Multimed Tools Appl (2018) 77:24701–24725
https://doi.org/10.1007/s11042-018-5669-2

A new image encryption algorithm based on heterogeneous chaotic neural network generator and dna encoding

Gururaj Maddodi[1] · Abir Awad[2] · Dounia Awad[3] · Mirna Awad[3] · Brian Lee[3]

Received: 15 June 2017 / Revised: 19 December 2017 / Accepted: 15 January 2018 /
Published online: 22 February 2018

Abstract This paper presents a new combined neural network and chaos based pseudo-random sequence generator and a DNA-rules based chaotic encryption algorithm for secure transmission and storage of images. The proposed scheme uses a new heterogeneous chaotic neural network generator controlling the operations of the encryption algorithm: pixel position permutation, DNA-based bit substitution and a new proposed DNA-based bit permutation method. The randomness of the generated chaotic sequence is improved by dynamically updating the control parameters as well as the number of iterations of the chaotic functions in the neural network. Several tests including auto correlation, 0/1 balance and NIST tests are performed to show high degree of randomness of the proposed chaotic generator. Experimental results such as pixel correlation coefficients, entropy, NPCR and UACI etc. as well as security analyses are given to demonstrate the security and efficiency of the proposed chaos based genetic encryption method.

Keywords Image security · Encryption · Chaos theory · Neural networks · DNA encoding

✉ Gururaj Maddodi
g.maddodi@uu.nl

Abir Awad
a.awad@ljmu.ac.uk

Dounia Awad
dounia.awad@gmail.com

Mirna Awad
m.awad@research.ait.ie

Brian Lee
blee@ait.ie

[1] Department of Information and Computing Science, Utrecht University, Utrecht, The Netherlands

[2] Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool, UK

[3] Software Research Institute, Athlone Institute of Technology, Athlone, Ireland

1 Introduction

Due to the advancements in networking and multimedia coding technology, media such as images are commonly stored and shared over the Internet. This makes them vulnerable to be used for malicious purposes. Hence, image security and encryption has become a much researched area to ensure confidentiality and stop un-authorised access to the digital content. Traditional symmetric ciphers such as Advanced Encryption Standard (AES) [7] are designed with good confusion and diffusion properties. However, standard encryption, methods such as AES seem not to be suitable to cipher data like images [15]. Since 1990, chaotic systems have gained a lot of interest in the field of cryptography due to their desirable properties such as high sensitivity to initial conditions, ergodicity and pseudo randomness. Chaotic systems have also demonstrated great potential for information security especially image encryption [1, 6, 10, 12, 14, 20, 22].

More recently, researchers have showed interest in combining neural network and chaos to develop enhanced encryption algorithms [4, 5, 13, 18, 19]. The authors in [13] have trained the neural network with chaos to replace the chaotic generator in the encryption algorithm. In [19], the chaotic map is used as transfer function for each neuron. They then proposed to use this generator in a stream encryption algorithm where the image is considered as a 1-dimensional array and each bit is encrypted using a binary chaotic output. In [5], neural network biases and weights are adjusted using a chaotic sequence, which would make the output of the neural network very random. The output of the neural network is then used to encrypt image using AES. However, this encryption method uses simple operations such as bit XOR which makes the proposed mechanism easy to break. In addition, the encryption time and computation processing in this case is also quite high.

As well as neural networks, DNA-based chaotic image encryption (and decryption) has been proposed in several works [9, 16, 17, 23, 24]. In Liu et al. [16], each pixel is encoded into four nucleotides and then transferred to their base pairs at random times using a chaotic generator. In Zhang et al. [23], chaos is used to disturb the positions and the values of the pixels and DNA encoding using quaternary code rules to encode the pixel values. In Enayatifar et al. [9], chaotic map function and DNA rules are used to determine a DNA mask which is improved by evolutionary algorithm and used for image encryption. In [17], a cryptanalysis system using DNA rules is proposed in order to break a cryptographic system. The authors in [24] studied security aspects of symmetric ciphers using DNA coding.

In this paper, we propose to combine neural network, chaotic systems and complementary DNA rules in one block encryption algorithm for secure image transmission and storage. Neural network and chaotic functions are used to implement a new heterogeneous chaotic pseudo-random sequence generator. The resulting chaotic sequences are then used as inputs for the proposed encryption operations including pixel position permutation, DNA-based pixel bit permutation and substitution.

The structure of this paper is as follows: Section 2 provides background information and reviews some of the related works on image encryption using neural network and DNA rules. Section 3 introduces the proposed chaos and neural network based pseudo-random sequence generator. Section 4 describes the proposed DNA rules based image encryption algorithm. The simulation results and security analysis of the proposed generator and encryption scheme are given in Section 5. Section 6 gives a conclusion of the results and security features of the proposed encryption system.

2 Background and related work

In this section, we briefly introduce the chaotic functions the most often used in the literature and which we have chosen to use in this paper. Later, we describe some of the most recent chaotic encryption algorithms based on neural network and DNA encoding.

2.1 Chaos and chaotic maps

Several chaotic maps have been proposed in the literature. In this section, we briefly introduce three chaotic maps: Logistic map, Piece Wise Linear Chaotic Map (PWLCM), and Logistic-Tent system.

1) **Logistic Map** is a simple one-dimensional chaotic map and it is described in (1) as follows:

$$x_{n+1} = p\, x_n\,(1 - x_n) \tag{1}$$

where p is the control parameter in the range [3.58, 4]. x_n and x_{n+1} are the outputs/states of the chaotic map at iterations n and n+1 and both are in the interval of (0, 1).

2) **Piece-Wise Linear Chaotic Map** or PWLCM is a map composed of multiple linear segments and it has better balance properties then the Logistic map. It is defined by (2) as follows:

$$\begin{aligned} x_{n+1} &= F\,[x_n] \\ &= \begin{cases} x_n/p & \text{if } 0 \le x_n < p \\ (x_n - p)\,/\,(0.5 - p) & \text{if } p \le x_n < 0.5 \\ F\,[1 - x_n] & \text{if } 0.5 \le x_n < 1 \end{cases} \end{aligned} \tag{2}$$

where p is the control parameter in the range (0, 0.5). x_n and x_{n+1} are the outputs/states of the chaotic map at iterations n and n+1 and both are in the interval of (0, 1). In the proposed generator, the control parameter is restricted to (0.2, 0.3) which produces better chaotic properties compared to the whole range.

3) **Logistic-Tent System** or (LTS) [25], combines the Logistic and Tent (piece-wise linear) maps into one map as described in (3) below:

$$x_{n+1} = \begin{cases} (px_n\,(1 - x_n) + (4 - p)\,x_n/2) \bmod 1 \\ \text{if } x_n < 0.5 \\ (p\,x_n\,\,(1 - x_n) + (4 - p)\,(1 - x_n)\,/2) \bmod 1 \\ \text{if } x_n < 0.5 \end{cases} \tag{3}$$

where p is the control parameter in the range [0, 4] for chaotic behavior, x_n and x_{n+1} are the outputs/states of the chaotic map at iterations n and n+1 and both are in the interval of (0, 1).

2.2 Neural network based encryption

In Singla et al. [19], a neural network based chaotic generator is proposed. The neural network parameters such as weights, biases and control parameters were initialized by a 1-D Cubic chaotic map. The neurons of the neural network are assigned PWLCM as their transfer function. The neural network takes 64-bits (eight 8-bit inputs) as the key for the first cycle of its operation and generates a single bit of the pseudo-random sequence on every cycle. After every cycle of the network, the control parameter of each layer is updated by the output of the same layer in order to keep the control parameters close to 0.5 (for best chaotic behavior). The generated sequence had good randomness properties as demonstrated by the tests. The encryption process is implemented by XORing the generated sequence with the image bitstream. There is a shortcoming of the proposed generator here that it produces just a single bit at the output layer after a complete neural network cycle with four layers. We believe it can be improved to reduce computation overhead. Also, the encryption method used is just to XOR the chaotic bits with the image pixel bits which is not very secure.

Kassem et al. [13] use artificial neural networks as a source of perturbation for their proposed generator. The neural network scheme is first trained with PWLCM datasets. Then, it is used to replace the PWLCM function to generate the pseudo random inputs to the system periodically. The proposed neural network perturbation approach enlarges the key space by means of the structure of the neural network (number of layers, activation functions, weights, biases). Adding perturbation increases the cycle's length and thus avoids the problem of dynamical degradation of the chaotic orbit.

