IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 11, 2019, accepted May 22, 2019, date of publication June 3, 2019, date of current version July 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920383

A New Hybrid Image Encryption Algorithm Based on 2D-CA, FSM-DNA Rule Generator, and FSBI

SAJID KHAN[1], LANSHENG HAN[2], HONGWEI LU[2], KHUSHBU KHALID BUTT[1], GUEHGUIH BACHIRA[1], AND NAIMAT-ULLAH KHAN[3]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Faculty of Computer Science Department, Huazhong University of Science and Technology, Wuhan 430074, China
[3]School of Communication and Information Engineering, Institute of Smart City, Shanghai University, Shanghai 200444, China

Corresponding author: Lansheng Han (hanlansheng@hust.edu.cn)

This work was supported by the Natural Science Foundation of China under Grant 61272033 and Grant 61572222.

ABSTRACT Image encryption is an efficient and vital way to protect classified and secret images. With the advancement of the processing power of the computer, AES, DES, or chaotic series type just alike image encryption schemes are not as secure as before. Therefore, in this paper, we present a new hybrid image encryption method for protecting secret and imperative images by employing logistic sine system (LSS) together with two-dimensional cellular automata and FSM-based DNA rule generator. The secure hash (SHA-256) algorithm is used to generate a secret key and to compute initial values for the LSS. In our proposed method, there are three stages and each stage has its own rule. After the scrambling process, the first stage is the Feistel structure-based bit inversion (FSBI) to change the pixels' value. The second stage is 2D-CA with Moore neighborhood structure-based local rules. The third is DNA conversion based on finite-state machine (FSM-DNA) rule generator. The proposed encryption scheme is robust against the well-known attacks, such as statistical attacks, brute force attacks, differential attacks, and pixel correlation attacks, and also possesses strong key sensitivity. The results show that our three-layer hybrid image encryption technique is robust against many well-known attacks and can be applied directly to all types of classified gray-scale images to make them more secure from such cryptography attacks.

INDEX TERMS Image encryption, DNA sequence, finite state machine, Feistel structure bit, logistic sine system, SHA-256, two-dimensional cellular automata.

I. INTRODUCTION

With the fast advancement of computerized innovations and Internet, data security including image encryption has turned out to be increasingly critical [1]. Advanced encryption techniques inquires about has been the new hot-spot of current cryptography [2] after DES and AES turned into the standard of content information encryption. Cryptography professionals endeavor to locate an ideal image encryption technique to serve it as a standard image encryption algorithm.

Contrasted with the content information, advanced image's qualities as an extensive degree of information, substantial relationship, enormous information excess, and others influence it to require a substantial amount of pseudo-arbitrary numbers as a key stream, which invigorates the exploration on pseudo-random numbers [3], [4].

Below, we will introduce some close related work to our paper, in order to point out the limitation of existing research work and emphasized the need for proposed research work. In [4], Logistic map based technique with permutation and image blocking proposed. In this particular technique, logistic map parameters and initial values functioned as the secret key. Moreover, this technique executed by the FPGA chip. This sort of permutation only images encryption algorithms have been demonstrated hazardous and was broken by chosen as well as known plain-text attack techniques [5].

Rather than utilizing the disorderly frameworks or chaotic system to produce the key-stream in [6], [7], the strategy utilized the 128-bit secret key to perform keyword dissemination and substitution forms. The particular encryption algorithms had certain normality and kept lake of robustness against the chosen plain-text attacks. The exceptional discussion on image encryption and decryption techniques advanced

The associate editor coordinating the review of this manuscript and approving it for publication was Habib Ullah.

the reimbursed improvement of image-based cryptography innovation [8], [9].

Current schemes of image encryption classified into two major categories, fixed [10], [11] and malleable bit length [12]. The prior is precisely fixed bit length and is not applicable to images having a different structure, while the latter is pixel level encryption scheme. Nonetheless, this kind of prevailing encryption strategies more often than not has an entangled structure, which may enormously take additional calculation time [13], [14].

