Security Improvement in Elliptic Curve Cryptography

Kawther Esaa Abdullah, Nada Hussein M. Ali
Department of Computer Science, College of Science
University of Baghdad
Baghdad, Iraq

***Abstract*—This paper proposed different approaches to enhance the performance of the Elliptic Curve Cryptography (ECC) algorithm. ECC is vulnerable to attacks by exploiting the public parameters of ECC to solve Discrete Logarithm Problem (DLP). Therefore, these public parameters should be selected safely to obviate all recognized attacks. This paper presents a new generator function to produce the domain parameters for creating the elliptic curve; a secure mechanism is used in the proposed function to avoid all possible known attacks that attempts to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP). Moreover, an efficient algorithm has been proposed for choosing two base points from the curve in order to generate two subgroups in a secure manner. The purpose of the aforementioned algorithm is to offer more confidence for the user since it is not built upon a hidden impairment that it could be subsequently utilized to retrieve user's private key. The Elliptic Curve Diffie Hellman (ECDH) algorithm is implemented to exchange a session key between the communicating parties in a secure manner. Beside, a preprocessing operation is performed on the message to enhance the diffusion property and consequently leads to increase the strength against cryptanalysis attack. Finally, the dual encryption/decryption algorithm is implemented using different session keys in each stage of the encryption to boost immunity against any attack on the digital audio transmission. The gained results show the positive effect of the dual elliptic curve system in terms of speed and confidentiality without needing any extra time for encryption.**

Keywords*—Elliptic curve cryptography; elliptic curve discrete logarithm problem; dual encryption/decryption; Elliptic Curve Diffie Hellman

I. INTRODUCTION

Elliptic curves were suggested by Neal Koblitz and Victor Miller independently in 1985 to design a public-key cryptographic system [1]. The Elliptic Curve Cryptography (ECC) is a public-key cryptosystem which playing an important role in cryptography world. The shorter key size in ECC provides an equivalent protection level for public-key algorithms which utilized the largest key size (e.g., Rivest Shamir Adleman (RSA)). In addition, the ECC offers more security compared to the RSA algorithm since it is based on DLP, while the latest algorithm based on the prime number factorization problem [2], [3]. ECC is based on an Abelian group, the main operation used in ECC is the addition operation; the multiplication is defined as a repeated addition. For example, $\times k = (a + a + \cdots + a)$, addition a with times k and it is performed over an elliptic curve. Cryptanalysis includes determining k given a and $(a \times k)$, this is called DLP. The definition of elliptic curve is based on the equation, two variables and two coefficients; the values of variables and coefficients are limited to elements of a finite field [1]. In this paper, the elliptic curve over the prime field GF_p is considered.

A. Mathematics of ECC Over Finite Field

In general, an elliptic curve E over prime field (F_p) denoted by E(F_p) is given by simplified the weierstrass equation as below [1]:

$$E: y^2 \ mod\ p \equiv (x^3 + ax + b) \ mod\ p \quad (1)$$

Where, b and $x, y \in F_p$, The value of variables are sets of elements from 0 to $p-1$. In addition, the coefficients must satisfy (2), where Δ denoted to the discriminant of E.

$$\Delta = (4a^3 + 27b^2) \ mod\ p \neq 0 \quad (2)$$

B. Point Addition

If two points on an elliptic curve were added to each other, the output result represents a third point that denotes the intersection of that curve. Graphically, drawing a straight line between any two points on a curve represents a tangent line and reflects a third point around the $x - axis$ as denoted in (5). The formula $P + Q = -R$ represents the addition operation between points P(x,y) with Q(x,y) to produce R(x,y) by applying (3) and (4) [1], [2].

C. Point Doubling

The output value of adding a point P(x,y) on the curve to itself in condition that $y_P \neq 0$ will yield the point R. One could draw a tangent line where the intersection of that line on the curve represents the cross reflection point on $x - axis$ (the R point), where $+P = 2P = -R$. Equations (3), (4) and (6) are used to compute second point R(x,y) and the tangent line (slope), respectively [1],[2].

$$x_R = (\lambda^2 - x_P - x_Q) \ mod\ p \quad (3)$$

$$y_R = (\lambda(x_P - x_R) - y_Q) \ mod\ p \quad (4)$$

Where,

$$\lambda = (y_2 - y_1 / x_2 - x_1) \ mod\ p \quad (5)$$

$$\lambda = (3x_1^2 + a / 2y_1) \ mod\ p \quad (6)$$

The strength of the ECC depends on the DLP. This means that logarithm Q to base ($l = \mathrm{Log}_G Q$), where l is a private key, G is a base point and Q is a public key (G, Q are publicaly parameters). There are some attacks on the Elliptic Curve Discrete Logarithm Problem (ECDLP) from given G, Q try to extract l. To avoid all recognized attacks on the ECDLP should be selected, the domain parameters for ECC cautiously and in a secure way.

The layout of this paper is composed of the following sections: the related work of elliptic curve cryptography is introduced in Section II, Section III describes the concepts of the advanced encryption standard and cryptographic hash function, the linear congruential generator is presented in Section IV, Section V clarifies the password based key derivation function, Section VI discusses the proposed system followed by the experimental results and discussion in Section VII, and finally, Section VIII presents the conclusions.

II. Related Work

In literature, many researchers have attempted to utilize the strength of the elliptic curve to implement in different tasks of the public key cryptography. Summarized below are some of the features of the linked work.

Rahul Singh, et al. [4] in 2014 investigated an implementation for ECC encryption and decryption audio file was presented.

Artan Luma, et al. [5] in 2015 presented the encryption and decryption for audio file transported through the network - based on ECC. In this study, the scholars have been concluded that ECC is suitable for large amounts of data and also the ECC is preferred comparing with RSA since it provided the same security with small key size.

Manish Kumar, et al. [6] in 2016 proposed a new method for image security by using DNA for encoding RGB image thereafter applied encryption based on Elliptic Curve Diffie–Hellman Encryption (ECDHE). This algorithm supplied a double layer of security.

Fang, Xianjin and Wu, Yanting. [7] in 2017 studied the details of the elliptic curve cryptography, this discussion includes the basic information about ECC and how to partition a message into blocks and encoding/decoding the message into points on the curve using the koblitz method. Also, is presented the encryption/decryption with the elliptic curve. The researcher concluded that the ECC is utilized for encryption, key exchange, and the digital signature with swift and lesser memory.