2.3 DNA based encryption

In Awad et al. [2], pixel bit substitution method based on DNA complementary rule is proposed. Authors use DNA complementary rules to calculate new values for the image pixels. Initially, the pixel value is decomposed into four "2-bits" values, and to each of them is assigned a DNA letter (A, T, C and G). Then, DNA letters are converted into binary values using a binary coding rule e.g. A=00, T=01 etc. Then, to every letter, a complement is assigned and denoted by C(X) e.g. C(A) = T, C(T) = C etc. A DNA transformation rule is applied on these letters for a number of iterations which, in its turn, calculated using the chaotic generator. The six allowed complementary transformations are shown below,

$$\begin{aligned}
Rule_0 &: (AT)(TC)(CG)(GA)\\
Rule_1 &: (AT)(TG)(GC)(CA)\\
Rule_2 &: (AC)(CT)(TG)(GA)\\
Rule_3 &: (AC)(CG)(GT)(TA)\\
Rule_4 &: (AG)(GT)(TC)(CA)\\
Rule_5 &: (AG)(GC)(CT)(TA)
\end{aligned}$$

Out of 8 bit chaotic output, the first 3 bits of the chaotic value are used to select the complementary transformation rule. Of the remaining 5 bits, the 4^{th} and 5^{th} are used to compute the number of iterations of the selected rule for the first DNA letter of the selected pixel block using equation as below:

$$it\,(1) = mod\left(Ch/2^3, 4\right) \tag{4}$$

where *it* is the no. of iterations and *ch* is the chaotic value. Similarly, 5^{th} and 6^{th} bits are used for the second DNA letter and so on. The transformation works by changing a DNA letter to its complement for each iteration: for example A becomes T after one iteration, then T becomes C for the second iteration and so on. After the iterations of the selected rule on each DNA letter, the resultant DNA letter is then converted into its binary format and substituted in place of the old bits. This process is applied for every pixel of the image to get substituted image.

In Zhang et al. [23], DNA rules based encoding is used to scramble the image pixel values. The image is mapped to a quaternary sequence and a randomly generated key selects the DNA encoding rule. Then, a DNA matrix of size (M, 4xN) is generated, where MxN is the size of the image. A chaotic map is used to generate a chaotic sequence C of length MxNx4 and is transformed to a matrix of size (M, 4xN). Then, for a DNA value, at position (i, j), in the DNA matrix, following rules are applied:

- if $C(i, j)$=0, then no change is done to the value
- if $C(i, j)$=1, then DNA complement is performed and XOR is done
- if $C(i, j)$=2 then DNA value is complemented
- if $C(i, j)$=3 then DNA complement is performed and NXOR (Not Exclusive-Or) is done.

After performing this operation, the new DNA sequence matrix obtained is the encrypted image.

Though these two encryption schemes make use of chaos for encryption, they do not use neural network for generating chaotic sequences which will provide much better performance due to its architecture having several parameters (weights, biases etc). Hence, we propose an encryption scheme which combines chaos generation using neural network and DNA encoding for encryption. In the next section, we present our proposed neural network and chaos based pseudo-random sequence generator.

3 Proposed generator

In this section, we present our proposed neural network based chaotic pseudo-random generator. As mentioned earlier, a neural network architecture is used for the generator implementation. Our proposed generator architecture is heterogeneous i.e. more than one chaotic map is used. Figure 1 depicts the proposed generator.

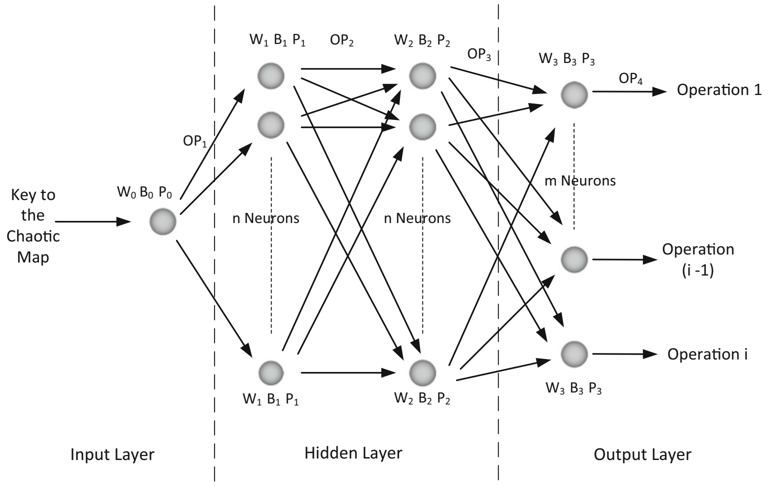

Fig. 1 Proposed neural network generator architecture

The proposed neural network generator contains four layers: one input layer, two hidden layers and an output layer. Each neuron of the neural network generator uses a chaotic map equation as its transfer function. In the proposed architecture, the input layer has a single neuron and receives a part of the 64-bit encryption key. The key is divided into two 32-bit parts: the first part is used as input for the first cycle of the neural network operation and the second part is used to initialize the neural network parameters. The two hidden layers contain n_{hl} neurons while the output layer contains m_{ol} number of neurons (depending on the number of operations to be performed).

A heterogeneous chaotic system is implemented by using different chaotic maps, as transfer functions, alternately in the neural network layers. For example, if the first layer is using Logistic map, then the second layer will have PWLCM and so on. The weight matrices of the four layers, i.e. 0 to 3, (see Fig. 1) are W_0, W_1, W_2, and W_3, and are of the sizes 1x1, n_{hl}x1, n_{hl}xn_{hl}, and m_{ol}xn_{hl} respectively. The bias matrix of the layers are B_0, B_1, B_2, and B_3. The control parameter matrix of the chaotic map transfer function are P_0, P_1, P_2, and P_3. Both B_0-B_3 and P_0-P_3 are of the sizes 1x1, n_{hl}x1, n_{hl}x1 and m_{ol}x1 in order. The 'm_{ol}' outputs of the neural network generator can be used for the different operations of the encryption algorithm.

To initialize the neural network generator (weight, bias, control parameter matrices and number of iterations for the first neural network cycle), a 1-D Cubic map is used. The second 32-bit part of the 64-bit key is used as the initial condition for the Cubic map described in (5) below:

$$x_{n+1} = \lambda\, x_n \left(1 - x_n^2\right) \tag{5}$$

where x_n is the initial condition for the first cycle or the previous state for further cycles of the chaotic function, x_{n+1} is the current state of the chaotic function. Both x_n and x_{n+1} are in the range of (0, 1). λ is the control parameter of the chaotic function with a value of 2.59. The map is run initially for 50 iterations in order for the chaotic nature to emerge. Thereafter, its outputs are used to initialize the neural network parameters.

In our paper, we use four outputs, i.e. m_{ol}=4 and then four chaotic sequences are generated:

- OP_{4w} - is used for pixel position permutation (PPP)
- OP_{4x} - is used for pixel bit permutation (PBP)
- OP_{4y} - is used for pixel bit substitution (PBS)
- OP_{4z} - is used for updating the number of iterations of the chaotic transfer functions of all the neurons

Figure 2 shows the outputs of the output layer (OP_{4w}, OP_{4x}, OP_{4y}, and OP_{4w}) assigned to the operations (PPP, PBS, PBP and updating the number of iterations) respectively. All the outputs of the output layer are XORed and the obtained value is fed back as input after each cycle of the neural network operation. The control parameter of the chaotic map, i.e. transfer function, of each neuron is updated after every cycle of the neural network using the output of the same neuron. To do so, the output of each neuron of every layer is transformed to the range of the control parameter of the chaotic map used for the neuron using (6) as follows:

$$P_{new} = startingValue + (range \times OP) \tag{6}$$

Where *startingValue* is the value from which the control parameter range begins, *range* is the difference between the lowest and the highest values of the control parameter and *OP* is the output of the neuron. For example, for a neuron using the Logistic map, *startingValue* is 3.58 and *range* is (4 - 3.58) = 0.42. The transformed output is used to replace the old

