CHAPTER 4
NONCE BASED ELLIPTIC CURVE CRYPTOSYSTEM

4.1 INTRODUCTION

This chapter presents the implementation of ECC by first transforming the message into an affine point on the EC, over the finite prime field. Also, it illustrates the process of encryption / decryption of a text message and image files in spatial domain by enhancing security using CLCG for better random number generation. This enables the breaking of cipher text almost impossible for the brute force attack.

The organization of this chapter is as follows. Section 4.2 describes the concepts of pseudo random number generation. Section 4.3 presents the methodology and algorithm used for the proposed scheme. Section 4.4 gives ECC based image encryption in spatial domain. Section 4.5 provides the security analysis of the proposed scheme and finally section 4.6 summarizes this chapter.

4.2 PSEUDO RANDOM NUMBER GENERATION

The security of cryptographic systems is strongly related to randomness, as the output of these systems must be seen by any adversary as a sequence of random values carrying the secret but revealing absolutely no clues about the precious information. For instance, in public key cryptography
the security depends on hiding a secret key. This key has to be accessible only to its authorized owner. It is computed using a random number generator. Once this generator is corrupted, an adversary is able to reveal the secret key and hence break the system.

As of today, the ability of producing random integers is crucial for the security of most cryptographic applications. Random number generators have been an interest of researchers, since the early days of computing. The favourite definition was given in 1951 by Berkeley professor D. H. Lehmer, a pioneer in computing and, especially, computational number theory: “A random sequence is a vague notion, in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians”.

Lehmer also invented the multiplicative congruential algorithm, which is the basis for many of the random number generators in use today. Random number generators are classified in three categories based on the nature of the randomness source and are presented briefly in the following.

- **True Random Number Generators (TRNG)** where the source is a natural physical phenomenon and the properties of independence and unpredictability of the generated values are guaranteed by physical laws.

- **Unpredictible Random Number Generators (URNG)** are based on the unpredictability inherent to human computer interaction and on the undeterminism introduced by the complexity of the underlying phenomenon.
• **Pseudo Random Number Generators (PRNG)** where the source of randomness is a random initial value, called seed, which is expanded by means of a deterministic recursive formula, providing a modality for generating random sequences using only software methods.

The unpredictability level resumes to the unpredictability of the seed value and the output is completely determined by the starting state of the generator and therefore by the seed. Still, the practical features of PRNGs such as high generation speed, good statistical results and no need for additional hardware devices, made these generators very attractive and are the most widely used random number generators in cryptographic systems.

### 4.2.1 Linear Congruential Generator

A Linear Congruential Generator (LCG) is often used to generate pseudo random numbers. The recurrence relation is defined by \( x_{i+1} = ax_i + b \mod m \) where \( a, b \) and \( m \) are known and \( x_0 \) is seed which is kept secret defines a LCG. If the length of the sequence generated by LCG is \( m \), then it is said to be full period. Also, the LCG is said to have a fixed point this implies that there exists \( i \) such that \( x_{i+1} = x_i \) when \( (1 - a)^{-1} \mod m \) exists. When this occurs the maximum period of the sequence is \( m - 1 \), if the fixed point is not used as an initial condition. The maximum period occurs when the following conditions are satisfied:

- \( b \) and \( m \) are relatively prime.
- \( (a - 1) \) is divisible by every prime factor of \( m \).
- \( (a - 1) \) is divisible by 4 if 4 divides \( m \).

There are many results about the different forms of \( a, b \) and \( m \) and their interrelation to obtain a sequence of pseudo random numbers with large
period (Knuth 1980). For cryptographic use, the numbers generated should not be predictable, if the modulus $m$ is known, then it is easy to solve for $a$ and $b$ given two consecutive numbers in such a sequence. Knuth considers a variation of this generator where the modulus $m$ is a power of two but only the high order bits of the numbers are output.

Some results on congruential generators are as follows. Marsaglia (1968) questions the claims of sufficient random behaviour for sequences produced using LCGs and Reeds (1977), Knuth (1980) and Plumstead (1982). They have all cast considerable doubt on the cryptographic value of sequences generated using the multiplicative congruential generator.

Hastad et al (1985) have argued that LCG is an insecure method. It is possible to recover the seed $x_0$ if at least 1/3 of the leading bits of three consecutive numbers in the sequence are known. A paper by Frieze et al (1988) undermines confidence in techniques to use fragments of integers derived from linear congruences.

