scientific reports

Check for updates

OPEN

A generalized framework for elliptic curves based PRNG and its utilization in image encryption

Sherif H. AbdElHaleem[1]✉, Salwa K. Abd-El-Hafiz[1]✉ & Ahmed G. Radwan[1,2]

In the last decade, Elliptic Curves (ECs) have shown their efficacy as a safe fundamental component in encryption systems, mainly when used in Pseudorandom Number Generator (PRNG) design. This paper proposes a framework for designing EC-based PRNG and maps recent PRNG design techniques into the framework, classifying them as iterative and non-iterative. Furthermore, a PRNG is designed based on the framework and verified using the National Institute of Standards and Technology (NIST) statistical test suite. The PRNG is then utilized in an image encryption system where statistical measures, differential attack measures, the NIST statistical test suite, and system key sensitivity analysis are used to demonstrate the system's security. The results are good and promising as compared with other related work.

Rapid developments in the digital world highlighted the need for securing digital content, especially images as they are used and shared extensively. Therefore, securing images has gained much attention from researchers in the last decade. However, methods of securing images vary a lot depending on the application. For example, image encryption obscures the image while image watermarking transparently embeds ownership. A source of randomness exists in the heart of any encryption system; this source provides the system with its strength and can vary from one system to another. Chaos-based, non-chaos-based, and elliptic curves are sources that proved their efficiency.

Chaos-based techniques gained much attention because of their sensitivity to system parameters and initial conditions. While some techniques added extra parameters to chaotic systems to increase their sensitivity and system key length[1–3], others generated dynamic S-box using either Henon map[4] or logistic-sine map[5], or generated random keystream using quantum logistic map[6]. On the other hand, non-chaos-based systems gained attention from the diversity of components that can be combined to achieve comparable security strength. For example, such systems can utilize the complex details of fractals in the PRNG process[7,8], use Linear Feedback Shift Register (LFSR) in image encryption[9], perform permutation and substitution using Feistel networks[10], or apply a DNA encoding process of image pixels[11]. Moreover, two assessment measures were developed for the performance of various chaotic and non-chaotic based permutation techniques[12]. A summary of several encryption system configurations, based on chaotic and non-chaotic generators, was proposed by Ref.[13] demonstrating the effect of each configuration on system security.

ECs are utilized because of the difficulty of the Discrete Logarithm Problem (DLP) and the ability to achieve high-security strength using a smaller key length than other public-key techniques. For instance, designing an authenticated encryption scheme for message mapping on EC[14,15], generating discrete chaotic sequences using the EC-based linear congruential method[16], using isomorphic elliptic curves in generating S-boxes[17], improving the ElGamal encryption technique by solving data expansion issue[18,19], or utilizing the Diffie–Hellman key exchange protocol and EC point addition in image encryption[20] are among the techniques that utilize ECs.

The main contributions of this paper are summarized as follows. First, a novel generalized framework for EC-based iterative and non-iterative PRNG is proposed and verified using recent literature. With the help of this framework, a simple PRNG based on ECs is designed using one EC point addition operation and two truncations. In addition, an image encryption system combining chaos and number theory is designed by utilizing the proposed PRNG. Finally, the PRNG and encryption system are evaluated using well-known standard criteria and they demonstrated good results.

The paper is organized as follows. After briefly describing the mathematical basics of ECs, a novel framework for EC-based PRNG is presented, and a PRNG is proposed based on it. An image encryption system is, then,

[1]Engineering Mathematics Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt. [2]School of Engineering and Applied Sciences, Nile University, Giza 12588, Egypt. ✉email: sherif.muhamed.a@eng-st.cu.edu.eg; salwaka@eng1.cu.edu.eg

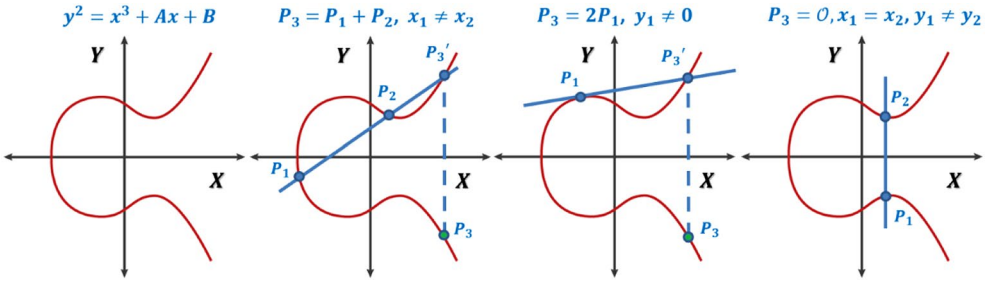

Figure 1. Example EC and the first three cases for point addition.

Figure 2. A generalized framework for PRNGs.

designed by utilizing the proposed PRNG. Furthermore, the different evaluation criteria are explained and used in assessing the PRNG and the encryption system. Finally, a comparison with related literature is given, followed by the conclusions.

Elliptic curves basics

A Weierstrass equation takes the form $y^2 = x^3 + \mathrm{A}x + \mathrm{B}$, where A and B are constants. An EC is defined over a field F when $\mathrm{A}, \mathrm{B} \in F$. For the cubic equation not to have multiple roots, a restriction is added over the values of A and B, which is $4A^3 + 27B^2 \neq 0$[21].

For cryptography applications, x, y, A, and B are taken to be elements from the finite fields F_p, where p is a large prime. Adding the point at infinity $\mathcal{O}$ to the set of all points satisfying the EC equation creates an additive abelian group with $\mathcal{O}$ being the identity element. Group operations are point addition and multiplication. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on an EC, E, then $P_3 = P_1 + P_2 = (x_3, y_3)$ is calculated using

$$P_3 = \begin{cases} \left(m^2 - x_1 - x_2, m(x_1 - x_3) - y_1\right), & m = \frac{y_2 - y_1}{x_2 - x_1}, \quad x_1 \neq x_2 \\ \left(m^2 - 2x_1, m(x_1 - x_3) - y_1\right), & m = \frac{3{x_1}^2 + A}{2y_1}, \quad P_1 = P_2, y_1 \neq 0 \\ \mathcal{O}, & x_1 = x_2, y_1 \neq y_2 \\ \mathcal{O}, & P_1 = P_2, y_1 = 0 \\ P_1, & P_2 = \mathcal{O} \\ P_2, & P_1 = \mathcal{O} \end{cases} \tag{1}$$

The geometrical interpretation for the first three cases of point addition on EC is summarized in Fig. 1. Point multiplication by a value n is treated as successively adding the point to itself n times. An efficient implementation for point multiplication is the point doubling algorithm[21].

The order of a point P is the smallest positive integer k such that $kP = \mathcal{O}$. The order of a point P always divides the order of the group $E(F_p)$. Let G be a point on the EC,E, then G is called a generator point with order N for the cyclic subgroup consisting of the points $\{G, 2G, 3G, \ldots, NG = \mathcal{O}\}$.

Framework for EC-based PRNGs

A PRNG is a critical element in any encryption system as it provides the system with a pseudorandom keystream. A good design of a PRNG should be sensitive to the initial state, give uniform distribution of output bits, and the period should be large enough to resist cryptanalysis attacks[22].

EC points are the primary source for any EC-based PRNG, which can generally fall into two schemes. The first scheme picks a generator point with a large order group and applies group operations to calculate new points and extract the random bits from the coordinates of each point. On the other hand, the second scheme calculates all required EC points, and then the coordinates of the points are used in producing the random bits. In this sense, a framework can be established where both design schemes can fit in. While the first scheme is called iterative because the points are generated one at a time, the second scheme is called non-iterative since all points are generated simultaneously. The proposed framework, shown in Fig. 2, consists of the following four main blocks.

- Parameters initialization: In this stage, EC parameters are initialized. In some design cases, other systems are integrated into the process and, hence, those system parameters are also initialized in this stage. For example, suppose that a chaos-based system is integrated into the design to enhance the randomness of the process and add extra complexity against different attacks. In this case, all parameters required by this chaotic system are initialized.

	Iterative design	Non-iterative design
EC selection	Predefined secure ECs/randomly generated ECs	Random generation of ECs
EC prime p	Very large (in the order of 192 bits or more)	Small (in the order of 16 bits or less)
EC points	Iteration over points of a cyclic subgroup	Evaluating the EC equation for all possible values of x
Period	Usually around the order of the generator point	It depends on the number of points generated
Suggested applications	Unknown or known data length (e.g., voice calls and video streaming/images)	Known data length (e.g., images and data files)

Table 1. Comparing iterative and non-iterative designs.

