A DNA BASED CHAOTIC IMAGE FUSION ENCRYPTION SCHEME USING LEA – 256 AND SHA – 256

SHRADHA MOHANTY[a1], ALKESHA SHENDE[b], K. ABHIMANYU KUMAR PATRO[c] AND BIBHUDENDRA ACHARYA[d]

[abcd]Department of Electronics and Telecommunication Engineering, National Institute of Technology, Raipur, Chhattisgarh, India

ABSTRACT

In today's scenario, security of image data is essential during communication over open network such as internet from various security attacks. It's a big challenge in terms of security and people need a good encryption scheme for secure communication of digital images in an open sharing network. This paper proposed a secure image encryption scheme which uses the sensitivity of chaos systems, the flexibility of hash functions, DNA operations and LEA encryption. In this proposed scheme, we used the hash value of a random key image as the 256 – bit key for LEA encryption. Using this key and the initial sequence derived from input image, LEA encryption is performed. This output along with chaos generated pseudo random binary sequences are DNA encoded then XOR operation performed and finally DNA decoded to get cipher image. The simulation results and security analysis shows that the resultant ciphered output is highly unintelligible and confused which has incredible resistance against common attacks.

KEYWORDS: Chaotic Map, DNA Sequence Operation, Image Encryption, Image Fusion, LEA-256, SHA-256.

Today, being the age of social media, most of the information is shared in the form of digital images. Thus an efficient and secure image encryption scheme is the need of the hour. Traditional symmetric block ciphers like Data Encryption Standard (DES) [Pub, 2001], Advanced Encryption Standard (AES) [Coppersmith, 1994] were used for encryption, however due to small block size and other vulnerabilities their use for encryption of digital images was deemed to be unadvisable. Several other block cipher in turn have been proposed like Light Encryption Device (LED) block cipher [Guo et. al., 2011], Lightweight block (Lblock) cipher [Wu and Zhang, 2011], which are light weight, easier to implement than AES or DES and much faster. These follow a simple ARX (Addition – Rotation – XOR) structure. However even their block size is small to be used for images and to ensure sufficient security. In 2013, Hong et al. proposed the LEA block cipher, yet another simple ARX based block cipher, which has a bigger block size of 128 – bits, three size options for key, light weight and incredibly fast, thus more attractive than traditional block ciphers.

To introduce diversity several newer options were explored and chaos became the trending topic in the field of cryptography. Due to extreme sensitivity to initial conditions and system parameters, incorporation of chaotic sequences in encryption has become very common [Chai et. al., 2016 & Wang et. al., 2015]. Several new and diverse schemes that have been proposed use chaos in their algorithm. In 1995, Lloyd et al., studied coupled logistic maps. These maps were 2 – D and hence an improvement over the traditional 1 – D Logistic map, Tent map, Sine map and so on. Over the years even higher dimensional maps have been used like Chai et al. [2016] and Zhang et al. [2009] used 4 – D hyper – chaotic maps, Yuan et al. [2017] used 5 – D hyper – chaotic map and Wu et al. [2016] used 6 – D hyper – chaotic map. Some methods to diversify chaotic maps without increasing their dimensions have also been proposed like Wu et al. [2015] and Zhou et al. [2014] used combination of 1 – D chaotic maps.

Along with chaos, DNA computing has also become a very highly explored field. In 2006, Xiao et al., studied DNA as an emerging tool in cryptography. The immense storage capacity of DNA bases and high amount of parallelism has made their use popular. In 2007, Shyam et al., proposed an encryption scheme based on DNA XOR operation, the same is used in several other schemes [Guesmi et. al., 2016] [Zhang et.al., 2013], [Wu et. al., 2015], while DNA addition and subtraction is used in [Mondal and Mandal, 2016], [Zhang et. al., 2009]. Apart from that, simple DNA sequence conversion can also be used for effective encryption as in [Yunpeng et. al., 2011].

While encryption is used for confidentiality, hash functions have been used to provide authentication of data [Stallings, 2006]. Several hash functions have been created and used for the same purpose. A property that makes hash functions very convenient to use is that they do not have any fixed size for input while the output size for each function is fixed and hardware implementation is easy [Lloyd, 1995]. In 2003, Gilbert et al., studied several hash functions (SHA – 256, SHA – 384 and SHA – 512) and analysed their properties. He found that these were resistant to Chabaud and Joux's attack, Dobbertin – style attacks and even differential and linear attacks. This provides a concrete reason to

[1]Corresponding author

include hash functions in an encryption scheme. However hash functions are not only limited to providing authentication, they be used differently too as in [Guesmi et. al., 2016] SHA – 256 is used for key generation. This shows how broad the range of application of hash functions can be.

In this paper we have used the LEA with 256 bit key for encryption. The key we have used is a randomly generated image. We have used SHA – 256 to get the 256 – bit hash value of the random key image which in turn is used as the key for LEA encryption. To balance complexity and security we have used cross – coupled logistic map as in [http://sprott.physics.wisc.edu]. We generate a pseudo random binary sequence from the map and use this along with DNA operations to carry out image fusion with the LEA cipher to get the encrypted image. This process has been described in the following sequence. The succeeding section describes in detail each of the components of the system. Then in section 3 we described the proposed scheme. Section 4 contains the results we obtained and security analysis, while section 5 concludes the paper.

COMPONENTS OF THE SYSTEM

LEA – 256

LEA is a 128 – bit block cipher with key size of 128 – bits, 192 – bits or 256 – bits [Hong et. al., 2013]. It uses only three operations – modular addition, bitwise rotation and bitwise XOR. It is faster than traditional block ciphers like AES and DES [Coppersmith, 1994], [Pub, 2001]. In our model we have used 256 – bit key. Using this key and key constants we generate 6, 32 – bit round keys. These round keys are used to generate the cipher text.

$\mathrm{C} = \mathrm{LEA} - 256\ (\mathrm{P}, \mathrm{K})$

where P is the 128 – bit plaintext and K is the 256 – bit key.

The process for encryption is as follows.

Step 1:$\mathrm{Key\_constant} = [3287280091\ 1147300610\ 2044886154\ 2027892972\ 1902027934\ 3347438090\ 3763270186\ 3854829911]$

Step 2: Divide the 256 – bit key K into 8 sequences of 32 – bit each such that $\mathrm{K_i} = (32\mathrm{i}+1)^{\mathrm{th}}\ \text{bit of K to}\ (32\mathrm{i}+32)^{\mathrm{th}}\ \text{bit of K}$, where $\mathrm{i} = 0$ to 7.

