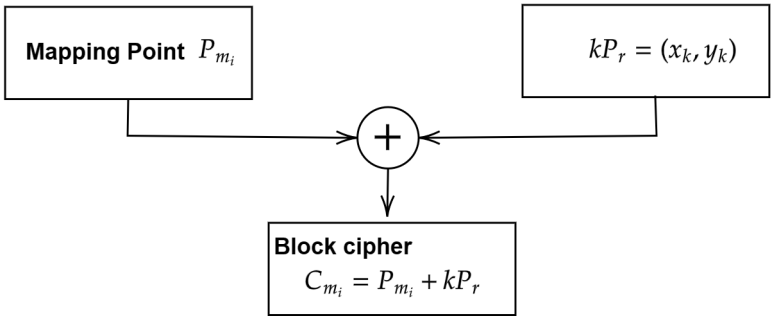


Figure 5. Mapping point encryption using ElGamal-v2.

Algorithm 4 Proposed encryption scheme

Require: $kP_r = (x_k, y_k)$, $(P_{m_i})_{1 \leq i \leq B}$, ID_s, $Counter$

Initialize m^*
$Counter \leftarrow Counter + 1$
Compute $m^* \leftarrow H(x_k \oplus y_k | x_{m_1} \oplus y_{m_1} | \ldots | x_{m_B} \oplus y_{m_B} | ID_s | Counter)$
Encode leftmost bits of m^* into a mapping point P_{m^*}
Encrypt P_{m^*} as $C_{m^*} \leftarrow P_{m^*} + kP_r$
for i in $\{1, 2, \ldots, B\}$ **do**
    Encrypt mapping point P_{m_i} as $C_{m_i} \leftarrow P_{m_i} + kP_r$
end for
return Ciphertext $C = \{kG, (C_{m_i})_{1 \leq i \leq B}, ID_s, C_{m^*}\}$

3.5. Decryption

The decryption is described in Algorithm 5. It is a pivotal component of the proposed cryptosystem, as it handles data transmitted through an untrusted communication channel. Its primary objective is to ensure the integrity of the received ciphertext. To achieve this, the algorithm performs ElGamal ECC-based decryption to recover the mapping points $(P_{m_i})_{1 \leq i \leq B}$. Subsequently, it computes $C_{\widetilde{m}^*}$ and compares the result with the received tag C_{m^*} while updating the *Counter* value to match the sender's *Counter*. If the tag verification is successful, the algorithm accepts the mapping points $(P_{m_i})_{1 \leq i \leq B}$. Otherwise, it rejects the ciphertext and returns $\perp$, as shown in Figure 6. In our security analysis section, we will explore how this process enhances the system's resilience against various attacks.

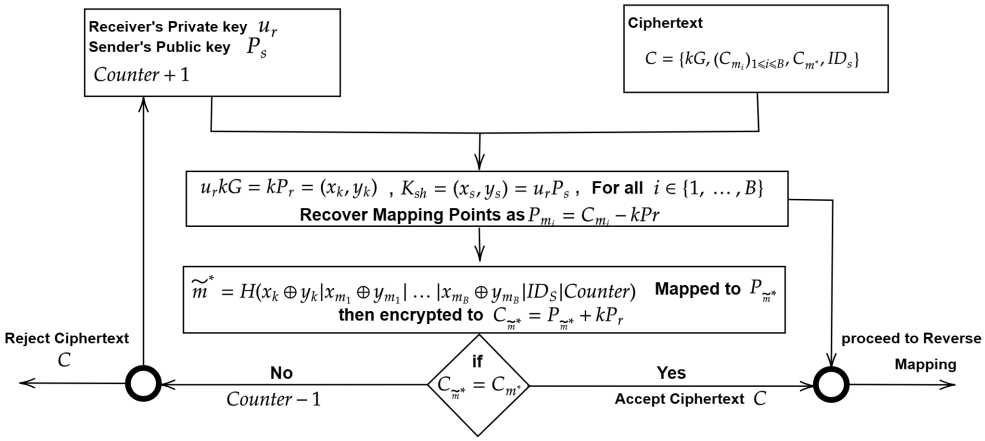


Figure 6. Decryption and integrity verification process.

3.6. Reverse Mapping and Decoding Process

The reverse mapping Algorithm 6 serves as a critical component within the proposed cryptosystem. Its primary function is to reverse the mapping process applied during encryption, thereby facilitating the recovery of the original message. Through this essential process, the algorithm ensures the integrity of the transmitted data. To achieve its goal, the algorithm operates iteratively for each $i = 1, 2, \ldots, B$, where the receiver calculates the slope, S_i, of the line passing through the points K_{sh} and P_{m_i}. Subsequently, he retrieves m_i as the y-intercept of this line, excluding the first and last digits. Figure 7 summarizes the complete procedure. The resultant number sequence $(m_i)_{1 \leq i \leq B}$ undergoes a decoding transformation. For each index $i = B, B-1, \ldots, 2$; we define the integer m'_i as the result of the bitwise exclusive OR (XOR) operation between m_i, i, and m_{i-1}. Additionally, for $i = 1$, m'_1 is defined as the bitwise XOR between m_1, 1, and $H^{(Counter)}(x_k)$, where x_k is the x-coordinate of the session key kP_r. The next step involves converting the resulting integers into text format. This is achieved by dividing the binary representation of each integer m'_i into 8-bit blocks, which are then decoded using the ASCII table to obtain characters corresponding to their respective integer values. Finally, these text components

are concatenated to reconstruct the original plaintext message M. Figure 8 illustrates how the integer sequence $(m_i)_{1\leq i\leq B}$ resulting from reverse mapping is decoded to the original plaintext message.

Algorithm 5 Decryption algorithm

Require: Ciphertext $C = \{kG, (C_{m_i})_{1\leq i\leq B}, ID_s, C_{m^*}\}$, Private Key u_r, $Counter$.
Ensure: $(P_{m_i})_{1\leq i\leq B}$ or reject.
1: Initialize $\widetilde{m}^*$
2: $Counter \leftarrow Counter + 1$
3: Compute $u_r kG = kP_r = (x_k, y_k)$
4: **for** i in $\{1, 2, \ldots, B\}$ **do**
5: Calculate $P_{m_i} = C_{m_i} + (-kP_r)$
6: **end for**
7: Compute $\widetilde{m}^* = H(x_k \oplus y_k | x_{m_1} \oplus y_{m_1} | \ldots | x_{m_B} \oplus y_{m_B} | ID_s | Counter)$
8: Encode leftmost bits of $\widetilde{m}^*$ into a mapping point $P_{\widetilde{m}^*}$
9: Encrypt $P_{\widetilde{m}^*}$ as $C_{\widetilde{m}^*} = P_{\widetilde{m}^*} + kP_r$
10: **if** $C_{\widetilde{m}^*} = C_{m^*}$ **then**
11: **return** $(P_{m_i})_{1\leq i\leq B}$, and proceed to reverse mapping process.
12: **else**
13: $Counter \leftarrow Counter - 1$
14: **return** $\perp$ and reject the ciphertext
15: **end if**

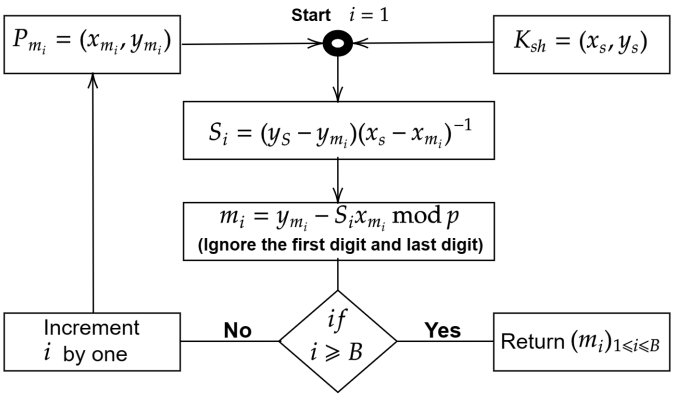


Figure 7. Point-reverse mapping.

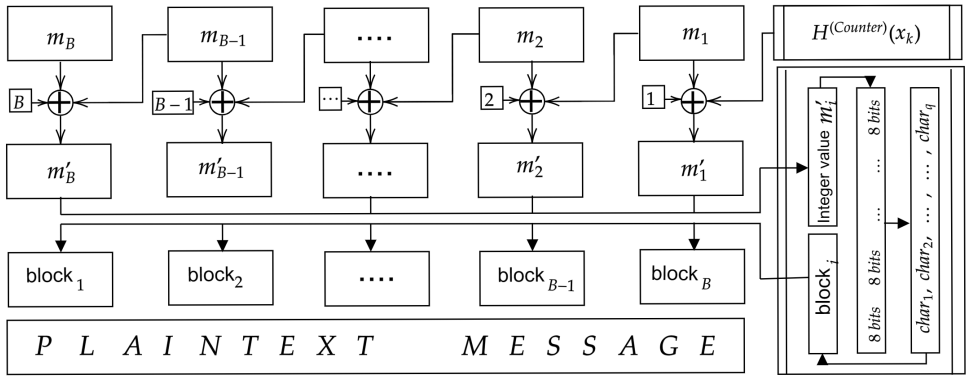


Figure 8. Plaintext message decoding.

Algorithm 6 Reverse mapping and decoding

Require: Mapping Points $(P_{m_i})_{1\leq i\leq B}$, K_{sh}, $kP_r = (x_k, y_k)$, p, $Counter$.
Ensure: Original Plaintext M.
1: Initialize M as an empty string.
2: Initialize $IntBlocks$ to an empty set of block integers.
3: **for** i from 1 to B **do**
4: Compute $S_i \equiv (y_s - y_{m_i})(x_s - x_{m_i})^{-1} \pmod{p}$.
5: Compute $m_i \equiv (y_{m_i} - S_i x_{m_i}) \pmod{p}$ (Remove first and last digits).
6: Append the resulting m_i to $IntBlocks$.
7: **end for**
8: $m_0 \leftarrow H^{(Counter)}(x_k)$.
9: **for** i from 0 to $B-1$ **do**
10: $j \leftarrow B - i$.
11: $m_j \leftarrow IntBlocks[j]$.
12: $m'_j = m_j \oplus j \oplus m_{j-1}$.
13: Convert m'_j to text.
14: Concatenate the resulting text to the left of M.
15: **end for**
16: **return** M.

4. Performance Evaluation

In this section, we present an assessment study for the proposed scheme and provide a comparison with some of the related methods. The experiments were conducted on a virtual machine hosted in an Intel Core i5-8350U processor running at 1.70 GHz, 1 GB of memory, and 2 processors.