Zhang et al. structured a novel encryption scheme for an image in [15], block diffusion and bit permutation have been implemented through matrix rotation. Similarly, Zhou et al. introduced another security technique in [16] that can altogether do encryption and compressing of images utilizing a 2D compressive sensing and hyper-chaotic framework.

Albeit, a wide range of encryption schemes created to ensure the security of classified images, but to the best of our insight, some of them have inadequacies in various perspectives. To begin with the enhancement of computation speed and power along with improvement of cryptanalysis techniques, some encryption schemes have already been the menace of being broken [17], [18]. For instance, as the basic chaotic theory has the properties of capriciousness and ergodicity [19], and broadly used to construct encryption algorithms to protect such images. Nevertheless, because of the execution confinement of such chaos frameworks, some early chaotic based cryptosystems exposed to retaining low-security echelons [20].

On the other side, because of the massive parallelism and unprecedented data, DNA based computing got popularity in image encryption. Generally, DNA computing involves few mathematical operations on DNA sequence, such as XOR operation, multiplication, addition as well as subtraction operations. So DNA based schemes utilize such operation for encryption and decryption of certain images [21].

Encryption algorithms based on DNA structure typically performs ciphering and deciphering after converting it into the binary form [22]. The particular encryption scheme is based on Cellular network and DNA operation. A Chaotic framework and DNA based two novel encryption schemes proposed in [23], [24] respectively. In [25] the author has introduced confusion and diffusion-based two rounds scheme along with chaos and DNA sequence for images encryption.

In [26] the proposed encryption scheme encrypts the seed image and plain image by DNA rule operation and later further DNA operations to transform the image in cipher form. In all the above-proposed schemes on the most occasion, permutation part is included to enhance the security level. In general, there is eight DNA rule operation like addition, subtraction, XOR operations. These rules remain fixed in numerous encryption algorithms during the encoding and decoding process of the plain image or else some transformed encryption rules implemented with some additional rule for decoding. In some case to the purpose of choosing these rule is to enlarge the key-space [27].

Conclusively, identical encoding rules for all components of the plain image and seed image result in no relation of these rules with a plain image. Thus, diminishing the knack of encryption strategies to resist common cryptanalysis attacks such as known plain text, statistical or chosen plain-text attacks

Similarly, the distinctive characteristic of Cellular Automata (CA) is its unpretentious fundamental instructions that can work very efficiently. The composite behaviors of CA make it perfect for developing CA-based cryptography. CA is discrete inputs and outputs based numerical model of system framework. It characterizes the chronological behavior of multiple consistent cells that are arranged in steady manners with having a discrete set of possible values [28].

Nowadays, CA is widely used in image encryption schemes. For instance, First, pseudo-random numbers are generated through 2D-CA and then through the mathematical operation like XOR or substitution process with these random numbers apiece pixel of an image is encrypted [29]. The security of these algorithms relies on the randomness of these numbers. Thus, the security of these encryption schemes depends on the haphazardness of these numbers.

A two-dimensional CA-based encryption scheme proposed in [30], [31], while the previous and current state of its neighbor determined the next state of a particular cell. Recursively, to get the initial state, the algorithms store the recent two states of each cell. Consequently, the result is in twice the size of the cipher image as compared to the plain image.

Based on the analysis given above, some common shortages of those work are twice the size the of cipher image, more number of rounds to get the satisfactory security level results, fixed DNA rule for whole encryption or decryption part and small key size. So to overcome these shortcomings a new hybrid encryption technique is proposed in this paper. 2D-CA with Moore neighborhood structure strategy has adopted with a local rule to update the matrix configuration. The first local rule for block matrix determined through initial configuration matrix with the help of formula. As our proposed algorithm has a complex rule structure so in result and simulation section, we will prove that it is robust against many cryptanalysis attacks e.g. known and chosen plain text attack. Besides Feistel structure based bit diffusion and fast pixel scrambling algorithm enhanced the efficiency and speed along with the promising security. Our paper has made three contributions with each having corresponding merits.

