Design of DNA Based Color Image Cryptosystem and its Security Analysis

ARTICLE INFO

Keywords:
Color Images Encryption
Color Images Decryption
DNA based Encoding
DNA based Decoding
Experimental Analysis

ABSTRACT

The goal of image cryptosystems is to protect image transmission when there are network adversaries present. To ensure secrecy, images are subject to encryption to produce unintelligible cipher images; the techniques used for this process differ significantly from those applied to text data. The majority of the cryptosystem consider complicated or confusion–diffusion architectures that changes and permute the values of the pixels. These frequently entail binary operations like bitwise XOR, plus–minus, DNA operations, etc; are carried out utilising chaotic maps, each of which have certain limitations. This paper employs a binary function that can be applied to both traditional and DNA techniques for coloured natural images, and can be applied to any kind of image cryptosystem. Each of the color component of the image follows some steps which start with inter channel mixing and mix rows which takes the value from initial vector and mixes it, then Arnold cat map algorithm is applied to shuffle the pixels. Secondly spiral mixing of the pixel is applied in four different directions to prevent differential attack. Finally, encoding, substitution, and decoding based on DNA is carried out. A multiple collapse chaotic map is used to derive initialization vector, rule charts and DNA substitution map which are used in encoding and decoding processes. Analysis and experimental findings shows that our cryptosystem has significant performance and different metrics shows it can withstand different type of attacks.

1. Introduction

The ease with which technology is now available, such as smartphones and the Internet, has made sharing images quite simple. These days, a growing number of multimedia images are routinely sent across public networks, and there is an increasing focus on how to safeguard them. One of the best techniques is picture encryption, which converts the important images into noise-like ones that can be sufficient to protect the hidden images. In contrast to text data, images are stored in several formats based on their intended use and have highly associated properties. Both lossless and lossy formats are used to store natural images. Cryptosystems intended for images are distinct from those used for other purposes since they are made to change pixel intensity values and guarantee the removal of statistical and visual information.

DNA based color images refer to the encoding of the color images into DNA sequences called as DNA computing. The basic idea behind DNA computing is that DNA can be used to store and process information in a way that is similar to how a computer uses binary digits (bits) to represent data. Instead of representing images using traditional pixel values the image data is converted into a sequence of DNA nucleotides(A for Adenine, T for Thymine, C for Cytosine and G for Guanine). This process of converting color images's pixel values into a DNA sequence involve mapping the pixel value of a image to specific sequence of DNA bases. For each color channel i.e. RGB of pixel might be represented by a certain DNA bases or combination of bases. Overview of Conversion- The value of pixel 0-255 can be converted into DNA sequence. Using a set of predefined rules we can map every two bit of a 8 bit binary sequence to one of the DNA nucleotides.

Most image cryptosystems produce quasirandom sequences, wherein the values are predetermined by the initial values supplied to them; these sequences serve as the algorithm's symmetric key, by using

specific chaotic maps such as the 2D-Infinite Collapse Map [1], Logistic Maps [2] 3D chen chaotic sysytem [3], Hyper Chaotic Systems [4], Logistic Tent System [5], 4D memristive Hyper Chaos [6], Fractional Order Hyperchaotic System [7], Fractional Fourier Transform [8] etc.

Most of the cryptosystem follows complex architectures, Xiuli Chai et al.[3] devised a method of encryption that creates stochastic sequences using the Chen chaotic system, which are then used to form arrays for key stream creation and image permutation. The two stages of their image encryption approach are diffusion and permutation. Mohammad Hossein Moattar and Abolfazl Yaghouti Niyat [4] proposed a cryptosystem in which color image is firstly decomposed into R(Red), G(Green) & B(Blue) channels, permutation is used to mix up all three components. Using DNA encoding rules permuted components are converted into DNA matrices. After this, Diffusion is applied. To enhance security they applied second confusion scheme. In paper [6] Zhentao Liu et al. suggested use of 4D memristive hyper chaos with dynamic DNA for colour image encryption in which the process to obtain the R, G, and B DNA matrices, dynamic encoding is applied independently to each of the image's three channels. For all three matrices, dynamic dispersion and dispersal are utilised. Finally, DNA decoding and component integration resulted in a single encrypted image. In the paper[8] M. Amine Ben Farah et al. introduced a cryptosystem with the operations like DNA encoding, XOR with encoded image and SHA-2 algorithm, substitution, DNA encoding, Fractional fourier transform applied on encoded image, XOR is applied on transformed image with Chaotic sequence obtained from Lorenz system, again substitution, again Fractional Fourier tranform of this substituted image, again XOR with chaotic sequence and obtained substituted image, finally Partial Fourier Transform and XOR with third chaotic sequence and ciphered image is obtained.

Different authors have used number of algortihms for color image's encryption. To improve the execution of the cryptosystem, they entails several operations including DNA based encoding, DNA-XOR, DNA complement, DNA decoding, etc. The majority of cryptosystems rely on addition–subtraction (+,−) operations [3] or XOR operations [2,3,4], each of which has certain drawbacks. The disadvantages of using XOR in these operation are proved in paper [9]. In the same paper to overcome these limitation a new binary operator is proposed which is applicable in both conventional and DNA based cryptosystems.

This research will use a binary operator (⊗) proposed in paper[9]. The operator satisfies the requirements to be utilised with DNA nucleotides and image with pixel depths of 8 and 16 bits.

A new cryptosystem that can be used to safely and losslessly encrypt color images has been developed in order to demonstrate the operator's viability with color images as well. The cryptosystem performs fairly because it functions well for all color images with different dimensions and it can withstand a variety of attacks.

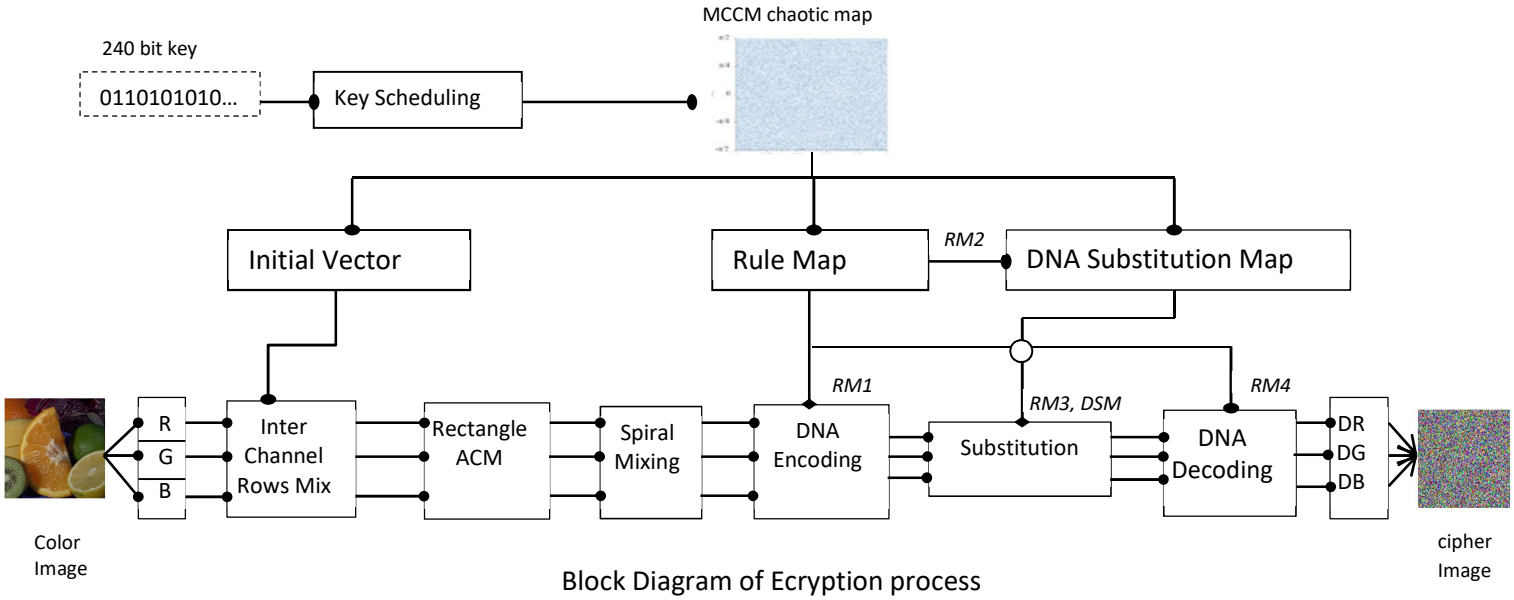
The operator will be applied on RGB colour images in this study, and its output will be compared to the output of other cryptosystems that are utilised as references.