While the generation of such sequences can be convenient and there are analytic results which provide assurances on some of the basic properties of the sequences generated, LCGs cannot be recommended for cryptographic use. Surprisingly perhaps, a paper by Lagarias and Reeds (1988) implies that there might be little extra cryptographic security gained by moving to more sophisticated recurrences which involve polynomial expressions. Krawczyk (1990) has extended both this work and that of Plumstead to apply a very general analysis to the problem of predicting sequences generated using different forms of polynomial recurrence relation.
4.2.2 Comparative Linear Congruential Generator

Comparative or Coupled Linear congruential generator (Katti et al 2008) is defined as follows:

\[
\begin{align*}
    x_{i+1} &= ax_i + b \mod m \\
    y_{i+1} &= cy_i + d \mod m \\
    z_{i+1} &= \begin{cases} 
        1 & \text{if } x_{i+1} > y_{i+1} \\
        0 & \text{otherwise}
    \end{cases}
\end{align*}
\]

(4.1)

**Example 4.1**: Let \(a = 5, b = 5, c = 3, d = 2\) and \(m = 8\). Both sequences \(x_i\) and \(y_i\) have a period of 8 and are hence full period. If the initial condition or the seed is \((x_0, y_0) = (3, 6)\), then the sequences are,

\[
\begin{align*}
    \{x_i\} &= (4, 1, 2, 7, 0, 5, 6, 3) \\
    \{y_i\} &= (4, 6, 4, 6, 4, 6, 4, 6)
\end{align*}
\]

The bit sequence \(z_i\) therefore is

\[
\{z_i\} = (0, 0, 0, 1, 0, 0, 1, 0)
\]

Assume \(a, b, c, d, m\) are known and the seed \((x_0, y_0)\) is secret. Let the way of guessing the initial condition or seed \((x_0, y_0)\) of CLCG be given the bit sequence \(z_i\). Then, the numbers \((x_i, y_i)\) are taken to be positive integers between 0 and \((m - 1)\) for the computation of \(z_{i+1}\). This makes the computation of \(x_{i+1}\) and \(y_{i+1}\), from \(z_{i+1}\) very difficult. It is easy to see that the \(k^{th}\) output of an LCG \(x_{i+1} = ax_i + b \mod m\), is given as:

\[
x_k = a^k x_0 + b \sum_{i=0}^{k-1} a^i \mod m
\]

(4.2)

This implies that if the \(k^{th}\) output of the CLCG is \(z_k\), then the following inequality holds based on whether \(z_k\) is 1 or 0.
\[ a^k x_0 + b \sum_{i=0}^{k-1} d_i \mod m > c^k x_0 + d \sum_{i=0}^{k-1} c_i \mod m \text{ if } z_k = 1 \]

\[ a^k x_0 + b \sum_{i=0}^{k-1} a_i \mod m \leq c^k x_0 + d \sum_{i=0}^{k-1} c_i \mod m \text{ if } z_k = 0 \] (4.3)

The number of inequalities that can be set as above can be the maximum number of bits that are known from the \( z_k \). For example if \( u \) bits of the output \( z_k \) are known, then only \( u \) number of inequalities can be set up as \( E_k \), \( 1 \leq k \leq u \), where \( E_k \) is an inequality of the form described above.

### 4.2.3 Advantage of the CLCG

Solving of the CLCG problem is difficult since it requires solving inequalities of the form specified by equation (4.3). If the inequalities are converted into equalities, then lattice like methods can be used to obtain the solutions. The main difficulty in solving with inequalities is the fact that there is no ordering over integers modulo \( m \). This is because \( x \mod m \) can be both less than and greater than another integer \( y \). The reason for this is the fact that \( y \) and \( y - m \) are congruent.

Another difficulty with an inequality of the form given in equation (4.3) is the fact that it cannot be manipulated. This implies that this inequality cannot be converted to the following form:

\[ ax + b - cy - d \mod m > 0 \mod m \]

Even if converted, it would lead to incorrect solutions for \((x, y)\). Also, lattice methods for solving modular equalities lead to exponential complexity with respect to the size of the problem (\( \log m \) is the input size). In the following
section, it is demonstrated how the CLCG system can be extended to enhance the security of the elliptic curve cryptosystem further.

4.3 ECC AND CLCG

ECC based security offers a similar level of security that can be achieved with shorter keys than existing methods which are based on the difficulties of solving discrete logarithms over integers or integer factorizations.