First, DNA encoding rules are dynamical. In the above paper's analysis we found DNA rules are almost kept fixed or in some cases varying from **1** to **8** in sequence but in our algorithms, a new model of FSM-DNA rule generator provide random DNA rule to each sub-matrix according to the first and last bit arrangements. So the proposed algorithm has high sensitivity relating to the image pixels values because two specific bits of each particular sub-block employed to decide the DNA rule for the next sub-block.

Second, in new Feistel structure based bit inversion model; bit selection is totally random and depends upon sub-matrix

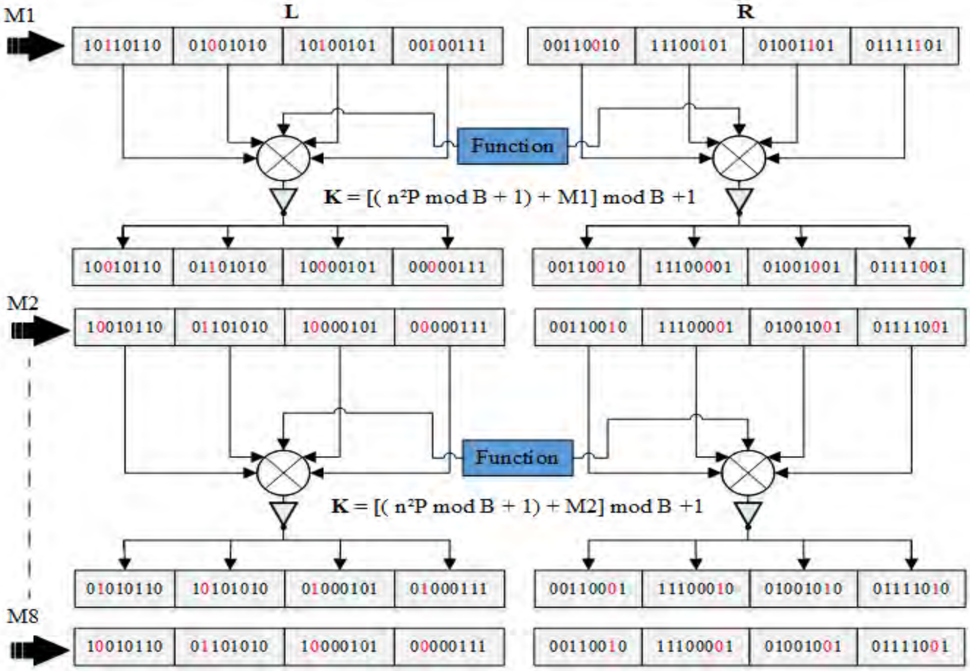


FIGURE 1. Feistel structure based bit inversion.

multiplier along with corresponding integer value of the double hash key string. That means different images will have a different structure of bit selection for starting and ending bit of each sub-block.

Third, the Moore neighborhood structure based Local rules are dynamical. Each matrix has a different local rule structure. Also, every sub-matrix has a direct relation with the previous matrix. Except that efficient and fast pixel scrambling algorithm reduce the strong correlation between neighboring pixels. The remainder of the paper arranged as follow; in Section in Section 2. preliminary work, Section 3. image encryption scheme. Section 4. results and discussion while Section 5. is security analysis parts with the subsequent Section 6. conclusion and future work.

