A FRACTAL BASED IMAGE CIPHER USING KNUTH SHUFFLE METHOD AND DYNAMIC DIFFUSION

Shafali Agarwal

Plano, Texas 75025, USA

ABSTRACT

This paper proposes a fractal-based image encryption algorithm which follows permutation-substitution structure to maintain confusion and diffusion properties. The scheme consists of three phases: key generation process; pixel permutation using the Knuth shuffle method; and the dynamic diffusion of scrambled image. A burning ship fractal function is employed to generate a secret key sequence which is further scanned using the Hilbert transformation method to increase the randomness. The chaotic behavior of the fractal strengthens the key sensitivity towards its initial condition. In the permutation phase, the Knuth shuffle method is applied to a noisy plain image to change the index value of each pixel. To substitute the pixel values, a dynamic diffusion is suggested in which each scrambled pixel change its value by using the current key pixel and the previously ciphered image pixel. To enhance the security of the cryptosystem, the secret key is also modified at each encryption step by performing algebraic transformations. The visual and numerical analysis demonstrates that the proposed scheme is reliable to secure transmission of gray as well as color images.

KEYWORDS

Burning ship fractal, Knuth shuffle method, Image encryption, Hilbert transformation, dynamic diffusion

1. INTRODUCTION

A cryptographic system plays an important role to achieve the requirement of a secure system to transfer the encrypted data over the unsecured network. The system has two major phases, i.e. secret key generation to encrypt/decrypt the data/image/audio/video and the other is the encryption/decryption process.

With the continuous development in the digital world, conventional ciphers with small key space have become highly prone to be affected by a brute - force attack. Designing an image encryption algorithm with suitable keyspace and high key sensitivity is inevitably a challenging job in this computer world. An image has various crucial parameters to be considered while developing an encryption algorithm such as image size and a high correlation value between its adjacent pixels which can be used in cryptanalysis.

A fractal function exhibits randomness behavior and highly sensitive to its initial condition, suitable to design a secure cryptosystem. A fractal image can be defined as a fragmented geometric shape which on split gives an approximate reduced copy of the whole [1]. It is generated by iterating a mathematical function for a finite number of times. A fractal image possesses high variation in detail at discrete scales, also determines a wide choice in terms of key space. Few real-life examples are cauliflower, coastlines, clouds, tree, etc. A well-known fractal introduced by Benoit Mandelbrot, in 1979, a very complex & perturbed structure that is known as

DOI: 10.5121/ijcnc.2019.11405

Mandelbrot set[2]. To enhance the randomness to the key generation process, a pseudo-random number generator could be used with the fractal function. Nowadays, random number sequences are used in various fields like traffic simulator, gambling, cryptography and, also in other areas in which unpredictable behavior is desired. There are lots of approaches followed by the scientists to generate these numbers such as a physical method (coin flipping), computational methods (pseudo-random number generator algorithm), using the probability density function and of course by human as well.

The rest of the paper is organized as follows: section 2 starts with discussion of the related work. In section 3, the methodologies like burning ship fractal, Hilbert transform, and Knuth shuffle method are introduced. The next section describes the proposed encryption/decryption scheme in detail. In section 5, simulation and performance analysis results are presented to verify the efficiency of the given method. At last, section 6 concludes the paper.

2. LITERATURE REVIEW

A project was carried out in 2003 to encrypt a message with the help of random numbers and Mandelbrot set fractal function. The author succeeded to encrypt the data, but was unable to design a perfect decoder for the same[3]. Later, In 2004, USA navy experts identified the importance of a fractal function in generating a secret key and published the patent for the same[4]. A complex and fractional fractal geometry helps to enhance the complexity of the cryptosystem design. A new approach to encryption using fractal geometry by generating a fractal using some initial parameters and subsequently encrypt a predetermined length of the message by using fractal orbits to corresponding alphabet mapping[5]. A Mandelbrot function and Julia set function has a strong relation between them as the Mandelbrot function is a set of points in complex c-plane starting at z=0 whereas Julia set is an image for a fixed c value starting non-zero z[6]. The author [7] generated a set of the public and private key and designed a public key cryptosystem by utilizing the connection between them. While in[8], a symmetric key encryption algorithm was proposed by generating a security key using Mandelbrot function only. Later, the author pointed out the weaknesses of [8] by analyzing the results of a chosen-plaintext attack, chosen ciphertext attack and known plaintext attack under the assumption of direct use of plain text. Hence, an improved method was proposed in which Arnold map was used to shuffle the plain image pixel before executing modulo operation using fractal key[9]. In the same way, Suthikshn[10] also designed a cryptosystem using the Mandelbrot set and encrypted the message using RSA. Further, to improve key sensitivity, a Hilbert curve was embedded with the Mandelbrot set and Julia set to design a secret key[11], [12]. SiavashSattari et al. [13]developed a cryptosystem in which two rounds of encryption/decryption were executed using a fractal based secret key and chaotic map function respectively. The Chaotic map always enhanced the randomness effect while mixing with fractal function and the advantage has been taken by the author while using fractal key along with a two-dimensional logistic map to design a cryptosystem[14]. The author utilized a DNA sequence as a keystream to permute and diffuse the plain image pixels[15]. A multiphase symmetric key encryption algorithm was proposed by the author using finite field cosine transformation (FFCT) in which a fractal was used as a source of the one-time-pad keystream, provides a secure cryptosystem[16]. A cryptosystem will be relatively more secure if a set of different keys is used to encrypt the plain image on each iteration[17]. In[18], multiple fractal images were used to generate keystream. The method showed an improved performance by adding several parameters: feedback delay, multiplexing, and independent horizontal and vertical shifts. Similarly, in[19], multiple fractal images contributed towards the pseudorandom keystream generation using a non-linear network with a delay block to make more randomize output stream.

Although, because of the high variation in detail and global irregularity, the fractal images also employed in the second phase of a cryptosystem design i.e. encryption/decryption process[20]–[22]. A fractal function can be used in conjunction with the chaotic map to introduce more randomness and butterfly effect. An idea has been suggested by the author in which the initial values provided to the 2D composite chaotic map by iterating a fractal function to get its fixed point[20]. In[21], a fractal seed is obtained by hashing the entered key to introduce the confusion about the key size. The plain data was XORed with the generated fractal to get ciphered data file. A fractal dictionary encoding method was used to reconstruct a good quality image from a compressed cipher the image in a stream cipher encryption algorithm [22]. The large size of the image leads to the evolution of compression followed by encryption[23].

