Improved authenticated elliptic curve cryptography scheme for resource starve applications

Esau Taiwo Oladipupo[1], Oluwakemi Christiana Abikoye[2]
[1]Department of Computer Science, The Federal Polytechnic Bida, Bida, Niger State, Nigeria
[2]Department of Computer Science, University of Ilorin, Ilorin, Kwara State, Nigeria

Article Info

Article history:

Received May 19, 2022
Revised Oct 28, 2022
Accepted Nov 9, 2022

Keywords:

Authenticated encryption
Bit modification
Cipher block chaining mode
Columnar transposition
Elliptic curve cryptosystem
Verification

ABSTRACT

Elliptic curve cryptography (ECC) remains the best approach to asymmetric cryptography when it comes to securing communication among communication partners in low-computing devices such as wireless sensor networks (WSN) and the Internet of Things (IoT) due to its effectiveness in generating small keys with a strong encryption mechanism. The ECC cuts down on power use and improves device performance, so it can be used in a wide range of devices that don't have a lot of resources. However, most of the existing ECC implementations suffer from implementation flaws that make them vulnerable to cryptanalysis attacks. In this study, flaws in the existing implementation of ECC are identified. A new scheme where the identified flaws are remedied was developed. The results of the security analysis show that the new scheme is an indistinguishable authenticated adaptive chosen ciphertext attack (IND-CCA3), resistant to malleability and man-in-the-middle attacks (MIMA). The results of comparative security analysis show that the mapping scheme employed in the new scheme maps any blocks of plaintext to distinct points on an elliptic curve, which makes it resistant to all attacks that the existing schemes are vulnerable to without having a negative effect on its encryption and decryption time, throughput, or power consumption.

Corresponding Author:

Esau Taiwo Oladipupo
Department of Computer Science, The Federal Polytechnic Bida
Nigeria
Email: taiwotheophilus@gmail.com, oladipupo.esau@fedpolybida.edu.ng

1. INTRODUCTION

Elliptic-curve cryptography (ECC) has been described as the most important tool in modern cryptography [1]. This applauding statement was due to the fact that the world is increasingly moving towards the use of resource-constrained applications where computational speed, storage, and bandwidth are limited [2]. With ECC, public key encryption, digital signatures, non-interactive key exchange, and a host of other security features are possible. ECC offers these security features with high speed, small space consumption, and bandwidth savings. These outstanding characteristics make ECC suitable for security applications in IoT devices [3]. These characteristics also explain why ECC is becoming more popular for security purposes [4] than its counterpart public key cryptography algorithms such as Rivest, Shamir Addleman (RSA), Diffie Helman (DH), and digital signature algorithm (DSA), particularly on resource-constrained devices such as wireless sensor networks (WSN), radio frequency identification (RFID), and the Internet of Things (IoT) [5]. ECC can use elliptic curves to implement cryptography algorithms based on discrete logarithm problems and ElGamal algorithms in an efficient way [6], [7].

Elliptic curve random generator has been defined by National Institute of Standard and Technology (NIST) as a method of grouping random digits based on curves [8]. Unlike other cryptosystems that encrypt and decrypt messages directly, be it text or image, elliptic curve cryptosystem (ECC) is capable of encrypting and decrypting only points on the elliptic curves [7]. Since ECC can only encrypt/decrypt point on an elliptic curve, cryptographers who use ECC must find a way of converting the message to be encrypted or decrypted to points on an elliptic curve. This process of converting message to point on an elliptic curve is known as mapping. Moreover, to retrieve the actual message when a point on the elliptic curve is decrypted, it requires that the point be converted back to messages. The process of converting points on an elliptic curve to message is called reverse mapping [6].

The major problem of the existing implementation of ECC such as [7], [9]–[11], is that, many of them failed to adhere to the guidelines given by [6] for a good message mapping and reverse mapping [12]. Also noticed that most of the existing systems such as [13]–[15], and [16] that claimed to have used ECC for securing plaintext failed to give the detail of how the plaintext were encoded into numerical values for use in ECC's mapping phase. In addition, the authenticated ECC encryption scheme proposed by [12] has security flaws which make the scheme vulnerable to cryptanalysis attacks. In this paper, security flaws in [12] are identified. A new scheme which removes the flaws is proposed. Security, comparative security, and comparative performance analysis of the proposed scheme are carried out on the proposed system.

The rest of this paper is organized: section 2 gives the overview of ECC. In section 3, review of the related works is dealt with. The methodology where the details of how identified problem was solved is given in section 4. Results and discussions on the results are given in section 5 while conclusion and recommendation are given in section 6.

2. OVERVIEW OF ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

The idea of using elliptic curve in cryptography was first introduced by [17] in 1987. Nowadays, ECC is widely applied for securing data on devices in resource constrained environment such as IoT and WSN devices [12]. This suitability of ECC in resource starve devices stems from the fact that ECC provides equal security for lesser key bit size than RSA [18] as shown by the comparison of the key sizes in Table 1. As a result, ECC supports low computation device capabilities, enabling them to perform more effectively [4].

Table 1. Comparison of ECC and RSA key sizes [3]

Key Size of ECC (Bits)	Key size of RSA (Bits)	Ratio (Bits)
106	512	1: 4
132	768	1:5
160	1024	1:6
224	2048	1:9
256	3072	1:12
384	7680	1:20
512	16380	1:30

ECC is based on the algebraic structure of elliptic curves over finite fields and the difficulty in solving the elliptic curve discrete logarithm problem (ECDLP). ECDLP deals with the problem of calculating the number of steps or hops it takes to move from one point to another point on the elliptic curve [19]. In mathematics, elliptic curves are described by (1)

$$A\,x^3 + B\,x^2\,y + C\,x\,y^2 + D\,y^3 + E\,x^2 + F\,x\,y + G\,y^2 + H\,x + I\,y + J = 0, \tag{1}$$

While the type of elliptic curve that is being used in cryptography is a simplified form (Weierstras form), which is defined by (2):

$$y^2 = x^3 + ax + b \tag{2}$$

The type of elliptic curves used by ECC are those elliptic curves in which the variables and coefficients are restricted to elements of a finite field. The two families of elliptic curves defined for use in cryptography are: prime curves defined over odd prime field $\boldsymbol{F_p}$ (where the field size p >3) and binary curves defined over Galois field $\mathrm{GF}(2^{\mathrm{m}})$ (where the field size $\mathrm{p} = 2^{\mathrm{m}}$) [7]. The operations on elliptical curves in cryptography are point addition, point multiplication and point doubling [20]–[22]. The pictorial representation of point addition where two distinct points P and Q are added together is shown in Figure 1(a). Figure 1(b) shows the pictorial representation of point doubling. Point doubling occurs when two points P and Q where P= Q are to be added together. The addition of two points P and Q where P= Q= 0 isi infinity as depicted in Figure 1(c). The addition of two mirror point ie P= -Q is infinity as shown in Figure 1(d). Equation (3) holds for all instances of (a) – (d) in Figure 1.

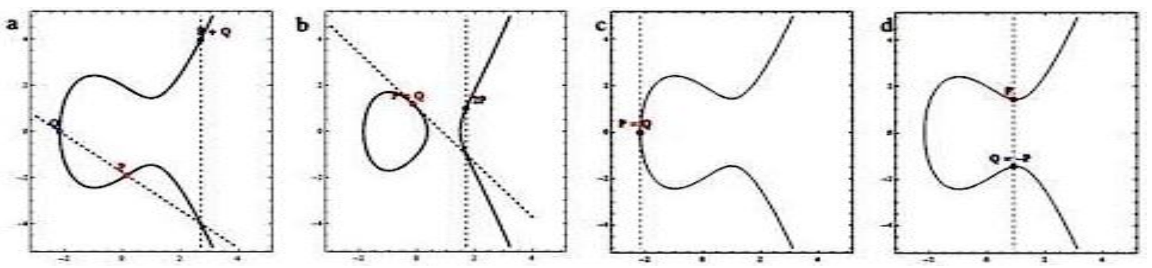


Figure 1. Pictorial representation of operations in ECC [23]; (a) Point addition; (b) Point doubling; (c) Point at infinity when y coordinates are both 0; and (d) Point at infinity when the coordinates are mirror image of each other

$$R = P + Q = \begin{cases} \infty, \ if x_1 = x_2 \ and y_1 = -y_2 \ OR \ x_1 = x_2 \ and \ y_1 = y_2 = 0 \\ P, \ if \ Q = \infty \\ Q, \ if \ P = \infty \\ (x_3, y_3), \ \ otherwise \end{cases} \tag{3}$$

where

$$x_3 = \begin{cases} \lambda^2 - x_1 - x_2, & if \ P \neq \pm Q \\ \lambda^2 - 2x_1, & if \ P = Q \end{cases}$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & if \ P \neq \pm Q \\ \dfrac{3{x_1}^2 + a}{2y_1}, & if \ P = Q \end{cases}$$