2. Proposed Work

An encryption and decryption technique utilising an operator ⊗ is presented in the proposed work. Measurements of the cryptosystem's performance have also been made using the experimental results and analysis. Lastly, a comparison with other operators, such as XOR and addition-subtraction, is carried out.

- The new methods used in the algorithm, such as inter channel mixing, mix rows, spiral mixing are straightforward but have the power to significantly reduce the visual information in the original images.
- The cryptosystem utilizes the modified ACM algorithm proposed in paper [9], only images with similar dimensions can be shuffled using traditional ACM. Unevenly sized images shuffling is also possible with the modified ACM.
- To improve security and give the encryption process non-linearity, the suggested cryptosystem makes use of DNA encoding, substitution based on DNA with the proposed operator and DNA decoding.
- The cryptosystem has been tested using a number of statistical and security measures, together with experimentation and analysis to improve its performance, to ensure that it is resistant to a variety of threats.
- The comparative analysis shows that suggested operator outperforms other operators, such as XOR or (+,−) with the color images as well, and hence overcomes their shortcomings.

2.1 Encryption Algorithm Flow Chart



Block Diagram of Ecryption process

2.2 The Proposed Crptosystem

The cryptosystem makes use of six 2D-Multiple Collapse Chaotic Maps, which are produced by the Key Scheduling Algorithm from a 240-bit symmetric key. The nonlinear maps produce quasirandom sequences in the interval [−1, 1], from which certain functions are used to construct values and maps such as DNA substitution maps, rule maps, and initial vector. The different procedures required to encrypt the image make use of these variables and mappings.

The first stage involves intermixing of channels of the image then utilising the proposed operator ⊗ to operate the rows of the intermixed images with the initial array. Using a modified version of Arnold's Cat Map [10], which can be applied to both square and rectangular images, the pixels of the composite image are jumbled. The suggested operator is then used to further mix the pixels in the directions of forward spiral row, forward spiral column, reverse spiral row, and reverse spiral column. These keyless procedures are necessary to eliminate the link between pixels and the image's visual perception. Lastly, a non-linear layer has been added by combining the suggested operator with the idea of DNA cryptography. One of the eight encoding principle identified by the rule chart is applied to the mixed image that was obtained in the preceding phases, converting it into a sequence of DNA nucleotides. At this stage DNA encoded image is obtained and these encodings are further substituted using proposed operator with the substitution map. Lastly, one of the eight DNA decoding guidelines identified by a different rule chart is applied to decode the substituted DNA sequences.

2.2.1 Private Key and Chaotic Map

The suggested approach to create the initial conditions of the 2D MCCM map, which has been thoroughly covered in the paper[11], uses a symmetric key with a length of 240 bits.

These equations are used to construct the 2D MCCM as specified in [11]:

$$Xn+1 = \arctan\left(\frac{b}{10 * a * yn} + \tan(a * \pi * xn)\right)$$

$$Yn+1 = \arctan\left(\frac{b}{a * xn} + \tan(10 * a * \pi * yn)\right)$$

where the values of the tuning parameters, a and b, are taken from the key. The starting conditions of the kth dynamical map *M(k)* are represented by (*x0*, *y0*). The M(1) map is constructed using key and other maps are constructed using last values of previous maps.

The operator ⊗ in paper[9] is given as

$$c \otimes d = \{(c + 1) \times (d + 1) \bmod p\} - 1$$

a) *Initial Vector*

With each row of the given image, the initial array (IV) is utilised to generate a quasirandom sequence

that may be operated with ⊗. For an image of size *m * n* and bit depth P, the IV-dimension is *1 × m*. The values are obtained in the following way from $M(1)$.

$$\mathrm{InV}\,(1,\mathrm{i}) = |\mathrm{xi}\,| \times |\mathrm{yi}\,| \times (2^{31} - 1)\mathrm{mod}\,(2^{\mathrm{p}}),$$

$$xi\,, yi \in M(1), i = 1, 2, \ldots m$$

b) *Rule Map*

To construct one of the eight DNA encoding-decoding principle, as listed in Table 1, the rule mappings are utilised to create a quasirandom sequence of integers from 1 - 8. Because every two bits are mapped to one nucleotide, an image of size m * n of bit depth p will have a rule map of dimension m × (n · (p∗2)). The dynamical map is sclaed to the dimension of image in order to produce the rule map. Modular operation is used to transform the precept, which are preferably selected from each dimension, to values between 1 and 8.

Table 1
DNA Encoding rules

Binary digits	R1	R2	R3	R4	R5	R6	R7	R8
00	A	A	C	G	C	G	T	T
01	C	G	A	A	T	T	C	G
10	G	C	T	T	A	A	G	C
11	T	T	G	C	G	C	A	A

C) *DNA Substitution Map*

Using the proposed operator ⊗, the DNA Substitution chart—a pseudorandom string of DNA nucleotides (A, C, T, G)—is utilised to work with the mixed image which is DNA encoded. With size of *m*n* an image with a *p* bits pixel depth will be enlarged to a DNA sequence of size m × (n.(p/2)) which the DNA Substitution Chart's size ought to correspond to. The instruction for constructing a DNA Substitution chart is described in paper[9].

After obtaining the necessary maps, the images can be encrypted. The decryption procedure also requires the same maps. The next subsection provides a description of the encryption procedure.

2.2.2 *Encryption*

An image in color is changed into an incomprehensible distorted image during the inscribe process. The following steps make up the encryption process:

a) Split the R, G, B pixel matrices from the color image into three different matrix say red, green, blue. We need to apply all steps to every matrix inorder to encrypt the image in best way.

b) *Inter Channel Mixing and Mix Rows*

In this stage, the channels of the image are intermixied with each other using some equations. The values from the initial vector (IV), which has already been generated, are mixed and distributed throughout the inter mixed image's rows. The IV will have a dimension of 1 * m for a plain image I with dimensions m * n. To create image I', the first value in each row of the matrix is operated with ⊗ using the appropriate values of the initial array.

