A Novel One-dimensional Chaotic Map, Novel Diffusion and DNA Encoding baed Image Encryption Scheme

[1]Mohit Dua, [2]Rohit Kumar, [3]Shelza Dua, [4]Nidhi Chakravarty
[1,2]Department of Computer Engineering, National Institute of Technology Kurukshetra
[3]Department of Electrical Engineering, National Institute of Technology Kurukshetra
[4]Department of Computer Science and Engineering, TIET, Patiala
[1]er.mohitdua@nitkkr.ac.in, [2]rohitdahiya102852@gmail.com, [3]shelzadua@gmail.com,
[4]nidhichakravarty03@gmail.com

Corresponding Author: Nidhi Chakravarty

Abstract

The advancements in data storage and network technologies give rise to the urgent need for new image encryption techniques that are computationally fast but complex enough to withstand all types of security attacks. This paper introduces a novel one-dimensional chaotic map named Delta Sine-Cosine(DSC) chaotic map that uses trigonometric functions. With that, a new encryption scheme is proposed, which pairs random sequences generated by a proposed DSC map with DNA (Deoxyribonucleic Acid) encoding. The complete encryption process is divided into three steps, diffusion, permutation and confusion. The first step is novel diffusion operation in which the image pixel values are mixed using XOR operation using a pattern inspired by the movement of a chess piece knight. In the second step, permutation is applied using the chaotic sequence generated by the proposed chaotic map. In the last confusion step, the permuted image is converted into a DNA sequence which is DNA XORed followed by DNA addition with the two different DNA encoded chaotic sequences that are generated using the proposed DSC map. This DNA encoded image is converted into noise like binary image or cipher image. The performance of this setup has been measured by using various performance parameters like Number of changing pixel rate (NPCR), Unified averaged changing intensity (UACI), Correlation Coefficient(CC), histogram analysis, Peek Signal-to-Noise Ratio (PSNR) and Entropy, which prove that the proposed encryption framework is capable of handling different types of attacks.

Keywords: DSC map, DNA cryptography, UACI, CC, Entropy.

1. Introduction

The advancements in network technologies and computer hardware, especially storage devices, in the last two decades have made human civilization dependent on digital data like never before.

The generation, creation and transmission of data is at an all-time high and is going up exponentially. All type of sensitive data needs to be secured. This increase in data dependence gives rise to challenges for the information security of the data. One such type of data that stands out is image data which needs security measures urgently that are efficient and fast. Image data may contain sensitive data concerning a normal citizen, a big firm or a nation. For example, images taken by remote sensing of national borders concerns national security [1], biometric images of citizens of a country or medical images of patients transferred to other doctors is a privacy concern [2].

Cryptography experts have developed many types of image encryption techniques to protect data such as stenography [3, 4], watermarking [5], mathematical encryption, etc. In this paper, the focus is on mathematical encryption method. There are various mathematical encryption methods ranging from traditional methods like Data Encryption Standard (DES), Advanced Encryption Standard (AES) and Rivest–Shamir–Adleman (RSA) [6, 7, 8] to modern methods that are based on spatial, frequency or hybrid domains like chaotic maps, DNA-encoding, optical systems, neural networks or cellular automata [9, 10]. In recent years, chaotic map-based image encryption techniques have shown promising results. Chaotic theory features like unpredictability, non-convergence and high sensitivity to initial conditions have attracted attention of researchers who work in the area of building cryptosystems [10].

A chaotic map is a mathematical model that describes behavior of a dynamic system. Chaos theory was summarized by Edward Lorenz as — Chaos: “When the present determines the future but the approximate present does not approximately determine the future.” A small change in the initial value can lead to a drastic change in the end result, but the exact initial value will result in exactly the same outcome. This means a chaotic system is deterministic, but is unpredictable if initial conditions are not known. This attracts a lot of researchers' attention towards chaotic maps. Its non-deterministic behavior makes it resistant towards statistical and brute-force attacks, and that makes it a powerful tool for encryption techniques.

Chaotic maps, on the basis of input, are termed as discrete or continuous, and on the basis of dimension are called as one-dimensional or multi-dimensional. Dimensions refer to the number of control parameters. A one-dimensional chaotic map is simple to implement and fast in performing encryption and decryption. On the contrary, multi-dimensional maps are complex to implement

and take more time to perform encryption and decryption. However, multi-dimensional chaotic maps are more secure than on-dimensional chaotic maps.

Trigonometric function based chaotic maps are a preferred over other chaotic maps for image encryption methods, due to three main reasons. First, the design and implementation of trigonometric maps are straightforward, allowing other researchers to integrate them into their encryption scheme. Second, trigonometric values can be computed rapidly thus such maps enjoy low encryption and decryption time, which is suitable for real-world applications. Third, these maps generally have a larger range in comparison to other maps. Hence, this paper also proposes a novel one dimensional chaotic map that uses trigonometric functions.

Exploiting biological medium-based computing and storage devices is still a developing technology. However, the possibilities are endless. Research published in 2017 claims to found a scheme that allows them to store 255 petabytes of data in 1 gram of Deoxyribonucleic Acid (DNA) [11]. Another research claims to achieve a storage density of 5.5 petabits/mm3 using DNA and mentions the theoretical possibility of storing 455 exabytes (1 exabyte = 1+e18 bytes) of data in 1 gram of single-stranded DNA (ssDNA) material [12]. More importantly, DNA shows remarkable stability, ease of replication, backup capabilities (copying large amounts of data) and hybridization (searching). These potential benefits make taking a step toward DNA encoding worthwhile.

Data is stored in DNA in the form of long chains of four bases — adenine[A], cytosine[C], guanine[G], and thymine[T]. These are the fundamental blocks of data in DNA-based storage schemes. Bases A and T work as complements of each other and bases C and G work as complements of each other. Each base is associated with two bits of information. That gives 24 rules of association but only eight of the rules follow the complementary conditions [13]. Any of the eight rules can be used to encode data. There are DNA operations that are equivalent to basic logical operations of binary data — addition, subtraction and XOR. These can be applied to the DNA sequences. One noteworthy characteristic of these operations is that if applied to sequences of the same length, these will produce a DNA sequence of exactly the same length. Hence, keeping these features in mind the work proposed in this paper performs DNA encoding of chaotic sequences and input image in conjunction with DNA operations to propose a novel image encryption scheme that uses novel proposed one-dimensional Delta Sine-Cosine chaotic map.

The section wise details of the remaining paper is described as: Section 1.1 discusses some of the recent works related to the proposed work, motivation and pointwise contribution of the

proposed work. Section 2 describes background details of the the proposed encryption scheme. Section 3 presents the proposed scheme architecture, and then explains the encryption scheme as well as decryption scheme in detail. Section 4 describes the experimental setup and result analysis. Finally, Section 5 ends the paper with the conclusion and future work.

1.1 Related Works

In last two decades, many research work has been published, which uses chaotic maps for image encryptions. In 2020, Pak et al. [14] used a one-dimensional chaotic model to introduce improved Logistic and Sine chaotic maps. These chaotic maps were used with steganography to embed information within images. The chaotic sequence generated determines the specific locations within the image where the data is hidden. Farah et al. [15] proposed an improved chaotic map by combining Logistic map, Sine map and Tent map into one by simply joining output of one to input of other. Authors used it in combination with an optimized substitution box that is generated by a different algorithm called chaotic Jaya optimization algorithm. The encryptions scheme showed promising results.

In the year 2021, Talhaoui and Wang [16] proposed a new one-dimensional fractional chaotic maps in order to overcome the small chaotic space faced by other 1D maps, and speed issues faced by other multi-dimensional maps. In the work, a new encryption scheme was also proposed that performs substitution and permutation simultaneously, which increases the speed. The proposed scheme also used multiple rounds to increase the quality of encryption, and proved to do the job it is made for. Khairullah et al. [17] proposed 1D improved Logistic map and 1D improved quadratic map. The scheme used the keys that are generated using the sum of all pixels, and sum of all pixels in rows as well as columns with MD5 algorithm. Then, a sequence is generated and index is associated with each number. When the sequence is sorted, indices gets mixed, which is exploited in permutation phase. Finally, encryptoin ends with the diffusion phase. Author also proposed a key expansion method to reduce the number of chaotic map iterations required. Cun et al. [18] created a new trigonometric chaotic map that use simple mathematical and sine operation. The encryption scheme took a new approach that uses Local Graph Structure algorithm proposed by Abusham E A. The scheme selects some regions of the image and apply encryption on that selected region and then merge it with the unselected region. The image is encrypted using multiple chaotic sequences modified in pretty complex manner. Later, DNA encoding is applied using multiple rules at the same time. It proved to be slightly faster than other dynamic DNA encoding schemes.

In the year 2022, Lu et al. [19] improved the model of sine map by introducing a cosine term in the product. This resulted in a model with improved chaotic behavior. The work also introduced a new encryption scheme that used SHA-256 which is then divided in to 64 hexadecimal characters. Two sequences were used for column and row permutation, and third was used for diffusion using bitwise XOR operation.

In the year 2023, Dhingra and Dua proposed a new 1D chaotic trigonometric by coupling Sine and Tangent functions, and used it for video encryption [20]. The video is first broken into frames, and for each frame an S-box is generated using their chaotic map. The authors proved that the chaotic map proposed has more complex behavior and significantly superior chaotic performance than other existing chaotic maps. Liang and Zhu [21] created a new one dimensional chaotic map which they named as Sine-Cosine chaotic map (SCCM). The mathematical model used simple mathematical operations and simple trigonometric functions that make it computationally light weight. A new encryptions scheme is also proposed which creates a 512-bit hash using SHA-512. The input requires five external keys, which are mixed with this hash and fed into the chaotic map. The sequence generated are used for diffusion, permutation and DNA encoding.

In the year 2024, Feng et al. [22] introduced a new chaotic system by combining already existing chaotic systems. The encryption scheme used SHA-256 to generate a hash which in turn used to generate initial values and parameters for the model. The model is then exploited to be used in confusion, and finally the algorithm ends with diffusion which used DNA encoding. It results in noise like image which can be restored loselessly. The analysis also shows that this scheme in extremely sensitive to input image.

The exploration of chaotic maps for image encryption presents a promising alternative. Chaotic systems are inherently sensitive to initial conditions and exhibit behavior that is both random and deterministic, making them ideal for generating random streams that are difficult to predict or replicate if initial conditions are unknown. This aligns with the need for encryption methods that should not only be secure but also efficient and adaptable to the dynamic nature of image data. In this context, our paper introduces a novel one-dimensional chaotic map, the Delta Sine-Cosine(DSC) map, which demonstrates superior chaotic behavior and a broader range than existing maps. This map is combined with DNA encoding to create an encryption scheme. Our approach aims to improve fast yet complex encryption techniques that can withstand various security attacks, contributing to the field of secure image encryption. The complete encryption

process is divided into three steps, diffusion, permutation and confusion. Firstly, the image pixel values are mixed using XOR operation using a pattern inspired by the movement of a chess piece knight to perform novel diffusion operation. Secondly, permutation is applied using the chaotic sequence generated by the proposed chaotic map. Finally, the permuted image is converted into a DNA sequence which is DNA XORed followed by DNA addition with the two different DNA encoded chaotic sequences that are generated using the proposed DSC map. This DNA encoded image is converted into noise like binary image or cipher image. The point-wise contribution of the proposed paper is as follows:

1. The work proposes a new one-dimensional chaotic map i.e. Delta Sine-Cosine(DSC) map. The parameters Bifurcation diagram and Lyapunov Exponent are checked to prove its effectiveness in comparison to other standard maps.
2. A new image encryption scheme is proposed, which uses a novel diffusion method inspired by the movement of a chess piece knight, followed by permutation, which is performed using the proposed chaotic map sequence.
3. The final step performs confusion using DNA XOR and DNA addition operations on DNA encoded permuted image and DNA encoded two different chaotic sequences.
4. The proposed encryption scheme has been applied to three sample images, and is analyzed using various parameters which are NPCR, UACI, histogram analysis, PSNR, entropy, and Correlation Coefficient(CC) analysis. A comparative analysis has also been given to prove the proposed scheme effectiveness over the existing image encryption schemes.