2.1. Scalar multiplication

Let P be a random point on the elliptic curve. The operation of multiplication over P is done by the repetitive addition. To achieve encryption and decryption using ECC, k|P| plays a vital role as in exponentiation operation. k[P]= P + P + P +⋯+P times i.e, 5P = P+P+P+P+P (additions). Point doubling can be used to reduce the number of additions.
P+P=2P(Doubling)
2P+2P=4P(Doubling)
4P+P=5P(addition**)**

2.2. Domain parameters for elliptic curves cryptosystem and their derivations

To use ECC, all parties involved have to agree on all basic elements concerning the elliptic curve E being used. In general the domain parameters for ECC algorithm is a sextuple given by (p,a,b,G,n,h) where

– p is prime that specifies the size of the finite field.
– a and b are the coefficients of the elliptic curve (2) - $\boldsymbol{y^2 = x^3 + ax + b}$.
– G is the base point that generates the subgroup of the elliptic curve whose order is a large prime.
– n is the order (number of points) of the subgroup. The order n of G is the smallest integer n such that nG = 0, n is a large prime.
– h is the cofactor of the subgroup which is the ratio $\frac{|E|}{|E_p|} = \frac{\boldsymbol{order\ of\ elliptic\ curve\ E}}{\boldsymbol{order\ of\ elliptic\ curve}\ \textbf{defined over prime field}\ \boldsymbol{E_p}}$. h should be small $\boldsymbol{h \leq 4}$, preferably, h =1

When two communicating parties are to communicate, each of them must have private and public keys. Each retain their private key while the public is made available to the public. Private key is generated by randomly selecting an integer from [1..n-1]. Public key is obtained by multiplying the generated private key Ka and point G with coordinates (x,y) on the elliptic curve together. The two communicating parties can therefore generate a shared secret key (SSK). For example, if Ks is the private key of the sender and Kr is the private key of the recipient. The public key of the sender will be spub = Ks *G and the public key of the recipient will be rpub = Kr *G. The sender and recipient can generate SSK as: sender: SSK = Ks * (Kr*G) and recipient: SSK = Kr * (Ks *G).

3. REVIEW OF RELATED WORKS

A lot of research works which exploited the strength of ECC to achieve various tasks of asymmetric cryptography like authentication, digital signature, key agreement and encryption had been carried out by various researchers in the past but gap has been found in the literature. An encryption scheme similar to Diffie-Hellman key exchange protocol but faster by around 20 percent was proposed by [24]. A text-based cryptography where ECC was used for transforming the message into ASCII values before mapping the transformed message to affine point on elliptic curve through point addition of the ASCII value times the Generator was proposed by [25]. The method adopted by some authors such as [26]–[28] where ASCII table was used to obtain the decimal value of each character of the message and then mapped each of the obtained decimal value directly onto point on elliptic curve are not only vulnerable to chosen plaintext attack (CPA) but also waste bandwidth during transmission of the message from sender to receiver. Other schemes that introduced manipulation of the ASCII table by multiplying it by a secure number that is agreed on by both parties [11] are vulnerable to man-in-the-middle attack (MIMA) because 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. Again, the approach wastes bandwidth during transmission of the message from sender to receiver because each character is represented by a point on the elliptic curve.

In [6], it is presented a message mapping and reverse mapping scheme that is based on grouping a fixed number of characters of the message before mapping. The ASCII value of each character in each group is converted to a binary value of 8 bits. These binary forms of each character in the group are concatenated together, and an integer value of the concatenated binary string is obtained and mapped to a point on the elliptic curve. In order to prevent non-mapping results, the authors proposed padding each set of characters to 8 bits to add one bit each time the mapping failed to find a corresponding y value. Although this method saves bandwidth, the approach is vulnerable to CPA when the set of characters is equal. The author also employed the use of El-Gama for encryption of the mapped point, which requires that for each mapped point, two points on the EC form encrypted points. This encryption scheme is not efficient enough, as the transmission of two encrypted points per block requires more bandwidth.

In an image encryption scheme using ECC proposed by [23], the encryption computation was reduced to pixel grouping into a single integer, where the number of pixels in each group is based on the ECC key size; and mapping of the grouped pixel to one large integer, which is then mapped to the elliptic curve. The performance of this scheme depends on the number of pixels that exist in one group. Significantly, a large number of pixels in one group decreases the computation overhead and, in this way, increases the performance. However, the number of pixels in one group depends on the key size, where large keys increase the pixel count. While this may be true, incrementing the key size leads to an increase in the encryption and decryption computation and storage overhead. As a result, the performance is affected in both cases.

In [7] the use of block chaining operation was introduced into a mapping scheme in ECC. In this approach, the plaintexts to be mapped are divided into fixed blocks. Exclusive OR (XOR) operation is carried out between the first block and an initialization vector IV, the result of the first XORed value is used in the second XOR operation, and the process continues until all blocks are treated. Although the approach maps the message to distinct points on an elliptic curve, but it is vulnerable to CPA and CCA when the plaintext is divided into a set of blocks and all blocks are the same (e.g., the plain text is a repeated character i.e. blocks B1, B2, …, Bn, where B1= B2=…=Bn). The first XOR operation produces B1' = InV $\oplus$ B1, the second XOR operation results in the value of the InV as InV $\oplus$ B1' $\oplus$ B2 = InV $\oplus$ B1$\oplus$ B2 = InV $\oplus$ B1$\oplus$ B1= InV. Therefore, the result of the XOR operation produces the sequence B'_1 , InV, B'_1, InV, ... Moreover, the author is silent about how the initialization vector is to be securely shared among the communicating parties. The need to securely share Inv among the communicating parties makes the scheme vulnerable to MIMA.

The authenticated encryption (AE) scheme proposed by [12] was proved to be secure against several encryption attacks, such as CPA, CCA, and malleability attacks. A proof of the resistance of the proposed scheme against specific encryption attacks and performance evaluation were carried out. The results of the analysis and performance evaluation show that the proposed scheme outperforms the security of other schemes and maintains the same computational overhead. However, taking a critical look at the mapping scheme shown in Figure 2.

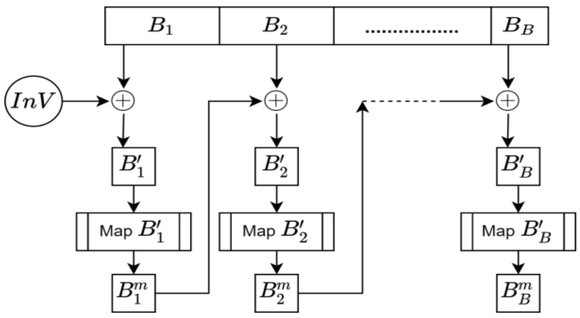


Figure 2. Existing technique of resisting encryption attacks using cipher block chaining (CBC) [12]

As presented by the author, the scheme failed to take into consideration a situation where an attacker tests the scheme with the plaintext that is divided into a set of blocks and all blocks are the same (e.g., the plain text is a repeated character). For instance, the vulnerability of the scheme stems from special cases when the plaintext blocks are equal such that $B_1 = B_2 = B_3 = \cdots = B_n$. It often happens that the value send to the maping function is not altered. This means that the returned value from mapping is still the same value that was sent to the mapping function. This returned mapped value is then XORed with the next block B_2which happen to be the same value as B_1 in such a case the following happen:

$$
\begin{aligned}
&B'_1 = InV \oplus B_1 \longrightarrow map(B'_1) \longrightarrow B^m{}_1\\
&B'_2 = B^m{}_1 \oplus B_2 \longrightarrow map(B'_2) \longrightarrow B^m{}_2\\
&B'_3 = B^m{}_2 \oplus B_3 \longrightarrow map(B'_3) \longrightarrow B^m{}_3\\
&B'_4 = B^m{}_3 \oplus B_4 \longrightarrow map(B'_4) \longrightarrow B^m{}_4\\
&\ but\ B'_1 = B^m{}_1\\
&B'_1 = InV \oplus B_1 = \ B^m{}_1\\
&B'_2 = InV \oplus B_1 \oplus B_2 = \ B^m{}_2, \qquad but\ B_1 = B_2\\
&B'_2 = InV \oplus B_1 \oplus B_1 = \ B^m{}_2,\\
&\therefore B'_2 = InV = \ B^m{}_2\\
&B'_3 = B^m{}_2 \oplus B_3 = \ B^m{}_3, \qquad but\ B_1 = B_2 = B_3\ and\ InV = \ B^m{}_2\\
&\therefore B'_3 = InV \oplus B_1 = \ B^m{}_3\\
&\therefore B'_3 = B'_1 = \ B^m{}_3 = B^m{}_1\\
&B'_4 = B'_1 \oplus B_4 = B^m{}_4\\
&B'_4 = InV \oplus B_1 \oplus B_1 = B^m{}_4\\
&B'_4 = InV = \ B^m{}_2 = B^m{}_4
\end{aligned}
$$

This sequence continues as $B'_1 = B'_3 = B'_5 = B'_7 = \cdots$ and $B'_2 = B'_4 = B'_6 = B'_8 = \cdots$ so also $B^m{}_1 = B^m{}_3 = B^m{}_5 = B^m{}_7 = \cdots$ and $B^m{}_2 = B^m{}_4 = B^m{}_6 = B^m{}_8$=…

Hence, under this condition, the scheme is vulnerable to CPA and chosen ciphertext attack (CCA). Again, CBC applied in the scheme uses an initialization vector that must be generated at random. This initialization vector must be harmonized between the transmitting and receiving correspondent for correct decryption. There must be a secure way of doing this if the security of the data is not going to be broken. This makes the technique to also vulnerable to MIMA.

