Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 257904, 16 pages
http://dx.doi.org/10.1155/2015/257904

Research Article

Pseudorandom Bit Sequence Generator for Stream Cipher Based on Elliptic Curves

Jilna Payingat and Deepthi P. Pattathil

Department of Electronics and Communication Engineering, National Institute of Technology, Calicut, Kerala 673 601, India

Correspondence should be addressed to Jilna Payingat; jilnaprakash@yahoo.co.in

Received 1 April 2015; Revised 17 August 2015; Accepted 24 August 2015

Academic Editor: David Bigaud

This paper proposes a pseudorandom sequence generator for stream ciphers based on elliptic curves (EC). A detailed analysis of various EC based random number generators available in the literature is done and a new method is proposed such that it addresses the drawbacks of these schemes. Statistical analysis of the proposed method is carried out using the NIST (National Institute of Standards and Technology) test suite and it is seen that the sequence exhibits good randomness properties. The linear complexity analysis shows that the system has a linear complexity equal to the period of the sequence which is highly desirable. The statistical complexity and security against known plain text attack are also analysed. A comparison of the proposed method with other EC based schemes is done in terms of throughput, periodicity, and security, and the proposed method outperforms the methods in the literature. For resource constrained applications where a highly secure key exchange is essential, the proposed method provides a good option for encryption by time sharing the point multiplication unit for EC based key exchange. The algorithm and architecture for implementation are developed in such a way that the hardware consumed in addition to point multiplication unit is much less.

1. Introduction

Wireless sensor networks have a wide range of applications such as habitat monitoring, home automation, and military and medical applications [1, 2]. Compared to conventional wireless networks, wireless sensors have limited resources that demand cryptographic solutions with reduced complexity. Due to the resource constrained nature, WSNs employ symmetric key encryption techniques that necessitate key management schemes suitable for these constrained applications. A detailed analysis of the proposals available in literature for key distribution shows that only the one-way function based schemes can provide security when a node is compromised in the initialisation phase. The light weight cryptographic algorithms based on random key predistribution [3, 4], polynomial based key distribution [5], and so forth offer no security in this scenario. All these schemes assume that a node cannot be compromised in the initialisation phase which is not true. For such schemes the time-out period of the initialisation phase cannot be kept large because it increases the probability that a node is compromised in the initialisation phase. On the other hand if the time-out period is kept small, then the connectivity of the network is affected. So there exists a trade-off between security and connectivity in such schemes whereas for the one-way function based methods no such trade-off exists. Thus for high security applications like military or medical applications, the one-way function based key management schemes are preferred.

Elliptic curve cryptography (ECC) is a promising solution in such scenarios because of the increased security per bit of the key, compared to other one-way functions [6–8]. All sensor networks require a message authentication code (MAC) and pseudorandom generator for secret key establishment and data transfer. If these two functions are implemented using standalone algorithms like SHA and AES along with ECC for key exchange, then the overall hardware complexity of the system will be very high. If the point multiplication unit used for key exchange can be time shared to perform the other two functions, the complexity of the entire system can be reduced. In this paper, an EC based pseudorandom sequence generator is proposed. The proposed method is developed in such a way that the hardware required to build the pseudorandom bit sequence generator in addition to EC point multiplication unit is much less. So this provides a highly suitable option for light weight encryption in systems using EC based key exchange.

2. Related Works

In [9] Blum and Micali introduced the concept of generating CSPBSG (cryptographically strong pseudorandom bit sequence generator) using a cryptographic one-way function. Since then there are several approaches which make use of the cryptographic one-way operation of EC point multiplication for constructing stream ciphers. The concept of linear congruential generator is extended to EC and a generator for pseudorandom bit sequence from points on the elliptic curve is described in [10]. The sequence is proved to have good randomness properties but the security is dependent on the secrecy of the base point P. In 2000 Shparlinski introduced the Naor-Reigngold generator [11]. The seed is a vector of random integers given as $(e_0, e_1, \ldots, e_{n-1})$. The key for the ith iteration is $k_i = (e_0^{i_0}, e_1^{i_1}, \ldots, e_{n-1}^{i_{n-1}})$ where $i = (i_0, i_1, \ldots, i_{n-1})$. The output bit sequence is generated by applying truncation function to the x-coordinate of the point (k_iP) in each iteration. The security of the random number generator is vested in ECDLP but the number of input random bits required to generate the sequence is high. The elliptic curve power generator (ECPG) [12] published in 2005 makes use of an integer e as the random seed. The ith iteration key $k_i = e^i$ and the output point is k_iP. The bit sequence is generated by truncating the x-coordinate of the output point. The periodicity of the generator is very low and the period reveals some of the properties of the seed e. The pseudorandom sequence generator based on EC published in [13] makes use of a single point multiplication in each iteration. The output bit sequence is the x-coordinate of the output point sequence which is generated as $Q_i = k_iP$ where $k_i = x_{i-1} + (i-1)$. The sequence is proved to have good statistical properties but the security analysis is not done. The dual EC generator proposed by Elaine Barker and John Kelsey was chosen as standard random number generator by NIST [14]. The random seed is an integer e and makes use of two points P and Q on the EC. The iteration key $k_i = X(k_{i-1}P)$ and the output point is k_iQ. The output of the generator is $t(X(k_iQ))$ where t is the truncation function. The periodicity of the generator is found to be very low because of the method used for generating the iterating key. To increase the periodicity the iterating key is modified as $k_i = X(k_{i-1}P) + i$. But then it is found that as $X(k_iP) \ll i$ the sequence becomes independent of the seed. New stream cipher designs based on EC are proposed in [15]. The three algorithms proposed are derived from the dual EC generator, linear congruential generator, and the Naor-Reigngold generator. The authors have proved that the sequences generated using these algorithms have large periodicity, but the hardware complexity is high. A stream cipher based on ECDLP is described in [16]. The method consists of three stages of operation: (i) initialization stage, (ii) key stream generation, and (iii) encryption stage. The mapping of key to a point on the EC is carried out in the initialization stage which increases the hardware and computational complexity and makes it less suitable for resource constrained applications. A key generation based on EC over finite prime field is published in 2012 [17]. The output is generated by truncating the x-coordinate of the point $Q_i = r_iP_i$ where r_i is the random value from the LFSR and P_i's are points on the EC. The method described requires a lot of parameters, that is, the feedback polynomial of LFSR, seed value, EC parameters, and so forth, to be kept secret. The security of the sequence depends entirely on the secrecy of these parameters.

3. Mathematical Background

Elliptic curves (EC) over a field F are set of points (x, y) that satisfy the Weierstrass equation given as

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \quad (1)$$

The variables x, y and the constants a_1, a_2, a_3, a_4, and a_6 are all elements of the field F. The definition of EC also includes a single element called the "point at infinity" or the "zero point" denoted by O.

The set of points on an EC is an abelian group under an addition operation, and O is the identity element. The addition operation is defined such that if P, Q, and R are three points on EC lying on the same straight line, then $P+Q+R = O$.

The cryptographic operation on EC is point multiplication. Given an integer "k" and a point P on the EC computing "$R = kP$" where "R" is a new point on the EC is called point multiplication. This is a one-way function because computing "kP" is easy but given R and P finding "k" is difficult. This is known as the elliptic curve discrete logarithm problem (ECDLP). The EC defined over GF(2^m) (Galois field) are more suitable for hardware implementation. These curves are classified as super singular and nonsuper singular curves. The MOV (Menezes, Okamoto, and Vanstone) reduction method shows that ECDLP is harder in nonsuper singular curves [19]. Point addition and point doubling are the two mathematical operations defined on an EC. The point multiplication is done by repeated point addition and doubling.

Rules for point addition and point doubling on nonsuper singular curves over GF(2^m) are as follows.

(1) Point addition: if $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, then $P + Q = R = (x_3, y_3)$ is

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_1 + x_2 + a_2,$$
$$y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1. \quad (2)$$

(2) Point doubling: if $P = (x_1, y_1)$, then $2P = R = (x_3, y_3)$ is given by

$$x_3 = x_1^2 + \frac{a_6}{(x_1)^2},$$
$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3. \quad (3)$$

The security of EC point multiplication is increased by truncating the "m" bit representation of the x-coordinate of the point to "k" bits and giving out as output. In [20] the authors have proved that, for an EC defined over GF(2^m), if the "m" bit representation of the x-coordinate is truncated to "k" bits, the statistical distance between the output of truncation function and a random "k" bit string is 2^{k-m}. Hence it is hard to determine whether a sequence is generated by truncating the x-coordinate of a point on the EC or if it is chosen uniformly at random. This is known as truncation point problem (TPP).

4. Analysis of EC Based Pseudorandom Sequence Generators

In this section, the analysis of various EC based pseudorandom sequence generators available in literature is carried out. For analysis the EC chosen is $E : y^2 + xy = x^3 + x^2 + 1$ defined over GF(2^m). A point (3, 39) on the EC means (α^3, α^{39}) where α is root of the polynomial used for constructing the finite field.

4.1. EC Based Linear Congruential Generator. In EC based linear congruential generator, the output point sequence is generated as iP where i is the iteration number and P is a point on the elliptic curve which is kept secret. The sequence passes through the complete cyclic subgroup of point P. Thus the period of the sequence reveals the order of point P which reduces the search space to a smaller value. The symmetric properties of the generated sequence also help to make cryptanalysis easier. The detailed cryptanalysis of this generator is given in [15]. Though the sequence has a good linear span and statistical properties, it cannot be used as key stream for stream cipher because of reduced security.