2. Fundamentals of Proposed Chaotic Map

This section discusses the fundamentals of three basic chaotic maps and proposed Delta Sine-Cosine(DSC) map

2.1 Logistic Map

The Logistic Map (LM) is one of the most popularly known one dimensional chaotic map. The reason behind its popularity is the simplicity of its mathematical equation. Chaotic behavior generated by such a simple mathematical model makes it a standard map for comparison and also the reason to be researched extensively. It is mathematically defined as:

$$X_{n+1} = rX_n(1 - X_n) \tag{1}$$

where, $X_n \in (0, 1)$ is the value at the nth step and $r \in (0, 4]$ is the control parameter. After $r = 4$ the bifurcation collapses to a line. The Bifurcation diagram, Lyapunov Exponent diagram and

Shannon entropy diagram for LM are presented in Figure 1(a), Figure 2(a) and Figure 3(a), respectively, from which it can clearly be observed that that LM suffers from issues like lower chaotic range, blank and stable windows.

2.2 Sine Map

It is a one dimensional trigonometric chaotic map. The mathematical equation that describes the Sine map is given in as:

$$X_{n+1} = r sin(\pi X_n) \tag{2}$$

where, the range of r is [0, ∞). For r in the range [0,1), the range of X_n is (0, r] and shows bifurcation like a LM. The Bifurcation diagram, Lyapunov Exponent diagram and Shannon Entropy diagram for Sine map are shown in Figure 1(b), Figure 2(b) and Figure 3(a), respectively. The Lyapunov Exponent goes above 0 somewhere near 0.8. For r > 1, X_n has the range [-r,+ r]. The bifurcation goes in +ve and -ve both, showing good chaotic behaviour, but it periodically collapses to points near + r and - r. It can also be observed that the Lyapunov Exponent goes +ve and suddenly dips -ve periodically, making it more complex to implement. It has larger chaotic range than LM, making it more suitable for applications.

2.3 Tent Map

The mathematical model of Tent map is presented as:

$$X_{n+1} = \begin{cases} r\, X_n : for\ 1/2 > X_n \\ r(1 - X_n) : for 1/2 \leq X_n \end{cases} \tag{3}$$

where, r ranges in (0,2]. For the control parameter in the range (0, 1] the output of the map converges to 0 eventually, which makes this range not suitable for any application. After the control parameter crosses 1, the map starts showing chaotic behavior. The map is called a Tent map because the graph of the Equation (3) forms a symmetric inverted V that looks like a tent. Bifurcation diagram and Lyapunov Exponent diagram and Shannon entropy diagram of the Tent map are given in Figure 1(c), Figure 2(c) and Figure 3(c), respectively. Bifurcation of Tent map is more uniform, and also it does not have any periodic intervals or order. Lyapunov Exponent is positive and doesn't goes back to 0. But is has a shorter range than sine map.

2.4 Delta Sine-Cosine(DSC) Map

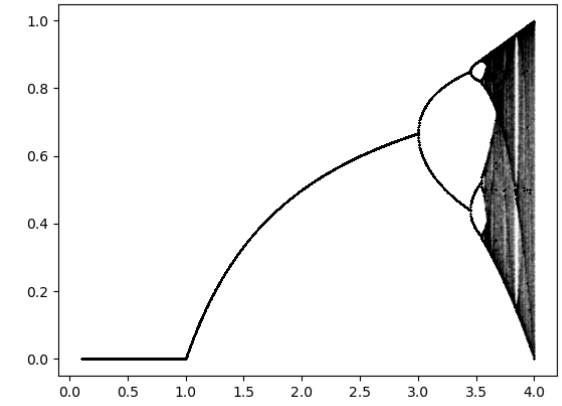
The mathematical representation of the proposed novel one dimensional chaotic map i.e., Delta Sine-Cosine (DSC) map is given in Equation (4) and Equation (5).

$$\theta = (-rX_n) + X_n^3 - \left(r\ tan(X_n)\right) \tag{4}$$

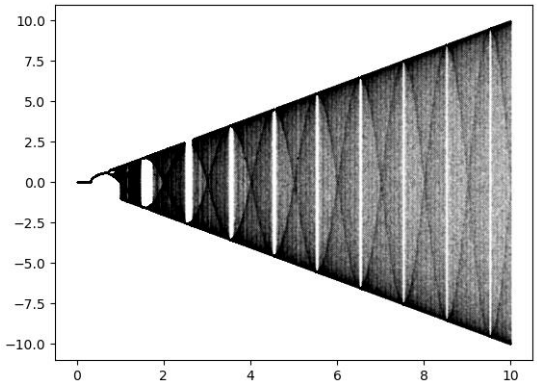
$$X_{n+1} = \left||sin(\theta)| - |cos(\theta)|\right| \quad (5)$$

where, r theoretical range of (0, ∞). One thing to mention here is that as tan () has the parameter that always lies in the range 0 to 1, it will never shoot to infinity, making computation possible. Bifurcation diagram and Lyapunov Exponent diagram and Shannon entropy diagram, are shown in Figure 1(d), Figure 2(d) and Figure 3(d), respectively.

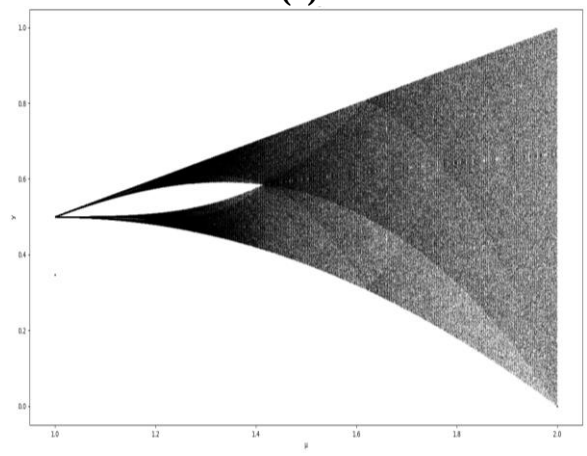
The Bifurcation diagram and Lyapunov Exponent diagram for the control parameter with range [2, 10000] with 10 sample points for each of the 10000 different r values taken, are given in Figure 1(e) and Figure 2(e), respectively. The proposed DSC map's Bifurcation diagram is denser than the other chaotic maps. The range is infinite just like Sine map, and Shannon entropy is the factor that shows its superiority than other maps. It is above 8 for r >= 2. It is recommended to have Shannon entropy above 8 making it more effective to create random sequences.



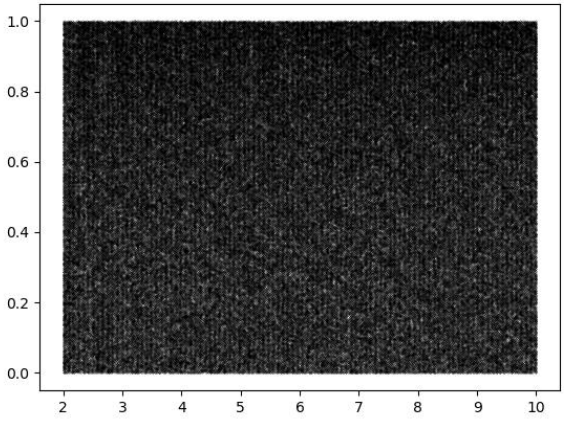
(a)



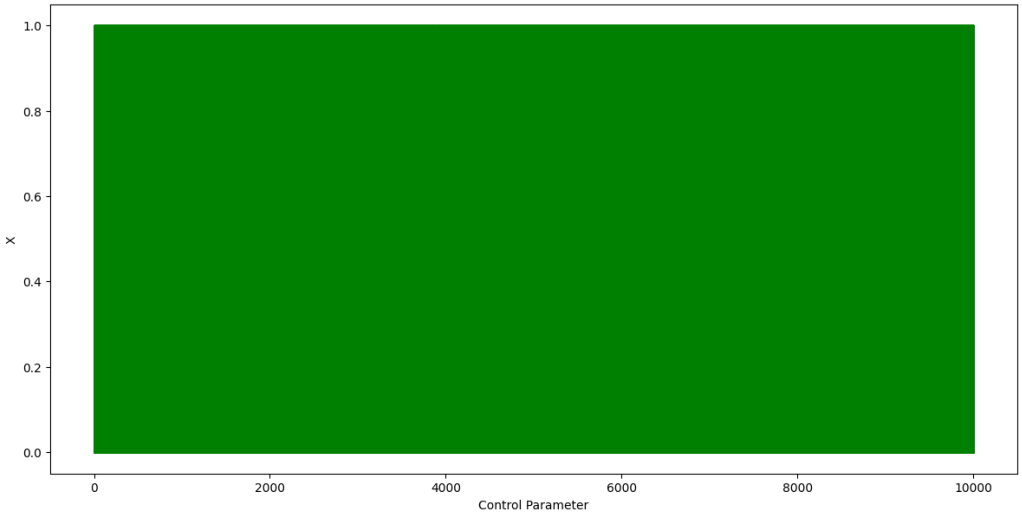
(b)



(c)

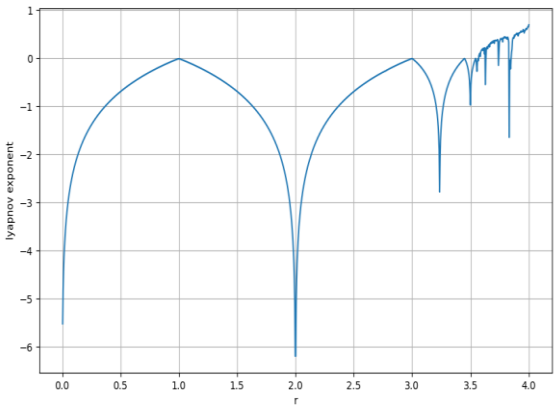


(d)

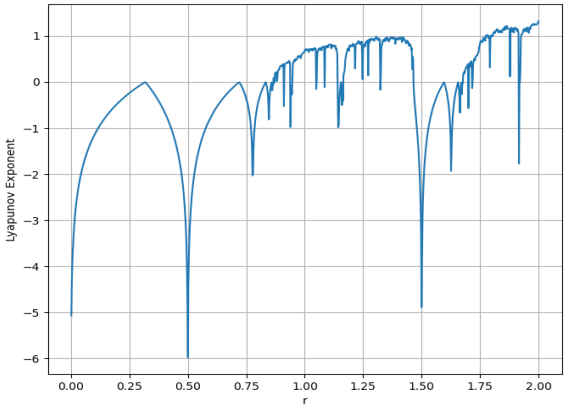


(e)