4. METHOD

Obviously, from the reviewed literature, it can be established that existing implementation of ECC are suffering from different cryptanalysis attacks. The enhancement done by [12] suffers from CPA and CCA because of the condition analyzed in the literature review. There are two identified deficiencies; i) the vulnerability to MIMA, and ii) the vulnerability to CPA and CCA.

The first problem is solved by making use of the SSK for the production of the seed that can be used for the production of initialization vector. Since both the sender and recipient can generate it independently without having to transmit anything using any channel, the problem of MIMA is solved. The second problem is solved by XORing InV with B1, map the result to point on elliptic curve and encrypt the mapped point to obtain $\boldsymbol{B^e}_\mathbf{1}$. To encrypt B2, the x coordinate of the encrypted point $\boldsymbol{B^e}_\mathbf{1}$ is XORed with B2, the result is mapped to point on elliptic curve and the mapped point is encrypted to obtain $\boldsymbol{B^e}_\mathbf{2}$. This process is repeated until all the blocks are mapped and encrypted. This process works because the encrypted B1 disorganizes the pattern that make XOR vulnerable to the above analysis such that when it is XORed with B2 an entirely new value is generated and no XOR function can be *used* to trace it back to InV. Figure 3 illustrates how these feet are achieved in the improved authenticated elliptic curve cryptosystem (IAECC) scheme. By this process, the vulnerability to cryptanalysis attack is greatly reduced as cryptanalysts have to solve ECDLP before any attack can be lunched.

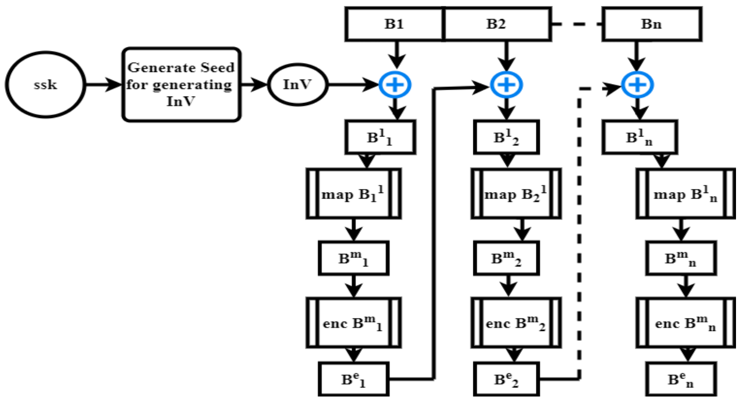


Figure 3. Securing blocks against encryption attacks using CBC in IAECC

The encryption algorithms for the IAECC is given in Algorithm 1. The encryption process requires the plaintext, the elliptic curve name, the sender's private key and the recipient public key.

Algorithm 1: IECC Encryption Process

```
Module encryption (plaintext,curvename,rpublickey,sprivatekey)
Input: Plaintext: the text to be encrypted, rpublickey: publickey key of the recipient
           Sprivatekey: private key of the sender, N: number of bits to be reserved for
mapping block to
           Points, curvename: standard elliptic curve name sent from recipient
Output: Cm: array of Encrypted points on the elliptic curve
    1.  START
    2.  curve = Obtain elliptic curve using curvename
    3.  nbits = obtain curve order and convert it to binary form
    4.  COMPUTE blocksize = $\boldsymbol{FLOOR}\left(\frac{nbits-8}{8}\right)$
    5.  COMPUTE nblocks = $\frac{LENGTH(plaintext)}{blocksize}$
    6.  COMPUTE ssk = sprivatekey * rpublickey
    7.  COMPUTE diff = ABS (ssk.x - ssk.y)
    8.  COMPUTE encseed = MOD (diff, $2^{32}$)
    9.  COMPUTE Inv = RANDOM (encseed, blocksize)
    10. LET j = 1
    11. INITIALIZE Array cm
    12. FOR i = 1 TO nblocks (taken blocksize characters at a time)
         i.    LET block = plaintext (j to j + blocksize -1)
        ii.    LET j = j + blocksize
       iii.    block = CONVERT block to binary form
        iv.    Inv = CONVERT Inv to binary form
         v.    block = XOR (block, Inv)
        vi.    LET X = CONVERT block to integer
       vii.    LET d = X * 16
      viii.    LET Pm = CALL blockmapping(d, curvename)
        ix.    LET C2 = Pm + ssk //C2 is a point (x,y) on the elliptic curve
         x.    Cm[i] = C2
        xi.    Inv = CONVERT C2.x to binary form
       xii.    Inv = Inv [1..blocksize]
        END FOR
    13. signedCm = CALL Appendsignature (Cm)
STOP
```

Algorithm 1 makes use of mapping function whose algorithm is described in Algorithm 2. This mapping function modifies the numerical value X of the block of plaintext to ensure that the point (X,Y) satisfies the equation $\boldsymbol{Y^2} = (X^3 + aX + b)\text{MOD P}$. When all the blocks that were mapped unto points on the elliptic curve are encrypted the encrypted points are signed by the sender using the module Appendsignature whose description is given in Algorithm 5. Both the signature and the signed encrypted points are transmitted to the recipient.

Algorithm 2: Algorithm for mapping a block of characters to a point on the Elliptic curve

```
Module mapoint = blockmapping (X, curvename)
INPUT: X an integer representing a block of ASCII characters.
OUTPUT: A point (x,y) on the Elliptic curve curve(a,b) corresponding to the Message block
   1. START
   2. SET solution = FALSE
        a. WHILE solution = FALSE
              i.  FIND Y from the equation $Y^2 = (X^3 + aX + b)\text{MOD P}$
             ii.  IF Y does not have solution THEN X = X + 1
                  ELSE solution = TRUE
                  ENDIF
           ENDWHILE
   3. RETURN point with coordinate (X,Y)
STOP
```

4.1. IAECC decryption process

The decryption process is just the exact reverse of the encryption process. The first operation to be performed is the verification of the signature by the recipient. The description of the signature verification algorithm is given in Algorithm 3. If the signature is verified to be valid, then the decryption process will be carried out otherwise invalid error message will be displayed and the program come to a halt. Figure 4 illustrates how decoding and conversion of the decrypted points into a plain text is achieved and Algorithm 4 gives the description of the decryption process in the proposed IAECC.