4.3.1 EC Point Generation

Initially, to do operations with EC points in order to encrypt and decrypt, the points are to be generated. The algorithm ‘genPoints’ describes the process of generating the points for the given parameters \( a, b \) and \( p \).

**Algorithm genPoints**\((a, b, p)\)

// Input : Elliptic curve parameters \( a, b \) and \( p \)

// Output : Elliptic curve points

\{
  x = 0;
  While (\( x < p \))
    \( y^2 = (x^3 + ax + b) \mod p \);
    If (\( y^2 \) is a perfect square in GF(\( p \)))
      Output \((x, \sqrt{y}), (x, -\sqrt{y})\);
    \( x = x + 1; \)
\}
For demonstration purposes typical EC is represented by
\[ y^2 \mod 37 = x^3 + x + 1 \mod 37 \] where \( a = 1, b = 1 \) and \( p = 37 \). The generated
points on the curve can be found as shown in Table 4.1.

### Table 4.1 Elliptic curve points over \( E(F_{37}) \)

<table>
<thead>
<tr>
<th>Elliptic curve points</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
</tr>
<tr>
<td>(0, 36)</td>
</tr>
<tr>
<td>(1, 15)</td>
</tr>
<tr>
<td>(1, 22)</td>
</tr>
<tr>
<td>(2, 14)</td>
</tr>
<tr>
<td>(2, 23)</td>
</tr>
<tr>
<td>(6, 1)</td>
</tr>
<tr>
<td>(6, 36)</td>
</tr>
<tr>
<td>(8, 15)</td>
</tr>
<tr>
<td>(8, 22)</td>
</tr>
<tr>
<td>(9, 6)</td>
</tr>
<tr>
<td>(9, 31)</td>
</tr>
<tr>
<td>(10, 7)</td>
</tr>
<tr>
<td>(10, 30)</td>
</tr>
<tr>
<td>(11, 14)</td>
</tr>
<tr>
<td>(11, 23)</td>
</tr>
<tr>
<td>(13, 18)</td>
</tr>
<tr>
<td>(13, 19)</td>
</tr>
<tr>
<td>(14, 13)</td>
</tr>
<tr>
<td>(14, 24)</td>
</tr>
<tr>
<td>(17, 11)</td>
</tr>
<tr>
<td>(17, 26)</td>
</tr>
<tr>
<td>(19, 16)</td>
</tr>
<tr>
<td>(19, 21)</td>
</tr>
<tr>
<td>(21, 12)</td>
</tr>
<tr>
<td>(21, 25)</td>
</tr>
<tr>
<td>(24, 14)</td>
</tr>
<tr>
<td>(24, 23)</td>
</tr>
<tr>
<td>(25, 0)</td>
</tr>
<tr>
<td>(25, 0)</td>
</tr>
<tr>
<td>(26, 18)</td>
</tr>
<tr>
<td>(26, 19)</td>
</tr>
<tr>
<td>(27, 8)</td>
</tr>
<tr>
<td>(27, 29)</td>
</tr>
<tr>
<td>(28, 15)</td>
</tr>
<tr>
<td>(28, 22)</td>
</tr>
<tr>
<td>(29, 6)</td>
</tr>
<tr>
<td>(29, 31)</td>
</tr>
<tr>
<td>(30, 13)</td>
</tr>
<tr>
<td>(30, 24)</td>
</tr>
<tr>
<td>(31, 1)</td>
</tr>
<tr>
<td>(31, 36)</td>
</tr>
<tr>
<td>(33, 9)</td>
</tr>
<tr>
<td>(33, 28)</td>
</tr>
<tr>
<td>(35, 18)</td>
</tr>
<tr>
<td>(35, 19)</td>
</tr>
<tr>
<td>(36, 6)</td>
</tr>
<tr>
<td>(36, 31)</td>
</tr>
</tbody>
</table>

The base point \( G \) is selected as \((0, 1)\). Base point implies that it has
the smallest \((x, y)\) coordinates which satisfy the EC.

#### 4.3.2 Nonce Generation

In the ECC method, first generate a nonce that is a random integer,
which needs to be kept secret. Here, the nonce in the elliptic curve
cryptosystem is generated using a CLCG. Let the source be \( A \) and destination
be \( B \). The private key of the user \( B \) be \( n_B \) and a random number chosen by
user \( A \) as \( k (k < p) \). The values of \( k \) and \( n_B \) are generated by a random number
generator that is CLCG to give credibility.