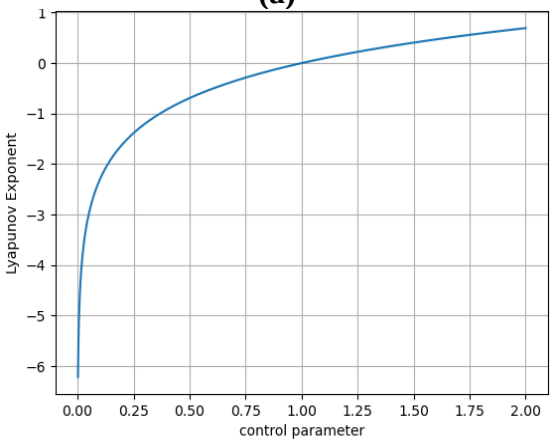
Figure 1: Bifurcation Diagram **(a)** LM **(b)** Sine Map **(c)** Tent Map **(d)** DSC with Small Range **(e)** DSC with Large Range



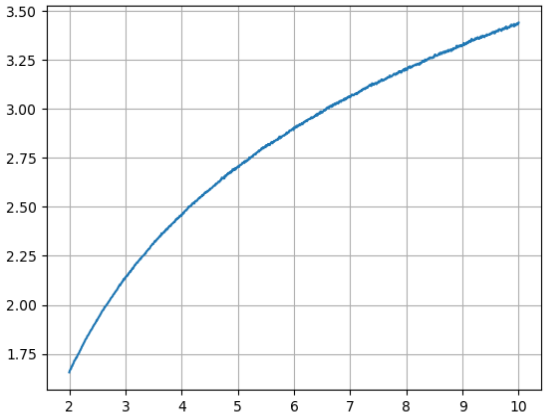
(a)



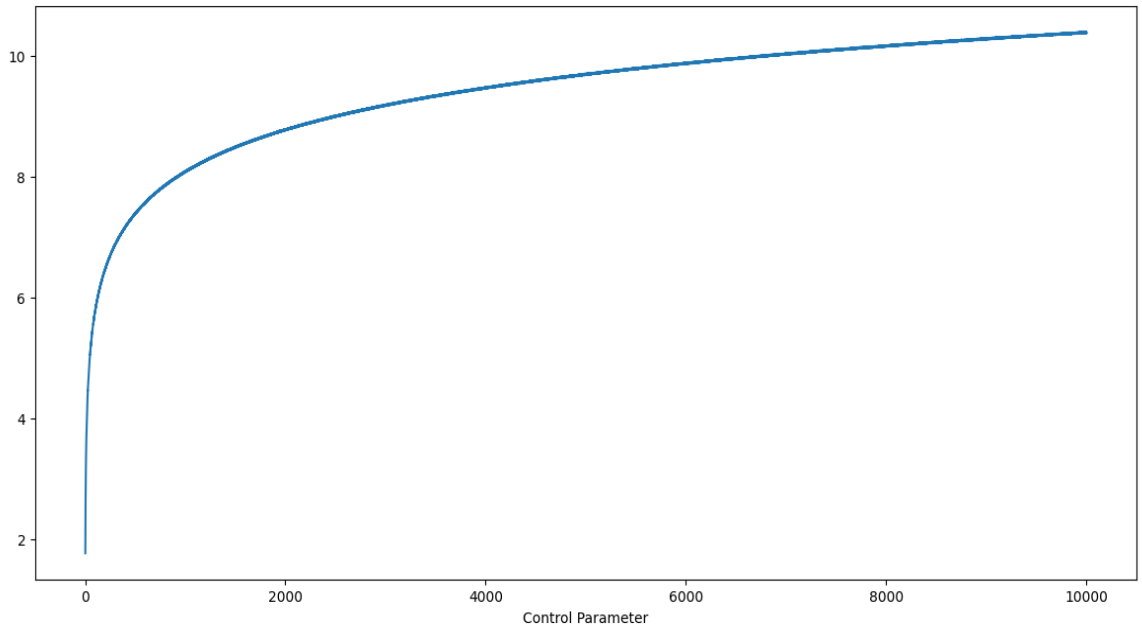
(b)



(c)

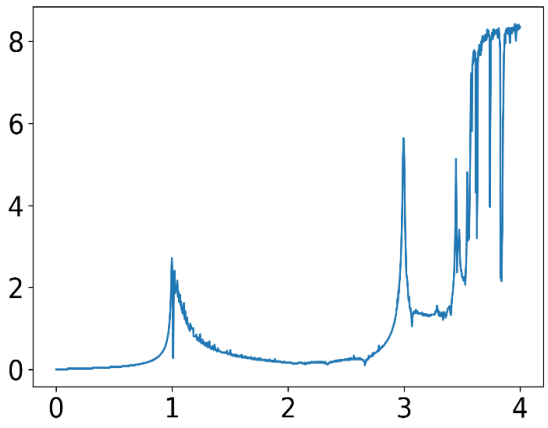


(d)

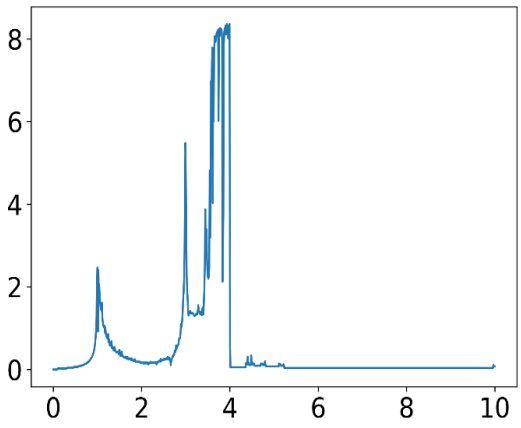


(e)

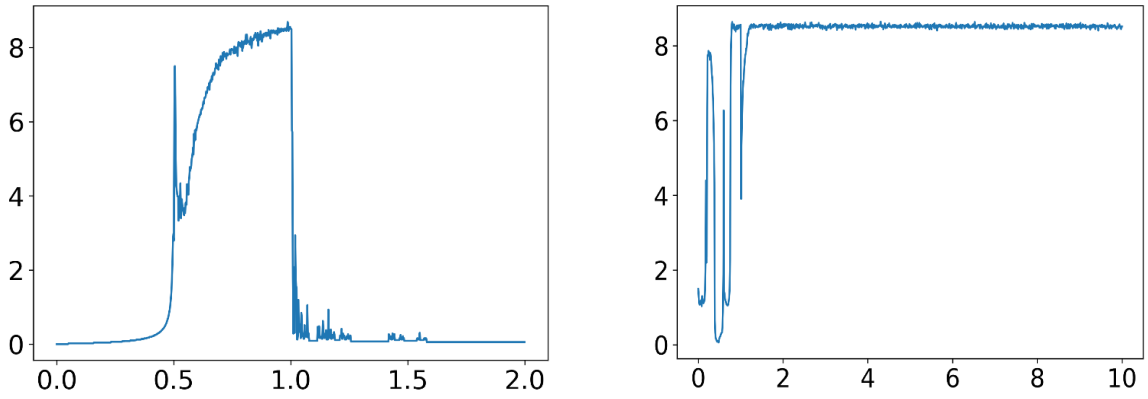
Figure 2: Lyapunov Exponent **(a)** LM **(b)** Sine Map **(c)** Tent Map **(d)** DSC with Small Range **(e)** DSC with Large Range



(a)



(b)



(c) **(d)**

Figure 3: Shannon Entropy **(a)** LM **(b)** Sine Map **(c)** Tent Map **(d)** DSC Map

3. Proposed Approach

The approach used in this paper mainly has three steps to encrypt an image. The first step is diffusion. Diffusion refers to the spread of influence of a single pixel change across the entire encrypted image. In the proposed approach a novel diffusion method has been proposed, where pixel values are mixed with other pixels using the bitwise XOR operation in a pattern inspired by a chess knight move. In vague terms, first, we move along the diagonal from top left to bottom right, and second, we move along a diagonal from the bottom right to the top left, i.e., during this movement pixel values are mixed. The second step is permutation which means to rearrange or shuffle the pixels. Permutation here is based on the order of random numbers generated by the chaotic map according to the given parameters. Finally, confusion is performed which aims to make the relation between the original and cipher image extremely minimal. For this DNA encoding and DNA operations are employed. First, the permuted image is converted into a DNA sequence. Along with that two different DNA sequences are generated using the proposed chaotic map. Then, two DNA operations XOR and addition, one after the other, are applied. The complete encryption process uses three keys and two control parameters. Two keys and two control parameters are used to generate DNA sequences. The third key, and any one of the two control parameters are used for generating the random numbers for permutation. The decryption algorithm is just the reverse order of the encryption algorithm. First, the two DNA sequences are generated. The first DNA sequence is subtracted from the image's DNA sequence and the second one is XORed. Figure 4 shows the architecture of the proposed image encryption scheme. The remaining part of this section discusses these encryption and decryption steps in detail.

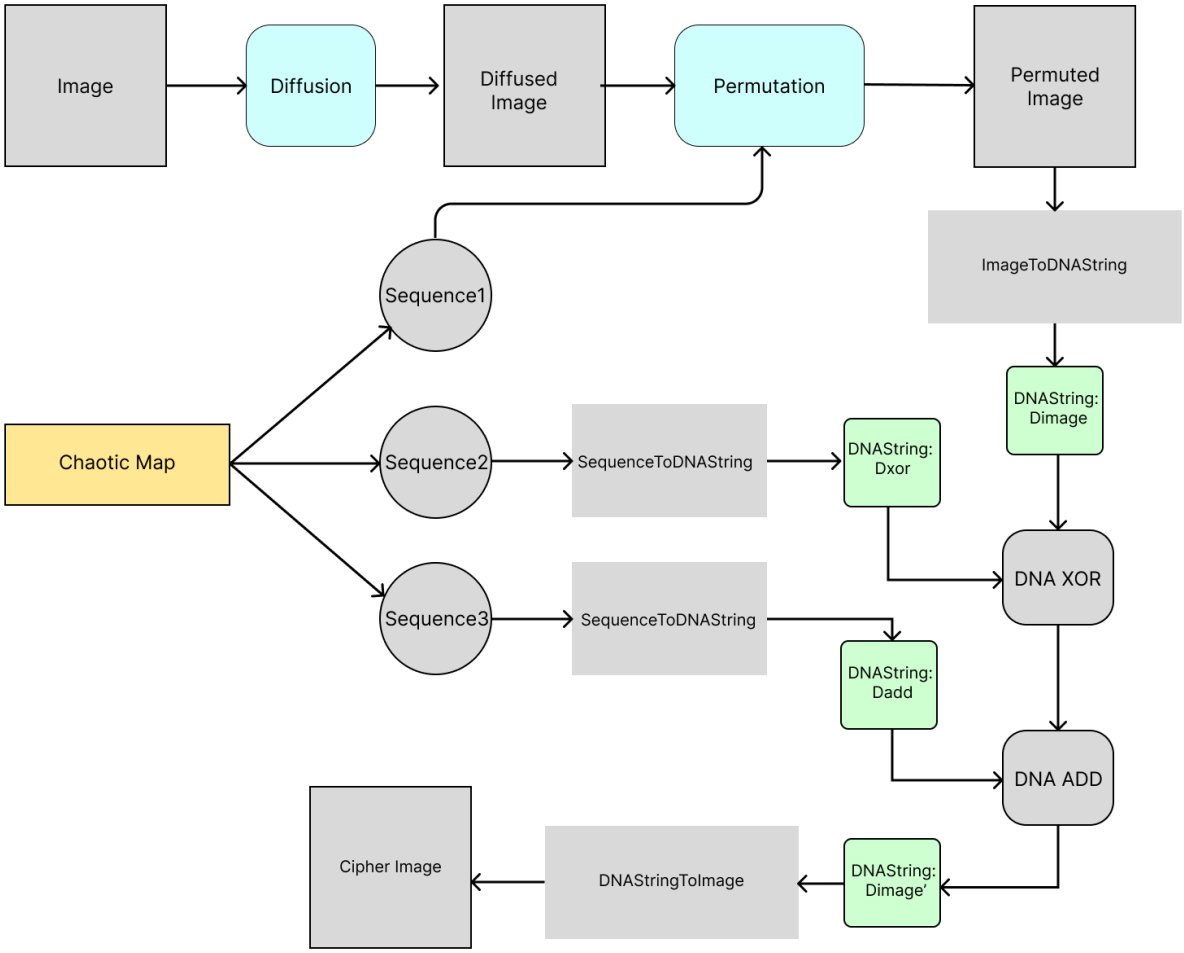


Fig 4: Encryption procedure

3.1 Diffusion

Diffusion spreads the influence of a single pixel change across the entire encrypted image. In the proposed approach, a novel diffusion method has been proposed, where pixel values are mixed with other pixels using the bitwise XOR operation in a pattern inspired by a chess knight move. The step-by-step procedure of applying diffusion method the proposed scheme is described below:

Step 1: Read the given input image as a matrix that has R no. of rows and C no. of columns, where $P(x, y)$ represents the RGB or grayscale value depending on the image type, of the pixel in row number x and column number y and N be the total number of pixels in the image.

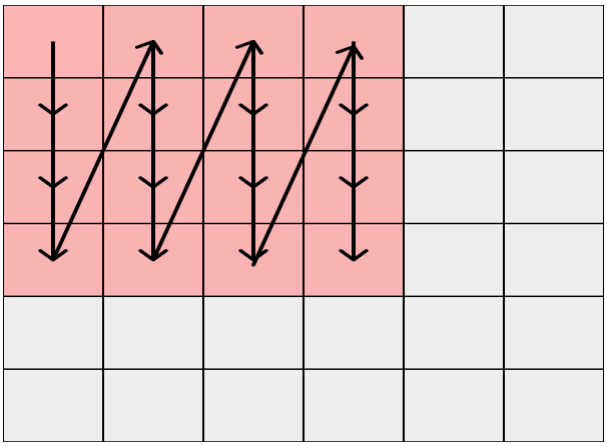
Step 2: Starting from the top left, iterate over all the pixels in a column-wise manner except the pixels in the last two columns (rightmost) as well as in the last two rows (bottommost), an example is shown in Figure 5(a). Consider the current pixel is A having coordinate (x, y). Then define pixel

B which has coordinates$(x+2, y+1)$ and pixel C which has coordinates $(x+1, y+2)$ as explained in Figure 5(b). Apply Equation (6) here. Set the value of pixel B with bitwise XOR of pixels A and B. Similarly, update the value of pixel C with bitwise XOR of pixels A and C.

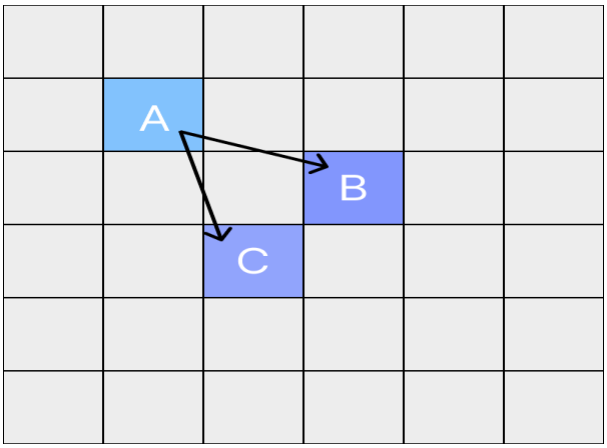
$$\begin{cases} A \rightarrow P(x, y) \\ B \rightarrow P(x+2, y+1) \\ C \rightarrow P(x+1, y+2) \\ B = A \; xor \; B \\ C = A \; xor \; C \end{cases} \tag{6}$$

Step 3: Perform a similar operation as of, but this time start from bottom right and iterate over all pixels in column-wise manner leaving pixels in top 2 rows and rightmost 2 columns, explained with an example in Figure 5 (c). Consider the current pixel is A having coordinate (x, y). Then, define pixel B which has coordinates $(x-1, y-2)$ and pixel C which has coordinates $(x-2, y-1)$ as explained in Figure 5(d). Apply Equation (7) here. Set the value of pixel B with bitwise XOR of pixels A and B. Similarly, set the value of pixel C with bitwise XOR of pixels A and C. The pseudo-code for steps 2 and 3 is presented in Function 1.

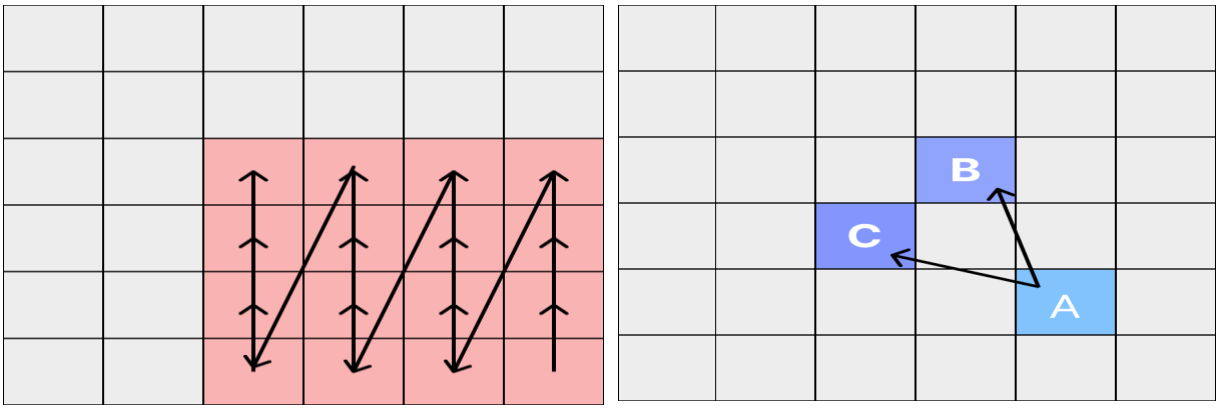
$$\begin{cases} A \rightarrow P(x, y) \\ B \rightarrow P(x-1, y-2) \\ C \rightarrow P(x-2, y-1) \\ B = A \; xor \; B \\ C = A \; xor \; C \end{cases} \tag{7}$$



(a)



(b)



(c) **(d)**

Figure 5: (a) Diffusion path 1 for a 6x6 image **(b)** XOR pattern 1 **(c)** Diffusion path 2 for a 6x6 image **(d)** XOR pattern 2

Function 1: Diffusion
Input: original_image
1. width = originalImage.width *2. height = originalImage.height* *3. diffusedImage = originalImage.copy()* *4. for c ← 1 to width -2* *5. for r ← 1 to height -2* *6. diffusedImage[r+2, c+1] = diffusedImage[r+2, c+1] xor diffusedImage[r, c]* *7. diffusedImage[r+1, c+2] = diffusedImage[r+1, c+2] xor diffusedImage[r, c]* *8. for c ← width to 3* *9. for r ← height to 3* *10. diffusedImage[r-2, c-1] = diffusedImage[r-2, c-1] xor diffusedImage[r, c]* *11. diffusedImage[r-1 c-2] = diffusedImage[r-1, c-2] xor diffusedImage[r, c]*
Output: diffused_image

3.2 Permutation

Permutation refers to rearranging the pixels of the image. Imagine cutting the image into tiny squares, shuffling them around, and then sticking them back together in a new order. The detail steps for applying permutation are given below:

Step 1: Associate numbers 1 to N to each pixel using Equation (8).

$$n = x * C + y \tag{8}$$

where, x and y are the row and column of the pixel. Also, assume the Equation (9).

$$K(n) = P(x, y) \tag{9}$$

Step 2: Associate numbers 1 to N with each number in the generated sequence according to their position. Let $A(x)$ represents the number associated with the numbers $x \in S$. Sort the sequence in ascending order without disturbing the association as in Figure 6.

Step 3: Generate a new black image of the same shape.

Step 4: From the original image put pixels in the new image using Equation (10). To apply permutation step, follow Function 2 and Function 3. The result of permutation is illustrated through an example given in Figure 7.

$$K_{new}(z) = K\big(A\big[S[z]\big]\big): \forall z \epsilon \{1, 2, 3 \ldots, N\} \quad (10)$$

where, $K(n)$ is defined in Equation (9).

X	.58	.83	.42	.25	.77	.96	.76	.10	.17	.54	.73	.05	.39	.64	.14	.93
$A(x)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sequence S																

(a)

X	.05	.10	.14	.17	.25	.39	.42	.54	.58	.64	.73	.76	.77	.83	.93	.96
$A(x)$	12	8	15	9	4	13	3	10	1	14	11	7	5	2	16	6
Sorted Sequence S																

(b)

Figure 6: (a) Example of generated sequence and number association **(b)** Association after sorting the sequence

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
original			

12	8	15	9
4	13	3	10
1	14	11	7
5	2	16	6
permuted			

Figure 7: Example of image after permutation

Function 2: chaotic Map.generate Numbers

Input: contorl Parameter, initial Value, length, transient Iterations = 0 (default Value)

```
1. numbersGenerated = emptyArray(length)
2. for i ← 1 to transientIterations:
3.         initalValue = chaoticMap.nextValue(contorlParameter, initialValue)
4. numbersGenerated[0] = intialValue
5. for i ← 2 to length:
6.         numbersGenerated[i] = chaoticMap.nextValue(contorlParameter,
   numbersGenerated[i-1])
```

Output: numbers Generated

Function 3: permutation
Input: image, key, control Parameter
1. W = image.width() *2. H = image.height* *3. randomNumbers = chaoticMap.generateNumbers(controlParameter, key, W * H)* *4. hashMap = null* *5. for i ← 1 to W * H:* *6. hashMap[i] = randomNumbers[i]* *7. seq = sort(randomNumbers)* *8. permutedImage = newBlackImage(image.size())* *9. for i ← 1 to W * H:* *10. permutedImage[i % W][i / W] = image[seq[i] % W][seq[i] / W]*
Output: permuted Image

3.3 Confusion and Cipher Image Creation

Confusion step aims to make the relationship between the original image (plaintext) and the encrypted image (ciphertext) highly complex using DNA encoding [23], [24]. The detail steps to apply confusion are given below:

Step 1: Generate the DNA string of this new image call it 'Dimage'. And generate two DNA strings and name them 'Dxor', and 'Dadd' using key1, control parameter1 and key2, control parameter2 respectively. To generate 'Dimage' consider it as a binary string. And replace every binary pair, starting from the beginning of the string, with the corresponding DNA character they represent.

