sensors

MDPI

Article

A Secure and Efficient ECC-Based Scheme for Edge Computing and Internet of Things

Hisham AlMajed † and Ahmad AlMogren *,†

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia; 438105079@student.ksu.edu.sa
* Correspondence: ahalmogren@ksu.edu.sa
† These authors contributed equally to this work.

Received: 20 August 2020; Accepted: 26 October 2020; Published: 29 October 2020

Abstract: Recent growth in the Internet of Things (IoT) has raised security concerns over the confidentiality of data exchanged between IoT devices and the edge. Many IoT systems adopt asymmetric cryptography to secure their data and communications. A drawback of asymmetric cryptography is the sizeable computation and space requirements. However, elliptic curve cryptography (ECC) is widely used in constrained environments for asymmetric cryptography due its superiority in generating a powerful encryption mechanism with small key sizes. ECC increases device performance and lowers power consumption, meaning it is suitable for diverse applications ranging from the IoT to wireless sensor network (WSN) devices. To ensure the confidentiality and security of data and communications, it is necessary to implement ECC robustly. A special area of focus in this regard is the mapping phase. This study's objective was to propose a tested and trusted scheme that offers authenticated encryption (AE) via enhancing the mapping phase of a plain text to an elliptic curve to resist several encryption attacks such as Chosen Plaintext Attack (CPA) and Chosen Ciphertext Attack (CCA). The proposed scheme also undertakes evaluation and analysis related to security requirements for specific encryption attributes. Finally, results from a comparison of the proposed scheme and other schemes are presented, evaluating each one's security characteristics and performance measurements. Our scheme is efficient in a way that makes so suitable to the IoT, and in particular to the Industrial IoT and the new Urbanization where the demands for services are huge.

Keywords: authenticated encryption; asymmetric cryptography; chosen cipher text attack; chosen plain text attack; edge computing; elliptic curve cryptography; encryption; internet of things; industrial internet of things

1. Introduction

The continuous growth of industrialization and urbanization in recent years has led to the estimate that by 2025 there will be 21.5 billion actively connected Internet of Things (IoT) devices worldwide [1] as depicted in Figure 1. This remarkable growth of IoT shows that edge computing is increasingly used in today's society [2]. In particular, the constrained properties of IoT devices, which include low performance in terms of computational resources and storage space, led to the adoption of edge computing [3]. For these reasons, the need to maintain data confidentiality and integrity has increased, which has caused an explosion of interest in cryptography schemes [4]. There are two types of encryption scheme, which are based on the nature of the key that will be used to encrypt and decrypt the data: which known as, symmetric cryptography and asymmetric cryptography [5]. The first type uses a one key for encryption and decryption, and it is useful if the two parties know the key before exchanging data. However, if the sender and the recipient cannot agree on a secure secret key exchange, then asymmetric cryptography is needed. In asymmetric cryptography, elliptic curve cryptography (ECC) is known for its superiority in producing a powerful encryption mechanism with small key sizes [6]. Therefore, ECC plays a valuable role in securing communications in constrained and resource-limited environments, including the IoT and wireless sensor networks (WSNs) [7].

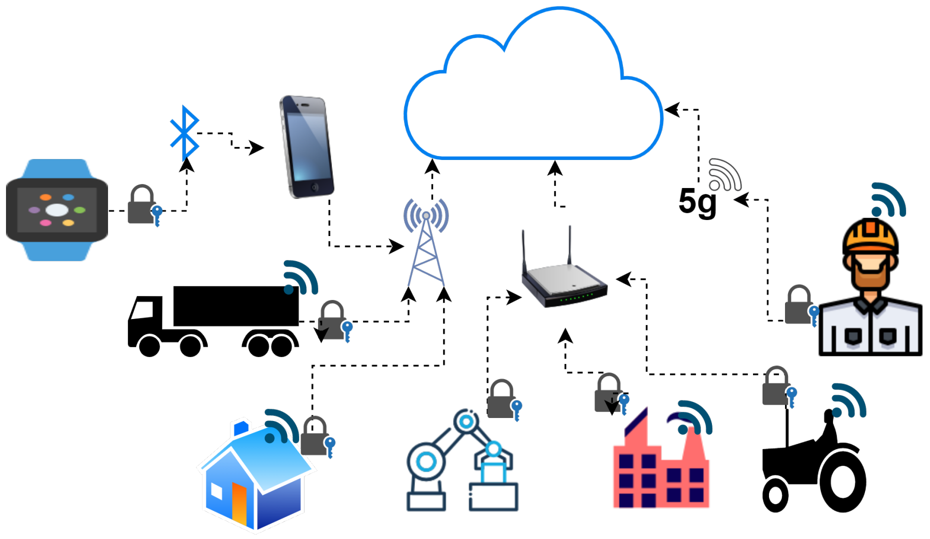


Figure 1. New technologies raised by the growth of industrialization and urbanization.

1.1. Encryption Attacks on Asymmetric Cryptography

Information security consists of three aspects: confidentiality, integrity, and availability (CIA) [8–10]. Confidentiality and integrity are assured via cryptography schemes. One such scheme is asymmetric cryptography, which uses two key types: a public key and a private key [11]. The public key is used by the sender to encrypt the plain text, which gives it its name. This key is publicly available for use by anyone. By contrast, the recipient uses the private key to decrypt the received cipher text, which means that it must be known only by the recipient.

Asymmetric cryptography settles the dilemma of securing a shared key between two parties. At the same time, however, asymmetric cryptography suffers from a drawback in the key sizes used to encrypt and decrypt messages. Larger key sizes correspond to larger computation overheads for encrypting and decrypting plain text. However, ECC provides the same level of security as the Rivest–Shamir–Adleman (RSA) algorithm with short keys [12]. Therefore, ECC has emerged as the preferred approach for solving key size issues and for maintaining performance in constrained environments [13].

Many known attacks weaken existing cryptography schemes. They exploit vulnerabilities in the encryption process. For instance, the Known Plain Text Attack (KPA), Chosen Plain Text Attack (CPA), Cipher Text Only Attack (COA), and several types of Chosen Cipher Text Attack (CCA) have been identified. The KPA occurs when an attacker obtains a plain text and its corresponding cipher text. Specifically, the attacker attempts to obtain the encryption key [14]. The CPA occurs when an attacker selects random plain texts and requests the corresponding cipher texts for each text. Thus, the attacker aims to reduce the security of the scheme by analyzing both the plain text and cipher text [15]. In COA, the assumption is made that an attacker only has access to a set of cipher texts, where they can extract the secret key and/or the plain text [16]. The final attack, the CCA, involves the attacker gaining information by obtaining a sample of decrypted cipher texts of his/her choice [17].

In this research, the following attacks are used for the security analysis phase, denoted as [18]:

- IND-CPA: Indistinguishable under chosen plain text attack
- IND-CCA: Indistinguishable under chosen cipher text attack

1.2. Elliptic Curve Cryptography

The first curve used in elliptic curve cryptography (ECC) was introduced by Koblitz in 1987 [19]. ECC is widely used for devices in constrained environments, including the IoT and WSN devices. This stems from ECC's value in affording the same level of hardness in terms of encryption as other asymmetric cryptography schemes, with significantly smaller key sizes and a lower computation overhead. As an illustration, the complexity of encryption in the RSA algorithm with a 1024-bit key is the same as the ECC algorithm with a 160-bit key. This noticeable difference in key sizes reduces the requirement for computation, and it lowers the storage needed to perform encryption. As a result, ECC supports low computation device capabilities, enabling them to perform more effectively [20]. ECC applications vary in the ways keys are exchanged between senders and recipients, as well as the approach used to secure communications between the two parties. In addition, ECC maintains message integrity by signing plain texts to prevent forgery.

The elliptic curve group operations are denoted as "+" for the addition of two points. For instance, let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, Therefore, the addition of $P + Q$ can be expressed as $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$. In some cases, where $P = Q$, the group operation is denoted as multiplication $\alpha \times P$ on elliptic curve over $\mathbb{Z}_p$, $p > 3$ that satisfy Equation (1), such that $a, b \in \mathbb{Z}_p$ and $4 \times a^3 + 27 \times b^2 \neq 0 \ mod \ p$ where α is an integer value. For instance, $P + P = (x_1, y_1) + (x_1, y_1) = 2P)$, similarly, $3P$ is equal to $(x_1, y_1) + (x_1, y_1) + (x_1, y_1)$, and so on. It is important to note the the EC must be nonsingular curves (i.e., have no multiples roots).

$$y^2 \equiv x^3 + a \times x + b \ mod \ p \tag{1}$$

The main operation in elliptic curves is the group multiplication operation [21–24]. For dP, this refers to d times of the addition of point P, which results in a new point (x_j, y_j). The private key is a large integer d, and the value of the multiplication operation (x_j, y_j) is known as the public key. The hardness of ECC is caused by the hardness of the mathematical problem, which states that, by knowing the public key point (x_j, y_j) and the starting point P, it is not possible to compute d in polynomial time [25]. This hardness is known in the literature as the elliptic curve discrete logarithm problem (ECDLP).

Several phases are involved in ECC to secure communications and encrypt transmitted data [26–28]. These phases are used together and/or separately, and they are listed as follows:

- Initializing system parameters
- Converting text values to numerical values
- Mapping numerical values to the elliptic curve
- Encrypting mapped points
- Hashing the message (for signing)

Similarly, the decryption phases are listed below:

- Verifying integrity of received message (signature verification)
- Decrypting cipher text
- Decoding mapped points to numerical values
- Converting numerical values to text values to represent plain text

The main computation process involved in the first phase is the calculation of the public keys derived by the multiplication of d (i.e., the private key) and G (i.e., the base point) [29–32]. The following phase involves the conversion of the plain text into numerical values, which is required because ECC depends on the use of numbers [33]. Therefore, the text must be securely converted in order to resist encryption attacks. Similarly, the third phase involves mapping the encoded value x_i to the generated elliptic curve to find corresponding value of y_i such that $(x_i, y_i) \in E_p(a, b)$. If the mapping process fails in the first round, then x_i is incremented by 1 until mapping is successful [34–37]. In the encryption phase, the cipher text is combined with the mapped point and secret key point $(x_{mapped}, y_{mapped}) + (x_{key}, y_{key})$. The fifth phase maintains the integrity of the cipher text and ensures the sender's nonrepudiation [38–40]. This is achieved by signing the cipher text using the sender's signature, and it depends on the following steps [41–43]:

- Compute r, s.t. r is the $x_R \; mod \; p$ of $(x_R, y_R) = k * G$, where k is a random number and G is a base point
- Compute $e = HASH(ciphertext)$ and obtain $z = leftmostp \; bits \; of \; e$
- Compute s, where $s \equiv (d + z * r)k^{-1} \; mod \; p$, and where d is the sender's private key
- The sent message is $(ciphertext, (r, s))$

Having provided an introductory overview, the rest of this paper is organized as follows: a literature review of other schemes is provided in Section 2; in Section 3, the details of the proposed scheme are described; in Section 4, security analysis and performance evaluations are presented in detail; and finally, concluding remarks and future research are discussed in Section 5.

2. Related Works

Elliptic curve cryptography (ECC) is frequently used to reduce the computational overhead caused by the limited capabilities of devices in constrained environments. Many schemes use ECC to secure communications between two parties by safeguarding the shared key exchange process. In particular, the elliptic curve integrated encryption scheme (ECIES) first employed the asymmetric approach by generating the shared key between two parties using ECC, after which the plain text was encrypted using the symmetric approach under the AES scheme [44–47]. On the other hand, many of these schemes failed to provide detail on how ECC was used to secure the plain text and/or how they were encoded into numerical values for use in ECC's mapping phase.

Various proposed systems have enhanced key elements of the encryption process in ECC, but gaps have been identified in the literature. For instance, schemes proposed in [48–50] employ ECC without providing sufficient detail about how the plain text was encoded and mapped onto an elliptic curve. Therefore, enhancing these schemes tended to focus on performance rather than security. Similarly, [51–53] provided efficient algorithms for scalar multiplication in ECC that speed up the multiplication process. However, many schemes focus on securing ECC, and they offer more insights into the approach used to encode plain text and map it onto an elliptic curve. The rest of this literature review focuses on these schemes and, in particular, on the question of how the plain texts are encoded into numerical values, in addition, how these values are mapped onto the elliptic curve.

Several schemes proposed ways to secure plain text by encoding the characters to numerical values, thus giving them the ability to be mapped onto the elliptic curve. Many approaches use these schemes to encode the plain text. For instance, using the ASCII table, each character is converted into its decimal number [26,54–56]. In this case, the plain text "Hello" would be encoded to become "72" "101" "108" "108" "111". These values are then mapped directly onto the elliptic curve as cipher text. However, this approach falls under the chosen plain text attack (CPA). This is because the attacker has the power to decrypt the chosen cipher text (in this case, a commonly used scheme). For this reason, other schemes manipulate the ASCII table by multiplying it by a secure number that is agreed on by both parties [57].

Critically, the issue of sharing the secure number is similar to the challenge of agreeing on the sharing of a secure key between the two parties. Such a scheme could also fall under the CPA and CCA. Similarly, other schemes are based on different encoding approaches that rely on matrix-based methods to conceal the matching table [58]. Specifically, these schemes use secret mapping tables that are unknown to anyone except the recipient, and which are used to encode and decode the plain text in a secure manner. While this may be true, there are two weakness associated with this approach: first, the dilemma of how to secure the delivery of the matrix table to the recipient in a secure way, thereby preventing CPA; and second, it is known that if the matrix-table is assumed to be securely delivered, then the cipher text falls under the CCA. This is because the same encrypted characters are repeated for the same plain characters in each encryption process.

The third approach to overcome flaws in ECC involves Block Chaining operations. The first step is to divide the plain text into a set of fixed-size blocks. In turn, the first block of the plain text is XORed with an initial vector InV [59]. Following this, the result of the first XORed value is used for the second XOR operation with a second block, and the process is repeated for all plain text blocks. This is a good approach, but it is vulnerable to CPA and CCA when the plain text is divided into a set of blocks and all blocks are same (e.g., the plain text is a repeated character). As a result, the second XORing process results in the value of the InV as following $InV \oplus B_1 \oplus B_2 = InV$. This is because $B_1 = B_2$. Therefore, the result of the XORing operation becomes $B_1', InV, B_1', InV, ...,$ where $B_1' = B_1 \oplus InV$. Additionally, this scheme lacks AE property, which increases vulnerability to cipher text tampering attacks and nonrepudiation issues.