4.1. Overlapping Problem

The proposed message mapping algorithm is a probabilistic method. Utilizing the secret key point K_{sh} and the numerical value m_i corresponding to a message block B_i, the algorithm iterates in an attempt to identify an intersection point P_{m_i} between the line passing through the point $K_{sh} = (x_s, y_s)$ with a slope of $S_i = (y_s - m_i)x_s^{-1}$. However, it is important to note that this method may fail to produce such a point. In this paragraph, we investigate the probability of success for the proposed message mapping scheme.

Embedding the integer m_i into a point $P_{m_i} \in E(\mathbb{F}_p)$ using Algorithm 3 is equivalent to finding a solution for the quadratic Equation (9)

$$x^2 + (x_s - S_i^2)x + x_s^{-1}(-b + m_i^2) \equiv 0 \pmod{p}.$$

This equation admits a solution if and only if its discriminant, Δ, is either zero or a quadratic residue. Consequently, the probability that Equation (8) has a solution is equal to the probability that Δ is either zero or a quadratic residue. Given that $\mathbb{F}_p$ contains one zero and $\frac{p-1}{2}$ quadratic residues [32], we can ascertain that

$$\mathbb{P}(\Delta \text{ is zero or a quadratic residue}) = \frac{p+1}{2p}.$$

Furthermore, the probability of Algorithm 3 succeeding in finding a mapping point for a message block in exactly k rounds is $p_k = \left(\frac{p-1}{2p}\right)^{k-1}\left(\frac{p+1}{2p}\right)$. Thus, the algorithm produces a mapping point through n rounds with a probability of

$$\mathbb{P}_n = \sum_{k=1}^{n} p_k = \sum_{k=1}^{n}\left(\frac{p-1}{2p}\right)^{k-1}\left(\frac{p+1}{2p}\right) = 1 - \left(\frac{p-1}{2p}\right)^{n}.$$

However, due to the largeness of p, this can be effectively approximated as 1. Figures 9–12 illustrate the empirical determination of the requisite number of rounds to successfully map 1000 randomly selected message blocks throughout various NIST elliptic curves. The results indicate that our proposed method effectively accomplishes the mapping of 50% of blocks within the initial rounds, achieving complete mapping (100%) in 8 rounds over the majority of cases. These findings align efficiently with theoretical predictions, affirming that Algorithm 3 exhibits a probability ($\mathbb{P}_1$) exceeding $\frac{1}{2}$ to map a block within one round and a probability ($\mathbb{P}_8$) exceeding 0.9961 to map a block within 8 rounds. Nevertheless, as illustrated in Figure 11, the algorithm may necessitate additional rounds. Consequently, each encoded block undergoes padding with one digit from '1' to '9' on the left and another digit from '0' to '9' on the right until a mapping point is identified. As a result, the algorithm is designed to allow a maximum of 90 mapping rounds per block, thereby ensuring its correctness with a high degree of confidence ($\mathbb{P}_{90} \approx 1$).

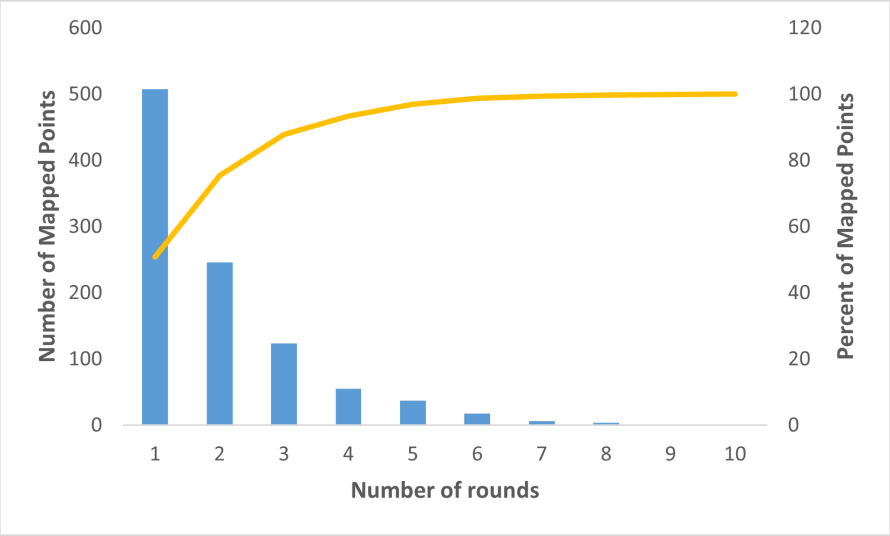


Figure 9. Algorithm performance analysis for secp192r1.

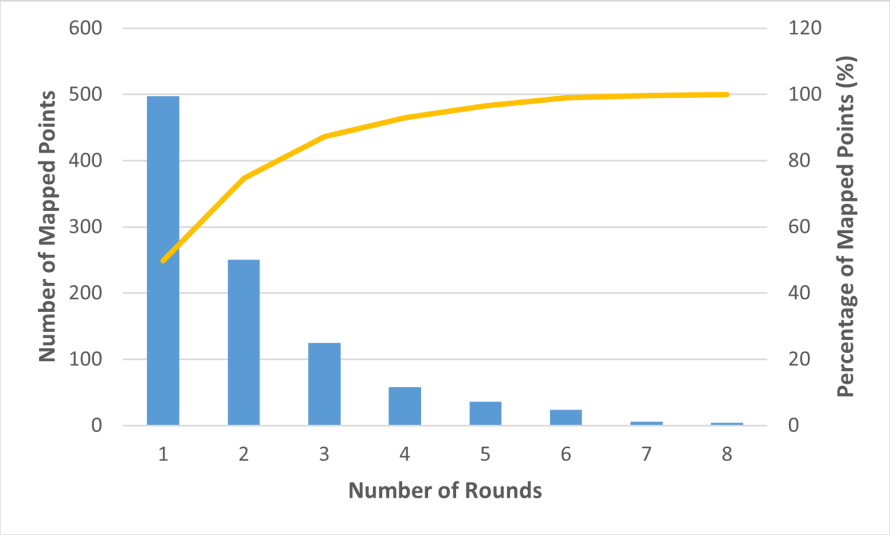


Figure 10. Algorithm performance analysis for secp256r1.

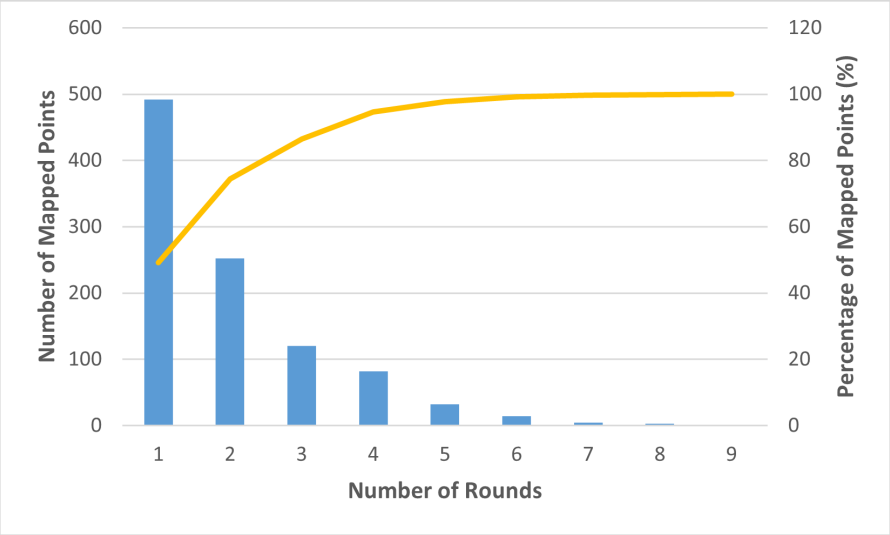


Figure 11. Algorithm performance analysis for secp384r1.

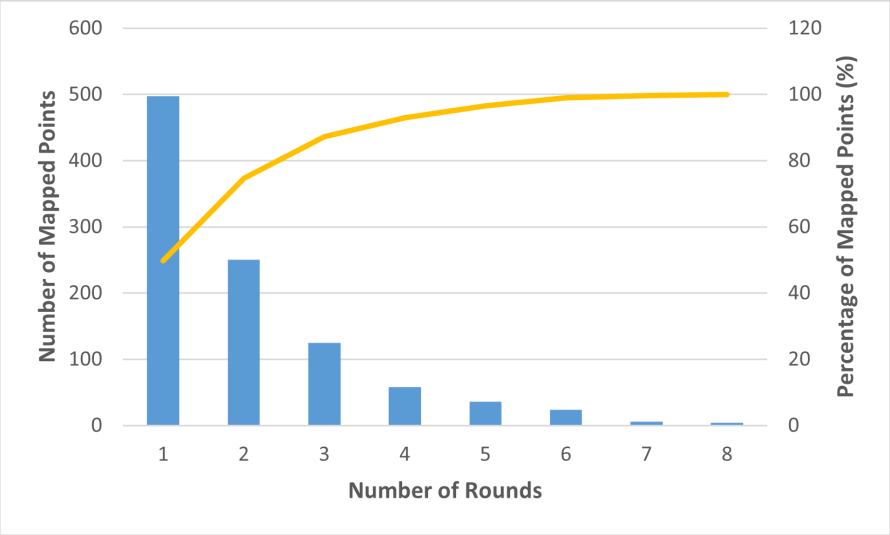


Figure 12. Algorithm performance analysis for secp521r1.

4.2. Complexity Analysis

The complexity of the overall encryption algorithm is influenced by the number of rounds needed to map message blocks onto a given elliptic curve. To assess this, the probability distribution of our proposed mapping algorithm allows us to determine the average number of rounds required to map a message block. Taking into consideration that p is a large prime number, this calculation is represented as follows:

$$\mu = \sum_{k=1}^{+\infty} k p_k = \sum_{k=1}^{+\infty} k \left(\frac{p-1}{2p}\right)^{k-1} \left(\frac{p+1}{2p}\right) = \frac{2p}{p+1} \approx 2.$$

In parallel, a statistical test is conducted by generating a random plaintext with a size of 10 KB. Subsequently, this plaintext is encrypted using various NIST elliptic curves, and the number of rounds required to map the message onto each curve is measured. The testing phase is carried out using the proposed methods and algorithms outlined in [16,24]. The results, as presented in Table 3, illustrate that the moderate number of rounds needed for each block is approximately 2, closely aligning with the theoretical result. Therefore, the computational complexity of the message mapping is $\mathcal{O}(2B)$, which is equivalent to $\mathcal{O}(n)$, where $B = \lfloor n/q \rfloor + \epsilon$, for a message of size n characters and blocks of size q characters.

Consequently, both the encryption and decryption processes have a complexity of $\mathcal{O}(n)$, as they only require B-point addition operations. Hence, the running time growth of message mapping, encryption, decryption, and reverse mapping is linear versus the size of the plaintext.