4.2. PBSG-B. PBSG-B in [15] is a modification of the EC based linear congruential generator such that the periodicity is independent of the order of point P and the output sequence does not have any symmetric properties which makes the cryptanalysis easier. For security, the authors assume that both point P and the seed of the LFSR are kept secret. But the analysis shows that the security is dependent only on secrecy of point P, which cannot be quantified.

For analysis, assume point P is known to the attacker. The attacker can generate the entire sequence by choosing an arbitrary value as seed of LFSR. As the LFSR passes through the same sequence of states, the output bit sequence generated will only be a shifted version of the original sequence. If a part of the key stream is known to the attacker

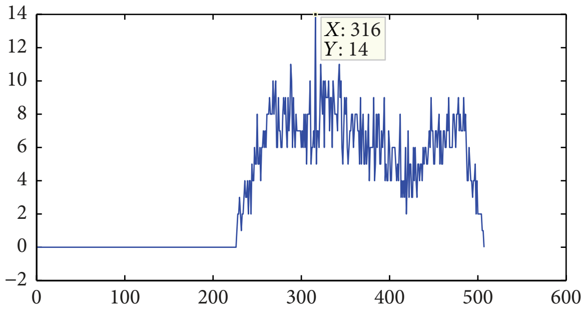

FIGURE 1: Cross-correlation function.

(considering a known plain text attack), the shift can be easily computed from the plot of the cross-correlation of the generated sequence and the known bit sequence. Let the EC be defined over GF(2^7). The feedback polynomial of the LFSR is x^7+x^4+1, $P = (3, 39)$, and the initial seed $k_0 = 126$. Assume the attacker knows a few initial bits of the sequence, that is, 1010001101000101001111001011 0. Let the initial seed chosen by the attacker to generate the sequence be 42. The plot of cross-correlation between the sequence generated with $k_0 = 42$ and the known sequence is given in Figure 1. From the position of the peak value in the cross-correlation function, the position of the known sequence (316 − 254 = 62) and hence the LFSR seed (LFSR value at the 31st iteration = 126) can be easily determined.

4.3. Elliptic Curve Power Generator. The output point sequence in ECPG is generated as $Q_i = k_i P$, where $k_i = e^i$ and $e \in$ GF(2^m) is the initial secret key. The output point sequence is the truncated x-coordinate of the point Q_i. Let l be the order of point P. The period of the sequence is determined by the order of point P and the seed e and is given as $T = \text{ord}_l(e)$. Thus, the periodicity of the sequence is much less compared to the order of point P. Moreover, knowledge of the periodicity of the sequence reduces the search space for the seed e.

The following analysis shows that the security of the generator is also very low. Let α be the primitive element of the finite field over which the EC is defined. Generate a lookup table with i and $\alpha^i P$ as entries of the table. Let α^j be the initial seed e. Assume that the attacker has identified two consecutive output points from its truncated version. Let $e^i = \alpha^{k_1}$ and $e^{i+1} = \alpha^{k_2}$ be the corresponding iteration keys identified from the lookup table. Then, $\alpha^{k_2} = \alpha^{k_1} * e = \alpha^{k_1} * \alpha^j$. This implies $k_2 = k_1 + j$ or $j = k_2 - k_1$. Thus the initial secret seed $e = \alpha^j$ can be easily determined.

4.4. EC Based Random Number Generator. The random number generator proposed in [13] has reduced latency and increased periodicity with a single point multiplication operation in each iteration. The output point sequence is $Q_i = k_i P$ and $k_i = (i - 1) + x_{i-1}$ where x_{i-1} is the x-coordinate of Q_{i-1}. The random number generator has good statistical properties and high periodicity. But it is found that the output

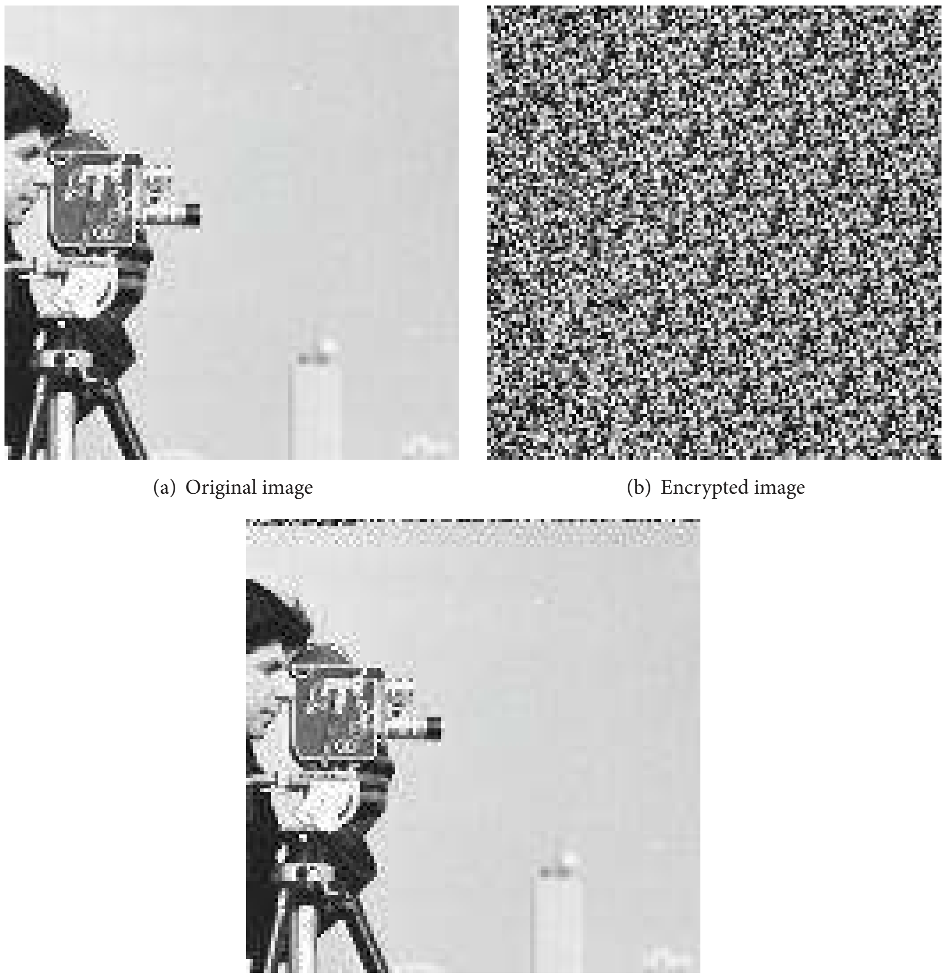
(a) Original image (b) Encrypted image (c) Image decrypted with wrong key

Figure 2: Original image, encrypted image, and image decrypted with the wrong key.

sequence becomes independent of the key as the iteration number increases.

For illustration, an image is encrypted using the pseudorandom sequence generated with this algorithm. The EC is defined over GF(2^{11}) and the output is truncated to 5 bits. The initial key seed is chosen as $k_0 = 104$. The original and encrypted images are shown in Figures 2(a) and 2(b), respectively. The encrypted image is now decrypted with a different seed $k_0 = 84$. The decrypted image is shown in Figure 2(c). From the image it is clear that, except for a very small region, the image could be decrypted with a different key. This shows that the sequence generated with this algorithm becomes independent of the initial key seed after a few iterations and is insecure as key stream for stream cipher.

4.5. Dual EC Generator. The dual EC generator in [14] is published as a standard random number generator by NIST in 2007. The generator makes use of two points on the EC P and Q, one for generating the iterating key as $k_i = X(k_{i-1}P)$ and the other for generating the output bit sequence as $t(X(k_iQ))$ where t is the truncation function. But it is found that if points P and Q are not properly chosen, then the periodicity of the sequence is very low. Consider EC defined over GF(2^7). Let $P = (22, 10)$ and $Q = (11, 5)$. The output point sequence generated with initial key $k_0 = 9$ is shown as follows:

$$(118, 123),\ (21, 43),\ (62, 47),\ (23, 18),\ (74, 108),\ (69, 66),\ (118, 123),\ (21, 43), \ldots .$$

As seen from the above, the period of the output point sequence is only 5. The output point sequence generated with $P = (62, 47)$ and $Q = (21, 43)$ with key = 9 is given as follows:

$$(79, 91),\ (75, 9),\ (124, 93),\ (93, 50),\ (59, 125),\ (93, 50),\ (59, 125), \ldots .$$

It can be clearly seen that, after three iterations, the output loops in two points on the EC.

4.6. PBSG-A. A modification of the dual EC generator with increased periodicity named PBSG-A is published in [15]. In PBSG-A the iteration key is modified as $k_{i+1} = k_iP + i * C$ where $C = X(eP)$ and "e" is the seed value. In addition to two point multiplication operations the modified algorithm requires a finite field multiplication of iteration number "i" and the value "C" to be carried out in each iteration. This increases both the hardware complexity and the time complexity of the system. Though PBSG-A is a stream cipher algorithm which generates key stream with high periodicity in such a way that security depends on ECDLP, the structural complexity and latency are very high.

5. Proposed Stream Cipher Design

The algorithm proposed in this paper addresses the drawbacks of the methods available in literature such as reduced periodicity, dependence on iteration number, security, and time complexity. The proposed method of stream cipher generation based on EC makes use of a single point P on the EC and a single point multiplication operation in each iteration. This reduces the time complexity to a large extent in comparison with two point multiplication operations carried out in each iteration in dual EC and PBSG-A. In the proposed method, a blinded version of the iterating key is used to generate the output point sequence so that even solving ECDLP does not reveal the exact iterating key to the attacker there by increasing the security. Also the proposed method is designed to have reduced hardware complexity without compromising the security.