Let \( a = 5, b = 4, c = 3, d = 1 \) and \( m = 8 \). Both sequences \( x_i \) and \( y_i \)
have a period of 8 and are therefore full period. If the initial condition or the
seed is \((x_0, y_0) = (1, 0)\), then the sequences are:
\{x_i\} = (1, 1, 1, 1, 1, 1, 1, 1)
\{y_i\} = (1, 4, 5, 0, 1, 4, 5, 0)
The bit sequence \(z_i\) therefore is:
\{z_i\} = (0, 0, 0, 1, 0, 0, 0, 1)
The decimal equivalent values are \(n_B = 17\). Similarly, the value \(k\) is generated as 13.

4.3.3 Key Deployment

Both the users in the cryptosystem agree upon domain parameters \((a, b, p, G, n)\) of ECC. \(G\) is called base or generator point and \(n\) is the order of \(G\). Now, user \(B\) selects a random number \(n_B < n\) as private key and calculates the public key using the algorithm ‘\textit{distributePublicKey}’. Therefore, the public key of user \(B\) is evaluated by \(P_B = n_B G = 17(0, 1) = (21, 12)\). Similarly, user \(A\) generates a random number \(n_A < n\) as private key and calculates the public key \(P_A\). Then, both the users distribute their public key to other users.

\textbf{Algorithm distributePublicKey( )}

// Input : Domain parameters \((a, b, p, n, G)\)
// Output : Public key \(P_A\) and \(P_B\), private key \(n_A\) and \(n_B\).
// Let \(U_A\) and \(U_B\) be legitimate users.
{
    If \((U_A)\)
    {
        Select \(n_A < n\) using CLCG
        Compute \(P_A = n_A G\) // \((P_A, n_A)\) Key pair for \(U_A\)
        Send \((P_A, U_B)\) // Send the public key of \(U_A\) to \(U_B\)
    }
}
If \((U_B)\)
{
    Select \(n_B < n\) using CLCG
    Compute \(P_B = n_B G\) // \((P_B, n_B)\) Key pair for \(U_B\)
    Send \((P_B, U_A)\) // Send the public key of \(U_B\) to \(U_A\)
}

4.3.4 Encoding Plaintext as Points on an Elliptic Curve

Assuming, user \(A\) wants to encrypt and transmit the message to user \(B\), before encrypting, the following steps are carried out to encode the plaintext into an EC point.

- First each character in the message is represented by its ASCII value. For example, the ASCII value of the character ‘#’ is 35.
- In each of these, the ASCII values are transformed into an affine point on the EC over finite prime field, by using a starting point called \(P_m\). For this, \(P_m\) is another affine point, which is picked out of a series of affine points evaluated for the given EC. The base point \(G\) could be retained as \(P_m\) itself. However, for the purpose of individual identity, \(P_m\) is chosen to be different from \(G\). Let \(P_m = (1, 15)\). Varying values of \(P_m\) can be chosen as part of an exercise to work with ECC process on the given EC.
- The encoded message \((P_{m1})\) is computed as the scalar multiplication of ASCII value and the randomly chosen point \(P_m\) on EC. Therefore, \(P_{m1} = 35(1, 15) = (2, 14)\). In this way, the entire message becomes a sequence of EC points.
The encoded message \((P_{m1})\) is an affine point which should fit into the EC. This conversion is done for two reasons. Initially, the ASCII key representation of the text message is mapped into a \((x, y)\) coordinate of the EC. Secondly, it will be completely hidden from the hackers. These steps are introduced to add some level of complexity even before the message is encrypted using ECC.

### 4.3.5 Encryption Operation

The algorithm ‘encryptNoncebasedECC’ describes the process of nonce based encryption on EC field. The encoded message \((P_{m1})\) has been encrypted by the application of the ECC method. For this, the generated random number \(k\) using CLCG and the recipient’s public key \(P_B\) are multiplied, which is carried out with a series of doubling and additions, depending on the value of \(k\). Efficient procedure can be adapted for optimal number of doublings and additions.

\[
kP_B = 13(21, 12) = (21, 12)
\]

\[
P_{m1} + kP_B = (2, 14) + (21, 12) = (30, 24)
\]

\[
kG = 13(0, 1) = (0, 1)
\]