Table 3. Average rounds across 3 algorithms: Refs. [16,24], This paper.

Elliptic Curve	Algorithm	Random Plaintext		
		Num. of Blocks	Num. of Mapping Rounds	Average of Mapping Rounds
secp192r1	Ref. [16]	435	858	1.972
	Ref. [24]	435	891	2.048
	This paper	435	878	2.018
secp256r1	Ref. [16]	323	652	2.018
	Ref. [24]	323	658	2.037
	This paper	323	643	1.990
secp384r1	Ref. [16]	213	430	2.018
	Ref. [24]	213	421	1.976
	This paper	213	428	2.009
secp521r1	Ref. [16]	157	314	2
	Ref. [24]	157	317	2.019
	This paper	157	315	2.006

Figures 13–16 illustrate analyses of the running time of the proposed algorithm across various recommended NIST elliptic curves for different sizes of random plaintexts. It is important to note that the total encryption time comprises the time required for both message mapping and the encryption algorithm itself. Similarly, the total decryption time includes both reverse mapping and the decryption algorithm execution. The experimental findings align with the theoretical results, depicting timing graphs for all processes that exhibit a consistent linear increase with a low slope concerning plaintext size.

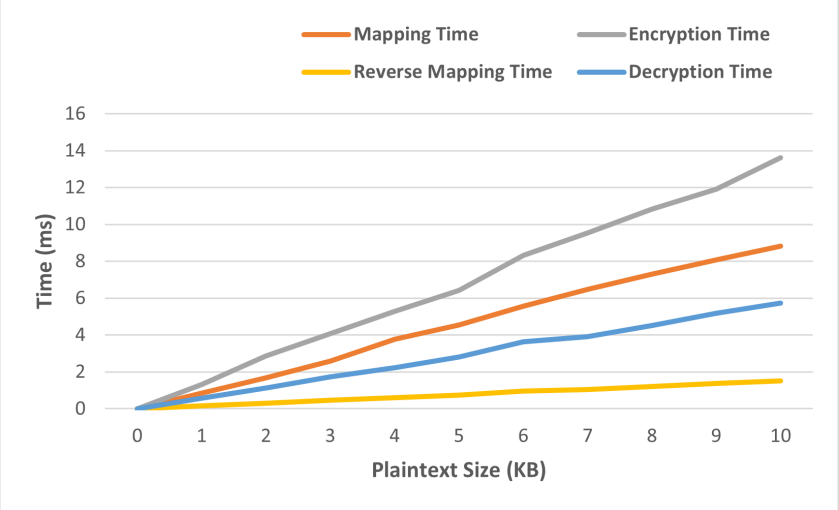


Figure 13. Algorithm performance analysis: running time vs. plaintext size for secp192r1.

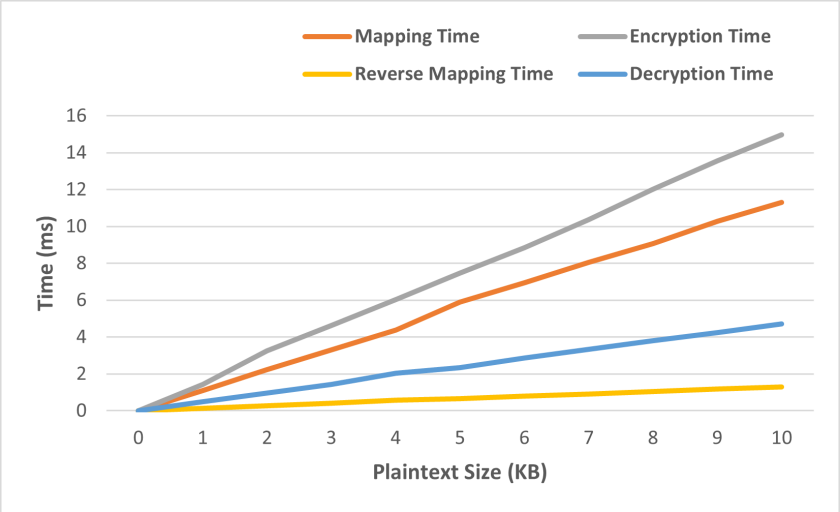


Figure 14. Algorithm performance analysis: running time vs. plaintext size for secp256r1.

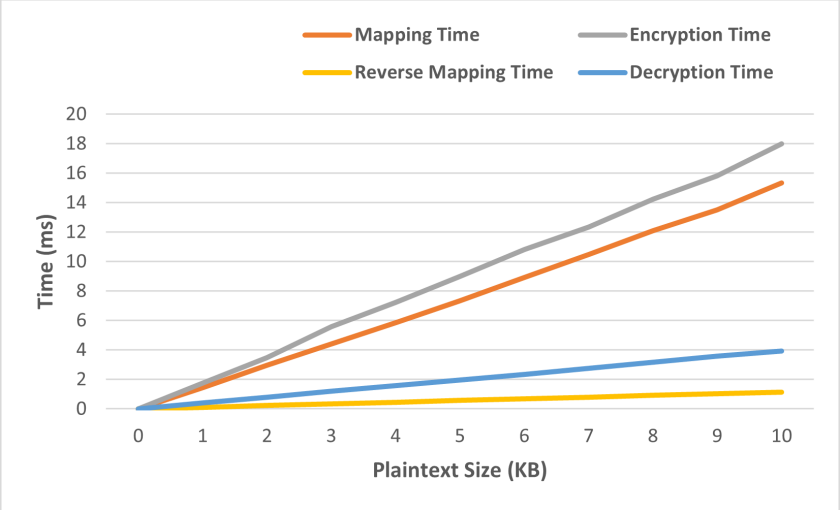


Figure 15. Algorithm performance analysis: running time vs. plaintext size for secp384r1.

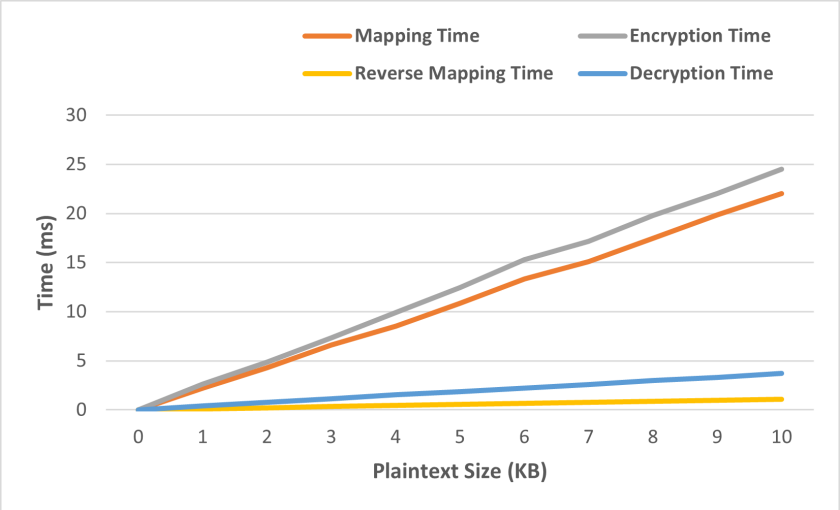


Figure 16. Algorithm performance analysis: running time vs. plaintext size for secp521r1.

5. Sensitivity Assessment and Security Analysis

5.1. Sensitivity Assessment

5.1.1. Key Sensitivity Analysis

Key sensitivity describes how a ciphertext responds to small changes in the encryption key and reflects the impact of the key on the encryption procedure. If this influence is substantial, around 50%, it indicates that the encryption scheme is key-sensitive. Low sensitivity can make a system vulnerable to various attacks, such as differential attacks.

Throughout the analysis, we use the NIST elliptic curve spec256k1, and binary key spaces $I_i = \{K_{i,j} \in \{0,1\}^{256} / j = 1 \dots 101\}$ are created, where $K_{i,j}$ is obtained by altering randomly one bit of the randomly chosen key $K_{i,1}$, where all keys in $I = \bigcup_{1 \le i \le 100} I_i$ are pairwise distinct, and a random plaintext, m, is used. Then, we run the key sensitivity analysis on the following model:

$$\begin{cases} C_i = \texttt{Enc}(m, I_i), \\ i = 1 \dots 100, \end{cases} \tag{10}$$

where C_i is the set of ciphertexts obtained by encrypting a message, m, using a key in I_i each time. Furthermore, we measure the rate of change by XORing the first cipher with all remaining ciphers in C_i, then counting the number of bits equal to 1. Similar key sensitivity analyses were carried out for the mapping scheme. The findings are presented in Figures 17, 18, and Table 4, showing that the proposed scheme fits the confusion property and the strict avalanche criterion [33], with an average change rate of 50% and only slight deviations.

Table 4. Change rate analysis for message mapping and encryption: average rate and deviation.

Change Rate of	Number of Tests	μ	σ
Mapping scheme	10000	50.10%	2.57%
Encryption scheme	10000	50.02%	2.81%

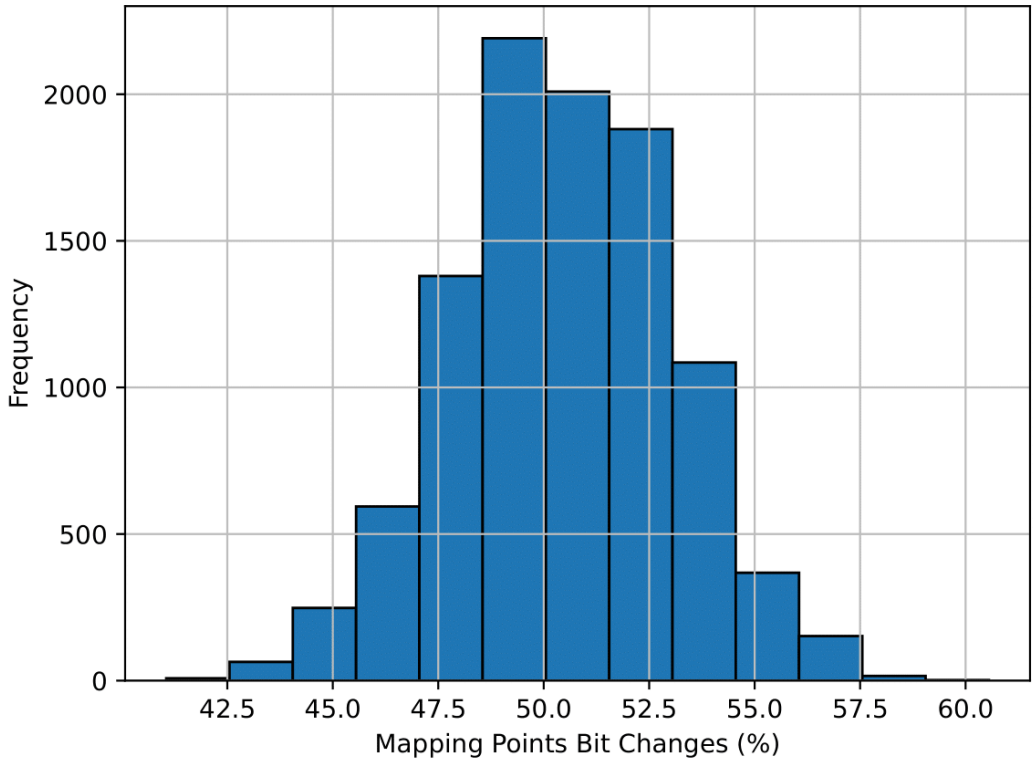


Figure 17. Mapping change rate from a single-bit key alteration.