Kawther, Esaa and Nada, Hussein [8] in 2018 have been investigated a new mapping method based on x-coordinate values of an elliptic curve to generate a secret lookup table, this table is used to convert samples of an audio file (or even any data type) into points on the elliptic curve and vice versa. Besides, the changing form of samples before applying the proposed method to make cryptanalysis more difficult to guess the points on the curve by an intruder (through exploiting statistical analysis) to achieve diffusion, the obtained results indicate that the proposed method is faster, more secure and less time-consuming when embedding a message into a point on the curve.

III. Advanced Encryption Standard and Cryptographic Hash Functions

In 2001 the National Institute of Standards and Technology (NIST) has issued the Advanced Encryption Standard (AES). The AES belongs to a symmetric encryption algorithms family and the type of process is a block cipher, it replaces the Data Encryption Standard (DES) as a criterion for an enormous domain of applications. The block size of a plain-text input to the AES is 128 bits and key size is 128,192,256 bits. The number of rounds in AES algorithm is altered rely on the key size, 10 rounds for key size 16 bytes, 12 rounds for 24 bytes and 14 rounds for 32 bytes. Each round includes four transformation functions are (SubBytes, ShiftRows, MixColumns, and AddRoundKey) but the last round comprises the three transformations except MixColumns [1], [9], [10].

Cryptographic hash functions play a significant part in the computer security and can be used in various applications such as in Message Authentication, Digital Signatures, hash-based key derivation functions and also in Pseudorandom Number Generator. Hash functions take an arbitrary size of input and produce a fixed size called (hash code or message digest). The basic features of Hash functions are preimage resistant (infeasible to obtain a message from knowing its hash code), second preimage resistant (from given a message x impossible to find a message y has the same hash code of x) and collision resistant (impossible to get any pair (x, y) has the same hash code) [11], [12]. In 2002, NIST designed a new version called the family of the Secure Hash Algorithm (SHA-2) content on three algorithms are SHA-256, SHA-384 and SHA-512 the length of bits produced for each one of them are 256, 384 and 512 bits respectively [1], [12].

IV. Linear Congruential Generators

A linear congruential algorithm was suggested by Lehmer and it is mostly used for pseudorandom number generator in order to generate a sequence of numbers such as $x_j = x_0,x_1,x_2,\ldots,x_n$ by relying on the following repetition equation [1]:

$$x_{j+1} = ax_j + b \ \ mod\ n \tag{7}$$

Where, x_j is the seed value $0 \leq x_j < n$, a represents the multiplier $0 < a < n$, b is the increment value $0 \leq b <$ n, n represents the modular $n > 0$. Choosing the values of (a, b *and* n) accurately helps to produce ideal sequence numbers. The characteristics of this generator are easy to execute and pass the next statistical tests: the frequency test, run test, serial test, poker test and etc., it can be regarded as the best selection for generating robust random numbers [1], [13]. In this study, the proposed system specifies the value of a and x are prime numbers to give a large period in the outcome of the Linear Congruential Generator (LCG).

V. Password Based Key Derivation Function

With the issue of PKCS #5 v2.0 and RFC 2898 [14], is one of the most famous key-derivation functions, Password-Based Key Derivation Function #2 indicated as PBKDF2, with a view to deriving cryptographic keys from (human entrance) passwords. PBKDF2 aims to frustrate expected attacks on password like the dictionary and brute force attacks by incrementing the time necessary for checking every password through two significant parameters in PBKDF2 are the salt and iteration count, it will be difficult to execute these attacks [15]. PBKDF2 is a pseudo-random number PRF that based on some parameters: Salt S, Iteration Count C, Password P and Len_{key} which is the length of derived key also called master key M_{key} actually $(2^{32}-1)$ × Length of hash function (Len_{hash}). Usually, the PRF includes an HMAC (Hash Message Authentication

Code) structure depends on a cryptographic hash function, which can select from the designer. The general expression of PBKDF2 is defined as [16]:

$$M_{key} = PBKDF2_{(PRF,C)}(P, S, Len_{key})$$

The iteration count *C* is a constant value represents the number of iterated requisitions of the PRF in order to produce one block of the M_{key}, according to RFC 2898, a minimum *C* of 1000 is recommended, where the benefit of *C* increment the calculated cost of carrying out a dictionary attack on a password. While the value of salt offers a wide range of keys for all password. Further, information can be seen in [14].

VI. Proposed System

The proposed system is divided into four main phases, the main idea of each part of this system is to strength the security against the cryptanalyst and to reduce the risks of compromised the private key in ECC algorithm. The following sections will demonstrate the procedures for each one.

A. The New H-AES-LCG Generator

The first phase of the proposed system is assigned for generating the domain parameters (Prime number $(p), a, b$) that used for creating the ECC. The purpose of the H-AES-LCG generator is to form an elliptic curve in a random and secure manner to avoid all possible known attacks against it. The steps of constructing the H-AES-LCG generator are described as follows:

Stage 1: Apply the family of SHA-2 includes (SHA-512, SHA-384, SHA-256) and MD5 (Message Digest 5) sequentially on a seed value *(String Password)* shown as below:

1) The inputs to the family of SHA-2 are:

- **String Password:** The Left Circular Shift (LCS) process has been done with a password before it is entering the hash algorithms. The amount of rotation is determined randomly and denoted by (R_{len}), each letter in the password is shifted with a different R_{len} through converting characters of the password into bytes to store it in a byte array denoted by ($Pass$), the length of a password denotes as P_{len}. The Pseudo-code for applying the LCS is shown as below:

  a. $\boldsymbol{For}\ i = 0\ \boldsymbol{to}\ P_{len}$
  b. $R_{len} = (R_{len} + i)\ \boldsymbol{mod}\ 8$
  c. $Pass\_rotation\ (i) = ((Pass(i) \ll R_{len}) | (Pass(i) \gg (8 - R_{len})))\ \&\ 127$
  d. $\boldsymbol{End}$

The main purpose of the above operation is to increase the strength of the password by permutation its

- **Salt Value:** The time of recording event on the computer (Time Stamp) has been taken as a salt value, it includes the date and time, for example, (2018-02-08 02:11:55 AM). This operation is considered as a one-time pad key to increase the security.

2) Execute the family of SHA-2 (SHA-512, SHA-384, SHA-256) with the salt value on the String Password after applying the rotation process as explained in point 1.