Step 2: To generate the random DNA strings 'Dxor' and 'Dadd' first, K random numbers are generated using the chaotic map. The numbers generated are in the range 0 to 1. Multiply them with 256 and take bitwise AND with $(00000011)_2$ to get a number in {0,1,2,3}. Each number works as an index which is then associated with a DNA character according to the rule used. Finally, join all the characters.

Table 1: DNA addition and subtraction operations

ADD	A	G	C	T	SUB	A	G	C	T
A	A	G	C	T	A	A	T	C	G
G	G	C	T	A	G	G	A	T	C
C	C	T	A	G	C	C	G	A	T
T	T	A	G	C	T	T	C	G	A

Step 3: Apply DNA xor operation, given in Table 2, between 'Dimage' and 'Dxor' and store the result in 'Dimage' itself. Then apply DNA addition operation, given in Table 1, between 'Dimage' and 'Dadd' and store the result in 'Dimage' again. The resulting DNA sequence will be the

encrypted DNA sequence. The steps 1,2 and 3 can be applied using Function 4, Function 5 and Function 6.

Table 2: DNA xor and complement operation

XOR	A	G	C	T	COMPLEMENT	
A	A	G	C	T	A	T
G	G	A	T	C	G	C
C	C	T	A	G	C	G
T	T	C	G	A	T	A

Step 4: Convert the encrypted DNA sequence back to digital cipher image by replacing each DNA character with the corresponding binary number.

Function 4: generateDnaString

Input: arrayOfNumbers

```
# this function is specifically made for the array of numbers generate by out chaotic map
1. dnaString = ""
2. size = len(arrayOfNumbers)
3. rule = ["A", "G", "C", "T"]    # replace with the rule you are using
4. for I ← 1 to size:
5.         dnaString = dnaString + rule[(sequence[i] * 256) bitxor 3]
```

Output: dnaString

Function 5: generateDnaStringFormImage

Input: image

```
1.  W = image.width()
2.  H = image.height()
3.  dnaString = ""
4.  rule = ["A", "G", "C", "T"]    # replace with the rule you are using
5.  temp1 = [192, 48, 12, 3]
6.  temp2 = [6, 4, 2, 8]
7.  for w ← 1 to W:
8.          for h ← 1 to H:
9.                  for k ← 1 to 3:    # iterate colors r, g, b
10.                         for t ← 1 to 4:
11.                                 index = (image[w][h][k] bitAnd temp[t]) bitRightShift
    temp2[t]
12.                                 dnaString = dnaString + rule[index]
```

Output: dnaString

Function 6: dnaEncoding

Input: image, controlParameter1, key1, controlParameter2, key2

```
1.  W = image.width()
```

2. *H = image.height()* 3. *seq1 = chaoticMap.generateNumbers(controlParameter1, key1, W*H)* 4. *seq2 = chaoticMap. generateNumbers(contorlParameter2, key2, W*H)* 5. *Dxor = generateDnaString(seq1)* 6. *Dadd = generateDnaString(seq2)* 7. *Dimage = generateDnaStringFromImage(image)* 8. *Dimage = dnaXorOperation(Dimage, Dxor)* 9. *Dimage = dnaAdditionOperation(Dimage. Dadd)*
Output: Dimage

3.4 Decryption Scheme

The decryption process is reverse of the encryption process. The stepwise description is explained below.

Step-1: Convert the digital cipher image to DNA string.

Step-2: Call the DNA strings of the cipher image as 'Dimage'. And generate the DNA strings from the chaotic map as mentioned in step 8 of encryption scheme. Here, the DNA string 'Dadd' is called 'Dsub'.

Step-3: First apply DNA subtract operation, given in Table 1, between 'Dsub' and 'Dimage' and store the result in 'Dimage'. Then apply DNA XOR operation, given in Table 2, between 'Dimage' and 'Dxor' and store the result in 'Dimage'.

Step-4: Now convert the DNA string 'Dimage' into a digital image in the matrix form of the same width and height as the original image by using the same DNA rule which was used at the time of encryption. Call this as intermediate image. Function 7 is used to apply steps 2, 3.

Function 7: reverseDnaEncoding
Input: cipherDnaImage, controlParameter1, key1, controlParameter2, key2
1. *W = cipherDnaImage.width()* 2. *H = cipherDnaImage.height()* 3. *seq1 = chaoticMap.generateNumbers(controlParameter1, key1, W*H)* 4. *seq2 = chaoticMap. generateNumbers(contorlParameter2, key2, W*H)* 5. *Dxor = generateDnaString(seq1)* 6. *Dsub = generateDnaString(seq2)* 7. *resultImage = dnaSubtractionOperation(resultImage. Dsub)* 8. *resultImage = dnaXorOperation(Dimage, Dxor)* 9. *resultImage = generateImageFromDnaString(cipherDnaImage)*
Ouput: resultImage

Function 8: reversePermutation
Input: permutedImage, key, controlParameter
1. *W = image.width()* 2. *H = image.height*

```
3.  randomNumbers = chaoticMap.generateNumbers(controlParameter, key, W * H)
4.  hashMap = null
5.  for i ← 1 to W * H:
6.          hashMap[i] = randomNumbers[i]
7.  seq = sort(randomNumbers)
8.  originalImage = newBlackImage(image.size())
9.  for i ← 1 to W * H:
10.         originalImage[seq[i] % W][seq[i] / W] = permutedImage[I % width][i / W]
```

Output: originalImage

Function 9: reverseDiffusion

Input: diffusedImage, key, controlParameter

```
1.  width = diffusedImage.width()
2.  height = diffusedImage.height()
3.  originalImage = diffusedImage.copy()
4.  for w ← 3 to width:
5.          for h ← 3 to height:
6.                  originalImage[w-2][h-1] = originalImage[w-2][h-1] bitxor originalImage[w][h]
7.                  originalImage[w-1][h-2] = originalImage[w-1][h-2] bitxor originalImage[w][h]
8.  for w ← (width-2) to 1:
9.          for h ← (height-2) to 1:
10.                 originalImage[w+1][h+2] = originalImage[w+1][h+2] bitxor originalImage[w][h]
11.                 originalImage[w+1][h+2] = originalImage[w+1][h+2] bitxor originalImage[w][h]
```

Output: originalImage

Step-5: Generate N random numbers using the chaotic map using the same key and control parameter as used in the encryption algorithm which should be key 1 and control parameter 1. Assume this sequence is called S.

Step-6: Create a new black image of the same size as the intermediate image.

Step-7: Associate a number to each pixel to both images using the mapping defined in Equation (8).

Step-8: Associate numbers 1 to N with each number of the sequence S according to their position in the sequence as shown in Figure 8(a). Let this mapping be represented by $A(x)$ for all $x \in S$. Now, sort the generated sequence S without disturbing the association.

Step-9: Iterate through each pixel of the intermediate image and put them in the new image by using the mapping defined in Equation (11).

$$K_{new}\left(A\left[S[z]\right]\right) = K_{intermediate}(z): \forall z \epsilon \{1, 2, 3 \ldots, N\} \quad (11)$$

Function 8 contains the sample code for steps 4,5,6,7 and 8

Setp-10: Now only diffusion remains to be reversed. To do so, iterate over the pixels starting from the pixel at (3,3) going to (width, height), move in a column-wise fashion towards the bottom, leaving pixels in topmost two rows as shown in Figure 8(a). According to the current pixel, A having coordinates say x and y, select pixels B and C having coordinates $(x-2, y-1)$ and $(x-1, y-2)$, respectively. Set the values of pixel B with bitwise XOR of pixels A and B. Similarly, update the values of pixel C with bitwise xor of pixels A and C, shown in Equation (12).

$$\begin{cases} A \rightarrow P(x, y) \\ B \rightarrow P(x-2, y-1) \\ C \rightarrow P(x-1, y-2) \\ B = A \; xor \; B \\ C = A \; xor \; C \end{cases} \quad (12)$$

Step-11: Iterate over the pixels starting from the pixel from (hieght-3, width -3) to (1,1), and move in a column-wise fashion towards the top, leaving pixels in bottom two rows and moving left as shown in Figure 8(c). According to the current pixel A having coordinates say x and y, select pixels B and C having coordinates $(x+2, y+2)$ and $(x+2, y+1)$ respectively. Set the values of pixel B with bitwise XOR of pixels A and B. Similarly, Set the values of pixel C with bitwise XOR of pixels A and C, shown in Equation (13). The pseudo-code for reverse diffusion steps is provided in Function 9.

$$\begin{cases} A \rightarrow P(x, y) \\ B \rightarrow P(x+1, y+2) \\ C \rightarrow P(x+2, y+1) \\ B = A \; xor \; B \\ C = A \; xor \; C \end{cases} \quad (13)$$

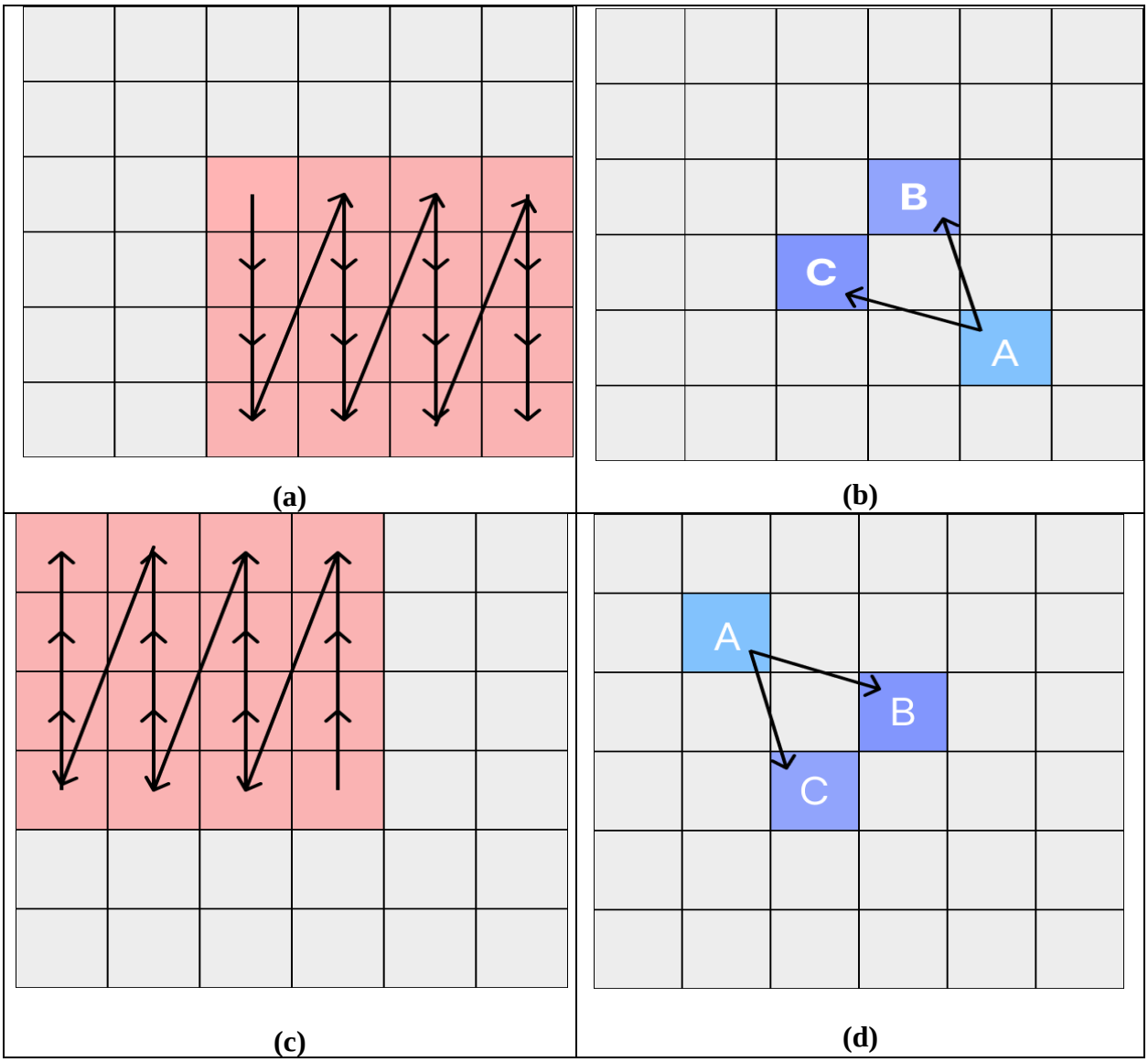


Figure 8: (a) Reverse diffusion path 1 for a 6x6 image **(b)** XOR pattern 1 **(c)** Reverse diffusion path 2 for a 6x6 image **(d)** XOR pattern 2