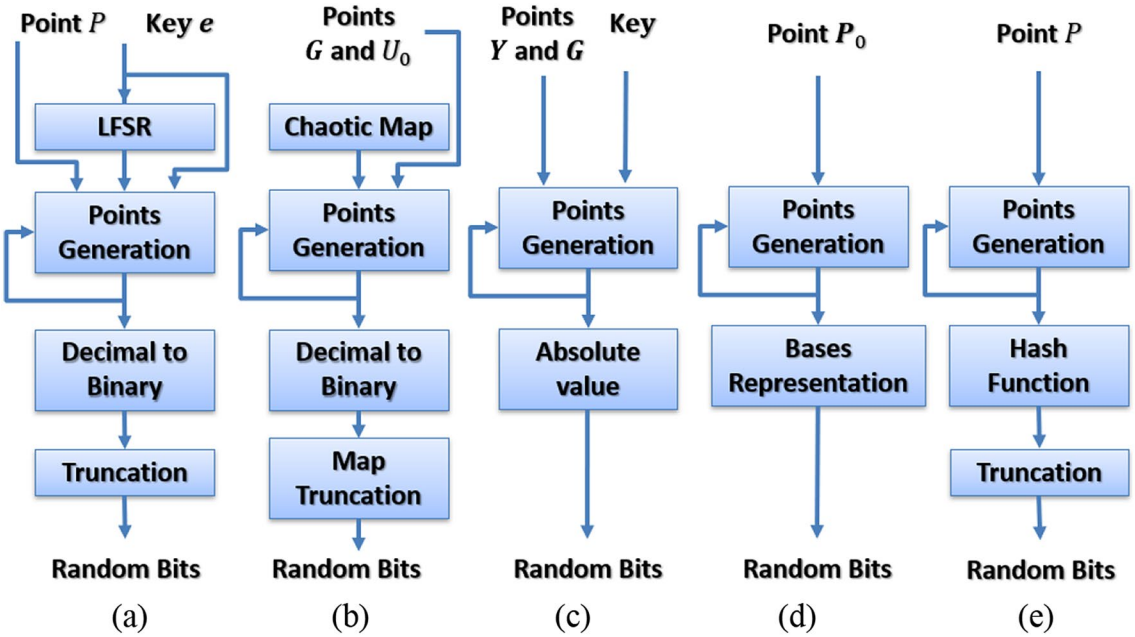

Figure 3. Simplified block diagrams for the iterative techniques in (**a**) Ref.[23], (**b**) Ref.[24], (**c**) Ref.[25], (**d**) Ref.[26], and (**e**) Ref.[27].

- Points generation: In the case of iterative designs, only one point is generated per iteration using an iterative equation. In general, the iterative equation consists of group operations such as point addition, doubling, and multiplication. The more operations exist, the more complex the generator is. In the case of non-iterative designs, all points required by the generator are calculated by evaluating the EC equation for all possible values of x or randomly selected values of x using some criteria.
- Points manipulation: In this stage, the produced points are processed based on some design criteria. For instance, the coordinates of the points can be converted into binary form. Other designs can use the coordinates values and apply mathematical formulas to produce a number.
- Bits extraction: This stage processes the output from the previous stage and generates the required pseudorandom bits. For example, a common logic in this stage includes bit truncation to satisfy particular design criteria.

Table 1 compares iterative and non-iterative designs with respect to different aspects. Clearly, each design category has its advantages. Depending on the application, the designer should choose the design that is more suitable. For instance, in applications that work with unknown data lengths like voice calls, it is better to use an iterative design as the period of the PRNG will be long enough to cover the amount of data that needs to be encrypted. In applications that work with known data length, like images, non-iterative designs can pick an EC with enough points to achieve the required period for PRNG. In the following subsections, some recent EC-based PRNG literatures are discussed and mapped into the proposed framework, which demonstrates the framework's flexibility.

Iterative designs. Several iterative PRNG algorithms were introduced during the last decade, such as the techniques shown in Fig. 3, where a simplified block diagram for each technique is depicted. Table 2 demonstrates the mappings of those techniques into the proposed framework.

Ref. no.	Parameters initialization	Points generation	Points manipulation	Bits extraction	Notes
Ref.[23], 2015	Point P on the curve and a key e Using e, find K_0 Using e, initialize LFSR	LFSR outputs C_i $K_i = X(K_{i-1}P) + C_{i-1}$ $S_i = K_iP + K_0P$	Convert the x-coordinate of S_i to binary form	Apply truncation on x-coordinate bits	The LFSR increased the period and introduced randomness in the keystream
Ref.[24], 2015	Pick an EC, E, and a generator point G on E Point $U_0 \in E(F_p)$	Increment index i Use the chaotic map to get the binary sequence b_i $U_i = i(1 + b_i)G + U_0$	Convert the point U_i into its binary form	Apply the map $U_{2\times2}(x, y)$ or the map $U_{3\times3}(x, y)$ on the point U_i, where $U_{k\times k}$ takes the rightmost k bits from x and y coordinates	Different chaotic maps can be used The chaotic map increased the randomness of the bitstream
Ref.[25], 2017	Two points Y and G SK_1 = primary key	$A = SK_iG$ $B = A + Y$ $C = B + G$ $SK_{i+1} = y_A + y_B + y_C$	$Z_i = \lvert x_A \times x_B \times x_C \rvert$	Read the value Z_i	The two points Y and G have very high orders
Ref.[26], 2019	Point P_0 of order n Pick $r \in [1, n-1]$ let $\alpha_1, \ldots, \alpha_p$ be a basis of F_{2^p}	$P_k = r^kP_0$ $x_k = X(P_k)$	Writing x_k $= s_k{}^{(1)}\alpha_1 + \cdots + s_k{}^{(p)}\alpha_p$	Read the sequence $s_k{}^{(i)}, i = 1, \ldots, p$	n has a large prime order r has a large multiplicative order mod n
Ref.[27], 2020	Select secure EC Select point P $S_0 = X(P)$	φ is a truncation function H is a hash function $S_i = \varphi(x[S_{i-1}P])$ $h_i = \varphi(H(S_i))$	Apply φ on the x-coordinate of $S_{i-1}P$ Apply φ on $H(S_i)$	Read lower-order bits from h_i	The hash function enhanced the statistical properties of the output bits

Table 2. Mapping of the surveyed iterative techniques into the proposed framework.

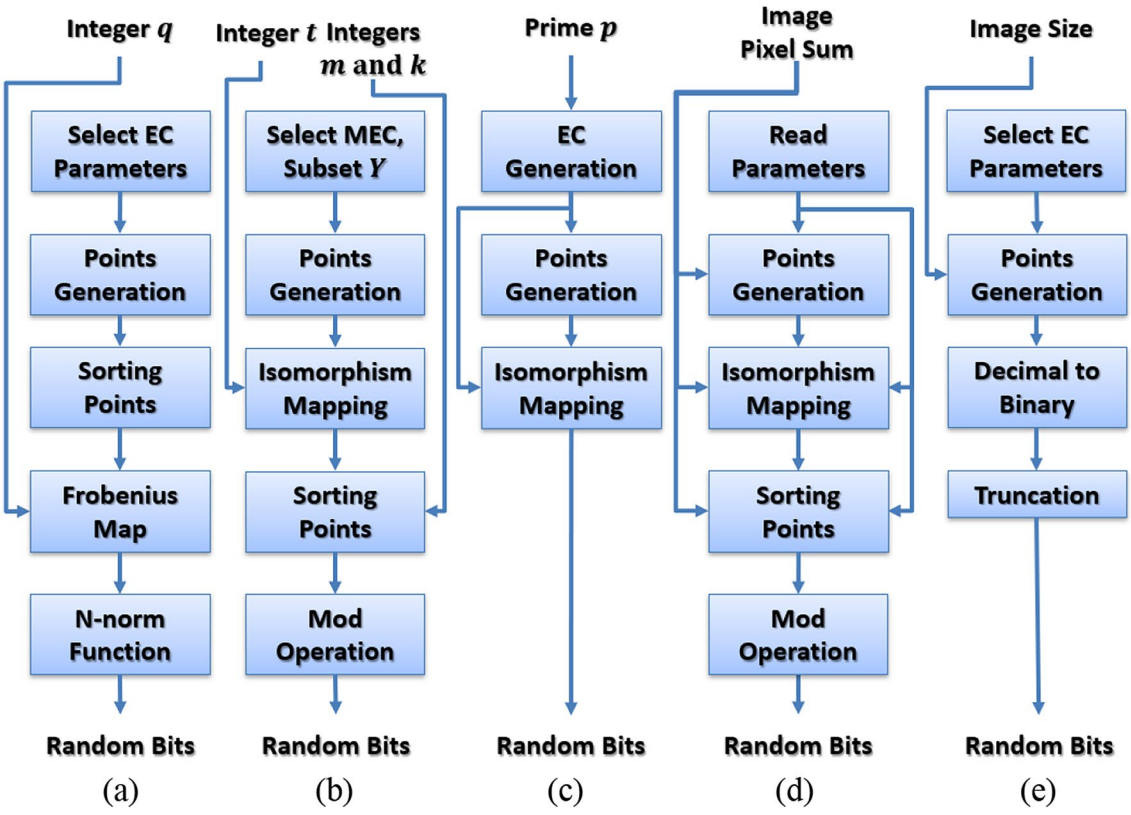

Figure 4. Simplified block diagrams for the non-iterative techniques in (**a**) Ref.[28], (**b**) Ref.[29], (**c**) Ref.[30], (**d**) Ref.[31], and (**e**) Ref.[32].

Non-iterative designs. Several non-iterative PRNG designs were proposed during the last decade, such as the designs shown in Fig. 4, where a simplified block diagram for each design is depicted. Table 3 demonstrates the mappings of those designs into the proposed framework.