3) Merge between the results of the family SHA-2 to obtain a string of size **1152**-bit denoted as a $\boldsymbol{Hpassword}$. The encoding system implemented in the present study is Unicode which uses two bytes (16-bit) for each character, in this case the $\boldsymbol{Hpassword}$ composed of **72** characters.

4) Expanding the $\boldsymbol{Hpassword}$ to make it as an input to the MD5 algorithm through converting the $\boldsymbol{Hpassword}$ to characters, pick some characters from it and insert some ASCII letters chooses randomly in a specific manner to form a new string. This technique can be described as follows:

At the beginning, determine some secret parameters which are (*start, amount of characters and jump)* to extract a new string from the existing string that has been produced in point 3, the pseudo-code is illustrated as below:

- Initializing the Parameters:

  a. $\boldsymbol{Ch}$**:** array of character to store characters in the $\boldsymbol{Hpassword}$.
  b. $\boldsymbol{St}$**:** represents the start value, which is a positive integer value chosen randomly and specified from a range between (0 - size of Ch).
  c. $\boldsymbol{No.ofCh}$**:** amount of characters (length of a new string) is a positive integer chosen according to the length of string that needs.
  d. $\boldsymbol{jump_1}$**:** a positive integer value to represent the amount of jump between characters.

- Processing:

  a. $ExHpassword = Ch(St)\ ||\ ASCII\ Character$
  b. $\boldsymbol{For}\ i = 2\ \boldsymbol{to}\ No.ofCh$
  c. $St = (St + jump_1)\ \boldsymbol{mod}\ length\ of\ Ch$
  d. $ExHpassword = ExHpassword\ ||\ Ch(St)\ ||\ ASCII\ Character$
  e. $\boldsymbol{End}$

The resultant $ExHpassword$ string considered as an input to MD5 algorithm. The aforementioned mechanism is applied in order to make an attack for the MD5 algorithm is difficult to guess the string password.

Stage 2: The requirements of AES algorithm for generating random sequence bits are:

1) ***Plain-text:*** The output of the MD5 algorithm makes as a plain-text to the AES, the number of bits produced from the MD5 corresponds to the size of the plain-text for AES algorithm which is 128 bits.

2) ***Key:*** Prepare a key for the AES algorithm using PBKDF2 function, it applied to derive a master key (M_{key}) for AES. The parameters to perform PBKDF2 comprised from:

- **Password:** Use the LCG algorithm to generate the pseudo-random numbers and make it as a password denoted by ($Random_{Password}$). The values of

$Random_{Password}$ are specified in the range between $[0, \ldots, 1024]$. The pseudo-code is shown as the following:

a. Initialization four parameters to use in (7), two prime numbers p_1 is a multiplier, q_1 is seed value, (e.g., the value of p_1, q_1 are 10247 and 8161 respectively) and two positive integer b_1, n_1 where, b_1 is representing the increment equal to 1 and n_1 is a modulus which is equal to 1024 in the present work.
b. $\boldsymbol{For}\ i = 0\ \boldsymbol{to}\ n_1$
c. $q_1 = (p_1 \times q_1) + b_1\ \boldsymbol{mod}\ n_1$
d. $Random_{Password}(i) = q_1$
e. $\boldsymbol{End}$

- **Salt value:** A time stamp (Date and Time) is regarded as a salt value (S).
- **Iteration Count:** Number of iterations to derive master key denoted as C, in this work C equal to 10000 iterations to increment the calculated cost of carrying out a dictionary attack on a password.
- **Length:** Indicate to the length of derived key (M_{key})in bits and denoted by(Len_{key}), in this work Len_{key} equal to 256 bits.

The computation of both the plaintext from MD5 algorithm and the key from PBKDF2 are used as input for AES algorithm. The later algorithm is implemented to generate a single encrypted block stored in an array called $Single\ encrypted_{Block}$ of size equal to 128 bits.

S**tage 3:** In order to increase the randomness of crop from the AES algorithm the Exclusive-OR (XOR) bit wise operation is implemented between, the $Single\ encrypted_{Block}$ and the random numbers generator that generated from the LCG algorithm $(RandomSequece)$ by applying (7). The values of the random number are in the range $[0, \ldots, length\ of\ Single\ encrypted_{Block}]$, the pseudo-code of the above process is illustrated as below:

a. Initialize four parameters include two prime numbers are p_2, q_2 (e.g., the value of p_2, q_2 are 7933 and 4093 respectively) and two positive integer numbers, $b_2 = 1$, $n_2 = length\ of\ Single\ encrypted_{Block}$ to satisfy Equation (7).
b. $\boldsymbol{For}\ i = 0\ \boldsymbol{to}\ n_2$
c. $q_2 = ((p_2 \times q_2) + b_2)\ \boldsymbol{mod}\ n_2$
d. $Random_{Sequece}(i) = q_2$
e. $RandomSequeceBits(i) = Single\ encrypted_{Block}(i) \oplus Random_{Sequece}(i)$
f. $\boldsymbol{End}$

The proposed H-AES-LCG algorithm usually generates 128-bit and can be used by any algorithm needs a random number generator. The parameters needed to extract any random number from the string produced by proposing algorithm are:

- $\boldsymbol{RandomSequeceBits}$: Represents the random sequence bits that produced from the H-AES-LCG generator.
- $\boldsymbol{start}$: A positive integer number specified in the range $[0 - 127]$ in order to represent an index of a bit that stored in $RandomSequeceBits$.
- $\boldsymbol{size}$: Refers to how many number of bits that needs.
- $\boldsymbol{jump_2}$: A positive integer number represents jump between the random sequence bits.

This paper has been applied the proposed generator (H-AES-LCG) to extract the domain parameters (p, a, b) for the ECC algorithm, these parameters are used to plot a secure elliptic curve from the output of H-AES-LCG generator. Algorithm 1 shows how to extract the domain parameters (p, a, b) for the ECC algorithm. Also, the aforementioned procedures of H-AES-LCG generator are demonstrated in Fig. 1.