4. Experimental Setup and Results

The proposed work and its analysis has been performed using processor 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz with 8GB RAM on Windows 11. Python language has been used for coding purposes. For analyzing the performance of the proposed encryption scheme three images ‘baboon.png’, ‘lena.png’, and ‘beach.png’ have been used. The images used are in PNG format. PNG format is chosen because it has data stored in a normal pixel matrix and not in compressed form. For each image, we used exactly the same keys and control parameters. Three

keys and two control parameters were used as mentioned in the encryption scheme. Visual results of image encryption and decryption are shown in Figure 9.

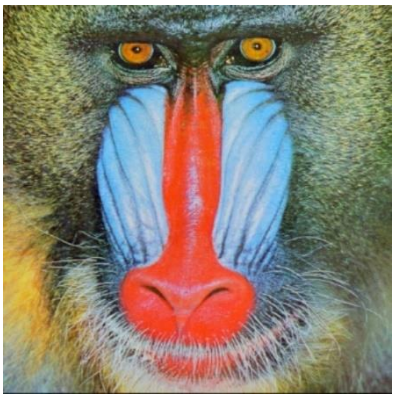
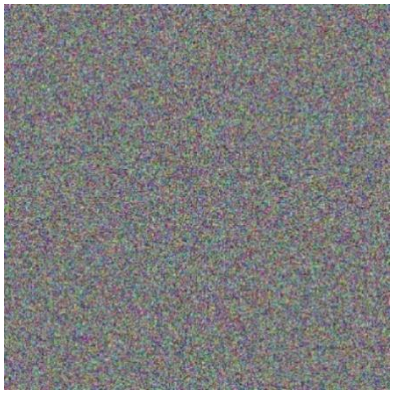
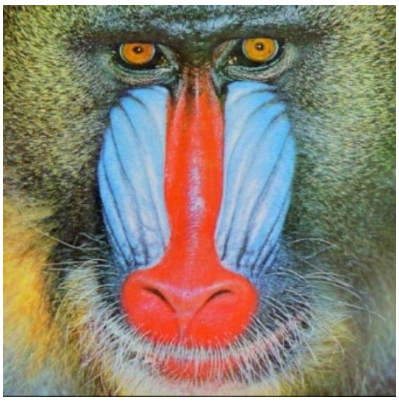
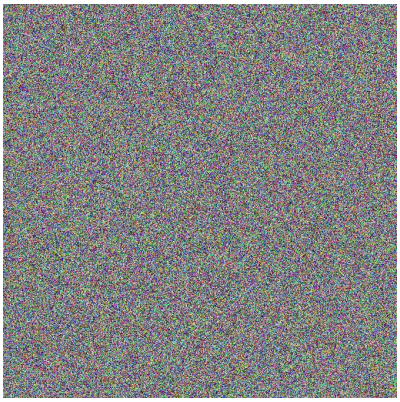
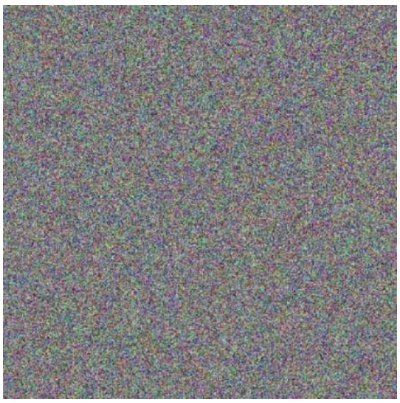
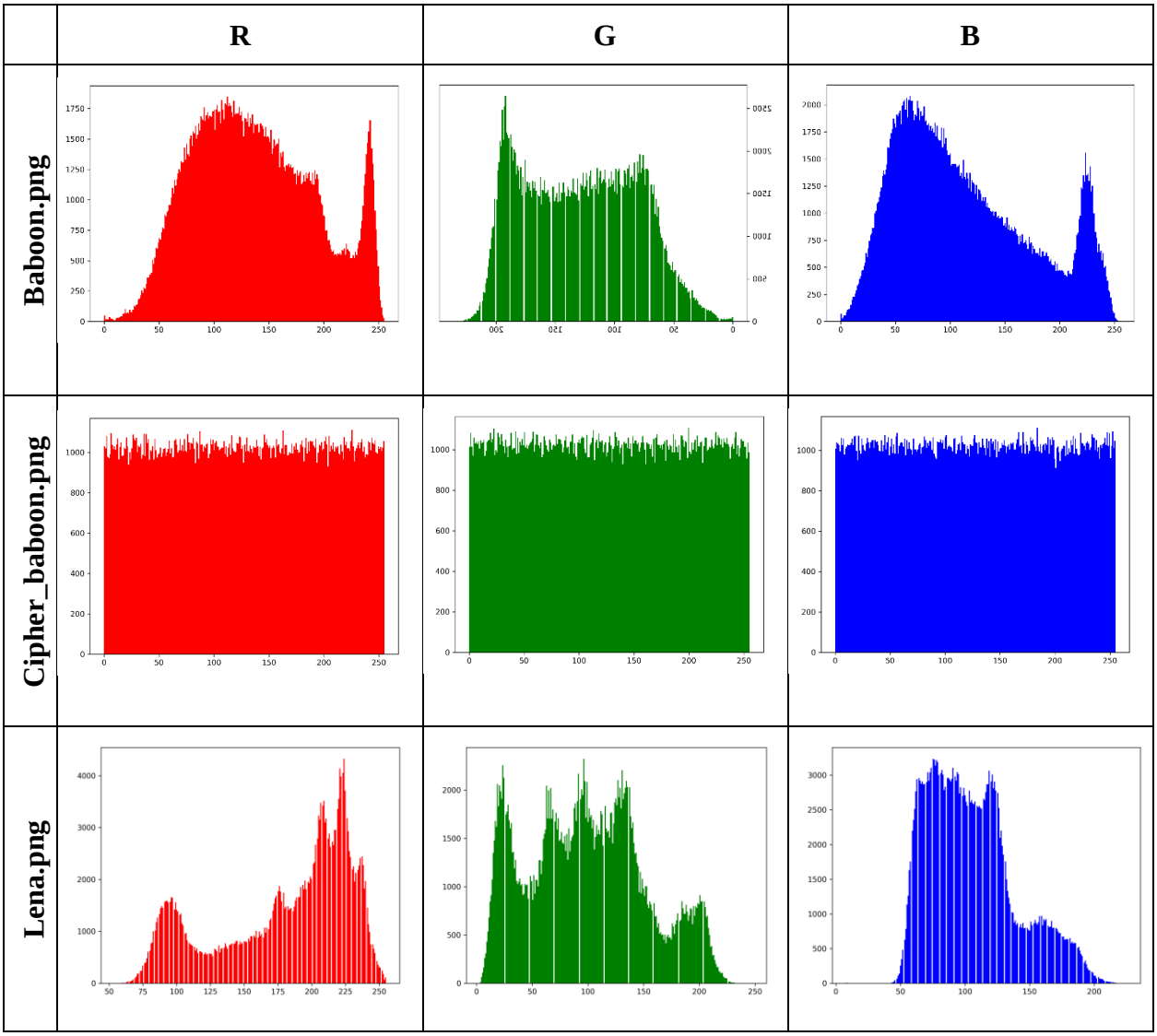
Plain Image	Encrypted Image	Decrypted Image
 **(a)**	 **(b)**	 **(c)**
 **(d)**	 **(e)**	 **(f)**
 **(g)**	 **(h)**	 **(i)**

Figure 9: (a) Original baboon.png **(b)** Encrypted baboon.png **(c)** Decrypted baboon.png **(d)** Original lena.png **(e)** Encrypted lena.png **(f)** Decrypted lena.png **(g)** Original beach.png **(h)** Encrypted beach.png **(i)** Decrypted beach.png

4.1 Histogram Analysis

One method the attackers may use is to predict the pixel values in certain regions based on the histograms of cipher images [25]. These are graphs that represent the relation between the intensity level of a pixel versus the frequency of pixels having that intensity. Thus it is required for a good encryption method should results in a cipher image that has a uniformly distributed histogram. The histograms of the images used are shown in Figure 10. The histograms are uniformly distributed as the frequencies are all ranging in very small intervals.