In summary, EC point coordinates, in their binary form, can serve as a good source for random bits. The surveyed literature can be grouped into two categories, iterative and non-iterative. The main disadvantage of the first category is that the iterative equation can include too many EC group operations and may be combined with other operations regarding non-EC elements, which can be complex in limited resource systems. The main disadvantage of the second category is that it cannot be used with large prime numbers, where safe recommended ECs exist, because it is not possible to calculate all curve points. Therefore, this paper proposes to design an

Ref. no.	Parameters initialization	Points generation	Points manipulation	Bits extraction	Notes
Ref.[28], 2019	Randomly select EC parameters (p, a, b) Pick q as a parameter for Frobenius map	Apply brute force search on EC	Sorting points Apply Frobenius map on points Apply n-norm on projected points, then approximate to the nearest integer	Read integers after approximation	Azam et al.[33] introduced the ordering of EC points used to sort the points of EC
Ref.[29], 2021	Select a Mordell Elliptic Curve (MEC) E Select a subset $Y \subseteq [0, p-1]$ Select two integers m and k Select $t \in [1, \frac{p-1}{2}]$ Select a total order operator $<^*$	For each integer y in Y find the point (x, y) Calculate the point (t^2x, t^3y) then add it to set A	Sort the set A using the total order operator $<^*$	Read the y-coordinate $modm$ from the sorted list	MEC has the property of $a = 0$ $p \equiv 2 \ mod \ 3$
Ref.[30], 2021	Select large prime P Generate the curve E_a^P using brute force technique	Apply brute force search on E_a^P	$\varphi_\gamma(u, v) = \frac{v+\gamma u}{v-\gamma u}$ $\gamma^2 = a$ Use isomorphism φ_γ to map all points of E_a^P to F_p	Read mapped integers	$E_a^P : y^2 = x^3 + ax$ $\gamma \in F_p$
Ref.[31], 2021	Read input parameters Calculate S_I from plain text	Calculate isomorphic parameter t_r Map points of $E_{p,b}$ to $E_{p,t_r^6 b}$ using isomorphic parameter t_r	Select ordering O_r Select subset $A \subset E_{p,t_r^6 b}$ Sort A using O_r Pick a subset $A_r \subseteq [0, p-1]$ Select an integer h_r Sort A_r using ordering $<^*$ which depend on A, O_r and h_r	Calculate m_r Apply $modm_r$ to elements of A_r Read reduced elements of A_r	The PRNG is based on MECs The PRNG output is very sensitive to plain text
Ref.[32], 2022	Read EC secp256r1 parameters Read image size	For all pixels in the image, generate random points from the curve	Convert the y-coordinate of each point to binary form	Read the least significant 8 bits from each y-coordinate	The random generation of points is based on a predefined function

Table 3. Mapping of the surveyed non-iterative techniques into the proposed framework.

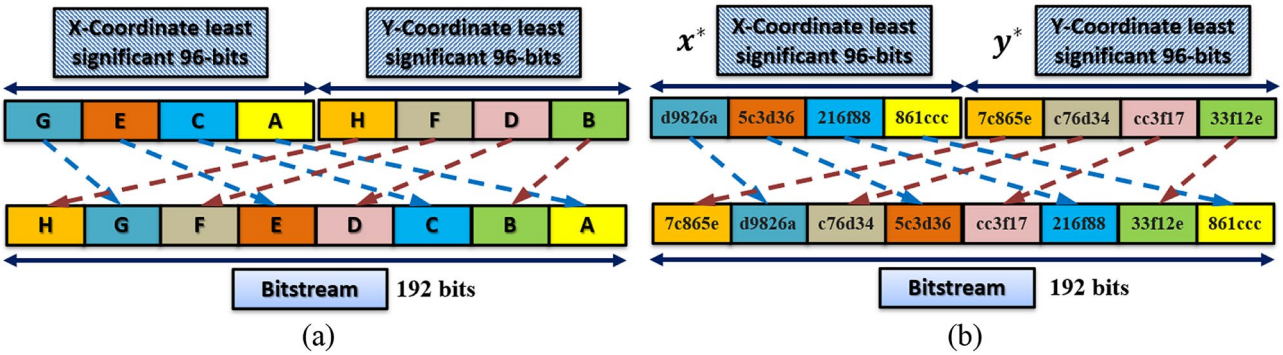

Figure 5. (**a**) Conversion from a point on EC to bitstream representation and (**b**) an example.

iterative PRNG with only one addition operation, which makes it suitable in a limited resource system and can be used in real-time applications using NIST-recommended safe ECs.

Proposed PRNG

With the proper choice of the EC parameters and a generator point G with a high order, usually a large prime number, the cyclic subgroup generated by the point G can be iterated. Moreover, using each point coordinate, pseudorandom numbers can be extracted. In this paper, a simple PRNG is designed and used in image encryption.

The PRNG is based on the iterative equation

$$P_{n+1} = P_n + P_0, \tag{2}$$

where $P_0 = KG$ is the initial base point of the PRNG,K is the system key value, and G is the generator point. If P_0 is changed, a completely new sequence of points is generated. For each point, the x and y coordinates are converted into their equivalent binary representation. Then, the least significant 96 bits from each coordinate are mixed to create a stream of 192 bits, as shown in Fig. 5a. For the example shown in Fig. 5b, consider a point $P(x, y)$. The least significant 96-bits, x^* and y^*, are extracted from each coordinate, respectively. Then, each 24-bits from x^* and y^* are extracted and mixed to form the final bitstream. The resulting bits are random because the hopping from one point to another gives an entirely different point regarding coordinate values, and because of the mixing between the x and y coordinates. It is important not to extract more bits from each coordinate because higher bits are not chaotic enough, and the more bits used, the more the bitstream is not secure and can be attacked as pointed out in Ref.[34].

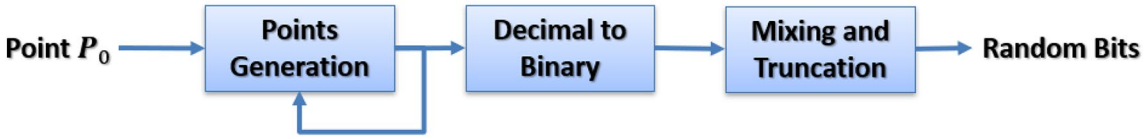

Figure 6. Simplified block diagram for the proposed PRNG.

Parameters initialization	Points generation	Points manipulation	Bits extraction	Notes
Select secure EC Select K $P_0 = KG$	Increment index n $P_{n+1} = P_n + P_0$	Convert the x and y coordinates of the point P_n into its binary form	Read least 96 bits from both x and y coordinates Mix the bits from x and y coordinates	Any secure curve can be used K is at least 128 bits

Table 4. Mapping of the proposed PRNG into the proposed framework.

In the PRNG design, every point from the EC can produce 192 bits, and since the generator is used to encrypt images, every 24 bits (no. of bits in each pixel) are parsed from the bitstream and then used to encrypt the image pixel. Hence, in pixel terms, a total of $192/24 = 8$ pixels can be encrypted using only one point from the EC.

The PRNG design is inspired by the proposed framework, where the number of operations in each stage is minimized to achieve better performance. Figure 6 shows the simplified block diagram for the proposed PRNG, whereas Table 4 shows the mapping of this design into the proposed framework. The proposed PRNG has only one EC addition operation in the points generation stage, which helps in speeding up the time consumed in this stage. Furthermore, only decimal to binary conversion is applied in the points manipulation stage, and mixing (bit shifting) and truncation operations are performed in the bits extraction stage. In this sense, the design of the PRNG is optimized for speed and low resources.

In practice, the EC parameters and G should be chosen such that the order of G is a large prime number. Hence, the period of such PRNG is significantly large enough to be used in encryption applications. In this work, the PRNG uses Curve-192, although any other recommended secure curve can be used as well. This curve is one of the NIST's recommended curves[35]; its prime modulus p is 192 bits, the base point G has 189 bits and 187 bits in the x and y coordinates, respectively, and its order n is 192 bits. Iterating the cyclic group generated by G, the average number of bits in each point x and y coordinates is close to that of the generator point G.

Proposed encryption system

The block diagram of the proposed encryption system is shown in Fig. 7, where the system consists of two main stages necessary to achieve Shannon's confusion and diffusion properties[36]. The first stage is the substitution stage, where pixel values are changed. The second stage is the permutation stage, where pixel locations are shuffled across the image. For the system to be sensitive to input changes, the algebraic sum of all pixels in the three channels is calculated and used to modify the permutation stage parameters. In this sense, the system is protected from different differential attack attempts.

Substitution stage. In this stage, the output from the PRNG is *Xored* with the image pixel. In addition, a delay element is used to make the current encrypted pixel's value dependent on the last encrypted pixel value. Hence, this provides the system with more strength against differential attacks.

The substitution phase can be represented using the equation

$$E_R = RN_i \oplus I_R \oplus D_R, \tag{3a}$$

$$E_G = RN_{i+1} \oplus I_G \oplus D_G, \tag{3b}$$

$$E_B = RN_{i+2} \oplus I_B \oplus D_B, \tag{3c}$$

where E_R, E_G, and E_B are the encrypted pixel values for the red, green, and blue channels, respectively.RN_i is the ith byte from the PRNG bitstream. I_R, I_G, and I_B are the image pixel values for the red, green, and blue channels, respectively. D_R, D_G, and D_B are the previous encrypted pixel values for the red, green, and blue channels, respectively, and each is initialized with the value of 0.