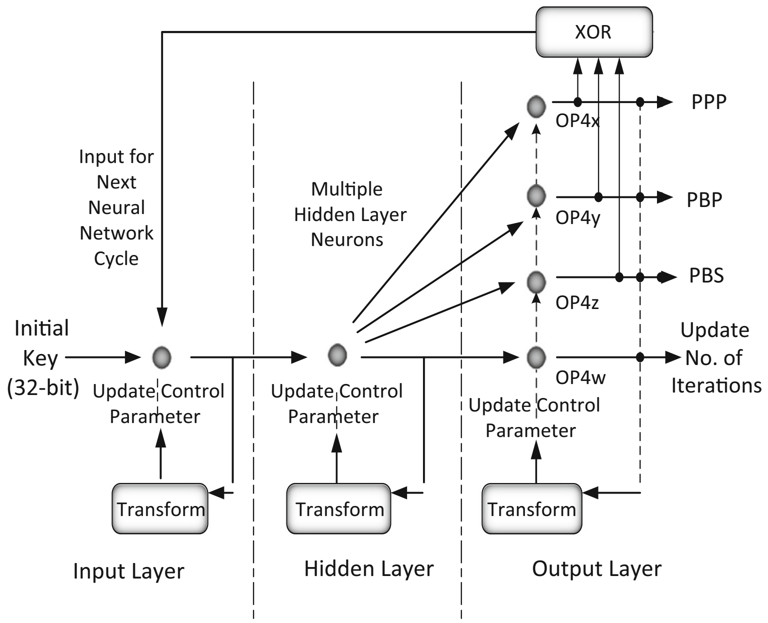

Fig. 2 Execution of the proposed generator

control parameter of the neuron and the new value will be used in the next cycle of the neural network.

Subsequently, in this paper, the outputs of the generator are named as per operation that is performed using them, such as PPP sequence for operation pixel position permutation etc. as has been shown in Fig. 2.

4 Proposed image encryption algorithm

In this section, we describe our proposed image encryption algorithm and a new DNA rules based pixel bit permutation algorithm. Figure 3 shows the architecture of the proposed image encryption algorithm. As it can be seen, the outputs from the generator are being used for different operations of the encryption algorithm.

The encryption operation can be divided into two distinct blocks. The first block is the pixel position permutation (PPP-box) block which performs the diffusion of the image pixels, and the second block is the substitution and permutation block, or the SP-box, which is performed at pixel bit level and performs the confusion and diffusion operations.

4.1 Pixel position permutation

In the PPP-box, pixel positions in the image are shuffled by identifying the initial position and the target pixel positions in the image. The output PPP from the generator is used to calculate the initial and target positions.

For an $N*N$ image, the first N bits of the PPP sequence is used to calculate the initial position of the random pixel to be permuted. The last N bits are used to calculate the target

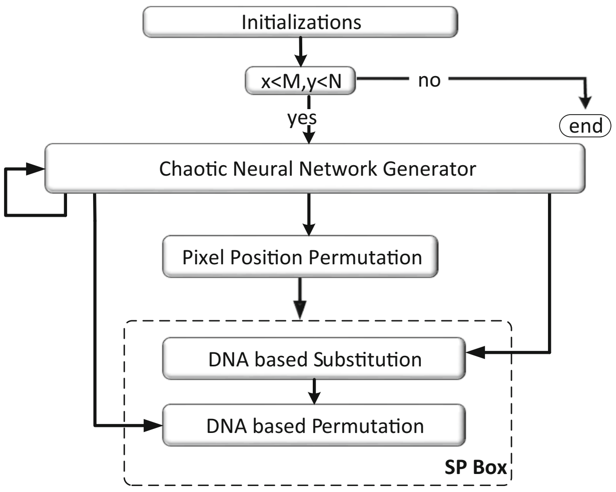

Fig. 3 Proposed encryption algorithm

position for the selected pixel. This process is repeated N*N number of times ensuring that all the pixels' positions are permuted. If a pixel in initial position is already permuted or a target position is already filled, the nearest available position is selected. Figure 4 depicts the PPP operation where a value in the PPP sequence is used to shift a pixel in position 3 to position *N*-1.

4.2 SP-box operation

Input to the SP-box is the pixel position permuted image obtained after the PPP-box. The SP-box contains two separate operations which are performed on per pixel basis of the input image, first chaotic output of PBS and PBP is used for pixel 1, second output PBS and PBP

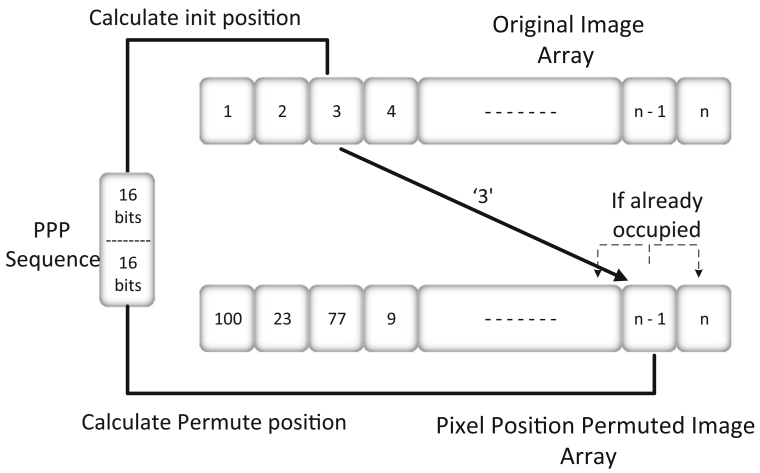

Fig. 4 Pixel position permutation operation of the proposed encryption algorithm

for pixel 2, and so on. First, the chosen image pixel is subjected to the pixel bit substitution operation. Then, the bit substituted pixel undergoes bit permutation operation. In the pixel bit substitution and pixel bit permutation operations, the 32 bits of the chaotic outputs PBS and PBP respectively are used by divided into four bytes as follows:

$$Ch = \{Ch1, Ch2, Ch3, Ch4\}$$

and each part is used for the four iterations, i.e. *i*=1 to 4, of the SP-box operation for the said pixel.

For bit substitution, we use the DNA based method from [2] that was described earlier in the paper in Section 2. Let us assume that *i*=4, hence we are using Ch4 and the let the output pixel be P_{PBS} after bit substitution. The P_{PBS} will be used as the input for the proposed pixel bit permutation operation that is described in the next subsection.

4.3 Proposed DNA-based pixel bit permutation

In this section, we propose a new bit permutation algorithm based on complementary DNA rules. In the bit permutation operation, we permute the bit positions of each pixel. As explained earlier, SP-box operation is iterated four times. For each of the four iterations of the SP-box, one of the four bytes *Chi=Ch1,...,Ch4*, where *i* = (1, 4), of the 32-bit PBP value is used.

Since we are considering 4^{th} iteration, i.e., *i*=4, of the SP-box, the input is P_{PBS}. We start by considering an array *Arr* whose values represent the positions of each bit of P_{PBS} bit substituted pixel. *Arr* can be represented as:

$$Arr = [0, 1, 2, 3, 4, 5, 6, 7]$$

After applying DNA-based bit permutation operation, we get a new permuted pixel P_{PBP}. In subsection below, we explain proposed DNA-based pixel bit permutation.

4.3.1 Pixel bit permutation operation

Within an iteration *i* of SP-box, bit permutation operation is repeated four times, which we represent by *t* where *t* = 1 to 4. We represent the 8 bits of the chaotic output as,

$$Chi = \{Chi_{b0}, Chi_{b1}, ..Chi_{b7}\}$$

During the i^{th} iteration of the SP-box, for the t^{th} repetition of the permutation operation, the byte *Chi* is decomposed as follows: the first three chaotic bits $\{Chi_{b((t-1)mod8)}, Chi_{b(tmod8)}, Chi_{b((t+1)mod8)}\}$ are used to find the initial selected position and is identified as value *V* in the array *Arr*. The bits in *V* are then used to get the DNA letters on what the DNA transformation rules are applied. The following three bits $\{Chi_{b((t+2)mod8)}, Chi_{b((t+3)mod8)}, Chi_{b((t+4)mod8)}\}$ are used to choose the $Rule_r$, i.e. the DNA rule number to apply. The last two bits $\{Chi_{b((t+5)mod8)}, Chi_{b((t+6)mod8)}\}$ are used to determine the number of times n_{rule} the DNA rule $Rule_r$ is applied.