The channel are Inter Mixed using these equations:

$$R' = R \otimes (\mathrm{G} \otimes \mathrm{B})$$
$$G' = G \otimes (\mathrm{R}' \otimes \mathrm{B})$$
$$B' = B \otimes (\mathrm{R}' \otimes \mathrm{G}')$$

The value of the first element in each row is propagated along its neighbor pixels using the following equation to skew the value in the neighbouring pixels:

$$I'(\mathrm{g}, \mathrm{h}) = \mathrm{I}(\mathrm{g}, \mathrm{h}) \otimes \mathrm{I}'(\mathrm{g}, \mathrm{h} - 1);$$
$$h = 2, 3 \ldots, n;\ \forall g \in \{1, \ldots, m\}$$

The process of inter channel mixrows will be applied on the image received after splitting color image into matrices i.e. R, G, B, respectively as.

$$R' = interMix(r, g, b)$$
$$G' = interMix(g, R', b)$$
$$B' = interMix(b, R', G')$$

$$mixedImageRed = mixRows(R', IV)$$
$$mixedImageGreen = mixRows(G', IV)$$
$$mixedImageBlue = mixRows(B', IV)$$

Algorithm for InterMix

```
InterMix(r, g, b):
  m= length of r
  n= length of first row
  imix = [][]
  brac = g will operate with b using ⊗
  imix = r will operate with brac using ⊗
  return imix
```

The algorithm for mixrows is same as defined in paper[9]

c)*Rectangular ACM*

Arnold's Cat Map(ACM)[10] is a disorganised chart that is used to rearrange the given image. When a 2D array has dimensions of L*L, ACM is expressed as:

$$\begin{matrix} g' \\ h' \end{matrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{matrix} g \\ h \end{matrix} (mod\ L)$$

where g, h, g′, h′ ∈ [1, N]. (g, h) and (g′, h′) represents, respectively, the original and scrambled images' position indices. The drawback of employing directly ACM algorithm is that its definition is limited to identical size images. Nonetheless, uneven proportions are a regular occurrence in both natural and medical imaging. The image is separated into minimal squares, and ACM is applied independently to each of them in order to fix this problem.

Let m*n represent an image's dimension. To divide it into as few squares as feasible, the length of each square's side will be N = min(m, n). Therefore, the proportion of squares that are obtained will be α = ⌈max(m, n)/N⌉. If m and n are distinct combination of one another, then L = N − (max(m, n) mod N)), an additional length L pixel remains with the longer side. There will be some overlap between the squares in order to accommodate this additional space within the image. There can be a maximum of α − 1 overlaps to ensure almost equal overlapping; the duration of the last overlap i.e. eta=⌊L/(α − 1)⌋. The last overlap modifies the excess portion that is still present.

Given that m < n for an image, N = m and the initial value of I[1,1] of first square, a = 1, b = 1. For the initial α − 1 squares, the subsequent values will be ascertained as b′ = b + N − eta. To accommodate the extra square that remains, the y-coordinate for the last square will be y–N. For images when m>n, same computations can be performed by transposing the m and n values. Next, ACM is applied to each separated square individually. On the other hand, ACM is regular; that is, the original image is restored following a definite number of iterations. The image N's dimension determines this regularity. To achieve a well-shuffled image and avoid regularity, ACM will be applied to the image itself t number of times, where t is the highest prime number less than or equal to (N/2), N being the size of each image.

The process of applying ACM on images will be applied on the image received after mixrows operation on every matrix i.e. R, G, B, respectively as

$$imageACMR = applyACM(mixedImageRed)$$
$$imageACMG = applyACM(mixedImageGreen)$$
$$imageACMB = applyACM(mixedImageBlue)$$

d) *Spiral Mixing*

In order to prevent differential attacks on the image, this stage job is to propagate the pixel's value information throughout it in spiral manner. This procedure helps remove the image's visual information even if it doesn't require an encryption key. Let m and n represent the input image I's row and column counts, respectively. There are four parts to the mixing step.

- Forward Spiral Row Mixing
  The mixing is started from I[0,0] and goes to [m,n] in spiral fashion. Every element value is propagated

Algorithm for Forward Spiral Row

```
loop e starting from 0 to m:
    if e is even:
      if e!=0: (edge case)
         img[e,0] = operate img[e,0] ⊗ img[e-1,0]
      for h starting from1 to n:
         img[e,h] = operate img[e,h] ⊗ img[e,h-1]
      img[e+1,h] =operate img[e+1,h] ⊗ img[e,h]
    else:
      for h starting from n-2 to 0:
        img[e,h] = operate img[e,h] ⊗ img[e,h+1]
```

- Forward Column Mixing
  As forward row, in this step we move in column from im[0][0] to im[0][n]. The values of each element is propagated in the columns.

Algorithm for Forward Spiral Column

```
for each column starting from 0 to n-1:
    if h is even:
      if h ≠ 0: (edge case)
        im[0,j] = operate im[0,j] ⊗ im[0,j-1]
        for each row starting from 1 to m-1:
          im[e,h] = operate im[e,h] ⊗ im[e-1,h]
        im[e,h+1] = operate im[e,h+1] ⊗ im[e-1,h]
    else:
      for each row starting from m-2 to 0:
        im[e,h] = operate im[e,h] ⊗ im[e+1,h]
```

- Backward Spiral Row Mixing
  Reverse spiral rows mixing is the reverse of rows mixing in which we start from ig[m,n] and go towards ig[0,n]
  The values are operated with each other.

Algorithm for Backward Spiral Row

```
for loop e starting from m to 0:
    if e is even:
      for each column h starting  from 1 to n-1:
          ig[e,h] = operate ig[e,h] ⊗ ig[e,h-1]
   else: (if e is odd)
       if e != m-1:
          ig[e,n-1] = operate ig[e,n-1]⊗ig[e+1,n-1]
      for each column h from  n-1 to 1:
          ig[e,h-1] = operate ig[e,h-1] ⊗ ig[e,h]
      ig[e-1,0] =operate  ig[e-1,0] ⊗ ig[e,0]
```

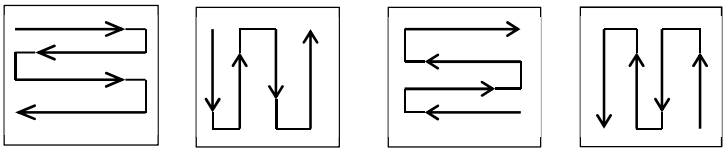
- Backward Spiral Column Mixing
  In this step we go backward from ig[m,n] towards zero in column way
  The element value is propagated from the very last value to the first value of the matrix

Algorithm for backward spiral column mixing

```
for each column h starting from n-1 to 0:
     if h is even:
       for each row e starting from 1 to m-1:
            ig[e,h] =operate  ig[e,h] ⊗ ig[e-1,h]
    else (if j is odd):
        if j ≠ n-1:
           ig[e,h] =operate  ig[e,h] ⊗ ig[e,h+1]
       for each row e from m-1 to 1:
          ig[e-1,h] = operate ig[e-1,h] ⊗ ig[e,h]
       ig[e-1,h-1] =operate  ig[e-1,h-1] ⊗ ig[e-1,h]
```

These four steps are required to be processed on the image one after other in continuation on every image taken into consideration. The process of mixing image will be applied on the image received after applying ACM on every matrix i.e. R, G, B, respectively as

$$imgmixR = spiralMixing(imageACMR)$$
$$imgmixG = spiralMixing(imageACMG)$$
$$imgmixB = spiralMixing(imageACMB)$$



[a] [b] [c] [d]

Figure: a – d are the spiral mixing operations described earlier.

e) *DNA Encoding*

One of the encoding principle transforms each pixel value in the image into a string of DNA nucleotides. At a time, two bits are extracted and transformed into a nucleotide by applying the rule found in the obtained rule map. As a result, a string half of length i.e. (p/2) nucleotides per pixel will be created from an image with a bit depth of p-bit.