Step 3: $\mathrm{IK_i} = \mathrm{K_i}$ for $\mathrm{i} = 0$ to 7.

Step 4: Round Key generation:

for $\mathrm{i} = 1$ to 32

$$\mathrm{IK}_{(6\mathrm{i}\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{IK}_{(6\mathrm{i}+1\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}+1\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}+1\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{IK}_{(6\mathrm{i}+2\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}+2\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}+2\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{IK}_{(6\mathrm{i}+3\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}+3\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}+3\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{IK}_{(6\mathrm{i}+4\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}+4\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}+4\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{IK}_{(6\mathrm{i}+5\ \mathrm{mod}\ 8)+1} = \mathrm{RotL_1}\left(\mathrm{IK}_{(6\mathrm{i}+5\ \mathrm{mod}\ 8)+1} \boxplus \mathrm{RotL_i}\left(\mathrm{Key\_constant}_{(6\mathrm{i}+5\ \mathrm{mod}\ 8)+1}\right)\right)$$

$$\mathrm{RK_i} = [\mathrm{IK}_{(6\mathrm{i}\ \mathrm{mod}\ 8)+1}, \mathrm{IK}_{(6\mathrm{i}+1\ \mathrm{mod}\ 8)+1}, \mathrm{IK}_{(6\mathrm{i}+2\ \mathrm{mod}\ 8)+1}, \mathrm{IK}_{(6\mathrm{i}+3\ \mathrm{mod}\ 8)+1}, \mathrm{IK}_{(6\mathrm{i}+4\ \mathrm{mod}\ 8)+1}\ \mathrm{IK}_{(6\mathrm{i}+5\ \mathrm{mod}\ 8)+1}]$$

end for

where $\mathrm{IK} = \text{intermediate key}, \mathrm{RK} = \text{round key}, \mathrm{RotL_i} = \text{rotate left by i bits and a} \boxplus \mathrm{b} = \mathrm{dec2bin}\ \left(\left(\mathrm{bin2dec(a)} + \mathrm{bin2dec(b)}\right) \bmod 2^{32}\right)$

Step 5: Divide the 128 – bit plaintext P into 4 blocks of 32 – bit each, such that $\mathrm{P_i} = (32\mathrm{i}+1)^{\mathrm{th}}\ \text{bit of P to}\ (32\mathrm{i}+32)^{\mathrm{th}}\ \text{bit of P}$, where $\mathrm{i} = 0$ to 3 and the intermediate text IT is generated as

$$\mathrm{IT_i} = [\mathrm{P_i}(25\ \text{to}\ 32)\ \mathrm{P_i}(17\ \text{to}\ 24)\ \mathrm{P_i}(9\ \text{to}\ 16)\ \mathrm{P_i}(1\ \text{to}\ 8)]$$

Step 6: Encrypt the intermediate text IT:

for $\mathrm{i} = 0$ to 31

$$\mathrm{IT_{i+1}}\{0\} = \mathrm{RotL_9}\left((\mathrm{IT_i}\{0\} \oplus \mathrm{RK_i}\{0\}) \boxplus (\mathrm{IT_i}\{1\} \oplus \mathrm{RK_i}\{1\})\right)$$

$$\mathrm{IT_{i+1}}\{1\} = \mathrm{RotR_5}\left((\mathrm{IT_i}\{1\} \oplus \mathrm{RK_i}\{1\}) \boxplus (\mathrm{IT_i}\{2\} \oplus \mathrm{RK_i}\{2\})\right)$$

$$\mathrm{IT_{i+1}}\{2\} = \mathrm{RotR_3}\left((\mathrm{IT_i}\{2\} \oplus \mathrm{RK_i}\{2\}) \boxplus (\mathrm{IT_i}\{3\} \oplus \mathrm{RK_i}\{3\})\right)$$

$$\mathrm{IT_{i+1}}\{3\} = \mathrm{IT_i}\{0\}$$

end for

where $\mathrm{RotR_i} = \text{Rotate right by i bits}$

Step 7: Cipher text C is given by:

$$\mathrm{C} = [\mathrm{IT_{32}}\{0\}, \mathrm{IT_{32}}\{1\}, \mathrm{IT_{32}}\{2\}, \mathrm{IT_{32}}\{3\}]$$

Cross – coupled Logistic Map

The term “chaos” simply defines “a state of disorder” or “a state of confusion” or “a state of unpredictable behavior” which have the properties of determinacy, non – linearity, ergodicity, non – periodicity. Moreover, it is very much sensitive to initial conditions and systems parameters [8]. It focused on the behavior of dynamical systems. A chaotic map is a map which follows the state of the chaotic behavior. One of the simplest chaotic maps is the Logistic map described as below.

$$\mathrm{a_{n+1}} = \gamma \mathrm{a_n}(1 - \mathrm{a_n}) \qquad \textbf{(1)}$$

where $\gamma \in (0,4)$ and $\mathrm{a_n} \in (0,1)$. The logistic map is chaotic when $\gamma \in (3.5699456,4)$.

Cross-coupled logistic maps [http://sprott. physics.wisc.edu] can be represented by the equation given below:

$$\left.\begin{aligned} \mathrm{a_{n+1}} &= (1 - \epsilon)\eta_0 \mathrm{a_n}(1 - \mathrm{a_n}) + \epsilon\eta_1 \mathrm{b_n}(1 - \mathrm{b_n}) \\ \mathrm{b_{n+1}} &= (1 - \epsilon)\eta_1 \mathrm{b_n}(1 - \mathrm{b_n}) + \epsilon\eta_0 \mathrm{a_n}(1 - \mathrm{a_n}) \end{aligned}\right\} \textbf{(2)}$$

where ϵ is the coupling constant and the value of $\epsilon, a, and\ b$ vary from 0 to 1. For $\epsilon = 0$ the maps are said to be decoupled and for $\epsilon = 1$, they are fully coupled. The values $\eta_0\ and\ \eta_1$ belong to the range (3, 4).

The pseudo random bit sequence generation, which is proposed by Mondal and Mandal, 2016 is as shown below:

$$f(a_k, b_k) = \begin{cases} 1, a_k > b_k \\ 0, a_k \leq b_k \end{cases} \qquad \textbf{(3)}$$

The function $f(a, b)$ gives the required pseudo random binary sequence.