There are other means also suggested to generating a key sequence such as chaotic maps[24], biometric images[25], and DNA [26] sequences. A hierarchical combination of three maps was utilized to generate an n-ary keystream which was proved a secure sequence by executing keystream distribution, information entropy and sensitivity to initial condition parameter[24]. The author in [25] suggested an external biometric key of the length of 256 bits which in turn manipulated to calculate the seed values of logistic map and tent map. A DNA base vectors were mathematically processed to generate the symmetric cryptographic key(s) by applying linear computation[26]. Ahyperchaotic map because of its complex dynamic characteristics and improved key sensitivity is utilized to design a cryptosystem[27]–[31]. In [27], the plain image plays an important role to generate a secret key followed by pixel level and bit level permutation to strengthen the security of the proposed cryptosystem. The authors cryptanalyzed the approach given in [32]and found that the plain image can be recovered under chosen-plaintext attack. Further, the scheme has improvised by including the scrambling procedure followed by modifying plain image pixel using the keystream code extracted from the previous chaotic sequence [28]. A hyperchaotic Lu system and a logistic map are employed to generate the key stream. The cryptosystem design commonly consists of two phases: permutation and substitution introduce confusion and diffusion between the image pixels. The author in [29] executed pixel permutation by using pixel-swapping technique and the substitution by generating a plain text dependent key sequence using the logistic map. A chaotic sequence is generated using 5-D hyperchaotic system to get a more complex dynamic sequence[30]. Later on, DNA encoding, DNA XOR operation, and DNA complimentary rule have applied to get the cipher image. In [31], a breadth-first search algorithm and in [33]the Josephus traversing method has adopted to permute the image pixels. The Josephus problem was further used by the author to shuffle the image pixels followed by a randomly generated image filter to diffuse scrambled image[34].
A new phenomenon was introduced by inserting a group of random numbers as a one-time pad to the shuffled image. Further, the process executed as a combination of permutation-insertion-diffusion method[35]. Theoretically, 1D chaotic map is not considered secure while using in image encryption applications. However, the complexity of 1D chaotic maps was improvised by proposing cosine transform based chaotic systems[36]. Also, combining 1D chaotic maps give a larger trajectory and better sensitivity towards its initial condition [37], [38]. The author in [39] utilized the chaotic properties of the 3D Rabinovich-Fabrikant Equations to confuse the image pixels and also diffuse the confused pixels by applying MOD and bitXOR operation using chaotic sequence generated by the map. ZhenjunTang et al. in [40]exploited double spiral scanning with he map and Lu map to design a hybrid image encryption scheme. The pixels scrambling process was executed using a double spiral scan under the control of Henon chaotic map. Further, a secret matrix was generated for XOR operation using a 3D discrete Lu chaotic map and obtained the cipher image.

3. THE METHODOLOGIES

3.1 Burning Ship Fractal

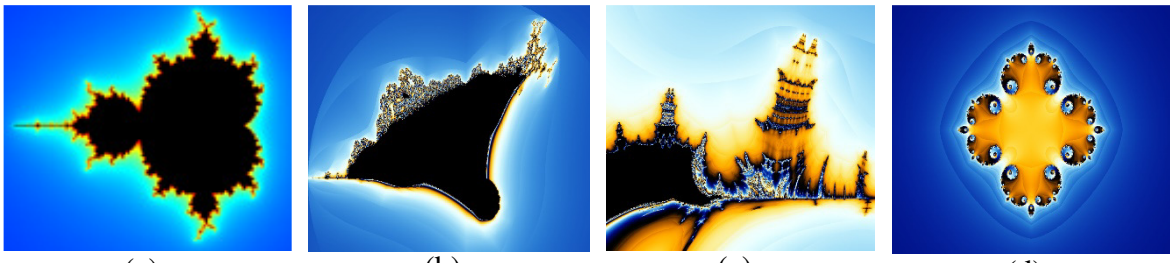
The Mandelbrot set function can produce various distinguish fractal images after doing minute change in the base function. Many researchers studied the various format of Mandelbrot set function and generated beautiful images consecutively[41], [42]. In 1992, Michelitsch and Rossler has modified the standard Mandelbrot set function by considering the absolute value of the complex variable and got the completely new fractal images[43]. Appearances of obtained images are incredibly beautiful and different from the previous fractal images like resembling a ship going into flame i.e. known as Burning ship fractal. The function to generate the burning ship fractal is as:

$$z_{n+1} = (|Re(z_n)| + i|Im(z_n)|)^2 + c \quad (1)$$

The real and imaginary components of the complex quadratic equation can be calculated as:

$$x_{n+1} = x_n^2 - y_n^2 - c_x \text{ and } y_{n+1} = 2x_n y_n - c_y \quad (2)$$

For x_0 = 0 and y_0 = 0, the above equations yield the fractal images in the c-plane (parameter space) which will either escape or remain bounded. A Mandelbrot set function does not count the absolute part of the used complex variable, whereas the given burning ship fractal function considered its absolute form in terms of its real and imaginary components before squaring. Therefore, both functions generate relatively different fractal images from each other. As can be seen in figure 1 that, the Mandelbrot set function focused on classical beauty, and ornate scrollwork whereas the burning ship function contains beautiful patterns look like a paw print, tokens, and towers. The author applied the Mann iteration and Ishikawa iteration method to generate magnificent images of burning ship fractal function[44]. It is a kind of escape time fractal in which escape time to infinity from the covered area is measured in steps (Iterations). Later, the convergence rate of the Mann iterated function proved the stabilization in less number of iterations[45].

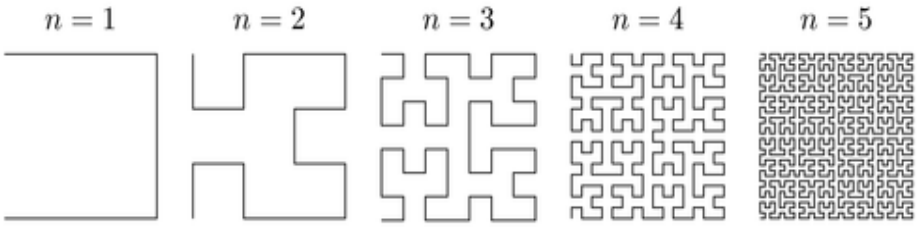


(a) (b) (c) (d)

Figure 1. Fractal images (a) Mandelbrot set (b) Burning ship (c) Mann iterated burning ship (d) Mann iterated Julia set

3.2 Hilbert Transformation

Image pixels are closely related to each other which arises an opportunity to a hacker to identify the whole image by guessing a part of it. A Hilbert curve scanning is used to rearrange the image pixels in a predefined order. The process starts scanning of the pixels from one end (top left, top right, bottom left or bottom right) and ends after tracing each and every pixel of that image[46]. The key idea is to scramble the image pixels to reduce the adjacent correlation and convert a 2D array into a 1D array after scanning. A sample structure of the Hilbert scrambling process with an example of the 8*8 matrix is depicted in the given figure 2:



(a)

143	234	76	173	127	70	222	209
94	148	81	132	75	167	153	114
100	142	180	183	37	87	185	156
61	97	82	118	87	185	110	97
88	83	95	181	161	151	98	75
55	45	64	79	108	215	123	70
63	19	37	112	15	142	160	103
51	15	30	25	52	21	45	92

(b)

143	180	127	185	161	160	25	45
234	82	75	156	108	103	30	83
148	118	167	97	215	92	37	88
94	183	70	110	151	45	112	55
100	132	222	185	98	21	79	63
61	81	209	87	75	142	181	19
97	76	114	37	70	15	95	15
142	173	153	87	123	52	64	51

(c)

Figure 2. (a) Hilbert scrambling pattern; An example of Hilbert transformation using 8*8 matrix (b) Original Matrix (c) Transformed Matrix

3.3 Adding noise to plain image

The idea of adding noise to a plain image is to increase the randomness of the cipher process. A noise field can be embedded either uniformly or exponentially. In this paper, a uniform random noise is added to the plain image. This step helps to produce two distinct cipher images while encrypting the same noisy plain image using the same secret key. However, adding noise does not affect the image visual quality from the human visual view point. Still, the difference between two noisy images leads to completely change cipher images.