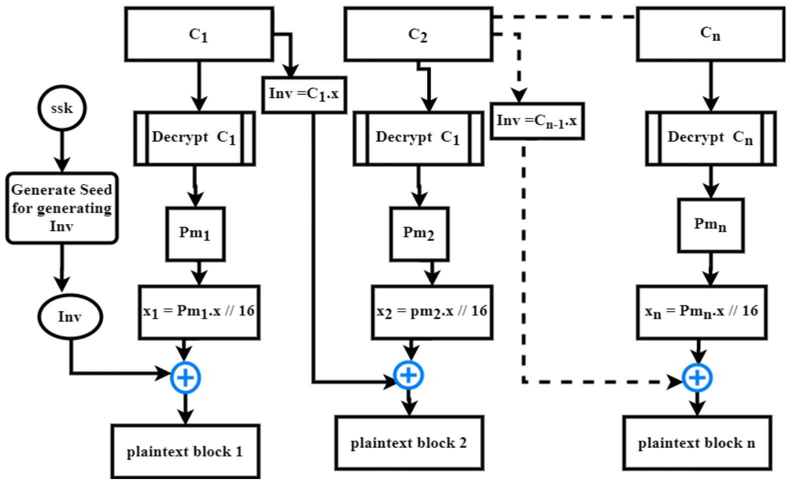


Figure 4. Reverse mapping scheme in proposed IAECC

Algorithm 4: Decryption process in proposed IECC

```
Module Message = Decryption(cm,curvename,spub, rpriv)
Input: cm: List of encrypted points, curvename: name of the curve used, spub: Sender's
public key,
            rpriv: recipient's private key, signature: Sender's signature of the encrypted
message
Output: Plaintext: decrypted text obtained from Cm
    1. START
    2. GET Cm and signature
    3. Msent = CONCATENATE(Cm, Signature,spub)
    4. Cmvalidity = CALL Verifysignature(Msent,signature, PA))
    5. IF Cmvalidity = TRUE THEN
         i.    GET order N of the curve using the curvename
        ii.    GET rpriv
       iii.    COMPUTE blocksize = $\boldsymbol{FLOOR}\left(\frac{nbits-N}{N}\right)$
```

```
     iv.    COMPUTE nblocks = LENGTH (cm)
      v.     COMPUTE ssk  =  spub * rpriv //shared secrete
     vi.    COMPUTE  diff = ABS(ssk.x  - ssk.y)
    vii.    COMPUTE encseed = MOD(diff, 2^32)
   viii.    COMPUTE Inv = RANDOM(encseed, blocksize)
     ix.    Message = EMPTY STRING
      x.    FOR  i = 1  to nblocks
                i.  C =  Cm[i]
               ii.  LET ivs = C.x
              iii.  Pm = C - ssk
               iv.  x = Pm.x//16
                v.  block = XOR(x, Inv)
               vi.  Message = CONCATENATE(Message, block)
              vii.  Inv = ivs
         END FOR
    ELSE
                  DISPLAY (“Error message”)
6.  STOP
```

4.2. Authentication of IAECC encryption and decryption processes

If the message being transmitted between communicating parties will not compromise its confidentiality, security and integrity, authentication encryption scheme must be applied. In authentication encryption scheme, confidentiality is maintained by the encryption and decryption phases of the scheme. In order to maintain the integrity of the message being transmitted in authentication encryption scheme, the sender must sign the message using his/her private key and the recipient should verify the received message using the public key of the sender before decryption.

4.2.1. IAECC signing process

In the developed IAECC, the sender signs the message Msent using his/her private key ds, which relies on elliptic curve digital signature algorithm (ECDSA), before sending it to the recipient through public channel. Msent consists of a set of tuples, which contains points on the elliptic curve used in the cryptosystem. Each point C_i represents the encryption point for each block of the message being transmitted. The signing process is illustrated in Figure 5 and Algorithm 5 gives the description of the algorithm for signing process.

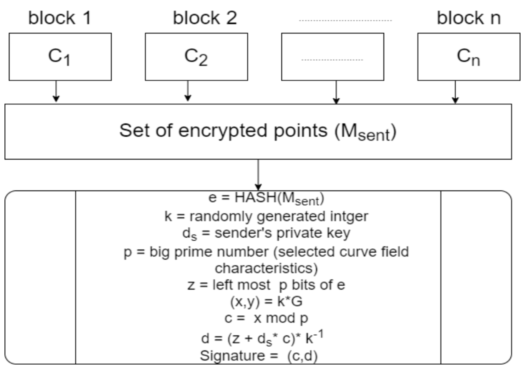
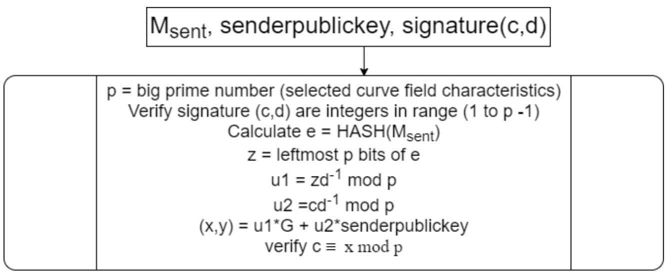

block 1
block 2
block n
C_1
C_2
C_n
Set of encrypted points (M_{sent})
e = HASH(M_{sent})
k = randomly generated intger
d_s = sender's private key
p = big prime number (selected curve field characteristics)
z = left most p bits of e
(x,y) = k*G
c = x mod p
d = (z + d_s* c)* k^{-1}
Signature = (c,d)

Figure 5. Ensuring the Integrity of message by sender’s signature

Algorithm 5: Process of Signing the message
```
Module appendsignature(Msent)
Input: The Message Msent
Output: Signature pairs (c,d)
   1.  calculate e = HASH($M_{sent}$);
   2.  calculate z = le f t most p bits o f e
   3.  select random value k
   4.  calculate (x, y) = k × G
   5.  calculate c = x mod p where r ≠ 0
   6.  If c = 0 go to 3
   7.  calculate d = (z + $d_s$ * r) k −1
   8.  If d = 0 go to 3
   9.  (c,d) ← ciphertext signature
   10. Stop.
```

4.2.2. IECC verification process

The receiver of a signed message verifies it using the sender's public key. The verification process is illustrated in Figure 6. Algorithm 6 outlines the steps involved in verifying the integrity of the received message using the sender's public key.



M_{sent}, senderpublickey, signature(c,d)
p = big prime number (selected curve field characteristics)
Verify signature (c,d) are integers in range (1 to p -1)
Calculate e = HASH(M_{sent})
z = leftmost p bits of e
u1 = zd^{-1} mod p
u2 = cd^{-1} mod p
(x,y) = u1*G + u2*senderpublickey
verify c ≡ x mod p

Figure 6. Recipient verifies the received message

Algorithm 6: Recipient Verification of a Signed message

```
Module Verifysignature(Msent,signature, senderpublickey)
Input: Msent, signature (c,d), senderpublickey
Output: Verified ciphertexts
 1.  Start
 2.  check (c,d) are integers ∈ 1, 2, 3, ..., p − 1)
 3.  calculate e = HASH(Msent)
 4.  calculate z = leftmost p bits of e
 5.  calculate u1 = $ed^{-1}$ mod p
 6.  calculate u2 = $rd^{-1}$ mod p
 7.  calculate (x, y) = u1 ∗ G + u2 ∗ senderpublickey
 8.  Verified ← c ≡ x mod p
Stop.
```

5. RESULTS AND DISCUSSIONS

The results of the security analysis, comparative security analysis and comparative performance analysis of IAECC with the existing ECC implementations are discussed in this section. The results of the security analysis of IAECC are based on indistinguishability, malleability and replay attacks. Comparative security analysis with the existing ECC scheme is based on mapping scheme used by each scheme and resistance of each scheme to different security attacks. Encryption/decryptime, throughputs and energy consusmption are used as metrics for comparative performance analysis of IAECC with the existing ECC.

5.1. Security and performance analysis of IAECC

5.1.1. Security analysis based on indistinguishability

Security in terms of indistinguishability is normally presented as a game, where the cryptosystem is considered secure if no adversary can win the game with significantly greater probability than an adversary who must guess randomly. A cryptosystem is considered" secure in terms of in distinguishability" if no adversary A, given an encrypted form of a message randomly chosen from a two-element message space determined by the adversary, can identify the message choice with probability significantly better than that of random guessing (1/2). If any adversary can succeed in distinguishing the chosen ciphertext with a probability significantly greater than 1/2, then this adversary is considered to have an advantage in distinguishing the ciphertext, and the scheme is not considered secure in terms of in distinguishability. Various form of indistinguishability against different forms of attacks are denoted [29]: Indistinguishable under chosen plaintext attack (IND-CPA), Indistinguishable under chosen ciphertext attack (IND-CCA), Indistinguishable under non-adaptive chosen ciphertext attack (IND-CCA1), Indistinguishable under adaptive chosen ciphertext attack (IND-CCA2), and Indistinguishable authenticated adaptive chosen ciphertext attack (IND-CCA3) [30]. While IND=CPA can only provide guarantee against passive security attacks, IND-CCAs are capable of providing security guarantee against active security attacks [31]. Any cryptography scheme that is evaluated against IND-CCA3 means that it is also verified against IND-CCA2, IND-CCA1, and IND-CPA [4]. Therefore, IAECC is verified against IND-CCA3.