Barman et al. [60] proposed an encryption method to secure IoT communications informed by DNA-based ECC. For each set of characters in the plain text, a DNA genome sequence is mapped to it. There are many genome sequences publicly available, and so randomizing the selection of genome sequences is part of the encryption process for the plain text. Furthermore, the decryption process should use the same DNA genome sequences. Therefore, both parties, the sender and the recipient, must use the same sequences before encrypting and decrypting the plain text. For this reason, the DNA genome sequences should be securely used only by the sender and recipient. If an attacker discovers the sequence, the scheme will be vulnerable to encryption attacks such as CPA. Additionally, even if the sequence is delivered securely, the cipher text will be the same for each repeated plain text encryption process. Resultantly, the scheme is also vulnerable to other encryption attacks, including CCA.

Duarah et al. [61] introduced Securing IoT Using Machine Learning and ECC. In their scheme, the authors first classified the data set to enhance the transmitted data, where the accurate data were transmitted only to reduce computation efforts. If the data were clean, then they were moved into a second stage, namely encryption via ECC. However, if the data is malicious then it is discarded to save the encryption computation efforts. In the encryption phase, the authors wrote the key generation algorithms that use the ECC scalar multiplication operation $\delta = (P) * (d)$ s.t. P is a point in the elliptic curve, and d is the private random integer. In their encryption algorithm, the authors defined how the shared key is constructed in order to encrypt the plain text by the addition operation between the plain text and the shared key. Although the scheme produced a new strategy for performance enhancement in the IoT environment by limiting data transmission only to clean data, the security analysis indicates that it is vulnerable to CPA when the same data are encrypted and sent using the same scheme.

Joglekar et al. [62] proposed Lightweight ECC for Data Integrity and User Authentication in Smart Transportation IoT System. In their scheme, the authors use One Time Password (OTP) to exchange the shared key securely to prevent the man in the middle attack (MITM). The 4-digit OTP is encrypted using ECC and transmitted to the recipient to complete the registration process. The shared key between the two parties is constructed as follows: $S = d_A Q_B = d_B Q_A$, where d_A, d_b are private keys for Sender A and Recipient B, respectively. Similarly, $Q_B = d_B G$ and $Q_A = d_A G$ are the public keys for Sender A and Recipient B. Using OTP to prevent the MITM is a good approach, but the authors neglected to state whether there is an assumption about the 4-digit OTP. It is relevant that the brute force attack needs $10^4 = 10,000$ possible choices to break the OTP, which can be completed in several minutes. Furthermore, the authors did not state how to encrypt the OTP or how to map it onto an elliptic curve.

Finally, Das & Giri [55] proposed two encoding algorithms, which generate sets of numerical values via the sum of weight n with base b $IntegerDigits[n, b] - 1$. The first encoding algorithm is used when the value of the base b is a dynamic integer. In this case, the highest accepted value is $65,536$ (i.e., the highest value of ASCII table). In addition, n is the size of the prime field, where the authors suggest the use of a 192-bit key. Resultantly, the set of groups that can be combined based on their method is $IntegerDigits[192bit, 65, 536] - 1 = 11$. The base b can be reduced below $65, 536$, which increases the set of groups more than 11 based on b. However, the reduction of b should also contribute to a reduction in ASCII table mapping, and the authors did not provide a safe reduction mechanism. When b is not dynamic in the second algorithm, the suggested set of the combing group is equal to $\frac{Number\ of\ p\ digits}{IntegerDigits[n,b]-1}$. Based on the scheme described by the authors, it is $\frac{58-1}{11} = 6$, and the number of groups based on both algorithms is small. Thus, the computation overhead will increase compared to other schemes. Notably, the encoding and mapping phases did not manipulate the plain text characters after using the ASCII table values, which heightened the scheme's vulnerability to CPA.

All schemes in the literature are vulnerable to CPA and/or CCA, such that the encryption of the same plain texts always produce same cipher texts. Besides, these schemes use the appending method in the mapping phase which results in increasing the computation overhead. Although, some schemes that use the probability method to map points to EC failed to justify the chosen value of k. Moreover, many of the existing schemes fail to offer secure AE scheme, which means they are vulnerable to tampering attacks. An attacker can modify the transmitted cipher text without detection by the receiver. The cipher text offers confidentiality only; it does not offer integrity by itself. Therefore, to prevent tampering, to maintain the integrity of the cipher text, to ensure that the transmitted message is secured against encryption attacks, and to enhance the mapping phase performance, the proposed scheme ensures the AE property, resists CPA and CCA, and enhances the mapping phase performance.

3. The Proposed Scheme

The proposed scheme contains nine phases: initializing system parameters; converting the plain text message into numerical values (encoding); finding the mapping points on the elliptic curve; encrypting the mapped points; signing the aggregated mapped points as cipher text; and finally, undertaking the reverse of the previous phases by verifying and decrypting the received cipher text, decoding the mapped points, and converting the mapped points into plain text.

This study's main contribution is to offer a secure and efficient encryption scheme in the form of ECC. It facilitates secure communication by creating a shared key for a group of parties, which they can use to secure their shared messages. Notably, shared key creation is undertaken in the first phase, and many recent studies neglect to highlight the importance of having a shared key between parties for shared message encryption. In addition, this study provides a secure and enhanced method to encode and map plaintext to EC. Therefore, this study focuses on the three phases that start the ECC because it constitutes a major feature of any system for securing group communication (e.g., IoT environments). Figure 2 shows the nine phases of the proposed scheme.

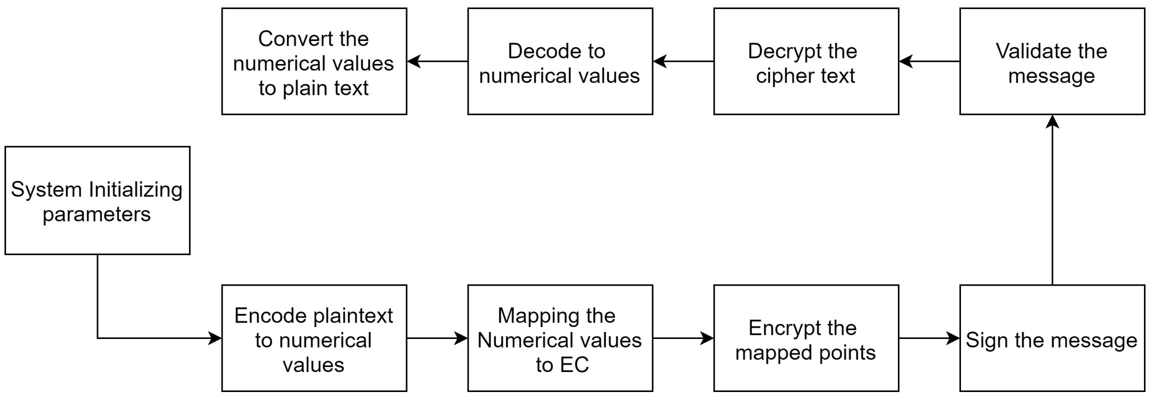

Convert the numerical values to plain text
Decode to numerical values
Decrypt the cipher text
Validate the message
System Initializing parameters
Encode plaintext to numerical values
Mapping the Numerical values to EC
Encrypt the mapped points
Sign the message

Figure 2. High-level overview of proposed scheme.

3.1. Generating System Parameters

The aim of this phase is to generate the parameters needed to secure communication between all parties by setting up public and private keys. This phase contributes to the creation of the shared group key gk_{sh}, which is used to encrypt the message shared between group members. The notations that the system generates, which are used for each session in the proposed scheme, are illustrated in Table 1.

Table 1. Notation relevant to the proposed scheme.

Notation	Description
id_{edge}	Edge identification number
id_{ni}	Node identification number
d_{edge}	Edge private key
d_{ni}	Node private key
G	EC base point
PU_{edge}	Edge public key = $d_b * G$
PU_{ni}	Node public key = $d_{ni} * G$
p	Large prime number (192-bit)
a, b	EC coefficients, s.t. $4a^3 + 27b^2 \ mod \ p \neq 0$
$y^2 \equiv x^3 + ax + b \ mod \ p$	EC map points equation
H_1	Hash function used only by edge
$nList$	Node list containing references and $H_1(id_{ni})$
$HASH$	Signing message C_M hash function
k_{sh}	Shared group key
gk_{sh}	Shared group point ($k_{sh} \times G$)
InV	Random initial vector (192-bit)
PRK	Private random key (192-bit)
k	Random integer chosen from $[1, p-1]$
C_M	Cipher text (all encrypted points)
$\oplus$	XOR used in mapping phase to secure mapped points
$+$	Addition operation used in ECC to encrypt mapped points with gk_{sh}

It is important for the edge to create and maintain the shared group key k_{sh}. This enables the parties to decrypt cipher text effectively. Therefore, *edge* creates the initial k_{sh} using its $H_1(id_{edge})$ and PRK. The key generation process is illustrated in Algorithm 1.

Algorithm 1: Edge algorithm for generating initial shared group key k_{sh}.

```
Input: $id_{edge}$; $H_1$; $PRK$
Output: $k_{sh}$
1 $k_{sh} = H_1(id_{edge}) \oplus PRK$;
2 $Initial\ shared\ group\ key \leftarrow k_{sh}$
```

In a similar way, for every new node that joins the group, k_{sh} is updated by the following Equation (2).

$$k_{sh} = H_1(id_{ni}) \oplus k_{sh} \tag{2}$$

To maintain the forward and backward secrecy, it is important to ensure that all new nodes that join cannot decrypt the cipher text sent before it joined the group using k_{sh}. Similarly, any nodes that leave the group cannot decrypt the cipher text sent after their departure. To achieve this, the proposed scheme allows each node to perform certain operations to maintain k_{sh}.

For a new node to join the group, the edge sends the joined node hashed ID $H_1(id_{ni})$ to all current nodes. In turn, each node adds the hashed ID to its node list $nList$, and it simultaneously updates k_{sh} using Equation (2). Similarly, the edge updates its k_{sh} using the same equation. In turn, the edge sends k_{sh} to the newly joined node, along with a list of all hashed IDs of existing node (with the exception of the hashed ID of the new node). Figure 3 illustrates this process, and Algorithm 2 describes the steps required to perform the joining process.

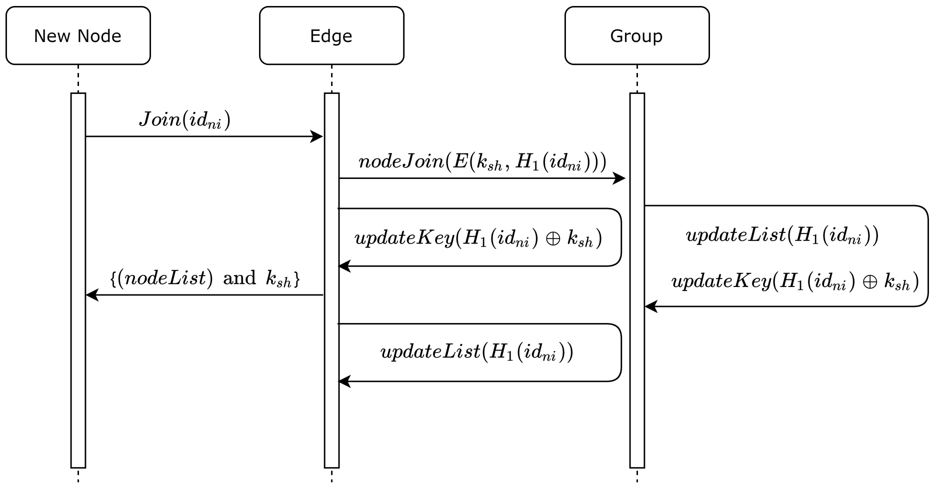

New Node
Edge
Group
$Join(id_{ni})$
$nodeJoin(E(k_{sh}, H_1(id_{ni})))$
$updateKey(H_1(id_{ni}) \oplus k_{sh})$
$updateList(H_1(id_{ni}))$
$updateKey(H_1(id_{ni}) \oplus k_{sh})$
$\{(nodeList)$ and $k_{sh}\}$
$updateList(H_1(id_{ni}))$

Figure 3. Sequence diagram for new node joining the group.

Algorithm 2: Updating k_{sh}, gk_{sh}, and $nList$ for a newly joined node.

```
Input: $id_{ni}$; $H_1$; $k_{sh}$
Output: Updated $k_{sh}$ ,$gk_{sh}$ and $nList$
1 New node requests $join(id_{ni})$;
2 New node and Edge start MA process;
3 Edge broadcast $H_1(id_{ni})$ encrypted by current $gk_{sh}$;
4 Edge and current nodes update key $k_{sh} = H_1(id_{ni}) \oplus k_{sh}$;
5 Edge and current nodes update shared point $gk_{sh} = k_{sh} \times G$;
6 Edge sends $k_{sh}$ and $nList$ to new node;
7 Edge and current nodes update their $nList$
```

Similarly, when the current node leaves the group, the edge sends its reference ID to all currently joined nodes, and each node updates k_{sh} using Equation (2). At the same time, the corresponding hashed IDs are removed from the node list. Additionally, the edge updates k_{sh} using Equation (2), and it removes the corresponding hashed ID from its node list. Figure 4 presents a sequence diagram illustrating the process, and Algorithm 3 describes the steps required to perform the leaving process.