DNA Operations

DNA sequences are used in cryptography due to their high storage capacity, vast parallelism and energy efficiency [Yunpeng et. al., 2011]. DNA sequences contain four bases namely Adenine, Cytosine, Guanine and Thymine, represented by A, C, G and T respectively. Here A and T are the complement of each other and so are G and C. There are 8 kinds of coding schemes that satisfy the Watson – crick complement rule, which are shown in Table I. In our algorithm we have used Rule 1 of the following Table I. Table II shows the XOR operation of 4 DNA bases.

Secure Hash Algorithm SHA – 256

Hash functions provide integrity and authenticity [Algredo-Badillo et. al., 2013] and they are flexible in the sense that the input does not have a particular size but the output size of hash function is fixed. In our scheme we have used SHA – 256 and a key image to generate 256 – bit key [Guesmi et. al., 2016] for LEA encryption. Even a slight change in the key image will completely change the output hash value. Thus this method of key generation provides high security against brute – force attack.

PROPOSED IMAGE ENCRYPTION SCHEME

In this section, we present the procedure for DNA based chaotic image fusion encryption scheme using a Light – weight encryption scheme, LEA – 256 and a Secure Hash Algorithm, SHA – 256. The proposed scheme includes five subsections. In the first subsection, key for LEA encryption is generated by using the random key – image and the secure hash algorithm SHA – 256. For the requirement of Cipher Block Chaining (CBC) mode of LEA encryption, we need an initialization vector (IV). So in the second subsection, a 128 – bit initial sequence is generated by using original image and XOR operation. This 128 – bit initial sequence is regarded as an initialization vector for CBC mode of LEA encryption. In the third subsection, CBC mode of LEA encryption is performed. In the fourth subsection, pseudo random binary sequence is generated by using cross – coupled logistic map. Finally, in the last subsection, DNA image fusion operation is performed. The proposed image encryption scheme is shown in Fig. 1.

Key Generation

The proposed method uses an image as a key. This key – image of size $M_1 \times N_1$ is generated randomly by using the function $randi([0\ 255], M_1, N_1)$. Randomness of the key ensures more confused cipher. Then apply SHA – 256 to this key image to generate 256 – bit hash value. This 256 – bit hash value serves as the key for LEA encryption. For decryption the same 256 – bit hash value of the key image is required.

Table I: DNA encoding and decoding rules

	1	2	3	4	5	6	7	8
A	00	00	01	01	10	10	11	11
T	11	11	10	10	01	01	00	00
G	01	10	00	11	00	11	01	10
C	10	01	11	00	11	00	10	01

Table II: XOR operation for DNA sequence

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

Initial Sequence Generation

As LEA is a block cipher, CBC mode of operation is used for encryption. For the CBC mode of LEA encryption, a 128 – bit initialization vector is required. This will be calculated by first convert the $M \times N$ sized image into a binary sequence of length $M \times N \times 8$ then divide the whole into the number of blocks, each of size 128 – bit and finally, XOR all the blocks. The output is what we call initialization vector which is to be used in CBC mode of LEA encryption.

$$Initial\_sequence\ = XOR_{128}(I)$$

where I is the input image of size $M \times N$.

LEA – 256 Encryption

LEA encryption is carried out as follows:

Step 1: Convert the input image I of size $M \times N$ to binary sequence BI of length $M \times N \times 8$ such that $bin2dec\big(BI(8i-7\ to\ 8i)\big) = P_i$ where $i \in (1, M \times N)$

Step 2: Generate IV from BI

$$IV = XOR_{128}(BI)$$

Step 3: Generate Key from Key Image

$$Key = SHA256(Key\ Image)$$

Step 4: Divide the image into blocks of 128 – bit each and encrypt it using CBC mode of LEA encryption

```
for $i = 1\ to\ n$
    $IPT = IV \oplus BI(128(i-1)+1\ to\ 128i)$
    $C_{LEA_i} = LEA256\ (IPT, Key)$
    $IV = C_{LEA_i}$
end for
```

where $n = (M \times N)/16$ and IPT is the intermediate plain text

Step 5: LEA Cipher is given by

$$C_{LEA} = \left[C_{LEA_1}\ C_{LEA_2}\ C_{LEA_3}\ \cdots\ C_{LEA_n}\right]$$

Pseudo Random Binary Sequence Generation

The pseudo random binary sequence proposed by Mondal and Mandal, 2016 is generated in the following way by using the input parameters shown in Table III.

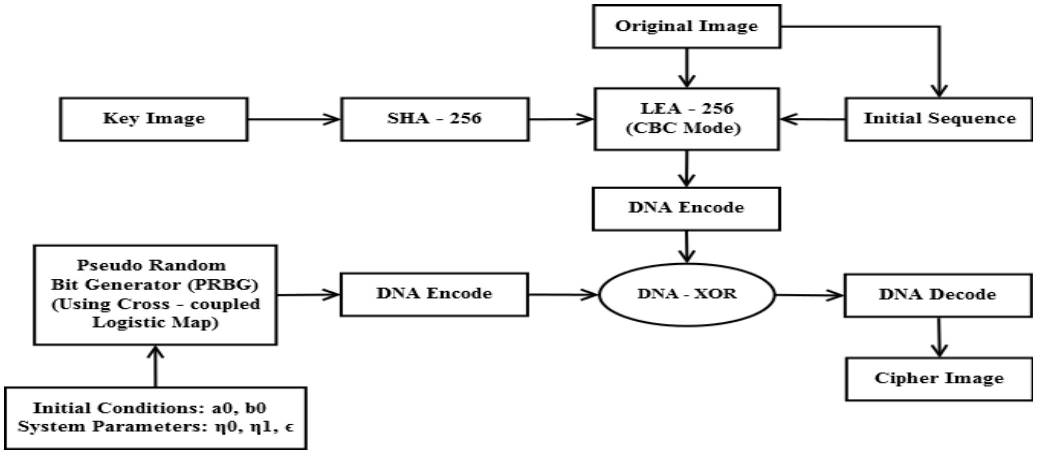


Figure 1: Block diagram of encryption scheme

Table III: Parameters for pseudo random binary sequence generation

Parameter	η_0	η_1	ϵ	a_0	b_0
Value	3.927	3.965	0.042	0.02	0.005

Step 1: Set the initial parameters according to Table – 3 and generate random sequences a and b by using the cross – coupled logistic map such that

$a = [a_1\ a_2\ a_3\ \cdots\ a_{M \times N \times 8}]$ and $b = [b_1\ b_2\ b_3\ \cdots\ b_{M \times N \times 8}]$

Step 2: Generate the pseudo random binary sequence which is as follows:

$PRBS(i) = 1$, if $a_i \geq b_i$

Otherwise,

$PRBS(i) = 0$

where $PRBS$ is the pseudorandom binary sequence

Image Fusion

In this paper, we used DNA XOR operation for performing image fusion. The pseudo random binary sequence and the LEA output, both are encoded by using DNA encoding Rule 1. Then both the DNA encoded sequences are XORed by using DNA XOR method and finally the DNA XORed sequence is decoded to get the cipher image. The process for image fusion operation is described as below:

Step 1: Encode pseudo random binary sequence into DNA sequence by using DNA encoding Rule – 1.