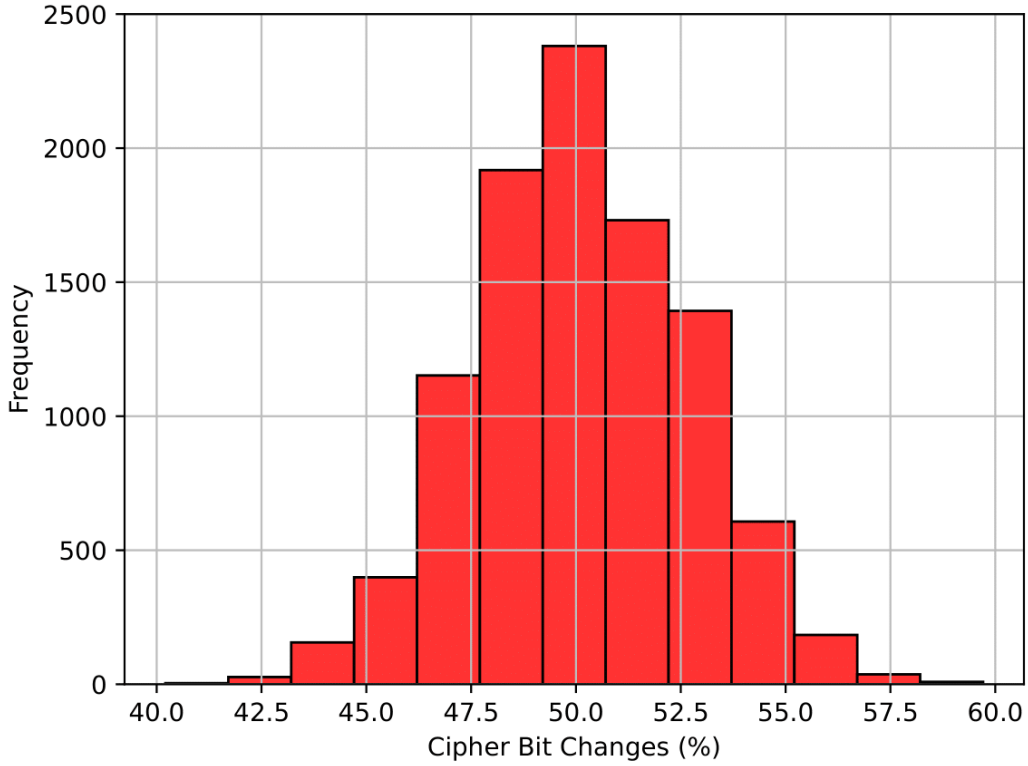


Figure 18. Ciphertext change rate from a single-bit key alteration.

5.1.2. Plaintext Sensitivity Analysis

Plaintext sensitivity measures the rate of change in the ciphertext with respect to a slight bit alteration in the plaintext, while the encryption key stays unchanged. In this test, a random plaintext, m, with 26 characters is selected; we generate a sequence of plaintexts $(m_{i_j})_{1\leq j\leq 100}$ retrieved by altering the bit number i_j of m each time, where i_j is chosen randomly. The resulting data are encrypted with the same key, together with m. Then, we analyze the changes in mapping points and ciphertexts. Figure 19 displays the mapping and ciphertext change rate values, which fall within the interval $[48\%, 52\%]$. These values accurately align with the principles of plaintext sensitivity. Moreover, the scatter plots of mapping points and cipher points show no correlation, as indicated in Figures 20 and 21.

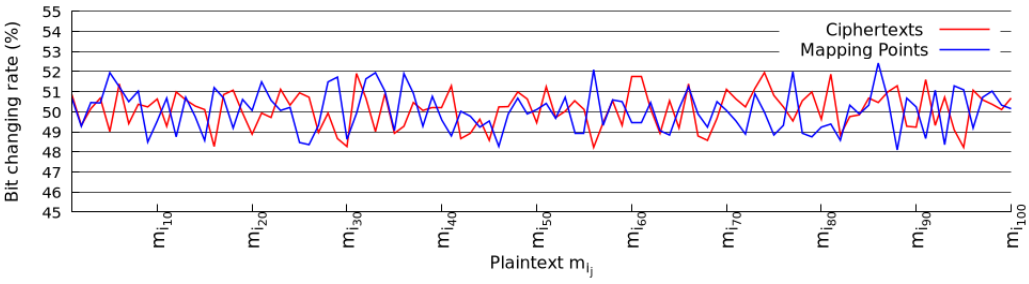


Figure 19. Bit changing rate under plaintext bit alteration.

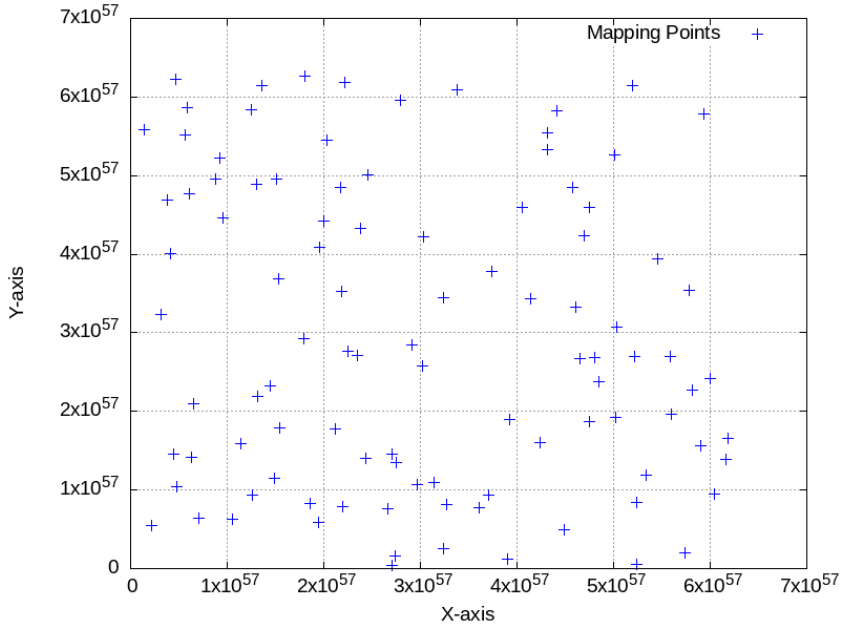


Figure 20. Mapping points scatter.

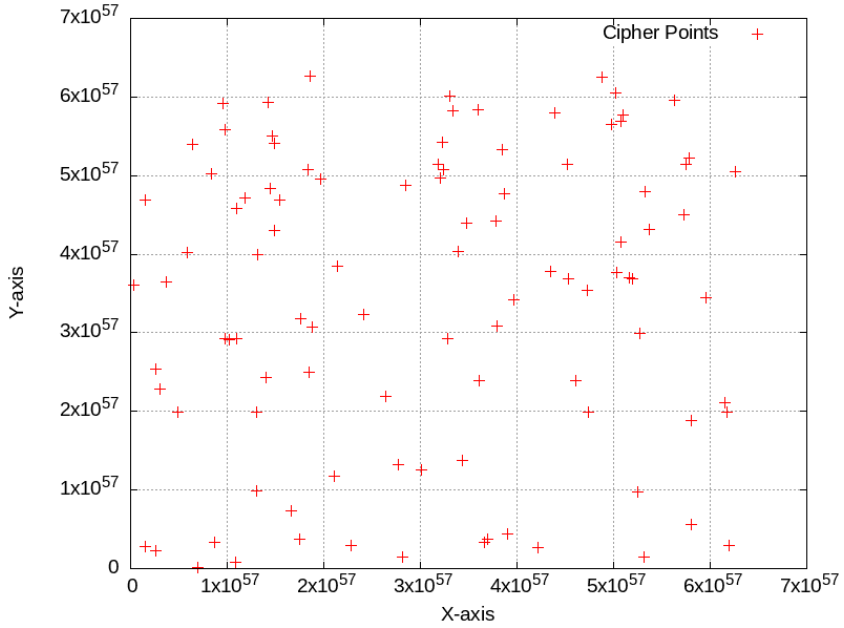


Figure 21. Ciphertext points scatter.

5.1.3. Statistical Analysis

The NIST Statistical Test Suite consists of statistical tests used to measure the randomness of data generated by a pseudo-random number generator (PRNG) or a cryptographic algorithm. To evaluate the randomness of data generated by the proposed encryption scheme, we encrypt random plaintexts and store the resulting ciphertexts as binary sequences in a file of size 20 MiB, and we set the significance level at $\alpha = 0.01$. The algorithm's ciphertexts pass a test if the number of rejected sequences is less than or equal to $n\left(\alpha + 3\sqrt{\frac{\alpha(1-\alpha)}{n}}\right)$, where n represents the number of tested sequences. The length of sequences being tested is initially set at 2000 bits but may vary based on specific testing requirements. In assessing the data randomness, we conducted NIST statistical tests, which generally yielded positive results. However, the serial test, as documented in Table 5, revealed an anomaly. As depicted in Figure 22, the remarkable consistency between the

expected and observed counts of rejected sequences underscores the continued randomness of the ciphertext data from an eavesdropper's perspective. This finding serves as evidence of the algorithm's resilience against statistical attacks.

Table 5. NIST statistical tests assessment.