Each iteration in the proposed method consists of two stages: (i) generation of key k_i for the ith iteration and (ii) generation of the ith output bit sequence. Various steps in algorithm can be detailed as follows. Let $e = e_1 \parallel e_2$ be the shared secret key. The value e_1 is the initial key k_0 for generating the iteration key and e_2 is the seed point of the LFSR. The LFSR is clocked once in each iteration. The key for the ith iteration is taken as the sum of x-coordinate of $k_{i-1}P$ and the content of LFSR C_{i-1}. That is, $k_i = X(k_{i-1}P) + C_{i-1}$. The addition of the LFSR value in generating the iteration key introduces randomness in the key steam, increases the periodicity, and increases the attack complexity. Moreover, this makes the iteration key less dependent on the EC points generated in each iteration and the iteration number. Replacing the GF multiplication $i*C$ in PBSG-A with an LFSR results in reduced time and hardware complexity.

The output point S_i for each iteration is computed as $S_i = k_iP + e_1P$ where the point multiplication "e_1P" is a precomputation stage. Providing this offset "e_1" helps in blinding the exact iteration key k_i from the attacker. An output point computation in the proposed method thus involves a point multiplication operation k_iP and a point addition, that is, $k_iP + e_1P$. The x-coordinate of the EC point k_iP is used to generate the key for $(i + 1)$th iteration. The truncated x-coordinate of the point S_i is given out as the bit sequence which further increases the security.

5.1. Algorithm. Let P be a point on the EC defined over GF(2^m). Let "e" be the shared secret. Length of "e" = $2m$ bits.

Input: point P, secret key "e".

Output: bit sequence s_i.

(1) Get e_1 and e_2 from e by truncating it to the required number of bits.
(2) Generate $Q = k_1P$.
(3) Initial key $k_0 = e_1$ and seed of LFSR $C_0 = e_2$.
(4) Generate the key for ith iteration $k_i = X(k_{i-1}P) + C_{i-1}$.
(5) Advance the LFSR count C_i.
(6) ith output point $S_i = k_iP + Q$.
(7) Output bit sequence $s_i = \text{trunc}(X(S_i))$.
(8) Return (s_i).
(9) Go back to step (4).

6. Period Analysis

In the proposed method the key advancement is done as $k_{i+1} = X(k_iP) + C_i = x_i + C_i$ where $x_i = X(k_iP)$. The use of LFSR value increases the period of the generator. If this value is not added, then the output sequence will depend only on the point multiplication operation. The sequence will start repeating whenever $x_n = x_{n-i}$ where $1 \leq i < n$. But if LFSR value is added to x_i the period will be governed by the period of the LFSR which is shown in the analysis below.

Assume that the output in the tth iteration is the same as the output in the rth iteration where $t > r$. That is, $S_r = S_t$. This implies $k_rP = k_tP$ or $x_r = x_t$. In the rth iteration, $k_{r+1} = x_r + C_r$ and in the tth iteration $k_{t+1} = x_t + C_t$. If the output of the $(t + 1)$th iteration is also equal to that of the $(r + 1)$th iteration, then $x_{t+1} = x_{r+1}$; that is, $k_{t+1}P = k_{r+1}P$ or $k_{t+1} = k_{r+1}$. This shows that $x_t + C_t = x_r + C_r$ or $C_t = C_r$ since $x_t = x_r$.

Since C_t and C_r are values of the LFSR, the same value will be repeated only after one period of the LFSR. For an LFSR of length "m" bits the period is $p = 2^m - 1$. Hence the output pattern will repeat only after integer multiples of the period "p". In general, the period of the sequence can be represented as "$c * p$" where $c \geq 1$. Thus the period of the point sequence in the proposed method is at least the period of LFSR. Consider the implementation of the proposed method done over GF(2^{163}). Let the 163-bit representation of the x-coordinate of the output point sequence be truncated to 100 bits. Then the period of the output bit sequence is at least $100 * p = 100 * (2^{163} - 1)$, that is, approximately 2^{169} which is a large value when compared with other existing schemes.

7. Security Analysis

This section analyses the security of the proposed stream cipher against various attacks. The analysis is carried out with the assumption that the EC is defined over GF(2^m) and P is a generator point on the EC. The EC chosen, the underlying field, and point P are known to the attacker.

7.1. Known Plain Text Attack. In the proposed method, the input secret is a random integer "e". A part of this secret key (e_1) is used as the initial key for point multiplication and the other part (e_2) is used as the seed value of the LFSR. The iteration key $k_{i+1} = X(k_iP) + C_i$ where C_i is the content of

LFSR after "i" clock cycles. In each iteration, the iteration key k_i is blinded by adding it with the secret key "e_1" and the output point S_i is computed as $S_i = k_iP + e_1P$. The x-coordinate of the point S_i is truncated to generate the output sequence. These output random bits are XORed with the message bits to generate the encrypted data. To break this cryptosystem the attacker needs to retrieve the iteration key and the internal state of the LFSR.

In a known plain text attack, we assume that attacker has knowledge about a part of the message stream. This reveals a part of the bit sequence generated by the algorithm. Thus an attacker possesses a truncated version of the x-coordinate of the points on the EC generated in a few iterations. For a successful attack, the attacker needs to identify the EC point from its truncated x-coordinate. The security for this stage is provided by the truncation point problem. The truncation point problem states that it is hard to identify whether a sequence is generated by truncating the x-coordinate of a point on the EC or if it is chosen uniformly at random.

Once the attacker has identified the point S_i from its truncated version, the next step in the attack is to solve the ECDLP to identify the integer $k_i + e_1$ for point multiplication. The most common attacks on ECDLP are the Pollard-rho attack and the baby step-giant step algorithm [21].

Pollard-Rho Attack. Let P be a point on the EC defined over GF(2^m). Let N be the order of point P. Then the complexity of Pollard-rho attack is given as $(\pi N/2)^{1/2}$. By Hasse's bound, if P is a generator point, then order of P is approximately the size of the field. Therefore, the complexity in solving ECDLP is approximately $O(2^{(m-1)/2})$.

Baby Step-Giant Step Algorithm. This is another common algorithm for attack on both DLP (discrete logarithm problem) and ECDLP. The complexity of this attack depends on the order of the point P. For a point P of order N, the attack requires computation of $N^{1/2}$ points on the EC and memory to store these $N^{1/2}$ points. Thus the time complexity of solving ECDLP in the proposed method is $O(2^{m/2})$ point multiplication operations if point P is a generator point.

The values retrieved by solving ECDLP are $k_i + e_1$, $k_{i+1} + e_1$, $k_{i+2} + e_1$, and so forth where e_1 is the secret key. Assume that no blinding operation is done in generating the output sequence. Then the values retrieved by solving ECDLP are the iteration keys k_i, k_{i+1}, and so forth. These iteration keys are related as $k_{i+1} = X(k_iP) + C_i$.

If two successive iteration keys are known to the attacker, that is, k_i and k_{i+1}, then the state of the LFSR can be easily found out as $C_i = k_{i+1} - X(k_iP)$.

Thus by solving ECDLP for just two points in the output sequence the attacker can generate the whole key stream.

In the proposed method, because of the offset, the attacker can retrieve only the blinded iteration keys $k_i + e_1$, $k_{i+1} + e_1$, and so forth where $k_{i+1} + e_1 = X(k_iP) + C_i + e_1$; that is, $C_i = k_{i+1} - X(k_iP)$.

But neither k_i nor k_{i+1} are known to the attacker. For a successful attack, the attacker has to solve for e_1. Let k_1 be the initial iteration key such that $k_1 = X(k_0P) + C_0$. Then $k_2 = X(k_1P) + C_1$, $k_3 = X(k_2P) + C_2$, and so forth. The output point sequence is generated as $S_1 = [X(k_0P) + C_0 + e_1]P$, $S_2 = [X(k_1P) + C_1 + e_1]P$, and so forth.

As can be seen from the above expressions, solving ECDLP for any number of output points yields little information about the offset e_1 and the iterating key k_i. Thus given $k_i + e_1$ the only attack possible to find the value k_i or e_1 is the brute force attack which has a complexity of $O(2^{m-1})$. The key for the next iteration is computed as $X(k_iP) + C_i$ where C_i is the content of the LFSR after "i" clock cycles. This method of generating the iterating keys introduces randomness into the key sequences and increases the complexity of attack without much increase in hardware or computational complexity. For an attacker who has arbitrarily chosen the key for ith iteration, generation of the key for the $(i + 1)$th iteration requires the knowledge of the content of the LFSR. For the randomly chosen value of k_i' or e_1', the attacker has to find the LFSR state C_i' such that $X(k_iP) + C_i' + e_1' = k_{i+1} + e_1$. The key stream is generated with these values of k_i', e_1', and C_i' and compared with the original key stream. This demands that the attacker has retrieved the output bit sequence of at least three consecutive iterations. If the generated key stream is different from the original then a new value for k_i or e_1 is chosen and the above process is repeated. Thus the complexity of a known plain text attack on the proposed system is $O(2^{m-1} + 2^{(m+1)/2}) \approx O(2^{m-1})$.

The various steps in the attack can be summarised as follows.

(1) Get a few bits of the key stream s from the known plain text.

(2) Get S_i from trunc$(X(S_i))$ (TPP).

(3) Solve for integer for ith iteration, that is, $(k_i + e_1)$ (ECDLP).

(4) Randomly choose k_i' and compute $e_1' = k_i + e_1 + k_i'$ (brute force).