3.4 Knuth shuffle method

The confusion property breaks the strong correlation between adjacent image pixels by randomly scrambling the pixel positions[47]. The Knuth shuffling method is given by Donald E. Knuth and is an in-place algorithm used to randomly permute a finite sequence. The algorithm starts by picking a number randomly and exchange it with the indexed number in the same array that has not been selected. The same step repeats until no remaining unshuffled number in the array.

Algorithm 1: Knuth shuffle algorithm

Input: An array *arr* = {A, B, C, D, E}
for i = 2 to *n* do
a) select w randomly such that $1 \leq w \leq i$
b) Exchange arr(i) and arr(w)
End for
Output: A shuffled array *arr* (in-place shuffling)

Example:Start with the given array:

{A, B, C, D, E}
{A, **B**, C, D, E} => {A, **B**,C, D, E}, for w=2
{**A**, B, **C**, D, E} => {**C**, B, **A**, D, E}, for w=1
{C, B,**A**, **D**, E} => {C, B, **D**,**A**,E}, for w=3
{**C**, B,D, A, **E**} => {**E**, B, D, A, **C**}, for w=1

This is our resulting permutated array, i.e. {E, B, D, A, C}. The algorithm used a single array to permute the number sequence. Hence, the execution time is proportional to the *n* number of elements being shuffled in the array $O(n)$.

4. THE PROPOSED CRYPTOSYSTEM

This section discussed the proposed permutation-substitution based image cryptosystem which utilized dynamic and complex burning ship fractal function to generate a secret key sequence. The method composed of three steps mainly: 1).secret key generation using burning ship fractal, 2). encrypting noisy plain image: applying Knuth scrambling method followed by image diffusion using a secret key, and 3).Image decryption by executing all previous steps in reverse order. The given method is quite fast as the whole process is executed only once and still exhibits significant performance from the security perspective. As a result, a cipher image is obtained which can be securely transmitted to the receiver. The detail description of each process is as follows:

4.1 Key Generation Process

The proposed key generation scheme works on block operations using a burning ship fractal image and an external key depends on the size of the fractal image to be used in the process. Figure 3 shows the block diagram to represent the flow of operations to get a security key. Step by step procedure of key generation is explained in detail:

Step 1: The method starts with the generation of a burning ship fractal image using initial value x_0 and y_0. The initial values (x_0 and y_0) are set to (0, 0) inputted to the burning ship function. A fractal image *P* of size *M*N* is generated using the given eq. (1) & Eq. (2) which is shown in figure 1.

Step 2: In order to compute a secret key *SK*, an external key *K* of random integer of size (8*sqrt(*M*N*)) is employed. Further, an external key *K* is divided into groups of size (256), referred to as session keys:

K= k1, k2, k3, k4, k5, k6, k7, k8.

Step 3: Apply Hilbert scrambles to shuffle the fractal image pixels and obtained corresponding scrambled pixel sequence (*SP*).

Step 4: The scrambled pixels of fractal image *SP* are expanded in the order from left to right and top to bottom to get a one-dimensional array $PI = \{SP_1, SP_2, \ldots, SP_{M*N}\}$ of length $M*N$ and, also convert each element into an integer number as:

$$SP = floor(SP.* 2^{32}) \tag{3}$$

$$SP = round(mod(SP(:,:)/10^8, 256)) \tag{4}$$

Step 5: Compute the eight vectors (C1,C2, …, C8) by assigning eight different types of operations to each group. In table 1, a detailed description of the fractal image pixel range and its corresponding operation is given.

Table 1: A description of keys, index values of image pixels and corresponding operation

Key No.	**Index Range**	**Operation**
x1	(i>=1) && (i<=256)	C1 = x1⊕PI
x2	(i>=257) && (i<=512)	C2 = x2⊕NotPI
x3	(i>=513) && (i<=768)	C3 = Not (x3)⊕PI
x4 and x5	(i>=769) && (i<=1024)	C4 = x4⊕PI⊕x5
x5	(i>=1025) && (i<=1280)	C5 = mod((x5+256)-PI),256)
x6	(i>=1281) && (i<=1536)	C6 = if(i%2==0), x6⊕PI
x7	(i>=1537) && (i<=1792)	C7 = if(i%2!=0), x7⊕PI
x8	(i>=1793) && (i<=2048)	C8 = mod((x8+PI),256)

Step 6: After executing a single round of operations, modify the session key as follows:

$$k_i = \begin{cases} mod(floor(x_i + x_{i+1}), 256), & if(1 \le i \le 7) \\ mod(floor(x_1 + x_i), 256), & if\ (i == 8) \end{cases} \tag{5}$$

Step 7: Further, the same set of operations is executed to get the temporary blocks of a secret key (temp key/C). The number of rounds to execute the given set of operations depend on the size of used fractal image (i.e. P_{M*N}). For a 256*256 image, the set of operations will be executed 32 times using modified session keys every time.

Step 8: Set (n1, n2) = size(C). To increase the randomness and complexity of the generated sequence, update C_i*'s* for $1 \le b \le$ n2 as:

$$C_i(b) = C_i(mod((b + sh - 1), n2) + 1) \tag{6}$$

Where $1 \le i \le 8$, and $sh=n2/M$.

Step 9: Apply to sort to all C_i's and obtained sorted arrays (CS1, CS2, …, CS8) and its corresponding sorting index (S1, S2, …, S8). Permute each vector value and arrange all C_i's according to the obtained index value *Si's* as:

$$blockkey = C(Si) \tag{7}$$

Step 10: Finally, a key is obtained by concatenating all blockkeyi's and reshape it to the 2D array.

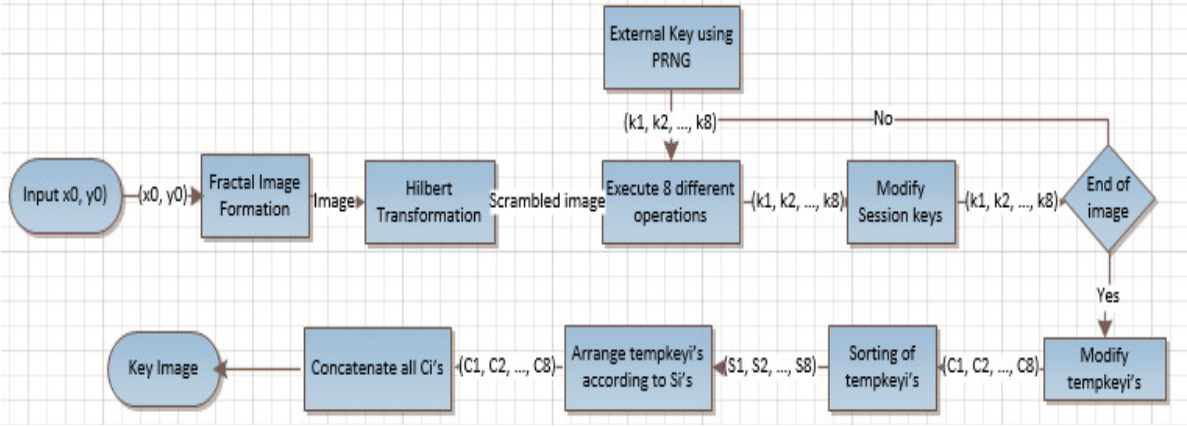


Figure 3. Key Generation Process

4.2 Encryption Process

Input: A fractal-based key sequence *SK* and a plain image $P = \{P_{M*N}\}$.