$D_1 = bin2DNA(PRBS)$

Step 2: Encode LEA output into DNA sequence by using DNA encoding Rule – 1.

$D_2 = bin2DNA(C_{LEA})$

Step 3: XOR the two DNA sequences thus obtained

$D = D_1\ XOR\ D_2$

Step 4: Convert the XORed DNA sequence into binary

$B = DNA2bin(D)$

Step 5: Convert the binary sequence B to decimal, taking 8 – bit at a time.

Let the sequence of decimal numbers obtained be called IC, i.e. intermediate cipher.

Step 6: Reshape IC to get Cipher Image

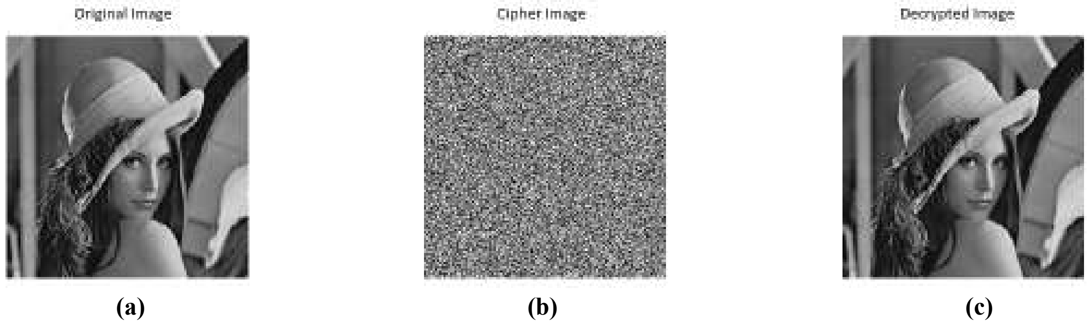
$Cipher\ Image = reshape(IC, M, N)$

The process for decryption is reverse of the process for encryption. In the decryption process, we used the same key image whatever we used it in encryption process. The same pseudo random binary sequence we will use in decryption process whatever used in the encryption process.

SIMULATION RESULTS AND SECURITY ANALYSIS

In this section, the simulation results and the security analysis such as key space analysis, key sensitivity analysis, statistical analysis, differential analysis, and entropy analysis of the proposed image encryption algorithm is presented.

In this paper, the standard image 'Lena' having size 256 × 256 is used for the purpose of testing the simulation results. MATLAB R2012a is used for the process of simulation. The simulation was carried out in a system with windows 8.1 operating system, i5 processor, 1.80 GHz CPU, 8.00 GB memory and 500 GB hard disk. Fig. 2 shows that simulation result of 'Lena' image by using the proposed algorithm. In which Fig. 2 (a) shows the original image, Fig. 2 (b) shows the encrypted image which is completely Unintelligible as desired and completely different than the original image. It means the image is properly encrypted by the proposed encryption algorithm. Fig. 2 (c) shows the decrypted image which is completely identical to the original image indicating successful restoration from the cipher image. Thus it is obvious that the proposed scheme is secure and reliable.



(a) (b) (c)

Figure 2: Simulation results (a) Original 'Lena' image, (b) Encrypted 'Lena' image, (c) Decrypted 'Lena' image

Security analysis, which is essential as it analyzes the security of an encryption algorithm. A good encryption algorithm should resist all kinds of common attacks. The analyses of some of the known common attacks are as described below.

Key Space Analysis

The encryption scheme, which has large key space, has strong resistance to brute – force attack. In this scheme we used the keys are:

1. The parameters $a_0, b_0, \eta_0, \eta_1, \epsilon$ for generation of pseudo random binary sequence.
2. The 256 – bit hash value, used as a key for LEA encryption.
3. The 128 – bit initialization vector for CBC mode of operation.
4. The key image, used to generate the key for LEA encryption.

In this proposed algorithm, the precision values of the key parameters $a_0, b_0, \eta_0, \eta_1, \epsilon$ are $10^{14}, 10^{15}, 10^{12}, 10^{13}, 10^{15}$ respectively. This will generate the key space of $10^{15} \times 10^{18} \times 10^{15} \times 10^{16} \times 10^{17} = 10^{81}$. For the security of SHA – 256 with complexity of the best attack as 2^{128}. Further, the 128 – bit initialization vector generate the key space of 2^{128}. In addition, the key image is also used as a key. So, the total key space will be $10^{81} \times 2^{128} \times 2^{128}$ including key image. So this concludes that the proposed encryption scheme has large key space to resist brute – force attack very effectively.

Key Sensitivity Analysis

Here the key used for LEA encryption is the 256 – bit key obtained by using SHA – 256 on a random image. The image is of size 256 × 256 and each pixel of 8 – bits, thus a total of 524288 bits. A change in a single bit will change the pixel value and thus completely change the 256 – bit key. Also the chaotic systems are highly sensitive to their initial conditions. The initial conditions of chaotic systems are also represented as key in our proposed cryptosystem. So to examine the sensitivities of keys in this cryptosystem, we perform certain tests. In our first test, we changing the value of a_0 to $a_0 + 10^{-15}$ and then by using this changed key, we generate the corresponding cipher image, also generate the difference image of original cipher image and changed cipher image and finally generate the decrypted image from the changed cipher image. Similarly we test the same things by changing the value of b_0 to $b_0 + 10^{-15}$, η_0 to $\eta_0 + 10^{-15}$, η_1 to $\eta_1 + 10^{-16}$, ϵ to $\epsilon + 10^{-17}$. Fig. 3 shows the key sensitivity test of our proposed cryptosystem. Fig. 3 (a), (d), (g), (j), and (m) shows the cipher images by using changed keys. Fig. 3 (b), (e), (h), (k), and (n) shows the difference images of original cipher image and changed keyed cipher image and from this Fig. we observe that the keys are very much sensitive that means our cryptosystem totally dependent on the initial conditions. Fig. 3 (c), (f), (i), (l), and (o) shows the decrypted images generated from changed keyed cipher images and from the Fig.s we found that the original images are not recovered from the changed keyed cipher images. This concludes that our cryptosystem is highly sensitive to their keys.