ID	Test	Number of Sequences	Length of Sequences (bits)	Assessment
01	Monobit Test	8198	2000	Pass
02	Frequency Within Block Test	8198	2000	Pass
03	Runs Test	8198	2000	Pass
04	Longest Run of Ones in a Block Test	8198	2000	Pass
05	Binary Matrix Rank Test	279	77841	Pass
06	DFT Test	8198	2000	Pass
07	Non-Overlapping Template Matching Test	8198	2000	Pass
08	Overlapping Template Matching Test	10	2056039	Pass
09	Maurer's Universal Test	28	775698	Pass
10	Linear Complexity Test	21	1000033	Pass
11	Serial Test	57392	384	Fail
12	Approximate Entropy Test	8198	2000	Pass
13	Cumulative Sums Test	8198	2000	Pass
14	Random Excursion Test	10	2000072	Pass
15	Random Excursion Variant Test	10	2000072	Pass

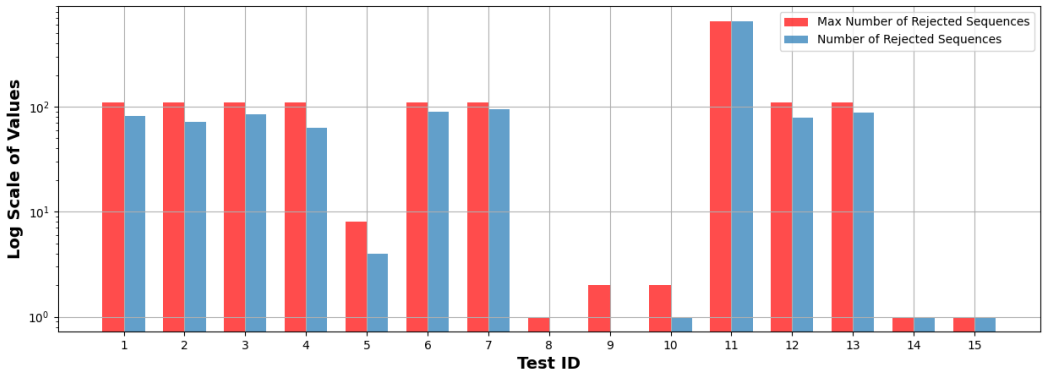


Figure 22. Rejected sequences and max rejections in NIST statistical tests.

5.2. Security Analysis

5.2.1. Key Spacing Analysis and Brute Force

Due to the high computing abilities of modern computers, it is crucial to ensure that the key size is sufficiently large to be immune from any probable brute force attack. To withstand such an attack, the key spacing should be greater or equal to 2^{100} [34]. As mentioned in [35], to process data and implement protection measures, NIST recommends using a minimum of 112-bit security strength until at least 2030, necessitating an elliptic curve with a base point order of 224 bits [36]. Even if we exclusively use keys greater than 2^{224}, we still have 2^{223} distinct secret keys, a significantly large number compared to the recommended range. This demonstrates that our proposed scheme satisfies key spacing requirements, thereby providing security against exhaustive search as a form of brute force attack.

5.2.2. Known Plaintext Attack (KPA)

In this attack, the adversary's objective is to extract additional information by leveraging knowledge of some plaintext and ciphertext pairs. The proposed message mapping algorithm attaches n characters (elements of an extended ASCII table) to a single point on the curve and then encrypts the results. Hence, the likelihood of encountering an identical sequence of characters is $\frac{1}{256^n}$. Furthermore, the use of a random key, k, and parameter, *Counter*, makes the algorithm produce different mapping points and ciphertexts for the same message. In addition, the mapping mechanism uses two secret points, K_{sh} and kP_r.

Thus, the attacker cannot generate a mapping point for a known message, owing to the assumed hardness of the elliptic curve discrete logarithm problem (ECDLP). For these reasons, the known plaintext attack is infeasible.

5.2.3. Ciphertext-Only Attack (COA)

Selecting an elliptic curve, a base point, and a prime field is crucial to the security of the scheme. In the proposed method, we recommend implementing standardized elliptic curve parameters to ensure the intractability of ECDLP and to prevent brute force attacks due to the large key space. In addition to the high entropy observed in the generated ciphers, frequency attacks are not feasible, as discussed in the previous subsection. Hence, the proposed method is secure against such attack models.

5.2.4. Chosen-Plaintext Attack (CPA)

Ensuring security against a chosen-plaintext attack (CPA) is crucial for concealing information within ciphertexts, thus upholding the confidentiality of an encryption scheme. In this attack, the adversary, equipped with an encryption oracle, requests ciphertexts for chosen plaintexts to extract secret information or identify patterns. Advanced security definitions, such as IND-CPA (indistinguishability under chosen-plaintext attack), extend the CPA model to provide a more comprehensive evaluation of encryption system security against such attacks. In IND-CPA, the adversary not only requests ciphertexts for chosen plaintexts but is also challenged to guess which one among a pair of chosen plaintexts is encrypted by the challenger based on his/her request. More formally, the steps in the adversary and challenger indistinguishability game under CPA are as follows:

1. The challenger generates a key pair (P_u, SK) with a security parameter k, shares the public key P_u with the adversary, and keeps the private key SK.
2. The adversary requests ciphertexts for a polynomially bounded number of chosen plaintexts and/or performs other operations.
3. The adversary submits two distinct chosen plaintexts, M_0 and M_1, with the same size.
4. The challenger selects a bit $b \leftarrow \{0, 1\}$ randomly, and sends the ciphertext $C = Enc(P_u, M_b)$ back to the adversary.
5. The adversary can conduct any desired number of additional computations or encryptions.
6. In the end, the adversary outputs a conjecture for the value of b.

The advantage of the adversary, A, in winning the IND-CPA game against the encryption system $\Pi = (Gen, Enc, Dec)$ is computed as $Adv^{\Pi}_{\text{IND-CPA}}(A, P_u) = |\mathbb{P}(A \text{ win the game}) - \mathbb{P}(A \text{ lose the game})|$. The cryptosystem Π is considered IND-CPA secure if the adversary's advantage is negligible, specifically expressed as

$$Adv^{\Pi}_{\text{IND-CPA}}(A, P_u) = |\epsilon(k)|,$$

where $\epsilon(k)$ is a negligible function in the security parameter k.

Claim 1. *Let $\Pi = (Gen, Enc, Dec)$ be an ECC-based encryption system, where the encryption scheme Enc uses ElGamal-v2, and a message mapping algorithm, denoted as $Mapp$, which operates independently of a secret key. Then Π is insecure under CPA.*

Proof. In this proof, we start by showing that Π is not IND-CPA. Subsequently, we elucidate how this vulnerability facilitates the recovery of secret information. Finally, we conclude that Π is insecure under CPA.

Consider an adversary, A, who selects and transmits two plaintexts, M_0 and M_1, of the same size to the challenger. For each $i \in \{0, 1\}$, the encoding of M_i results in two message blocks. Consequently, the challenger chooses a bit, b, randomly, and responds with the ciphertext, $C_b = Enc(M_b)$, containing two cipher points, $C_{b,0}$ and $C_{b,1}$. These cipher points are computed as $C_{b,i} = P_{b,i} + kP_u$, where kP_u is the point multiplication of the public key, P_u, by a random integer, k, and $P_{b,0}$ and $P_{b,1}$ represent the mapping points

of the first and second message blocks, respectively. Since the mapping algorithm is key-independent, the adversary computes the message mapping point $Mapp(M_0) = \{P_{0,0}, P_{0,1}\}$ of plaintext M_0. Subsequently, he/she computes $V_0 = P_{0,0} - P_{0,1}$ and $V_b = C_{b,0} - C_{b,1} = (P_{b,0} + kP_u) - (P_{b,1} + kP_u) = P_{b,0} - P_{b,1}$. Then, if $V_0 = V_b$, the adversary guesses that $b = 0$, else, $b = 1$. Thus, $Adv^{\Pi}_{\text{IND-CPA}}(A, P_u) = 1$, proving that Π is not IND-CPA. Furthermore, suppose that the adversary found that $b = 0$, they can compute kP_u as $C_{b,0} - P_{b,0}$ and use it to decrypt or encrypt any messages of their choice. In conclusion, the Π is insecure under CPA. This attack holds true even when the encryption system processes character by character. □

Claim 2. *The proposed scheme $\Pi_{ps} = (Gen_{ps}, Enc_{ps}, Dec_{ps})$ is CPA secure, and any probabilistic polynomial-time adversary cannot win the IND-CPA game with a probability better than a random guessing.*

Proof. The proposed scheme employs ElGamal-v2. The highlighted attack in the preceding proof is inapplicable due to the key-dependent nature of the mapping algorithm. Furthermore, by incorporating the shared secret point, K_{sh}, and the parameters *Counter* (incremented after each successful encryption attempt) and a random key k to derive the point $kP_r = (x_k, y_k)$, initiating the encoding process with $m_1 = m'_1 \oplus 1 \oplus H^{(Counter)}(x_k)$, and encoding the subsequent blocks in a manner that mirrors CBC behavior ($m_i = m'_i \oplus i \oplus m'_{i-1}$ for $i = 2, 3, \ldots, B$), this guarantees the generation of distinct mapping points even when faced with identical message blocks. Consequently, different cipher points C_m may arise for a message block m within the same encryption cycle or in a new cycle, introducing a non-one-to-one relationship between plaintext and ciphertext. Therefore, for any given plaintext M, it holds that $Enc^{(i)}_{ps}(M) \neq Enc^{(j)}_{ps}(M)$, where $Enc^{(i)}_{ps}$ and $Enc^{(j)}_{ps}$ denote distinct encryption cycles within the proposed scheme. Furthermore, the encryption scheme Enc_{ps} demonstrates a change rate of 50% in response to minor alterations in either the plaintext or the key and produces random ciphertexts from the adversary's perspective, as elaborated in Section 5.1. To be more precise, each bit of the ciphertext bears a $\frac{1}{2}$ probability of modification with even slight changes in the key or the plaintext. In the IND-CPA game, an adversary's strategy is no more effective than random guessing ($Adv^{\Pi_{ps}}_{\text{IND-CPA}}(A, P_u) =$ negligible), and the security of the scheme is directly linked to the adversary's ability to solve the ECDLP, making private key extraction or secret information from knowledge about pairs of chosen plaintexts infeasible. Therefore, the proposed scheme Π_{ps} is IND-CPA and provides a high level of security against chosen-plaintext attacks. □

Based on Claim 1, it can be asserted that the ECC-based encryption schemes [4,13,14,22] are vulnerable to CPA. This vulnerability arises from the incorporation of ElGamal-v2 and a key-independent message mapping, resulting in the generation of identical ciphertexts for a given message across different encryption cycles. Attempts to mitigate this potential vulnerability, as proposed in [8,24] by using a new initial vector (IV) for each encryption cycle. However, it is suggested that this measure falls short, as adversaries could still leverage the received IVs along with the ciphertext to compute mapping points and execute the described attack outlined in the proof of Claim 1. Furthermore, the proposed scheme succeeds in achieving CPA security with a balanced complexity and bandwidth usage, unlike other related encryption schemes that achieve CPA security through the incorporation of ElGamal-v1, which demands substantial resources in terms of power, computation, and bandwidth.

5.2.5. Malleability

Malleability describes the capacity of an adversary to create a valid ciphertext from a given one, so it can be decrypted to a plaintext related to the original plaintext. In other words, let C_m be the ciphertext corresponding to a plaintext, m, an attacker can create a new ciphertext, C, which can be decrypted to $f(m)$ for a known function, f; such an encryption scheme is malleable [37].