Permutation stage. Arnold's cat map is used in permuting the image pixels, as defined by:

$$\begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix} = \begin{pmatrix} 1 & a \\ b & 1+ab \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} mod M, \tag{4}$$

where $a, b \in \{1, 2, \ldots, M-1\}$, M is the square matrix size,$x, y \in \{1, 2, \ldots, M\}$ are the original pixel location and x_{new}, y_{new} are the new pixel location. The values of a and b are calculated from the system key and then modified using:

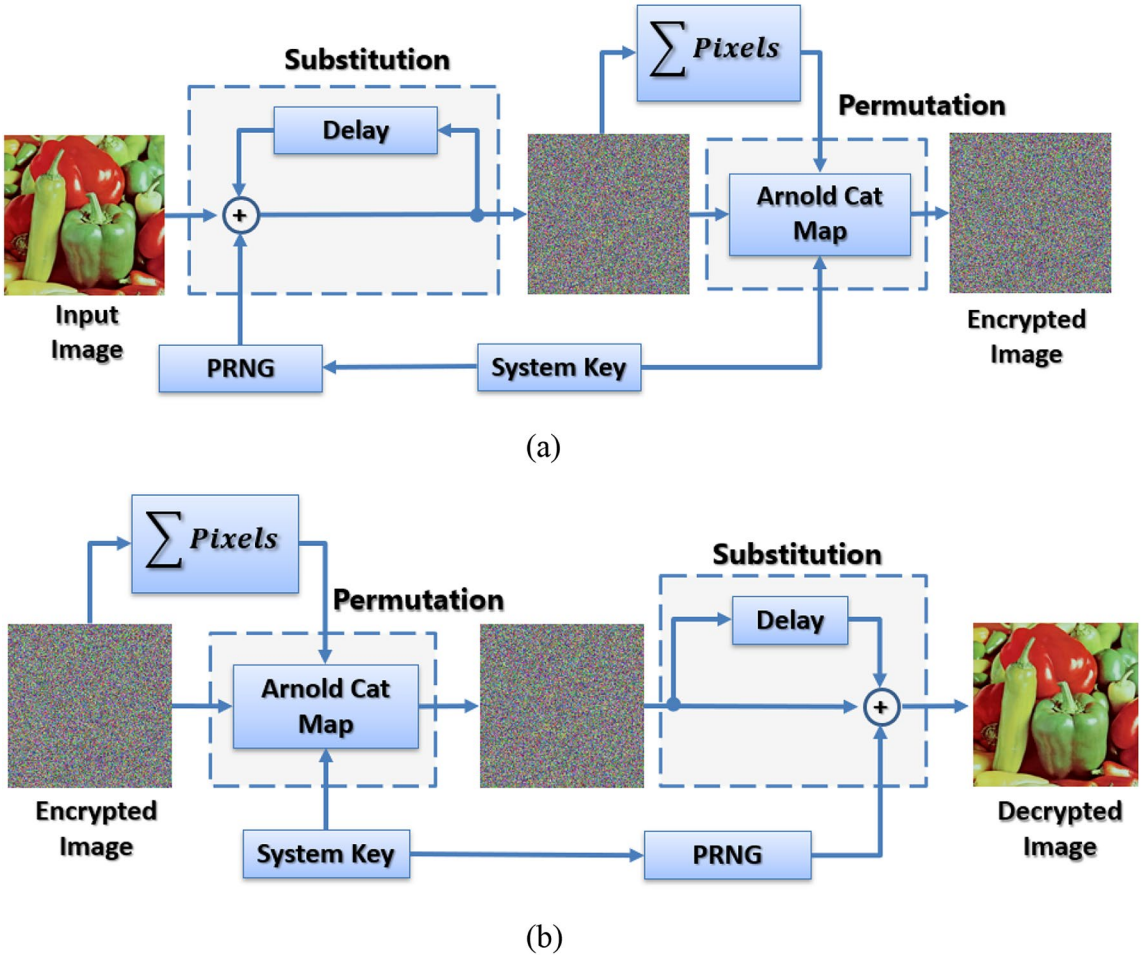

Figure 7. Block diagram for (**a**) the encryption system and (**b**) the decryption system.

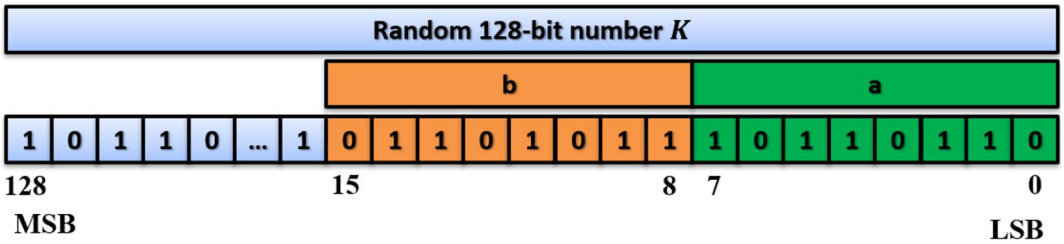

Figure 8. System key construction.

$$S = sum(image\ pixels), \tag{5a}$$

$$a = mod(S + a_{key}, M - 1) + 1, \tag{5b}$$

$$b = mod(S + b_{key}, M - 1) + 1, \tag{5c}$$

where a_{key}, b_{key} are 8-bit numbers extracted from the system key as shown in Fig. 8, and *mod* returns the remainder after division.

System key. The system key should be at least 128 bits, long enough to resist brute-force attacks in cryptographic applications. Furthermore, any change in the key, even a one-bit change, should produce completely different output from the original key. As shown in Fig. 8, a random 128-bit number K is selected to be the system key where Arnold's cat map parameters a and b are extracted from this key.

For security purposes, the generator point G provided by the NIST Curve-192 cannot be used as the base point of the PRNG. Therefore, in the beginning, the point $P_0 = KG$ is calculated. It is worth mentioning that the large value of K will not affect the speed of calculating the point P_0 as mentioned earlier in the introduction.

Evaluation criteria

This section discusses different evaluation criteria used to evaluate the proposed PRNG and encryption system.

NIST statistical test suite. NIST SP-800-22 is a group of 15 tests applied on bitstreams to decide the randomness of the bits[37]. If any of the tests failed, the bitstream is not recommended to be used in cryptography applications. The output from this test is validated by the P-value distribution (PV) and the proportion of passing sequences (PP). For a truly random sequence, the PV is equal to 1, while for a nonrandom sequence, the PV approaches 0. A significant value α controls the success of each test. If $\mathrm{PV} \geq \alpha$, then the sequence passes the test, otherwise, it fails the test. In case of cryptography applications, $\alpha = 0.01$, which means that if more than 1% of the sequence fails the test, then the complete sequence is considered nonrandom.

Correlation coefficients of image pixels. This metric measures how much image pixels are correlated to each other. This measure is generally applied to adjacent pixels in the horizontal, vertical, and diagonal directions. It is calculated using:

$$Cov(x,y) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\mathrm{x_i} - \frac{1}{\mathrm{N}}\sum\nolimits_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{x_j}\right)\left(\mathrm{y_i} - \frac{1}{\mathrm{N}}\sum\nolimits_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{y_j}\right), \tag{6a}$$

$$D(x) = \frac{1}{\mathrm{N}}\sum\nolimits_{\mathrm{i}=2}^{\mathrm{N}}\left(\mathrm{x_i} - \frac{1}{\mathrm{N}}\sum\nolimits_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{x_j}\right)^2, \tag{6b}$$

$$\rho = \frac{Cov(x,y)}{\sqrt{\mathrm{D(x)}}\sqrt{\mathrm{D(y)}}}, \tag{6c}$$

where N is the number of elements in the two vectors x and y. For typical images, the value of ρ is close to 1, while for encrypted images, the value of ρ should be closer to 0.

Differential attack measures. This attack studies the relationship between two encrypted images after changing one pixel in the source image. Three measures are used, which are the Mean Absolute Error (MAE), the Number of Pixels Change Rate (NPCR), and the Unified Average Changing Intensity (UACI)[38]. Expected values for MAE, NPCR, and UACI are around 100, 99.6%, and 33.34%, respectively. Let E be the source image, $E1$ be the encrypted image and $E2$ be the encrypted image after changing one pixel in the original image, then

$$\mathrm{MAE} = \frac{1}{W \times H}\sum_{\mathrm{i}=1}^{H}\sum_{\mathrm{j}=1}^{W}|E1(\mathrm{i},\mathrm{j}) - E(\mathrm{i},\mathrm{j})|, \tag{7a}$$

$$D(i,j) = \begin{cases} 0 & E1(i,j) = E2(i,j) \\ 1 & E1(i,j) \neq E2(i,j) \end{cases}, \tag{7b}$$

$$NPCR = \frac{1}{W \times H}\sum_{i=1}^{H}\sum_{j=1}^{W}D(i,j) \times 100\%, \tag{7c}$$

$$UACI = \frac{1}{W \times H}\sum_{i=1}^{H}\sum_{j=1}^{W}\frac{|E1(i,j) - E2(i,j)|}{255} \times 100\%, \tag{7d}$$

where W and H are the width and height of the image, respectively.

Mean square error (MSE). This metric is used to measure the error between two images. Let E be the source image and D be the wrong decrypted image, then

$$MSE = \frac{1}{W \times H}\sum_{i=1}^{H}\sum_{j=1}^{W}\left[E(i,j) - D(i,j)\right]^2. \tag{8}$$

Entropy analysis. Entropy is a measure of the predictability of random sources. For a source that produces N symbols with probabilities $P(S_i), i = 1, 2, .., N$, the entropy of that source is calculated using:

Test	K_1		$K_1^* = K_1 + 1$		K_2		K_3	
	PV	PP	PV	PP	PV	PP	PV	PP
Frequency	0.637	0.958	0.213	1.000	0.637	1.000	0.437	1.000
Block frequency	0.350	1.000	0.163	1.000	0.637	1.000	0.437	1.000
Cumulative sums	0.592	1.000	0.300	1.000	0.534	1.000	0.508	1.000
Runs	0.276	1.000	0.534	0.958	0.013	1.000	0.437	1.000
Longest run	0.437	1.000	0.740	1.000	0.213	1.000	0.122	1.000
Rank	0.740	0.958	0.740	1.000	0.350	0.958	0.122	1.000
FFT	0.035	0.958	0.637	1.000	0.534	1.000	0.122	1.000
Non-overlapping template	0.339	0.991	0.322	0.993	0.320	0.990	0.345	0.991
Overlapping template	0.834	1.000	0.025	0.958	0.276	0.958	0.013	0.958
Universal	0.163	1.000	0.025	0.958	0.911	1.000	0.437	0.958
Approximate entropy	0.437	1.000	0.740	1.000	0.091	1.000	0.637	1.000
Random excursions	0.055	0.992	0.167	1.000	0.088	0.993	0.311	0.981
Random excursions variant	0.098	1.000	0.218	0.996	0.066	0.964	0.401	0.987
Serial	0.034	0.958	0.437	1.000	0.451	0.979	0.209	1.000
Linear complexity	0.534	1.000	0.834	1.000	0.276	1.000	0.834	0.958
Final result	Success		Success		Success		Success	

Table 5. NIST results for the PRNG.