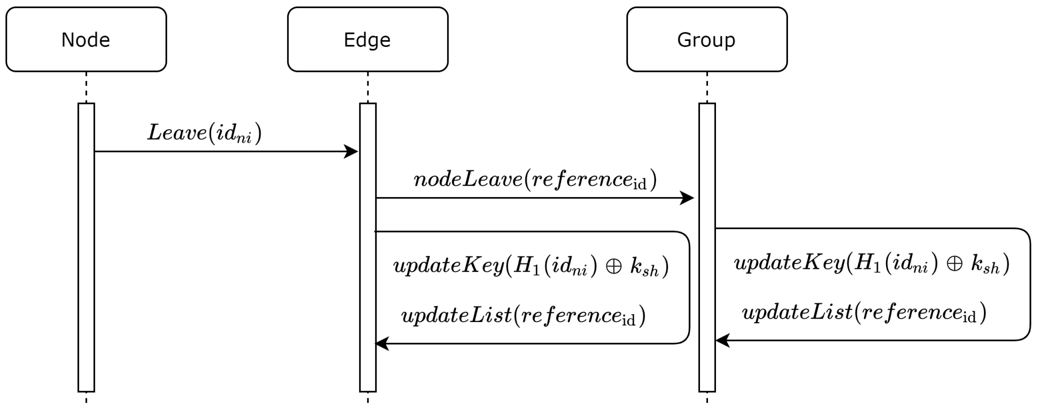

Node
Edge
Group
$Leave(id_{ni})$
$nodeLeave(reference_{id})$
$updateKey(H_1(id_{ni}) \oplus k_{sh})$
$updateList(reference_{id})$
$updateKey(H_1(id_{ni}) \oplus k_{sh})$
$updateList(reference_{id})$

Figure 4. Sequence diagram for node leaving the group.

Algorithm 3: Updating k_{sh}, gk_{sh}, and $nList$ for node leaving the group.

```
    Input: $id_{ni}$; $H_1$; $k_{sh}$
    Output: $Updated\ k_{sh}\ , gk_{sh}\ and\ nList$
1   Node sends $leave(id_{ni})$;
2   Edge broadcast $nodeLeave(Enc(gk_{sh}, reference_{id}))$;
3   Edge and current nodes update key $k_{sh} = H_1(id_{ni}) \oplus gk_{sh}$;
4   Edge and current nodes update shared point $gk_{sh} = k_{sh} \times G$;
5   Edge and current nodes update their $nList$
```

3.2. Encoding and Mapping the Plain Text

This study extended the encoding and mapping mechanism steps proposed in [63]. As described in related work, a security flaw exists in the study's encoding process. The main encryption flaw associated with the encoding and mapping phases in ECC stems from the fact that using the same encryption scheme produces the same ciphertext. As such, by analyzing the ciphertext, the adversary can gain important information about the plaintext. To illustrate this flaw, Figure 5 shows how the same letter is encoded with the same value every time. Thus, the corresponding ciphertext of the encoded value is transformed into the same ciphertext value each time, which allows the adversary to distinguish between the set of ciphertexts to extract the plaintext, as shown in Figure 6.

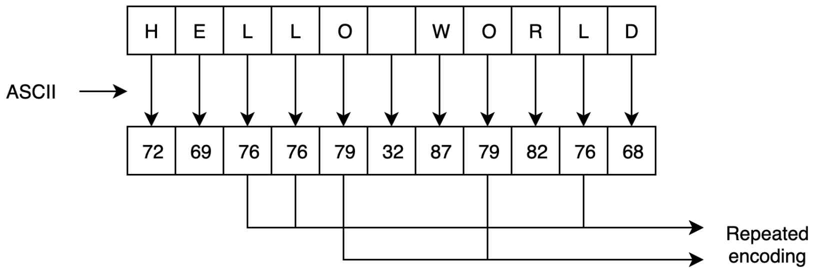


Figure 5. Repeated encoded characters using the ASCII table.

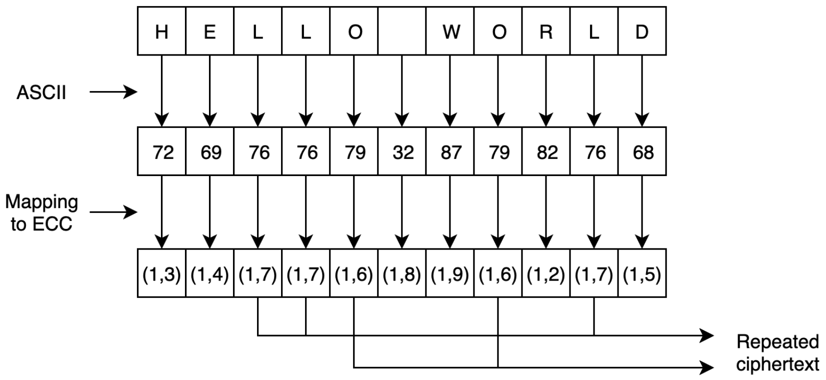


Figure 6. Repeated mapped points to the designated elliptic curve using the ASCII table.

Every scheme discussed in this study's literature review in Section 2 is vulnerable to an encryption flaw. Therefore, the study provides an improved encoding process to overcome this flaw by dividing the plain text into a set of blocks denoted by B. Then, the characters of each block B are counted, denoted by N, which can be calculated as follows:

$$N \leq \left\lfloor \frac{p-8}{8} \right\rfloor \tag{3}$$

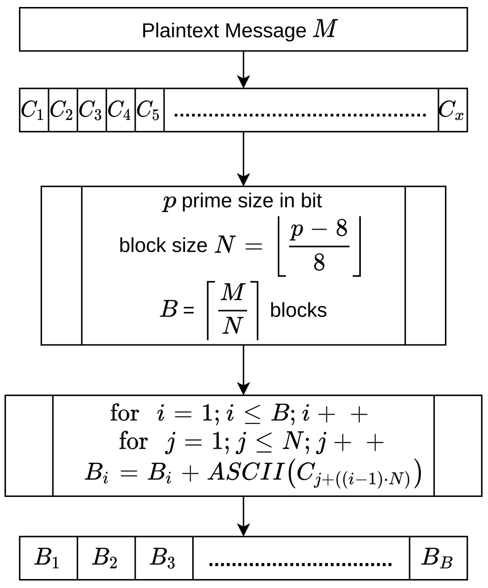
It is worth noting that 8 is subtracted from p to allow a maximum of 8 bits in terms of the padding bits used in the mapping phase. Therefore, in the proposed scheme, we can determine the value of N as follows:

$$N \leq \left\lfloor \frac{192-8}{8} \right\rfloor = 23 \tag{4}$$

In a similar way, the number of blocks B needed in each plain text, denoted by the number of characters M, is determined by the following equation:

$$B = \left\lceil \frac{M}{N} \right\rceil \tag{5}$$

The to divide the plaintext to this length is because every block B must be mapped to the 192-bit ECC (field size 192). As discussed previously, 8 bits in every block are reserved for the mapping phase, in which they serve as padding bits to secure the number of rounds required to find the mapping points for each block. Figure 7 illustrates the process needed to convert the plain text M into set of blocks.



Plaintext Message M
C_1 C_2 C_3 C_4 C_5 .... C_x
p prime size in bit
block size $N = \left\lfloor \frac{p-8}{8} \right\rfloor$
$B = \left\lceil \frac{M}{N} \right\rceil$ blocks
for $i = 1; i \leq B; i++$
for $j = 1; j \leq N; j++$
$B_i = B_i + ASCII(C_{j+((i-1)\cdot N)})$
B_1 B_2 B_3 .... B_B

Figure 7. Converting plain text into a set of blocks.

Algorithm 4 provides an overview of the steps involved in converting plain text into a set of blocks.

Algorithm 4: Converting a plain text into a set of blocks.

```
   Input: The plaintext M and p
   Output: Set of blocks
1  Sender : obtain the plaintext M;
2  Sender : calculate the size of each block;
3      $N \leftarrow \left\lfloor \frac{192-8}{8} \right\rfloor$;
4  Sender : calculate the number of blocks;
5      $B \leftarrow \left\lceil \frac{M}{N} \right\rceil$;
6  Sender : divide the palintext into B blocks of size N;
7     for $i = 1; i <= B; i++$;
8       for $j = 1; j <= N; j++$;
9       Sender : obtain the $B_i = B_i + ASCII(c_{j+((i-1)*N)})$;
10 Set of blocks ← the results
```

Following the conversion of M into a set of blocks, the next critical issue is to ensure that the blocks are secure against encryption attacks (e.g., CPA and CCA). Therefore, the Cipher Block Chaining (CBC) was used to make the blocks resistant to such attacks. Figure 8 shows the process of $\oplus$ the blocks to increase their security.

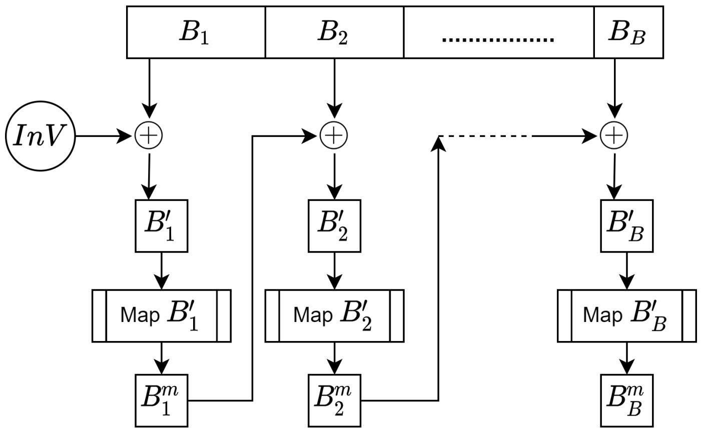


Figure 8. Securing blocks for resistance against encryption attacks using Cipher Block Chaining (CBC).

Algorithm 5 describes the process that secures the blocks.

Algorithm 5: Securing blocks for resistance against encryption attacks.

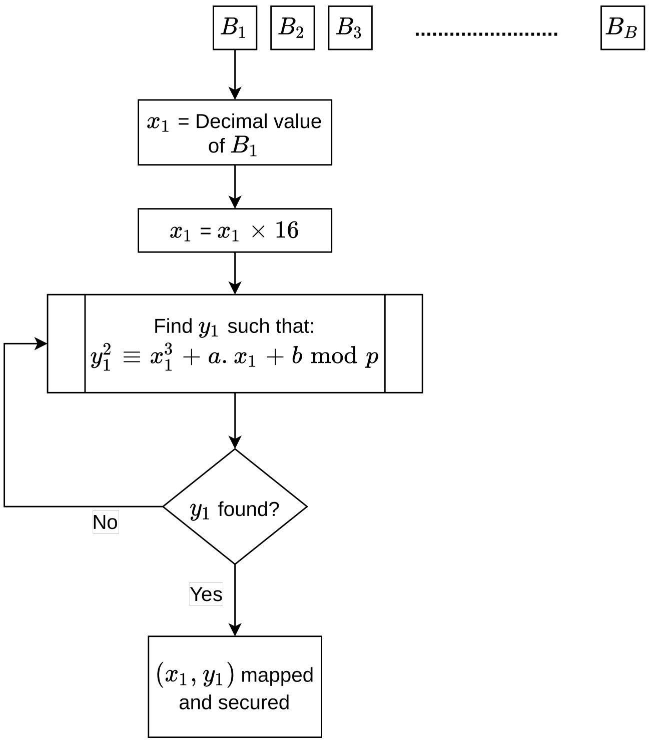
```
    Input: Blocks retrieved ($B_i$) from message and $InV$
    Output: Encoded blocks with $InV$
1 for $i = 0; i <$ no of blocks $B; i++$;
2     let $B_i{}' = B_i \oplus InV$;
3     let $B_i{}^m = map(B_i{}')$;
4     let $InV = B_i{}^m$;
5 Encoded message $\leftarrow$ the set of $B_i{}'$
```

Mapping a point to an elliptic curve means that (x_i, y_i) satisfies Equation (1). Therefore, it is important to find the value of y_i corresponding to x_i for each point. Every secured block is converted into a decimal value, and these values are then mapped to the elliptic curve generated in the first phase to find the corresponding value of y_i. Figure 9 illustrates the steps involved in mapping the secured blocks to an elliptic curve.

Algorithm 6 outlines the steps involved in mapping the secured blocks to an elliptic curve.

Algorithm 6: Mapping secured blocks to an elliptic curve.

```
    Input: Secured blocks
    Output: Mapped points
1 for $i = 0; i <$ no of secured blocks $B; i++$;
2     Sender : obtain the decimal value $x_i$ for the secured block;
3     Sender : $x_i = x_i \times 16$ ;
4     Sender : obtain corresponding $y_i$ using the equation;
5     Sender : if $y_i$ cannot satisfy the equation;
6     Sender : $x_i = x_i + 1$;
7 Sender : repeat the loop until find $y_i$;
8 Mapped points $\leftarrow$ the results
```



B_1
B_2
B_3
B_B
x_1 = Decimal value of B_1
$x_1 = x_1 \times 16$
Find y_1 such that: $y_1^2 \equiv x_1^3 + a.\, x_1 + b \bmod p$
y_1 found?
No
Yes
(x_1, y_1) mapped and secured

Figure 9. Mapping secured blocks to an elliptic curve.

3.3. Encrypting and Decrypting the Mapped Points

In many schemes, the assumption is made that the mapping phase is sufficient to secure the plain text and plays the encryption phase. However, this assumption is incorrect because the mapping phase to the elliptic curve means that these points belong to the generated elliptic curve and can benefit from the elliptic curve discrete logarithm problem (ECDLP). In our proposed scheme, we encrypt these points by adding them to gk_{sh}. Figure 10 illustrates the process for encrypting the mapped points.

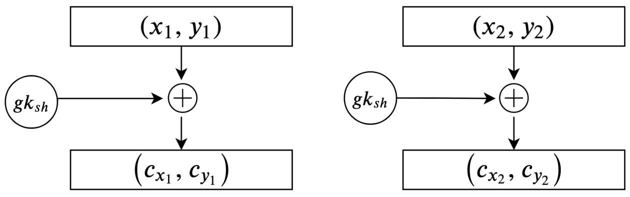


Figure 10. Encrypting mapped points using the shared group point.

The algorithm used to secure the transmitted message using the shared group point gk_{sh} is given below.

Algorithm 7: Encrypting mapped points using the shared group point.

```
Input: Mapped points $(x_i, y_i)$, $gk_{sh}$
Output: Ciphertexts
1 Sender : Calculate $gk_{sh}$;
2 for $i = 0; i <$ no of mapped points $B; i++$;
3     $(c_{xi}, c_{yi}) = gk_{sh} + (x_i, y_i)$;
4 Sender : repeat until finish;
5 $(c_{xi}, c_{yi}) \leftarrow$ the results
```

The remaining phases constitute a reversal of the previous phases. Therefore, the secured points are decrypted, and the recipient subtracts them with gk_{sh}. As illustrated before, all parties have the same shared group point gk_{sh}, which is used to encrypt and decrypt the messages. In Figure 11, an illustration of how to decrypt secured points is given.