II. PRELIMINARY WORK

A. LOGISTIC SINE SYSTEM (LSS)

LSS proposed in [32] is used to generate a pseudo-random sequence. Double-hash value is used to generate initial values that later used for indexing purpose. LSS-PRNG can be defined by the following equation

$$x_{t+1} = [rx_t(1-x_t) + (4-r)sin(\pi x_t/4)]mod1 \qquad (1)$$

where $\boldsymbol{r}$ is the control parameter and $\boldsymbol{x}_t$ is the iteration variable. In the above equation $r \in \{0, 4\}$ and $x_t \in \{0, 1\}$ whereas key (hash value) is used to generate the determined random sequence. The Array sequence is sorted and indexed as I, J. These two sorted arrays are used for creating a random matrix containing the indexed values with the shifted position of one array value according to the others array value. The detail description of this process is given in scrambling section.

B. FEISTEL STRUCTURE BASED BIT INVERSION

Fig. 1 depicts our proposed Feistel based structure for a binary form of image block of size $p \times q$. Firstly, it converts the particular matrix rows into two equal blocks of columns termed as $Left_{blocks}$ and $Right_{blocks}$. The purpose is to change the particular bit value of each pixel in every row of both columns blocks but in the opposite way. It inverts the bit 0 or 1 into opposite based on its current value. Like changes into **1** if it is **0** or else **0** if it's **1**.

For example, in Fig. 2 we can see that for $Left_{blocks}$ (L) the bit selection is from left, and for the $Right_{blocks}$ (R) the bit selection is from the right side. More clearly shown in Fig.1 that in $M1$, 3rd bit is selected from the left blocks (L) which is actually 6th if we count from the right similarly on right blocks (R), the 3rd bit is selected which is actually the 6th if we count from the left side so both are opposite just

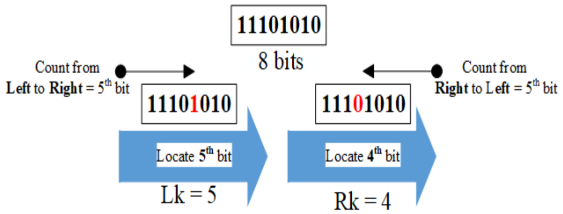


FIGURE 2. Feistel bit selection.

$Left_{bit} \rightarrow Right_{bit}$ or $Left_{bit} \leftarrow Right_{bit}$ confusion as shown in Fig.2. In the whole process, bit selection is based on the following formula.

$$Left_{blocks} : K = [(n^2 \times PmodB + 1) + M_i]modB + 1 \tag{2}$$

Here ***n*** depicts the multiplier value corresponds to the particular sub-block matrix, ***P*** is the integer value of the particular pair of the hash key string. ***K*** represents the bit location for operation and is random based on the hash value, while ***B*** represents the total number of bits in one pixel that is one byte or 8-bits in grey scale image. ***MOD*** is the modulus operator that gives the remainder between **0** to **7** after division. M_i is the row number where $i = 1, 2, 3, 4, 5 \ldots N$. While in our case the value of ***N*** is restricted to **8**.

Let suppose for the 1st sub-block matrix the value of ***P*** came **55** so the value of ***K*** for the first row $i = 1$ or $M_i = 1$ can be calculated as follow.

$$\begin{aligned} Left_{blocks} : K &= [(1 \times 53\ mod 8 + 1) + 1]mod 8 + 1 \\ &\Rightarrow [5 + 1 + 1]mod 8 + 1 = 7 + 1 = 8^{th} bit \end{aligned}$$

Similarly, for the 2nd row this 1st sub-block matrix, $i = 2$ or $M_i = 2$;

$$\begin{aligned} K &= [(1 \times 53\ mod 8 + 1) + 2]mod 8 + 1 \\ &\Rightarrow [5 + 1 + 2]mod 8 + 1 = 0 + 1 = 1^{st} bit \end{aligned}$$

And so on for the 3rd, 4th, 5th to up to 8th rows, we can get our desired bit through this formula. In contrast to the Left block column, the formula of bit selection for the right block column is as follow.