We are considering *i*=4 and let us say *t*=1 and Chi={0,1,0,1,0,1,0,1} as shown in Fig. 5. Then, we have *V*={1,1,0}=2, $Rule_r$={1,0,0}=5, and n_{rule}={0,1}=1(+1)=2. We start by assigning part of *Chi*, i.e. *w*=$\{Chi_{b((t-1)mod8)}, Chi_{b(tmod8)}, Chi_{b((t+1)mod8)}\}$. We take last two bits of *W*, i.e. {1,0}, and convert it into a DNA letter, i.e C. Then, we apply the DNA $Rule_4$, which gives new DNA letter A [2]. Then, we convert the new DNA letter A to binary format (which gives us {0,0}) and substituted to last two bits of *W* which gives new *W*={1,0,0}. For the next iteration of n_{rule}, we choose first two bits of *W*, i.e. {1,0}, which

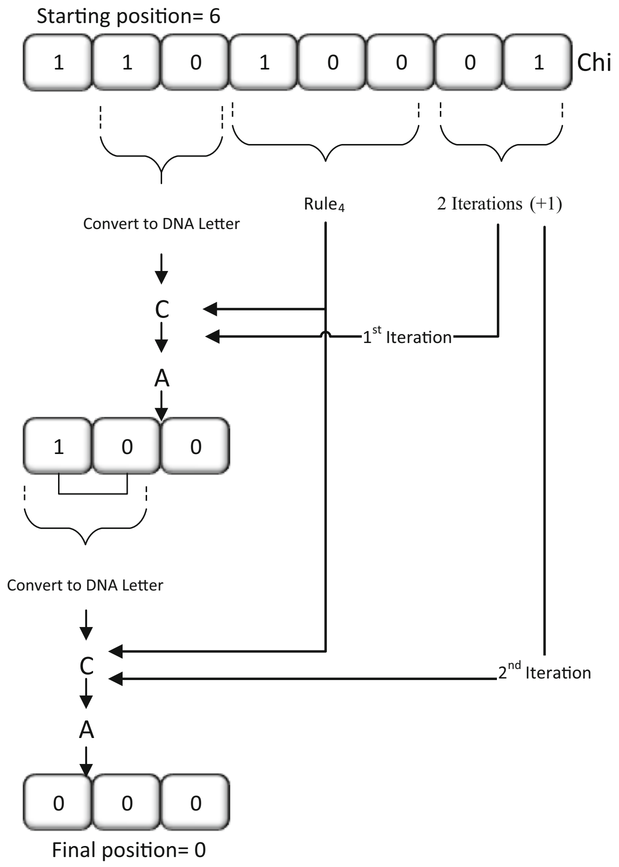

Fig. 5 Pixel bit permutation operation of the proposed encryption algorithm

gives us C and applying the DNA $Rule_4$ gives new letter A. The new DNA letter is converted to binary, i.e. {0,0}, and substituted at first two bits of W, we get new W={0,0,0}. In the example n_{rule} is 2, hence we stop at this point and the new value we of W is assigned as X. In the example V is 6 and X is 0. We then swap bit at V with bit at X in the array Arr to get Arr'.

$$Arr' = [6, 1, 2, 3, 4, 5, 0, 7]$$

We continue this process alternately selecting bits of W for n_{rule} number of times. For next repetition, i.e. t=2, by assigning t=2, we get V={1,0,1}=5, $Rule_r$={0,0,0}=$Rule_0$, and n_{rule}=2(+1)=3. Following the method, we get X={0,1,1}=3. By performing permutation of V and X, we get Arr'',

$$Arr'' = [6, 1, 2, 5, 4, 3, 0, 7]$$

The operation described above is repeated till $t = 4$. At this point, we have Arr''''=[6,1,2,5,4,3,0,7]. The final bit permuted pixel P_{PBP} by shuffling bits to positions of P_{PBS} as indicated by Arr''''.

5 Image decryption

For decryption of the encrypted image, the same 64-bit key, initially used for encryption, must be used. Decryption follows the reverse process of that of encryption, i.e. the encrypted image first undergoes SP-box and then the pixel position permutation is applied. In the SP-box also the pixel bit permutation operation must be performed prior to the pixel bit substitution operation. This is depicted in the Fig. 6 where the same key is used for the neural network generator parameter initialization and for generating chaotic sequence.

As seen in Fig. 6, the SP-box operation is first. Similar to encryption, we divide the 32-bit chaotic outputs into four 8-bit values. The difference in decryption is that, we reverse order of chaotic sequence for each iteration of SP-box, i.e. *i=(4, 1)*. It is as shown below,

$$Ch = \{Ch4, Ch3, Ch2, Ch1\}$$

Let us consider that we are have P_{PBP} and for decryption, we should have Chi={0,1,0,1,0,1,0,1}. Similar to encryption, bit permutation in decryption works by assuming an array as,

$$Arr_{decr} = [0, 1, 2, 3, 4, 5, 6, 7]$$

After 4 repetitions of bit permutation, we get the array Arr_{decr}=[6,1,2,5,4,3,0,7] which is same as *Arr””*. Therefore, by swapping bits of P_{PBP} to positions indicated by Arr_{decr}, we get P_{PBS}. The pixel P_{PBS} then undergoes decryption in bit substitution. Please refer [2] for decryption technique for pixel bit substitution. Again, similar to encryption, SP-box is repeated for four iterations.

The output from SP-box after four iterations is the original image pixel. After SP-box is applied to all the pixels, we obtain the actual values of all the pixels, but their positions are still permuted. Applying PPP-box in reverse will give us the original image with each pixel in their proper position. For PPP-box, the initial and target positions in the permutation operations are reversed, i.e. for an N*N image the first N bits are used to determine the target pixel position and the last N bits are used to find the initial position of the pixel.

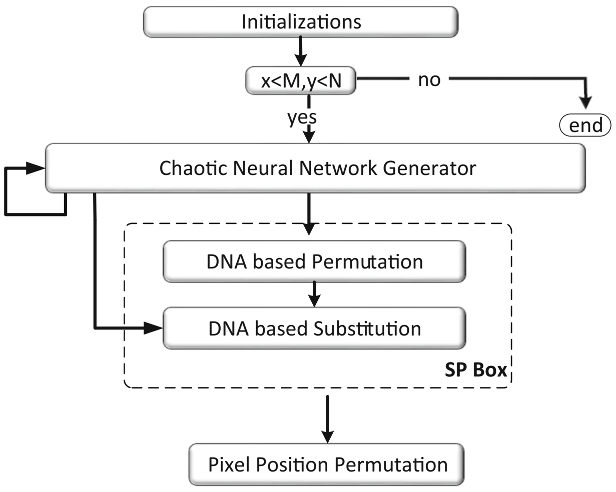

Fig. 6 Proposed decryption algorithm

6 Simulation results and security analysis

Experimental results are given in this section to demonstrate the efficiency of our proposed generator and encryption algorithm. The implementation was developed on Matlab 2012a running on Windows 7 machine with Intel Core i7-3770 3GHz processor and 12GB of RAM.

6.1 Testing the proposed generator

Our proposed generator was tested with several combinations of the chaotic maps that are most used in the literature, such as PWLCM, Logistic, LTS [25] etc. The effect of the number of neurons in the hidden layers and the number of iterations of the chaotic maps are also tested in order to choose the best parameters for the neural network generator.

6.1.1 Choosing generator parameters

To test the randomness properties of the generator with different chaotic map combinations, we have used NIST (National Institute of Standards and Technology) statistical tests [3]. NIST test suite is a statistical package that was developed to test the randomness of arbitrary long binary sequences produced by either hardware or software based cryptographic random or pseudo-random number generators. The test suite consists of 188 statistical tests for evaluating randomness. These tests focus on identifying a variety of different types of non-randomness that could exist in a binary sequence. NIST suite determines if there is any non-randomness or periodicity in the sequence being tested, and decides if the sequence has good enough randomness.

NIST tests were applied on sequences generated from 100 different keys each of length 10^6 bits. Note that to generate a sequence of length 10^6 bits, the generator took approximately 661.35 secs.

Table 1 shows the number of failed tests (out of a total of 13 tests) for a common set of keys for several map combinations. It can be seen from Table 1 that, Logistic and PWLCM maps combination passes all the tests. This showed that the approach of using heterogeneous chaotic generator gives the best randomness.

Table 2 shows the effect of the number of neurons in the hidden layers on the randomness of the generated chaotic outputs when Logistic and PWLCM are used. As we can see, increasing the number of neurons gives better randomness and the generator having 8 neurons passes all the tests.

Table 3 shows the effect of number of iterations of the chaotic maps in the neural network on the NIST results, for a fixed combination (Logistic + PWLCM) and the number of hidden layers (8 neurons).