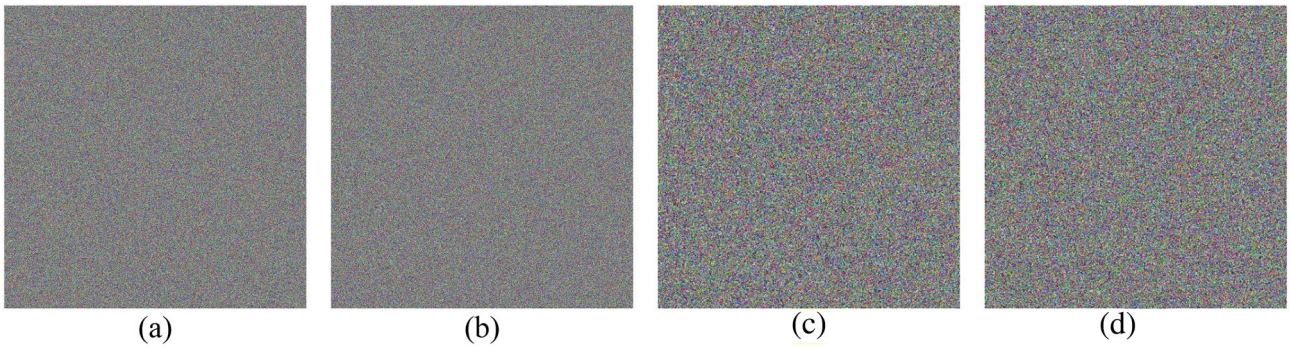

Figure 9. Output bitstreams of the PRNG represented as images in four cases: (**a**) key K_1, (**b**) key K_1^*, (**c**) key K_2, and (**d**) key K_3.

$$Entropy = -\sum_{i=1}^{N} P(S_i) log_2 P(S_i). \tag{9}$$

For a random source, this value approaches $\log_2 N$. In the case of color images, this value approaches 8 for each channel.

Analysis results

In this section, the randomness and efficiency of the PRNG are, first, demonstrated. Then, the encryption system is evaluated using the Peppers image of size 256×256 as well as some additional images from the USC-SIPI database[39] of size 512×512. The system key sensitivity is examined by changing one bit and observing the results. Finally, the computation complexity is analyzed and comparisons with related literature are given.

PRNG results. The proposed PRNG is evaluated using thirty different 128-bit random K values ($K_1, ..., K_{30}$). Let $K_1 = ede8a3004ce2b2579c937b3874aba2de$ be a 128-bit random number and let $K_1^* = K_1 + 1$. Let $K_2 = fe23c064b1cc841a0027ad705ac47d98$ and $K_3 = b6c575a9a76716fcbccdcf16740fb22b$. The choice of K_1^* was made to test the sensitivity of the PRNG for only a one-bit change in the key. In order to test the PRNG using the NIST test suite, a total of $25165824 = 24 \times 2^{20}$ bits are generated, equal to the number of bits found in a color image of size 1024×1024.

The NIST results for the PRNG are shown in Table 5. For the sensitivity test using K_1 and K_1^*, the results show that the bitstreams are random and have passed all 15 tests. Furthermore, the bitstreams are converted into two color images, and the results are shown in Fig. 9. Visual inspection of the images supports the NIST results. The correlation between the two bitstreams is calculated and found to be 0.0009, demonstrating the PRNG's sensitivity to one-bit change in the key. As for other test cases, $(K_2, K_3, \ldots, K_{30})$, similar results are achieved. Accordingly, Table 5 and Fig. 9 include the results for K_2 and K_3 as representatives for the remaining cases.

Ref. no.	EC operations	Non-EC operations	EC selection	Period T
Ref.[23], 2015	Two multiplications One addition	Clocking the LFSR One addition Truncation	ECs defined over the field F_{2^m}	$T = C \times (2^m - 1)$ where $C \geq 1$ and m is the length of LFSR in bits
Ref.[24], 2015	One multiplication One addition	Chaotic map iteration One addition One multiplication	EC defined over F_p	$T < p^{1-\delta}$ where $\delta > 0$
Ref.[25], 2017	One multiplication Two additions	Two multiplications Two additions One absolute value	The Internet Engineering Task Force (IETF)[40]	Not given
Ref.[26], 2019	One multiplication	One power Basis representation	Koblitz EC defined over F_p	$T = (n-1)/2$ where n is the order of generator point
Ref.[27], 2020	One multiplication	Hash function Truncation	EC defined over F_p	Not given
This Work	One addition	Two truncations	NIST recommended ECs	Order of generator point

Table 6. Comparison between iterative methods and this work.

	Ref.[28], 2019	Ref.[29], 2021	This work	
			MATLAB implementaion	C# implementaion
Bitrate in Mbps	0.070444	0.072140	0.09755	0.55869

Table 7. Comparison of bitrates in this work and in other PRNGs over ECs.

Table 6 compares some iterative methods with this work. Although all iterative methods can achieve a long period with the proper choice of the EC parameters, the complexity for each technique is not the same. The more operations involved in the design, the more complex the design is. Clearly, the proposed PRNG contains the least number of EC and non-EC operations and, hence, has the least complexity.

The proposed PRNG is examined to determine the bitrate that can be achieved. The experiment is conducted on a Dell laptop with processor Intel Core i7-1065G7 CPU @ 1.30 GHz, running Windows 10 with 16 GB of RAM. Two implementations for the PRNG are considered; the first one uses C# under .net framework 4.7 and the second one uses MATLAB R2015a. The proposed PRNG is run for 30 times, with 65,536 bytes generated in each run. Then, the average bitrate is calculated for both the MATLAB and C# implementations. In the case of MATLAB, the JAVA BigInteger class is used, leading to runtime overhead due to calls between MATLAB and JAVA. In the case of C#, however, no overhead is encountered as C# contains an implementation for the BigInteger class. Table 7 compares the bitrates achieved in Megabits per second (Mbps) by the proposed PRNG and other related PRNGs based on ECs. The bitrates achieved by this work are better than those achieved by other related works, which is attributed to the few used operations as shown in Table 6.

Encryption system results. Using the same system key K_1(see Fig. 8), the values for a_{key} and b_{key} are 222 and 162, respectively. Figure 10 shows the histogram plots for Peppers and encrypted Peppers where the input image has clear peaks while the encrypted image has a uniform distribution across all channels, as supported by the correlation results in Table 8. Furthermore, it is clear from the visual inspection that the encrypted output image shows complete randomness. Figure 11 shows the adjacent pixel values and correlation values in horizontal, vertical, and diagonal directions for the red channel of Peppers and encrypted Peppers. Similar results are achieved in the green and blue channels.

Table 8 shows the correlation results for encrypted Peppers in horizontal, vertical, and diagonal directions. The values are close to zero, indicating how much the pixels are not correlated anymore after the encryption. The differential attack measures are calculated by taking the average values after changing the pixel value in ten random pixels. It is clear from the results that the dependence of Arnold's cat map parameters on the image, as given by Eq. (5), enhanced the results of the differential attack measures. Furthermore, the MSE results show how far is the encrypted image from the source image. At the same time, the entropy values are very close to 8, which provides evidence of the randomness existing in the encrypted images.

In addition, Fig. 12 shows the statistical analysis results for encrypted Peppers using 30 different system keys ($K_1, K_2, \ldots, K_{30}$). For the box plot, the correlation results in the horizontal, vertical, and diagonal directions are given. The horizontal and vertical results are distributed symmetrically, while the diagonal results are positively skewed. The interquartile maximum range is 0.0027, which means that the three distributions are very concentrated. For the entropy histogram, NPCR histogram, and UACI histogram, it is clear that most of the results fall in the highest range for each test indicating the quality of the encrypted image regardless of the used system key.

Furthermore, Table 9 summarizes the statistical analysis results where all results are in the good, expected ranges. The results provide evidence that the system is stable with respect to different system keys. The small values of the standard deviation demonstrate that, for any system key, the results are expected to be very close to the average results achieved.