$$Right_{blocks} : [B - K]modB + 1 \tag{3}$$

As the bit for the first row is 8th so, $\boldsymbol{R}_{blocks}$ = [8-8]mod 8+1= 1st bit which means in case of the right blocks 1st bit while counting from $Left_{bit} \leftarrow Right_{bit}$. While the same bit is at the 8th position if count from $Left_{bit} \rightarrow Right_{bit}$. Similarly, for the 2nd, 3rd, 4th to up to N^{th} sub-blocks matrices, by multiplying the ***P*** value with 2^2, 3^2, 4^2 to N^2 respectively we can get the 1st bit location for the 1st row of particular sub-matrix and so on proceeding till all the sub-blocks matrix of an image get updated. While this bit selection remains random for each sub-block. Also, as different images have different pixels values so this bit selection is also different for the different images. Therefore, we can say it is hard to predict a particular bit of any sub-block without getting some prior information.

C. 2-D CELLULAR AUTOMATA

The CA is an abstract dynamical system in which state, space and time are all discrete. The regular lattice structure of cells has a finite number of states. Let **D** be a positive integer, an m-dimensional Cellular Space is Z^m. The elements of Z^m are called cells, similarly, let finite state set is $\dot{S}$ then elements of $\dot{S}$ are called states. Therefore, the configuration of m-dimensional CA with $\dot{S}$ is a function $c : Z^m \Rightarrow \dot{S}$ that assigns a state to each cell.

The state of the cell $n \in Z^m$ is $c(\bar{n})$, these states are updated synchronously rendering to a specified local rule of neighborhood interaction. In two-dimensional square space consisting of $P \times Q$ cells, 2D generalized CA can be calculated from $c = \alpha(i, j, t)$, $t \leq N-1$ where $i, j \geq 0$. In Von-Neumann 2D cellular structure with each cell having two states, there are $2^{2^5} = 2^{32}$ possible evolutions while for Moore neighborhood if we exclude the central cell, there are $2^{2^8} = 2^{256}$ possible evolutions. CA has numerous features such as its simple regular structure, random behavior, local interaction, and enormous parallelism. Because of these features, it has been applied in various image encryption techniques.

In our paper, a 2D-CA having Moore neighborhood (Diagonal + Von-Neumann) structure has been used as shown in Fig. 3, each and every cell has two states either **0** or **1**. The general rule for a cell having a Von-Neumann neighbors that have a radius equivalent to **1** is as follows:

$$S^{t+1}_{N(i,j)} = \delta(S^t_{i-1,j}; S^t_{i,j+1}; S^t_{i,j}; S^t_{i,j-1}; S^t_{i+1,j}) \tag{4}$$

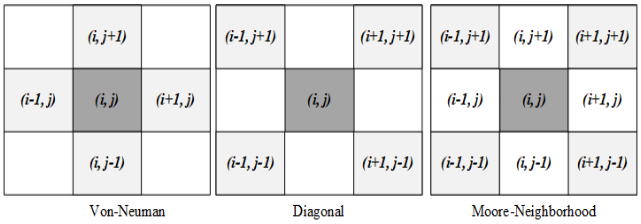


FIGURE 3. **2D-CA different neighboring cell structure.**

In this above equation δ is a state transition function or Boolean function that gives the new state, where $\delta : S \times \Sigma \rightarrow \rho(S)$. Like Von-Neumann, the rule for a cell having diagonal neighbors is as follow.

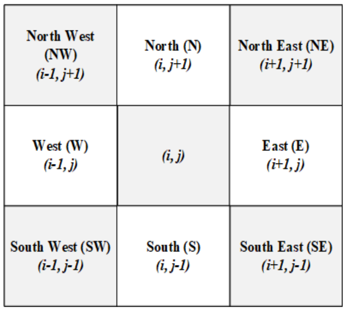
$$S^{t+1}_{D(i,j)} = \delta(S^t_{i-1,j-1}; S^t_{i-1,j+1}; S^t_{i,j}; S^t_{i+1,j-1}; S^t_{i+1,j+1}) \tag{5}$$