Table 1 Number of Failed Tests in NIST Test Suite Using Different Chaotic Map Combinations With 8 Neurons in the Hidden Layers

Map combinations	PPP	PBS	PBP
Logistic Only	13	13	13
PWLCM Only	1	1	1
LTS Only	8	7	8
Logistic + PWLCM	0	0	0

Table 2 Number of Failed Tests of NIST Test Suite using different No. of neurons in the Hidden layers of the chaotic generator

No. of hidden layer neurons	PPP	PBS	PBP
3	6	6	6
5	1	2	4
8	0	0	0

As can be seen the randomness increases when the number of iterations increases and the generator passes all the tests when five iterations were used.

Based on the above results of NIST tests using different parameters for the generator, we chose to use Logistic and PWLCM combination, with 8 neurons in the hidden layers and 5 iterations of the chaotic maps.

6.1.2 NIST test for the chosen generator

The results of NIST test for the three chaotic outputs of the generator (PPP, PBS, PBP), with the above chosen parameters, for hundred different sequences each of length 10^6 bits are shown in the Table 4.

It can be seen from Table 4 that all the 3 sequences passed all the tests of NIST test suite. It also means that 3 different sequences can be generated simultaneously for different encryption operations.

Figure 7 shows the chaotic signal for the first output PPP of our proposed generator.

6.1.3 Autocorrelation

The autocorrelation test performed on the generated sequence PPP is shown in Fig. 8. As we can see, the sequence does not contain any periodicity and produces similar result of a random signal with zero value for non-zero time shift.

6.1.4 0/1 balance test

The 0/1 balance property is the ratio of the number of bits with a value ‘0’ to that of number of bits with value ‘1’. A value of 50 for this property means, good randomness, i.e. the sequence has equal number of “0s” and “1s”.

For this test, we calculated the 0/1 balance for hundred sequences (each of length 10^6 bits) using different keys. The average values for the three chaotic outputs of our generator are shown in Table 5.

Table 5 shows that the generated sequences have good 0/1 balance properties comparing to the results obtained in [13] and [19]. Note that the best results obtained, for the 100 sequences tested, are 50, 50.001 and 50 for the three sequences PPP, PBS and PBP.

Table 3 Number of failed tests of NIST test suite using different No. of iterations for the chaotic maps in the generator

No. of iterations of chaotic map	PPP	PBS	PBP
2	1	3	3
4	1	1	1
5	0	0	0

Table 4 NIST Test Results for 100 Sequences of Length 106 Bits with 100 Different Keys

NIST test	PPP	PBS	PBP
Frequency	100	98	98
Block Frequency	98	99	99
Cumulative Sums	100	98	98
Runs	98	99	99
Longest Runs	99	99	100
Rank	99	100	100
FFT	96	98	97
Non-overlapping Templates	100	100	100
Overlapping Templates	100	97	99
Universal	100	99	99
Approximate Entropy	98	99	99
Random Excursions	100	100	100
Random Excursions Variant	100	100	100
Serial	100	100	99
Linear Complexity	100	99	100

6.1.5 *Attractor test*

Phase space is the plot of a system's previous state to its new state. It is also called attractor for the reason that the plot shows if the system is being attracted to a steady state. It can indicate randomness in a system if the system does not seem to have a steady state region. Figure 9 shows the 2D phase space plot for the proposed generator. From the plot, it can be seen that there is no steady state region in the plot, which signifies that the signal generated is highly random and chaotic. It also shows that the system has an inherent perturbation mechanism due to neural network architecture and dynamic control parameter variation which prevents the system from going into a steady state.

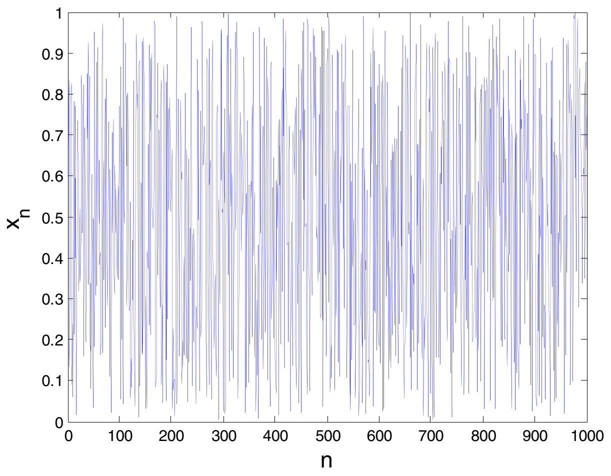

Fig. 7 State sequence of the proposed generator for one of the sequences

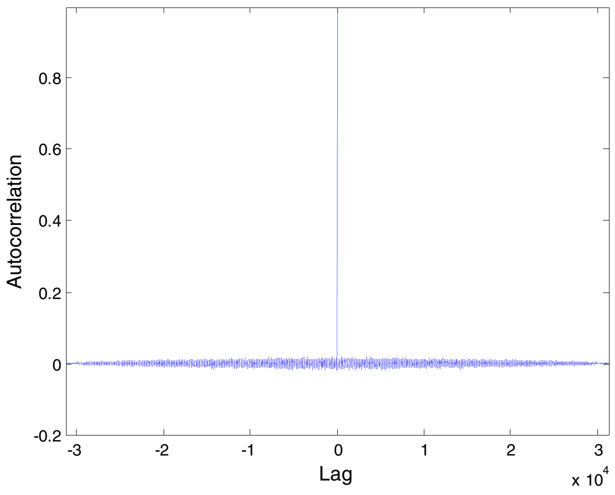

Fig. 8 Autocorrelation function of the generated chaotic sequence PPP

6.1.6 *Histogram analysis*

The histogram of the generated sequence is shown in Fig. 10. As we can see, the chaotic values in the range of (0, 1) are distributed equally, which means that the generated sequence is highly random.

6.1.7 *Key sensitivity test*

In this test, two different keys with just 1-bit difference were used to generate two sequences using our proposed generator. We have used PPP sequence to perform this test but similar results were obtained for PBS and PBP. In order to compare them, their cross-correlation is plotted as shown in Fig. 11.

As it can be seen, the two sequences are entirely different which proves the sensitivity of the generator to the secret key and initial conditions. Similar cross correlation figures were obtained for the other two sequences PBS and PBP.

Figure 12 shows the cross correlation plot between PBP and PBS sequences.

As it can be seen, there is no similarity between the generated chaotic sequences from different neurons in the output layer of the generator.

6.2 Testing the proposed encryption algorithm

We applied our encryption algorithm on several test images of various sizes such as Lena and Mandrill of sizes 256*256 and 512*512 for both gray-scale and RGB colour images.

Table 5 Comparison of 0/1 Balance

Proposed generator			[13]	[19]
PBP	PBS	PPP		
50.004	49.992	49.998	49.92	50.01

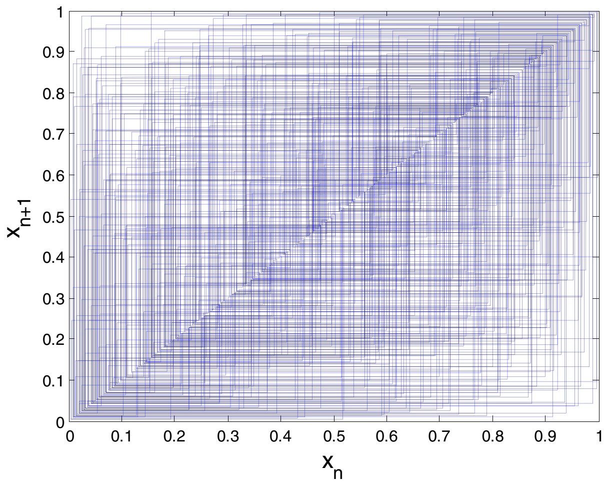

Fig. 9 2D phase space plot of the proposed generator

In this section, several tests were performed to measure the randomness of the encrypted images and resilience of the algorithm against attacks and the results are compared to a number of existing encryption algorithms. The encryption time for a 256*256 Lena image is approximately 74 secs.

6.2.1 Histogram analysis

The histogram of an image shows the distribution of the pixel intensity values in the range of (0, 256). If the intensity of the pixels are uniformly distributed, then the image is highly random and highly resistant against statistical attacks.