Algorithm 1: Extract the Domain Parameters to Plot a Secure Curve

Input:

- $\boldsymbol{Password, R_{len}, Salt\ value}$**:** Serves as input to the family of SHA-2
- $\boldsymbol{ASCII\ letters, St, No.ofCh, jump_1}$**:** Parameters for producing $\boldsymbol{ExHpassword}$
- $\boldsymbol{p_1, q_1, n_1, b_1}$: Parameters for LCG algorithm to produce $\boldsymbol{Random_{Password}}$
- $\boldsymbol{Random_{Password}, S, C, Len_{key}}$**:** Parameters for PBKDF2 to produce $\boldsymbol{M_{key}}$
- $\boldsymbol{p_2, q_2, n_2, b_2}$: Parameters for LCG algorithm to produce $\boldsymbol{Random_{Sequece}}$
- $\boldsymbol{start, size, jump_2}$: Parameters for extract a value from $\boldsymbol{RandomSequeceBits}$

Output:

- $\boldsymbol{p, a, b}$

1: Parameters Setting

Generate $RandomSequeceBits$ using the H-AES-LCG generator
$ML = 1$ // is a positive integer value initialization by one to make a number of multiples two.

$Count = 1$ // To avoid exceeding the number of bits that specified.

2: Extract the domain parameters from invocation the Function1

$\boldsymbol{p}$ = Function1$(RandomSequeceBits, start, size, jump_2)$
$\boldsymbol{a}$ = Function1$(RandomSequeceBits, start, size, jump_2)$
$\boldsymbol{b}$ = Function1$(RandomSequeceBits, start, size, jump_2)$

3: **Check the primality of p using Miller-Rabin algorithm and values of the coefficient (a, b) are satisfied Equation (2), If p is not prime increament p by two and go to Miller-Rabin algorithm. Check values of the coefficient (a, b) are not satisfied Equation (2) go to Step 2.**

Function1 $(\boldsymbol{RandomSequeceBits}, \boldsymbol{start}, \boldsymbol{size}, \boldsymbol{jump_2})$

$number = RandomSequeceBits(start)$
For $i = 2$ ***to*** $size$

$start =$
$(start + jump_2)$ ***mod*** $length\ of\ RandomSequeceBits$
$number = number + RandomSequeceBits\ (start) \times ML$
$ML = ML \times 2$
$Count = Count + 1$
if $(Count > size)$
$ML = 1$

End
Return *number*

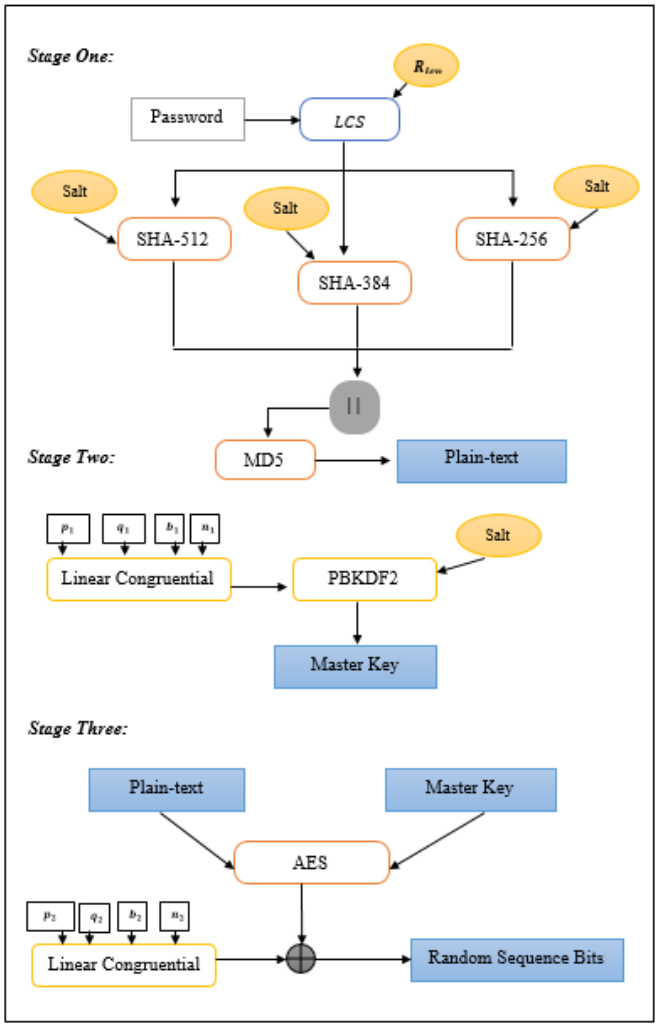

Fig. 1. The block diagram for the H-AES-LCG generator.

B. Base Point Selection from an Elliptic Curve Over Prime Field

After determining the domain parameters (p, a, b) for an elliptic curve denoted by E in a secure mechanism that based on the H-AES-LCG generator, it will be selected two base points from the points on the E. There are two important matters to improve the adequacy of ECC: the determination of the base point from E and the point multiplication operation. The choice of two base points from E that depends on an efficient technique in order to generate two subgroups to implement dual encryption. Each point on elliptic curve should be satisfied as (1) by computing the quadratic residues $mod\ p$ denoted as Q_p in order to obtain the value of $y - coordinate$. The quadratic residues obtained by substitution the value of x in the range $0 \leq x < p$ in (1) and comparing the crop from (1) with Q_p if equal then get the value of $y - coordinate$.

Every parity in the proposed technique must agree on the value of $x - coordinate$, which determine randomly in the range$[0, \dots, p-1]$ to select two base points. Then, applying the right- hand side of (1) to check the quadratic residue of the crop which is denoted as (α) computed by Euler criteria [17], it is defined as below.

$$\alpha^{(p-1)/2} = \begin{cases} 1 & is\ quadratic\ residue \\ -1 & is\ non-quadratic\ residue \end{cases} \quad (8)$$

Where:

α is an integer belong to prime field p.

The checking quadratic residue of $x - coordinate$ to obtain the value of $y - coordinate$ will leads for retrieving the first base point (α, y) denoted by G_1 and second base point (x, y) labeled as G_2, both achieved through use (1). Algorithm 2 illustrates the proposed technique to determinate two points on an elliptic curve (E).