Formally, the IND-CCA3 advantage measure is defined:

Given adversary A, and an oracle encryption scheme$\prod = (\boldsymbol{\kappa}, \boldsymbol{\varepsilon}, \boldsymbol{D})$, the advantage measure of IND-CCA3 is defined by (4):

$$Adv_{\prod}^{IND-CCA3}(A) = Pr\left[K \overset{\$}{\leftarrow} \varkappa\colon\ A^{\mathcal{E}_K(.),D_K(.)} \Rightarrow 1\right] - Pr\left[A^{\mathcal{E}_K(\$|.|.),\perp(.)} \Rightarrow 1\right] \tag{4}$$

Where $Adv_{\prod}^{IND-CCA3}(A)$ is a measure of adversarial advantage of IND-CCA3, in [32] showed that every encryption scheme meeting the IND-CCA3 notion also meets both the IND-CPA notion and the AUTH notion and vice versa. The AUTH notion guarantee the security of ensuring the integrity of ciphertext. Formally, the measure of adversarial advantage of IND-CPA and AUTH are defined by (5) and (6) respectively.

$$\mathrm{Adv}_{\prod}^{\mathrm{IND-CPA}}(\mathrm{A}) = \mathrm{Pr}\left[\mathrm{K} \overset{\$}{\leftarrow} \varkappa\colon\ \mathrm{A}^{\mathcal{E}_\mathrm{K}(.)} \Rightarrow 1\right] - \mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(\$|.|.)} \Rightarrow 1\right] \tag{5}$$

$$\mathrm{Adv}_{\prod}^{\mathrm{Auth}}(\mathrm{A}) = \mathrm{Pr}\left[\mathrm{K} \overset{\$}{\leftarrow} \varkappa\colon\ \mathrm{A}^{\mathcal{E}_\mathrm{K}(.)}\ \mathrm{forges}\right] \tag{6}$$

Equation (6) can be recast as an experiment in which the adversary, given an $\mathcal{E}_{\boldsymbol{K}}\ (\cdot)$ oracle, attempts to distinguish between a real decryption oracle and a bogus decryption oracle that returns Invalid on every input ciphertext. This recast can be represented by the (7)

$$\mathrm{Adv}_{\prod}^{\mathrm{auth}*}(\mathrm{A}) = \mathrm{Pr}\left[\mathrm{K} \overset{\$}{\leftarrow} \varkappa\colon\ \mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\mathrm{D}_\mathrm{K}(.)} \Rightarrow 1\right] - \mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\perp(.)} \Rightarrow 1\right] \tag{7}$$

Equation (4) can be rewritten in terms of (5) and (7):

$$\begin{aligned}\mathrm{Adv}_{\prod}^{\mathrm{IND-CCA3}}(\mathrm{A}) &= \mathrm{Pr}\left[\mathrm{K} \overset{\$}{\leftarrow} \varkappa\colon\ \mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\mathrm{D}_\mathrm{K}(.)} \Rightarrow 1\right] - \mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(\$|.|.),\perp(.)} \Rightarrow 1\ \right] \\ &= \left(\mathrm{Pr}\left[\mathrm{K} \overset{\$}{\leftarrow} \varkappa\colon\ \mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\mathrm{D}_\mathrm{K}(.)} \Rightarrow 1\right] - \mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\perp(.)} \Rightarrow 1\right]\right) + \left(\mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(.),\perp(.)} \Rightarrow 1\right] - \mathrm{Pr}\left[\mathrm{A}^{\mathcal{E}_\mathrm{K}(\$|.|.),\perp(.)} \Rightarrow 1\right]\right)\end{aligned}$$

Let B1 and B2 be the two adversaries capable of AUTH and IND-CPA respectively, then the measure of adversarial advantage $Adv_{\prod}^{IND-CCA3}(A)$ can be measured in terms of the adversarial advantage of B1 and B2.

$$Adv_{\prod}^{IND-CCA3}(A) = \ Adv_{\prod}^{auth*}(B1) + Adv_{\prod}^{IND-CPA}(B2) \tag{8}$$

IAECC is evaluated against $IND - CCA3$ in this research. A cryptosystem is said to be secure under $IND - CCA3$ if the adversarial advantage $Adv_{\prod}^{IND-CCA3}(A)$ is negligible. The game for IND-CPA and IND-CCA is simulated in order to verify whether or not IAECC is IND-CCA3.

a. IND-CPA: the game 0 in Table 2 gives distinct value of encrypted points for each block. Table 3 and Table 4 show the results of the encryption of two blocks that contains the same value. In each case it can be seen that. each time the attacker submits the same messages $(\boldsymbol{m_{i,j}},\ \boldsymbol{m_{i,j}})$ IAECC encryption oracle returns $(\boldsymbol{c_{i,j}}\ \neq\ \ \boldsymbol{c_{i+1,j}})$. This shows that the attacker will find it very difficult to find out which ciphertext belongs to which plaintext even with the samples of plaintext/ciphertext pair. This is because the mapping between plaintext block and ciphertext block is not one-to-one mapping.
b. IND-CCA: Table 5 shows a block of plaintext m_0 and how it was encrypted using IAECC with appended signature. In Table 6, the encrypted point is modified by XORing the iv with a random value R to obtain a modified encrypted point. This modified encrypted point was then submitted for decryption but IAECC decryption returns an error message. This prevents the attacker from having the opportunity to compare modified point with the encrypted point in Table 5. Hence the attacker cannot win the game.

Table 1. IND-CPA game 0

	$m_{i,0}$ $m_{i,1}$: 33333333333333333333333344444444444444444444444		
i	$m_{i,b}$	b	$c_{i,b}$
1	3333333333333333333333	0	(1097748114407966986605873703748467691829460338770591897237, 4240406011164217225030410622972769495523064509111595973836)
2	4444444444444444444444	1	(2905408739093402878637059497371287570771625614334891806504, 3821259102381211389142430789426336915721906240279458918383)

Table 2. IND-CPA Game 1a

$m_{i,0}, m_{i,0}$:	3333333333333333333333333333333333333333333333	
$m_{i,b}$	b	$c_{i,b}$
3333333333333333333333	0	(1097748114407966986605873703748467691829460338770591897237 , 4240406011164217225030410622972769495523064509111595973836)
3333333333333333333333	1	(2449612076858759344596169117205506126915740111109049671270 , 1876668022142370300698035327836552245714157263410117886939)

Table 3. IND-CPA Game 1b

	$m_{i,1}, m_{i,1}$:	4444444444444444444444444444444444444444444444	
i	$m_{i,b}$	b	$c_{i,b}$
1	4444444444444444444444	0	(3482519432774353763529248619562031910947934849325919717209, 4550294802951404491094046444713769726958573788939094842781)
2	4444444444444444444444	1	(3006760824386382941774675144870981847120905949496660971403, 3773187468547889137107325909042525720566923649883216749471)

Table 4. IND-CCA Game 0

m_0	3333333333333333333333
$c_0, iv, sign$	0x2cc5066bb6eb20a4ca01f22882f225305fb9412b8d2c56950x0', ECDSASignature(c=3091638748216676367144710101552819383424206014947390681713, d=2672717718749855055751385032704713632596933926364840628833)

Table 5. IND-CCA Game 1

$c_0, iv⊕R, sign$	'0x2f07b82f948bff8cc7c7af2c96211b423584244720d7f0000x0', ECDSASignature(c=3091638748216676367144710101552819383424206014947390681713, d=2672717718749855055751385032704713632596933926364840628833)
$m_b⊕R \leftarrow dec(k, c_0, iv⊕R, sign)$	Error message: invalid ciphertext

The adversaries B1 B2 could not win IND-CPA and IND-CCA games respectively. i.e. $Adv_{\prod}^{IND-CPA}(B2) = 0$ and $Adv_{\prod}^{auth*}(B1) = 0$, this implies that $Adv_{\prod}^{IND-CCA3}(A) = Adv_{\prod}^{auth*} + Adv_{\prod}^{IND-CPA}(B2) \quad = 0 + 0 = 0$, hence, IAECC is IND-CCA3.

5.1.2. Security against malleability attack

An encryption algorithm is said to be malleable if an adversary can transform a cipher-text into another ciphertext which decrypts to a related plaintext. That is given an encryption of a plaintext m, it is possible to generate another ciphertext which decrypts to f(m), for a known function f, without necessarily knowing or learning m. Malleability attack is a type of cryptanalysis attack where modification of encrypted data result in modification to decrypted message. With this attack, an attacker can give misleading information to the recipient without the knowledge of the sender. A good cryptosystem should be non-malleable. Table 7 shows sample of output when a sample plaintext was encrypted using IAECC encryption module and an attempt is made to decrypt the modified encrypted data.

Table 6. Results of malleability attack test

plaintext	8888888888888888888888
Ciphertext	['0x8f85de8b5a6d730f222c74e9ef7b8189f03633f9bbd6c2f70x0', ECDSASignature(c=3091638748216676367144710101552819383424206014947390681713, d=621931775176316690818926789351489873483157764012435327223)]
Modified ciphertext	['0x53f91632bc6bcf268d444e70dc82847590fc036f5d3718f40x0', ECDSASignature(c=3091638748216676367144710101552819383424206014947390681713, d=621931775176316690818926789351489873483157764012435327223)]
Results of attempted decryption of modified ciphertext	invalid ciphertext

As can be seen from Table 7. IAECC gives an error message stating that the ciphertext is invalid. This result shows that IAECC is non-malleable.