In our proposed method (see Section 3.4), a ciphertext of a message, m, is of the form

$$C = \{kG, (C_{m_i})_{1 \leq i \leq B}, ID_s, C_{m^*}\}.$$

To exploit the malleability of such a cipher, an adversary has to add a chosen point, Q, to a cipher point C_{m_i} in a manner that the receiver decrypts the new cipher $C_{\widetilde{m_i}} = Q + P_m + kP_r$, and obtains $P_{\widetilde{m_i}} = Q + P_m$ instead of P_{m_i}. Once all mapping points are recovered by the receiver (r), verification is initiated through the computation of

$$\widetilde{m^*} = H(x_k \oplus y_k | x_{m_1} \oplus y_{m_1} | \ldots | x_{\widetilde{m_i}} \oplus y_{\widetilde{m_i}} | \ldots | x_{m_B} \oplus y_{m_B} | ID_s).$$

The receiver then computes and encrypts its corresponding mapping point $P_{\widetilde{m^*}}$ to $C_{\widetilde{m^*}}$. The decryption is rejected if $C_{\widetilde{m^*}} \neq C_{m^*}$. Moreover, $\widetilde{m^*}$ is sensitive to any changes in ciphertext parts, even a small change like the order of cipher points $(C_{m_i})_{1 \leq i \leq B}$, which preserves the integrity of the message. From the previous reasons, we conclude that the proposed method is a non-malleable scheme.

5.2.6. Replay Attack (RA)

The replay attack is a significant security threat within the realm of cryptographic communication. In this type of attack, an adversary intercepts and then maliciously retransmits previously captured data or messages with the intention of deceiving the recipient system or gaining unauthorized access. In [38], the authors executed a replay attack against an implantable cardioverter-defibrillator (ICD) device, successfully compromising the safety and privacy of the simulated patient. Figure 23 illustrates a scenario involving a replay attack against a healthcare provider. In this scenario, the attack deceives a doctor by causing them to repeat a telemedicine action, even though the adversary is merely retransmitting an old ciphertext from the sender. This example highlights the real-world implications and risks associated with replay attacks in various contexts, including healthcare and beyond.

In the proposed scheme, a shared parameter named *Counter* starts at zero and is consistently incremented during both the encryption and the decryption, ensuring synchronization between the sender and receiver during communication. This mechanism prevents the reuse of the same *Counter*, which is included in the ciphertext's tag C^*, and as demonstrated in the malleability Section 5.2.5, this tag is not forgeable. Hence, the decryption oracle rejects all received ciphertexts with an outdated *Counter* and returns $\perp$. Based on the preceding arguments, it can be concluded that the proposed scheme is secure against replay attacks (RAs).

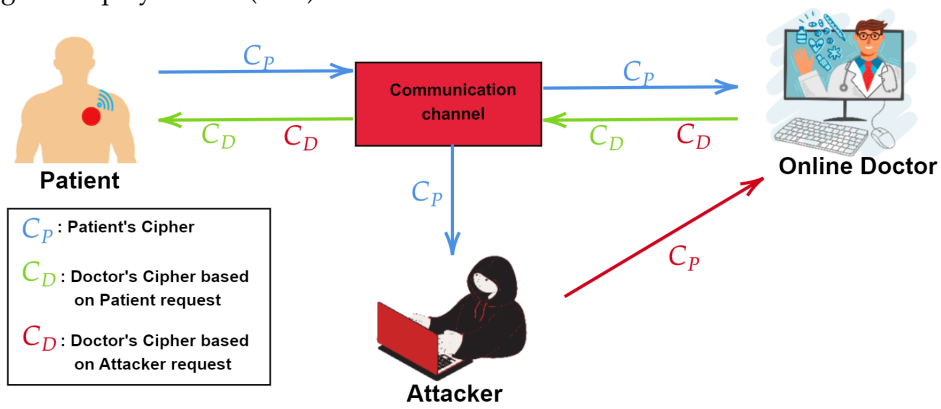


Figure 23. Replay attack against healthcare provider.

5.2.7. Chosen-Ciphertext Attack (CCA)

In addition to ensuring security against CPA, it is crucial to consider security against chosen-ciphertext attacks (CCAs) to further strengthen the encryption system. In a CCA attack, the adversary gains access to the decryption oracle, allowing them to request

the decryption of ciphertexts of their choice. This attack aims to exploit vulnerabilities in the decryption process and potentially reveal sensitive information. To evaluate the security of an encryption system against CCA, advanced security definitions like IND-CCA (indistinguishability under chosen-ciphertext attack) are employed. In IND-CCA, the adversary is not only allowed to request ciphertexts for chosen plaintexts but can also submit ciphertexts for decryption and observe the corresponding plaintexts. The goal is to make it computationally infeasible for the adversary to distinguish between two equal-length ciphertexts, one encrypted under a random key and the other either encrypted under the target key or chosen at random. In addition, the adversary engages in an IND-CCA game with the challenger under the following steps.

1. The challenger generates a key pair (P_u, SK) with a security parameter k, shares the public key P_u with the adversary, and keeps the private key SK.
2. The adversary requests ciphertexts for a polynomially bounded number of chosen plaintexts and/or performs other operations.
3. The adversary requests the decryption of a polynomially bounded number of chosen ciphertexts and/or performs other operations.
4. The adversary submits two distinct chosen ciphertexts, C_0 and C_1.
5. The challenger selects a bit $b \leftarrow 0, 1$ randomly, and sends the decryption $M = Dec(SK, C_b)$ back to the adversary.
6. The adversary can conduct any desired number of additional computations or encryptions.
7. In the end, the adversary outputs a conjecture for the value of b.

The advantage of the adversary, A, in winning the IND-CCA game against the cryptosystem $\Pi = (Gen, Enc, Dec)$ is computed similarly to IND-CPA

$$Adv^{\Pi}_{\text{IND-CCA}}(A, P_u) = |\mathbb{P}(A \text{ win the game}) - \mathbb{P}(A \text{ lose the game})|.$$

The cryptosystem Π is considered IND-CCA secure if the adversary's advantage is negligible, expressed as $Adv^{\Pi}_{\text{IND-CCA}}(A, P_u) = |\epsilon(k)|$, where $\epsilon(k)$ is a negligible function in the security parameter, k. The combination of security against CPA and CCA ensures a robust and secure encryption system.

Claim 3. *Let $\Pi = (Gen, Enc, Dec)$ be an ECC-based encryption system, where the encryption scheme Enc uses ElGamal-v2, and a message mapping algorithm, denoted as $Mapp$, which operates independently of a secret key. Then Π is insecure under CCA.*

Proof. Similar to the proof of Claim 1, the adversary chooses and submits two ciphertexts, C_0 and C_1, of the same size, ensuring that each ciphertext, C_i, contains at least two cipher points, $C_{i,0}$ and $C_{i,1}$, where $i \in \{0, 1\}$.

Subsequently, the challenger randomly chooses a bit $b \leftarrow \{0,1\}$, decrypts $M_b = \text{Dec}(C_b, SK)$, and sends back the result to the adversary. To distinguish whether the challenger decrypted the ciphertext C_0 or C_1, the adversary computes $Mapp_(M_b) = \{P_{b,0}, P_{b,1}\}$ as the mapping points of M_b and the following values:

$$\begin{cases} V_b = P_{b,0} - P_{b,1}, \\ V_0 = C_{0,0} - C_{0,1} = (P_{0,0} + kP_u) - (P_{0,1} + kP_u) = P_{0,0} - P_{0,1}. \end{cases} \tag{11}$$

Then, if $V_b = V_0$, the adversary deduces that $b = 0$, otherwise, they state that $b = 1$. In this strategy, the adversary's probability of winning the IND-CCA game is 1, hence, $Adv^{\Pi}_{\text{IND-CCA}}(A, P_u) = 1$, and Π is insecure under IND-CCA. Once b is computed, they recover kP_u as a result of $C_{b,0} - P_{b,0}$ and use it for further encryptions or decryptions. In conclusion, Π is insecure under CCA. □

Claim 4. *The proposed scheme $\Pi_{ps} = (Gen_{ps}, Enc_{ps}, Dec_{ps})$ is CCA secure, and any probabilistic polynomial-time adversary cannot win the IND-CCA game with a probability better than a random guessing.*

Proof. The proposed scheme Π_{ps} effectively guards against replay attacks. Consequently, the decryption oracle within this scheme is designed to reject any replayed ciphertext during the tag C^* verification process. The adversary's ability to generate valid ciphertexts is then contingent upon successfully forging a tag or encrypting chosen plaintexts. Achieving such feats is only possible by solving the ECDLP. This is due to the fact that the message mapping, $Mapp_{ps}$, encryption, Enc_{ps}, and tag, C^*, creation in Π_{ps} require secret keys such as K_{sh}, kP_r, and additional parameters like $Counter$. As a result, the proposed scheme, Π_{ps}, demonstrates indistinguishability under chosen-ciphertext attack, providing robust protection against CCA. □

According to Claim 3, the ECC encryption schemes [4,13,14,22] exhibit vulnerabilities against chosen-ciphertext attacks, allowing adversaries to successfully win the IND-CCA game with a probability of 1. Moreover, diverse methods advocate the use of random initial vectors (IVs) for each encryption attempt, aiming to ensure distinct ciphertexts for identical plaintexts when employing a block cipher, as proposed in [8,24]. Nevertheless, adversaries can leverage weaknesses in these schemes to achieve success in the IND-CCA game. The process involves computing $\{P_{b,0}, P_{b,1}\} = Mapp(M_b, IV_0)$ and $\{P'_{b,0}, P'_{b,1}\} = Mapp(M_b, IV_1)$, representing the mapping points of the recovered message M_b using initial vectors IV_0 and IV_1 for ciphertexts C_0 and C_1, respectively. We define the following values as follows:

$$\begin{cases} V_b = P_{b,0} - P_{b,1} \text{ and } V'_b = P'_{b,0} - P'_{b,1}, \\ V_0 = C0,0 - C_{0,1} = (P_{0,0} + kP_u) - (P_{0,1} + kP_u) = P_{0,0} - P_{0,1}. \end{cases} \tag{12}$$

If $V_b = V_0$ or $V'_b = V_0$, the adversary outputs $b = 0$, otherwise, they output $b = 1$. This assertion remains valid even in the presence of integrity mechanisms, assuming that C_0 and C_1 are captured in communication traffic and replayed without alterations. Therefore, these cryptosystems are lacking in terms of indistinguishability under the chosen-ciphertext attack (IND-CCA).