Algorithm 2: Selection Two Base Points from an Elliptic Curve

---

Input:

- $\boldsymbol{p}$: Prime number
- $\boldsymbol{a}, \boldsymbol{b}$: The coefficients values using in (2)
- $\boldsymbol{x}$: An integer value specified in the range $[\boldsymbol{0}, \dots \boldsymbol{p-1}]$

Output:

- $(\boldsymbol{\alpha}, \boldsymbol{y})$, $(\boldsymbol{x}, \boldsymbol{y})$**:** Two Base Points

---

1: **Parameters Setting**
$s = 0$
$y = 1$

2: **Compute the quadratic residue of x denoted as Q_x by implementing the right-hand side of Equation (1). After that, checking the value of x has quadratic residue using Equation (8).**

$\alpha = (x^3 + ax + b)$ ***mod*** p
$Q_x = (\alpha^{(p-1)/2})$ ***mod*** p

While $(\alpha == 0 \mid Q_p != 1)$

$x = (x + 1)$ ***mod*** p

go to step 2

End

3: Recover two base points

For $i = 1$ *to* p

$s = s + i$ ***mod*** p

if $(s == \alpha)$
Return
$(\alpha, y), (x, y)$
else
$y = y + 1$
$i = i + 2$

End

Furthermore, the proposed technique provides an enhancement in checking the quadratic residue of $mod\ p$ for a value of x and recovers the value of $y - coordinate$ at the same time. The total time complexity of recovering two base points in the proposed method is $O(\frac{p-1}{2})$.

Moreover, it would be generated two subgroups by applying the *Doubling-Addition* operations; are the basic operations in ECC; from the two base points also called generator points, all one separately. At the beginning, it executes the *Doubling* operation on the base point with itself and then, the *Addition* operation is implemented on the first base point and the outcome from doubling operation and so on until reach to the infinity point (∞), where the doubling operation implements only once in the beginning. In addition, it would be applied the same mentioned above in the second base point in order to generate two subgroups, where the first base point produces the first subgroup and the second base point generates the second subgroup. The points in the subgroup are considered as the public keys and the number of points in the subgroup (the order of subgroup) specify the range of the private keys, where each a private key scalar corresponding to a public key point in the subgroup.

C. Elliptic Curve Diffie Hellman Key Exchange

Elliptic Curve Diffie Hellman is used to exchange a session key between the parties. It has been described as a method to generate two session keys by applying ECDH algorithm in order to use them with dual ECC encryption. This procedure is illustrated below:

Step1: Both parties agree on the domain parameters $(p, a, b, G_1, order_1, G_2, order_2)$ in a secure manner depended on the H-AES-LCG generator and the method for selection the two base points. Where, $order_1$ is the number of points that is generated from the G_1 and $order_2$ represents number of points that is generated from G_2.

Step2: Each party (Alice and Bob) selects two private keys (N_{a1}, N_{a2}, N_{b1}, N_{b2}) respectively, which are smaller than order. The selection of the first private keys (N_{a1}, N_{b1}) for each party is from the range of $order_1$ (the first subgroup), while the second private keys (N_{a2}, N_{b2}) from the range of $order_2$ (the second subgroup).

Step3: Calculate two public keys for each party through multiply private key with the base point, where (P_{a1}, P_{a2}, P_{b1}, P_{b2}) are the public keys for Alice and Bob respectively. The (P_{a1}, P_{b1}) are points in the first subgroup and (P_{a2}, P_{b2}) are points in the second subgroup, and are computed as follows.

$$P_{a1} = N_{a1} \times G_1 \ mod\ p \quad (9)$$

$$P_{a2} = N_{a2} \times G_2 \ mod\ p \quad (10)$$

In the same way, Bob can compute his public keys as Alice.

Step4: The computation of two session keys ($\boldsymbol{k_{s1}}, \boldsymbol{k_{s2}}$), at the beginning, exchange the public keys between the parties (Alice/Bob) and then, each one computes the session key by multiplying his/her private key with the public key for the corresponding party. This operation is demonstrated as below:

Alice:

$$k_{s1} = N_{a1} \times P_{b1} \ mod\ p \quad (11)$$

$$k_{s2} = N_{a2} \times P_{b2} \ mod\ p \quad (12)$$

Bob:

$$k_{s1} = N_{b1} \times P_{a1} \ mod\ p \quad (13)$$

$$k_{s2} = N_{b2} \times P_{a2} \ mod\ p \quad (14)$$

Analysis of the above is similar on both sides.

After the two parties agreed of the two session keys ($\boldsymbol{k_{s1}}, \boldsymbol{k_{s2}}$), which utilize in dual encryption phase that will be discussed later. Here multiplication is not implied simple multiplication, which is an algebra, rather it is repeated addition of points by the point multiplication operation (scalar multiplication) in ECC.

$k_{s1} = (N_{a1} \times P_{b1}) \ mod\ p$	$k_{s1} = (N_{b1} \times P_{a1}) \ mod\ p$
$(N_{a1} \times N_{b1} \times G_1) \ mod\ p$	$\equiv \quad (N_{b1} \times N_{a1} \times G_1) \ mod\ p$
$k_{s2} = (N_{a2} \times P_{b2}) \ mod\ p$	$k_{s2} = (N_{b2} \times P_{a2}) \ mod\ p$
$(N_{a2} \times N_{b2} \times G_2) \ mod\ p$	$\equiv \quad (N_{b2} \times N_{a2} \times G_2) \ mod\ p$

D. The Pre-processing Operations

In cryptography, there are two important operations confusion and diffusion to make the cipher more secure against attacks. According to *Shannon*, confusion means that every bit on the cipher-text should depend on many parts of the key, concealing the relation between the two and make it as complex as possible. Diffusion means that if it changes only one bit (digit) of the plain-text, statistically, half of the cipher-text should change, and vice versa, therefore, making the cryptanalysis so difficult. This complexity generally implemented through of substitutions and permutations. In the present work, the XOR and $Circular\ Shift$ bitwise operations are applied to achieve diffusion. The diffusion spreads any change in only one bit of the data to the entire cipher-text, so the sensitivity increased. Moreover, these operations are efficient to perform, less time consuming and provide more security against statistical analysis. These advantages are obtained by removing the characteristics that exploit by an

intruder such as repeated plain text values. The following steps demonstrate the above process:

- The Right Circular Shift (RCS) process is implemented on the samples of an audio file($WavFile$). The amount of shifting is determined randomly and denoted by $Rlen$. Each value in the message is shifted with different $Rlen$. The Pseudo-code of RCS is shown as below:

  a. $\boldsymbol{For}\ i = 0\quad \boldsymbol{to}\quad length\ of\ WavFile$
  b. $Rlen = (Rlen + i)\ \boldsymbol{mod}\ 8$
  c. $\boldsymbol{If}\ (Rlen == 0)$
  d. $Rlen = 3$
  e. $RCS(i) = (\ WavFile\ (i) << Rlen\ ) \& 255 \mid WavFile\ (i)\ >> 8$
  f. $\boldsymbol{End}$