5.1.3. Security against replay attack

A replay attack occurs when a cybercriminal snoops on a secure network communication, intercepts it, and then illegally delays or resends it to mislead the recipient into doing what the hacker wants. Figure 7(a) illustrates the adversary's capability of carrying out replay attack. To prevent the possibility of hacker's capability of carrying out replay attack in IAECC, both sender and receiver establish a completely random session key for each communication session. This means that no two sessions can use the same session key because session key will only be valid for one transaction and can't be used again. As the adversary need to establish new session key for resending the intercepted message and does not have access to new parameters which are needed for generating valid new session key only the old session key which is at the attacker's disposal becomes invalid. This invalid session key will make the signature invalid, hence the recipient will ignore the message. Figure 7(b) illustrates how IAECC prevents replay attack.

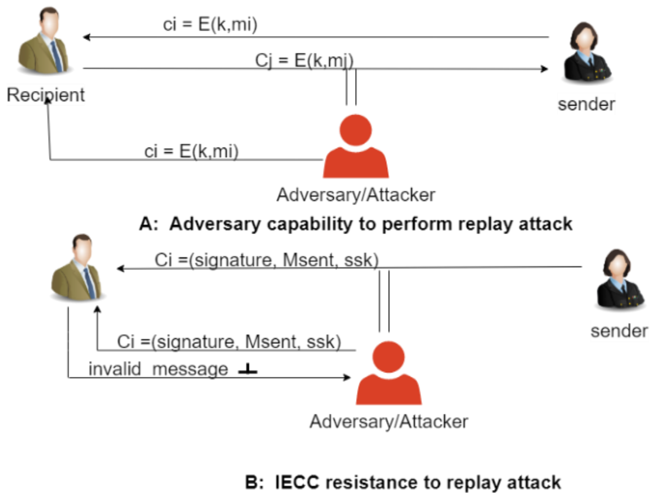

ci = E(k,mi)
Cj = E(k,mj)
Recipient
sender
ci = E(k,mi)
Adversary/Attacker
A: Adversary capability to perform replay attack
Ci =(signature, Msent, ssk)
sender
Ci =(signature, Msent, ssk)
invalid message ⊥
Adversary/Attacker
B: IECC resistance to replay attack

Figure 7. Security against replay attack (a) Adversary capability to perform replay attack and (b) IAECC resistance to replay attack

5.1.4. Comparative security analysis of IAECC with existing ECC schemes

A. Comparison based on mapping schemes

Table 8 shows the results of the experiment. In [6] scheme mapped all the four blocks to the same point. In [12] mapped the four blocks to two different points. The mapping scheme in the developed IAECC however mapped each of the four blocks to different point on the elliptic curve.

Table 8. Comparison of existing mapping schemes with mapping schemes used in IECC

Plaintext: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333		
[6]	[12]	IAECC
[[78463771692333509547947367790 0958302012794430558004314 09,85774 0959971287332615284443694443093 4097884101358628786 7],	[[23490935719613477959858806903515 9 25766546455338628348773 0, 20494837381314156694024169495562 32 11243309800590856921214 3],	[[20686706978545444595443045279 1228993428014594458238323123, 9920775885157922303737053531 73 608652754090981423936289006],
[784637716923335095479473677900 9 5830201279443055800431409, 48577409599712873326152844436 94 4430934097884101358628786 7],	**[26125541889626826888848470238373 1 67123837357122846345696 1, 16995380338504431436768488153722 68 696579123829363961918060**],	[19639389190147561121898299323807 014805129220684070310056 2, 38569271092547243772762024810842 2 3365079818205547885883432],
784637716923335095479473677900 9 5830201279443055800431409, 48577409599712873326152844436 94 4430934097884101358628786 7],	[23490935719613477959858806903515 9 25766546455338628348773 0, 20494837381314156694024169495562 32 11243309800590856921214 3],	[37544631019679213631330991209 6594836626453316948075710704, 20661500913629428628317664931 2 1507846165941885050229288074],
[784637716923335095479473677900 9 5830201279443055800431409, 48577409599712873326152844436 94 4430934097884101358628786 7]]	**[26125541889626826888848470238373 1 67123837357122846345696 1, 16995380338504431436768488153722 68 696579123829363961918060]]**	[**38263204739887138018468286236658 975052367301396740793904 3, 11780643845972330785368831943312 1 41149194581991385103892 43**]]

These results show that the mapping scheme in IAECC enables the scheme to be IND-CPA and IND-CCA which the existing systems are not.

B. Comparative security analysis of IAECC with existing ECC cryptosystem

Table 9 shows the results of the comparative security analysis of IAECC with the other existing ECC schemes. Comparison was carried out on the selected schemes based on whether the scheme is/is not IND-CCA3, resistance to malleability attack (MA), checking the integrity of the ciphertext (IT), offering authentication encryption (AE), or offering nonrepudiation (NR), is resistant to MIMA. Y and N represents YES, NO respectively. Yes, means that the scheme is offering the feature in question, NO means that the scheme is not offering the feature. The comparison in Table 9 shows that IAECC offers resistance to different cryptanalysis attacks as well as offering authentication encryption and non-repudiation.

Table 9. Security analysis comparison of proposed scheme and other schemes

ECC schemes	IND-CCA3	MA	IT	AE	NR	MIMA
[33]	N	N	N	N	N	N
[7]	N	N	N	N	N	N
[6]	N	N	N	N	N	N
[34]	N	N	N	N	N	N
[28]	N	N	N	N	N	N
[35]	N	N	N	N	N	N
[12]	N	Y	Y	Y	Y	Y
IAECC	Y	Y	Y	Y	Y	Y

5.1.5. Comparative performance analysis of IAECC

The two enhancements done to [12] that brought about the proposed IAECC are: the introduction of the generation of initialization vector (InV) and modification of the mapping scheme as earlier explained. The purpose of these enhancements are to solve the problem of how the initialization vector is to be securely shared among the communicating parties and also to improve resistance to encryption attacks. It is therefore necessary to check if these enhancements have negatively affected performance. Therefore, the performances of the developed IAECC scheme are compared with existing schemes using encryption/decryption time, throughput, and energy consumption during encryption and decryption processes as performance metrics.

Three schemes were selected for comparative performance analysis based on the above perfomance metrics. Each of the three chosen schemes is a representation of three different groups: [6] represents the group of schemes that do not apply Inv or any other methods to manipulate mapping points, [12] represents the groups that uses InV and apply CBC mode at mapping level while IAECC represents the group that uses InV and apply CBC at the end of encryption. An experiment was set up to measure the encryption/decryption time of MCC key security process using timer built into the IAECC cryptosystem application. Different plaintext sizes (in bytes) were encrypted and decrypted. When a particular is presented, the encryption and decryption processes are carried out 10 rounds. In each round, the time taken for the encryption/decryption to complete is measured. These 10 different measured time were added together and the average time was taken as the encryption/decryption time. This procedure was used for each of the three schemes that were selected for the performance comparison. For the calculation of encryption and decryption throughputs, the formulae in (9) and (10) were used.

$$\text{Encryption Throughput} = \frac{\text{Data size}}{\text{Time taken to encrypt the data}} \tag{9}$$

$$\text{Decryption Throughput} = \frac{\text{Data size}}{\text{Time taken to decrypt the data}} \tag{10}$$

In order to calculate the energy consumption by the system during encryption/decryption processes, the approximate average current consumption of the laptop used for the experiment when it is busy and the CPU voltage were both collected from manual of the hp laptop used in conducting the experiment. The average current consumption I is 100mA while the CPU voltage V_{cc} is 1.25v. The Energy consumption E in Joule (J) of the laptop used for the experiment during encryption/decryption of different key sizes were calculated using the formula: $\boldsymbol{E = V_{cc} \times I \times T}$, where T represents encryption/ decryption time. Twenty different data sizes were used, the data sizes in byte, the encryption time and decryption time obtained from the experiment as well as calculated encryption throughput, decryption throughput, power consumption during encryption and power consumption during decrption for each data size.

In order to ascertain the effects of changing the sizes of keys on the encryption/decryption time, throughput and energy consumption during encryption and decryption processes on the schemes under consideration, average encryption/decryption time, throughput and power consumption during encryption/decryption processes were calculated and compared.

Table 10 shows the recorded encryption and decryption time during the experiment when different key sizes were encrypted and decrypted using three different ECC schemes. Average encryption and decryption time is calculated as shown in Table 11. Table 12 shows the calculated encryption and decryption throughputs for the three ECC schemes under consideration. The energy consumption during encryption and decryption processes for each of the three ECC schemes are shown in Table 13. Although the corresponding encryption and decryption time in Table 10 appear to be the same for the three schemes for each data size, there is variation in the values when the number of decimal places is considered. This explains the reason why the average encryption/decryption time calculated are not the same for the three schemes under consideration.