Plaintext Sensitivity Analysis

Our proposed cryptosystem also sensitive to the plaintext, this proves, by changing one random pixel in the plaintext, then generate the corresponding cipher image and finally generate the difference of original cipher image and the changed cipher image. In this cryptosystem, we generate three cipher images by changing three random pixel positions (64, 64), (128, 128), and (256, 256). Fig. 4 shows the plaintext sensitivity test of our proposed cryptosystem. Fig. 4 (a), (d), and (g) shows the original cipher images. Fig. 4 (b), (e), and (h) shows the ciphers images by changing the pixel values at locations (64, 64), (128, 128), and (256, 256) respectively. Fig. 4 (c), (f), and (i) shows the difference images of original cipher image and changed cipher image. From the difference images we found that our cryptosystem is very much sensitive to plaintext, hence the proposed cryptosystem is secure.

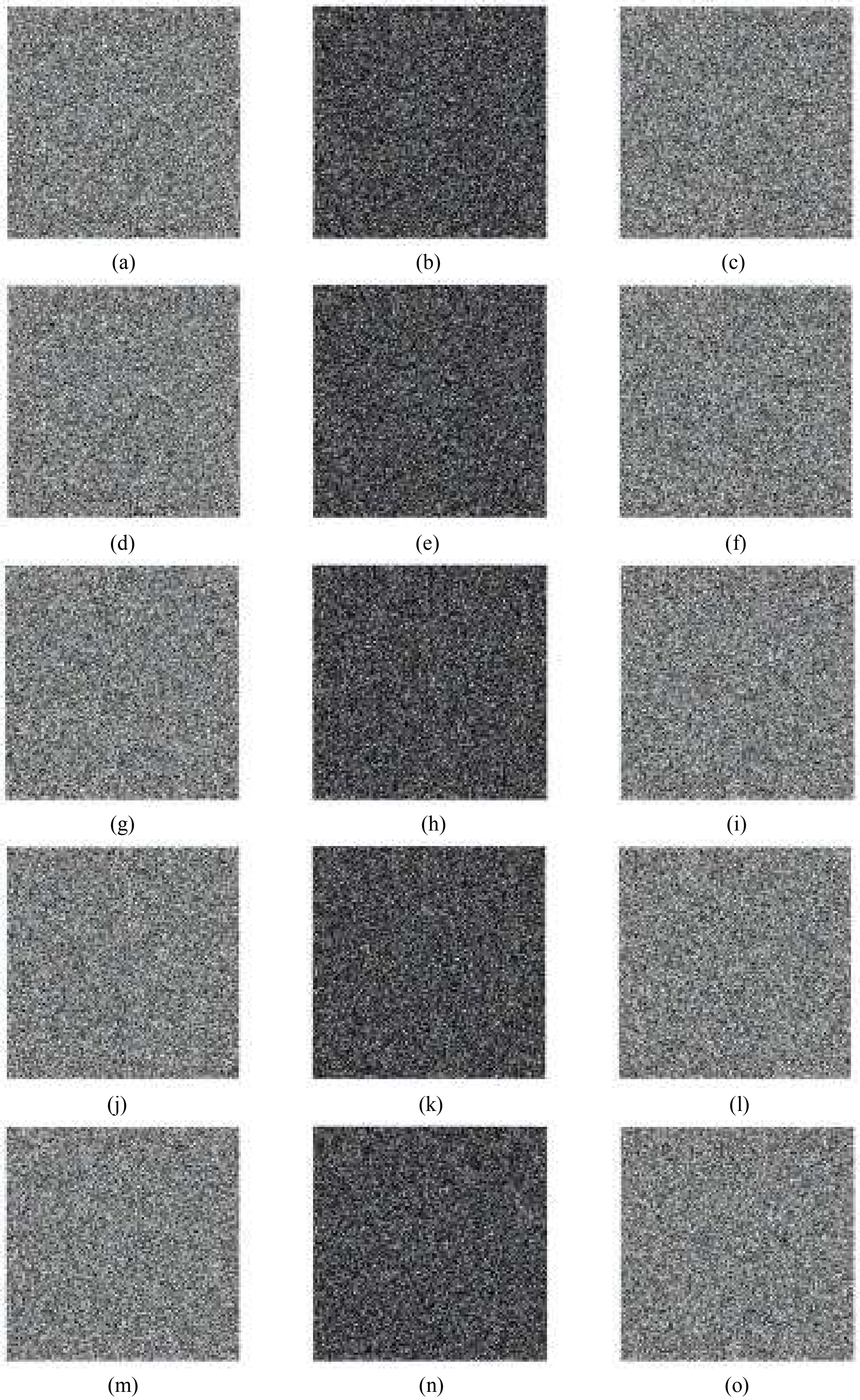
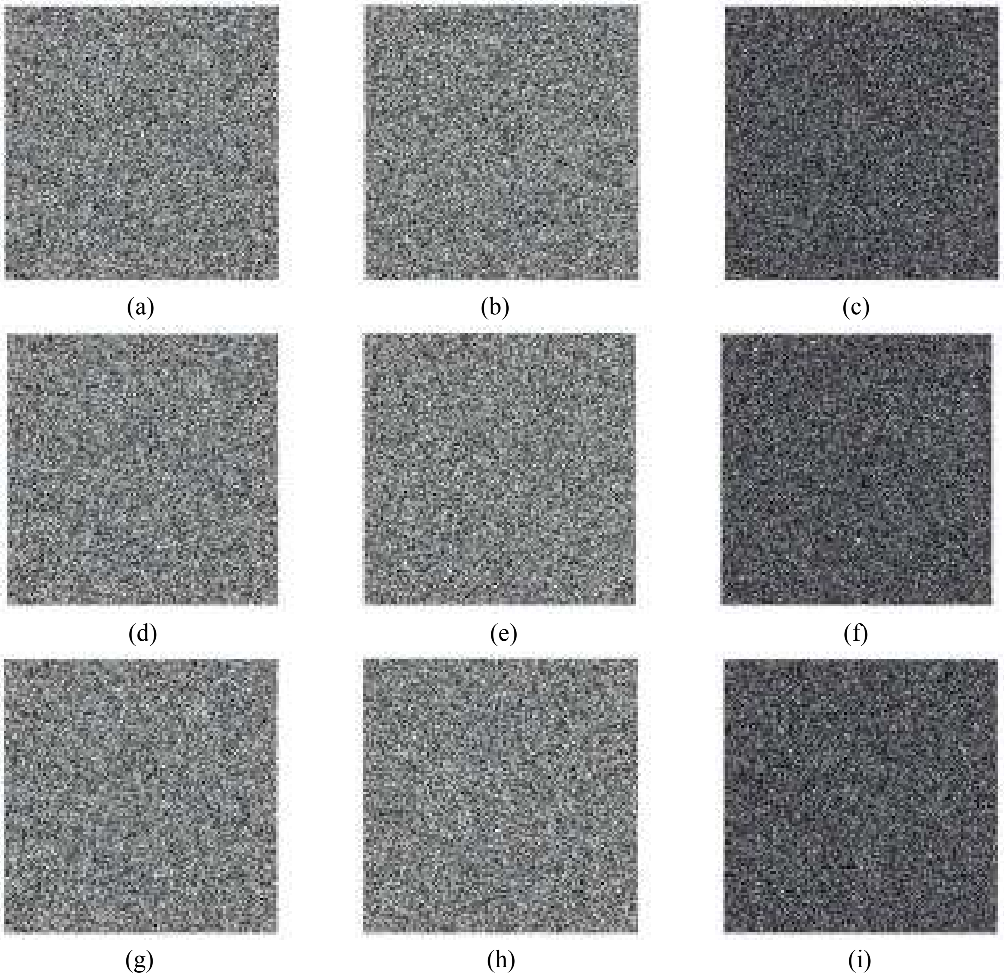