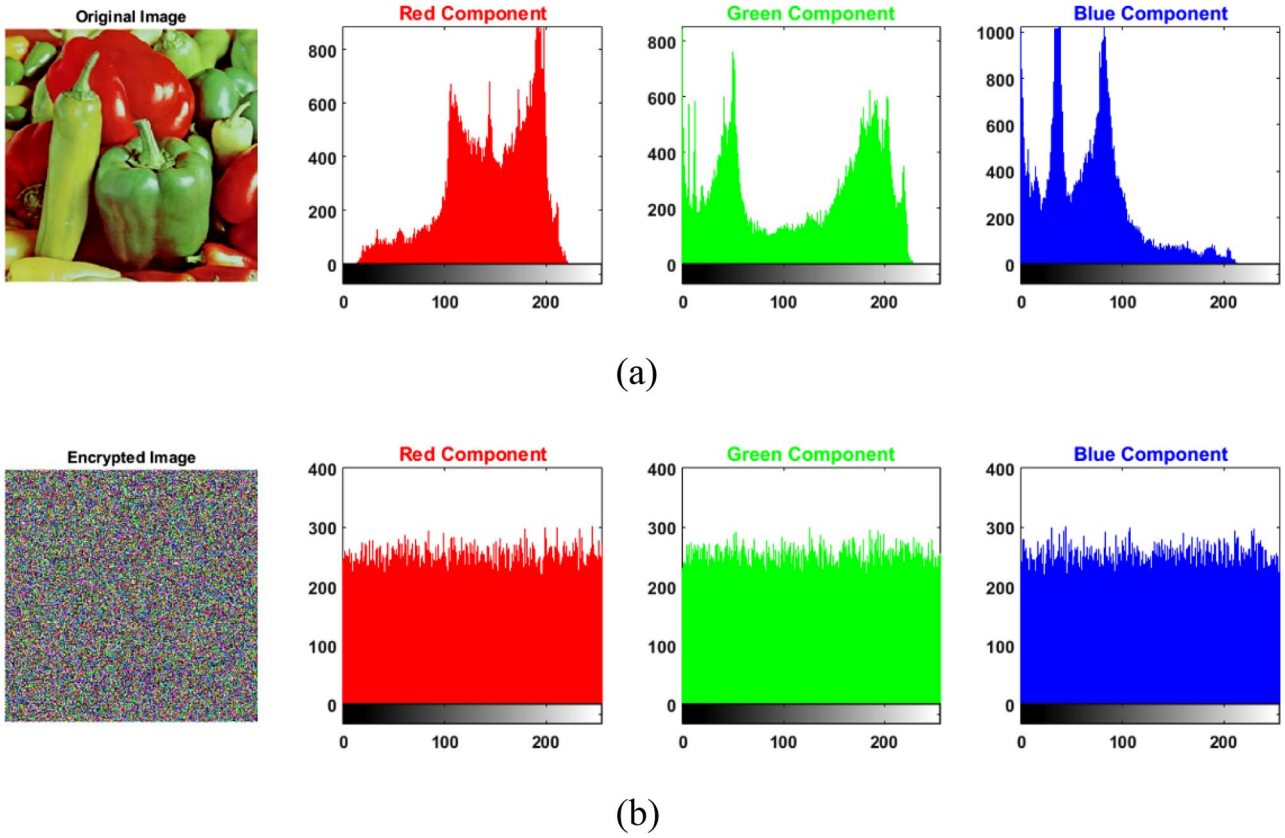

Figure 10. Histograms for the three-color channels in (**a**) Peppers and (**b**) encrypted Peppers.

	Pixel correlations			MSE	Entropy	Differential attack measures		
	Horz	Vert	Diag			MAE	NPCR (%)	UACI (%)
R	− 0.0052	− 0.0001	− 0.0013	7703.80	7.9971	72.8052	99.6043	33.4037
G	− 0.0045	0.0008	− 0.0015	11,068.50	7.9973	85.9236	99.6022	33.4538
B	− 0.0028	− 0.0019	0.0001	11,467.20	7.9967	87.5722	99.6123	33.3678
Avg	0.0042	0.0009	0.0010	10,079.84	7.9970	82.1003	99.6063	33.4085

Table 8. Analysis results for encrypted Peppers using the system key K_1.

Moreover, Table 10 shows the analysis results for different standard images from the USC-SIPI image database[39] of size 512 × 512 and the black image. The results show that the system successfully encrypts all images giving good measure values within the required ranges.

System key sensitivity results. The sensitivity of the system key is examined by changing one bit in it, then decrypting an image with this wrong key and checking the results. Since the system key value is used in calculating the base point P_0 used by the PRNG, any change in any bit produces a new base point. Hence, the PRNG will not be synchronized with the encrypted image. Two cases are examined, Case I, where the least significant bit is changed, and Case II, where the 9th bit is changed. In Case I, the value of a_{key} is changed, whereas the value of b_{key} is unchanged. While in Case II, the value of a_{key} is kept unchanged, whereas the value of b_{key} is changed.

Table 11 shows the results for the two test cases. The PRNG was not synchronized with the encrypted image in cases I and II. Therefore, the results for the MSE are large, and entropy values indicate the complete randomness of the wrong decrypted images. These results are supported by visual inspection of the decrypted images, as shown in Fig. 13.

Computation complexity. The system's time complexity can be derived by using the system block diagram of Fig. 7. Assuming that the image size is equal to M × N, then the PRNG takes (M × N)/8 iterations to produce the required random numbers. Therefore, the complexity for the PRNG is $\mathcal{O}((M \times N)/8) \approx \mathcal{O}(M \times N)$. The substitution stage performs XOR operations for each pixel in the image and, hence, the complexity for the substitution stage is $\mathcal{O}(M \times N)$. Next, the summation block cumulatively adds all pixel values and, hence, the complexity of this block is $\mathcal{O}(M \times N)$. Finally, the permutation stage maps each pixel location to a new

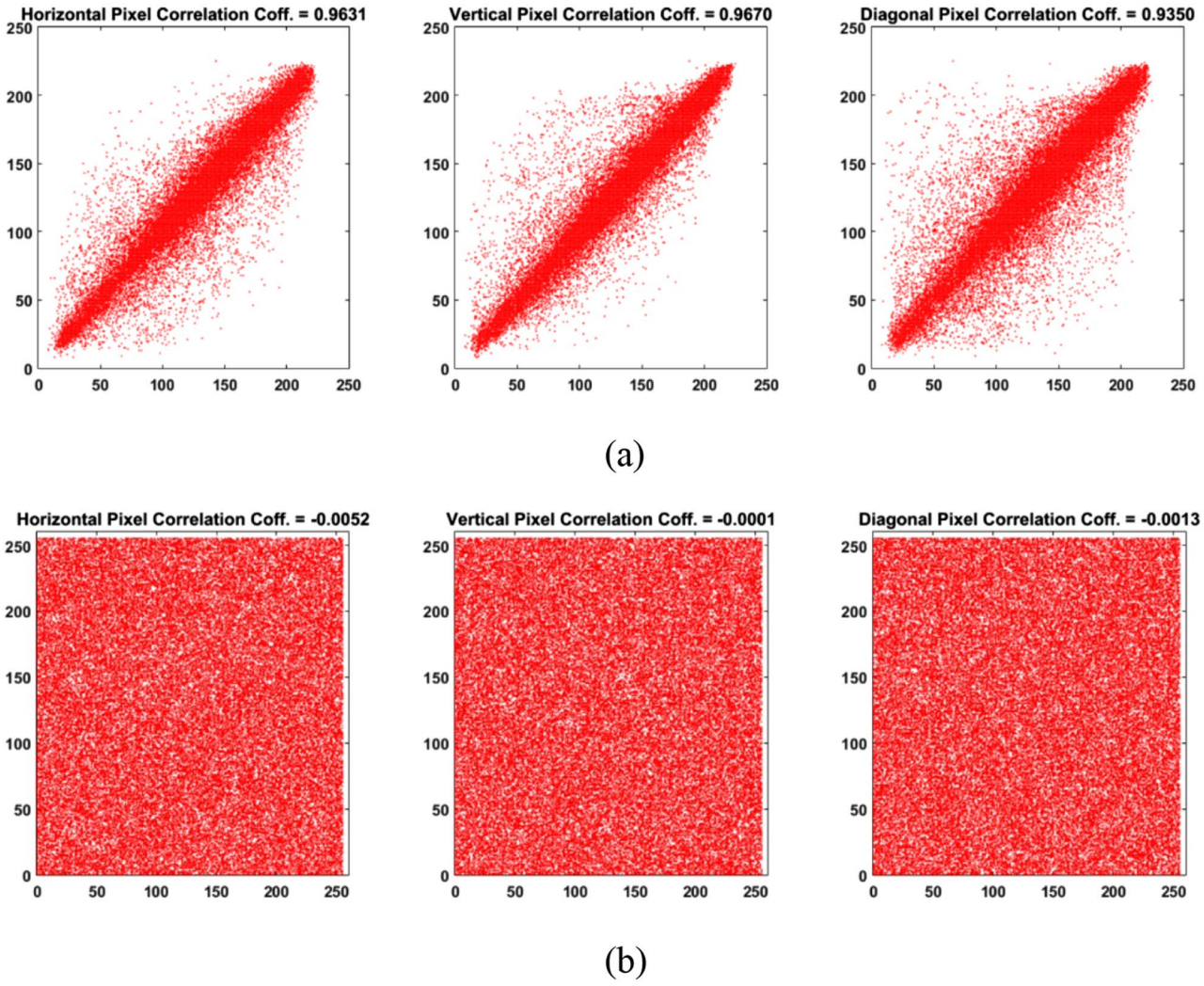

Figure 11. Adjacent pixel values in horizontal, vertical, and diagonal directions in (**a**) Peppers and (**b**) encrypted Peppers for the red channel.

location and the complexity for this stage is also $\mathcal{O}(M \times N)$. Therefore, the total complexity for the system is $\mathcal{O}(M \times N) + \mathcal{O}(M \times N) + \mathcal{O}(M \times N) + \mathcal{O}(M \times N) = 4 \times \mathcal{O}(M \times N) \approx \mathcal{O}(M \times N)$.