5.2.8. Side-Channel Attacks

Side-channel attacks [39] are a significant threat to cryptographic systems, including ECC-based implementations. These attacks exploit unintended information leakage from the physical implementation of a cryptographic algorithm, such as timing information, power consumption, electromagnetic leaks, or even sound, to infer secret data.

Recent studies emphasize the importance of mitigating these attacks through various countermeasures. In [40], Socha et al. provided a comprehensive overview of the impact of side-channel attacks on both hardware and software cryptographic implementations. Additionally, Vanhoef et al. [41] presented a timing attack against an encoding method to elliptic curve points, exploiting timing leaks generated by arithmetic operations behind ECC. Effective countermeasures such as constant-time operations, point blinding, and masking have been highlighted as essential strategies to protect against these attacks [42]. Implementing these strategies in our ECC-based cryptosystem is crucial and will be a key area for future work to ensure the robustness of our cryptosystem against such attacks.

5.2.9. Man-in-the-Middle Attack (MitM)

Based on the hardness assumption of the ECDLP, our scheme provides security against several attacks, effectively mitigating many weaknesses as discussed in Section 5 and preventing the extraction of secret keys involved during its process (e.g., K_{sh}, kP_r, users' private keys, etc.). However, a significant concern arises regarding the possibility of adversaries impersonating legitimate users through man-in-the-middle (MitM) attacks. This form of attack involves intercepting and substituting public keys with malicious ones, thus compromising the confidentiality and integrity of data exchanged between users. To mitigate this threat in our proposed scheme, robust verification mechanisms are essential. Implementing protocols for public key authentication enables users to verify

the legitimacy of received keys [43] while leveraging certificate authorities (CAs) provides digital certificates to validate the identities of communication parties [44]. Additionally, adopting trust-on-first-use (TOFU) mechanisms allows users to verify public keys during initial interactions [45]. Continuous monitoring and verification of public key authenticity in real-time, using automated systems equipped with anomaly detection algorithms, helps detect and mitigate impersonation attempts [46]. These measures collectively strengthen the resilience of our encryption method against MitM attacks, ensuring secure data exchange between users.

Table 6 presents a security comparison between the proposed methods and schemes in some related works, where the symbols ✓ (representing 'Secure') and × (representing 'Not Secure') are used to denote the resilience or vulnerability against specific several attacks.

Table 6. Comparison of schemes based on security against various attacks.

Scheme	Security					
	KPA	**COA**	**CPA**	**Malleability**	**Replay Attack**	**CCA**
Refs. [20,21]	✓	✓	✓	×	×	×
Refs. [13,22]	✓	✓	×	×	×	×
Ref. [24]	✓	✓	×	✓	×	×
This paper	✓	✓	✓	✓	✓	✓

6. Implementation

To implement the proposed scheme, we leverage the parameters and computations specified in Table 7. First, we utilize the elliptic curve parameters defined in [47], including the coefficients a and b, the prime number p, the generator point G, and its order $\#G$. Next, we proceed with the key exchange process between a sender (s) and a receiver (r). The sender generates their private key u_s and computes their public key P_s using scalar multiplication with the generator point G. Similarly, the receiver generates their private key u_r and computes their public key P_r. During the key exchange, the sender derives points kP_r and kG using a random parameter k and the receiver's public key, while the receiver derives the same point kP_r using their private key u_r and point kG. This point kP_r acts as a session key, providing forward secrecy by ensuring that each session uses a new and unique key. Therefore, it is essential to update kP_r for each new session by choosing a new random number k. Finally, both parties compute the shared secret point using their respective private keys and the derived points. This shared secret point serves as the basis for establishing a secure communication channel between the sender and the receiver.

To encrypt the message M = "Cryptography", we have to compute the block size $q = \lfloor (l_p - 2)/\log(256) \rfloor = 3$, where $l_p = 10$ represents the number of digits of the prime p. Then we segment the message into four blocks: B_1 = "Cry", B_2 = "pto", B_3 = "gra", and B_4 = "phy". Each block B_i is then converted to a decimal value m_i, which is encoded to m_i' using the proposed encoding process in Figure 2. This encoding process incorporates a hash value $H^{(Counter)}(x_k)$, computed for two encryption attempts, numbered $Counter = 1$ and 2, where the hash function used is SHA256 is adjusted to define $H^{(Counter)}(.)$. Subsequently, each m_i' is mapped to a point P_{m_i} using Algorithm 3. Furthermore, for message integrity verification, the integer m^* is computed as described in Algorithm 4, then mapped to P_{m^*} prior to tag creation. Finally, ElGamal-v2 is used to compute cipher points as $C_{m_i} = P_{m_i} + kP_r$, and the tag as $C_{m^*} = P_{m^*} + kP_r$. Mapping details for encryption attempts numbered $Counter = 1, 2$ are provided in Tables 8 and 9, respectively.

Table 7. Elliptic curve key parameters and setup computations.

Section	Parameter	Value
EC Parameters	Equation's Coefficient a	537680305
	Equation's Coefficient b	1059676324
	Prime Number p	3946183951
	Generator Point G	(1152222263, 3133703258)
	Generator Point Order $\#G$	3946206427
	Elliptic Curve Equation: $y^2 = x^3 + 537680305x + 1059676324$	
Sender (s) Parameters	Private Key u_s	2441052953
	Random Parameter k	655846922069
	Public Key $P_s = u_sG$	(2365961577, 2642871165)
	Point kG	(2634041647, 2827619405)
	Point $kP_r = (x_k, y_k)$	(1135983708, 381375075)
	Shared Secret Point $K_{sh} = u_sP_r$	(3400454298, 1533875807)
Exchange	Send $P_r \uparrow$	Send $P_s, kG \downarrow$
Receiver (r) Parameters	Private Key u_r	1429753181
	Public Key $P_r = u_rG$	(2830164285, 942020493)
	Shared Secret Point $K_{sh} = u_rP_s$	(3400454298, 1533875807)
	Point $kP_r = u_rkG = (x_k, y_k)$	(1135983708, 381375075)

Table 8. Mapping details for encryption attempt number 1 ($Counter = 1$).

Block B_i	Decimal Value m_i	Encoded Value m_i'	Mapping Point P_{m_i}
B_1	4420217	6419119	(1851134498, 2848058768)
B_2	7369839	1148610	(3450323358, 1700398742)
B_3	6779489	7795872	(1240965444, 3560781816)
B_4	7366777	433373	(2171276352, 3891369897)

Mapping points P_{m_i}, point kP_r, $Counter$, and sender's ID_s = "Sender (s)" hashed to $m^* = 2186729$, then mapped to $P_m^* = (3378637932, 274720553)$ prior to tag creation.

Table 9. Mapping details for encryption attempt number 2 ($Counter = 2$).

Block B_i	Decimal Value m_i	Encoded Value m_i'	Mapping Point P_{m_i}
B_1	4420217	4346998	(2024477685, 3189489356)
B_2	7369839	3285019	(2313133495, 3782223150)
B_3	6779489	5591673	(1363974206, 480205177)
B_4	7366777	2439684	(3918942662, 300263470)

Mapping points P_{m_i}, point kP_r, $Counter$ and sender's ID_s = "Sender (s)" hashed to $m^* = 2355984$, then mapped to $P_m^* = (1535397971, 3904424142)$ prior to tag creation.

A step-by-step example illustrating the different rounds required to map a message $m_1 = 6419119$ corresponding to the block B_1 = "Cry" is presented in Table 10. Initially, m_1 is transformed to $m_1 = 164191190$. Then, iterating over various combinations of the first and last digits of it, in each iteration, the slope S_1 is computed to define and solve Equation (9) until a solution $x_{m_1} =$ is found. Subsequently, $y_{m_1} = S_1x_{m_1} + m_1 \pmod p$ is computed. Upon receiving the cipher point $C_{m_1} = (1665341891, 2413987307)$, the receiver ($r$) can decrypt it to obtain the original mapping point $P_{m_1} = (2024477685, 3189489356)$ by performing the operation $C_{m_1} - kP_r$. Subsequently, the receiver calculates the slope S_1 using the formula $S_1 = (y_s - y_{m_1})(x_s - x_{m_1})^{-1}$, yielding the value 1029763873. This slope enables the receiver to reconstruct the original block B_1 = "Cry" by decoding the message $m_1 = y_s - S_1x_s$ while disregarding its first and last digits.

Table 10. Message mapping rounds of the block B_1 = "Cry" ($Counter = 2$).

Round	m_1	S_1	Equation (9) Has a Solution
1	143469980	2389772309	No, $x_{m_1} = None$
2	143469981	1029763873	Yes, $x_{m_1} = 2024477685$

7. Conclusions

We introduce a set of innovative contributions to the processes of message encoding, mapping, reversal, and decoding within the ElGamal ECC-based scheme. Our emphasis is on maintaining a delicate balance between the security and computational efficiency of the resulting cryptosystem. To address various attack models, the plaintext encoding incorporates a parameter, *Counter*, and a nested composition hash function of a secret key before the message mapping process. This introduces a non-one-to-one relationship between the plaintext and mapping points, enhancing security. Furthermore, the mapping process is fortified by utilizing a shared secret point negotiated through the ECDHP.

Our new cryptosystem underwent a comprehensive analysis to assess its performance and security aspects, including complexity, overlap, resistance against chosen-plaintext attacks (CPA), resilience against chosen-ciphertext attacks (CCA), and replay attacks. Furthermore, we investigated the sensitivity of the ciphertext to minor changes in the cryptosystem inputs, and we examined the randomness of the ciphertext from the eavesdropper's perspective. Notably, the design considerations were aimed at enhancing security while maintaining operational efficiency.