The encoding process of an image will be applied on the image received after mixing on every matrix i.e. R, G, B, respectively as

$$encodedR = encoding(imgmixR)$$
$$encodedG = encoding(imgmixG)$$
$$encodedB = encoding(imgmixB)$$

25	54	65
12	78	68
76	98	31

⇨

CACA	CGCG	ACAC
CCTT	ACTG	ACCT
ACTT	ATAG	CATA

Figure: Illustration of Encoding process

f) *Substitution*

The DNA sequences D that are received after the image has been encoded are substituted in the replacement process with ⊗ using the sequence that is obtained from the DNA replacement Map (DSM). The mapping of Σ = {A, C, T, G} to S = {0, 1, 2, 3} for DNA operation requires the selection of one among eight encoding rules from the values derived from the rule map (RM). Eight options will appear when ⊗ is used, one for each of the encoding rules that are applied to a nucleotide's pair; yet, some pairs may give identical outcomes for distinct rules. Since of this, the dynamical DNA sequence that is left over after this procedure is resistant to algebraic attack since it depends on the rule's table as well as the elements of the DNA substitution table, both of which are necessary to decrypt the cryptic image and determine the original key.

Table 2: DNA Substitution Rules

	A	C	G	T
A	A	C	G	T
C	C	T	A	G
G	G	A	T	C
T	T	G	C	A

R1

	A	C	G	T
A	A	C	G	T
C	C	T	A	G
G	G	A	T	C
T	T	G	C	A

R2

	A	C	G	T
A	G	A	T	C
C	A	C	G	T
G	T	G	C	A
T	C	T	A	G

R3

	A	C	G	T
A	C	T	A	G
C	T	G	V	A
G	A	C	G	T
T	G	A	T	C

R4

	A	C	G	T
A	G	A	T	C
C	A	C	G	T
G	T	G	C	A
T	C	T	A	G

R5

	A	C	G	T
A	G	A	T	C
C	A	C	G	T
G	T	G	C	A
T	C	T	A	G

R6

	A	C	G	T
A	T	G	C	A
C	G	A	T	C
G	C	T	A	G
T	A	C	G	T

R7

	A	C	G	T
A	T	G	C	A
C	G	A	T	C
G	C	T	A	G
T	A	C	G	T

R8

Both the rule map (RM) and the DNA substitution map (DSM) must be used during the substitution procedure. Following encoding on each matrix, R, G, and B, respectively, the process will be applied to the image received as

$$substitutedR = substitution(encodedR, DSM, RM)$$
$$substitutedG = substitution(encodedG, DSM, RM)$$
$$substitutedB = substitution(encodedB, DSM, RM)$$

Encoding, substitution and decoding all three processes uses three different rule maps. As shown in flow chart substitutionuses rule map 3.

g) *Decoding*

One of the eight decoding rules is used to the chaotic DNA sequences in order to transform them to decimal values; the rule map's sequences determine which rule is applied. Then, a single cipher color image is created by combining the three matrices—cipher red, cipher green, and cipher blue obtained after decoding. The outcome of this decoding process is the final cipher image. The algorithm for DNA decoding to decimal value is given in paper[9].

The process of decoding image will be applied on the image received after applying substitution on every matrix i.e. R, G, B, respectively as

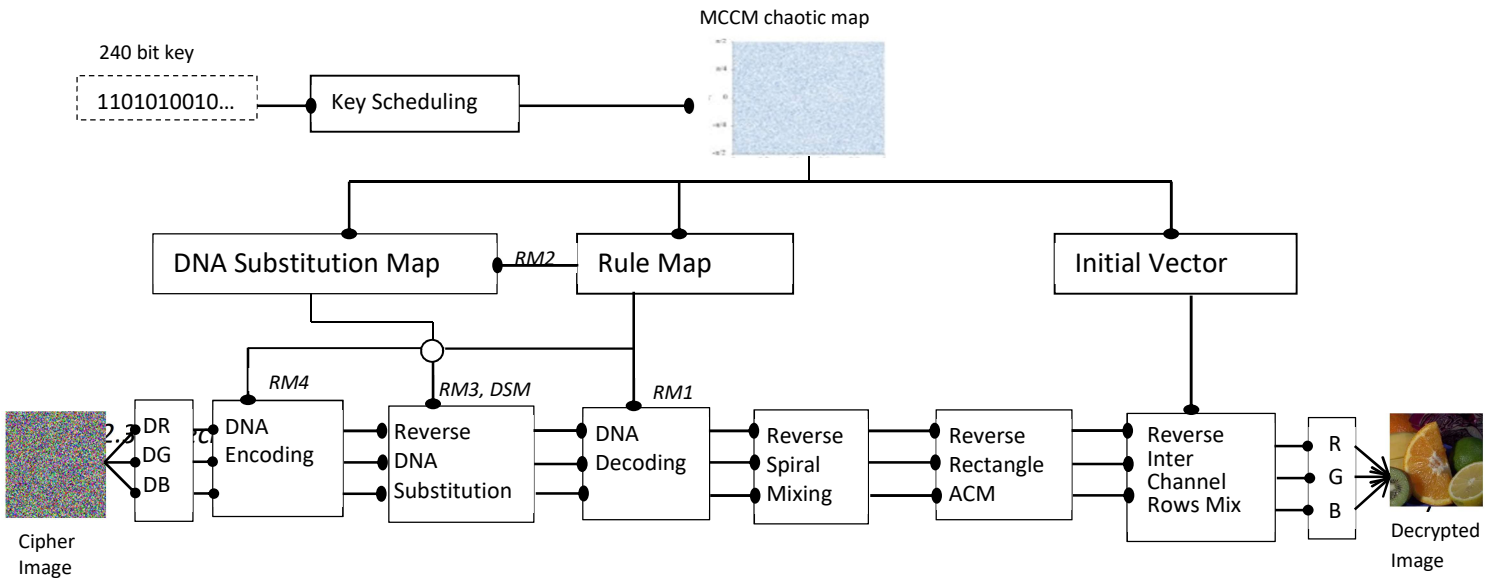
$$finalCR = decoding(substitutedR)$$
$$finalCG = decoding(substitutedG)$$
$$finalCB = decoding(substitutedB)$$

Merging all three decoded matrix into final cipher image. The function merge is inbuilt function of python's opencv library.

$$cipherImage = merge(finalCR, finalCG, finalCB)$$

The cipher image that is produced is noise-like and random, meeting the criteria for a secure image. This cipher picture is resistant to cryptographic attacks, has low correlation, and high entropy. The decryption procedure and security analysis are displayed up front.

2.3 Decryption Algorithm Flow chart



After obtaining cipher image we proceed for its decryption as on reciever side it needs to be decrypted. The encryption and decryption processes are inverted. The steps that are taken in last stages will now be taken in initial stages as we move in reverse fashion. As presented in flow chart, we again split the cipher images into R, G, B components and applies DNA encoding. If we compare encyption and decryption flow chart we can see we are dealing with DNAs in the later stages in encryption but in decryption we are using DNAs in initial stages. At first we do DNA encoding then we apply reverse DNA substitution using reverse substitution rules followed by DNA decoding. Encoding and Decoding have the same algorithm but substitution have minor change, it uses reverse substitution rules. After decoding we go for reverse spiral mixing which internally follows reverse fashion of mixing in encryption. Then we apply reverse modified ACM followed by reverse mix rows and reverse inter channel mixing. At last we again merge RGB into final decrypted image. The main point of decryption is to use the same key as it was used in encryption for getting 2D MCCM, initial vector, rules maps and DNA substitution map. The steps are shown ahead.

a) Splitting final crypted image into R for Red, G for Green and B for Blue components.

b) *DNA encoding*

As a first step now DNA encoding is performed on the R, G, B, matrices received after splitting. There is no change in the algorithm of encoding.

c) *Reverse Substitution*

For reverse substitution we define reverse encoding rules between DNA nucleotides.