Comparison with related literature. Table 12 compares the results accomplished by this work with other related work in terms of pixel correlations, differential attack measures, and entropy of an encrypted gray-scale image of size 256×256. The results show that the security measures are close to each other. Furthermore, Table 13 gives the total execution time, using MATLAB R2015a, for the proposed encryption and decryption systems compared to other related work. The proposed system performance is clearly better.

Conclusions

The presented PRNG has a simple and efficient design, which was achieved by utilizing the proposed framework through minimizing the EC and non-EC operations. Consequently, the introduced encryption system utilizes low computational resources and, hence, it is a good candidate for real-time applications.

In conclusion, ECs are good candidates for designing PRNGs. The number of bits in each point coordinate is suitable for bit extraction in secure curves with large prime numbers. Furthermore, the system's security is inherited from the difficulty of the DLP. Finally, the proposed framework for designing PRNGs can help in optimizing the system design by simplifying each block as much as possible, resulting in fast and secure bitstream output. Future work includes enhancing bit extraction criteria to increase the number of bits extracted from each point coordinate and utilizing ECs in generating dynamic S-boxes.

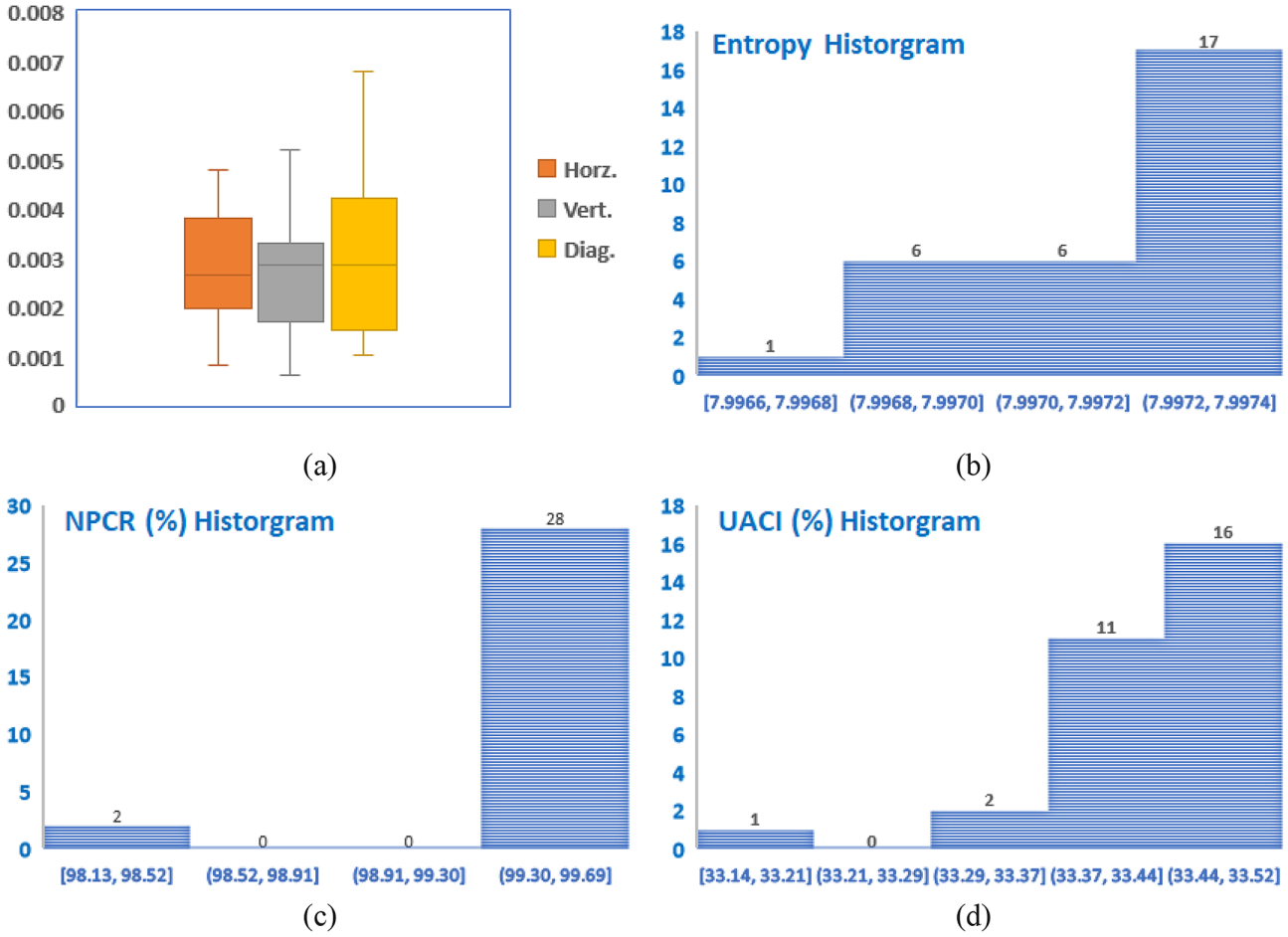

Figure 12. Statistical analysis results for encrypted Peppers using 30 different system keys ($K_1, K_2, \ldots, K_{30}$): (**a**) correlation box plot, (**b**) entropy histogram, (**c**) NPCR histogram, and (**d**) UACI histogram.

	Pixel correlations			MSE	Entropy	Differential attack measures		
	Horz	Vert	Diag			MAE	NPCR (%)	UACI (%)
Min	0.0008	0.0006	0.0010	9996.03	7.9966	81.6007	98.1324	33.1360
Max	0.0048	0.0069	0.0068	10,132.41	7.9974	82.3008	99.6155	33.5182
Avg	0.0029	0.0028	0.0031	10,055.13	7.9971	81.9509	99.5061	33.4360
Std	0.0011	0.0014	0.0015	28.91	0.0002	0.1504	0.3502	0.0731

Table 9. Summary of the statistical analysis results for encrypted Peppers using 30 different system keys ($K_1, K_2, \ldots, K_{30}$).

Img	Original image corr			Encrypted image corr			Entropy		Differential attack measures		
	Horz	Vert	Diag	Horz	Vert	Diag	Orig	Enc	MAE	NPCR (%)	UACI (%)
House	0.9550	0.9563	0.9190	0.0009	0.0023	0.0007	7.3602	7.9993	78.8235	99.6081	33.4772
San Diego 2.1.02	0.7937	0.7731	0.6973	0.0008	0.0023	0.0023	7.1394	7.9994	75.8555	99.5761	33.4548
Oakland 2.1.04	0.7572	0.7814	0.6810	0.0022	0.0006	0.0017	6.3841	7.9993	72.0607	99.5990	33.4842
Woodland 2.1.06	0.9073	0.8948	0.8429	0.0010	0.0022	0.0012	7.3475	7.9993	72.9742	99.6104	33.4827
Earth 2.1.11	0.9629	0.9680	0.9416	0.0010	0.0026	0.0013	6.9287	7.9993	72.0452	99.6077	33.4328
Splash 4.2.01	0.9858	0.9871	0.9751	0.0008	0.0018	0.0027	6.6530	7.9993	86.6092	99.6126	33.4819
Mandrill 4.2.03	0.8986	0.8373	0.8097	0.0009	0.0018	0.0027	7.6444	7.9992	76.3159	99.6054	33.4402
Airplane 4.2.05	0.9648	0.9533	0.9272	0.0018	0.0016	0.0017	6.5768	7.9993	83.0794	99.6035	33.4515
Boat 4.2.06	0.9661	0.9632	0.9493	0.0018	0.0001	0.0020	7.3896	7.9992	82.2794	99.6117	33.4810
Peppers 4.2.07	0.9704	0.9715	0.9576	0.0017	0.0009	0.0025	7.2978	7.9993	82.1370	99.6011	33.4705
Black Image	1.0000	1.0000	1.0000	0.0017	0.0014	0.0009	0.0000	7.9993	127.4746	99.5825	33.4626

Table 10. Analysis results for some images from the USC-SIPI image database and the black image.

Test	MSE			Entropy		
	Red	Green	Blue	Red	Green	Blue
Exact key	0.00	0.00	0.00	7.2946	7.5483	7.0823
Case I	7732.79	10,904.86	11,449.54	7.9975	7.9974	7.9974
Case II	7694.79	11,013.62	11,400.54	7.9973	7.9979	7.9972

Table 11. Decryption results with different keys.

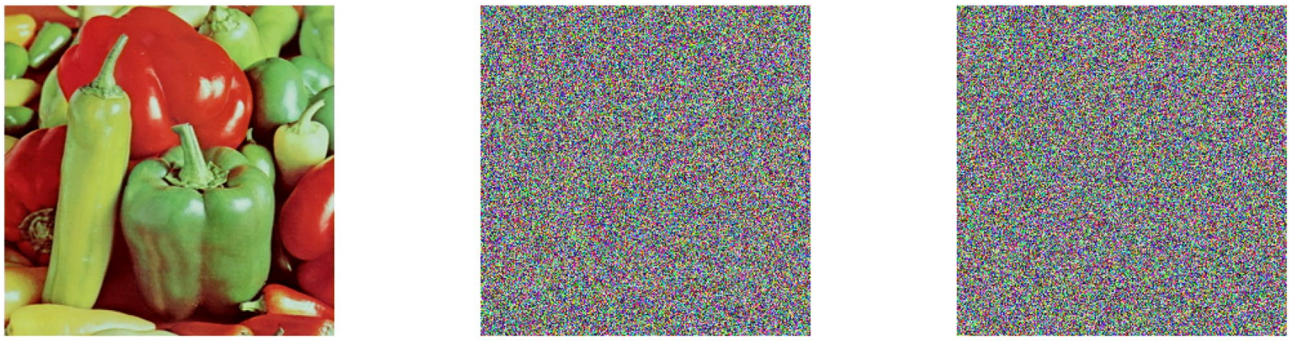

Figure 13. Decryption using (**a**) exact key, (**b**) case I, and (**c**) case II.