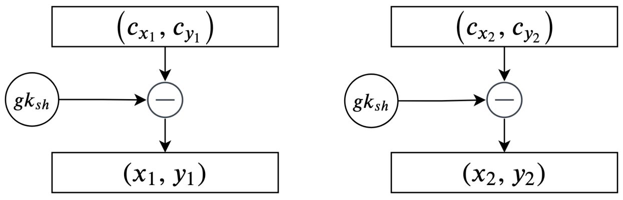


Figure 11. Decrypting secured points using the shared group key.

Algorithm 8 provides an overview of the steps involved in completing the decryption process:

Algorithm 8: Decrypting cipher texts using the shared group key.

```
Input: Encrypted points $(c_{xi}, c_{yi})$, $gk_{sh}$
Output: Decrypted points
1 Recipient : Calculate $gk_{sh}$;
2 for $i = 0; i <$ no of Encrypted points $B; i++$;
3     $(x_i, y_i) = (c_{xi}, c_{yi}) - gk_{sh}$;
4 Recipient : repeat until finish;
5 $(x_i, y_i) \leftarrow$ the results
```

3.4. Decoding and Converting the Decrypted Points into Plain Text

The next phases are concerned with completing the previous phase, which focus on decoding and converting the decrypted points into a plain text M. There are two pairs on each set of these points, denoted x_i and y_i. It is worth noting that y_i is used in two phases only: the encryption and decryption phases. In these phases, the values are used to add and subtract two points on the elliptic curve. Therefore, in this phase, we are only concerned with x_i, that represent the binary values of the characters in the plain text. The process that describes the decoding phase is illustrated in Figure 12.

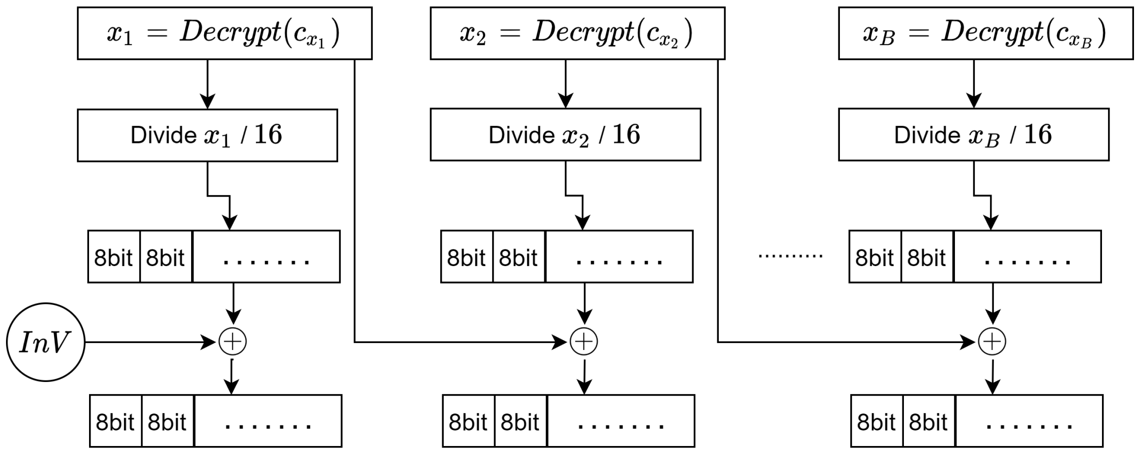


Figure 12. Converting and decoding x_i values into binary values.

Algorithm 9 demonstrates the steps required to convert the decrypted points x_i into binary values:

Algorithm 9: Decoding decrypted points into binary values.

```
Input: $Decrypted\ points\ x_i,\ InV$
Output: $Binary\ values$
1 $Recipient:\ obtain\ x_i\ value\ for\ the\ decrypted\ points$;
2 $for\ i = 0; i <\ no\ of\ decrypted\ points\ B; i++$;
3     $let\ InV_{save} = x_i$;
4     $let\ x_i = x_i \div 16$;
5     $let\ x_i = x_i \oplus InV$;
6     $let\ InV = InV_{save}$;
7 $Recipient:\ repeat\ until\ finish$;
8 $Binary\ values \leftarrow the\ results\ x_i$
```

Finally, the binary values obtained in the previous phase are converted into their corresponding characters. These binary values represent the plain text message M, and so they need to be converted into their ASCII values. The conversion process is illustrated in Figure 13.

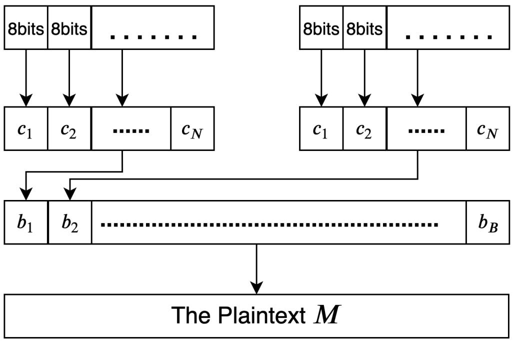


Figure 13. Converting binary values into plain text M.

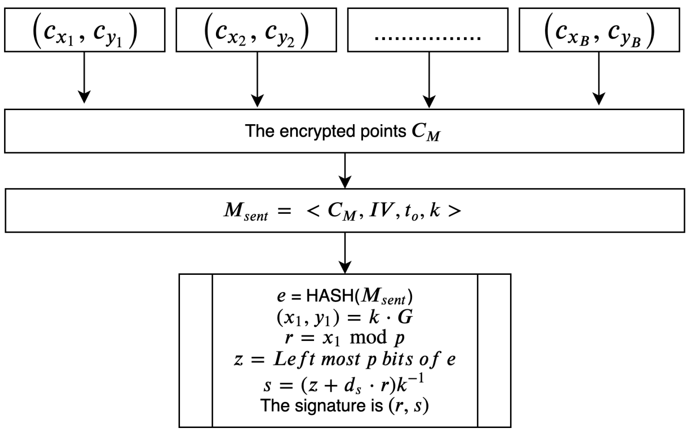
Algorithm 10 shows how to convert binary values into a plaintext M.

Algorithm 10: Converting binary values into plain text .

```
H] Input: $Binary\ values\ , B$
Output: $The\ plaintext\ message\ M$
1 $Recipient:\ Get\ the\ binary\ values$;
2 $for\ i = 0; i <\ no\ of\ binary\ values\ B; i++$;
3     $convert\ each\ 8bits\ into\ its\ corresponding\ ASCII\ code$;
4     $for\ each\ N\ char\ aggregate\ to\ single\ block$;
5 $Recipient:\ repeat\ until\ finish$;
6 $The\ message\ M \leftarrow the\ results$
```

3.5. Signing and Verifying the Encrypted Message

The intention of AE schemes using the encrypt then sign approach is to transmit messages between parties without compromising confidentiality, security, and integrity. In the proposed scheme, confidentiality (encryption) is maintained by the previous phases. Thus, to assure integrity (sign), the sender signs the message using his or her private key d_s, which relies on ECDSA. The message M_{sent} consists of a set of tuples, which contain the encrypted points C_M and InV, the timestamp t_o, and a random signature integer k. The signing process is illustrated in Figure 14.



(c_{x_1}, c_{y_1})
(c_{x_2}, c_{y_2})
....
(c_{x_B}, c_{y_B})
The encrypted points C_M
$M_{sent} = < C_M, IV, t_o, k >$
e = HASH(M_{sent})
$(x_1, y_1) = k \cdot G$
$r = x_1 \bmod p$
$z = Left\ most\ p\ bits\ of\ e$
$s = (z + d_s \cdot r)k^{-1}$
The signature is (r, s)

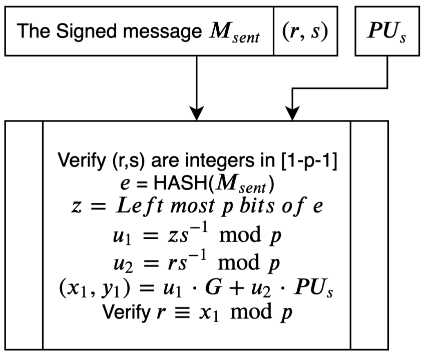
Figure 14. Sender ensures message integrity by signing it with d_s.

The following algorithm outlines the process used to sign the C_M:

Algorithm 11: Process of signing the message.

```
Input: The Message $M_{sent}$
Output: Signature pairs $(r, s)$
1 Sender : calculate $e = HASH(M_{sent})$;
2 Sender : calculate $z = left\ most\ p\ bits\ of\ e$;
3 Sender : select random value $k$;
4 Sender : calculate $(x_1, y_1) = k \times G$;
5 Sender : calculate $r = x_1\ mod\ p\ where\ r \neq 0$;
6 Sender : for $r = 0$ go to 3;
7 Sender : calculate $s = (z + d_s * r)\ k^{-1}$ for $s = 0$ go to 3;
8 Sender : $(r, s) \leftarrow$ ciphertext signature
```

The receiver of a signed message verifies it using the sender's public key. The verification process is illustrated in Figure 15.



The Signed message M_{sent}
(r, s)
PU_s
Verify (r,s) are integers in [1-p-1]
e = HASH(M_{sent})
$z = Left\ most\ p\ bits\ of\ e$
$u_1 = zs^{-1} \bmod p$
$u_2 = rs^{-1} \bmod p$
$(x_1, y_1) = u_1 \cdot G + u_2 \cdot PU_s$
Verify $r \equiv x_1 \bmod p$

Figure 15. Receiver verifies signed message.

Algorithm 12 outlines the steps involved in verifying the integrity of the received message using the sender's public key.

Algorithm 12: Recipient verification of a signed message.

```
Input: $M_{sent}$, $(r, s)$, $PU_s$
Output: $Verified\ ciphertexts$
1 $Recipient: check\ (r, s)\ are\ integers \in 1, 2, 3, ..., p-1)$;
2 $Recipient: calculate\ e = HASH(M_{sent})$;
3 $Recipient: calculate\ z = leftmostp\ bits\ of\ e$;
4 $Recipient: calculate\ u_1 = es^{-1}\ mod\ p$;
5 $Recipient: calculate\ u_2 = rs^{-1}\ mod\ p$;
6 $Recipient: calculate\ (x_1, y_1) = u_1 * G + u_2 * PU_s$;
7 $Verified \leftarrow\ r \equiv x_1\ mod\ p$
```

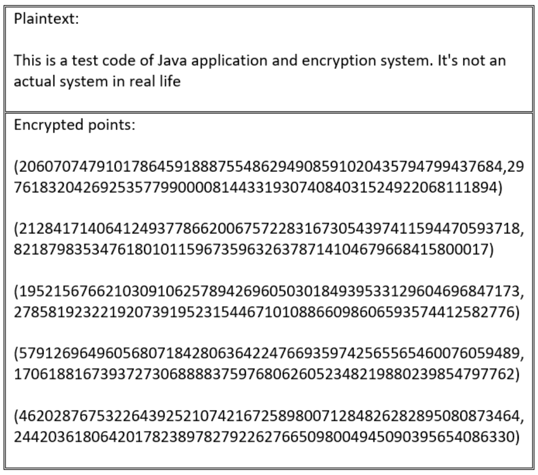
4. Security Analysis and Performance Evaluation

Every scheme discussed in this study's literature review is vulnerable to an encryption flaw. For instance, the encryption schemes presented in [55,60–62] produce the same cipher text when repeated several times, as outlined in Figures 16 and 17.

These schemes suffer from an encryption weakness because they offer the same cipher text if the sender encrypts the plain text using the same encryption key. The main factor that underlies this flaw is that the encryption process in these schemes neglects to consider the importance of manipulating plain text before encrypting and decrypting it. This ensures that a distinct cipher text is produced for every encryption process. Therefore, in the absence of a manipulation step, the adversary can learn from the cipher text, determining whether certain cipher texts are the same or different.

In the scheme proposed by [59], this flaw is resolved by using XOR with InV. Figure 18 shows that the cipher text outputted from the encryption process is different to the cipher text generated by the process depicted in Figure 19, which uses the same plain text. However, these schemes are vulnerable to the same flaw, particularly in the special cases outlined in Figure 20. This flaw occurs when the blocks divided for XOR with InV are equal to $B_1 = B_2 = B_3 = ... = B_n$. In this case, $B'_1 = InV \oplus B_1$, $B'_2 = B'_1 \oplus B_2$, $B'_3 = B'_2 \oplus B_3$, and so on. Therefore, in the event that all blocks are the same, an attacker can exploit the encryption flaw because XOR with InV operations become:

$$\begin{aligned} B'_1 &= InV \oplus B_1 \\ B'_2 &= B'_1 \oplus B_2 \\ B_2 &= B_1 \rightarrow \\ B'_2 &= InV \oplus B_1 \oplus B_1 \\ Result \rightarrow B'_2 &= InV \end{aligned} \tag{6}$$


Plaintext:

This is a test code of Java application and encryption system. It's not an actual system in real life

Encrypted points:

(20607074791017864591888755486294908591020435794799437684,29761832042692535779900008144331930740840315249220681118 94)

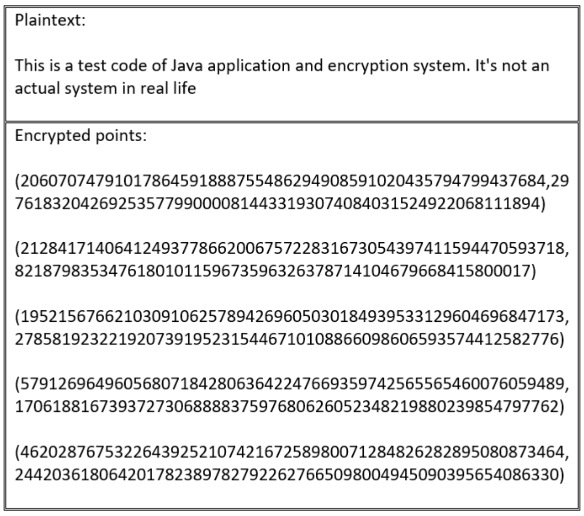
(212841714064124937786620067572283167305439741159447059371 8,821879835347618010115967359632637871410467966841580001 7)

(195215676621030910625789426960503018493953312960469684717 3,278581923221920739195231544671010886609860659357441258277 6)

(579126964960568071842806364224766935974256556546007605948 9,170618816739372730688883759768062605234821988023985479776 2)

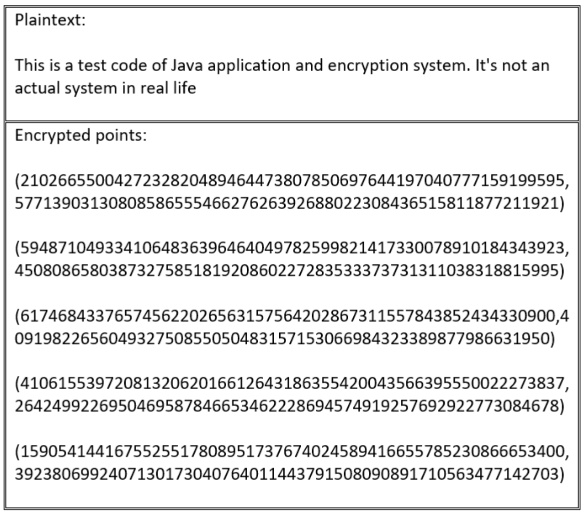
(462028767532264392521074216725898007128482628289508087346 4,244203618064201782389782792262766509800494509039565408633 0)

Figure 16. Cipher text generated by first encryption process.