Table 3: Reverse Substitution Rules

	A	C	G	T
A	A	G	C	T
C	C	A	T	G
G	G	T	A	C
T	T	C	G	A

R1

	A	C	G	T
A	A	G	C	T
C	C	A	T	G
G	G	T	A	C
T	T	C	G	A

R2

	A	C	G	T
A	C	A	T	G
C	T	C	G	A
G	A	G	C	T
T	G	T	A	C

R3

	A	C	G	T
A	G	T	A	C
C	A	G	V	T
G	G	C	G	A
T	C	A	T	G

R4

	A	C	G	T
A	C	A	T	G
C	T	C	G	A
G	A	G	C	T
T	G	T	A	C

R5

	A	C	G	T
A	G	T	A	C
C	A	G	C	T
G	T	C	G	A
T	C	A	T	G

R6

	A	C	G	T
A	A	G	C	T
C	G	T	A	C
G	C	A	T	G
T	A	G	C	T

R7

	A	C	G	T
A	A	G	C	T
C	C	A	T	G
G	C	A	T	G
T	A	G	C	T

R8

Algortihm for reverse substitution

```
substituted = [ ]
for loop i starting from length of enImage to 0:
   lst = [ ]
   for j starting from length of encodedImage to 0:
      dM = DSM [ ( I * len(encodedImage[0]) ) + j ]
      rule = RM [e][h]
      dI = getRevRule(rule, dM, encodedImage[e][h])
      lst.append(dI)
   substituted.append( lst )
```

The algorithm will require to use DSM map and rule map generated at the time of decryption with the same key and same chaotic maps.

$$sred = revSubstitution(ered, DSM, RM)$$
$$sgreen = revSubstitution(egreen, DSM, RM)$$
$$sblue = revSubstitution(eblue, DSM, RM)$$

d) *DNA Decoding*

The decoding process has the same algorithm as we have used in encryption process defined in paper[9].

For further steps of decryption we reverse the operator $\otimes$ in decryption algorithm as

$$(a \otimes^{R} b) = \big((a+1) * modinv(b+1, p)\big) mod\ p - 1$$

e) *Reverse Spiral Mixing*

In reverse mixing we apply mixing in reverse fashion. Rev mixing have same four steps but in reverse order.

- Backward Spiral column mixing
  In the process of decryption we move from last operation towards first.

Algorithm For Reverse Backward Spiral Column

```
for each column h from 0 to n-1:
  if h is even
    for each row e from m-1 to 1
        ig[e,h] =operate  ig[e,h] ⊗^R ig[e-1,h]
    if j < n-1:  ig[e-1,h] = operate ig[e-1,h] ⊗^R ig[i-1,j+1]
  else (if h is odd)
    for each row e from 0 to m-2
        ig[e,h] = operate ig[e][h] ⊗^R ig[e+1][h]
    if j < n-2: ig[e+1,h] =operate ig[e+1,h] ⊗^R ig[i+1,j+1]
```

- Backward Spiral row mixing

Algorithm for Reverse Backward Spiral Row

```
for each row e from 0 to m-1:
  if e is even:
     for each column h from n-1 to 1:
        ig[e,h] = operate ig[e,h] ⊗^R  ig[e,h-1]
     if i<m-1: ig[e,h-1] = operate ig[e,h-1] ⊗^R ig[e+1,h-1]
  else (if e is odd)
     for each column h from 0 to n-2
        img[e,h] = operate ig[e,h] ⊗^R ig[e,h+1]
    if i < m-1:  ig[e,h+1] = operate ig[e,h+1] ⊗^R ig[e+1,h+1]
```

- Reverse Forward Spiral column mixing

Algorithm for Reverse Forward Spiral Column

```
For each column h from n-1 to 0:
  if h is even:
    For each row e from m-1 to 1:
        ig[e,h] =operate ig[e,h] ⊗^R  ig[e-1,h]
    if j > 0, ig[e-1,h] = operate ig[e-1,h] ⊗^R  ig[e-1,h-1]
  Else (if h is odd):
    For each row e from 0 to m-2:
     ig[e,h] = operate ig[i,j] ⊗^R ig[e+1,h]
    If j > 0, ig[e+1,h] =operate ig[e+1,h] ⊗^R ig[i+1,j-1]
```

- Reverse Forward Spiral row mixing

Algorithm for Reverse Forward Spiral Row

```
for each row e from m-1 to 0:
  If e is even
    For each column h from n-1 to 1
        ig[e,h] = operate ig[e,h] ⊗^R ig[e,h-1],
    If i ≠ 0, ig[e,h-1] =operate ig[e,h-1] ⊗^R ig[e-1,h-1],
  else (if e is odd):
    For each column h from 0 to n-2
        ig[e,h] = operate ig[e,h] ⊗^R ig[e,h+1]
    If i ≠ 0: ig[e,h+1] = operate ig[e,h+1] ⊗^R ig[i-1,j+1]
```

f) *Reverse Modified ACM*

In reverse modified ACM we just reverse the loops of ACM algorithm.

g) *Reverse Inter channel mixing and Mix Rows*

The rows of the images obtained after applying reverse modified ACM are treated with the same IV and reverse of operator ⊗.

Algorithm for reverse mixrows

```
For s starting from m-1 to 0:
      for t starting from n-1 to 1:
         Dimg[s][t] = image[s][t] ⊗^R image[s][t-1]
      Dimg[s][0] = image[s][0] ⊗^R  IV[t]
```

After applying mix rows, the channel are reversly diversed to get the original r, g, b matrices. We defined inter mix equations in such a way that they can be reversed easily. Below equations shows the reverse procedure. For reverse intermix we start with getting B channel.

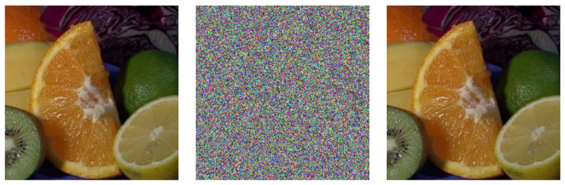
$$B = B' \otimes^{R} (R' \otimes G')$$
$$G = G' \otimes^{R} (R' \otimes B)$$
$$R = R' \otimes^{R} (G \otimes B)$$

Algorithm for Reverse InterMix

```
revInterMix(r, g, b):
  m= length of r
  n= length of first row
  Imxr = [][]
  brack = g will operate with b using ⊗
  imixr = r will operate with brack using ⊗^R
  return imixr
```

h) Merge the decrypted R, G, B matrices to get final decrypted image.