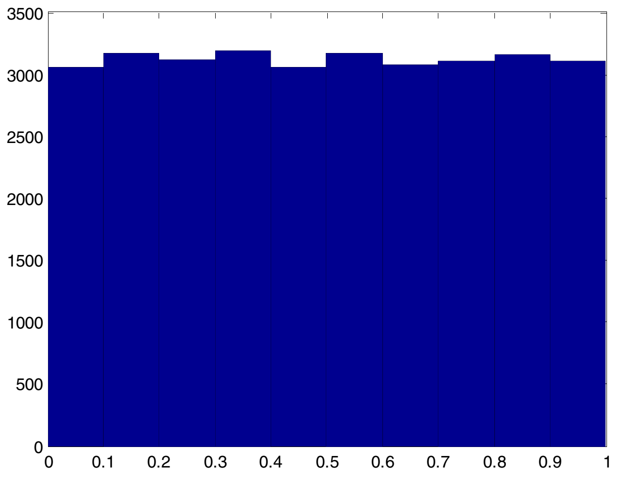

Fig. 10 Histogram plot of the proposed generator

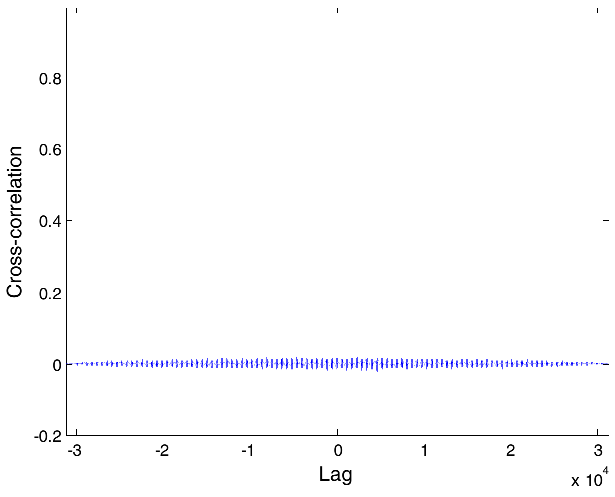

Fig. 11 Cross correlation of 2 PPP sequences with 1-bit change in the initial key

Figure 13 shows the plain gray-scale Lena image of size 256*256, its corresponding encrypted image and histograms of plain and encrypted Lena images. It can be seen from Fig. 13b and d that the histograms of the plain and encrypted images are very different and that the pixel intensity values are uniformly distributed. The average pixel intensity obtained for 100 encrypted images is 127.4612 which is very close to the ideal value of 127.5 and is better than the result of 127.5689 found in [19]. Note that the best average pixel intensity value obtained for one of the 100 tested encrypted images was 127.5067.

6.2.2 Correlation of two adjacent pixels

Adjacent pixels on an average are correlative to an extent of 8 to 16 pixels. Randomness is better if correlation between adjacent pixels is as low as possible. We calculate

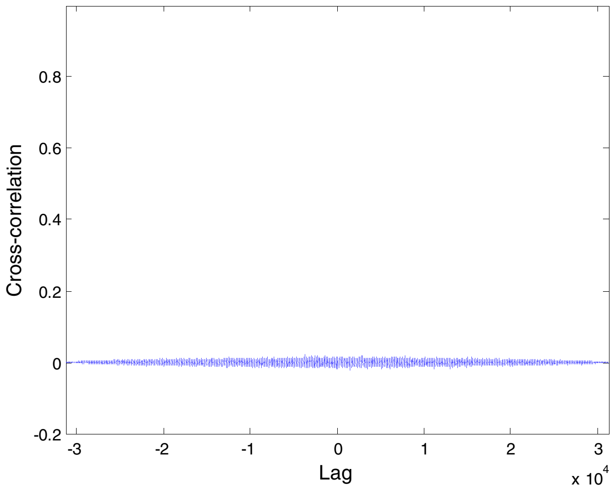

Fig. 12 Cross correlation between multiple generated sequences at the same time

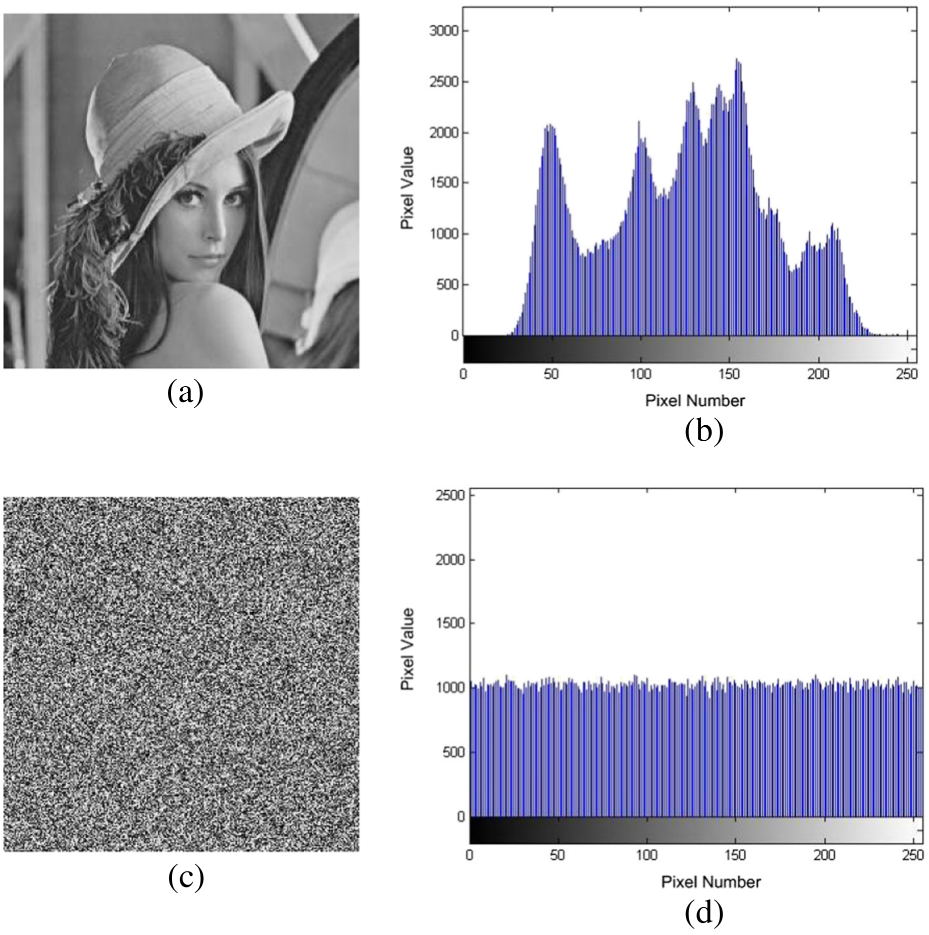

Fig. 13 **a** Plain Lena Image, **b** Histogram of Plain Lena Image **c** Encrypted Lena Image, and **d** Histogram of Encrypted Lena Image

the correlation coefficients for horizontally, vertically and diagonally adjacent pixels using (7–10):

$$E(x) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} P(i, j) \tag{7}$$

$$D(P) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [P(i, j) - E(P(i, j))]^2 \tag{8}$$

$$Cov(P, C) = \tfrac{1}{M \times N} \textstyle\sum_{i=1}^{M} \sum_{j=1}^{N} [P(i, j) - E(P(i, j))] \\ [C(i, j) - E(P(i, j))] \tag{9}$$

$$r_{PC} = \frac{Cov(P, C)}{\sqrt{D(P)}\sqrt{D(C)}} \tag{10}$$

where *P(i, j)* and *C(i, j)* are the intensity values of the original and encrypted pixels.