Step 1: Generate Noisy image: Process starts by embedding noise to the plain image*P*and got a noisy image NoiseIm of size *M*N*as:

$$NoiseIm = P + \big(2 * rand\big(size(P)\big) - 1\big) * temp \tag{8}$$

Where temp ϵ {10, 100}.

Step 2: Noisy image pixel permutation: Apply the Knuth shuffle method in a forward direction to confuse the noisy image pixels and produce a scrambled image (ScrambleIm) of size *M*N*using algorithm 1.

Step 3: Scrambled image pixel substitution: Diffuse scrambled image pixels using secret key *SK* which needs to be updated at each step. The following arithmetic operations would be performed to generate the required cipher image:

$$Sumkey = 0 \tag{9}$$

$$(r, c) = size(ScrambleIm) \tag{10}$$

For each scrambled image pixel, the variable Sumkey will be calculated as:

$$Sumkey = Sumkey + SK_{ij} \tag{11}$$

Where$1 \le i \le r\ and\ 1 \le j \le c$ and SK_{ij} would be updated depending on the pixel index as defined in Eq. 12:

$$SK_{ij} = \begin{cases} \left(SK_{ij} * 10^8\right) mod\ 255, if\ i = 1\ and\ j = 1 \\ \left(SK_{ij} + SK_{ij-1} * 10^8\right) mod\ 255, if\ i = 1\ and\ j! = 1 \\ \left(SK_{ij} + SK_{i-1c} * 10^8\right) mod\ 255, if\ i! = 1\ and\ j = 1 \\ \left(SK_{ij} + SK_{i-1j-1} * 10^8\right) mod\ 255, Otherwise \end{cases} \quad (12)$$

The next step is to compute cipher image by performing the following steps:

$$temp = \begin{cases} \left(Sumkey + SK_{ij}\right) mod\ 255, if\ i = 1\ and\ j = 1 \\ bitxor((Sumkey + SK_{ij})\ mod\ 256, \left(Sumkey + CI_{ij-1}\right) mod\ 255), if\ i = 1\ and\ j! = 1 \\ bitxor((Sumkey + SK_{ij})\ mod\ 256, (Sumkey + CI_{i-1c})\ mod\ 255), if\ i! = 1\ and\ j = 1 \\ bitxor((Sumkey + SK_{ij})\ mod\ 256, \left(Sumkey + CI_{i-1j-1}\right) mod\ 255), \quad Otherwise \end{cases} \quad (13)$$

And

$$CI_{ij} = bitxor(ScrambleIm, temp) \quad (14)$$

After executing the above steps, a cipher image *CI* of size *M*N* will be obtained. This method is implemented only once to encrypt a grayscale image (256*256). However, it also works well to encrypt a color image as well (refer performance analysis section for results). In case of a color image, it needs to divide into its three components R, G, and B and then apply the given method individually to each color-component. At last, combine all components to get a final cipher image.

4.3 Decryption Process

The decryption process is just the opposite of the proposed encryption algorithm. To decrypt the cipher image, receiver required the same secret key and the cipher image to be decrypted. The given image encryption algorithm composed of two steps:

1. Image permutation by applying the Knuth shuffle method
2. Image diffusion using a secret key

The decryption process will start with image diffusion using the same secret key followed by executing the Knuth shuffling method in reverse order. Finally, the decrypted image will be obtained which is our initial plain image.

5. SIMULATION RESULT AND SPEED ANALYSIS

This section shows the visual results of the proposed image encryption method, implemented using MATLABR2016b with system configuration Intel® Atom™ x7-z8700 CPU @1.60GHz and 4 GB RAM. The method is executed on various randomly chosen test images like Lena(G), cameraman(G), tree(C), etc. and a black and a white image of size 256*256. The simulation results shown in figure 4 depict that the given method is able to:

1. Convert an intelligent image (gray/color/black/white) into its corresponding unintelligible image.
2. Reconstruct the original image after applying the decryption algorithm at the receiver end.

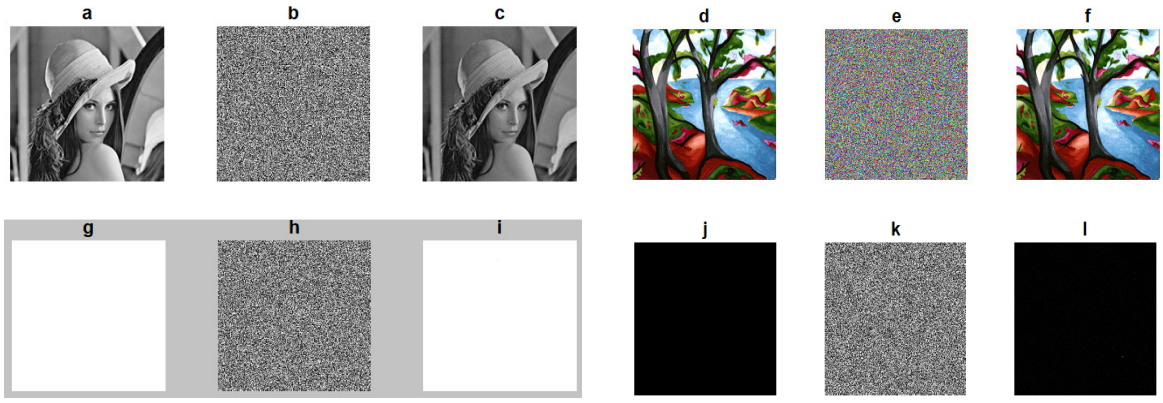


Figure 4. The plain images and their corresponding cipher and decipher images

The term speed analysis refers to the time required to implement the given algorithm, depends on various factors such as CPU structure, OS, RAM size, etc. The cipher process executes image confusion and diffusion process only once. With respect to the above-given system configuration, the actual time complexity of simulation result is around 1.3762±0.0577s for a gray image of size 256*256.

5.1 Impact of adding noise

Noise embedding to the plain image has a special impact to generate two distinctive cipher images while encrypting the same plain image with the same key twice. Therefore, it prevents the unauthorized user to get the plaintext information and thus reduce the risk of image information leakage. The result can be seen in the figure 5.