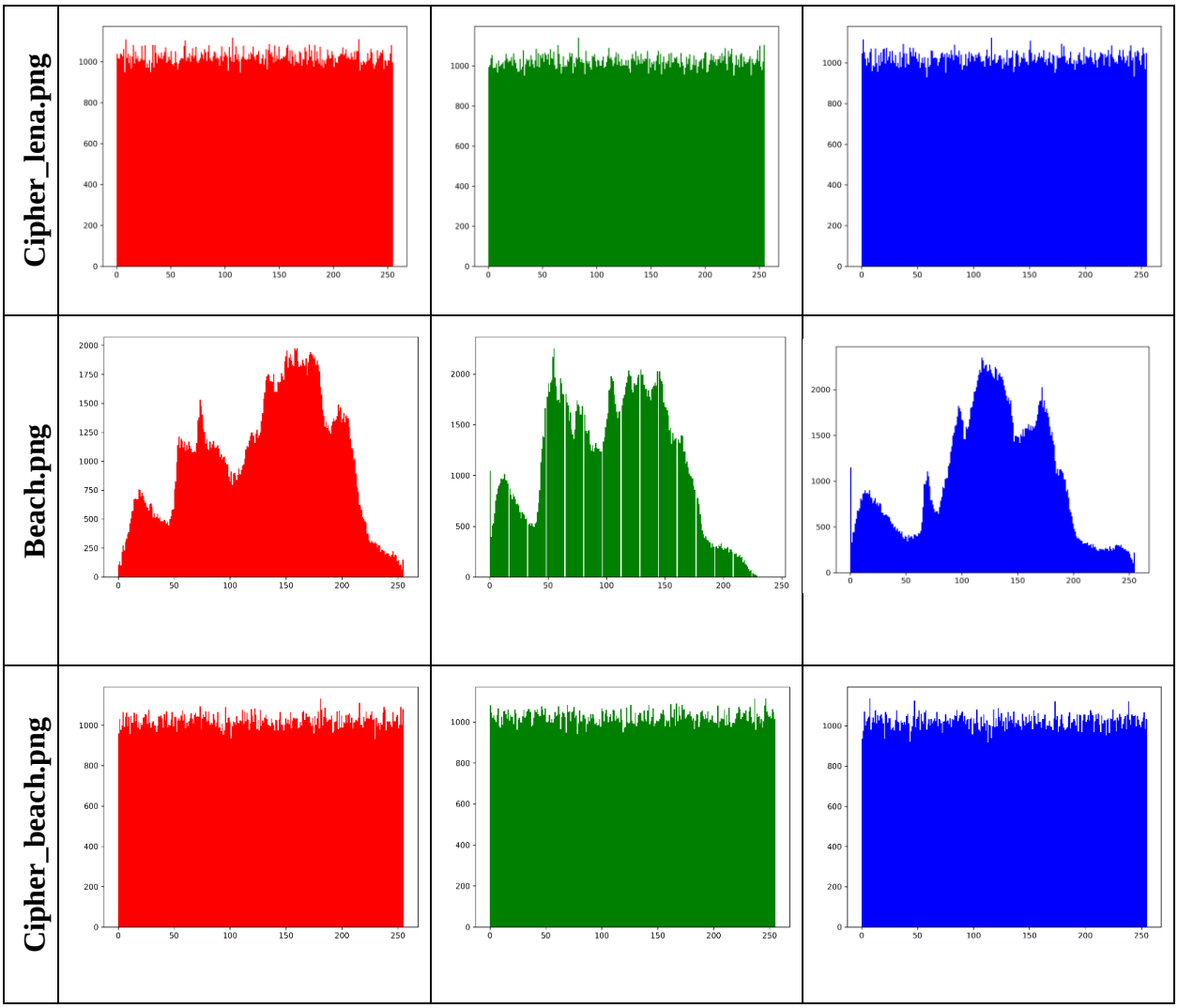


Figure 10: Histogram Analysis

4.2 Entropy Analysis

The entropy of an image is the measure that represents how much a wide range of pixel values are there in the image. Its ideal value is 8 [26]. A good cipher image should have an entropy close to 8. It is calculated using the formula represented by Equation (14) and Equation (15).

$$P_i = V(i)/N \tag{14}$$

$$E = \sum_{i=0}^{L} P_i log_2(P_i) \tag{15}$$

where, for an RGB image, N represents several different pixel values (size * 3). $V(i)$ represents the number of pixel values equal to i. P_i denotes the probability of value i in the image, and E denotes entropy. For the used images entropy values are shown in Table 3, and it can be observed that the cipher image has value close to the ideal value, which proves efficiency of the proposed scheme against various types of attacks.

Table 3: Entropy Analysis

Images	Entropy	
	Original image	**Cipher image**
Baboon.png	7.762436	7.9997
lena.png	7.75019	7.9998
Beach.png	7.764465	7.99975

4.3 Correlation Coefficient Analysis

Correlation is a quantitative measure of relationship, whether casual or not, between two variables. Its value lies in the range -1 to +1. The correlation of adjacent pixels of a cipher image should ideally be 0 [27]. Otherwise, it can be exploited by some illegitimate person to decrypt (maybe a portion of) the encrypted data and get their hands on the sensitive information it withholds. In this paper, the correlation between adjacent pixels has been calculated in horizontal, vertical and diagonal ways for each of the R, G, B components separately. To calculate it for each component Equation (16) to Equation (19) are used.

$$E(X) = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{16}$$

$$D(X) = \frac{1}{N}\sum_{i=1}^{N} (X_i - E(X))^2 \tag{17}$$

$$cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N} (X_i - E(X))(Y_i - E(X)) \tag{18}$$

$$C(X,Y) = \frac{cov(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}} \tag{19}$$

where, N is the total number of pixels in the image. $E(X)$ represents the expectation of image X i.e. mean of all the pixel values divided by N. $D(X)$ represents the standard deviation of the image. $cov(X,Y)$ represents the covariance of the pixels in the images X and Y. $C(X,Y)$, represents the correlation of the two images.

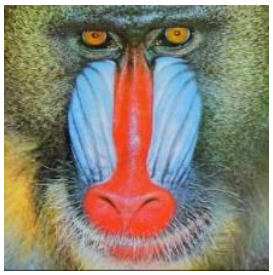
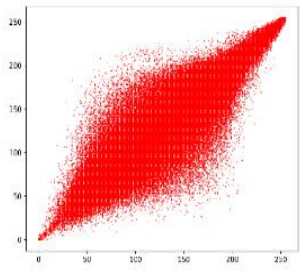
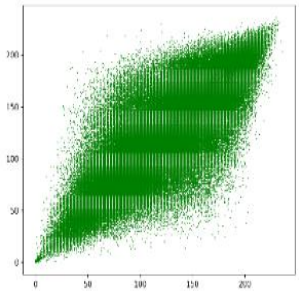
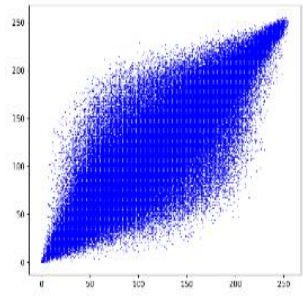
The correlation values of the adjacent pixels calculated in different directions are shown in Table 4. The graphs of correlation between horizontally adjacent pixels of R, G, B components for all the test images are shown in Table 5.

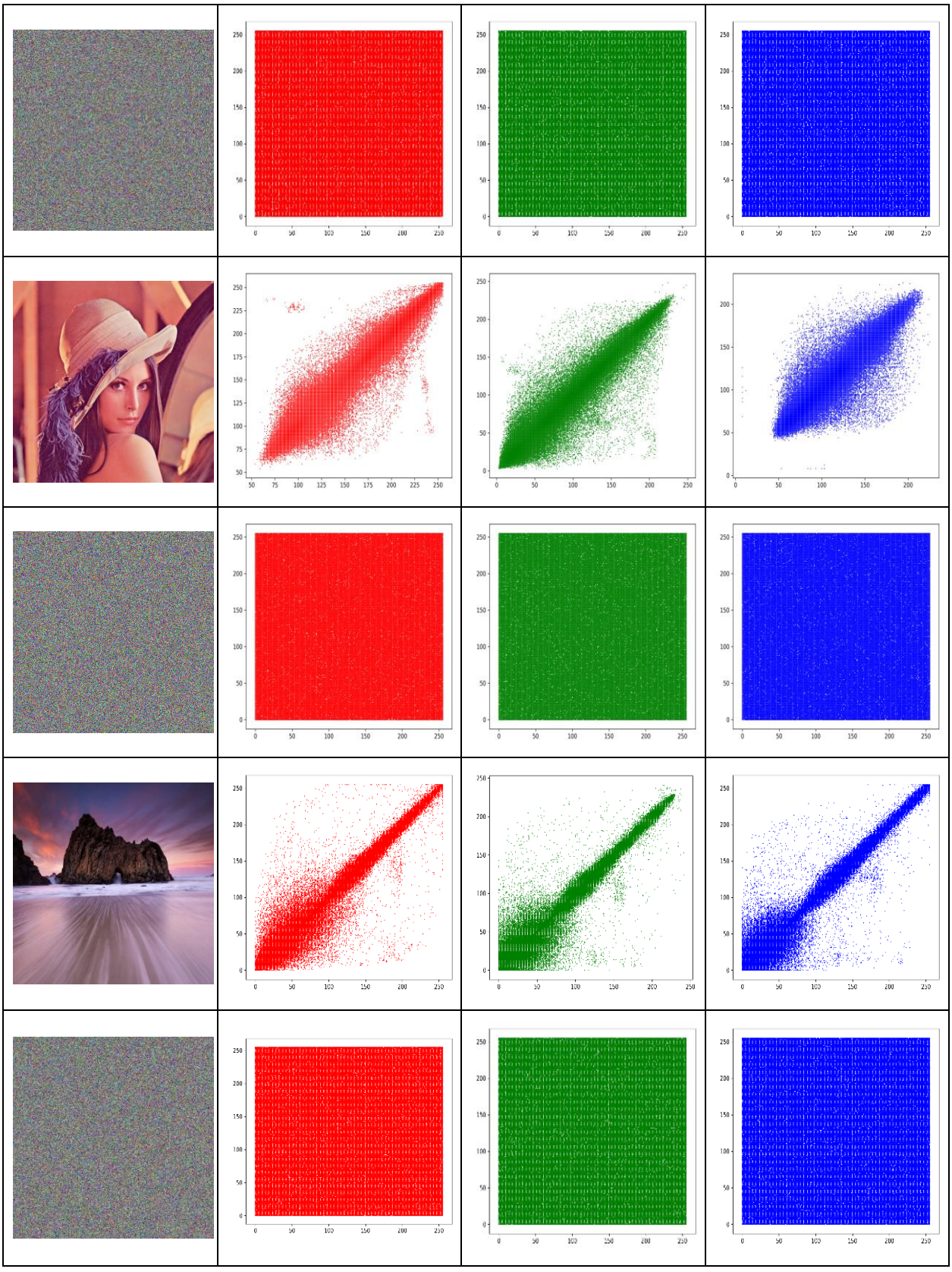
Table 4: Correlation Coefficient Results

Image	Component	Original			Cipher		
		Horizontal	**Vertical**	**Diagonal**	**Horizontal**	**Vertical**	**Diagonal**

Baboon.png	R	0.921741	0.862415	0.837784	0.001027	0.003691	-0.00159
	G	0.864328	0.759122	0.728305	0.00394	0.000002	0.000764
	B	0.907116	0.878229	0.837784	-0.00143	0.003917	-0.00159
	RGB	0.904537	0.84829	0.81992	0.00118	0.00253	-0.00051
Lena.png	R	0.97747	0.98800	0.91569	0.00233	0.00155	0.000809
	G	0.96615	0.98170	0.95258	0.00011	0.00136	-0.00180
	B	0.93040	0.95678	0.91569	0.00045	-0.00033	0.00080
	RGB	0.97800	0.98752	0.97026	-0.00058	0.00108	-0.00088
Beach.png	R	0.982042	0.982874	0.971309	0.000479	0.000865	-0.00093
	G	0.981945	0.980507	0.968097	0.00454	0.00048	-0.00159
	B	0.983665	0.983149	0.971309	0.001498	0.002289	-0.00093
	RGB	0.98339	0.98313	0.97123	0.00217	0.00121	-0.00074

Table 5: Correlation between Adjacent Pixels

Image	**Red component**	**Green component**	**Blue component**
			



4.4 Plain Image Sensitivity

The sensitivity of cipher image towards change in the plain image should be high for a good encryption method. If it's low then, hackers can effectively decrypt the image using chosen plaintext attacks. This indicator is measured by two measures: Number of pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI). NPCR and UACI are quantitative measures that represent change in the image if a new cipher image is generated with a change in one pixel only [28]. To calculate these measures in percentage, Equation (21) and Equation (21) are used.

$$NPCR(X,Y) = \frac{1}{M \times N} \sum_{i,j} D(i,j) \times 100\ \% \tag{20}$$

$$UACI(X,Y) = \frac{1}{M \times N} \sum_{i,j} \frac{|X(i,j) - Y(i,j)|}{255} \times 100\ \% \tag{21}$$

where, M x N represents the size of the image i.e. width*height. $X(i,j)$ and $Y(i,j)$ represent the pixel value at the ith row and jth column in cipher images X and Y, respectively. $D(i,j)$ is 1 if $X(i,j)$ is equal to $Y(i,j)$ otherwise 0. When calculating for one component then only that component of pixel value is considered as well as compared. When calculating all the components together, then the difference in any of the components will give value to $D(i,j)$ 1.