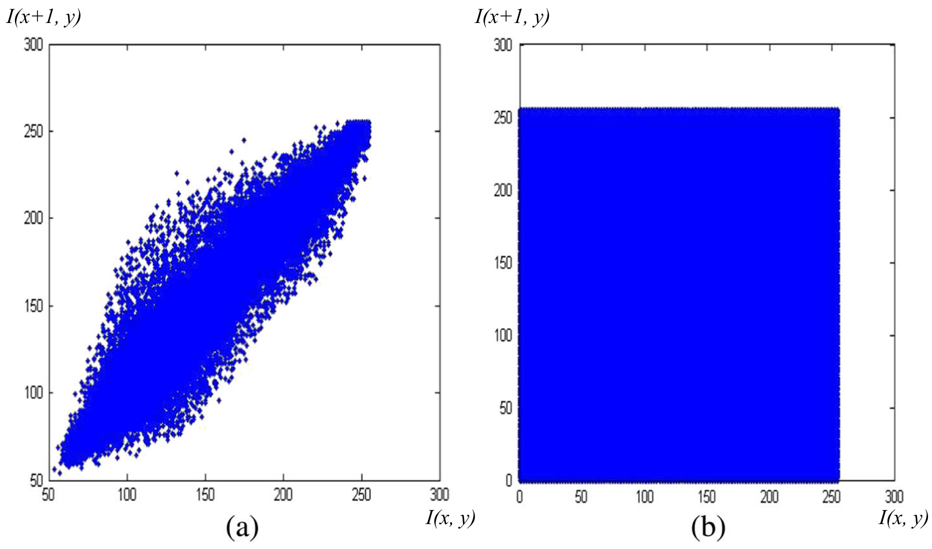

Fig. 14 Horizontal Pixel Correlation of Lena Image **a** Plain and **b** encrypted

Figure 14a and b show the correlation distribution of horizontally adjacent pixels for plain and encrypted images of Lena.

As we can see, a high correlation appears in Fig. 14a. However, this correlation is suppressed in the encrypted image where the values of adjacent pixels are quite different.

The correlation coefficients calculated for the original and encrypted images are shown in Table 6 for a number of grayscale images and in Table 7 for coloured images. In addition, a comparison is given with existing encryption algorithms.

Table 6 Adjacent Pixel Correlation for Grayscale plain and encrypted images

Algorithm	Horizontal	Vertical	Diagonal
*Lena 256*256*			
Original	0.94296	0.97121	0.920326
[19]	0.002661	−0.00194	0.000713
[2]	−0.0026	−0.0020	−0.0029
[11]	−0.0043	0.0014	–
[16]	0.0004	0.0021	−0.0038
[23]	0.0046	0.0040	0.0017
Proposed	0.00046	0.0011	0.0031
*Lena 512*512*			
Original	0.9719	0.9850	0.9593
[9]	0.0058	0.0072	0.0031
Proposed	0.000235	−0.0032	−0.002
*Mandrill 256*256*			
Original	0.8665	0.7586	0.7261
[9]	0.0109	0.0139	0.0081
Proposed	0.0015	−0.00082	0.0027

Table 7 Adjacent Pixel Correlation for Color Plain and Encrypted Images

Algorithm	Horizontal	Vertical	Diagonal
*Mandrill 256*256*			
Original	0.9001	0.8925	0.8569
[13]	0.0016	0.0024	0.0032
Proposed	−0.0026	0.0023	−0.0022

Tables 6 and 7 show that the values of the correlation coefficients are very close to '0', which proves that encrypted image is very random. As it can be seen, the obtained correlation values are better than [2, 19] and [23] at least in two of the three dimensions. Also compared to other works described earlier such as [13, 16] and [9], we get better results in all the dimensions.

6.2.3 *Information entropy*

Information entropy is a statistical tool to measure the randomness present in an image and characterizes the texture of the image. Entropy *H(m)* of a message source m is given by the (11),

$$H\left(m\right)=\sum_{i=0}^{2^{N}-1}p\left(m_i\right)\log_2\frac{1}{p(m_i)} \tag{11}$$

where $p(m_i)$ is the probability of occurrence of a message symbol m_i. A message source m is considered to be random if there are 2^8 or 256 possible outcomes of the message each of which has equal probability of occurrence. Hence, a value of 8 for the entropy means that the message source is highly random. Table 8 presents the average entropy value obtained for 100 encrypted images with 100 different keys for several test images. The obtained results are compared with existing algorithms.

As we can see, the entropy values are very close to 8 which means that the information leakage is negligible and the encryption system is secure against entropy attack. When comparing the results obtained for our algorithm to similar related works for different images, we can see that our proposed algorithm gives higher entropy than the results shown in [2, 11, 13, 16] and [9] (for Mandrill 512*512 grayscale), similar to [19] and [9] (for Lena 512*512

Table 8 Entropy Values for the Proposed Algorithm With Related Works

Test image		Related work	Proposed
Name	Size		
Gray Lena	256*256	7.9975 [2]	7.9976
		7.9978 [23]	
		7.9874 [16]	
		7.989 [11]	
		7.9976 [19]	
	512*512	7.9992 [9]	7.9992
Gray Mandrill	256*256	7.9895 [11]	7.9969
	512*512	7.9984 [9]	7.9993
Color Mandrill	256*256	7.89 [13]	7.9989

gray) and it is slightly lower than [23] with a difference of 0.0002. However, the correlation of adjacent pixels are better for our proposed algorithm comparing to [23].

6.2.4 NPCR and UACI

Common measures like NPCR (Number of Pixels Change Rate) and UACI (Unified Average Changing Intensity) are used to test the difference between the original image P and the encrypted one C as given by (12–14):

$$D(i,j) = f(x) = \begin{cases} 1 \text{ if } P(i,j) \neq C(i,j) \\ 0 \text{ else} \end{cases} \tag{12}$$

$$NPCR = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} D(i,j)}{M \times N} \times 100 \tag{13}$$

$$UACI = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{|P(i,j) - C(i,j)|}{255} \times 100 \tag{14}$$

where *P(i, j)* and *C(i, j)* are the intensity values of the pixels at the position *(i, j)* in the plain and encrypted images of size *M*N* respectively.

Table 9 below shows the NPCR and UACI results for a number of test images encrypted by our proposed algorithm and some of the related works.

We compared the NPCR and UACI between the plain image and encrypted grayscale Lena image (of size 256*256) produced by our proposed encryption algorithm and compared with some of the related work. We found that our calculated NPCR is higher than the values found in [2] when the UACI is slightly lower with a difference of 0.0154. The NPCR is slightly lower than [19] with a difference of 0.0045. However, as we mentioned earlier, the adjacent pixels correlation is better with our algorithm and the entropy is similar.

We also calculated the NPCR and UACI for the color image of Mandrill (of size 256*256) as shown in Table 10. Our obtained NPCR was slightly lower than [13] by 0.0576. However, the percentage of adjacent pixels as well as the entropy was better with our algorithm. Furthermore, the obtained NPCR and UACI are high enough to prove that the plain and encrypted image are very different.

6.2.5 Key-sensitivity test

For any encryption algorithm, it is important that a small change in the key produces an entirely different cipher text. In order to test this, we used two keys with just a single bit difference and used them to encrypt the same Lena gray-scale 512*512 image. Figure 15 shows the difference between the two cipher images. It can be seen that the difference is

Table 9 NPCR and UACI Results for the Proposed Encryption Algorithm and Related Works

Test image	NPCR	UACI		
	Related work	Proposed algorithm	Related work	Proposed algorithm
Grayscale Lena 256*256	99.62 [19]	99.6155	–	28.567
	99.5834 [2]		28.5824 [2]	
Colored Mandrill 256*256	99.67 [13]	99.6124	–	29.4367

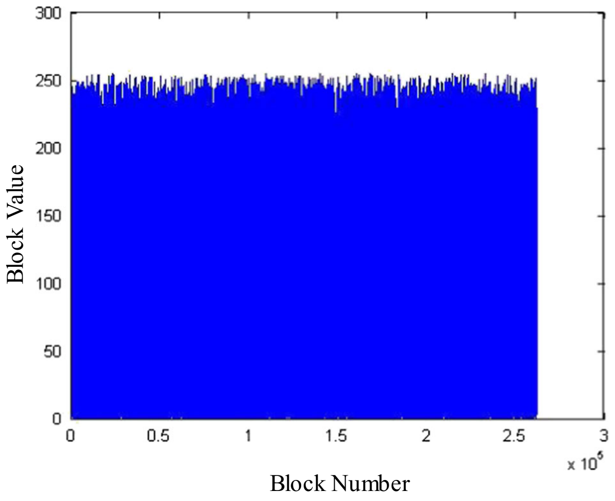

Fig. 15 Difference between 2 encrypted Lena images with 1-bit change in the initial key

not equal to zero for nearly all the pixels which means that the two encrypted images are entirely different even with just a single bit difference in the keys used to encrypt them.

We calculated the percentage of changed pixels in the two cipher images and we got a value of 99.60%, which indicates that nearly all the pixels are dissimilar, and hence proves that the two encrypted images are completely different [21].