2.4 The Final Results

a) 

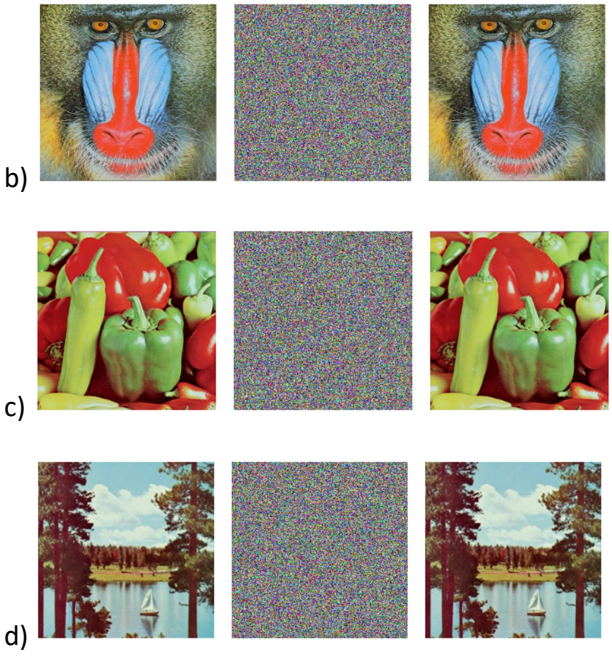


Figure 1. Original, en-crypted & de-crypted image of a) Fruit b) baboon c) Pepper d) Lake

3. Experimental Results and Analysis

Numerous colour images were encrypted, and the cipher image's characteristics were examined in order to evaluate the cryptosystem's performance.

3.1 Space and Sensitivity of key

The encryption's resistance to brute-force attacks is tested by the length of a key. In aspect of cryptography, a high degree of security requires a key length that is greater than 2^{100}. In our proposed cryptosystem, The key used has a secured key space of 2^{240} and is 240 bits long. The key used in encryption and decryption is same because it is symmetric.

High sensitivity to the key is a fundamental requirement for the encryption system; otherwise, it will not be able to obtain the right decryption image, even in the event that there is a very slight shift in the input key. A small modification to the initial key will result in a unseen new decryption image for a perfect encryption scheme. In our experiment, we each alter four RGB colour images by merely varying the lone bit of the initial key. The decryption outcomes are displayed, along with the notable variations between the original (256 x 256) and decrypted images.[3]

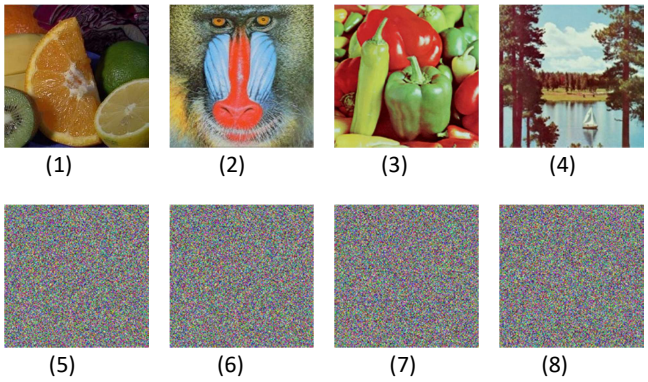


Fig 2. The results of sensitivity of a key of (1) Fruit, (2) Baboon, (3) pepper and (4) Lake. (5) to (8) are decrypted images of respective color images.

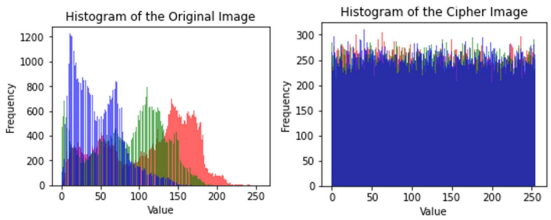
3.2 Histogram Analysis

A histogram describes how the image's pixel intensity values are distributed. The histograms of the plain and encrypted images are present in Figure 3. It is evident that the cypher image's pixel values are consistently distributed throughout the interval [0, 255], in stark contrast to the plain image's pixel values which follows uneven trend. Single image shows the histogram for all three channels. Additionally, we use histogram variances to statistically test the regularity of histograms. Mathematically, histogram variances are defined as :

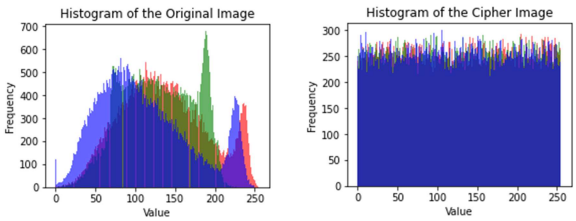
$$\text{variance}(\mathrm{I}) = \frac{\sum_{x=1}^{m}\sum_{y=1}^{n}(\mathrm{I}(x,y) - \mu)^2}{m * n}$$

Colour images have low variance since they only have eight bits per pixel and have a non-uniform distribution of pixels in each R, G, and B component image. Table 5 displays natural images with variances less than 3000, while the matching cipher images of the corresponding Re, Gr, and Bl components have variances greater than 5400.

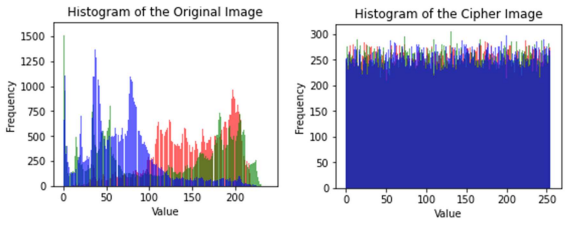
1.



2.



3.



4.

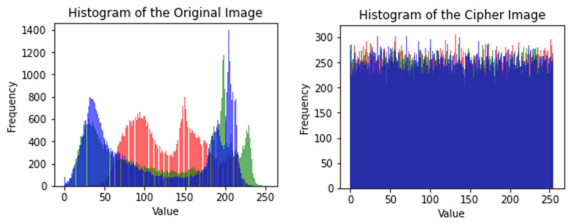


Figure 3. 1) plain and cipher histogram of Fruit **2**) plain and cipher histogram of baboon **3**) plain and cipher histogram of pepper **4**) plain and cipher histogram of lake.

Histogram of color images are having different graphs and bins of different sizes of all three color channels but cipher image have all three channel equally distributed on the graph.

3.3 Chi-Square Analysis

The encrypted image's histogram offers a visual depiction of the data, while the chi-Square test offers a statistical analysis of the variation of pixel intensity readings. The chi^2 readings for an image of size $m \times n$ and pixel depth p bits can be computed as [15]

$$\chi^2 = \sum_{L=0}^{2^p - 1} \frac{(x^o_L - x_e)^2}{x_e}$$

where x^o_L is the frequency that has been observed for the Lth intensity, where L is between 0→ 2^p− 1, and x_e is the frequency expected, which has been computed as:

$$x_e = 2^p - 1/(m * n)$$

The table below displays the values for $\chi 2$ for each of the cipher images.

Table 4: chi-Square of cipher images

Image	Chi-Square		
	Cipher Image		
	Red	Green	Blue
Fruit	258.47	267.02	268.61
Baboon	232.32	268.43	267.44
Pepper	246.94	216.14	246.22
Lake	257.71	235.82	272.62

Better pixel intensities are distributed when $\chi 2$ is smaller. From the $\chi 2$-Inverse Cumulative Distribution Function [12], we can determine the crucial values (upper limit) of $\chi 2$ for 8-bit pictures with level of significances α = 20. These values are 273.79. Since the produced cipher images, $\chi 2$ values are smaller than the crucial values, The hypothesis holds for coming in the range of 20% significance, indicating a uniform variation of pixel magnitude across all of them.

3.3 Entropy Analysis

The uncertainty measurement is provided by entropy. The entropy $\varepsilon(S)$ is calculated for a source S that contains n symbols, each with a chance of presence of pi as[16]

$$\varepsilon(S) = \sum_{i=1}^{n} \mathrm{pi} * \log\left(\frac{1}{\mathrm{pi}}\right)$$

Entropy of plain images is low, but that of cipher images should be close to 7 high bits/symbol. Table ahead demonstrates that the crypted images produced by the suggested technique have an entropy larger than 7.9950 , indicating a high degree of randomness.