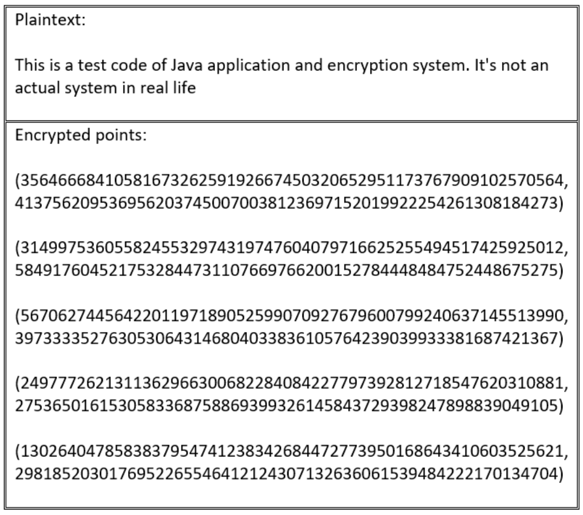
```
Plaintext:

This is a test code of Java application and encryption system. It's not an
actual system in real life

Encrypted points:

(20607074791017864591888755486294908591020435794799437684,29
76183204269253577990000814433193074084031524922068111894)

(2128417140641249377866200675722831673054397411594470593718,
8218798353476180101159673596326378714104679668415800017)

(1952156766210309106257894269605030184939533129604696847173,
2785819232219207391952315446710108866098606593574412582776)

(5791269649605680718428063642247669359742565565460076059489,
1706188167393727306888837597680626052348219880239854797762)

(4620287675322643925210742167258980071284826282895080873464,
2442036180642017823897827922627665098004945090395654086330)
```

Figure 17. Cipher text generated by second encryption process using the same plain text as in Figure 16.



```
Plaintext:

This is a test code of Java application and encryption system. It's not an
actual system in real life

Encrypted points:

(2102665500427232820489464473807850697644197040777159199595,
5771390313080858655546627626392688022308436515811877211921)

(5948710493341064836396464049782599821417330078910184343923,
4508086580387327585181920860227283533373731311038318815995)

(617468433765745622026563157564202867311557843852434330900,4
091982265604932750855050483157153066984323389877986631950)

(4106155397208132062016612643186355420043566395550022273837,
2642499226950469587846653462228694574919257692922773084678)

(1590541441675525517808951737674024589416655785230866653400,
3923806992407130173040764011443791508090891710563477142703)
```

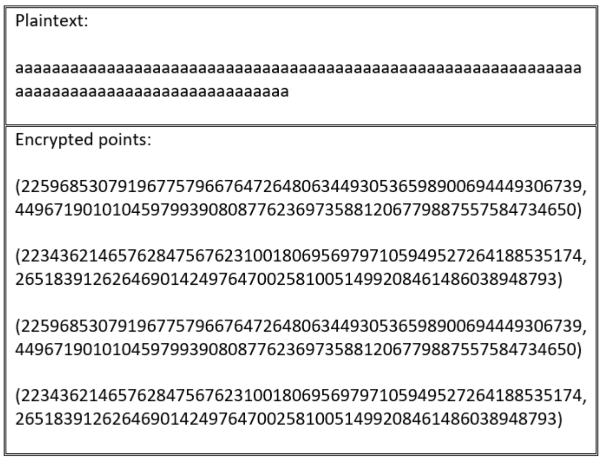
Figure 18. Cipher text generated by first encryption process using XOR with InV.



```
Plaintext:

This is a test code of Java application and encryption system. It's not an
actual system in real life

Encrypted points:

(3564666841058167326259192667450320652951173767909102570564,
4137562095369562037450070038123697152019922254261308184273)

(3149975360558245532974319747604079716625255494517425925012,
5849176045217532844731107669766200152784448484752448675275)

(5670627445642201197189052599070927679600799240637145513990,
3973333527630530643146804033836105764239039933381687421367)

(2497772621311362966300682284084227797392812718547620310881,
2753650161530583368758869399326145843729398247898839049105)

(1302640478583837954741238342684472773950168643410603525621,
2981852030176952265546412124307132636061539484222170134704)
```

Figure 19. Cipher text generated by second encryption process with XOR different InV using the same plain text as in Figure 18.

This encryption flaw is illustrated in Figure 20.


Plaintext:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Encrypted points:

(22596853079196775796676472648063449305365989006944493067 39,
44967190101045979939080877623697358812067798875575847346 50)

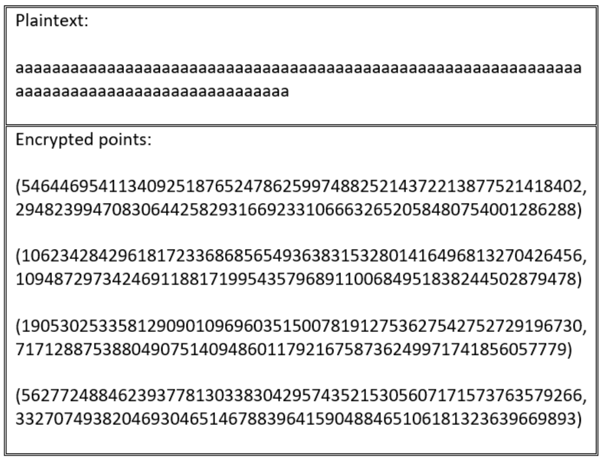
(22343621465762847567623100180695697971059495272641885351 74,
26518391262646901424976470025810051499208461486038948793)

(22596853079196775796676472648063449305365989006944493067 39,
44967190101045979939080877623697358812067798875575847346 50)

(22343621465762847567623100180695697971059495272641885351 74,
26518391262646901424976470025810051499208461486038948793)

Figure 20. Cipher text generated via encryption of plain text, followed by XORing with InV using a special text that produces the same blocks.

Based on the flaws identified in existing schemes, it is a valuable contribution to the literature to improve the encryption process with CBC. In the proposed scheme, all previous flaws are resolved by using CBC with InV derived from mapping points to secure the cipher text in all forms, as described in Algorithm 5. The encryption operation generated by the optimized encryption process is illustrated in Figure 21.


Plaintext:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Encrypted points:

(54644695411340925187652478625997488252143722138775214184 02,
29482399470830644258293166923310666326520584807540012862 88)

(10623428429618172336868565493638315328014164968132704264 56,
10948729734246911881719954357968911006849518382445028794 78)

(19053025335812909010969603515007819127536275427527291967 30,
71712887538804907514094860117921675873624997174185605777 9)

(56277248846239377813033830429574352153056071715737635792 66,
33270749382046930465146788396415904884651061813236396698 93)

Figure 21. Improved encryption process generating different encrypted points even with the same blocks.

To evaluate the proposed schemes, security analysis was used with IND-CPA and IND-CCA. In the next subsections, results are given to show that the schemes examined in the literature review are not IND-CPA and IND-CCA. Therefore, it is necessary to prove that the novel scheme proposed in the presented study is both IND-CPA and IND-CCA.

4.1. Indistinguishability under Chosen Plain Text Attack IND-CPA

A scheme known as IND-CPA if the adversary has the power to submit as many $m_{i,0}$, $m_{i,1}$ $i = [1, 2, .., q]$ as desired and also when he or she receives the cipher texts for each message. Then, the adversary must submit two distinct messages, m_0 *and* m_1, *where* $|m_0| = |m_1|$, to the encryption oracle, receiving $c \leftarrow Enc(k, m_b)$. This is illustrated in Figure 22. The challenge is that the adversary is required to guess the value of b with a probability of the following:

$$\begin{aligned} Adv_{IND-CPA}[A,E] = & \\ & = |Pr[EXP(0)=1] - Pr[EXP(1)=1]| \\ & = negligible \end{aligned} \tag{7}$$

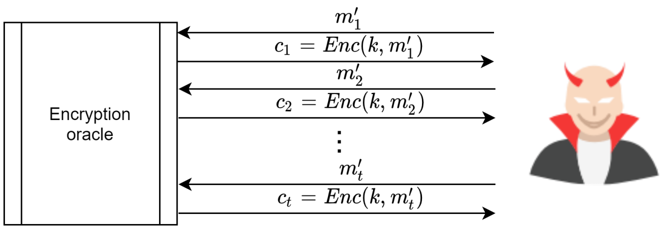


Figure 22. Adversary's power to submit unlimited messages to encryption oracle in CPA.

Thus, schemes proposed in [55,60–62] are insecure under CPA. An adversary can win a challenging game by submitting two identical messages, m_0, m_0, to the encryption oracle. In turn, the encryption oracle sends the cipher text c_0, which is the cipher text for m_0. In an experiment, the adversary sends two messages, m_0, m_1 where $|m_0| = |m_1|$, and the encryption oracle responds by encrypting one message and sending one cipher text c_b. The adversary is then challenged to guess the value of $b = 0 \; or \; 1$, and to determine which cipher text belongs to m_0 or m_1. However, the adversary can win the challenge with $Adv_{IND-CPA}[A,E] = 1$. This is because the encryption oracle in such schemes always encrypts the same messages with the same cipher texts and the same key. Therefore, since the adversary already has c_0, he or she can determine whether the value of b is 0 or 1. This is achieved by matching c_b with c_0. If $c_b = c_0$, then $b = 0$; otherwise, $b = 1$. This process is illustrated in Figure 23.

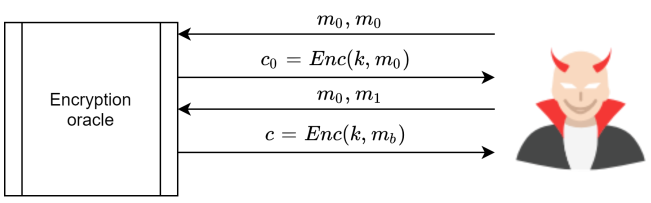


Figure 23. Steps needed for an adversary to win the Chosen Plain Text Attack (CPA) challenge.

Theorem 1. *The proposed scheme is (IND-CPA) such that the probability of the adversary to decrypt the message in the two experiments is negligible, as written in Equation (7).*

Proof. This theorem is proven by the fact that the proposed scheme uses a different value of InV for each experiment. Hence, the adversary can submit as many messages as he or she wants to the encryption oracle. The cipher texts generated by the experiment will not be the same, even for the same message. For example, in the second experiment, InV in $EXP(0)$ was different compared to InV in $EXP(1)$. Therefore, when the adversary submits the same message m_0, m_0 in $EXP(0)$, he or she will receive the corresponding cipher text in that experiment, which is c_0. However, when the adversary moves to $EXP(1)$ where the encryption oracle challenges the adversary, the two messages submitted by the adversary m_0, m_1 are not distinguishable. This is because the cipher text from m_0 in $EXP(1)$ is different from the cipher text from the same message m_0 in $EXP(0)$. For this reason, the probability that the adversary wins the challenge is $Adv_{IND-CPA}[A,E] = 0$

Let $\Pi = (Gen, Enc, Dec)$ be an ECC encryption scheme, and let us define the experiment between the challenger and the attacker A as follows:

- $IND - CPA_{\Pi}(A,k)$, where k is the key size:
  1. The challenger computes $(d, PU) \leftarrow Gen(2^k)$

2. The attacker A submits $1, 2, ..., q$ queries to the challenger to encrypt a plain text of their choice
3. The attacker A sends two messages (m_0, m_1) for the challenger to encrypt
4. The challenger computes $(c_b, InV) \leftarrow Enc(PU, (m_b \oplus InV))$
5. The attacker A outputs the value $b = 0$ or $b = 1$, and A wins if the probability of A guessing a correct value is not negligible

Consider the following games:

Game (0)

The attacker A makes a request for encrypting a message (m_0), and receives $(c_0, InV) \leftarrow Enc(PU, (m_0 \oplus InV))$.

Game (1)

The attacker A sends two messages (m_0, m_1), where m_0 is the same m_0 as in game 0. The challenger outputs $(c_b, InV') \leftarrow Enc(PU, (m_b \oplus InV'))$ to challenge A to guess the value of b.