The encrypted message is derived by adding \(P_{m1}\) with \(kP_B\), that is, \(P_{m1} + kP_B\). This yields a set of \((x_2, y_2)\) coordinates. Then, \(kG\) is included as the first element \((x_1, y_1)\) of the encrypted version. Hence, the entire encrypted version for purposes of storing or transmission consists of two sets of coordinates as follows:

\[
C_m = (C_1, C_2) = (kG, P_{m1} + kP_B)
\]

\[
C_1 = (x_1, y_1) = kG
\]

\[
C_2 = (x_2, y_2) = P_{m1} + kP_B
\]
Therefore, the encrypted version of the message $C_m$ as $((0, 1), (30, 24))$ where $x_1 = 0$, $y_1 = 1$, $x_2 = 30$ and $y_2 = 24$. Thus the modified plaintext has been encrypted by application of the ECC method.

**Algorithm encryptNoncebasedECC(file)**

// **Input** : EC domain parameters $(p, E, G, n)$, public key $P_B$, plaintext $m$.

// **Output** : Cipher text $(C_1, C_2)$.

{  
    While (!eof (file))
    {
        Represent the message $m$ by its ASCII value as $m_a$
        Select random point on EC as $P_m$
        Compute $P_{ml} = m_a P_m$ as a point in $E(F_p)$
        Select $k$ using CLCG
        Compute $C_1 = kG$
        Compute $C_2 = P_{ml} + kP_B$
        Write the cipher text $(C_1, C_2)$ to output file
    }
    Return output file
}

The selection of random i.e. the secret number dictates the complexity of encryption algorithm for breaking. For this, a novel random number selection process based on CLCG is introduced, which is not a part of any of the existing work on ECC.
4.3.6 Decryption Operation

The algorithm ‘decryptNoncebasedECC’ depicts the method of nonce based decryption on EC field. In order to pull out $P_{ml}$ from $P_{ml} + kPB$, recipient’s private key $n_B$ is scalar multiplied with $C_1$ so that, $n_BC_1 = n_BkG = kPB$. Subtract this from $C_2$, to get $P_{ml}$ that is, $P_{ml} = C_2 - n_BC_1 = P_{ml} + kPB - n_BkG$.

$$n_BkG = 17(0, 1) = (21, 12)$$
$$P_{ml} = (30, 24) - (21, 12) = (2, 14)$$

This subtraction is nothing but another ECC procedure involving doubling and addition with difference having its $y$ coordinate preceded by a minus sign. Hence, the determination of the value $P_{ml}$ follows the same point arithmetic procedure. Now apply discrete logarithm concept to get the ASCII value of character $m_a$ in a message $m_a (1, 15) = (2, 14)$. Then, decode the ASCII value of characters in a message into the original message. Since the ASCII value of '#' is 35, the character ‘#’ is retrieved.

**Algorithm decryptNoncebasedECC(file)**

// Input : Domain parameters ($p$, $E$, $G$, $n$), private key $n_B$, cipher text ($C_1$, $C_2$).

// Output : Plaintext $m$.

{  
   While (!eof (file))
   {  
      Compute $P_{ml} = C_2 - n_BC_1$
      Extract $m$ from $P_{ml}$ using discrete logarithm
      Write the plaintext $m$ to output file}
}
The security of the proposed nonce based cryptosystem lies in the fact that if only $G$ and $P_B$ are known to the opponent, it is difficult to determine the private key $n_B$ of recipient because of ECDLP. This fact enables the nonce based cryptosystem to remain secure while keeping the size of the field small. Furthermore, it is imperative that the private key $n_B$ is generated by CLCG in the sense that the probability of any particular value being selected must be sufficiently small to preclude an adversary from gaining advantage through optimizing a search strategy based on such probability.

4.4 ECC APPLIED ON IMAGE

The previous sections have demonstrated the generation of EC points, encryption of a character and decryption of the same with the aid of a simple example. A random number which is lesser than a small prime number was selected. Now, the nonce based elliptic curve text encryption is extended to an image encryption in spatial domain. Here, the encryption algorithm is applied in the pixel values of an image directly. National Institute of Standards and Technology (NIST) has listed a collection of recommended elliptic curves, with the private key lengths and underlying fields. It specifies how to represent field elements and provides for random generated curves and selected curves.