Table 10. Encryption and decryption time of three ECC schemes with different data sizes

		Encryption and Decryption time (s)					
		Sengupta		Almajed		IECC	
S/N	Datasize(byte)	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
1	115	0.389078	0.483301	0.389078	0.483301	0.389078	0.483301
2	230	0.434338	0.518787	0.434338	0.518787	0.434338	0.518787
3	345	0.465585	0.559502	0.465585	0.559502	0.465585	0.559502
4	460	0.519219	0.602797	0.519219	0.602797	0.519219	0.602797
5	575	0.560923	0.626595	0.560923	0.626595	0.560923	0.626595
6	690	0.606371	0.673110	0.606371	0.673110	0.606371	0.673110
7	805	0.646975	0.707926	0.646975	0.707926	0.646975	0.707926
8	920	0.689171	0.740678	0.689171	0.740678	0.689171	0.740678
9	1035	0.724982	0.779279	0.724982	0.779279	0.724982	0.779279
10	1150	0.766760	0.823967	0.766760	0.823967	0.766760	0.823967
11	1265	0.810980	0.853082	0.810980	0.853082	0.810980	0.853082
12	1380	0.856045	0.887785	0.856045	0.887785	0.856045	0.887785
13	1495	0.903110	0.926059	0.903110	0.926059	0.903110	0.926059
14	1610	0.939079	0.965609	0.939079	0.965609	0.939079	0.965609
15	1725	0.990609	0.998431	0.990609	0.998431	0.990609	0.998431
16	1840	1.028811	1.033914	1.028811	1.033914	1.028811	1.033914
17	1955	1.080697	1.078274	1.080697	1.078274	1.080697	1.078274
18	2070	1.167239	1.161422	1.167239	1.161422	1.167239	1.161422
19	2185	1.279685	1.268402	1.279685	1.268402	1.279685	1.268402
20	2300	1.226619	1.214720	1.226619	1.214720	1.226619	1.214720

The bar chart in Figure 8 represents the information in Table 11. The bar chart shows that the encryption and decryption time for the three ECC schemes are approximately the same. This result implies that the modification done in IAECC does not make its encryption and decryption time higher than that of the existing ECC schemes.

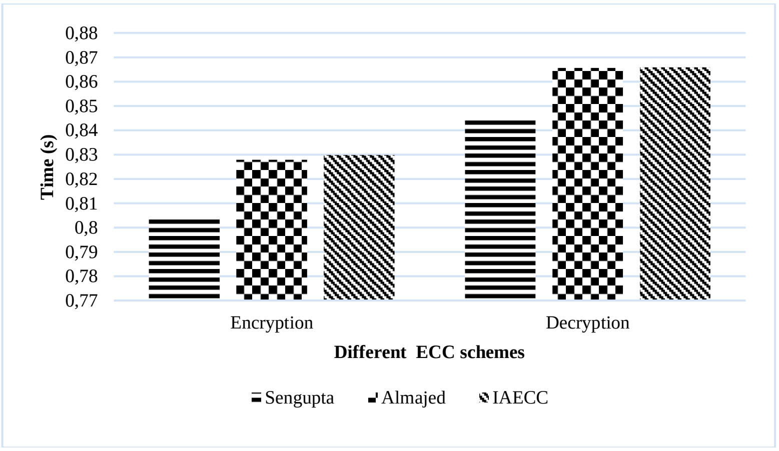


Figure 8. Comparative analysis of encryption/decryption time of ECC schemes

Table 11. Average encryption and decryption time of three ECC schemes

		Average Encryption and Decryption Time (s)		
S/N	scheme	Sengupta	Almajed	IAECC
1	Encryption	0.804314	0.827877	0.829769
2	Decryption	0.845182	0.865534	0.865895

The bar chart in Figure 9 represents the information in Table 12 where the encryption and decryption throughputs of the three ECC schemes were compared. As can be seen, the encryption and decryption throughputs of the three schemes are approximately equal. This result implies that the modification in IECC does not make its encryption and decryption throughputs lower than the encryption and decryption of the existing ECC schemes.

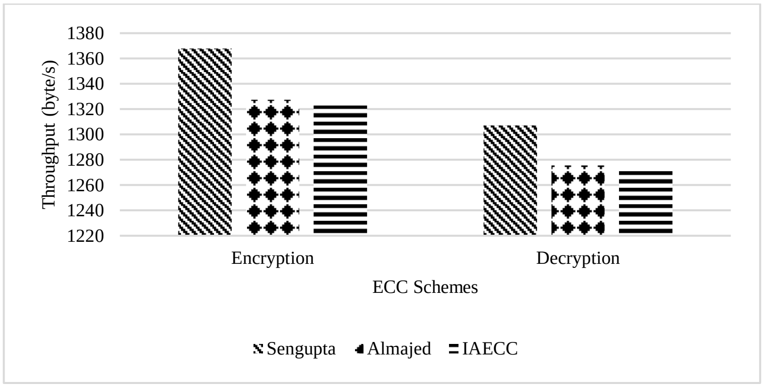


Figure 9. Comparative analysis of encryption/decryption throughput of ECC schemes

Table 12. Average encryption and decryption throughputs of three ECC schemes

		Average Encryption and Decryption Throughput (byte/s)		
S/N	scheme	Sengupta	Almajed	IAECC
1	Encryption	1367.911	1327.398	1322.962
2	Decryption	1307.233	1275.503	1273.339

Figure 10 represents the information in Table 13 where the encryption and decryption energy consumption of the three ECC schemes were compared. The results in the Figure 9 reveals that the energy consumption of the system during encryption and decryption are approximately equal. This result implies that the modifications in IECC does not make the energy consumption of IECC higher than the energy consumption in the existing ECC schemes.

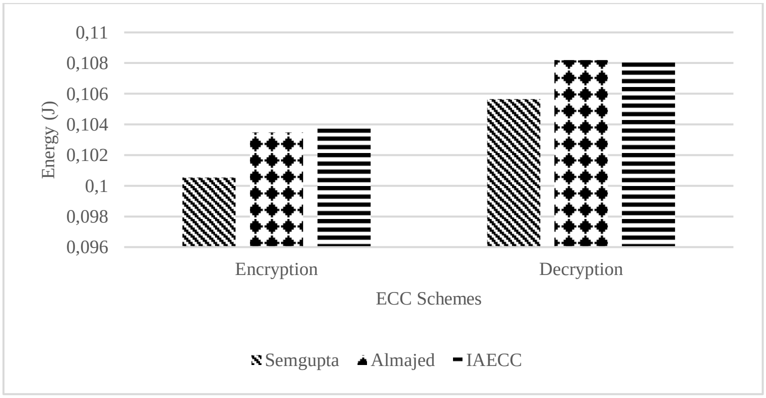


Figure 10. Comparative analysis of encryption/decryption energy consumption of ECC schemes

Table 13. Average encryption and decryption energy consumption of three ECC schemes

		Average Encryption and Decryption Energy Consumption (J)		
S/N	scheme	Semgupta	Almajed	IAECC
1	Encryption	0.100539	0.103485	0.103721
2	Decryption	0.105648	0.108192	0.108237

6. CONCLUSION

In this study, flaws in the existing implementation of ECC schemes are identified. A new scheme is created to address the flaws. This study also undertook a security analysis to present a proof for the resistance of the developed scheme against specific encryption attacks. Additionally, the study conducted a performance evaluation to compare the impact of the security enhancements of the proposed scheme on encryption and decryption times, throughput, and power consumption with those of other schemes. The experimental results of the security analysis show that IAECC is resistant to the security flaws that the existing systems are vulnerable to. Again, the results of the comparative performance analysis of IAECC with other systems show that the IAECC scheme performed just as well as the other schemes, with no noticeable increase in computation overhead. So, IAECC is more secure than the existing schemes while keeping the same amount of work to do.

REFERENCES

[1] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange, “Elligator: Elliptic-curve points indistinguishable from uniform random strings,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 967–979, 2013, doi: 10.1145/2508859.2516734.

[2] J. R. Shaikh, M. Nenova, G. Iliev, and Z. Valkova-Jarvis, “Analysis of standard Elliptic curves for the implementation of elliptic curve cryptography in resource-constrained E-commerce applications,” *2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems, COMCAS 2017*, vol. 2017-Novem, pp. 1–4, 2017, doi: 10.1109/COMCAS.2017.8244805.

[3] G. Dhamodharan and S. Thaddeus, “Multifarious mapping schemes on Elliptic curve cryptography for IoT security,” *Test Engineering and Management*, vol. 81, no. August, pp. 5128–5136, 2019.