Game (3)

The attacker A returns the value of b based on the comparison he or she can perform to distinguish between (m_0, m_1), which relies on game 0 and 1. However, the probability that A achieves this is 0 because the $c_0 \leftarrow Enc(PU, (m_0 \oplus InV))$ in game 0 is not the same as $c_0 \leftarrow Enc(PU, (m_0 \oplus InV'))$ in game 1. It is also relevant to note that the XOR operation in both games is undertaken with different IV. □

4.2. Indistinguishability Under Chosen Ciphertext Attack IND-CCA

An encryption scheme is IND-CCA if the adversary has the ability to submit as cipher texts as he wants $c_{i,0}$, $c_{i,1}$ $i = [1, 2, .., q]$ and to receive the plain text for that cipher text. In addition, the adversary submits any two distinct messages $m_{i,0}$ and $m_{i,1}$, where $|m_{i,0}| = |m_{i,1}|$ to the encryption oracle and receives $c \leftarrow Enc(k, m_b)$. This process is depicted in Figure 24. The challenge is that the adversary is required to guess the value of b to differentiate between the two experiment's games, with probability as follows:

$$\begin{aligned} Adv_{IND-CCA}[A, E] &= \\ &= |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| \\ &= negligible \end{aligned} \tag{8}$$

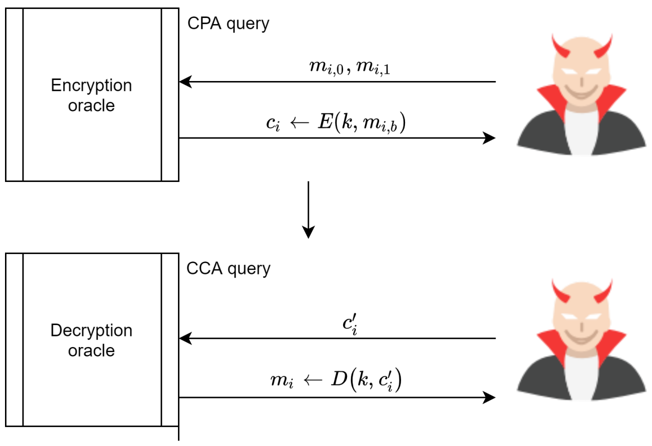


Figure 24. An adversary's power to submit countless messages to the encryption oracle in CCA.

With the above in mind, the scheme proposed in [59] is not secure under CCA. The adversary can win the challenging game by submitting two messages, m_0, m_1, to the encryption oracle. In turn, the encryption oracle sends the cipher text $c \leftarrow E(k, m_b) = (InV, c_b)$, which is the cipher text for either

m_0 or m_1. Subsequently, the adversary modifies c by XORing InV with R, where $R \neq 0$. This produces $c' = (InV \oplus R, c_b$ where $c_b = E(k, InV \oplus m_b$, which is sent to the decryption oracle for decryption.

The decryption oracle responds by decrypting the cipher text $D(k, c')$. This process relies on the modified $InV \oplus R$. The result is that the decryption process is $m_b \oplus InV \oplus (InV \oplus R) = m_b \oplus R$. Consequently, an adversary can guess the value of $b = 0$ or $b = 1$ by XORing $m_b \oplus R$, after which it can be compared directly with the received message m_b. Therefore, the adversary can win the challenge with $Adv_{IND-CCA}[A, E] = 1$. This process is illustrated in Figure 25.

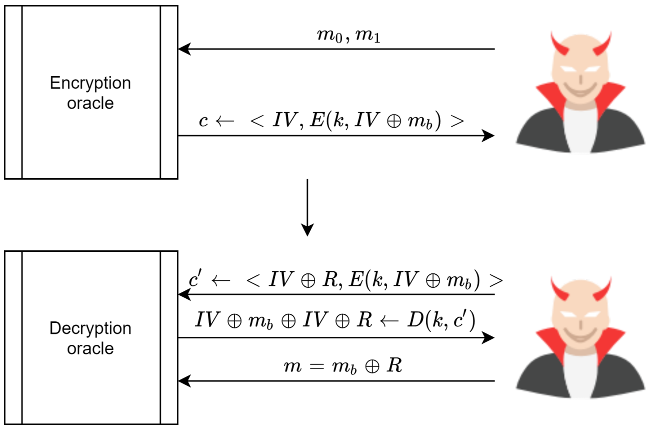

Encryption oracle
m_0, m_1
$c \leftarrow\ < IV, E(k, IV \oplus m_b) >$
Decryption oracle
$c' \leftarrow\ < IV \oplus R, E(k, IV \oplus m_b) >$
$IV \oplus m_b \oplus IV \oplus R \leftarrow D(k, c')$
$m = m_b \oplus R$

Figure 25. Steps needed for an adversary to win the CCA challenge.

Theorem 2. *The proposed scheme is indistinguishable under IND-CCA such that the probability of the adversary decrypting the message in the two experiments is negligible, as written in Equation (8).*

Proof. The proof for $Adv_{CCA}[B_1, E]$ starts by noting the adversary's ability to access both the encryption and decryption oracle. At the same time, the adversary cannot submit c_i received from the encryption oracle and then submit it for decryption to distinguish the experiment game. However, the adversary can modify the received $c_i \leftarrow E(k, m_i)$ and submit it for decryption as the modified $c_i' \neq c_i$. Thus, the adversary can win the challenging game by XORing InV with a random R, thereby guessing m_i (see Figure 25). The proposed scheme offers integrity to cipher texts. It also protects transmitted cipher texts from tampering. Therefore, the decryption oracle drops any modified c_i, meaning that the oracle responds to the adversary with $\perp \leftarrow < InV \oplus R, c_i) >$ for every modified message, as illustrated in Figure 26. Resultantly, the proposed scheme is negligible under Equation (8), meaning that it is indistinguishable under IND-CCA.

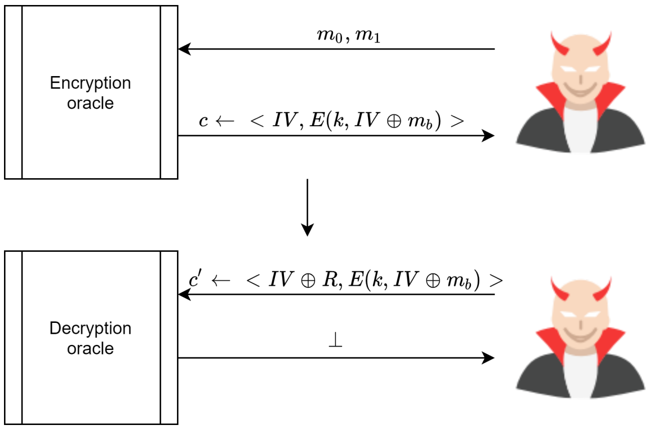

Encryption oracle
m_0, m_1
$c \leftarrow\ < IV, E(k, IV \oplus m_b) >$
Decryption oracle
$c' \leftarrow\ < IV \oplus R, E(k, IV \oplus m_b) >$
$\perp$

Figure 26. Proposed scheme IND-CCA proof.

Let $\Pi = (Gen, Enc, Dec, Sign, Ver)$ be an ECC encryption scheme, and let us define an experiment between the challenger and attacker A as follows:

- $IND - CCA_{\Pi}(A, k)$, where k is the key size:

  1. The challenger computes $(d, PU) \leftarrow Gen(2^k)$
  2. The attacker A submits $1, 2, ..., q$ queries to the challenger to decrypt their chosen cipher text
  3. The attacker A sends two messages (m_0, m_1) to the challenger for encryption
  4. The challenger computes $(c_b, InV, Sign) \leftarrow Enc(PU, (m_b \oplus InV))$
  5. The attacker A outputs the value of $b = 0$ or $b = 1$, where A wins if the probability of guessing A correctly is not negligible

  Consider the following games:

Game (0)
The attacker A makes a request for encrypting a message (m_0), and receives $(c_0, InV, Sign) \leftarrow Enc(PU, (m_0 \oplus InV))$.
Game (1)
The attacker A modifies the cipher text from game 0 $(c_0, InV, Sign)$ to XOR the InV with a random value $R \neq 0$. A request is made for the challenger to decrypt the cipher text $(c_0, InV \oplus R, Sign)$.
Game (3)
The attacker A can compute the plain text from $(m_b \oplus InV \oplus (InV \oplus R)) \leftarrow Dec(c_0, InV \oplus R, Sign)$, which results in $(m_b \oplus R)$. In this case, A can compare $(m_b \oplus R)$ with $(m_0 \oplus R)$ and $(m_1 \oplus R)$. However, A will fail to win because the challenger will discard the modified $(c_0, InV \oplus R, Sign)$. This stems from the fact that $Sign$ is invalid, meaning that probability of the success of A is 0. $\square$

4.3. Malleability Attack

Changing encrypted data can lead to the modification of the plain text after decryption, which is known as a malleability attack. For instance, the attacker modifies the InV with the value of 1, which means that the first block of the plain text message will be XORed with 1. As a result, the reset of blocks in the CBC will also be modified. The steps involved in the malleability attack are depicted in Figure 27.

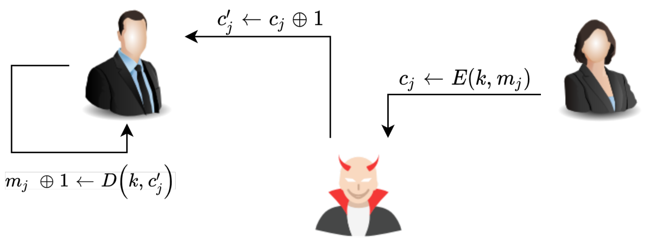

$c'_j \leftarrow c_j \oplus 1$
$c_j \leftarrow E(k, m_j)$
$m_j \oplus 1 \leftarrow D\left(k, c'_j\right)$

Figure 27. Overview of the malleability attack.

Theorem 3. *The proposed scheme is resistant to the malleability attack.*

Proof. It is known that an adversary can eavesdrop on the messages sent between the two parties. Adversaries can also change messages and return them to either party. However, the recipient checks the received message's integrity before decrypting the cipher text. Therefore, the recipient ignores the received message because it has an invalid signature. An illustration of this proof is given in Figure 28

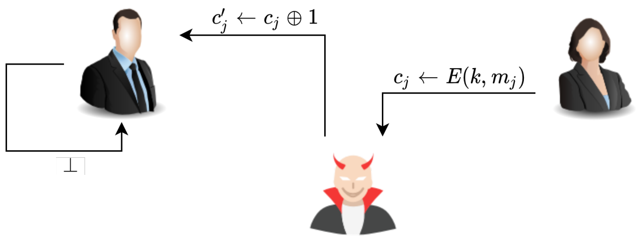


Figure 28. Resistance of the proposed scheme to the malleability attack.

□

A security analysis comparison was undertaken between the scheme proposed in this study and others mentioned in the literature (see Table 2).

Table 2. Security analysis comparison of proposed scheme and other schemes.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Barman, [60]	N	N	N	N	N	$\mathcal{O}(n \log n)$	n	N
Joglekar, [62]	N	N	N	N	N	N/A	N/A	N
Muthukuru, [59]	Y	N	N	N	N	$\mathcal{O}(n \log n)$	n	N
Duarah, [61]	N	N	N	N	N	$\mathcal{O}(n \log n)$	n	N
Das, [55]	N	N	N	N	N	N/A	N/A	N
Proposed scheme	Y	Y	Y	Y	Y	$\mathcal{O}(n \log n)$	2^n	Y

1. IND-CPA; 2. IND-CCA; 3. Resistance to malleability attack; 4. Integrity; 5. Authenticated encryption; 6. Complexity of mapping phase; 7. No of rounds based on n appending bits; 8. Offers nonrepudiation

4.4. Performance Evaluation

The main goal of the proposed scheme was to provide a secure scheme that resolved encryption flaws that yield to CPA and CCA attacks. The study also sought to offer a scheme with a suitable level of performance in constrained environments. The security aspect of the proposed scheme was proven in the previous section, and this section evaluated the performance of the scheme based on two criteria: enhancing the process of mapping points to an elliptic curve; and time, space overhead, and power consumption in a simulation environment.

4.4.1. Mapping Points to an Elliptic Curve

Mapping points to an elliptic curve should be undertaken correctly and efficiently. As illustrated in Figure 29, 50% of x_i points cannot be mapped to the EC as there is no y_i axis that meets the EC's equation. Therefore, it is necessary to increase the value of x_i by 1, and then to recompute y_i until a value is found that matches. As a result, the x_i point is eventually mapped. In many schemes, the characters in plain text are converted into numerical values based on the ASCII table, which facilitates their mapping onto the elliptic curve. Therefore, when these values are not mapped from the beginning, the value to be mapped increases. However, this step changes the original value, and the plain text is lost. For this reason, to overcome this problem, two methods are widely used to map points: the probability method and the appending method.

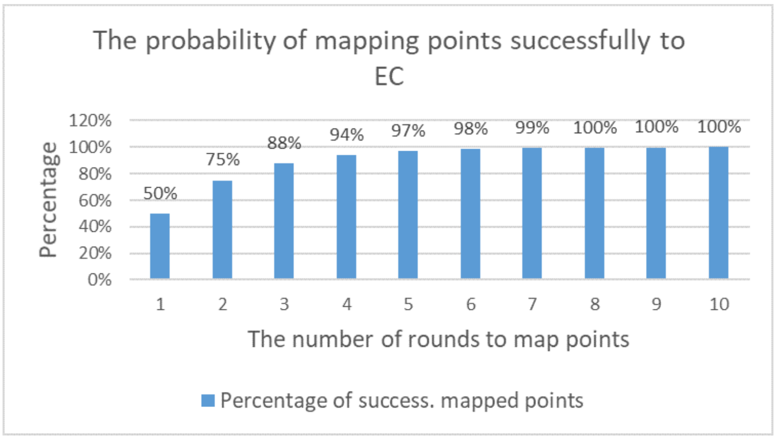


Figure 29. Probability of successfully mapping points to an elliptic curve based on number of rounds.

In the probability method, which was introduced by King (2009) [64], the value of k is defined, which represents the number of rounds needed to map the points. This value is multiplied by x_i, and the product is used in the mapping phase with ability to increment it in k rounds. In turn, the value of x_i can be restored by calculating the value of the mapped point: $\lfloor \frac{mapped\ point}{k} \rfloor$. In the second method, the appending method, x_i is appended by number of bits representing the number of rounds required. For instance, if the scheme defines the appended bits as 000, then the number of rounds that can be safely used in the mapping phase is $2^3 = 8$ rounds.

Each method has advantages and disadvantages relating to computational efficiency and the maximum number of rounds. The probability method is more efficient and less complex than the appending method. Using the Harvey-Hoeven algorithm [65], the multiplication operation complexity is $\mathcal{O}(n \log n)$, where n is the numerical value size in bits. However, the complexity of appending two texts (i.e., concatenations) is denoted as $\mathcal{O}(n^2)$ [66], which highlights the fact that the appending method is more complex than the probability method. This is illustrated in Figure 30.