The Appendix 1 provides the complete properties of the recommended EC domain parameters over $F_p$. Here, pseudo random curve 256-bit EC over $F_p$ is chosen for image encryption. The various EC parameters are $p, a, b, S, r, x, y, h$ and $n$ which are implemented as Big Integer,
since they are all 256 bits wide and are detailed as Appendix 1. Table 4.2 illustrates the original image size and execution time i.e., both encryption time and decryption time taken for various image files using ECC.

Table 4.2 Encryption and decryption time of various images

<table>
<thead>
<tr>
<th>Image File name</th>
<th>Image Size (rows, cols, colors)</th>
<th>Enc.Time (ms)</th>
<th>Dec.Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building1.jpg</td>
<td>132x93x3</td>
<td>7078</td>
<td>4156</td>
</tr>
<tr>
<td>Bird1.jpg</td>
<td>274x373x3</td>
<td>38750</td>
<td>22656</td>
</tr>
<tr>
<td>Bird2.jpg</td>
<td>390x545x3</td>
<td>64171</td>
<td>37907</td>
</tr>
</tbody>
</table>

Figure 4.1 illustrates the actual encryption and decryption time of various images. X-axis shows the size of the image in bytes and Y-axis shows the time taken in milliseconds. The decryption time is lesser than the encryption time due to the high computational complexity involved in the encryption process.

![Graphical representation of encryption and decryption time](image.png)

**Figure 4.1** The graphical representation of encryption and decryption time (millisecond) Vs the data size (bytes).
From Table 4.2, it can be observed that the encryption and decryption time realization is high, in real time applications such as multimedia data. Hence, strategies must be adopted to lower the execution time or parallel / distributed computing environment can be used to enhance the computing power.

Even though the encryption and decryption time depends on the size of the data, the ECC based encryption system depends on an iterative approach where in a large prime number is involved in the decision of the number of iterations. The large prime number is generated nearer to a random number (based on the current time in milliseconds in the proposed work). Due to this the numbers of iterations vary for each data and hence it exhibits the non-linear relation between the computation time and the data size.

4.5 SECURITY ANALYSIS

Recent computing power is capable of breaking encryption schemes in a real time if the system is not designed to look into these issues. Hence, a good encryption scheme should keep away from the possible attacks. The attacks are varying in nature such as statistical attack, brute force attack and so on. Hence, analysis of encryption schemes such as key space analysis and statistical analysis ensures right development of the security system.

4.5.1 Key Space Analysis

Security level in the cryptosystem is expressed by the key size in bits, meaning that for a key (scalar $k$) of size $l$, one would require $2^l$ steps to break the cryptosystem. Key space is defined as a set of all possible keys. The
key space that is being used for encryption must be large enough to prevent the brute force attackers / intruders.

In the proposed nonce based elliptic curve cryptosystem, the key size as per specification given in 256-bit pseudo random EC over $F_p$ is 256 bits. Therefore, it has $2^{256}$ different combinations of secret key to break the cryptosystem. Hence, this large key space is sufficient for the proposed nonce based elliptic curve cryptosystem which is immune to all kinds of brute force attacks.

4.5.2 Statistical Analysis

Statistical analysis generally depends on the measure of the randomness of the cipher media. Randomness means non-order / non-coherence in a sequence of symbols / steps. Also, it works on the relative frequency of the occurred cipher text. Since a random number generator based on CLCG is used, the strength of the proposed encrypted system is high compared to the existing methods because CLCG pass all NIST pseudo randomness test with high degree of consistency.

4.6 SUMMARY

In this chapter, the encryption and decryption of the text message using nonce based elliptic curve cryptosystem is demonstrated with an example. Also, the work is extended to the image applications. For the text applications, each character in the message is represented by its ASCII value. Each of these ASCII value representations of the character is transformed into an affine point on the EC. This transformation introduces nonlinearity in the character thereby completely disguising its identity. Then, the encoded
character of the message is encrypted by the ECC technique. The decryption of the encrypted message is itself quite a difficult task, unless the knowledge about the private key $n_B$, the secret integer $k$ and the affine point $P_{m1}$ is known.

It is shown that CLCG is a good candidate for nonce generation in the elliptic curve cryptosystem in terms of both security and computational efficiency thereby the drawbacks of the single LCG is removed. Therefore, the task of breaking CLCG is computationally infeasible for very large value of randomness. Transformation of the plaintext ASCII value by using an affine point and introducing the modified random number generator are the contributions of the present work. The next chapter, provides a secure group key computation scheme based on EC operation that can be used for group oriented applications.