Furthermore, it is crucial to consider the arithmetic operations involved in the scalar multiplication of a point in the curve and solving Problem (7) during mapping to mitigate potential security risks, particularly those related to side-channel attacks. Therefore, we will intentionally postpone the detailed implementation of these aspects for future investigation. While our proposed method remains robust, addressing these areas in future research will help prevent vulnerabilities that may arise from improper implementations.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L., S.L. and Y.A.; software, Y.L.; validation, S.L., Y.A. and A.N.; formal analysis, Y.L., S.L. and Y.A.; investigation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L., S.L., Y.A. and A.N.; visualization, Y.L.; supervision, S.L., Y.A. and A.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ECC	elliptic curve cryptography
EC	elliptic curve
NIST	National Institute of Standards and Technology
ECDHP	elliptic curve Diffie–Hellman protocol
ECDSA	elliptic curve digital signature algorithm
ECIES	elliptic curve integrated encryption scheme
CPA	chosen-plaintext attack
CCA	chosen-ciphertext attack
RA	replay attack
ECDLP	elliptic discrete logarithm problem
AES	advanced encryption standard

IND-CPA	indistinguishability under chosen-plaintext attack
IND-CCA	indistinguishability under chosen-ciphertext attack
MitM	man-in-the-middle
STS	station to station
KPA	known-plaintext attack
COA	ciphertext-only attack

References

1. Mid-Year Update to the 2023 SonicWall Cyber Threat Report | Threat Intelligence. Available online: https://www.sonicwall.com/2023-mid-year-cyber-threat-report/ (accessed on 1 May 2024).
2. IBM. Cost of a Data Breach 2023. Available online: https://www.ibm.com/reports/data-breach (accessed on 1 May 2024).
3. Gura, N.; Patel, A.; Wander, A.; Eberle, H.; Shantz, S.C. Comparing elliptic curve cryptography and RSA on 8-Bit CPUs. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3156, pp. 119–132. [CrossRef]
4. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
5. *1363-2000*; IEEE Standard Specifications for Public-Key Cryptography. IEEE: Piscataway, NJ, USA, 2000.
6. King, B. Mapping an Arbitrary Message to an Elliptic Curve when Defined over GF (2^n). *Int. J. Netw. Secur.* **2009**, *8*, 169–176.
7. Gayoso, V.; Hernandez, L.; Sanchez, C. A survey of the elliptic curve integrated encryption scheme. *J. Comput. Sci. Eng.* **2010**, *2*, 7–13.
8. Almajed, H.; Almogren, A. A secure and efficient ECC-based scheme for edge computing and internet of things. *Sensors* **2020**, *20*, 6158. [CrossRef] [PubMed]
9. Barker, E.; Chen, L.; Keller, S.; Roginsky, A.; Vassilev, A.; Davis, R. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017.
10. Ullah, S.; Zheng, J.; Din, N.; Hussain, M.T.; Ullah, F.; Yousaf, M. Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey. *Comput. Sci. Rev.* **2023**, *47*, 100530. [CrossRef]
11. Rukhin, A.; Soto, J.; Nechvatal, J.; Miles, S.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. SP800-22: A statistical test suite for random and pseudorandom number generators for cryptographic applications. In *National Institute of Standards and Technology*; National Institute of Standards and Technology: Gaithersburg, MA, USA, 2010; Volume 800, Issue April. [CrossRef]
12. Hankerson, D.; Menezes, A.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer Professional Computing; Springer: New York, NY, USA, 2006.
13. Sengupta, A.; Ray, U.K. Message mapping and reverse mapping in elliptic curve cryptosystem. *Secur. Commun. Netw.* **2016**, *9*, 5363–5375. [CrossRef]
14. Padma, B.; Chandravathi, D.; Roja, P.P. Encoding And Decoding of a Message in the Implementation of Elliptic Curve Cryptography using Koblitz's Method. *(IJCSE) Int. J. Comput. Sci. Eng.* **2010**, *2*, 1904–1907.
15. Srinath, M.S.; Chandrasekaran, V. Elliptic Curve Cryptography using Mirrored Elliptic Curves over Prime Fields. In Proceedings of the International Conference on Information and Knowledge Engineering, Las Vegas, NV, USA, 12–15 July 2010; pp. 271–277.
16. Koblitz, N. *A Course in Number Theory and Cryptography*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1994; Volume 114.
17. Boneh, D.; Franklin, M. Identity-based encryption from the weil pairing. *SIAM J. Comput.* **2003**, *32*, 2003. [CrossRef]
18. Icart, T. How to Hash into Elliptic Curves. In *Advances in Cryptology, Proceedings of the CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009*; Proceedings; Halevi, S., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5677, pp. 303–316. [CrossRef]
19. Vigila, S.M.C.; Muneeswaran, K. Implementation of text based cryptosystem using elliptic curve cryptography. In Proceedings of the 2009 First International Conference on Advanced Computing, Chennai, India, 13–15 December 2009. [CrossRef]
20. Kumar, D.S. Encryption of Data Using Elliptic Curve over Finite Fields. *Int. J. Distrib. Parallel Syst.* **2012**, *3*, 301–308. [CrossRef]
21. Reyad, O. Text Message Encoding Based on Elliptic Curve Cryptography and a Mapping Methodology. *Inf. Sci. Lett.* **2018**, *7*, 7–11. [CrossRef]
22. Amounas, F.; El Kinani, E. Fast mapping method based on matrix approach for elliptic curve cryptography. *Int. J. Inf. Netw. Secur. (IJINS)* **2012**, *1*, 54–59.
23. Kamalakannan, V.; Tamilselvan, S. Security Enhancement of Text Message Based on Matrix Approach Using Elliptical Curve Cryptosystem. *Procedia Mater. Sci.* **2015**, *10*, 489–496. [CrossRef]
24. Almajed, H.N.; Almogren, A.S. SE-Enc: A Secure and Efficient Encoding Scheme Using Elliptic Curve Cryptography. *IEEE Access* **2019**, *7*, 175865–175878. [CrossRef]
25. Reegan, A.S.; Kabila, V. Highly Secured Cluster Based WSN Using Novel FCM and Enhanced ECC-ElGamal Encryption in IoT. *Wirel. Pers. Commun.* **2021**, *118*, 1313–1329. [CrossRef]
26. Moosavi, S.R.; Izadifar, A. End-to-End Security Scheme for E-Health Systems Using DNA-Based ECC. In *Silicon Valley Cybersecurity Conference*; Springer: Berlin/Heidelberg, Germany, 2022. [CrossRef]

27. Tiwari, H.D.; Kim, J.H. Novel Method for DNA-Based Elliptic Curve Cryptography for IoT Devices. *ETRI J.* **2018**, *40*, 396–409. [CrossRef]
28. Das, S.; Namasudra, S. A Novel Hybrid Encryption Method to Secure Healthcare Data in IoT-enabled Healthcare Infrastructure. *Comput. Electr. Eng.* **2022**, *101*, 107991. [CrossRef]
29. Lahraoui, Y.; Amal, Y.; Lazaar, S. Definition and Implementation of an Elliptic Curve Cryptosystem using a New Message Mapping Scheme. In Proceedings of the 3rd International Conference on Networking, Information Systems & Security, San Jose, CA, USA, 9–12 March 2020. [CrossRef]
30. Brier, E.; Coron, J.S.; Icart, T.; Madore, D.; Randriam, H.; Tibouchi, M. Efficient indifferentiable hashing into ordinary elliptic curves. In *Advances in Cryptology—CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 237–254.
31. Berlekamp, E.R. Algebraic Coding Theory. In *Negacyclic Codes*; World Scientific: Singapore, 1968; pp. 207–217.
32. Burgess, D.A. The distribution of quadratic residues and non-residues. *Mathematika* **1957**, *4*, 106–112. [CrossRef]
33. Webster, A.F.; Tavares, S.E. On the Design of S-Boxes. In *Advances in Cryptology—CRYPTO'85 Proceedings*; Springer: Berlin/Heidelberg, Germany, 1986; Volume 218. [CrossRef]
34. Seyedzadeh, S.M.; Norouzi, B.; Mosavi, M.R.; Mirzakuchaki, S. A novel color image encryption algorithm based on spatial permutation and quantum chaotic map. *Nonlinear Dyn.* **2015**, *81*, 511–529. [CrossRef]
35. Barker, E.; Barker, W.C. *Recommendation for Key Management: Part 2—Best Practices for Key Management Organizations*; National Institute of Standards and Technology NIST: Gaithersburg, MA, USA, 2019. Available online: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf (accessed on 1 May 2024).
36. Chen, L.; Moody, D.; Regenscheid, A.; Robinson, A.; Randall, K. Recommendations for Discrete Logarithm-Based Cryptography. 2023. Available online: https://csrc.nist.gov/pubs/sp/800/186/final (accessed on 1 May 2024).
37. Dolev, D.; Dwork, C.; Naor, M. Non-malleable cryptography. In Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, New Orleans, LA, USA, 6–8 May 1991; pp. 542–552.
38. Halperin, D.; Clark, S.S.; Fu, K.; Heydt-Benjamin, T.S.; Defend, B.; Kohno, T.; Ransford, B.; Morgan, W.; Maisel, W.H. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In Proceedings of the 2008 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 18–22 May 2008. [CrossRef]
39. Zhou, Y.; Feng, D. *Side-Channel Attacks: Ten Years after Its Publication and the Impacts on Cryptographic Module Security Testing*; IACR Eprint Archive: Lyon, France, 2005. Available online: http://eprint.iacr.org/2005/388 (accessed on 1 May 2024).
40. Socha, P.; Miškovský, V.; Novotný, M. A Comprehensive Survey on the Non-Invasive Passive Side-Channel Analysis. *Sensors* **2022**, *22*, 8096. [CrossRef]
41. Vanhoef, M.; Ronen, E. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. Cryptology ePrint Archive, Paper 2019/383; IACR Eprint Archive: Lyon, France, 2019. Available online: https://eprint.iacr.org/2019/383 (accessed on 1 May 2024).
42. Gabsi, S.; Kortli, Y.; Beroulle, V.; Kieffer, Y.; Hamdi, B. Proposal of a lightweight differential power analysis countermeasure method on elliptic curves for low-cost devices. *Multimed. Tools Appl.* **2024**. [CrossRef]
43. Pavithran, D.; Shaalan, K. Towards Creating Public Key Authentication for IoT Blockchain. In Proceedings of the ITT 2019—Information Technology Trends: Emerging Technologies Blockchain and IoT, Ras Al Khaimah, United Arab Emirates, 20–21 November 2019. [CrossRef]
44. Kumagai, K.; Kakei, S.; Shiraishi, Y.; Saito, S. Distributed Public Key Certificate-Issuing Infrastructure for Consortium Certificate Authority Using Distributed Ledger Technology. *Secur. Commun. Netw.* **2023**. [CrossRef]
45. Wendlandt, D.; Andersen, D.G.; Perrig, A. Perspectives: Improving SSH-Style Host Authentication with Multi-Path Probing. In Proceedings of the 2008 USENIX Annual Technical Conference, USENIX 2008, San Diego, CA, USA, 14–19 June 2009; USENIX: Boston, MA, USA, 2008. Available online: https://www.usenix.org/legacy/event/usenix08/tech/full_papers/wendlandt/wendlandt_html/ (accessed on 1 May 2024)
46. Gultekin, E.; Aktas, M.S. Real-Time Anomaly Detection Business Process for Industrial Equipment Using Internet of Things and Unsupervised Machine Learning Algorithms. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 14108 LNCS*; Springer: Berlin/Heidelberg, Germany, 2023. [CrossRef]
47. Khudri, W. Sutanto, Implementation of Elgamal Elliptic Curve Cryptography Using Matlab. In Proceedings of the Conference on Instrumentation, Communication and Information Technology (ICICI), Bandung, Indonesia, 3–5 August 2005. Available online: https://wan.khudri.com/my_files/icici2005/paper_ICICI_2005.pdf (accessed on 1 May 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.