6.2.6 *NIST statistical tests*

NIST tests were performed on 100 encrypted Lena 512*512 gray-scale images, each encrypted with hundred different keys. Table 10 below shows the obtained results of NIST tests. As it can be observed, all the tests were passed with a minimum passing rate of 98%.

6.2.7 *Plaintext sensitivity*

In this section, we will discuss the sensitivity of our algorithm to the plain image. As it can be observed, the algorithm is shown and tested in the Electronic Codebook (ECB) mode i.e. the encryption and decryption of each block of the image is independent from the other blocks or pixels. Therefore, under a given key, any given plain image block always gets encrypted to the same encrypted image block and then the encryption in this mode is not very sensitive to the original image. If this property is undesirable, then as mentioned in [8], the ECB mode should not be used". A cryptographic mode of operation usually combines the basic encryption method with some sort of feedback and some simple operations. The most popular confidentiality modes of operation for symmetric key block encryption are the Electronic Codebook (ECB) mode, Cipher Block Chaining (CBC), Cipher Feedback (CFB) mode, Output Feedback (OFB) mode and Counter (CTR) modes. When a block is changed in the plain image (or in the encrypted image) results in a different number of changed number of encrypted blocks (or decrypted blocks) depending on the chosen mode of operation. For example, OFB mode avoids the propagation of errors in the decrypted image, when an error occurs in the encrypted image (a bit is changed). However, its robustness to the chosen

Table 10 NIST Results for 100 Encrypted Lena 512x512 Images

NIST test	P-value	Result of tests
Frequency	0.514124	Pass
Block Frequency	0.554420	Pass
Cumulative Sums	0.719747	Pass
Runs	0.851383	Pass
Longest Runs	0.946308	Pass
Rank	0.334538	Pass
FFT	0.657933	Pass
Non-overlapping Templates	0.867692	Pass
Overlapping Templates	0.816537	Pass
Universal	0.474986	Pass
Approximate Entropy	0.739918	Pass
Random Excursions	0.846579	Pass
Random Excursions Variant	0.969690	Pass
Serial	0.816537	Pass
Linear Complexity	0.964295	Pass

plaintext attack and differential attack is not guaranteed. CBC mode doubles the errors in the decrypted image but the algorithm in this mode is more resistant to this type of attack. As it can be concluded, this is relative to the chosen mode of operation and not to the proposed encryption algorithm. Our algorithm is compliant with NIST recommendations exposed in [8] and same results are obtained for the standard AES encryption algorithm.

7 Conclusions

In this paper, a heterogeneous chaotic generator, implemented using neural network, is proposed. The heterogeneity of the generator is obtained by alternating two different chaotic maps; Logistic and PWLCM, in the neural network layers. The proposed generator is highly random and possesses good chaotic properties as shown in the test results including auto-correlation, cross-correlation, 0/1 balance and NIST tests. The 0/1 balance was compared to similar works in pseudo-random generators and it was seen that our proposed generator difference between 50-50 distribution of 0s and 1s was on average 0.004 while similar works achieved at best 0.01. Hence, our proposed generator is nearly 10 orders better on 0/1 balance. The NIST was conducted on 100 sequences, and on average of 100 sequences 99.2 passed the test with lowest being 96. In addition, using our proposed generator, several chaotic sequences can be generated simultaneously by varying the number of neurons in the output layer which allows to perform a number of cryptographic operations with a minimum number of the neural network cycles. In the paper, we also proposed a new encryption algorithm using chaotic generator as input for its cryptographic operations including: pixel position permutation, DNA based substitution and DNA based bit permutation which was also introduced in this paper to enhance the statistical properties of the encrypted images. Several tests were performed on the encrypted images such as correlation, entropy, histogram, NIST statistical test etc. Histogram analysis shows that for 100 encrypted images, we get average value for pixel intensity of 127.46 which is close to ideal value of 127.5.

The entropy value for encrypted images is very close to 8, with average difference between entropy of proposed method to 8 being 0.003. The entropy value was compared to other works, and in most cases it is similar or better than similar works. NIST tests one 100 images showed that all the images passed all the tests. From the test and comparisons to similar works, it is proved that the proposed encryption algorithm has high cryptographic quality and it is compliant with NIST standards.

References

1. Avasare MG, Kelkar VV (2015) Image encryption using chaos theory. In: 2015 international conference on communication, information & computing technology (ICCICT). IEEE, pp 1–6
2. Awad A, Miri A (2012) A new image encryption algorithm based on a chaotic dna substitution method. In: 2012 IEEE International Conference on Communications (ICC). IEEE, pp 1011–1015
3. Bassham III LE, Rukhin AL, Soto J, Nechvatal JR, Smid ME, Barker EB, Leigh SD, Levenson M, Vangel M, Banks DL (et al) Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications
4. Burak D (2013) Parallelization of encryption algorithm based on chaos system and neural networks. In: Parallel Processing and Applied Mathematics, Springer, pp 364–373
5. Chauhan M, Prajapati R Image encryption using chaotic based artificial neural network
6. Chen J-X, Zhu Z-L, Fu C, Yu H, Zhang L-B (2015) A fast chaos-based image encryption scheme with a dynamic state variables selection mechanism. Commun Nonlinear Sci Numer Simul 20(3):846–860
7. Daemen J, Rijmen V (2013) The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media, Berlin
8. Dworkin M (2001) Recommendation for block cipher modes of operation. methods and techniques, Tech. rep., DTIC Document
9. Enayatifar R, Abdullah AH, Isnin IF (2014) Chaos-based image encryption using a hybrid genetic algorithm and a dna sequence. Opt Lasers Eng 56:83–93
10. Enayatifar R, Sadaei HJ, Abdullah AH, Lee M, Isnin IF (2015) A novel chaotic based image encryption using a hybrid model of deoxyribonucleic acid and cellular automata. Opt Lasers Eng 71:33–41
11. Hossain MB, Rahman MT, Rahman A, Islam S (2014) A new approach of image encryption using 3d chaotic map to enhance security of multimedia component. In: 2014 International Conference on Informatics, Electronics & Vision (ICIEV). IEEE, pp 1–6
12. Hu Y, Zhu C, Wang Z (2014) An improved piecewise linear chaotic map based image encryption algorithm. The Scientific World Journal, Cairo
13. Kassem A, Hassan HAH, Harkouss Y, Assaf R (2014) Efficient neural chaotic generator for image encryption. Digital Signal Process 25:266–274
14. Li X, Li C, Lee I-K (2016) Chaotic image encryption using pseudo-random masks and pixel mapping. Signal Process 125:48–63
15. Lian S (2009) A block cipher based on chaotic neural networks. Neurocomputing 72(4):1296–1301
16. Liu H, Wang X et al (2012) Image encryption using dna complementary rule and chaotic maps. Applied Soft Comput 12(5):1457–1466
17. Liu Y, Tang J, Xie T (2014) Cryptanalyzing a rgb image encryption algorithm based on dna encoding and chaos map. Opt Laser Technol 60:111–115
18. Qin K, Oommen BJ (2014) Cryptanalysis of a cryptographic algorithm that utilizes chaotic neural networks. In: Information Sciences and Systems 2014, Springer, pp 167–174
19. Singla P, Sachdeva P, Ahmad M (2014) A chaotic neural network based cryptographic pseudo-random sequence design. In: 2014 4th International Conference on Advanced Computing & Communication Technologies (ACCT). IEEE, pp 301–306
20. Tong X-J, Zhang M, Wang Z, Liu Y, Xu H, Ma J (2015) A fast encryption algorithm of color image based on four-dimensional chaotic system. J Vis Commun Image Represent 33:219–234

21. Wu Y, Noonan JP, Agaian S (2011) Npcr and uaci randomness tests for image encryption, Cyber journals: multidisciplinary journals in science and technology. Journal of Selected Areas in Telecommunications (JSAT)
22. Zhang J (2015) An image encryption scheme based on cat map and hyperchaotic lorenz system. In: 2015 IEEE international conference on computational intelligence & communication technology (CICT). IEEE, pp 78–82
23. Zhang Q, Liu L, Wei X (2014) Improved algorithm for image encryption based on dna encoding and multi-chaotic maps. AEU Int J Electron Commun 68(3):186–192
24. Zhang Y, Xiao D, Wen W, Wong K-W (2014) On the security of symmetric ciphers based on dna coding. Inf Scie 289:254–261
25. Zhou Y, Bao L, Chen CP (2014) A new 1d chaotic system for image encryption. Signal Process 97:172–182