(5) Find the LFSR state C_i' such that $X(k_i'P) + C_i' + e_1' = k_{i+1} + e_1$.

(6) Generate the new key stream s' and compare with s.

(7) If $s' \neq s$ go to step (4).

As the security of the proposed algorithm is vested in ECDLP, the elliptic curve must be chosen such that it can resist the MOV attack which uses Weil pairing to reduce the discrete logarithm problem on elliptic curves to the discrete logarithm problem (DLP) in finite field. This is due to the fact that various subexponential and quasipolynomial time algorithms for solving DLP are available in the literature. But studies reveal that MOV reduction is possible only for super singular curves and not for nonsuper singular curves. Therefore, a nonsuper singular curve needs to be chosen for secure implementations. The size of the finite field over which EC is defined can be determined based on the required security level as recommended by NIST. For example, NIST recommends the use of GF(2^{163}) for 80-bit security. The family of NIST standard curves guarantees this security and hence can be used for implementing the proposed algorithm with a specific security level.

7.2. Brute Force Attack. The complexity of brute force attack depends on the key space. In the proposed method the only secret is the value "e" which is a binary string of length "$2m$" bits for an EC defined over GF(2^m). In the proposed algorithm, m bits of the key is given as initial seed of the LFSR and the other m bits is initial key k_0 for point multiplication. These initial seeds can take any value other than an all zero pattern. Hence, the key space available for the proposed algorithm is approximately 2^{2m-1} and the complexity of the brute force attack is $O(2^{2m-2})$. Considering an implementation of the proposed algorithm over GF(2^{163}), the key space available is 2^{325} and the complexity of brute force attack is 2^{324}.

8. Statistical Analysis

This section deals with the statistical analysis of the proposed pseudorandom sequence generator based on NIST randomness test suite and TestU01. For analysis based on NIST test suite, the EC chosen is $y^2 + xy = x^3 + x^2 + 1$ defined over GF(2^7). $x^7 + x^3 + 1$ is the primitive polynomial used for the construction of the finite field. A point $(11, 19)$ on the EC means $(\alpha^{11}, \alpha^{19})$ where α is root of the polynomial $x^7 + x^3 + 1$. A sequence of 50 points generated by running the algorithm with $P = [3, 39]$ and two different key values is shown in Table 1. From the table it is clear that even for a single bit change in the key the sequences of points generated are entirely different. The output bit sequence is generated by truncating the x-coordinate of each point in the sequence to 3 bits. The output bit sequence generated for $P = [3, 39]$ and key = 1525 is shown as follows:

$$\begin{array}{c}
0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\
0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0 \\
1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1 \\
0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1
\end{array} \tag{4}$$

Considering the theoretical analysis in Section 6, the expected period of the sequence is 381.

The bit sequence generated using the proposed method has been tested for its randomness properties based on five statistical tests. For Monobit test and runs test, the threshold value is 0.01 according to the NIST statistical test suite; that is, if the sample sequence gives a value greater than 0.01, then the sequence is accepted as random [22]. Serial test and Poker test were also carried out. The test statistics for these two tests are chosen such that reference distribution is χ^2 distribution and if the test values of the sample sequence are less than the threshold, then the sequence is said to pass the test. For a significance value of 0.05 ($P(X > x) = 0.05$) the threshold values are 5.9915 and 24.9958, respectively [23]. The test result for various key values in Table 2 clearly demonstrates that the randomness property is satisfied. In addition, the autocorrelation function plotted in Figure 3 validates the randomness property and periodicity of the sequence.

TestU01 [24] is a software library implemented in ANSI C language which consists of utilities for statistical testing of uniform random number generators. It includes six predefined batteries as well as general implementation of classical statistical tests for pseudorandom sequences. Out of the six predefined batteries of TestU01, Crush, Big Crush, and Small Crush batteries are tests for sequences of real numbers and the batteries Rabbit, Alphabit, and Block Alphabit are for testing the binary sequences. Since the proposed pseudorandom sequence generator outputs a binary sequence, the sequence is subjected to Rabbit, Alphabit, and Block Alphabit test batteries. For analysis, EC is defined over GF(2^{17}) and the output is truncated to 4 bits. A sequence of 2^{20} bits is generated and tested and the sequence passed all the three tests. The test results are given in Table 3.

9. Statistical Complexity Analysis

In [25] the authors have shown that MPR statistical complexity can be used as a measure of randomness for pseudorandom sequence generators. Statistical complexity is defined as the product of disorder (entropy) of the system and the "distance" of the probability distribution from an equiprobable distribution in probability space. To analyse the MPR statistical complexity of a pseudorandom sequence, the normalised entropy and the complexity are plotted. The zero value of MPR statistical complexity indicates a truly random sequence and for pseudorandom sequences with good randomness, the complexity tends to zero and normalised entropy tends to 1. The expressions for computing normalised entropy (H_{norm}) and complexity (C^{MPR}) are given as follows:

$$\begin{gathered}
H_{\text{norm}} = -\sum_{i=1}^{N} \frac{p_i \log(p_i)}{\log(N)}, \\
Q = Q_0 \cos^{-1} \sum_{i=1}^{N} (p_i)^{1/2} \left(\frac{1}{N}\right)^{1/2},
\end{gathered} \tag{5}$$

where $Q_0 = 1/\cos^{-1}(1/N)^{1/2}$ and

$$C^{\text{MPR}} = H_{\text{norm}} * Q. \tag{6}$$

Here p_i represents the probability of symbol i and N is the number of symbols. The normalised entropy and the

Table 1: Output point sequence of the algorithm with $P = [3, 39]$.

Key = 1524		Key = 1525	
105 123	41 77	29 63	18 68
124 94	22 38	31 55	59 125
46 73	116 33	110 63	78 20
110 63	57 38	68 111	39 10
43 44	83 119	106 82	23 100
59 125	17 36	29 63	124 94
96 105	106 11	36 126	52 68
45 49	9 95	48 116	78 20
114 34	46 36	12 81	13 17
88 25	86 21	17 36	13 17
74 108	11 19	43 44	78 95
13 17	26 34	90 98	70 72
21 89	91 111	36 9	68 111
45 49	39 111	35 66	79 107
29 63	78 20	65 83	116 125
121 61	104 9	79 91	46 36
96 105	75 9	13 17	103 109
79 107	86 88	42 51	115 122
72 18	93 50	82 53	115 122
24 35	12 81	116 33	91 38
6 78	34 72	44 76	18 63
22 38	29 40	75 9	43 74
55 76	53 41	68 111	85 37
34 72	12 81	90 84	24 35
44 76	36 9	13 10	45 49

Table 2: Statistical test results with $P = [3, 39]$.

Key	Monobit test	Runs test	Serial test	Poker test
497	0.4382	0.7488	0.6673	13.2453
948	0.5610	0.8334	0.4784	9.9245
1440	0.8463	0.5622	0.4777	14.4528
1525	0.9228	0.3328	1.0706	14.1509

Table 3: Test results using TestU01.

Battery	Number of statistics	Result
Rabbit	38	Pass
Alphabit	17	Pass
Block Alphabit		Pass

MPR statistical complexity of the proposed pseudorandom sequence generator are given in Figures 5 and 6, respectively. For analysis the output bit stream is grouped into 8-bit words. The analysis shows that the proposed pseudorandom sequence generator exhibits good randomness.

10. Linear Complexity Analysis

Linear complexity and linear complexity profile of a pseudo-random sequence are two important characteristic parameters used to measure the security of the sequence when it is used as a key stream. The linear complexity $L(S)$ of an ultimately periodic binary sequence S is the length of the shortest LFSR that can generate S with the convention that $L(S) = 0$ if S is the zero sequence. Let $S = (s_1, s_2, \ldots, s_n)$ be a finite sequence over GF(2). Denote the linear complexity of the first i terms $(s_1, s_2, \ldots, s_i)$ by $L(s^i)$. Then the linear complexity profile of S is defined to be the sequence $(L(s^1), L(s^2), \ldots, L(s^n))$. For an LFSR of length "L", though the periodicity of the sequence generated is $2^L - 1$, the linear complexity is only "L". For a nonlinear sequence, the maximum possible linear complexity is the same as the period of the sequence.

One of the efficient methods to compute the linear complexity profile is the Berlekamp-Massey algorithm. The linear complexity profile of the proposed pseudorandom number generator computed using the Berlekamp-Massey LFSR synthesis algorithm is shown in Figure 4. The EC is defined over GF(2^7) and the output is truncated to 3 bits. The period of the generated sequence is 381. From the plot it can be seen that the linear complexity profile is close to the $n/2$ line for the first period which is a property satisfied by unpredictable sequences. As seen in the linear complexity profile, the proposed random number generator has a linear complexity which is the same as the period of the sequence. Thus the proposed method exhibits very high linear complexity compared to LFSR based methods.

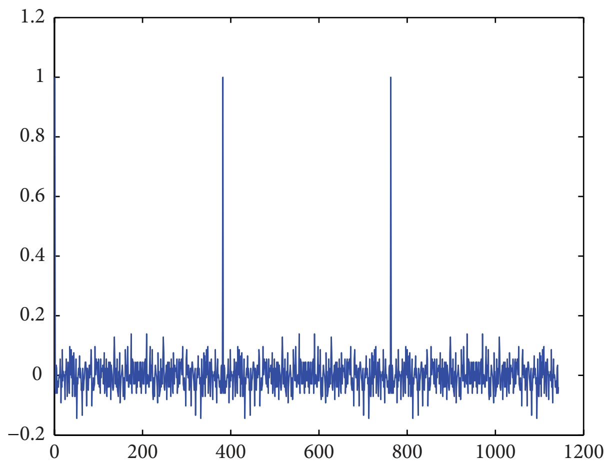

Figure 3: Autocorrelation function.