3.4 Correlation Analysis

Pixels have a strong correlation in plain images, however in crypted images, this correlation should be reduced. One among key statistical features of images is the correlation between neighbouring pixels, which indicates the limit of correlation between pixel values in adjacent positions in the image. The greater the efficiency of the developed encryption system, there is low correlation between the neighbouring pixels of the encrypted picture. Next, 10000 pixels are chosen from both the encrypted and original images in order to examine the pixel correlation. The coefficient of correlation ρ is calculated as [17]

$$\rho(x,y) = \frac{(\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Consider two vectors of size n, x and y, respectively, with means $\bar{x}$ and $\bar{y}$. -1 ≤ μ ≤ 1. The original images have a high positive correlation when the values are close to 1, while the cipher images benefit from a low correlation when the values are close to 0.

Table 5: Variances of plain and cipher images						
Image	Plain Image			Cipher Image		
	Red	Green	Blue	Red	Green	Blue
Fruit	3090.122	3090.257	3909.811	5453.239	5465.818	5475.349
Baboon	2889.991	1972.825	3383.311	5459.752	5444.902	5457.723
Pepper	1995.873	2556.325	1881.152	5469.795	5444.531	5485.011
Lake	1954.858	2572.911	2548.812	5469.953	5498.777	5421.363

Table 6: Entropy of plain and cipher images						
Image	Entropy					
	Plain Image			Cipher Image		
	Red	Green	Blue	Red	Green	Blue
Fruit	7.552	7.337	6.745	7.9971	7.9968	7.9974
Baboon	7.683	7.381	7.682	7.9972	7.9973	7.9971
Pepper	7.271	7.527	7.109	7.9972	7.9975	7.9972
Lake	7.331	7.626	7.341	7.9971	7.9974	7.9973

3.5 NPCR and UACI

N: Number of P: Pixel C: Chaging R: Rate and U: Unified A: Averaged C: Changing R: Rate are crucial metrics that assess a cryptosystem's resilience to differential attacks. Let X and X′ represent the *c*ipher images that are generated following the *e*ncryption of a plain *i*mage of $m * n$ size and a pixel depth of p i.e. X, and modifying one value of pixel of the initial image respectively and encrypting again is X′. The N.P.C.R. and U.A.C.I. are computed using the following formulas..

$$D(u,v) = \begin{cases} 1, & if\ X(u,v) \neq X'(u,v) \\ 0, & if\ X(u,v) = X'(u,v) \end{cases}$$

$$NPCR:\ N(X,X') = \sum_{o,p} \frac{D(u,v)}{m*n} * 100\%$$

$$UACI:\ U(X,X') = \sum_{o,p} \frac{|X(u,v) - X'(u,v)|}{2^p - 1 \cdot (m*n)}$$

where u = 1, 2, …, m; v = 1, 2, …, n & |t| denotes the absolute value of t.

In [13], the optimal settings for UACI and NPCR have been examined. The greatest value of Fmax for a picture with dimensions of m * n and bit depth of p will be F = 2^p – 1. The table ahead shows N.P.C.R and U.A.C.I values observed for different images

Table 7: NPCR of cipher images			
Images	NPCR		
	NPCR critical value = 99.534077		
	Red	Green	Blue
Fruit	99.6386	99.6118	99.5724
Baboon	99.6062	99.6062	99.6154
Pepper	99.5971	99.5986	99.6121
Lake	99.5994	99.5986	99.6185

Table 8: UACI of cipher images			
Images	UACI		
	UACI range: 33.1593 - 33.7676, α = 0.005		
	Red	Green	Blue
Fruit	33.3419	33.4274	33.4146
Baboon	33.3902	33.4853	33.2344
Pepper	33.4408	33.4035	33.4181
Lake	33.4693	33.3303	33.4835

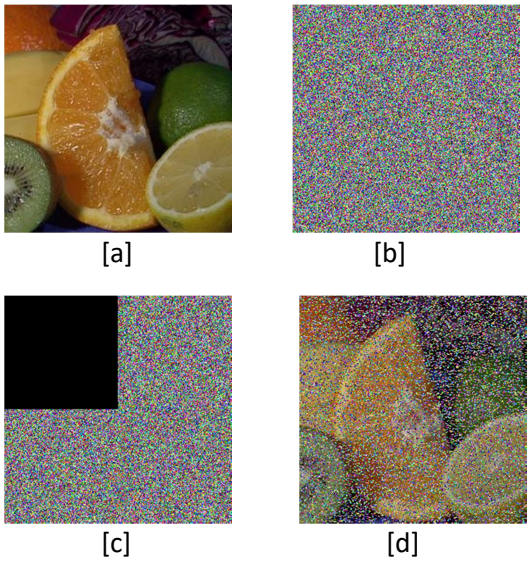
*3.6 **P**eak **S**ignal to **N**oise **R**atio*

Peak **S**ignal 2 **N**oise **R**atio (P.S.N.R) is a useful metric for evaluating the effectiveness of a cryptosystem against noise attacks and occlusion. Let Pl be the plain image that is encrypted to create the crypted image Z = En(Pl, key) using a key and an encryption

technique En. The crypted image is subjected to noise attack or occlusion to produce the image Z′. This image is then decrypted using the matching decoding algorithm Dc and the same key to provide the decrypted *i*mage D = Dc(Z′, key). The PSNR is defined in Equation as follows[3,7] if the image has dimensions of m * n and bit depth of p.

$$P.S.N.R = 10 * \log_{10}\left[\frac{(2^p - 1)^2}{\frac{1}{m*n}\sum_{u=1}^{m}\sum_{v=1}^{n}\{D(u,v) - \rho(u,v)\}^2}\right]$$

An image needs a high PSNR—ideally more than 30dB— to be visually discernible. However, because of their random noise like properties and high mean squared error in the associated pixels, cipher images have low PSNR values—typically less than 10dB.



[a] [b] [c] [d]

Figure-4. Occlusion attack – [a] Original Fruit image, [b] encrypted image, [c] occlusion attack on encrypted image, and [d] decrypted image.

Table 9: PSNR of cipher images

Images	PSNR		
	Red	Green	Blue
Fruit	8.23	8.25	8.21
Baboon	8.76	8.76	8.72
Pepper	8.84	8.76	8.76
Lake	8.81	8.81	8.82

4. Comparative Analysis

Using the same experimental setup and cryptosystem, the operator ⊗ defined in paper[9] has been compared with other operators such as X⊕R and (+/−) on several color images. The NPCR and UACI tests demonstrate that the operator ⊗ outperforms X⊕R in terms of resistance to differential attacks, since the operator ⊗ yields satisfactory results whereas XOR fails horribly in these tests using with the suggested cryptosystem. While the (+,−) operator yields fairly similar values for UACI and NPCR but also fails in some circumstances as well like for one channel fails and for other two channel give satisfactory result.

However, the main disadvantage of (+/−) operator is removing the pixel correlation and the suggested operator works significantly better in these situations. These findings support the assertion that the operator ⊗ outperforms the standard operators typically employed in traditional image cryptosystems. Table 11 alse provides an overview of various existing cryptosystem's performances and compare our values with theirs values.

Table 10 : Pixel correlation values between plain and cipher images.