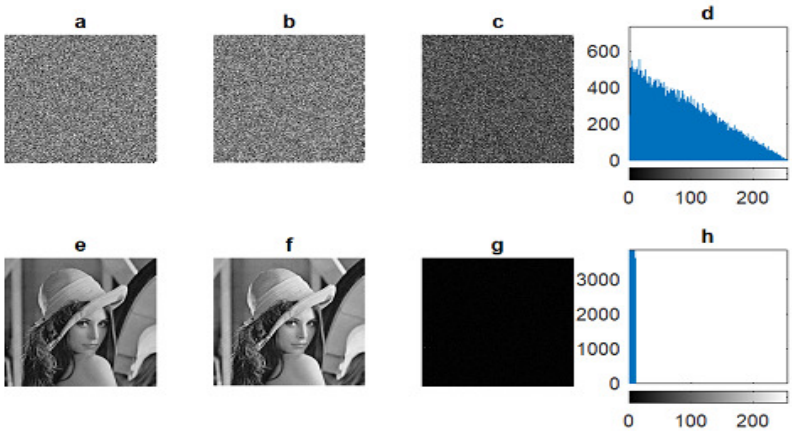


Figure 5. (a) Cipher image C1, (b) Cipher image C2, (c) |C1-C2|, (d) histogram difference of |C1-C2|, (e) Decipher image D1, (f) Decipher image D2, (g) |D1-D2|, (h) histogram difference of |D1-D2|

5.2 Histogram Analysis

To efficiently hide the image information, the image pixel intensity value should be uniformly distributed among the interval [0, 255]. Figure 6 shows the histogram of a plain image as well as the cipher image. In the case of a plain image, the pixel distribution is quite dependent on the image structure, whereas the pixel distribution of the cipher image is uniform irrespective of the corresponding plain image. Moreover, to measure the pixel distribution uniformity, chi-square goodness of fit test has been carried out. The formula to calculate it:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{15}$$

Where(O_i) and (E_i)represent observed and expected image intensity value respectively. The chi-square value for the degree of freedom (M-1) would be determined under the null hypothesis assuming significance level 0.05. If $\chi^2 < \chi^2_{0.05}(255)$, the null hypothesis will be accepted, i.e. the pixel distribution is uniform. Here, a grayscale lena.gif image has the χ^2 value 230.2968 which is less than the standard value 293.2478 [48], indicates the acceptance of the null hypothesis. Thus, the attacker would not be able to obtain the image information by looking at the cipher image graphical presentation.

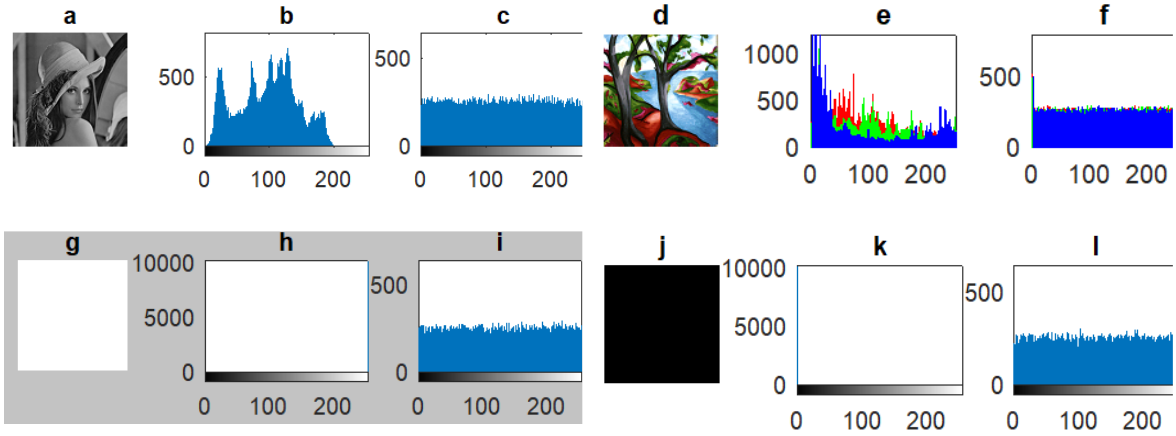


Figure 6. The plain image and cipher image histogram

5.3 Key Space Analysis

The key space refers to the all possible combinations of the keys that are available in a certain image encryption algorithm. A sufficient large key space makes the brute force attack infeasible. The secret key generated by the given algorithm requires ten initial values (x_0, y_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8), which are of a double datatype. According to IEEE 754 format, the computational precision of a double datatype is about 10^{15}. So, the possible key space will be $(10^{15})^{10}=10^{150}$ $\approx 2^{495}$.As suggested an ideal image encryption algorithm must have a key space greater than 2^{100}[49]. So as the calculated key space is sufficiently large to resist the brute force attack.

5.4 Information Entropy

The information entropy test is the measurement of the image randomness. The formula to calculate information entropy is given as[50]:

$$H(s) = -\sum_{i=0}^{2^n-1} p(s_i) log_2 p(s_i) \quad (16)$$

Here $p(s_i)$ represents the probability of occurrence of symbol *si* ϵ *s* (total number of symbols). A grayscale image has 256 symbols which require 8 bits to represent. In that case, the entropy value $H(s)$ of a cipher image should be close to 8. Or in other words, the uniform distribution of gray value leads to the high randomness. The test has been carried out to the gray & color images and their corresponding cipher images. Refer the table 2for results. The outcome verifies the randomness of the cipher image, subsequently able to resist the entropy attack.

Table 2. Information entropy of plain images and cipher images

Name	Lena(G)	Cameraman(G)	Pepper(C)	Tree(C)	Mandrill(C)	Black	White
Actual Entropy	7.4140	7.0097	7.6339	7.6140	6.8178	0	0
Ciphered Entropy	7.9974	7.9976	7.9990	7.9992	7.9990	7.9973	7.9971

5.5 Correlation Coefficient Analysis

A plain image always has highly correlated adjacent image pixels in horizontal, vertical and diagonal direction. An efficient image encryption algorithm must be able to break the pixel correlationin all directions to improve the resistance against the statistical attack. The adjacent pixel correlation can be computed as follows:

$$cc = \frac{cov(x,y)}{\sigma x * \sigma y} \quad (17)$$

Where $\sigma x = \sqrt{var(x)}$ and $\sigma y = \sqrt{var(y)}$

$$var(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \quad (18)$$

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)) \quad (19)$$

Here *x* and *y* denote the adjacent pixels of a plain/cipher image. In figure 7, the pixel correlation distribution of a plain image, as well as the cipher image, is shown in all three directions. Ideally, the correlation coefficient factor should be close to 1 for the plain image, whereas almost 0 for the

cipher image along with all directions. The quantitative results of the correlation value are listed in table 3 for various plain and cipher images. The result proves the efficiency of the proposed algorithm against statistical attack.

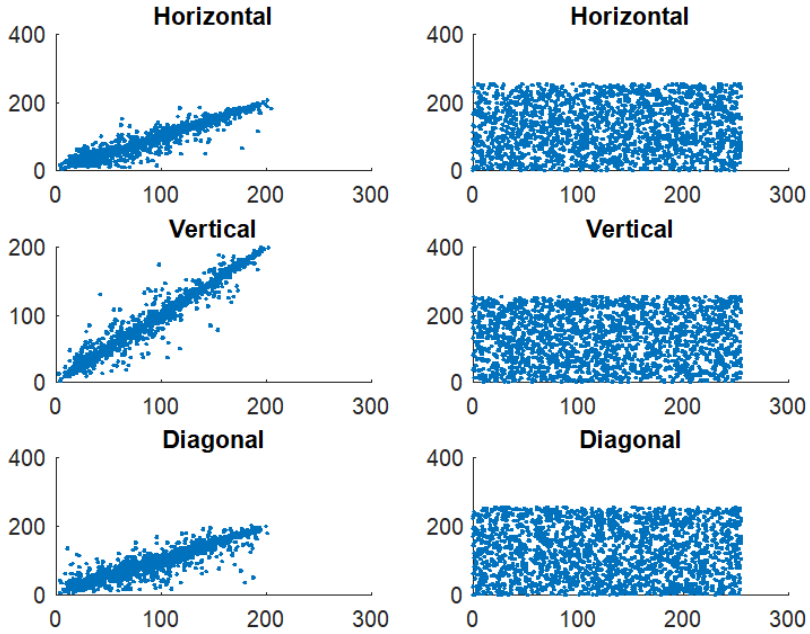


Figure 7. Pixel Correlation plot of plain image and cipher image in all directions