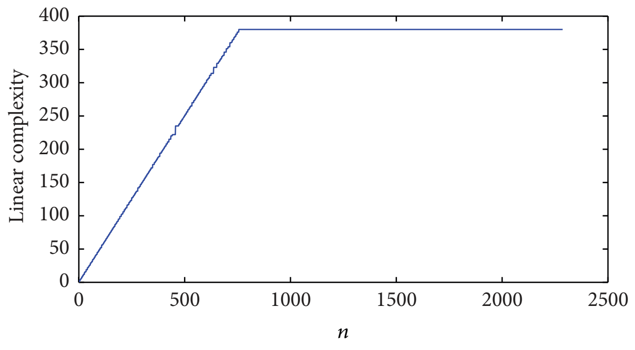

Figure 4: Linear complexity profile.

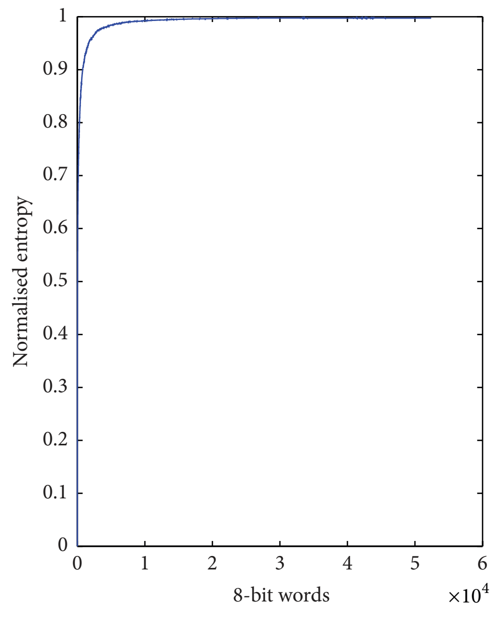

Figure 5: Normalised entropy function.

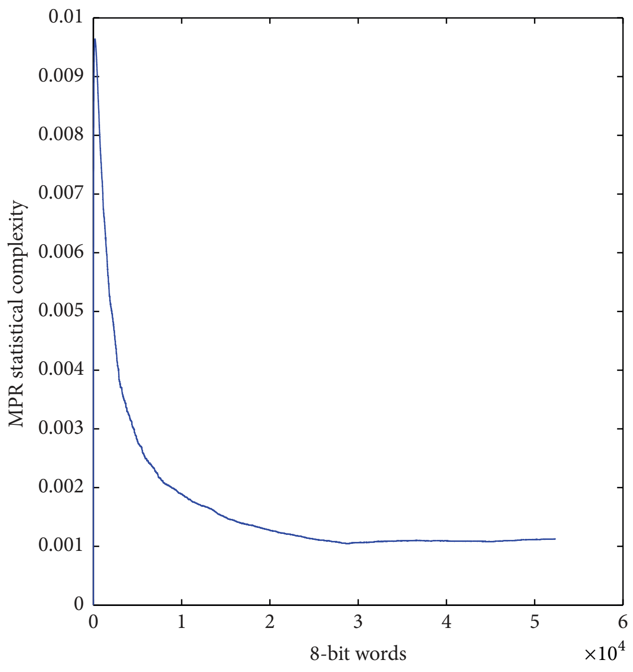

Figure 6: MPR statistical complexity.

11. Comparison with Other EC Based Stream Ciphers

The throughput (number of output bits per clock cycle), security, and hardware requirement of the proposed method are compared with EC based pseudorandom sequence generators available in literature.

11.1. Throughput. In the proposed method, truncation function is applied to the x-coordinate of the EC point to generate the output bit sequence. Consider an EC defined over GF(2^{163}). Let the x-coordinate be truncated to "100" bits by the truncation function. Let "n" be the number of clock cycles required for a point multiplication operation. Since each iteration involves only a single point multiplication operation, the throughput of the proposed system is approximately "$100/n$" bits per clock cycle.

The output bit sequence in linear congruential generator [10] and its variant PBSG-B [15] are generated by applying trace function to the x- and y-coordinates of the output point sequence giving out two output bits in each iteration. This reduces the throughput of the system. In these two methods, generation of a single bit in the key stream requires "$n/2$" clock cycles. This reduces the speed of operation of the encryption system and makes it not suitable for real time operations. Compared to "$2/n$" for a linear congruential generator and its derivatives, the proposed method has a throughput of "$100/n$" resulting in reduced latency and making it suitable for real time applications.

In dual EC generator [14] and various proposals based on this, each iteration consists of two point multiplication operations, one for generating the iteration key and the other for generating the output bit sequence. This highly increases the time complexity of the pseudorandom sequence generator. Assuming that "n" clock cycles are required for a single point multiplication operation and the output is truncated to "100" bits as considered above, the system generates "$100/2n$" bits per clock cycle. The throughput of PBSG-A [15], which is a variant of the dual EC generator with increased periodicity and security, is similar to the dual EC because of the two point multiplications in each iteration. As each iteration in the proposed method consists of a single point multiplication operation, the time complexity is reduced to a large extent. In the proposed method the number of output bits per clock cycle is "$100/n$." This shows that the throughput of the proposed system is increased by a factor of "two" when compared to dual EC generator and PBSG-A.

The two methods available in literature with a throughput similar to the proposed method are ECPG and the pseudorandom sequence generator in [13]. Both methods make use of a single point multiplication in each iteration and the output bit sequence is generated by applying truncation function to the output point.

11.2. Periodicity. This section analyses the period of various EC based pseudorandom sequence generators and compares it with the proposed method. Assume that the EC is defined over GF(2^m). Let l be the order of point P used for generating the sequence. The output is truncated to "t" bits using the truncation function. From the period analysis given in Section 6 it is clear that the periodicity of the proposed pseudorandom sequence is independent of the order of point P and is determined by the length of the LFSR or the size of the field over which the EC is defined. To analyse the dependence of the generated bit sequence on the initial seed, an image is encrypted using this sequence. The EC is defined over GF(2^{11}) and the output is truncated to 5 bits. The initial key seeds chosen are $k_0 = 104$ and $C_0 = 43$. The encrypted

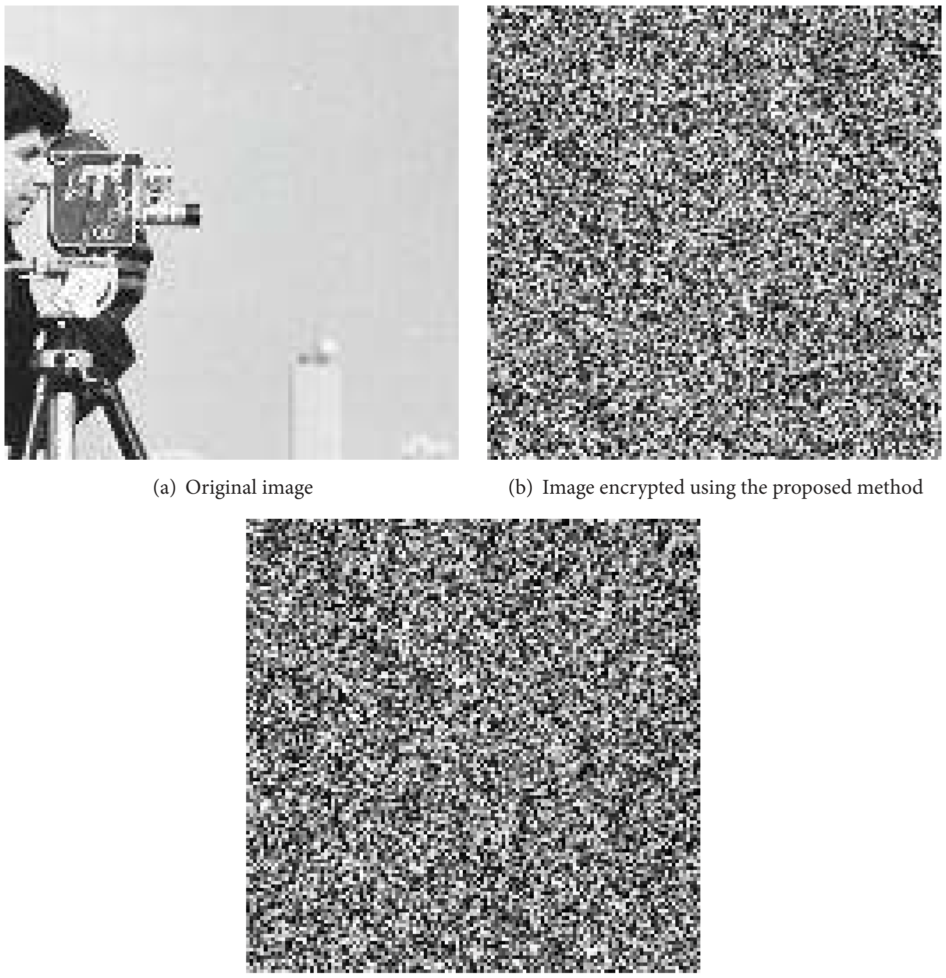
(a) Original image
(b) Image encrypted using the proposed method
(c) Image decrypted with wrong key

Figure 7: Original image, encrypted image, and image decrypted with the wrong key (based on the proposed method).

image is now decrypted with a wrong key $k_0 = 84$. The original image, encrypted image, and image decrypted with the wrong key are shown in Figures 7(a), 7(b), and 7(c), respectively. This shows that the sequence generated is highly dependent on the initial seed.