Image	Direction	Pixel correlation coefficient					
		Plain Images			Cipher Images		
		Red	Green	Blue	Red	Green	Blue
Fruit	Horizontal	0.9844	0.9879	0.9545	-0.0018	-0.0067	-0.0064
	Vertical	0.9856	0.9735	0.9857	0.0075	0.0018	-0.0077
	Diagonal	0.9433	0.9268	0.9402	0.0055	-0.0168	-0.0071
Baboon	Horizontal	0.9126	0.8962	0.9531	0.0031	0.0084	-0.0169
	Vertical	0.6595	0.7867	0.8541	0.0064	-0.0076	-0.0852
	Diagonal	0.7652	0.6806	0.8443	-0.0384	0.0239	0.0259
Pepper	Horizontal	0.9285	0.9465	0.9486	-0.0042	0.0092	-0.0086
	Vertical	0.9725	0.9772	0.9784	-0.0053	0.0041	0.0045
	Diagonal	0.9119	0.9455	0.8801	-0.0345	-0.0127	-0.0057

Lake	Horizontal	0.9322	0.9589	0.9647	0.0042	0.0082	0.0062
	Vertical	0.9794	0.9868	0.9863	-0.0085	-0.0063	0.0058
	Diagonal	0.8935	0.9496	0.9458	-0.002	0.0074	-0.0069

Table 11: Comparative Analysis

Image	Cryptosystem	Encryption Process	Entropy			NPCR			UACI		
			Re	Gr	Bl	Re	Gr	Bl	Re	Gr	Bl
Rgb color image		Original	7.268	7.597	6.971	~	~	~	~	~	~
rgb color image	proposed	using ⊗	7.997	7.996	7.997	99.638	99.618	99.572	33.342	33.427	33.414
		using XOR	7.997	7.996	7.996	51.416	51.416	51.428	18.39	18.38	18.38
		using + -	7.997	7.997	7.996	93.432	98.182	99.424	33.401	33.421	33.405
rgb color image	Refr [3]	Dynamic DNA encryption and chaos	7.997	7.996	7.997	99.6	99.6	99.61	33.56	33.46	33.45
rgb color image	Refr [7]	fractional-order hyperchaotic system and DNA-Coding	7.997	7.997	7.996	99.6	99.62	99.58	33.47	33.44	33.57
rgb color image	Refr [8]	fractional fourier transform and DNA sequence operation	7.997	7.997	7.999	99.56	99.55	99.57	33.41	33.44	33.45
rgb color Image	Refr [4]	hybrid chaotic system and DNA sequences	7.997	7.997	7.997	99.64	99.6	99.61	33.47	33.36	33.44

The values that fail the crucial test are shown by underlining in the NPCR and UAC

Color images can be successfully encrypted using the proposed cryptosystem and the operator ⊗ as we have proposed as well as plain images as shown in paper [8] . The comparison of the findings shows that the proposed cryptosystem performs comparably to the the most modern cryptosystems currently in use.

5. Conclusion

In our paper, we design a cryptosystem using a ⊗ operator for color images. The cryptosystem follows the properties of operator ⊗ which says it forms an abelian group due to which it is used for encryption as well as decryption. This operation's application is shown in a suggested cryptosystem that includes DNA encoding, substitution, and decoding after a unique mixing operation that makes advantage of the distributive and commutative features of ⊗. The cryptosystem also demonstrate decryption process and algortihms where the changes are required for decryption. In contrast to the existing cryptosystems, which employ intricate encryption techniques or a number of rounds for confusion-diffusion, the proposed cryptosystem just requires basic computational operations. Analysis and experimental findings demonstrate that the suggested work satisfies the specifications needed for a safe images cryptosystem. Comparative analysis with other commonly used operators with color images shows

that operator ⊗ outperforms them regarding color images encryption.

6. References

[1] W. Cao, Y. Mao, Y. Zhou, Designing a 2D infinite collapse map for image encryption, Signal Process. (2020) 107457, http://dx.doi.org/10.1016/j.sigpro. 2020.107457

[2] Q. Zhang, L. Guo, X. Wei, Image encryption using DNA addition combining with chaotic maps, Math. Comput. Modelling 52 (11–12) (2010) 2028–2035, http://dx.doi.org/10.1016/j.mcm.2010.06.005.

[3] X. Chai, X. Fu, Z. Gan, Y. Lu, Y. Chen, A color image cryptosystem based on dynamic DNA encryption and chaos, Signal Process. 155 (2019) 44–62, https://doi.org/10. 1016/j.sigpro.2018.09.029.

[4] Abolfazl Yaghouti Niyat & Mohammad Hossein Moattar, Color image encryption based on hybrid chaotic system and DNA sequences https://link.springer.com/article/10.1007/s11042-019-08247-z

[5] X. Chai, J. Bi, Z. Gan, X. Liu, Y. Zhang, Y. Chen, Color image compression and encryption scheme based on compressive sensing and double random encryption strategy, Signal Process. 176 (2020) 107684, http://dx.doi.org/10.1016/j.sigpro. 2020.107684.

[6] Z. Liu, C. Wu, J. Wang, Y. Hu, A color image encryption using dynamic DNA and 4-D memristive hyper-chaos, IEEE Access 7 (2019)78367–78378, https://doi.org/10.1109/ACCESS.2019.2922376.

[7] H. Dong, E. Bai, X.-Q. Jiang, Y. Wu, Color image compression-encryption using fractional-order hyperchaotic system and DNA coding, IEEE Access 8 (2020) 163524– 163540, https://doi.org/10.1109/ACCESS.2020.30223 98.

[8] M.A.B. Farah, R. Guesmi, A. Kachouri, M. Samet, A novel chaos based optical image encryption using fractional Fourier transform and DNA sequence operation, Opt. Laser Technol. 121 (2020) 105777.

[9] P. Mishra, C. Bhaya, A.K. Pal, A.K. Singh, A novel binary operator for designing medical and natural image cryptosystems, Signal Process. Image Commun. 98 (2021) 116377. https://doi.org/10.1016/j.image.2021.116377

[10] G. Petersen, Arnold cat map survey, Math 45 linear algebra, http://refhub.elsevier.com/S0923-5965(21)00173-9/sb20

[11] Yang, C., Wei, X. and Wang, C., 2021. S-Box design based on 2D multiple collapse chaotic map and their application in image encryption. *Entropy*, *23*(10),p.1312.https://www.mdpi.com/1099-4300/23/10/1312

[12] chi-square inverse cumulative distribution function - MATLAB chi2inv – Math Works India, 2021 ,https://in.mathworks.com/help/stats/chi2inv.html (Accessed 13 May 2021)

[13]] Y. Wu, J.P. Noonan, S. Agaian, et al., NPCR and UACI randomness tests for image encryption, Cyber J.: Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT) 1 (2) (2011) 31–38. http://refhub.elsevier.com/S0923-5965(21)00173-9/sb26

[14] X. Chai, H. Wu, Z. Gan, D. Han, Y. Zhang, Y. Chen, An efficient approach for encrypting double color images into a visually meaningful cipher image using 2D compressive sensing, Inform. Sci. 556 (2021) 305–340, http://dx.doi.org/10. 1016/j.ins.2020.10.007.

[15] X. Wang, L. Feng, H. Zhao, Fast image encryption algorithm based on parallel computing system, Inform. Sci. 486 (2019) 340–358, http://dx.doi.org/10.1016/ j.ins.2019.02.049.

[16]] X. Chai, H. Wu, Z. Gan, D. Han, Y. Zhang, Y. Chen, An efficient approach for encrypting double color images into a visually meaningful cipher image using 2D compressive sensing, Inform. Sci. 556 (2021) 305–340, http://dx.doi.org/10. 1016/j.ins.2020.10.007.

[17] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise Reduction in Speech Processing, Springer, 2009, pp. 1–4, http://dx.doi.org/10. 1007/978-3-642-00296-0_5