The result of the NPCR and UACI value of the images in percentage are shown in Table 6 and Table 7, respectively.

Table 6: Number of Pixel Change Rate

Images	**NPCR(%)**			
	RGB	R	G	B
Baboon.png	99.61955	99.60365	99.62807	99.62692
lena.png	99.61280	99.59487	99.62882	99.61471
Beach.png	99.61929	99.61586	99.61853	99.62349

Table 7: Unified Average Changing Intensity

Images	**UACI**			
	RGB	R	G	B
Baboon.png	49.98683	50.04539	49.91995	49.99515

lena.png	49.97663	50.00369	49.93621	49.99000
Beach.png	50.04635	50.05055	50.03881	50.04968

4.5 Peek Signal-to-Noise Ratio

Peek Signal-to-Noise Ratio(PSNR) is the measure of how good the quality of a decrypted image is, if the cipher image has corrupted due to noise [29] [30]. It is calculated in dB. For an image encryption method, its value should be high. A high value means that the image decrypted resembles more to the original image. Which means chances of recovering original image are more in case of some noise induced while encryption or storage. It is mathematically calculated using Equation (22) and Equation (23).

$$MSE = \frac{1}{M \times N} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [X(i,j,k) - Y(i,j,k)]^2 \quad (22)$$

$$PSNR = 10 \times log_{10} \left(\frac{L^2}{MSE} \right) \quad (23)$$

where, MSE is Mean Square Error (MSE), X is the original image and Y is the decrypted image of the noise-induced cipher image. $M \times N$ is the size of image. Y represents the largest value of a pixel.

To calculate the PSNR of the images used, a probability of 0.01 is chosen and random numbers are generated using Python's random library. Iterating through each pixel a new number is generated. If number is less than that probability then pixel value is set to (0,0,0), if it generates a number greater than (1 – probability) then the pixel value is set to max i.e. (255, 255, 255), otherwise left unchanged. The results are shown in Table 8.

Table 8: Peek-Signal-to-Noise Ratio

Image	PSNR
Baboon.png	42.44468
Lena.png	42.41304
Beach.png	42.41381

4.6 Comparison with Existing Works

This section shows the comparison of our proposed scheme and some of the recent encryption schemes. The basis of this comparison is `leng.png` of size 512x512. The results are shown in Table 9.

Table 9: Comparison of different schemes

	NPCR	UACI	Entropy	CCH	CCV	CCD
Zhao et al. [31]	99.62	33.46	7.9992	0.0022	-0.0010	0.0021
Wang et al.[32]	99.65	33.48	7.997.	0.0020	-0.0007	-0.0014
Zhang et al. [33]	99.61	33.49	7.9992	-0.0042	0.0005	-0.0036
Xu et al. [34]	99.62	33.51	7.9974	-0.0230	0.0019	0.0034
Proposed approach	99.61	49.97	7.9998	-0.0005	0.0010	-0.0008

5 Conclusion & Future Works

This work in this paper has proposed a novel one-dimensional chaotic map Delta Sine-Cosine map. The Bifurcation diagram and Lyapunov Exponent graph of the Delta Sine-Cosine map have confirmed its excellent chaotic behavior, better chaotic properties and larger range than existing one-dimensional chaotic maps. A new encryption scheme based on novel diffusion operation has been proposed that uses the sequence generated by the chaotic map with DNA encoding. The diffusion process changes up the pixel values of the image by mixing them using bitwise XOR operator. Thus it helps to propagate some change in one pixel to other pixels resulting in higher sensitivity to image data itself. Then one chaotic sequence is used to perform permutation, and two more chaotic sequences are created to create two DNA sequences to perform DNA operations. The performance measures show all the favorable characteristics that a good encryption algorithm should have. The diffusion process happens in a way such that each color component remains in its own domain i.e. they don't influence other colors. In future, that will be one thing to be taken care of. Secondly, right now the keys and control parameters used to generate the sequences by the chaotic map are completely dependent on the input provided. It will be the effort that image data itself influences the chaotic sequences generated in the encryption scheme, and finally, modifying it to be used as encryption algorithm for other types of multimedia.

References

[1] Masood, F., Boulila, W., Ahmad, J., Arshad, Sankar, S., Rubaiee, S., & Buchanan, W. J. (2020). A novel privacy approach of digital aerial images based on mersenne twister method with DNA genetic encoding and chaos. Remote Sensing, 12(11), 1893.

[2] Pankaj, S., & Dua, M. (2024). Chaos based Medical Image Encryption Techniques: A Comprehensive Review and Analysis. Information Security Journal: A Global Perspective, 33(3), 332-358.

[3] Jassim, K. N., Nsaif, A. K., Nseaf, A. K., Priambodo, B., Naf'an, E., Masril, M., ... & Putra, Z. P. (2019, December). Hybrid cryptography and steganography method to embed encrypted text message within image. In Journal of Physics: Conference Series (Vol. 1339, No. 1, p. 012061). IOP Publishing.

[4] Varghese, F., & Sasikala, P. (2023). A detailed review based on secure data transmission using cryptography and steganography. Wireless Personal Communications, 129(4), 2291-2318.

[5] Wazirali, R., Ahmad, R., Al-Amayreh, A., Al-Madi, M., & Khalifeh, A. (2021). Secure watermarking schemes and their approaches in the IoT technology: an overview. Electronics, 10(14), 1744.

[6] Sharma, M., & Garg, R. B. (2016, November). DES: The oldest symmetric block key encryption algorithm. In 2016 International Conference System Modeling & Advancement in Research Trends (SMART) (pp. 53-58). IEEE.

[7] Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. Cryptography and Network Security, 16(1), 11.

[8] Alsaffar, D. M., Almutiri, A. S., Alqahtani, B., Alamri, R. M., Alqahtani, H. F., Alqahtani, N. N., & Ali, A. A. (2020, March). Image encryption based on AES and RSA algorithms. In 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS) (pp. 1-5). IEEE.

[9] Kolivand, H., Hamood, S. F., Asadianfam, S., & Rahim, M. S. (2024). Image encryption techniques: A comprehensive review. Multimedia Tools and Applications, 1-36.

[10] Zhang, B., & Liu, L. (2023). Chaos-based image encryption: Review, application, and challenges. Mathematics, 11(11), 2585.

[11] Erlich, Y., & Zielinski, D. (2017). DNA Fountain enables a robust and efficient storage architecture. science, 355(6328), 950-954.

[12] Church, G. M., Gao, Y., & Kosuri, S. (2012). Next-generation digital information storage in DNA. Science, 337(6102), 1628-1628.

[13] Iqbal, N., Naqvi, R. A., Atif, M., Khan, M. A., Hanif, M., Abbas, S., & Hussain, D. (2021). On the image encryption algorithm based on the chaotic system, dna encoding, and castle. IEEE Access, 9, 118253-118270.

[14] Pak, C., Kim, J., An, K., Kim, C., Kim, K., & Pak, C. (2020). A novel color image LSB steganography using improved 1D chaotic map. Multimedia Tools and Applications, 79(1), 1409-1425.

[15] Farah, M. B., Farah, A., & Farah, T. (2020). An image encryption scheme based on a new hybrid chaotic map and optimized substitution box. Nonlinear Dynamics, 99(4), 3041-3064.

[16] Talhaoui, M. Z., & Wang, X. (2021). A new fractional one dimensional chaotic map and its application in high-speed image encryption. Information Sciences, 550, 13-26.

[17] Khairullah, M. K., Alkahtani, A. A., Bin Baharuddin, M. Z., & Al-Jubari, A. M. (2021). Designing 1D chaotic maps for fast chaotic image encryption. Electronics, 10(17), 2116.

[18] Cun, Q., Tong, X., Wang, Z., & Zhang, M. (2021). Selective image encryption method based on dynamic DNA coding and new chaotic map. Optik, 243, 167286.

[19] Lu, Q., Yu, L., & Zhu, C. (2022). Symmetric image encryption algorithm based on a new product trigonometric chaotic map. Symmetry, 14(2), 373.

[20] Dhingra, D., & Dua, M. (2023). A novel Sine–Tangent–Sine chaotic map and dynamic S-box-based video encryption scheme. The Imaging Science Journal, 71(6), 549-572.

[21] Liang, Q., & Zhu, C. (2023). A new one-dimensional chaotic map for image encryption scheme based on random DNA coding. Optics & Laser Technology, 160, 109033.

[22] Feng, S., Zhao, M., Liu, Z., & Li, Y. (2024). A novel image encryption algorithm based on new one-dimensional chaos and DNA coding. Multimedia Tools and Applications, 1-23.

[23] Dhingra, D., & Dua, M. (2024). Medical video encryption using novel 2D Cosine-Sine map and dynamic DNA coding. Medical & Biological Engineering & Computing, 62(1), 237-255.

[24] Kumar, A., & Dua, M. (2023). Audio encryption using two chaotic map based dynamic diffusion and double DNA encoding. Applied Acoustics, 203, 109196.

[25] Kumar, A., & Dua, M. (2024). A novel exponent–sine–cosine chaos map-based multiple-image encryption technique. Multimedia Systems, 30(3), 141.

[26] Bisht, A., Dua, M., & Dua, S. (2019). A novel approach to encrypt multiple images using multiple chaotic maps and chaotic discrete fractional random transform. Journal of Ambient Intelligence and Humanized Computing, 10, 3519-3531.

[27] Kumar, A., & Dua, M. (2023). A GRU and chaos-based novel image encryption approach for transport images. Multimedia Tools and Applications, 82(12), 18381-18408.
[28] Kumar, A., & Dua, M. (2021). Novel pseudo random key & cosine transformed chaotic maps based satellite image encryption. Multimedia tools and applications, 80(18), 27785-27805.
[29] Dua, M., Makhija, D., Manasa, P. Y. L., & Mishra, P. (2022). 3D chaotic map-cosine transformation based approach to video encryption and decryption. Open Computer Science, 12(1), 37-56.
[30] Shelza, S., & Ritu, V. (2020). A Pareto-optimal evolutionary approach of image encryption using coupled map lattice and DNA. Neural Computing & Applications, 32(15), 11859-11873.
[31] Zhao, J., Wang, S., Chang, Y., & Li, X. (2015). A novel image encryption scheme based on an improper fractional-order chaotic system. Nonlinear Dynamics, 80, 1721-1729.
[32] Wang, X. Y., Zhang, Y. Q., & Bao, X. M. (2015). A novel chaotic image encryption scheme using DNA sequence operations. Optics and Lasers in Engineering, 73, 53-61.
[33] Zhang, W., Yu, H., Zhao, Y. L., & Zhu, Z. L. (2016). Image encryption based on three-dimensional bit matrix permutation. Signal Processing, 118, 36-50.
[34] Xu, L., Li, Z., Li, J., & Hua, W. (2016). A novel bit-level image encryption algorithm based on chaotic maps. Optics and Lasers in Engineering, 78, 17-25.