For LCG and PBSG-A, the order of point P determines the period of the sequence. Hence to achieve high periodicity, the order l should be very large. However, determination of a point on the EC with high order for a large field size is computationally intensive and hence these approaches are not recommended. In case of ECPG, the periodicity is determined by the order of point P and the value of the initial seed e. The initial seed e must be chosen such that $\text{ord}_l(e)$ is a large value. This limits the number of possible choices for e and reduces the complexity of attack on the system. The period of a dual EC generator cannot be determined as it is dependent on points P and Q used for generating the sequence. For certain values of P and Q it is observed that the period is as small as 2. In PBSG-B, which is a variant of LCG, the period is determined by the length of the LFSR. In [13], the authors have shown that the period of the sequence is determined by the size of the field over which the EC is defined. But it is observed that the sequence becomes independent of the key seed after a few iterations. The output point sequence generated based on the algorithm in [13] for different values of key is given in Table 4. From the table it can be seen that after 20 iterations the system outputs the same sequence independent of the initial seed.

11.3. Security. In the proposed method, the security analysis shows that the security is as high as solving ECDLP many times. The iteration key k_i is blinded by adding an offset value e_1 so that only the blinded value $k_i + e_1$ can be retrieved by the attacker after solving ECDLP. From the security analysis of the proposed method it is clear that solving ECDLP for any number of points does not provide information about the iteration key or the secret value e_1. Thus even after solving ECDLP, which has a complexity of $O(2^{(m-1)/2})$ for an EC defined over GF(2^m), the attacker has to go for a brute force attack to break the cryptosystem. Thus, the computational complexity of an attack on the proposed system is $O(2^{m-1})$. From the analysis in Section 7.1, it can be seen that, to

Table 4: Output point sequence of generator in [13].

Key = 10	Key = 12	Key = 13
109 119	74 85	9 34
121 61	57 17	58 80
11 19	23 100	18 68
86 21	34 119	55 95
31 87	55 95	116 33
12 29	115 122	86 88
37 54	62 47	82 27
114 76	72 125	105 66
9 95	72 18	91 111
90 84	110 25	92 72
81 18	46 36	36 126
121 59	6 78	75 9
115 122	9 34	83 5
9 95	116 125	98 100
121 59	124 93	34 119
90 98	53 41	90 84
−Inf 0	68 17	−Inf 0
34 72	93 126	34 72
115 118	101 68	115 118
83 5	41 77	83 5
43 44	43 74	43 44
49 80	49 80	49 80
43 74	43 74	43 74
42 86	42 86	42 86
74 108	74 108	74 108
75 50	75 50	75 50
83 119	83 119	83 119
88 40	88 40	88 40
24 35	24 35	24 35
69 66	69 66	69 66
23 100	23 100	23 100

mount an attack on the proposed system, the attacker should have the output bit sequence corresponding to at least three consecutive iterations.

The linear congruential generator described in [10] and PBSG-B in [15] are proved to have good randomness properties but the security is dependent on the secrecy of point P and not on ECDLP. Thus the computational complexity of an attack on the system cannot be quantified. Moreover, in LCG the period of the sequence reveals the order of point P and the symmetric properties of the sequence make the cryptanalysis easier [15].

The random number generator in [13] has a throughput and periodicity similar to the proposed method. But the analysis in Sections 4.4 and 11.2 shows that the sequence becomes independent of the initial key seed after a few iterations. Thus the system is insecure and no security analysis is to be done.

The output point sequence and iteration key in ECPG [12] are computed as $Q_i = k_i P$ and $k_i = e^i$ where e is the initial secret seed. A successful cryptanalysis can be mounted on ECPG by preparing a lookup table with i and $\alpha^i P$ as entries, as a precomputation step. If two consecutive output points are known, then the initial secret key e can be easily found out using this lookup table as explained in Section 4.3. Thus, security of ECPG is only due to the truncation function and is not dependent on ECDLP.

In dual EC generator, the iteration key and output point sequence are generated as $k_i = X(k_{i-1}P)$ and $R_i = k_i Q$. If we assume that an attacker has retrieved the ith iteration key k_i by solving ECDLP, the iteration key k_{i+1} can be easily found out as $k_{i+1} = X(k_i P)$. Thus for a successful attack on the above system, solving ECDLP for a single point in the output sequence is sufficient. Hence, the attack complexity is $O(2^{(m-1)/2})$.

PBSG-A [15] is a modification of the dual EC generator with increased periodicity. The output point sequence and iteration key are generated as $R_i = k_i Q$ and $k_i = X(k_{i-1}P) + i * C$ where $C = X(e_0 P)$ and e_0 is the secret key. Assume that the attacker solved ECDLP and computed the iteration keys k_i, k_{i+1}, and k_{i+2}. These iteration keys are related as $k_{i+1} = X(k_i P) + i * C$ and $k_{i+2} = X(k_{i+1}P) + (i+1) * C$. Using the above two equations, the attacker can compute $i * C = k_{i+1} - X(k_i P)$ and $(i+1) * C = k_{i+2} - X(k_{i+1}P)$.

The difference of the above two expressions gives the value of the secret constant "C". Then the successive iteration keys can be easily found out as $k_{i+3} = X(k_{i+2}P) + (i+2)C$, and so forth.

Thus solving ECDLP for 3 consecutive output points results in a successful attack on the PBSG-A algorithm. The computational complexity can be expressed as $O(3 * 2^{(m-1)/2}) \approx O(2^{(m+3)/2})$.

A summary of these comparisons is given in Table 5. Here it is assumed that the EC is defined over GF(2^m) and the truncation function truncates "m" bits of the x-coordinate of the output point sequence to "t" bits of output bit sequence, for each point multiplication operation. Let l be the order of point P and let "n" be the number of clock cycles required for point multiplication. The comparison results in Table 5 clearly indicate that the proposed method has increased security, periodicity, and throughput compared to other EC based systems available in literature.

11.4. Structural Complexity. In this section, the approximate hardware resources required for the implementation of the proposed method are analysed.

The hardware requirement of various pseudorandom sequence generators is basically dependent on the method of generating the iteration key. In the proposed method, the generation of the iteration key involves the GF addition of contents of LFSR and the output of point multiplication unit. A basic structure of the proposed stream cipher is given in Figure 8. The hardware requirement in addition to the point multiplication unit is a register to store the initial seed, LFSR, a finite field addition unit, buffer to store the result of GF addition, multiplexer, and truncation unit. By properly

Table 5: Comparison with other EC based stream ciphers.

	Security	Throughput	Approx. number of output bits required to attack the system	Period
Linear congruential generator [10]	Secrecy of point P	$2/n$	—	$2l$
PBSG-B [15]	Secrecy of point P	$2/n$	—	2^{m+1}
Dual EC generator [14]	$O(2^{(m-1)/2})$	$t/2n$	t	—
PBSG-A [15]	$O(2^{(m+3)/2})$	$t/2n$	$3t$	$l * t$
Proposed method	$O(2^{m-1})$	t/n	$3t$	$(2^m - 1) * t$

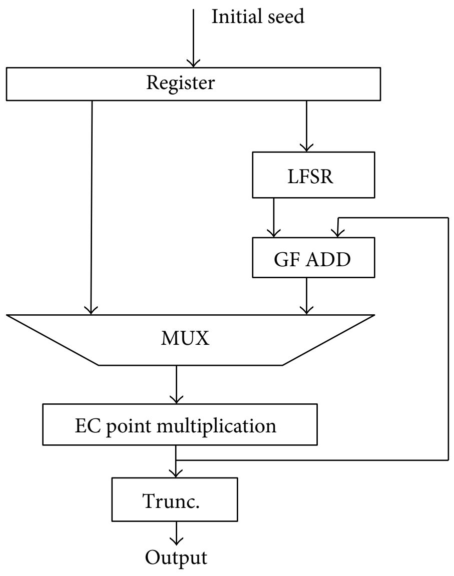

Figure 8: Basic structure of proposed stream cipher.

Table 6: Approximate gate equivalent of various hardware units.

Hardware unit	Approximate gate equivalent	
	Proposed method	PBSG-A [15]
Register (163 bits)	3 × 1,304	4 × 1,304
LFSR (163 bits)	1,304	0
GF add	408	408
MUX	815	3 × 815
EC point multiplication [18]	1,38,000	1,38,000
Total	1,44,439	1,46,069

choosing the basis for representing the field elements, the GF addition becomes a simple bitwise XOR operation so that the hardware complexity is reduced.

From Table 5, the only method with a security and periodicity comparable to the proposed method is PBSG-A. In PBSG-A the iteration key $k_{i+1} = X(k_i)P + i * C$ where "C" is a constant. The hardware for computing $i * C$ can be implemented using a GF addition circuit and two buffers for storing $i * C$ and C values so that $(i+1) * C = i * C + C$. Thus computing the iteration key involves two finite field addition operations. These two operations can be done using the same GF addition unit by giving the input values through a mux. The various hardware units required for implementing the PBSG-A algorithm are (i) EC point multiplication unit, (ii) multiplexers to input values to the GF addition unit and to input points P and Q alternately to the point multiplication unit, and (iii) buffers to store seed value, C, $i*C$, and k_i values.