Table 3: Pixel correlation coefficient values of the plain image and its correspondingcipher image

Image Name	Plain Image			Cipher Image			Correlation between two images
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	
Lena(G)	0.9541	0.9778	0.9320	0.0053	0.0011	-0.0038	-0.0037
Cameraman (G)	0.9334	0.9592	0.9086	0.0014	0.0053	0.0087	0.0043
Pepper(C)	0.9943	0.9950	0.9882	0.0018	-0.0007	-0.0036	-0.0027
Tree(C)	0.9430	0.9457	0.9180	-0.0064	0.0006	-0.0064	0.0005
Mandrill(C)	0.9309	0.9187	0.9019	0.0042	-0.0018	-0.0003	-0.0005

5.6 NPCR and UACI Test

The differential attack measures the impact on the cipher image while changing one bit or one pixel in the original image. The number of pixel change rate (NPCR) and unified average changed intensity (UACI) tests are executed to check whether the algorithm is resilience under given modification[51]. The test scan is be computed as:

$$NPCR = \frac{1}{M * N} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i,j) * 100\% \quad (20)$$

$$UACI = \frac{1}{M * N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{C_1(i,j) - C_2(i,j)}{255} * 100\% \quad (21)$$

Where $D(i,j)$ is calculated as:

$$D(i,j) = \begin{cases} 0, & if\ C_1(i,j) = C_2(i,j) \\ 1, & if\ C_1(i,j) \neq C_2(i,j) \end{cases} \quad (22)$$

Here C_1 and C_2 are two cipher images which are obtained after encrypting the plain image and one bit changed plain image respectively. The expected NPCR and UACI test values are 99.61% and 33.46% respectively[51]. Table 4 shows that the obtained NPCR and UACI values are quite close to the ideal value. It proved that the proposed method is highly sensitive to the image pixel change, consequently secured from the differential attack.

Table 4. NPCR and UACI values of cipher images

Image Name	**NPCR (%)**	**UACI (%)**
Lena(G)	99.6109	33.3519
Cameraman (G)	99.6185	33.5311
Pepper(C)	99.6139	33.4605
Tree(C)	99.6154	33.3745
Mandrill(C)	99.6261	33.3283

5.7 Key Sensitivity Analysis

A good encryption algorithm must be sensitive to the encryption key and a plain image. Even a pixel change in an image or minute change in key data should produce a completely different outcome as expected. The image sensitivity feature has been discussed in the previous section 5.6. Now we will discuss the impact of change in the initial value of a key. The key sensitivity is important in both phases of a cryptosystem i.e. encryption and decryption:

1. The cipher image must be completely different if the same plain image is encrypted by using a slightly changed secret key. (see enc(C1-C2) in figure 8).
2. A different plain image must be obtained if the same cipher image is decrypted by using a slightly changed secret key. (see dec(D1-D2)in figure 8).

Originally, the initial values used to generate a secret key *K* were (0, 0). Now the proposed method is tested against two pairs of changed value, i.e. (10^{-7}, 10^{-7}) and (10^{-11}, 10^{-11}). To show the impact of the modified key in the encryption process, the same plain image is encrypted twice (encrypted as C1 and C2) using keys *K*1 and *K*2. The impact has shown visually by calculating |C - C1| and quantitatively using NPCR and UACI test. (Refer figure 8 and table 5).The figure shows that a slight change in the key initial value changed the cipher image completely. And the NPCR and UACI test values also reflect the standard values as expected.

In the decryption phase, the correct cipher image has been decrypted using correct secret key *K* as well as incorrect secret keys *K*1 and *K*2. The corresponding decrypted images are shown in figure 8. The visual results exhibit that the modifier key was unable to decrypt the cipher images correctly. Therefore, it can be seen that if the correct key information is not available, then it's not possible to reconstruct the plain image accurately. Thus, the proposed system is strongly key sensitive towards encryption and decryption process.

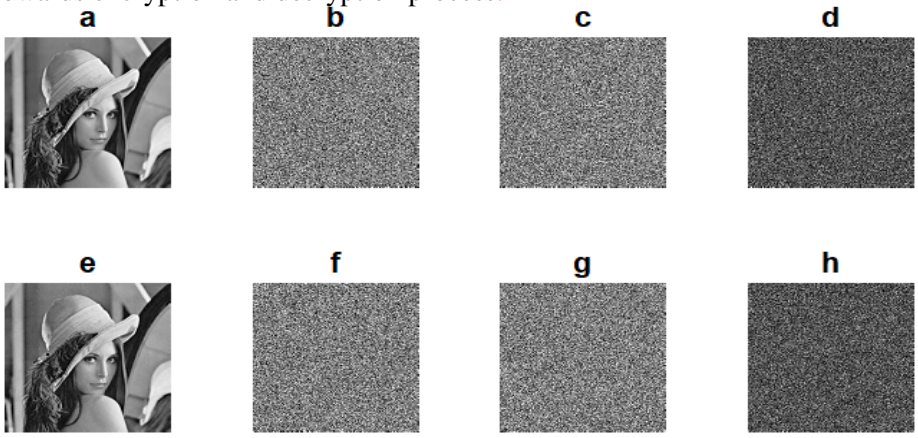


Figure 8. Key sensitivity Analysis; (a) Plain image P, (b) Cipher image C, (c) C1=enc(P, K1), (d) |C-C1|, (e) Decipher image D=dec(C,K), (f) D1=dec(C,K1), (g) D2=dec(C,K2), (h) |D1-D2|

Table 5. Key sensitivity analysis using NPCR and UACI tests

Initial key value	NPCR (%)	UACI (%)
(x0, y0) = (0.0, 0.0)	99.6109	33.3519
(x0, y0) = (10^{-7}, 10^{-7})	99.6154	33.5097
(x0, y0) = (10^{-11}, 10^{-11})	99.6261	33.5497

5.8 Noise Attack and Data Loss

A communication network always consists of different types of noise and data loss during transmission. It's not possible for a receiver to decrypt an error encrypted image into a correct plain image even using an accurate secret key. A cryptosystem should be quite robust to handle a certain level of errors and data loss.

The original Lena image is encrypted using the correct key and then tested against noise attack and data loss attack. A Salt and Pepper noise of density 2% is added to the encrypted Lena image and then decrypted using the same key. Similarly, to check the image feasibility against data loss, a block of 20*20 pixels is replaced with zeros in the encrypted image and then decrypted at the receiver end. Figure 9 shows the error encrypted images and their corresponding reconstructed images to demonstrate the efficacies of the method.