- The XOR bit-wise operation is implemented between the first sample in the (RCS) and an initial random value (IV), the output is fed back to XOR operation with the next sample in the (RCS) and so on to produce a chaining cipher. Therefore, if one bit of the plain-audio or the initial value altered, all the cipher-text will be changed. This operation is defined as follows:

$$New_{plain-audio}(0) = RCS(0) \oplus IV \quad (15)$$

$$New_{plain-audio}(i) = RCS(i) \oplus New_{plain-audio}(i-1) \quad (16)$$

Where:
i : $1 \geq i \leq length\ of\ WavFile$.
IV : an initial random value.
$\oplus$: Exclusive-OR operation.

The post-processing, the XOR, and the Left Circular Shift (LCS) operations are implemented on the message, where the amount of shifting and the initial value for XOR are determined randomly.

E. Dual Encryption/Decryption in ECC

The purpose of the dual encryption/decryption process for the audio file is to increase the immunity against any expected attacks. This process is implemented through the encryption of the audio file in two layers. Each layer uses different key pairs (Private and Public). Also, the decryption process is executed in the same manner but in reverse order. The procedure for this process is explained in what follows:

Step 1: Dual Encryption

Alice wants to send an encrypted message to Bob, in the present work the message is an audio file. At the beginning, she converts the audio file data into points on the curve denoted by $\boldsymbol{P_m}\ (\boldsymbol{x}, \boldsymbol{y})$; then encrypt them to produce the cipher text points denoted by $\boldsymbol{C_m}(\boldsymbol{x}, \boldsymbol{y})$. The dual encryption process composed of two phases: the first encryption applies the addition operation between $\boldsymbol{P_m}(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{k_{s1}}(\boldsymbol{x}, \boldsymbol{y})$, the second encryption implements the addition operation between $\boldsymbol{C_m}(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{k_{s2}}(\boldsymbol{x}, \boldsymbol{y})$. Equations (17) and (18), respectively define the aforementioned procedure as below:

$$C_{m1} = P_m + k_{s1}\ mod\ p \quad (17)$$

$$C_{m2} = C_{m1} + k_{s2}\ mod\ p \quad (18)$$

Step 2: Dual Decryption

Bob received the dual encrypted points and needs to decrypt it, Bob implements the decryption operation on the received points by using the subtraction operation, obtain a reflect coordinate of the subtracted point along an x-axis and execute point addition (i.e., $P(x_1, y_1) - Q(x_2, y_2) = P(x_1, y_1) + Q(x_2, -y_2)$) as clarified in (19) and (20) respectively as follows:

$$C_{m1} = C_{m2} - k_{s2}\ mod\ p \quad (19)$$

$$P_m = C_{m1} - k_{s1}\ mod\ p \quad (20)$$

After the dual decryption have been implemented on the received points, converting the decryption points into data of the audio file is the next step.

Here the addition and subtraction operations does not mean simple operations, which they are in algebra, rather they are addition and subtraction, the points in ECC. So, the subtraction operation is the same addition operation, just take the inverse of $y - coordinate$ i.e, the inverse of $P(x, y) = -P(x, -y) = (x, -y + p)$.

VII. Results and Discussion

The experiments run under Windows 10 professional operating system, Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 4 GB random access memory and 64-bit system type. The visual studio 2010 (C# programming language) is used to evolve the proposed system. Table I shows the randomness of different AES key sizes. The inputs setting for generating AES key from the PBKDF2 algorithm are:

- **Password** is composed of 1024-byte that generated from the LCG algorithm, where the inputs to LCG algorithm are ($p_1 = 10247, q_1 = 8161, b_1 = 1, n_1 = 1024$).
- **Salt Value** is " 2018-02-08 02:11:56 AM ".
- Iteration Count is 10000.
- **Length of key** is (128,192,256) bits.

TABLE I. The Randomness Tests for Different Aes Keys from the Pbkdf2 Algorithm

Size of AES keys	Frequency Test	Frequency within a block Test	Run Test
128-bit	0.193931	0.281692	0.079932
192-bit	0.288844	0.598714	0.902214
256-bit	0.900524	0.920130	0.937808

TABLE II. Execution Time And Three Randomness Tests for the H-Aes-Lcg Generator

Size of output from H-AES-LCG generator	Execution Time	Frequency Test	Block Test	Run Test
128-bit	0.576 sec	0.859684	0.857480	0.725681

Table II illustrates the CPU time and three randomness tests for the H-AES-LCG generator, the setting inputs for the family of SHA and AES algorithm are:

- **Password** is " K@vvTHer9064Isaa ".
- **The amount of rotation is 139.**
- **Salt Value** is " 2018-02-08 02:11:56 AM ".

The inputs setting for the LCG algorithm to generate the sequence numbers are ($p_2 = 7933, q_2 = 4093, b_2 = 1, n_2 = 128$).

Fig. 2 demonstrates a comparison between the time of the traditional method for generating all points on a curve and the proposed method for selecting two base points from the curve in a secure manner, that have been tested on the different curves (i.e., different sizes of the domain parameters (p, a, b)).

Tables III and IV illustrate the execution time for dual encryption/decryption process in ECC, different sizes of audio models and various elliptic curves are used to test the results.

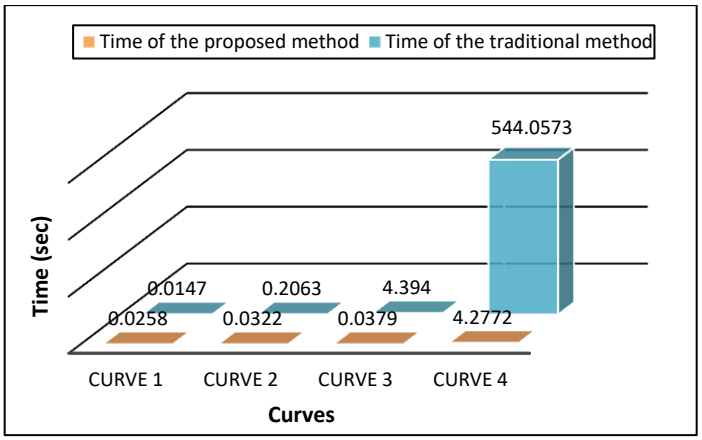

Fig. 2. A comparison between the time of the traditional and proposed method.