Ref. no.	Pixel correlations			Differential attack measures		Entropy
	Horz	Vert	Diag	NPCR (%)	UACI (%)	
Ref.[23], 2015	0.0025	0.0037	0.0011	99.63	33.56	7.9968
Ref.[28], 2019	0.0012	0.0003	0.0010	99.60	33.48	7.9993
Ref.[31], 2021	− 0.0044	− 0.0007	− 0.0031	99.60	33.34	7.9971
This work	0.0027	− 0.00004	− 0.0056	99.59	33.44	7.9971

Table 12. Comparison with related work for an image of size 256 × 256.

Ref. no.	Encryption + decryption time (s)
Ref.[32], 2022	21.27
Ref.[41], 2015	7.73
This work	3.78

Table 13. Comparing the execution times for an image of size 256 × 256.

Data availability

The data used in this paper are available from the corresponding author upon request.

Received: 14 May 2022; Accepted: 20 July 2022
Published online: 02 August 2022

References

1. Ismail, S. M., Said, L. A., Radwan, A. G., Madian, A. H. & Abu-ElYazeed, M. F. A novel image encryption system merging fractional-order edge detection and generalized chaotic maps. *Signal Process.* **167**, 107280 (2020).
2. Zhu, L. *et al.* A stable meaningful image encryption scheme using the newly-designed 2D discrete fractional-order chaotic map and Bayesian compressive sensing. *Signal Process.* **195**, 108489 (2022).
3. Gao, X., Yu, J., Banerjee, S., Yan, H. & Mou, J. A new image encryption scheme based on fractional-order hyperchaotic system and multiple image fusion. *Sci. Rep.* **11**, 1–21 (2021).
4. Ibrahim, S. & Alharbi, A. Efficient image encryption scheme using Henon map, dynamic S-boxes and elliptic curve cryptography. *IEEE Access* **8**, 194289–194302 (2020).
5. Abd El-Latif, A. A., Abd-El-Atty, B., Amin, M. & Iliyasu, A. M. Quantum-inspired cascaded discrete-time quantum walks with induced chaotic dynamics and cryptographic applications. *Sci. Rep.* **10**, 1–16 (2020).
6. Ye, G., Jiao, K. & Huang, X. Quantum logistic image encryption algorithm based on SHA-3 and RSA. *Nonlinear Dyn.* **104**, 2807–2827 (2021).
7. Abd-El-Hafiz, S. K., Radwan, A. G., Abdel Haleem, S. H. & Barakat, M. L. A fractal-based image encryption system. *IET Image Process.* **8**, 742–752 (2014).
8. Mikhail, M., Abouelseoud, Y. & Kobrosy, G. E. Two-phase image encryption scheme based on FFCT and fractals. *Secur. Commun. Netw.* **2017**, 1–13 (2017).
9. Deb, S. & Bhuyan, B. Chaos-based medical image encryption scheme using special nonlinear filtering function based LFSR. *Multimed. Tools Appl.* **80**, 19803–19826 (2021).
10. Guo, L., Du, H. & Huang, D. A quantum image encryption algorithm based on the Feistel structure. *Quantum Inf. Process.* **21**, 20 (2021).
11. Kumar, M., Iqbal, A. & Kumar, P. A new RGB image encryption algorithm based on DNA encoding and elliptic curve Diffie-Hellman cryptography. *Signal Process.* **125**, 187–202 (2016).
12. Abd-El-Hafiz, S. K., Abdelhaleem, S. H. & Radwan, A. G. Novel permutation measures for image encryption algorithms. *Opt. Lasers Eng.* **85**, 72–83 (2016).
13. Radwan, A. G., AbdElHaleem, S. H. & Abd-El-Hafiz, S. K. Symmetric encryption algorithms using chaotic and non-chaotic generators: A review. *J. Adv. Res.* **7**, 193–208 (2016).
14. Almajed, H. & Almogren, A. A secure and efficient ecc-based scheme for edge computing and internet of things. *Sensors (Switzerland)* **20**, 6158 (2020).
15. Almajed, H. N. & Almogren, A. S. SE-Enc: A secure and efficient encoding scheme using elliptic curve cryptography. *IEEE Access* **7**, 175865–175878 (2019).
16. Abbas, A. M., Alharbi, A. A. & Ibrahim, S. A novel parallelizable chaotic image encryption scheme based on elliptic curves. *IEEE Access* **9**, 54978–54991 (2021).
17. Haider, M. I., Ali, A., Shah, D. & Shah, T. Block cipher's nonlinear component design by elliptic curves: An image encryption application. *Multimed. Tools Appl.* **80**, 4693–4718 (2021).
18. Laiphrakpam, D. S. & Khumanthem, M. S. Medical image encryption based on improved ElGamal encryption technique. *Optik (Stuttg).* **147**, 4693–4718 (2017).
19. Banik, A., Shamsi, Z. & Laiphrakpam, D. S. An encryption scheme for securing multiple medical images. *J. Inf. Secur. Appl.* **49**, 102398 (2019).
20. Abdelfatah, R. I. Secure image transmission using chaotic-enhanced elliptic curve cryptography. *IEEE Access* **8**, 3875–3890 (2020).
21. Washington, L. C. *Elliptic Curves: Number Theory and Cryptography*, 2nd ed. (2008).
22. Johnston, D. *Random Number Generators—Principles and Practices: A Guide for Engineers and Programmers* (Walter de Gruyter GmbH, 2018).
23. Payingat, J. & Pattathil, D. P. Pseudorandom bit sequence generator for stream cipher based on elliptic curves. *Math. Probl. Eng.* **2015**, (2015).
24. Reyad, O. & Kotulski, Z. On pseudo-random number generators using elliptic curves and chaotic systems. *Appl. Math. Inf. Sci.* **9**, 31–38 (2015).
25. Toughi, S., Fathi, M. H. & Sekhavat, Y. A. An image encryption scheme based on elliptic curve pseudo random and Advanced Encryption System. *Signal Process.* **141**, 217–227 (2017).
26. Fan, X., Gong, G., Schoenmakers, B., Sica, F. & Sidorenko, A. Secure simultaneous bit extraction from Koblitz curves. *Des. Codes Cryptogr.* **87**, 1–13 (2019).
27. Reyad, O., Hamed, K. & Karar, M. E. Hash-enhanced elliptic curve bit-string generator for medical image encryption. *J. Intell. Fuzzy Syst.* **39**, 7795–7806 (2020).
28. Hayat, U. & Azam, N. A. A novel image encryption scheme based on an elliptic curve. *Signal Process.* **155**, 391–402 (2019).
29. Ullah, I., Azam, N. A. & Hayat, U. Efficient and secure substitution box and random number generators over Mordell elliptic curves. *J. Inf. Secur. Appl.* **56**, 102619 (2021).
30. Shah, D. *et al.* An efficient audio encryption scheme based on finite fields. *IEEE Access* https://doi.org/10.1109/ACCESS.2021.3119515 (2021).
31. Azam, N. A., Ullah, I. & Hayat, U. A fast and secure public-key image encryption scheme based on Mordell elliptic curves. *Opt. Lasers Eng.* **137**, 106371 (2021).

32. Adhikari, S. & Karforma, S. A novel image encryption method for e-governance application using elliptic curve pseudo random number and chaotic random number sequence. *Multimed. Tools Appl.* **81**, 759–784 (2022).
33. Azam, N. A., Hayat, U. & Ullah, I. Efficient construction of a substitution box based on a Mordell elliptic curve over a finite field. *Front. Inf. Technol. Electron. Eng.* **20**, 1378–1389 (2019).
34. Mefenza, T. & Vergnaud, D. Inferring sequences produced by elliptic curve generators using Coppersmith's methods. *Theor. Comput. Sci.* **830–831**, 20–42 (2020).
35. National Institute of Standards and Technology. FIPS PUB 186-4 FEDERAL: Digital Signature Standard (DSS). *Process. Stand. Publ.* (2013).
36. Shannon, C. E. Communication theory of secrecy systems. *Bell Syst. Tech. J.* **28**, 656–715 (1949).
37. Bassham, L. E. *et al.* SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010).
38. Wu, Y., Noonan, J. P. & Agaian, S. NPCR and UACI Randomness Tests for Image Encryption. *cyberjournals.com* (2011).
39. Weber, A. The USC-SIPI image database. *Signal Image Process. Inst. Univ. South. California*. https://sipi.usc.edu/services/database (1997).
40. Elliptic Curve Cryptography Public key cryptography. https://www.ietf.org/rfc/rfc5480.txt.
41. Ur Rehman, A., Liao, X., Kulsoom, A. & Abbas, S. A. Selective encryption for gray images based on chaos and DNA complementary rules. *Multimed. Tools Appl.* **74**, 4655–4677 (2015).

Acknowledgements

This paper is based upon work supported by Science, Technology, and Innovation Funding Authority (STIFA) under Grant number 45631.

Author contributions

Conceptualization: S.H.A., S.K.A., A.G.R. Implementation and initial draft: S.H.A. Reviewing and editing: S.K.A., A.G.R. All authors read and approved the final manuscript.

Funding

Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-17045-x.

Correspondence and requests for materials should be addressed to S.H.A. or S.K.A.-E.