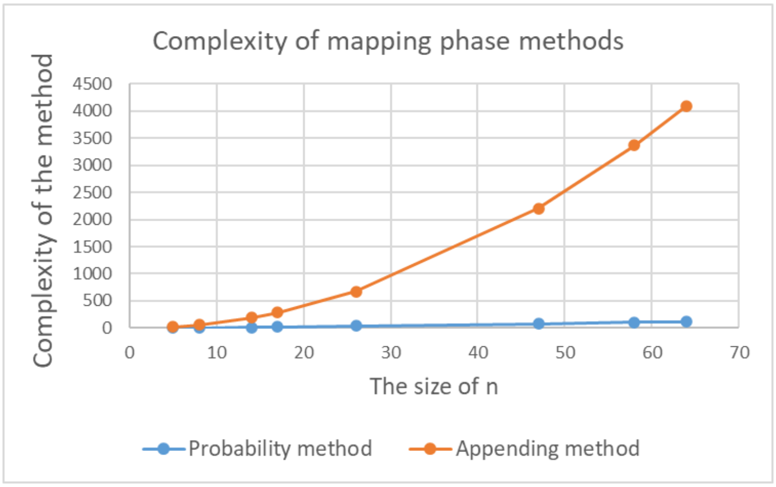


Figure 30. Complexity comparison of probability and appending mapping methods.

The appending method guarantees to provide a maximum number of rounds base on the appended bits. As the appending methods concatenate fixed bits b to the corresponding binary x_i value, x_i can be incremented 2^b times. In contrast, the probability method increases the same size of bits, but in many cases, it would allow fewer round increments compared to the appending method. This is outlined in Figure 31.

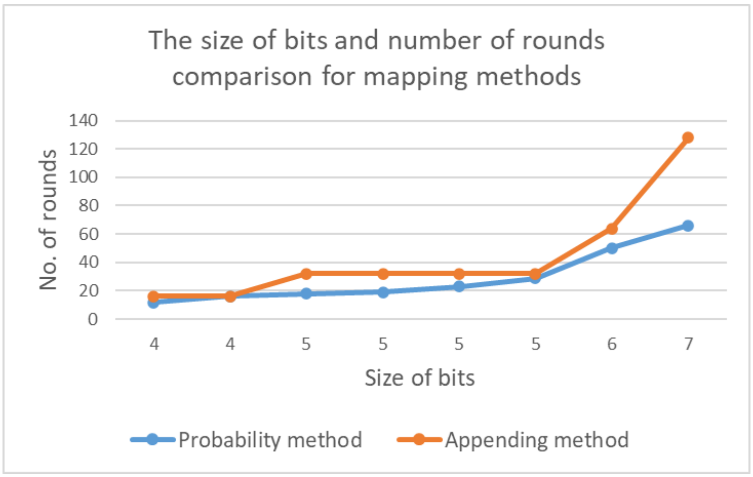


Figure 31. Comparison of number of rounds used in mapping phase method and the size of bits added to x_i.

In the proposed scheme, the probability method was used to improve performance. Hoevewr, to increase the efficiency of the choice of k, the proposed scheme took advantage of the appending method to select this value. This was achieved by recalculating $k' = 2^{\lceil logk \rceil}$, which was used to set the number of rounds to the maximum value of the extra bits added to the value of x_i, as shown in Figure 32.

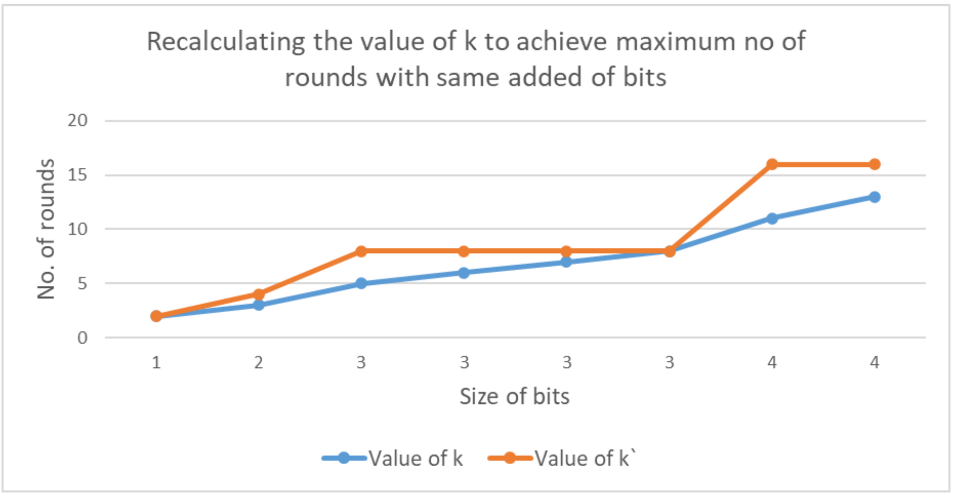


Figure 32. Recalculating k' to increase the maximum number of rounds to the size of added bits.

The enhancement made to the probability method increases the number of mapping rounds with same size of the padding bit. As a result, the efficiency was more than 80% in some cases. Figure 33 shows the percentage of improvement that the proposed enhanced selection of k method offers compared to the probability method.

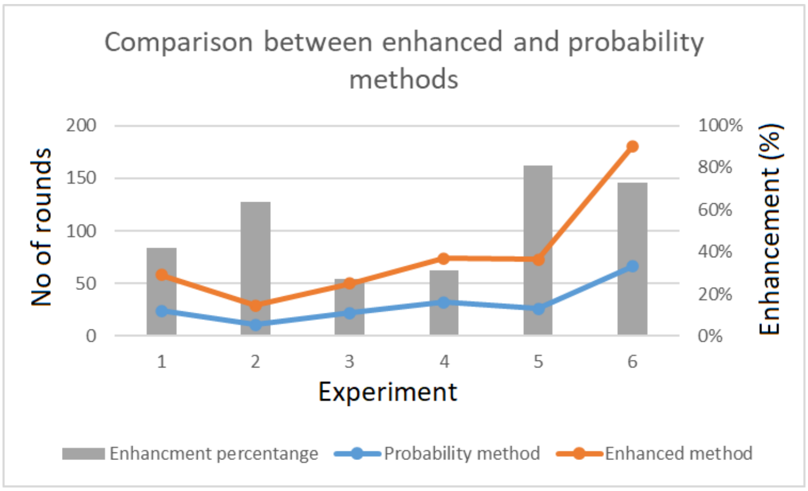


Figure 33. Percentage improvement between the enhanced selection of k provides in comparison with probability method.

4.4.2. Simulation Performance Evaluation

The performance evaluation of the proposed scheme focused on whether the enhancement of the encoding and mapping phases, which was intended to improve resistance to encryption attacks, negatively affected performance. Therefore, the performance of the proposed scheme against other schemes took place to evaluate time overhead, space usage, and power consumption. The schemes are divided to three groups: schemes that did not use InV or any other methods to manipulate the mapped points as [55,60–62]; schemes that used fixed InV to overcome CPA, including [59]; and finally, schemes that used CBC to overcome the CCA (in this case, only the proposed scheme).

The performance evaluation involved running an experimental simulation 5 times for 20 s each using a low computation device. Figure 34 illustrates time overhead in the three groups included in the performance evaluation. Notably, all three groups were associated with similar levels of utilization. Therefore, it is reasonable to conclude that improving the security of the proposed scheme did not affect the time overhead. Space usage variation was identified during the experiment, as illustrated in Figure 35. Additionally, power consumption for each group was determined based on values (high, medium, low, or none), which are shown in Figure 36. The results indicate that power consumption was similar across the three groups.

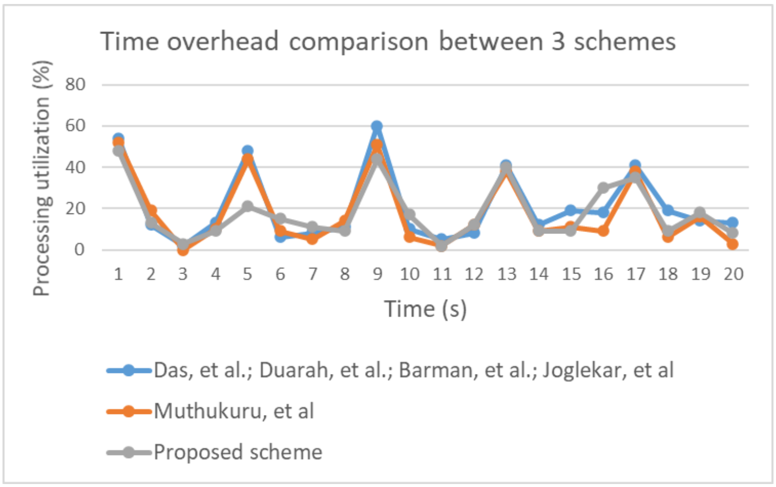


Figure 34. Comparison of time overhead between the proposed scheme and other schemes.

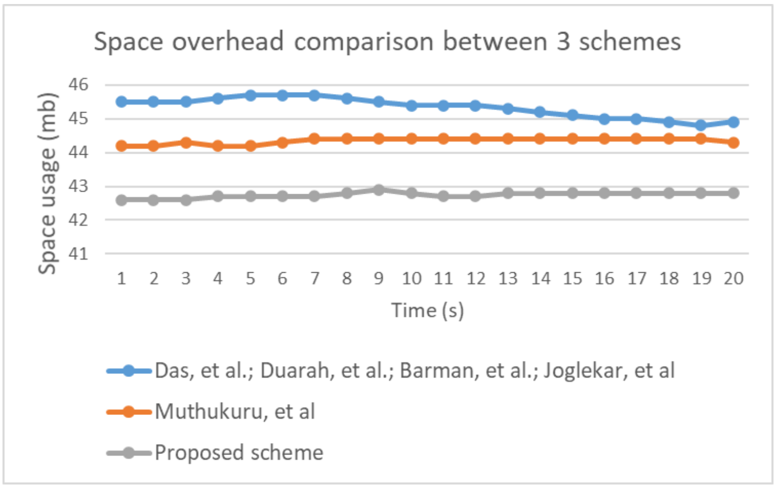


Figure 35. Comparison of space usage between the proposed scheme and other schemes.

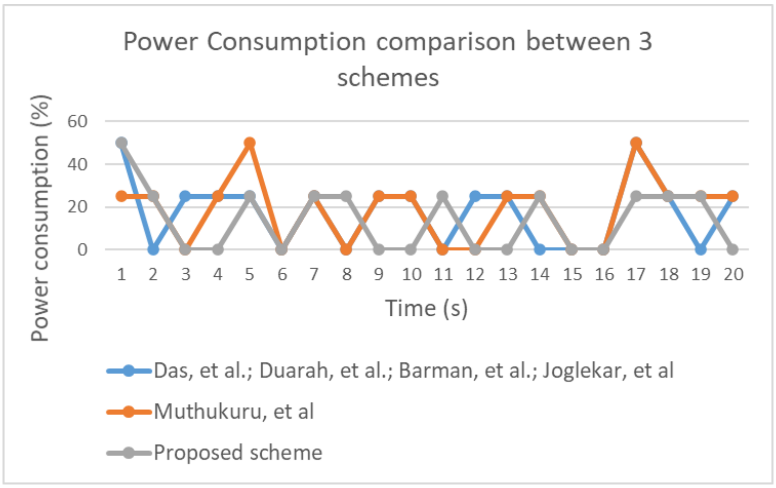


Figure 36. Comparison of power consumption between the proposed scheme and other schemes.

It is worth mentioning that, the proposed scheme was simulated, tested and compared to the state of the art schemes in a simulated environment using an Android Virtual Device (AVD). AVD has variety of low computation devices images built by Android OS, and it provides processing space and power consumption monitoring and logging. In addition, we used the Bluej Java Development Environment to code the proposed scheme. The platform used to host the AVD is based on Windows 10, and, in terms of hardware, an Intel Core i7-4510U was used with a 128GB SSD and 8GB RAM.

5. Conclusions and Future Research

This study proposes a novel approach to elliptic curve cryptography (ECC) that offers AE properties to secure cipher text and to enhance the encoding of text effectively and map the encoded text to an elliptic curve. Previous schemes neglect to consider the importance of the encoding phase, which makes them vulnerable to attack. Therefore, this study focused on the encoding phase, seeking to secure it against several encryption attacks, including CPA, CCA, and malleability attacks. This study also undertook a security analysis to present a proof for the resistance of the proposed scheme against specific encryption attacks. Additionally, the study conducted a performance evaluation to compare the impact of the security enhancement of the proposed scheme on time overhead, space usage, and power consumption to other schemes. The simulation experiment shows that the proposed scheme performed just as well as the other schemes, with no noticeable increase in computation overhead. As a result, the proposed scheme outperforms the security of other schemes and maintains the same computational overhead.

In future research, the authors intend to implement the proposed scheme in different environments such as enhance the protection using TPM-based for mobile agents. Moreover, adapt the scheme in Monitoring the cloud computing architecture to enhance Dynamic Security Properties. In addition, apply the proposed scheme with Policy Based Management to increase the Security of Cloud Computing. These application of proposed scheme lead to study the security analysis and performance evaluation in comparison with other similar schemes. Additionally, more security properties may be added to the study to increase the security requirements, which might be required in new environments.

Author Contributions: Conceptualization, H.A. and A.A.; methodology, A.A.; software, H.A.; validation, A.A.; formal analysis, H.A.; investigation, H.A.; resources, A.A.; data curation, A.A.; writing—original draft preparation, H.A.; writing—review and editing, A.A.; visualization, H.A.; supervision, A.A.; project administration, A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, "Ministry of Education" in Saudi Arabia for funding this research work through the project number IFKSURG-1437-035.