Figure 3: Key sensitivity results of 'Lena' image: (a, d, g, j, m) Cipher images by changing the values of a_0, b_0, η_0, η_1 *and* ϵ respectively; (b, e, h, k, n) Difference images of original cipher images and corresponding changed cipher images; (c, f, i, l, o) Corresponding decrypted images from the changed cipher images.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4: Plaintext sensitivity results of 'Lena' image: (a, d, g) Original cipher images; (b, e, h) Changed cipher images by changing the pixel values at locations (64, 64), (128, 128), and (256, 256) respectively; (c, f, i) Difference images of original cipher images and corresponding changed cipher images.

Statistical Analysis

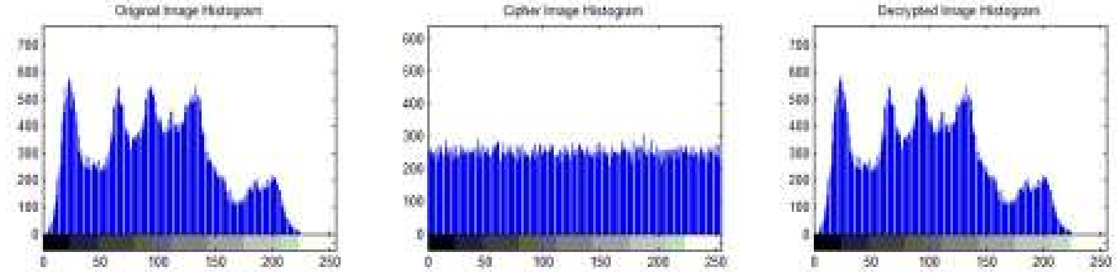
With two analysis, such as gray histogram analysis and correlation coefficient analysis, we perform the statistical analysis of the proposed image encryption scheme.

Gray Histogram Analysis

The histogram of an image is a statistical parameter that quantifies the number of pixels holding a particular gray level value. This distribution in an intelligible image is distinct and not uniform. By encrypting the image we desire to distort the distribution in the original image and make it very uniform. This is depicted in Fig. 5. Fig. 5 (a) shows the histogram of original 'Lena' image, Fig. 5 (b) shows the histogram of encrypted 'Lena' image and Fig. 5 (c) shows the histogram of decrypted 'Lena' image. From the three images, it is obvious that the original and decrypted images have the same distribution while the cipher image has a different and very uniform distribution.