Various optimised implementation methods of elliptic curve point multiplication unit in terms of area utilisation and speed are available in literature. Approximate gate equivalent of various hardware units in the proposed method and PBSG-A is shown in Table 6 [26]. This is done under the assumption that the implementation is done over GF(2^{163}). From the table it is clear that hardware implementation of the proposed method requires 1,630 gates less compared to PBSG-A. Moreover there is only a marginal increase in the hardware requirement for implementing the proposed pseudorandom number generator compared to the point multiplication unit. If the point multiplication unit is time shared for both key exchange and stream cipher generation, then the overall hardware complexity of the system can be reduced and makes it suitable for applications with limited resources.

12. Application of the Proposed Algorithm for Image Encryption

Image encryption is a potential application where stream cipher is highly preferred over block cipher due to the bulky nature of the data and high correlation between the adjacent pixels. The pseudorandom sequence used for image encryption must have good randomness properties and high periodicity so that the encrypted image is secure. In addition to the standard tests which check the randomness of pseudorandom sequences, the robustness and security of the sequence generator as a stream cipher for encrypting an image can be analysed by performing the security analysis of the ciphered image through evaluation of various parameters [27]. In this section, the pseudorandom sequence generated by the proposed algorithm is used for encrypting a 256 × 256 gray scale image and the security analysis of the ciphered image is carried out. The time required for encrypting a 256 × 256 image is evaluated on Intel Xeon 3.7 GHz CPU with 1 GB RAM as approximately 0.006 s.

The plain image is shown in Figure 9 and the encrypted image is shown in Figure 10. The various parameters analysed are distribution of the cipher text, correlation of two adjacent pixels, information entropy, avalanche criterion, and resistance to differential attack.

Figure 9: Original image.

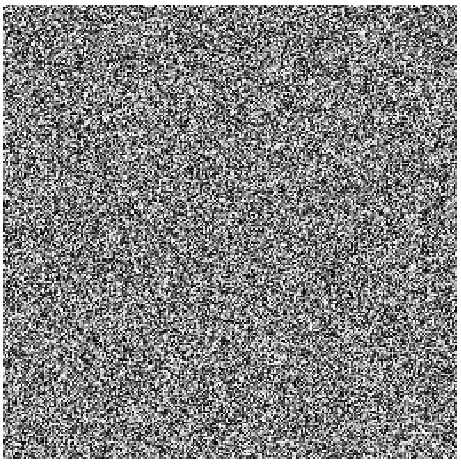

Figure 10: Encrypted image.

12.1. Distribution of the Cipher Text. The histogram of the plain image and the encrypted image are given in Figures 11 and 12 respectively. From Figure 12 it can be clearly seen that the pixels in the encrypted image are uniformly distributed.

12.2. Correlation of Adjacent Pixels. The adjacent pixels of the plain image are highly correlated as shown in Figure 13 and are prone to statistical attacks. To resist these attacks, the adjacent pixels of an encrypted image must be highly uncorrelated. The correlation of vertically adjacent pixels in the encrypted image is shown in Figure 14. It can be clearly seen that the pixels are uncorrelated and can resist the attacks. The vertical, horizontal, and diagonal correlations of adjacent pixels are computed using the following expressions and the results are summarised in Table 7:

$$r_{xy} = \frac{\operatorname{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}},$$
$$\operatorname{cov}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)). \quad (7)$$

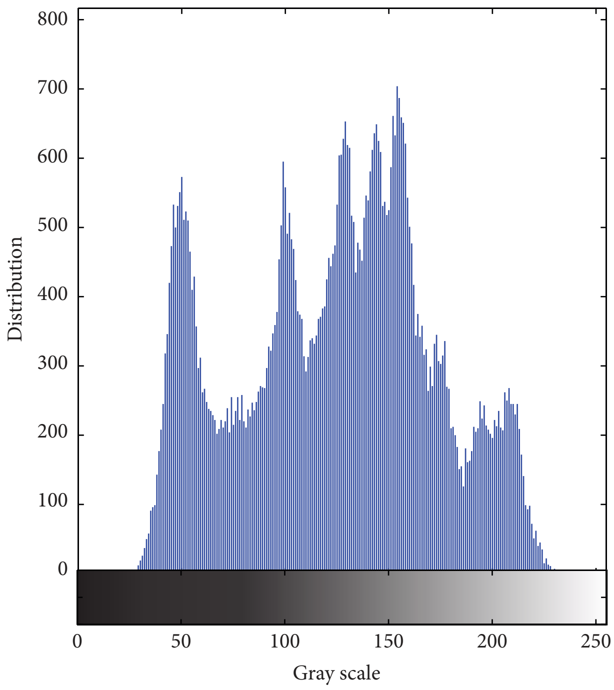

Figure 11: Histogram of original image.

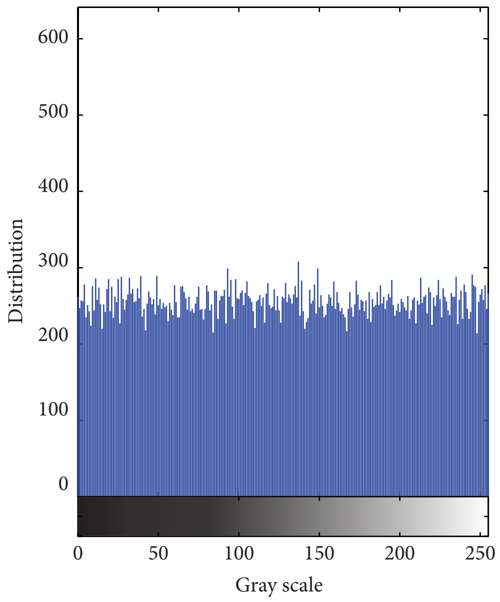

Figure 12: Histogram of encrypted image.

Here, $D(x) = 1/N\sum_{i=1}^{N}(x_i - E(x))^2$ is the variance of x and $E(x) = 1/N\sum_{i=1}^{N} x_i$ is the expectation of x.

12.3. Information Entropy. Information entropy of a source S is defined as

$$H(S) = -\sum_{i=1}^{N} p_i \log(p_i), \quad (8)$$

Table 7: Correlation coefficient of two adjacent pixels.

Direction	Original image	Encrypted image
Horizontal	0.9400	0.0025
Vertical	0.9709	0.0037
Diagonal	0.9141	0.0011

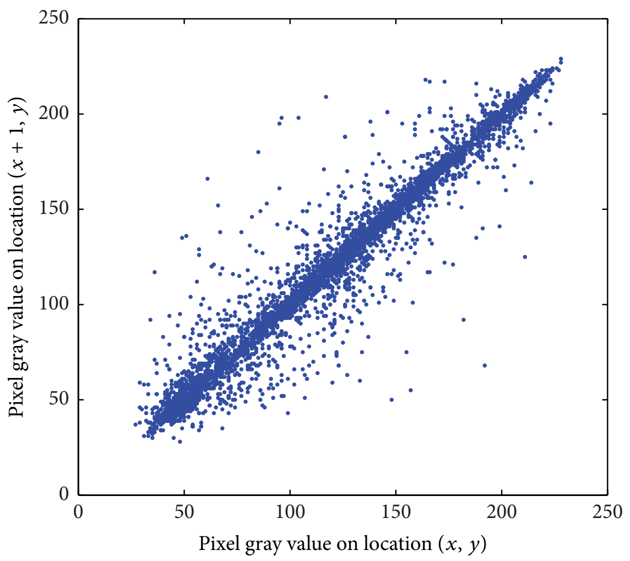

Figure 13: Correlation analysis of plain image.

where p_i is the probability of symbol i. For a gray scale image, the number of possible symbols is 2^8 and hence the maximum entropy we can get is 8. The entropy of an encrypted image must be close to 8 to ensure that the information leakage is zero. The entropy of the encrypted image in Figure 10 is computed and the value obtained is 7.9968 which ensures that the encryption algorithm is secure.

12.4. Sensitivity Analysis. In this section, the key sensitivity and plain text sensitivity of the proposed method are analysed. In key sensitivity, the change in the encrypted image for a change in single bit of the key is analysed and in plain text sensitivity the change in cipher image for a change in single pixel of the plain image is analysed. The two common measurements used to analyse the sensitivity are NPCR (number of pixels change rate) and UACI (unified average changing intensity). The NPCR and UACI values are computed using the following expressions:

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\%,$$
$$\text{UACI} = \frac{1}{W \times H}\left[\sum_{i,j}\frac{|C_1(i,j) - C_2(i,j)|}{255}\right] \times 100\%. \quad (9)$$

Here C_1 and C_2 are the two cipher images, $D(i,j) = 1$, if $C_1(i,j) \neq C_2(i,j)$, and $D(i,j) = 0$, if $C_1(i,j) = C_2(i,j)$; W and H are the width and length of the image.

Table 8: Sensitivity analysis.

	Key	Plain text
NPCR	99.63	99.59
UACI	33.56	33.42

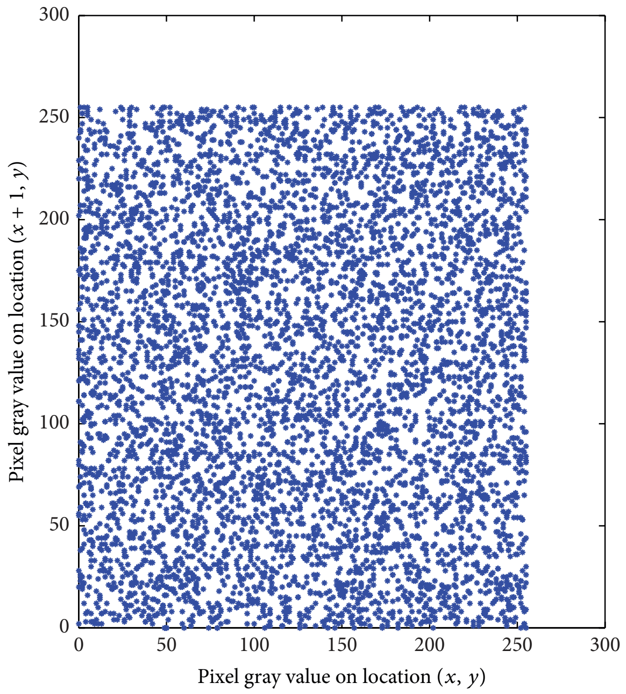

Figure 14: Correlation analysis of encrypted image.