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

Abbreviations

The following abbreviations are used in this manuscript:

ECC	Elliptic Curve Cryptography
IoT	Internet of Things
InV	Initial Vector
CBC	Cipher Block Chaining
AE	Authenticated Encryption
CPA	Chosen Plaintext Attack
CCA	Chosen Ciphertext Attack
ECIES	Elliptic Curve Integrated Encryption Scheme
ECDLP	Elliptic Curve Discrete Logarithm Problem

References

1. Labrado, C.; Thapliyal, H.; Prowell, S.; Kuruganti, T. Use of thermistor temperature sensors for cyber-physical system security. *Sensors* **2019**, *19*, 3905.
2. Abdullah, A.; Kaur, H.; Biswas, R. Universal Layers of IoT Architecture and Its Security Analysis. In *New Paradigm in Decision Science and Management*; Springer: Singapore, 2020; pp. 293–302.
3. Ram, R.S.; Kumar, M.V.; Ramamoorthy, S.; Balaji, B.S.; Kumar, T.R. An Efficient Hybrid Computing Environment to Develop a Confidential and Authenticated IoT Service Model. In *Wireless Personal Communications*; Springer: Cham, Switzerland, 2020; pp. 1–25.
4. Pasupuleti, S.K.; Varma, D. Lightweight ciphertext-policy attribute-based encryption scheme for data privacy and security in cloud-assisted IoT. In *Real-Time Data Analytics for Large Scale Sensor Data*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 97–114.
5. Vishnoi, P.; Shimi, S.; Kumar, A. Symmetric Cryptography and Hardware Chip Implementation on FPGA. In *Intelligent Communication, Control and Devices*; Springer: Singapore, 2020; pp. 945–955.
6. Chatterjee, S.; Samaddar, S.G. A robust lightweight ECC-based three-way authentication scheme for IoT in cloud. In *Smart Computing Paradigms: New Progresses and Challenges*; Springer: Singapore, 2020; pp. 101–111.
7. Moghadam, M.F.; Mohajerzdeh, A.; Karimipour, H.; Chitsaz, H.; Karimi, R.; Molavi, B. A privacy protection key agreement protocol based on ECC for smart grid. In *Handbook of Big Data Privacy*; Springer: Cham, Switzerland, 2020; pp. 63–76.
8. Yuen, K.K.F. Towards a Cybersecurity Investment Assessment method using Primitive Cognitive Network Process. In Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, 11–13 February 2019; pp. 068–071.

9. Biswas, C.; Gupta, U.D.; Haque, M.M. An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 7–9 Febuary 2019; pp. 1–5.
10. Tiburski, R.T.; Moratelli, C.R.; Johann, S.F.; Neves, M.V.; de Matos, E.; Amaral, L.A.; Hessel, F. Lightweight Security Architecture Based on Embedded Virtualization and Trust Mechanisms for IoT Edge Devices. *IEEE Commun. Mag.* **2019**, *57*, 67–73.
11. Verma, G.; Liao, M.; Lu, D.; He, W.; Peng, X.; Sinha, A. An optical asymmetric encryption scheme with biometric keys. *Opt. Lasers Eng.* **2019**, *116*, 32–40.
12. Paar, C.; Pelzl, J. *Understanding Cryptography: A Textbook for Students and Practitioners*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.
13. Almajed, H.N.; Almogren, A.S.; Altameem, A. A Resilient Smart Body Sensor Network Through Pyramid Interconnection. *IEEE Access* **2019**, *7*, 51039–51046.
14. Mangia, M.; Pareschi, F.; Rovatti, R.; Setti, G. Low-cost security of iot sensor nodes with rakeness-based compressed sensing: Statistical and known-plaintext attacks. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 327–340.
15. Liao, M.; He, W.; Lu, D.; Wu, J.; Peng, X. Security enhancement of the phase-shifting interferometry-based cryptosystem by independent random phase modulation in each exposure. *Opt. Lasers Eng.* **2017**, *89*, 34–39.
16. Ahmed, S.; Zaman, A.; Zhang, Z.; Alam, K.M.R.; Morimoto, Y.; others. Semi-Order Preserving Encryption Technique for Numeric Database. *Int. J. Netw. Comput.* **2019**, *9*, 111–129.
17. Davoli, L.; Veltri, L.; Ferrari, G.; Amadei, U. Internet of Things on Power Line Communications: An Experimental Performance Analysis. In *Smart Grids and Their Communication Systems*; Springer: Singapore, 2019; pp. 465–498.
18. Debnath, S.; Nunsanga, M.V.; Bhuyan, B. Study and Scope of Signcryption for Cloud Data Access Control. In *Advances in Computer, Communication and Control*; Springer: Singapore, 2019; pp. 113–126.
19. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209.
20. AlMajed, H.N.; AlMogren, A.S. Simple and Effective Secure Group Communications in Dynamic Wireless Sensor Networks. *Sensors* **2019**, *19*, 1909.
21. Yin, Y.; Wu, L.; Peng, Q.; Zhang, X. A Novel SPA on ECC with Modular Subtraction. In Proceedings of the 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 9–11 November 2018 pp. 179–182.
22. Galbraith, S.D.; Vercauteren, F. Computational problems in supersingular elliptic curve isogenies. *Quantum Inf. Process.* **2018**, *17*, 265.
23. Wu, T.; Wang, R. Fast unified elliptic curve point multiplication for NIST prime curves on FPGAs. *J. Cryptogr. Eng.* **2019**, *9*, 401–410.
24. Shahroodi, T.; Bayat-Sarmadi, S.; Mosanaei-Boorani, H. Low-Latency Double Point Multiplication Architecture Using Differential Addition Chain Over GF (2^m). *IEEE Trans. Circuits Syst.* **2019**, *66*, 1465–1473.
25. Mrabet, A.; El-Mrabet, N.; Lashermes, R.; Rigaud, J.B.; Bouallegue, B.; Mesnager, S.; Machhout, M. High-performance Elliptic Curve Cryptography by Using the CIOS Method for Modular Multiplication. In *International Conference on Risks and Security of Internet and Systems*; Springer: Cham, Switzerland, 2016; pp. 185–198.
26. Ganesh, M.G.G. Secure Method for Text Encryption using Elliptic Curve Cryptography. *Int. J.* **2018**, *3*, 11–15.
27. Mahto, D. Data Communication Security Modeling Using Elliptic Curve Cryptography and Biometrics. Ph.D. Thesis, Nit Jamshedpur, Jharkhand, India, 2018.
28. Kumar, R. Cryptanalysis of Protocol for Enhanced Threshold Proxy Signature Scheme Based on Elliptic Curve Cryptography for Known Signers. In *Knowledge Computing and Its Applications*; Springer: Singapore, 2018; pp. 191–211.
29. Liu, Z.; Seo, H. IoT-NUMS: Evaluating NUMS elliptic curve cryptography for IoT platforms. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 720–729.
30. Shah, D.P.; Shah, P.G. Revisting of elliptical curve cryptography for securing Internet of Things (IOT). In Proceedings of the 2018 Advances in Science and Engineering Technology International Conferences (ASET), Abu Dhabi, UAE, 6 Febuary–5 April 2018; pp. 1–3.

31. Fournaris, A.P.; Dimopoulos, C.; Moschos, A.; Koufopavlou, O. Design and leakage assessment of side channel attack resistant binary edwards Elliptic Curve digital signature algorithm architectures. *Microprocess. Microsyst.* **2019**, *64*, 73–87.
32. Reddy, A.G.; Das, A.K.; Odelu, V.; Ahmad, A.; Shin, J.S. A Privacy Preserving three-factor authenticated key agreement protocol for client–server environment. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 661–680.
33. AlSaad, S.N.; Naji, A.K. Elliptic Curve Video Encryption in Mobile Phone Based on Multi-Keys and Chaotic Map. *Al-Mustansiriyah J. Sci.* **2018**, *29*, 106–116.
34. Shah, D.P.; Shah, N.P. Implementation of Digital Signature Algorithm by using Elliptical Curve p-192. *Aust. J. Wirel. Technol. Mobil. Secur.* **2019**, *1*, 1–4. e-ISSN 2200-1883.
35. Abdullah, K.E.; Ali, N.H.M. Security Improvement in Elliptic Curve Cryptography. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 122–131.
36. Hu, X.; Zheng, X.; Zhang, S.; Li, W.; Cai, S.; Xiong, X. A High-Performance Elliptic Curve Cryptographic Processor of SM2 over GF (p). *Electronics* **2019**, *8*, 431.
37. Reyad, O. Text message encoding based on elliptic curve cryptography and a mapping methodology. *Inf. Sci. Lett.* **2018**, *7*, 7–11.
38. Kanchan, S.; Chaudhari, N.S. Signcrypting the Group Signature with Non-transitive Proxy Re-encryption in VANET. In *Recent Findings in Intelligent Computing Techniques*; Springer: Singapore, 2019; pp. 15–23.
39. Chen, F.L.; Wang, Z.H.; Hu, Y.M. A New Quantum Blind Signature Scheme with BB84-State. *Entropy* **2019**, *21*, 336.
40. Zhou, Y.; Li, Z.; Hu, F.; Li, F. Identity-Based Combined Public Key Schemes for Signature, Encryption, and Signcryption. In *Information Technology and Applied Mathematics*; Springer: Singapore, 2019; pp. 3–22.
41. Kittur, A.S.; Pais, A.R. A trust model based batch verification of digital signatures in IoT. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *11*, 313–327.
42. Gomez, A.I.; Gomez-Perez, D.; Renault, G. A probabilistic analysis on a lattice attack against DSA. *Des. Codes Cryptogr.* **2019**, *87*, 2469–2488.
43. Aldaya, A.C.; Brumley, B.B.; Sarmiento, A.J.C.; Sánchez-Solano, S. Memory Tampering Attack on Binary GCD Based Inversion Algorithms. *Int. J. Parallel Program.* **2018**, *47*, 621–640
44. Zuccherato, R. *Elliptic Curve Cryptography Support in Entrust*; Entrust Ltd.: Kanata, ON, Canada, 2000.
45. Tyagi, M.; Manoria, M.; Mishra, B. A Framework for Data Storage Security with Efficient Computing in Cloud. In *International Conference on Advanced Computing Networking and Informatics*; Springer: Singapore, 2019; pp. 109–116.
46. Louw, J.; Niezen, G.; Ramotsoela, T.; Abu-Mahfouz, A.M. A key distribution scheme using elliptic curve cryptography in wireless sensor networks. In Proceedings of the 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), Poitiers, France, 19–21 July 2016; pp. 1166–1170.
47. Kanda, G.; Antwi, A.O.; Ryoo, K. Hardware Architecture Design of AES Cryptosystem with 163-Bit Elliptic Curve. In *Advanced Multimedia and Ubiquitous Engineering*; Springer: Singapore, 2018; pp. 423–429.
48. Ferretti, L.; Marchetti, M.; Colajanni, M. Fog-based Secure Communications for Low-power IoT Devices. *ACM Trans. Internet Technol. (TOIT)* **2019**, *19*, 27.
49. Albalas, F.; Al-Soud, M.; Almomani, O.; Almomani, A. Security-aware CoAP Application Layer Protocol for the Internet of Things using Elliptic-Curve Cryptography. *Power* **2018**, *1333*, 151.
50. Khan, S.; Khan, R. Elgamal Elliptic Curve Based Secure Communication Architecture for Microgrids. *Energies* **2018**, *11*, 759.
51. Ay, A.U.; Mancillas-López, C.; Öztürk, E.; Rodríguez-Henríquez, F.; Savaş, E. Constant-time hardware computation of elliptic curve scalar multiplication around the 128 bit security level. *Microprocess. Microsyst.* **2018**, *62*, 79–90.
52. Liu, S.; Yao, H.; Wang, X.A. Fast elliptic curve scalar multiplication for resisting against SPA. *Int. J. Comput. Sci. Eng.* **2018**, *17*, 343–352.
53. Ezzouak, S.; Azizi, A. On the Efficiency of Scalar Multiplication on the Elliptic Curves. In *International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*; Springer: Cham, Switzerland, 2018; pp. 393–399.
54. Dawahdeh, Z.E.; Yaakob, S.N.; Othman, R.R.B. A New Modification for Menezes-Vanstone Elliptic Curve Cryptosystem. *J. Theor. Appl. Inf. Technol.* **2016**, *85*, 290.

55. Das, P.; Giri, C. An Efficient Method for text Encryption using Elliptic Curve Cryptography. In Proceedings of the 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 14–15 December 2018; pp. 96–101.
56. Keerthi, K.; Surendiran, B. Elliptic curve cryptography for secured text encryption. In Proceedings of the 2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Kollam, India, 20–21 April 2017; pp. 1–5.
57. Bh, P.; Chandravathi, D.; Roja, P.P. Encoding and decoding of a message in the implementation of Elliptic Curve cryptography using Koblitz's method. *Int. J. Comput. Sci. Eng.* **2010**, *2*, 1904–1907.
58. 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.
59. Muthukuru, J.; Sathyanarayana, B. Fixed and variable Size Text Based Message Mapping Technique using ECC. *Glob. J. Comput. Sci. Technol.* **2012**. Available online: https://computerresearch.org/index.php/computer/article/view/448 (accesded on 15 October 2020).
60. Barman, P.; Saha, B. DNA Encoded Elliptic Curve Cryptography System for IoT Security. *Int. J. Comput. Intell.* **2019**, *2*, 7.
61. Duarah, D.; Uma, V. Securing IoT Using Machine Learning and Elliptic Curve Cryptography. In *International Conference on Emerging Current Trends in Computing and Expert Technology*; Springer: Cham, Switzerland, 2019; pp. 482–491.
62. Joglekar, J.; Bhutani, S.; Patel, N.; Soman, P. Lightweight Elliptical Curve Cryptography (ECC) for Data Integrity and User Authentication in Smart Transportation IoT System. In *International Conference on Sustainable Communication Networks and Application*; Springer: Cham, Switzerland, 2019; pp. 270–278.
63. Almajed, H.N.; Almogren, A.S. SE-Enc: A Secure and Efficient Encoding Scheme Using Elliptic Curve Cryptography. *IEEE Access* **2019**, *7*, 175865–175878.
64. King, B. Mapping an Arbitrary Message to an Elliptic Curve When Defined over GF (2^n). *IJ Netw. Secur.* **2009**, *8*, 169–176.
65. Harvey, D.; van der Hoeven, J. Faster integer multiplication using short lattice vectors. *Open Book Ser.* **2019**, *2*, 293–310.
66. Rahman, A.N.M.B. We Don't Need StringBuilder for Simple Concatenation—DZone Java 2019. Available online: https://dzone.com/articles/string-concatenation-performacne-improvement-in-ja (accessed on 15 October 2020).