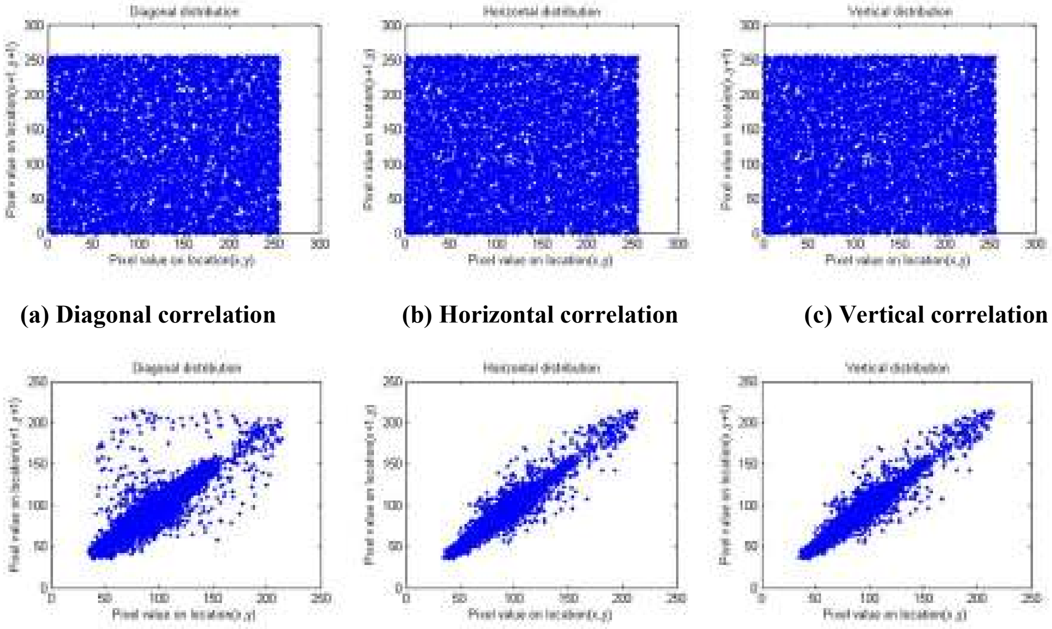
Correlation Coefficient Analysis

From correlation coefficient analysis, we can generate the linear relationship between two adjacent pixels of encrypted image and also for original image. The linear relationship is in terms of correlation that means how the two adjacent pixels are correlated in an image. It simply means that the degree of linear correlation between two adjacent pixels is calculated in correlation coefficient analysis. Let the correlation coefficient is denoted as r whose value ranges from $-1\ to\ 1$. For $r > 0$, indicates positive correlation, for $r < 0$, indicates negative correlation, for $r = 0$, indicates uncorrelated pixels and for $r = 1$, perfect correlation of pixels [Zhang et. al., 2014]. The correlation between two adjacent pixels in an original image is almost close to 1 and for an effective and secure encryption system the correlation between two adjacent pixels is almost close to 0 no matter in horizontal, vertical and diagonal directions [Zhang and Wei, 2013]. Ideally a good encryption scheme must reduce the value of correlation to zero. Fig. 6 shows the correlation distribution of two adjacent pixels for original 'Lena' image and also for encrypted 'Lena' image. Fig. 6 (a), (b) and (c) shows the correlation distribution of pixels of encrypted image along diagonal, horizontal and vertical directions respectively and Fig. 6 (d), (e) and (f) shows the correlation distribution of pixels of original image along diagonal, horizontal and vertical directions respectively. From Fig. 6, it observed that the pixels of original image are closely correlated along horizontal, vertical and diagonal directions indicates strong correlation while there is negligible or very little correlation between adjacent pixels in the cipher image as indicated by a highly scattered plot. So, weak correlation of adjacent pixels existed in an encrypted image.



(a) (b) (c)

Figure 5: Histogram of (a) Original 'Lena' image, (b) Encrypted 'Lena' image and (c) Decrypted 'Lena' image



(a) Diagonal correlation **(b) Horizontal correlation** **(c) Vertical correlation**

(d) Diagonal correlation **(e) Horizontal correlation** **(f) Vertical correlation**

Figure 6: (a), (b), and (c) Correlation distribution of two adjacent pixels of original 'Lena' image; (d), (e), and (f) Correlation distribution of two adjacent pixels of encrypted 'Lena' image

It's necessary to calculate the quantitative analysis of correlation coefficient. The equation for calculating the correlation coefficient quantitatively is

$$r_{ij} = \frac{E\left(\left(x_i - E(x_i)\right)\left(x_j - E(x_j)\right)\right)}{\sqrt{V(x_i)V(x_j)}} \quad \textbf{(4)}$$

where $E(x)$ is the expectation of x, $V(x)$ is variance of x and $r_{ij}$is the value of correlation between pixels with values x_i and x_j.

We randomly selected 4000 pairs of adjacent pixels along all the three directions (diagonal, horizontal, and vertical) from both the encrypted image and the original image, and calculated the correlation coefficient value along the three directions, which are as shown in Table IV. From Table IV we found that the correlation coefficient values for the original image is almost close to 1 along all the three directions and the correlation coefficient values for the encrypted image is almost close to 0, this shows that the adjacent pixels have strong correlation in original image and weak correlation in encrypted image.

Differential Analysis

An essential requirement for any encryption algorithm is that it should resist against differential attack. This can be ensured by designing such a scheme so that no two plaintext images that are distinct produce the same cipher image or introducing a minor change in the plaintext must give rise to a huge difference in the cipher produced. Resistance against differential attacks can be characterised by three parameters namely NPCR, UACI and MAE, each of which is described below.

Table IV: Correlation coefficient results of original image and encrypted image along all the three directions

S. No.	Images	Original Image			Encrypted Image		
		Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical
1	Lena	0.9178	0.9399	0.9693	-0.0035	0.0041	0.0011
2	Cameraman	0.9087	0.9335	0.9592	-0.0055	-0.0020	-0.0001
3	Boat	0.9222	0.9381	0.9713	0.0004	-0.0001	0.0009

NPCR

NPCR or number of pixel change rate computes the percentage of pixels different in the cipher produced by original image from the cipher produced by changing one pixel in the original image. It is calculated as follows:

$$NPCR = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} D_{ij} \times 100\% \qquad (5)$$

where $m \times n$ is the size of the image and D_{ij} is calculated as:

$$D_{ij} = \begin{cases} 0, if\ C_{ij} = C'_{ij} \\ 1, if\ C_{ij} \neq C'_{ij} \end{cases} \qquad (6)$$

C_{ij} is the cipher image produced from original image and C'_{ij} is the cipher image produced by changing one pixel in the original image. For a 256 gray – scale image, the ideal value of NPCR is found to be 99.6094 %. Higher of that NPCR value better is the encryption quality [Chattopadhyay et. al., 2015]. Table V shows the NPCR results of various images by using the proposed method. From Table V, we found that the NPCR results by using the proposed method get closer to the ideal NPCR value.

UACI

Unified average changing intensity is the average intensity difference in gray level of corresponding pixels between cipher of original image (C) and cipher produced by changing one pixel (C'). It is calculated as:

$$UACI = \frac{1}{m \times n} \sum_{k=1}^{m} \sum_{l=1}^{n} \frac{|C_{kl} - C'_{kl}|}{2^{L} - 1} \times 100\% \qquad (7)$$

where L is the number of bits in each pixel.

For a 256 gray – scale image, the ideal value of UACI is found to be 33.4635 %. Higher of that UACI value better is the encryption quality [Chattopadhyay et. al., 2015]. Table V shows the UACI results of various images by using the proposed method. From Table V, we found that the UACI results by using the proposed method get closer to the ideal UACI value.

MAE

Mean absolute error between plaintext image and cipher image is calculated as the average of absolute difference between corresponding gray levels. The formula for calculating MAE is:

$$MAE = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left| C_{ij} - I_{ij} \right| \qquad (8)$$

Higher the value of MAE better is the encryption quality. Table V shows the MAE results of various images by using the proposed method. From Table V, we found that the MAE results are high enough to resist against differential attack.

Entropy

Entropy is a statistical parameter that measures randomness and characterizes 'texture' of the image [https://in.mathworks.com]. The ideal value of entropy for an 8-bit gray scale image is 8 thus a value close to the ideal is desirable. The proposed scheme is able to attain different values of entropy for each random key image and the best values are tabulated in Table VI.