[4] H. N. Almajed and A. S. Almogren, “SE-Enc: A secure and efficient encoding scheme using Elliptic curve cryptography,” *IEEE Access*, vol. 7, pp. 175865–175878, 2019, doi: 10.1109/ACCESS.2019.2957943.

[5] Y. A. Abbas, A. S. Hameed, S. H. Alwan, and M. A. Fadel, “Efficient hardware implementation for lightweight mCrypton algorithm using FPGA,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 3, pp. 164–1680, 2021, doi: 10.11591/ijeecs.v23.i3.pp1674-1680.

[6] A. Sengupta and U. K. Ray, “Message mapping and reverse mapping in Elliptic curve cryptosystem,” *Security and Communication Networks*, vol. 9, no. 18, pp. 5363–5375, 2016, doi: 10.1002/sec.1702.

[7] J. Muthukuru and B. Sathyanarayana, “Fixed and variable size text based message mapping techniques using ECC,” *global journal of computer science and technology*, vol. 12, no. 3, pp. 12–18, 2012.

[8] R. Navatejareddy, M. Jayabhaskar, and B. Sathyanarayana, “Elliptical curve cryptography image encryption scheme with aid of optimization technique using gravitational search algorithm,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 1, pp. 247–255, 2022, doi: 10.11591/ijeecs.v25.i1.pp247-255.

[9] “Guide to Elliptic curve cryptography,” *Guide to Elliptic Curve Cryptography*, 2004, doi: 10.1007/b97644.

[10] E. H. El Kinani, “Fast mapping method based on matrix approach for Elliptic curve cryptography,” *International Journal of Information and Network Security (IJINS)*, vol. 1, no. 2, pp. 54–59, 2012, doi: 10.11591/ijins.v1i2.430.

[11] B. Padma, D. Chandravathi, and P. P. Roja, “Encoding and decoding of a message in the implementation of Elliptic curve cryptography using Koblitz’s method,” *(IJCSE) International Journal on Computer Science and Engineering*, vol. 02, no. 05, pp. 1904–1907, 2010.

[12] H. Almajed and A. Almogren, “A secure and efficient ecc-based scheme for edge computing and internet of things,” *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–31, 2020, doi: 10.3390/s20216158.

[13] L. Ferretti, M. Marchetti, and M. Colajanni, “Fog-based secure communications for low-power IoT devices,” *ACM Transactions on Internet Technology*, vol. 19, no. 2, 2019, doi: 10.1145/3284554.

[14] F. Albalas, M. Al-Soud, O. Almomani, and A. Almomani, “Security-aware CoAP application layer protocol for the internet of things using elliptic-curve cryptography,” *International Arab Journal of Information Technology*, vol. 15, no. 3A Special Issue, pp. 550–558, 2018.

[15] S. Khan and R. Khan, “Elgamal Elliptic curve based secure communication architecture for microgrids,” *Energies*, vol. 11, no. 4, 2018, doi: 10.3390/en11040759.

[16] A. U. Ay, C. Mancillas-López, E. Öztürk, F. Rodríguez-Henríquez, and E. Savaş, “Constant-time hardware computation of Elliptic curve scalar multiplication around the 128 bit security level,” *Microprocessors and Microsystems*, vol. 62, pp. 79–90, 2018, doi: 10.1016/j.micpro.2018.05.005.

[17] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48, no. 177, pp. 201–209, 1987.

[18] D. Mahto and D. Kumar Yadav, “Performance analysis of RSA and Elliptic curve cryptography,” *International Journal of Network Security*, vol. 20, no. 4, pp. 625–635, 2018, [Online]. Available: www.linkedin.com,.

[19] A. J. Zargar, “Encryption/decryption using Elliptical curve cryptography,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 7, pp. 48–51, 2017, doi: 10.26483/ijarcs.v8i7.4208.

[20] Y. Yin, L. Wu, Q. Peng, and X. Zhang, “A novel spa on ecc with modular subtraction,” *Proceedings of the International Conference on Anti-Counterfeiting, Security and Identification, ASID*, vol. 2018-Novem, pp. 179–182, 2018, doi: 10.1109/ICASID.2018.8693138.

[21] S. D. Galbraith and F. Vercauteren, “Computational problems in supersingular Elliptic curve isogenies,” *Quantum Information*

Processing, vol. 17, no. 10, 2018, doi: 10.1007/s11128-018-2023-6.
[22] T. Shahroodi, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "Low-latency double point multiplication architecture using differential addition chain over GF(2m)," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1465–1473, 2019, doi: 10.1109/TCSI.2018.2883557.
[23] L. D. Singh and K. M. Singh, "Image encryption using Elliptic curve cryptography," *Procedia Computer Science*, vol. 54, pp. 472–481, 2015, doi: 10.1016/j.procs.2015.06.054.
[24] V. S. Miller, "Use of Elliptic curves in cryptography," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 218 LNCS, pp. 417–426, 1986, doi: 10.1007/3-540-39799-X_31.
[25] S. M. C. Vigila and K. Muneeswaran, "Implementation of text based cryptosystem using Elliptic curve cryptography," *2009 1st International Conference on Advanced Computing, ICAC 2009*, pp. 82–85, 2009, doi: 10.1109/ICADVC.2009.5378025.
[26] Z. E. Dawahdeh, S. N. Yaakob, and R. R. Bin Othman, "A new modification for Menezes-Vanstone Elliptic curve cryptosystem," *Journal of Theoretical and Applied Information Technology*, vol. 85, no. 3, pp. 290–297, 2016.
[27] K. Keerthi and B. Surendiran, "Elliptic curve cryptography for secured text encryption," *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2017*, 2017, doi: 10.1109/ICCPCT.2017.8074210.
[28] P. Das and C. Giri, "An efficient method for text encryption using Elliptic curve cryptography," *Proceedings of the 8th International Advance Computing Conference, IACC 2018*, pp. 96–101, 2018, doi: 10.1109/IADCC.2018.8692087.
[29] S. Debnath, M. V. L. Nunsanga, and B. Bhuyan, "Study and scope of signcryption for cloud data access control," *Lecture Notes in Networks and Systems*, vol. 41, pp. 113–126, 2019, doi: 10.1007/978-981-13-3122-0_12.
[30] C. Boyd, B. Hale, S. F. Mjølsnes, and D. Stebila, "From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9610, pp. 55–71, 2016, doi: 10.1007/978-3-319-29485-8_4.
[31] M. Abdalla, F. Benhamouda, and D. Pointcheval, "Public-key encryption indistinguishable under plaintext-checkable attacks," *IET Information Security*, vol. 10, no. 6, pp. 288–303, 2016, doi: 10.1049/iet-ifs.2015.0500.
[32] T. Shrimpton, "A characterization of authenticated-encryption as a form of chosen-Ciphertext security," *IACR Eprint archive*, pp. 1–7, 2004, [Online]. Available: http://eprint.iacr.org/2004/272.
[33] O. S. Rao and S. P. Setty, "Efficient mapping methods for Elliptic curve cryptosystems," *International Journal of Engineering Science and Technology*, vol. 2, no. 8, pp. 3651–3656, 2010, [Online]. Available: http://www.ijest.info/docs/IJEST10-02-08-59.pdf.
[34] A. V. Vijayalakshmi, "Enhancing the security of Iot data using multilevel encryption," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 9, pp. 841–845, 2017, doi: 10.26483/ijarcs.v8i9.4959.
[35] J. Joglekar, S. Bhutani, N. Patel, and P. Soman, "Lightweight Elliptical curve cryptography (ECC) for data integrity and user authentication in smart transportation IoT system," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 39, pp. 270–278, 2020, doi: 10.1007/978-3-030-34515-0_28.

BIOGRAPHIES OF AUTHORS

Esau Taiwo Oladipupo received his first degree (B.Tech) Computer Engineering from the Department of Computer Science and Engineering, Ladoke Akintola University of Technology in 2003. He has also Master degree (M. Tech) Computer Science, from the same institution in 2016. He has been a Lecturer in the Department of Computer Science the Federal Polytechnic Bida since 2009. He is currently pursuing his Ph.D. in Computer Science from the University of Ilorin, Nigeria. His research area includes Computer and Communication Security, Crytography and Biometric. He can be contacted at email: taiwotheophilus@gmail.com or oladipupo.esau@fedpolybida.edu.ng.

Oluwakemi Christiana Abikoye has a B.Sc, M.Sc. and Ph.D degrees in Computer Science. She is an academic staff at the Department of Computer Science, Faculty of Communication and Information Sciences, University of Ilorin, Ilorin, Nigeria. Her research interests include Cryptography, Computer and Communication Network (Cyber) Security, Biometrics, Human Computer Interaction and Text and Data Mining. She can be contacted at email: abikoye.o@unilorin.edu.ng.