TABLE III. EXECUTION TIME FOR THE DUAL ENCRYPTION IN ECC

Prime Number	a,b	Record Time in sec	Data Size KB	Sample Rate Hz	Execution Time in sec
18787	3921,479	0.13	2.79	22050	0.0052
		0.251	5.4	22050	0.0103
		0.323	3.46	11000	0.0062
		0.420	4.5	11025	0.00805
		0.551	5.9	11025	0.008
		0.655	7.04	11025	0.0125
210487	155,180	0.13	2.79	22050	0.0052
		0.251	5.4	22050	0.0111
		0.323	3.46	11000	0.00705
		0.420	4.5	11025	0.009
		0.551	5.9	11025	0.0117
		0.655	7.04	11025	0.0144

TABLE IV. EXECUTION TIME FOR THE DUAL DECRYPTION IN ECC

Prime Number	a, b	Record Time in Sec	Data Size KB	Sample Rate Hz	Execution Time in sec
18787	3921,479	0.13	2.79	22050	0.0055
		0.251	5.4	22050	0.0108
		0.323	3.46	11000	0.0063
		0.420	4.5	11025	0.0081
		0.551	5.9	11025	0.0102
		0.655	7.04	11025	0.0125
210487	155,180	0.13	2.79	22050	0.006
		0.251	5.4	22050	0.01105
		0.323	3.46	11000	0.00705
		0.420	4.5	11025	0.0094
		0.551	5.9	11025	0.0119
		0.655	7.04	11025	0.0151

A. *Security Analysis*

Security analysis is a fundamental process to guarantee the power of the cryptography mechanism; the following metrics are used to evaluate the performance of the proposed system:

1) Entropy

The entropy is considered as one of the most significant measurements to measure the degree of secrecy (randomness). Where, a maximum entropy value in the first order entropy reach to 8 and 16 in the second order entropy, to compute the first order entropy $1^{st}\,E(\tau)$ and second order entropy $2^{nd}\,E(\tau)$ of a source τ are formulated as in the following equations [18][19]:

$$1^{st}\,E(\tau) = -\sum_{i=0}^{n-1} P(\tau_i)\log(P(\tau_i)) \quad (21)$$

$$2^{nd}\,E(\tau) = -\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} P(\tau_i,\tau_j)\log(P(\tau_i,\tau_j)) \quad (22)$$

Where, the total number of τ denotes by n, $P(\tau_i)$ referred to the probability of appearance of τ_i. Table V demonstrates the entropy values for different audio files where the curve parameters are: p=210487, a=155, b=180.

TABLE V. MEASURE THE ENTROPY OF AUDIO FILES BEFORE AND AFTER DUAL ENCRYPTION

		Entropy Before Encryption		Entropy After Encryption	
Audio Data Size in KB	Sample Rate in Hz	1st Entropy	2nd Entropy	1st Entropy	2nd Entropy
2.79	22050	5.30	10.58	7.93	15.81
3.46	22050	6.98	13.79	7.97	15.92
4.5	11025	6.01	11.87	7.94	15.80
5.4	11000	6.82	13.44	7.94	15.83
5.9	11025	5.84	11.50	7.91	15.69
7.04	11025	6.10	12.18	7.96	15.91
16.8	11025	6.26	12.38	7.98	15.97
28.6	16000	5.93	11.84	7.99	15.98
34.2	11000	6.32	12.61	7.99	15.98
47.06	22254	5.93	11.84	7.98	15.94

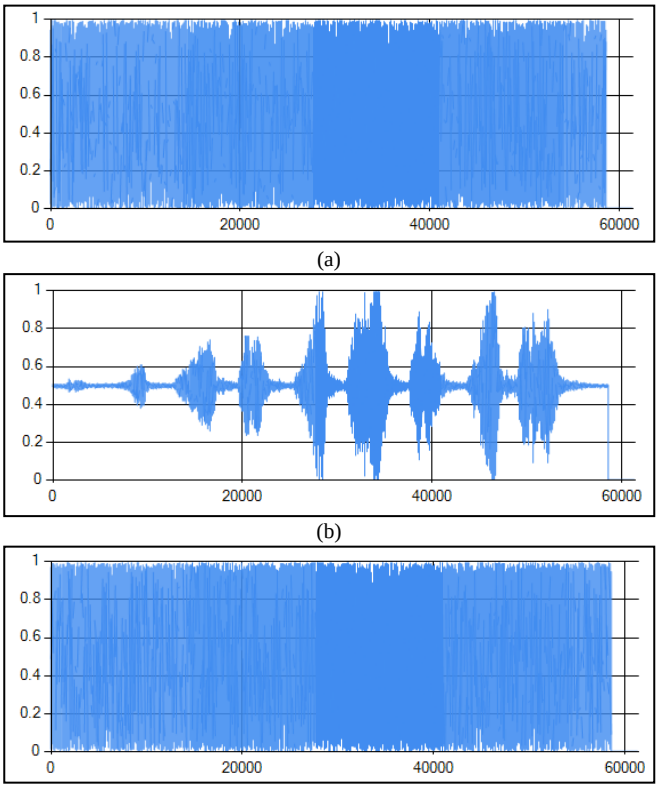

Fig. 3. (a) Cipher-audio; (b) Decrypted with right keys (k_{s1}, k_{s2});(c) Decrypted with key as (k_{s1}-1, k_{s2}-1).

2) Key Sensitivity Analysis

A fundamental feature of a good cryptosystem is a key sensitivity that assures the cryptosystem is secure contra the brute force attack. The key sensitivity is any simple change in the encrypted key gives a different result in recovering the plain-text from a cipher-text. Assume the user A sends a message (audio file) to the user B, the original session keys are $KS_1 = (2481,1143)$, $KS_2 = (4071,391)$. A slight alteration in the private keys (i.e., $n_{b1} - 1, n_{b2} - 1$) produce the different sessions keys ($KS_1 = (2937,907)$, $KS_2 = (3657,864)$ that lead to recovering a different message. Fig. 3 clarifies the decryption operation with the correct and wrong session keys, respectively.