The values obtained shows that proposed scheme is efficient and secure enough. Also we compare this entropy value to the entropy values obtained by various schemes, from that we found, the entropy value of encrypted 'Lena' image by using the proposed encryption scheme is higher than that obtained by various schemes like [Choi et. al., 2016] has entropy 7.8198246494, [Chai, et. al., 2016] has 7.9973, [Guo et. al., 2011] has 7.9974 and [Xing-Yuan and Qian 2014] has an entropy of 7.9970. Therefore our scheme has better randomness of pixels than the other schemes, so our scheme is better than the others.

Table V: NPCR, UACI and MAE results of various images

S. No.	Images	NPCR	UACI	MAE
1	Lena	99.6201	33.4741	103.1086
2	Cameraman	99.6033	33.4958	92.3286
3	Boat	99.5777	33.4423	90.0838

Table VI: Entropy of various images

S. No.	Images	Original Image	Cipher Image
1	Lena (256 × 256)	7.5694	7.9975
2	Cameraman (256 × 256)	7.0097	7.9976
3	Boat (512 × 512)	7.1914	7.9993

CONCLUSION

Proposed an encryption scheme that uses cross-coupled logistic map, DNA XOR operation based Image fusion, Hash function (SHA – 256) and the heart of the scheme i.e. LEA – 256. Such a complex amalgamation ensures an algorithm that is easy to implement yet difficult to decipher. The result analysis shows that the obtained cipher is highly unintelligible and secure. The security analysis shows that the algorithm is sufficiently strong to resist attempts of both simple and differential attack.

REFERENCES

Coppersmith D., 1994. "The Data Encryption Standard (DES) and its strength against attacks," IBM journal of research and development, **38**(3):243-250.

Pub NF., 2001. Advanced encryption standard (AES). Federal Information Processing Standards Publication, **197**:441-0311.

Hong D., Lee J. K., Kim D. C., Kwon D., Ryu K. H. and Lee D. G., 2013. "LEA: A 128-bit block cipher for fast encryption on common processors," In International Workshop on Information Security Applications, Springer International Publishing, pp. 3-27.

Choi J., Seok S., Seo H. and Kim H., 2016. "A fast ARX model-based image encryption scheme," Multimedia Tools and Applications, pp. 1-22.

Guesmi R., Farah M. A., Kachouri A. and Samet M., 2016. "A novel chaos-based image encryption using DNA sequence operation and Secure Hash Algorithm SHA-2," Nonlinear Dynamics, **83**(3):1123-1136.

Zhang Q., Guo L. and Wei X., 2013. "A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system," Optik-International Journal for Light and Electron Optics, **124**(18):3596-3600.

Xiao G., Lu M., Qin L. and Lai X., 2006. "New field of cryptography: DNA cryptography," Chinese Science Bulletin, **51**(12):1413-1420.

Mondal B. and Mandal T., 2016. "A light weight secure image encryption scheme based on chaos & DNA computing," Journal of King Saud University-Computer and Information Sciences.

Wu X., Kan H. and Kurths J., 2015. "A new color image encryption scheme based on DNA sequences and multiple improved 1D chaotic maps," Applied Soft Computing, **37**:24-39.

Zhang Q., Guo L., Xue X. and Wei X., 2009. "An image encryption algorithm based on DNA sequence addition operation," Fourth International Conference on Bio-Inspired Computing (BIC-TA'09), IEEE, pp. 1-5.

https://in.mathworks.com/help/images/ref/entropy.html [online available]

http://sprott.physics.wisc.edu/chaos/2logmaps.htm [online available]

Yunpeng Z., Yu Z., Zhong W. and Sinnott R. O., 2011. "Index-based symmetric DNA encryption algorithm," 4th International Congress on

Image and Signal Processing (CISP), IEEE, **5**:2290-2294.

Stallings W., 2006. Cryptography and network security: principles and practices. Pearson Education India.

Lloyd A. L., 1995. “The coupled logistic map: a simple model for the effects of spatial heterogeneity on population dynamics,” Journal of Theoretical Biology, **173**(3):217-230.

Chai X., Yang K. and Gan Z., 2016. “A new chaos-based image encryption algorithm with dynamic key selection mechanisms,” Multimedia Tools and Applications, pp. 1-21.

Wang Y., Wong K. W., Liao X. and Chen G., 2011. “A new chaos-based fast image encryption algorithm,” Applied soft computing, **11**(1):514-522.

Wang X., Liu L. and Zhang Y., 2015. “A novel chaotic block image encryption algorithm based on dynamic random growth technique,” Optics and Lasers in Engineering, **66**:10-18.

Algredo-Badillo I., Feregrino-Uribe C., Cumplido R. and Morales-Sandoval M., 2013. “FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256,” Microprocessors and Microsystems, **37**(6):750-757.

Xing-Yuan W. and Qian W., 2014. “A fast image encryption algorithm based on only blocks in cipher text,” Chinese Physics B, **23**(3).

Guo J., Peyrin T., Poschmann A. and Robshaw M., 2011. “The LED block cipher,” In International Workshop on Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, pp. 326-341.

Wu W. and Zhang L., 2011. “LBlock: a lightweight block cipher,” In International Conference on Applied Cryptography and Network Security, Springer Berlin Heidelberg, pp. 327-344.

Gilbert H. and Handschuh H., 2003. “Security analysis of SHA-256 and sisters,” In International Workshop on Selected Areas in Cryptography, Springer Berlin Heidelberg, pp. 175-193.

Zhou Y., Bao L. and Chen C. P., 2014. “A new 1D chaotic system for image encryption, Signal processing, **97**:172-182.

Shyam M., Kiran N. and Maheswaran V., 2007. “A novel encryption scheme based on DNA computing,” In 14th IEEE International Conference, Tia, India.

Yuan H. M., Liu Y., Lin T., Hu T. and Gong L. H., 2017. “A new parallel image cryptosystem based on 5D hyper-chaotic system,” Signal Processing: Image Communication.

Wu X., Wang D., Kurths J. and Kan H., 2016. “A novel lossless color image encryption scheme using 2D DWT and 6D hyper – chaotic system,” Information Sciences, **349**:137-153.

Zhang J., Fang D. and Ren H., 2014. “Image encryption algorithm based on DNA encoding and chaotic maps,” Mathematical Problems in Engineering.

Zhang Q. and Wei X., 2013. “RGB color image encryption method based on Lorenz chaotic system and DNA computation,” IETE Technical Review, **30**(5):404-409.

Chattopadhyay C., Sarkar B. and Mukherjee D., 2015. “Encoding by DNA relations and randomization through chaotic sequences for image encryption,” arXiv preprint arXiv: 1505.01795.