For key sensitivity analysis C_1 and C_2 are obtained by encrypting the plain image with two different keys k_1 and k_2 such that k_1 and k_2 differ only in a single bit. In the proposed algorithm, the pseudorandom sequence generated is independent of the plain text. To analyse the plain text sensitivity and the resistance of the algorithm against the differential attack, the input key of the pseudorandom sequence generator is made dependent on the plain image. This is done by generating the key to the sequence generator as the residue obtained by passing the plain text through a modular division circuit. The NPCR and UACI values are computed for both cases and are summarized in Table 8.

12.5. Avalanche Criterion. The avalanche criterion is used to prove the sensitivity of the algorithm to plain text. Two images with one pixel difference and their corresponding cipher images are generated. The effects of one bit change in the plain image and cipher image are shown in Figures 15 and 16, respectively. From Figure 16 it can be clearly seen that one pixel change in the plain image produces considerable change in the encrypted images.

13. Conclusion

Resource constrained applications like WSNs demand new algorithms for encryption which can offer reduced time

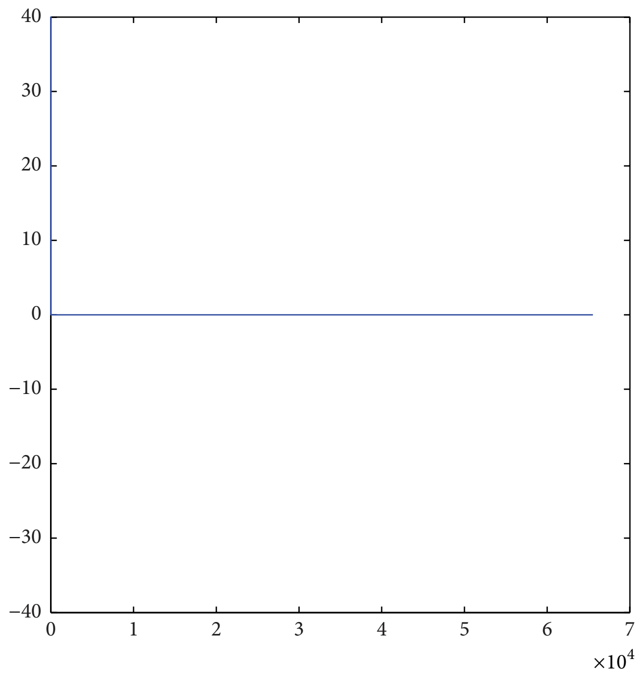

Figure 15: Difference between plain images.

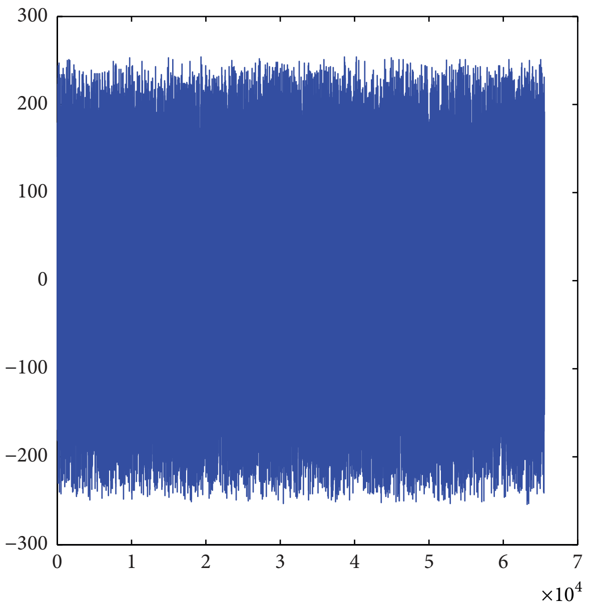

Figure 16: Difference between encrypted images.

complexity, structural complexity, increased security, and throughput. As ECC is a promising solution for key exchange with increased security, the point multiplication unit will be already available in the system as a part of key exchange. This paper describes a hardware efficient EC based random bit stream generator in which the point multiplication unit used for key exchange can be time shared to generate the pseudorandom sequence so that the overall hardware complexity is reduced. Five basic tests are done to check the randomness of the bit sequence generated and the sequence is found to have good statistical properties. The sequence exhibits very high periodicity and throughput in comparison with the EC based pseudorandom sequence generators available in literature. Similarly, compared to other EC based approaches, the computational complexity of known plain text attack on the system increases exponentially with the size of the key resulting in high security.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is partially funded by the E-Security Division of the Department of Electronics and Information Technology under Ministry of Communication and Information Technology of Government of India, as per order no. 12(16)/2012-ESD dated 01-08-2013.

References

[1] M. Erol-Kantarci and H. T. Mouftah, “Wireless sensor networks for cost-efficient residential energy management in the smart grid,” *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 314–325, 2011.

[2] S. Jimenez-Fernandez, P. De Toledo, and F. Del Pozo, “Usability and interoperability in wireless sensor networks for patient telemonitoring in chronic disease management,” *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 12, pp. 3331–3339, 2013.

[3] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS ’02)*, pp. 41–47, ACM, Washington, DC, USA, November 2002.

[4] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks,” in *Proceedings of the IEEE Symposium on Security and Privacy (SP ’03)*, pp. 197–213, ACM, May 2003.

[5] D. Liu, P. Ning, and R. Li, “Establishing pairwise keys in distributed sensor networks,” *ACM Transactions on Information and System Security*, vol. 8, no. 1, pp. 41–77, 2005.

[6] J. Nam, M. Kim, J. Paik, Y. Lee, and D. Won, “A provably-secure ECC-based authentication scheme for wireless sensor networks,” *Sensors*, vol. 14, no. 11, pp. 21023–21044, 2014.

[7] Y. Choi, D. Lee, J. Kim, J. Jung, J. Nam, and D. Won, “Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography,” *Sensors*, vol. 14, no. 6, pp. 10081–10106, 2014.

[8] P. Kotzanikolaou, E. Magkos, D. Vergados, and M. Stefanidakis, “Secure and practical key establishment for distributed sensor networks,” *Security and Communication Networks*, vol. 2, no. 6, pp. 595–610, 2009.

[9] M. Blum and S. Micali, “How to generate cryptographically strong sequences of pseudorandom bits,” *SIAM Journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984.

[10] G. Gong, T. A. Berson, and D. R. Stinson, “Elliptic curve pseudorandom sequence generators,” in *Selected Areas in Cryptography*, pp. 34–48, Springer, 2000.

[11] I. E. Shparlinski, "On the naor–reingold pseudo-random function from elliptic curves," *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, no. 1, pp. 27–34, 2000.

[12] T. Lange and I. E. Shparlinski, "Certain exponential sums and random walks on elliptic curves," *Canadian Journal of Mathematics*, vol. 57, no. 2, pp. 338–350, 2005.

[13] L.-P. Lee and K.-W. Wong, "A random number generator based on elliptic curve operations," *Computers & Mathematics with Applications*, vol. 47, no. 2-3, pp. 217–226, 2004.

[14] E. B. Barker and J. M. Kelsey, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*, US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory, 2007.

[15] P. P. Deepthi and P. S. Sathidevi, "New stream ciphers based on elliptic curve point multiplication," *Computer Communications*, vol. 32, no. 1, pp. 25–33, 2009.

[16] K. Suwais and A. Samsudin, "ECSC-128: new stream cipher based on elliptic Curve discrete logarithm problem," in *Proceedings of the 1st International Conference on Security of Information and Networks (SIN '07)*, May 2007.

[17] S. M. C. Vigila and K. Muneeswaran, "Key generation based on elliptic curve over finite prime field," *International Journal of Electronic Security and Digital Forensics*, vol. 4, no. 1, pp. 65–81, 2012.

[18] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 217–225, 2013.

[19] N. Koblitz, *A Course in Number Theory and Cryptography*, vol. 114, Springer, 1994.

[20] A. A. Ciss and D. Sow, "On randomness extraction in elliptic curves," in *Progress in Cryptology—AFRICACRYPT 2011*, pp. 290–297, Springer, 2011.

[21] D. Hankerson, S. Vanstone, and A. J. Menezes, *Guide to Elliptic Curve Cryptography*, Springer, New York, NY, USA, 2004.

[22] L. E. Bassham III, A. L. Rukhin, J. Soto et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Tech. Rep. SP 800-22, National Institute of Standards & Technology, Gaithersburg, Md, USA, 2010.

[23] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[24] P. L'Ecuyer and R. Simard, "Testu01: a C library for empirical testing of random number generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, article 22, 2007.

[25] C. M. González, H. A. Larrondo, and O. A. Rosso, "Statistical complexity measure of pseudorandom bit generators," *Physica A: Statistical Mechanics and Its Applications*, vol. 354, no. 1–4, pp. 281–300, 2005.

[26] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: grain-128," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '06)*, pp. 1614–1618, July 2006.

[27] A. Akhshani, A. Akhavan, S.-C. Lim, and Z. Hassan, "An image encryption scheme based on quantum logistic map," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4653–4661, 2012.