3) Cryptanalysis Attacks

The intruder attempts an analysis of encrypted message by exploiting some characteristics in cipher-text such as repeated cipher or knowing some pairs of plain-text/cipher-text. In addition, he/she tries to identify the nature of the algorithm to recover a plain-text or key [1].

Our Suggested Solution: The diffusion property enhanced in the proposed system by using the Exclusive-OR and Circular Shift that performed to increase strength against cryptanalysis attack. Besides, it uses the dual encryption to increase immunity against any expected attack.

4) Attacks on ECDLP

a) Brute-Force Attack (Exhaustive Search): An intruder seeks to compute all the points that generated from the base point (**G**) until a public key ($\mathbf{P_u}$) is obtained to recognize the private key ($\mathbf{P_r}$). The Seeking time in this type of attacks depends on the order (the number of points in a subgroup (**G**)) denoted by $\mathbf{O_r}$, the large the order ($O_r \geq 2^{80}$) makes the computational infeasible [1],[2].

b) Pohling-Hellman and Pollard's roh Attacks: These attacks in order to accelerate the calculation of the ECDLP, the countermeasure of these attacks the ($\mathbf{O_r}$) should be prime and $\mathbf{O_r} > 2^{160}$ [2].

Our Suggested Solution: It is choosing the public elements in a secure manner to avoid all recognized attacks, instead of a large number of points; it requires a high computational cost.

5) Man in the Middle Attack

When two parties exchange the public keys for each them an intruder intercepts the transmission to occupy the public keys in order to generate a fake shared key between it and each party through exploits the public parameters (prime number, base point, their public keys) [1].

Our Suggested Solution: In our methodology, it frustrates this attack by selecting the global parameters randomly and in a secure manner this helps us even if an intruder knows a public key impossible to obtain the private key because there are unknown parameters to solve ECDLP.

VIII. Conclusions

This paper implements a new design for the ECC cryptosystem in random, efficient and secure manner based on the H-AES-LCG generator function, besides it chooses the domain parameters of the ECC within a given safe mechanism in order to defeat all organized attacks on the ECDLP. The ECDH method is used to make the communication between two parties more secure during the key exchanged process. In addition, the encryption process implemented in double or dual stages, the aim of this is to provide secure transmission for the audio messages and increase immunity against any attack. The proposed methodology is faster, more secure and provides many positive aspects such as enhancements in the key exchange compared with Diffie-Hellman key exchange and ECC performance. In addition, the (H-AES-LCG) is useful for generating encryption key for some algorithms, a slate value for the hash function, a prime number for the RSA algorithm or generates the domain parameters for ECC in a random and safe style.

References

[1] W. Stallings, Cryptography and Network Security Principles and Practice, 6th ed., Pearson Education: United States of America, 2014.

[2] D. Hankerson, S. Vanstone, and A. J. Menezes, Guide to elliptic curve cryptography, Springer Science & Business Media: New York, 2004.

[3] A. Soleymani, M. J. Nordin, and Z. M. Ali, "A novel public key image encryption based on elliptic curves over prime group field," Journal of Image and Graphics, vol. 1, no. 1, 2013.

[4] R. Singh, R. Chauhan, G. Ritu, K. Vinit, and P. Singh, "Implementation of elliptic curve cryptography for audio based application, " International Journal of Engineering, vol. 3, no. 1, 2014.

[5] A. Luma, B. Selimi, and L. Ameti, "Using elliptic curve encryption and decryption for securing audio messages," in Transactions on Engineering Technologies, GC.Yang, SI. Ao, L. Gelman (eds.), Springer, Dordrecht, 2015.

[6] M. Kumar, A. Iqbal, and P. Kumar, "A new RGB image encryption algorithm based on DNA encoding and elliptic curve Diffie--Hellman cryptography, " Signal Processing, vol. 125, pp. 187–202, 2016.

[7] X. Fang and Y. Wu, "Investigation into the elliptic curve cryptography," In Information Management (ICIM), 2017 3rd International Conference on. IEEE, pp. 412–415 , 2017.

[8] K. E. Abdullah, N. H. M. Ali "A secure enhancement for encoding/ decoding data using elliptic curve cryptography," Iraqi Journal of Science, vol. 59, no. 1A, pp. 189–198, 2018.

[9] A. Hafsa, N. Alimi, A. Sghaier, M. Zeghid, and M. Machhout, "A hardware-software co-designed AES-ECC cryptosystem, " In Advanced Systems and Electric Technologies (IC_ASET), 2017 International Conference on. IEEE, pp. 50–54, 2017.

[10] Y. Lin, K. Kang, and Y. Shi, "Research on encryption model based on AES and ECC in RFID, " In Computer Sciences and Applications (CSA), 2013 International Conference on. IEEE, pp. 9–13, 2013.

[11] Q. Dang, "Changes in federal information processing standard (FIPS) 180-4, secure hash standard, " Cryptologia, vol. 37, no. 1, pp. 69–73, 2013.

[12] R. P. McEvoy, M. F. Crowe, C. C. Murphy, and P. W. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs, " In Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on. IEEE p. 6-pp, 2006.

[13] T. Van, C. Henk, and S. Jajodia, Encyclopedia of Cryptography and Security, 2nd ed., Springer Science + Business Media: New York, 2011.

[14] B. Kaliski, "PKCS# 5: Password-based cryptography specification version 2.0," 2000. [Online] Available: https://tools.ietf.org/pdf/rfc2898.pdf

[15] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, "Recommendation for password-based key derivation part 1: storage applications," NIST Special. Publication, vol. 800, pp.132, 2010.

[16] M. Dürmuth, T. Güneysu, M. Kasper, C. Paar, T. Yalcin, and R. Zimmermann, "Evaluation of standardized password-based key derivation against parallel processing platforms," in European Symposium on Research in Computer Security, pp. 716–733, 2012.

[17] A. J. Menezes, T. Okamoto, and S. A. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," IEEE Transactions on Information Theory, vol. 39, no. 5, pp. 1639–1646, 1993.

[18] J. Payingat and D. P. Pattathil, "Pseudorandom bit sequence generator for stream cipher based on elliptic curves," Mathematical Problems in Engineering, vol. 2015, 2015.

[19] A. A. A. El-Latif and X. Niu, "A hybrid chaotic system and cyclic elliptic curve for image encryption," AEU-International Journal of Electronics and Communications, vol. 67, no. 2, pp. 136–143, 2013.