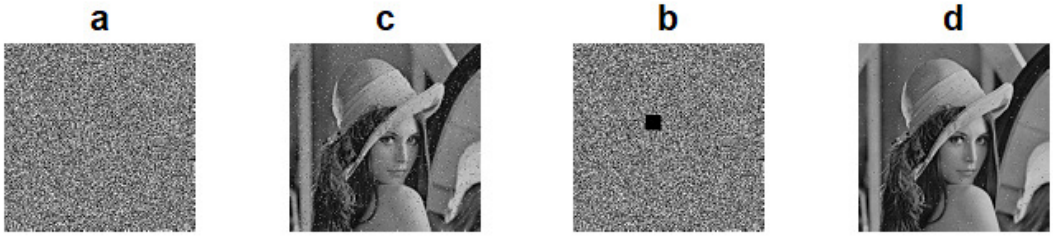


Figure 9. Method robustness against noise attack and data loss

5.9 Performance Comparison with Existing Methods

A performance comparison in terms of randomness and sensitiveness has been carried out between the proposed method and some recent image encryption schemes. Here, a grayscale Lena image of size 256*256 was considered as a key image. The comparison analysis is shown in table 6 for entropy, NPCR test, UACI test, and adjacent pixel correlation coefficient values in all three dimensions. The test scores represent that the proposed algorithm has passed the information entropy, NPCR and UACI tests with approximate close ideal values. A low correlation coefficient value also depicts that no adjacent image pixels are correlated to each other. From the given analysis table, it is proved that the proposed method has better performance that can be used in secure image transmission.

Table 6. Comparison between the proposed method and existing methods

Scheme	Entropy	NPCR	UACI	Correlation Coefficient		
				Horizontal	Vertical	Diagonal
Ours	7.9974	99.6109	33.3519	0.0053	0.0011	-0.0038
[31]	7.9979	99.61	33.46	-0.0156	-0.0022	-0.0028
[33]	7.9971	99.5986	33.4561	-0.0029	-0.0017	0.0004
[52]	7.9971	99.6689	33.4936	0.0053	-0.0027	0.0016
[18]	7.9997	99.740	33.470	0.0021	0.0009	0.0018
[53]	7.9976	99.6200	33.4169	0.0056	0.0037	0.0032

6. CONCLUSION

The paper introduced a standard permutation-substitution architecture-based cryptosystem using a fractal based secret key. The complete process is designed using three techniques which comprises of Hilbert transformation, Knuth scrambling method and dynamic diffusions at each step respectively.

The various procedures and their distinctive characteristics used in the proposed scheme can be stated as 1). A burning ship fractal function is employed to generate a secret key sequence which exhibits the chaotic behaviour; consequently, the key has a large keyspace, better ergodicity and high sensitivity to its initial condition. 2). The secret key is modified at each encryption stage by applying some algebraic transformation to enhance the randomness of the key. 3). Random noise is embedded in the plain image before applying any encryption primitive, because of that the same plain image converts into different cipher images while encrypting with the same secret key twice. 4). In the diffusion phase, each scrambled pixel is encrypted using a corresponding key pixel and the previously ciphered pixel to make more robust cryptosystem. 5). The algorithm attains good confusion-diffusion properties by implementing the permutation-substitution framework.

The effectiveness of the discussed cryptosystem has been verified by executing various tests on the grayscale image and color image. All theoretical tests such as key space analysis, speed analysis, and random noise insertion into the plain image can resist the brute force attack, known-plaintext attack and chosen-plaintext attack. Similarly, the experimental results with the help of chi-square test, NPCR and UACI tests, key sensitivity test, adjacent pixel correlation, and information entropy demonstrate the appropriateness of the secure image transmission over the unsecured communication channel.

REFERENCES

[1] C. A. Pickover, Computers, Pattern, Chaos, and Beauty: Graphics from an Unseen World. Courier Corporation, 2001.

[2] B. B. Mandelbrot, The fractal geometry of nature, vol. 173. WH freeman New York, 1983.

[3] B. Howell, A. Reese, and M. Basile, "Fractal Cryptology," New Mexico High School, Supercomputing Challenge Final Report, 2003.

[4] G. B. Huntress, "Encryption using a fractal key," US6782101 B1, 24-Aug-2004.

[5] M. Ivo, R. Jasek, and P. Varacha, "Analysis of the Fractal Structures For the Information Encrypting Process," International Journal of Computers, vol. 6, no. 4, pp. 224–231, 2012.

[6] R. M. Crownover, Introduction to fractals and chaos. Jones and Bartlett, 1995.

[7] M. Alia and A. Samsudin, "A new public-key cryptosystem based on mandelbrot and julia fractal sets," Asian Journal of Information Technology, AJIT, vol. 6, no. 5, pp. 567–575, 2007.

[8] V. Rozouvan, "Modulo image encryption with fractal keys," Optics and Lasers in Engineering, vol. 47, no. 1, pp. 1–6, 2009.

[9] A. Chopra, M. Ahmad, and M. Malik, "An enhanced modulo-based image encryption using chaotic and fractal keys," in Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in, 2015, pp. 501–506
.

[10] S. Kumar, "Public Key Cryptographic System Using Mandelbrot Sets," in MILCOM 2006 - 2006 IEEE Military Communications conference, 2006, pp. 1–5.

[11] Y. Sun, L. Chen, R. Xu, and R. Kong, "An image encryption algorithm utilizing Julia Sets and Hilbert Curves," PloS one, vol. 9, no. 1, p. e84655, 2014.

[12] Y. Sun, R. Kong, X. Wang, and L. Bi, "An image encryption algorithm utilizing Mandelbrot set," in Chaos-Fractals Theories and Applications (IWCFTA), 2010 International Workshop on, 2010, pp. 170–173
.

[13] S. Sattari, A. Akkasi, R. A. Lari, and M. Khodaparasti, "Cryptography in social networks using wavelet transform, fractals and chaotic functions," International Research Journal of Applied and Basic Sciences, Science Explorer Publications, ISSN, pp. 1627–1635, 2015.

[14] H. Kashanian, M. Davoudi, and H. Khorramfar, "Image Encryption using chaos functions and fractal key," International Journal of Computer Science and Network Security (IJCSNS), vol. 16, no. 10, p. 87, 2016.

[15] Q. Zhang, S. Zhou, and X. Wei, "An efficient approach for DNA fractal-based image encryption," Appl. Math. Inf. Sci, vol. 5, pp. 445–459, 2011.

[16] M. Mikhail, Y. Abouelseoud, and G. ElKobrosy, "Two-Phase Image Encryption Scheme Based on FFCT and Fractals," Security and Communication Networks, 2017. [Online]. Available: https://www.hindawi.com/journals/scn/2017/7367518/abs/. [Accessed: 09-Nov-2017].

[17] H. Oğraş and M. Türk, "A Robust Chaos-Based Image Cryptosystem with an Improved Key Generator and Plain Image Sensitivity Mechanism," Journal of Information Security, vol. 08, no. 01, pp. 23–41, 2017.

[18] S. K. Abd-El-Hafiz, A. G. Radwan, S. H. A. Haleem, and M. L. Barakat, "A fractal-based image encryption system," IET Image Processing, vol. 8, no. 12, pp. 742–752, May 2014.

[19] S. H. AbdElHaleem, A. G. Radwan, and S. K. Abd-El-Hafiz, "Design of pseudo random keystream generator using fractals," in 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), 2013, pp. 877–880.

[20] S. Agarwal, "Secure Image Transmission Using Fractal and 2D-Chaotic Map," Journal of Imaging, vol. 4, no. 1, p. 17, 2018.

[21] J. Shaw, O. Saha, and A. Chaudhuri, "An Approach for Secured Transmission of Data using Fractal based Chaos," in IJCA Proceedings on National Conference on Communication Technologies & its impact on Next Generation Computing, 2012, pp. 13–17.

[22] Y. Sun, R. Xu, L. Chen, and X. Hu, "Image compression and encryption scheme using fractal dictionary and Julia set," IET Image Processing, vol. 9, no. 3, pp. 173–183, 2014.

[23] S. Zhu, C. Zhu, and W. Wang, "A novel image compression-encryption scheme based on chaos and compression sensing," IEEE Access, vol. 6, pp. 67095–67107, 2018.

[24] K. Faraoun, "Chaos-Based Key Stream Generator Based on Multiple Maps Combinations and its Application to Images Encryption.," Int. Arab J. Inf. Technol., vol. 7, no. 3, pp. 231–240, 2010.

[25] A. M. Meligy, H. A. Diab, and M. S. El-Danaf, "Chaos Encryption Algorithm using Key Generation from Biometric Image," International Journal of Computer Applications, vol. 149, no. 11, 2016.

[26] S. M. Hussain and H. Al-Bahadili, "A DNA-Based Cryptographic Key Generation Algorithm."

[27] Y. Li, C. Wang, and H. Chen, "A hyper-chaos-based image encryption algorithm using pixel-level permutation and bit-level permutation," Optics and Lasers in Engineering, vol. 90, pp. 238–246, 2017.

[28] M. Ahmad, M. N. Doja, and M. S. Beg, "Security analysis and enhancements of an image cryptosystem based on hyperchaotic system," Journal of King Saud University-Computer and Information Sciences, 2018.

[29] C. Fu, G. Zhang, M. Zhu, Z. Chen, and W. Lei, "A New Chaos-Based Color Image Encryption Scheme with an Efficient Substitution Keystream Generation Strategy," Security and Communication Networks, vol. 2018, 2018.

[30] S. Sun, "A novel hyperchaotic image encryption scheme based on DNA encoding, pixel-level scrambling and bit-level scrambling," IEEE Photonics Journal, vol. 10, no. 2, pp. 1–14, 2018.

[31] Q. Yin and C. Wang, "A New Chaotic Image Encryption Scheme Using Breadth-First Search and Dynamic Diffusion," International Journal of Bifurcation and Chaos, vol. 28, no. 04, p. 1850047, 2018.

[32] S. Hammami, "State feedback-based secure image cryptosystem using hyperchaotic synchronization," ISA transactions, vol. 54, pp. 52–59, 2015.

[33] X. Wang, X. Zhu, and Y. Zhang, "An image encryption algorithm based on Josephus traversing and mixed chaotic map," IEEE Access, vol. 6, pp. 23733–23746, 2018.

[34] Z. Hua, B. Xu, F. Jin, and H. Huang, "Image Encryption Using Josephus Problem and Filtering Diffusion," IEEE Access, vol. 7, pp. 8660–8674, 2019.

[35] X. Huang and G. Ye, "An Image Encryption Algorithm Based on Time-Delay and Random Insertion," Entropy, vol. 20, no. 12, p. 974, 2018.

[36] Z. Hua, Y. Zhou, and H. Huang, "Cosine-transform-based chaotic system for image encryption," Information Sciences, vol. 480, pp. 403–419, 2019.

[37] M. Alawida, A. Samsudin, J. S. Teh, and R. S. Alkhawaldeh, "A new hybrid digital chaotic system with applications in image encryption," Signal Processing, vol. 160, pp. 45–58, 2019.

[38] M. Asgari-Chenaghlu, M.-A. Balafar, and M.-R. Feizi-Derakhshi, "A novel image encryption algorithm based on polynomial combination of chaotic maps and dynamic function generation," Signal Processing, vol. 157, pp. 1–13, 2019.

[39] H. J. Yakubu, E. G. Dada, S. B. Joseph, and A. K. Anukem, "A New Chaotic Image Encryption Algorithm for Digital Colour Images Using Rabinovich-Fabrikant Equations," International Journal of Computer Science and Information Security (IJCSIS), vol. 17, no. 1, 2019.

[40] Z. Tang, Y. Yang, S. Xu, C. Yu, and X. Zhang, "Image Encryption with Double Spiral Scans and Chaotic Maps," Security and Communication Networks, vol. 2019, 2019

.

[41] B. B. Mandelbrot, "Fractal aspects of the iteration of z→ Λz (1-z) for complex Λ and z," Annals of the New York Academy of Sciences, vol. 357, no. 1, pp. 249–259, 1980.

[42] M. Michelitsch and O. E. Rössler, "Spiral structures in Julia sets and related sets," in Spiral symmetry, World Scientific, 1992, pp. 129–134.

[43] M. Michelitsch and O. E. Rössler, "The 'burning ship' and its quasi-Julia sets," Computers & graphics, vol. 16, no. 4, pp. 435–438, 1992.

[44] S. Agarwal and A. Negi, "INVENTIVE BURNING SHIP," International Journal of Advances in Engineering & Technology, vol. 6, no. 4, p. 1788, 2013.

[45] S. Agarwal and A. Negi, "Burning Ship and Its Quasi Julia Images Using Mann Iteration," in Recent Advances in Intelligent Informatics, Springer, 2014, pp. 401–410.

[46] T. Sivakumar and R. Venkatesan, "Image encryption based on pixel shuffling and random key stream," International Journal of Computer and Information Technology, vol. 3, no. 06, 2014.

[47] T. K. Hazra and S. Bhattacharyya, "Image encryption by blockwise pixel shuffling using Modified Fisher Yates shuffle and pseudorandom permutations," in Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual, 2016, pp. 1–6.

[48] N. D. Gagunashvili, "Chi-square tests for comparing weighted histograms," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 614, no. 2, pp. 287–296, 2010.

[49] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," International Journal of Bifurcation and Chaos, vol. 16, no. 08, pp. 2129–2151, 2006.

[50] C. E. Shannon, "Communication theory of secrecy systems," Bell Labs Technical Journal, vol. 28, no. 4, pp. 656–715, 1949.

[51] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT), pp. 31–38, 2011.

[52] Y. Wu, Y. Zhou, J. P. Noonan, and S. Agaian, "Design of image cipher using latin squares," Information Sciences, vol. 264, pp. 317–339, 2014.

[53] J. Wu, X. Liao, and B. Yang, "Image encryption using 2D Hénon-Sine map and DNA approach," Signal Processing, vol. 153, pp. 11–23, 2018.

AUTHOR

Shafali Agarwal has received MCA degree from UPTU, Lucknow in 2004 and M.Phil in Computer Science from Alagappa University, Karaikudi, Tamil Nadu in 2013. She got her Ph.D. in Computer Science from Singhania University, India in 2014. She has served as a faculty member in the Department of Computer Applications in JSSATE, Noida till June 2016. She has published more than 15 research papers in various International journals and conferences indexed in Scopus, Emerging Sources Citation Index, Springer, ACM, Thomson Reuters, google scholar and in many more. She was awarded a best paper presentation award in a conference ICVISP held in Las Vegas, USA. Her research interest includes fractal, cryptography and image processing.