As we are dealing Moore neighborhood structure in our proposed algorithm, so by combining the Eq. (4) and Eq. (5) while excluding the central cell the final equation will become.

$$\begin{aligned} S^{t+1}_{DN(i,j)} = \delta(S^t_{i-1,j}; S^t_{i,j+1}; S^t_{i,j-1}; S^t_{i+1,j}; S^t_{i-1,j-1}; \\ S^t_{i-1,j+1}; S^t_{i+1,j-1}; S^t_{i+1,j+1}) \end{aligned} \tag{6}$$

In this Eq. (6), there are a total of eight state variables so for the ease of implementation, we are representing these **8** neighbors cells with the eight normal direction vectors, that are *North-West* ***(NW)***, *North-East* ***(NE)***, *South-West* ***(SW)***, *South-East* ***(SE)***, *North* ***(N)***, *South* ***(S)***, *West* ***(W)*** and *East* ***(E)*** shown in Fig. 4.

Also as in our case, the bit depth of the grey-scale image is **8** so we mapped above **8** state variables with these eight bits, while excluding the central state variable $S^t_{(i,j)}$. Also we will update the initial configuration matrix on the basis of this



North West (NW) *(i-1, j+1)*	North (N) *(i, j+1)*	North East (NE) *(i+1, j+1)*
West (W) *(i-1, j)*	*(i, j)*	East (E) *(i+1, j)*
South West (SW) *(i-1, j-1)*	South (S) *(i, j-1)*	South East (SE) *(i+1, j-1)*

FIGURE 4. Moore-neighborhood.

local rule structure and then later it will be XORed with the $S_{P(i,j)}^{t+1}$ to get the final updated matrix as follow.

$$S_{U(i,j)}^{t+1} = S_{DN(i,j)}^{t+1} \oplus S_{P(i,j)}^{t} \tag{7}$$

In this Eq. (7), the $S_{U(i,j)}^{t+1}$ is the updated configuration of particular sub-matrix after time step t+1, $S_{DN(i,j)}^{t+1}$ is the updated state of initial configuration following diagonal and Von-Neumann neighborhood structure and $S_{P(i,j)}^{t}$ is the current state of particular sub-matrix during time **t**.

As each variable has two states **0** or **1**, while the local rules surge at the exponential law followed by an increasing number of states or cell numbers. So $2^{2^8} = 2^{256} = 1.1579208923732E+77$ possible evolution will increase the algorithm complexity but also the security level to a large extent. So in order to improve the efficiency and reduce the computation complexity of our algorithm, we, therefore have mapped the above directional vectors with our **8** bits as follow; ***NW***, ***NE***, ***SW***, ***SE***, ***N***, ***S***, ***W***, ***E*** to $[11111111]_2$. The aim of this mapping was to create local rules for updating the sub-matrices, so because of this mapping Eq. (6) will also be updated as follow.

$$\begin{aligned} S_{DN(i,j)}^{t+1} = &(NW \times S_{i-1,j+1}^{t}) \oplus (NE \times S_{i+1,j+1}^{t}) \\ &\oplus (SW \times S_{i-1,j-1}^{t}) \oplus (SE \times S_{i+1,j-1}^{t}) \oplus (N \times S_{i,j+1}^{t}) \\ &\oplus (S \times S_{i,j-1}^{t}) \oplus (W \times S_{i-1,j}^{t}) \oplus (E \times S_{i+1,j}^{t}) \end{aligned} \tag{8}$$

In this updated form of Eq. (8); ***NW***, ***NE***, ***SW***, ***SE***, ***N***, ***S***, ***W*** and ***E*** are the variables with the values either **0** or **1**. So particular cell will take part in the state update process or XOR operation only if its respective bit will equivalent to **1**, else it will not take part in the state update process.

The combination of above direction variables is employed to decide which and whom cell or cells will contribute to the state update process. So different combination means different local rules equation for updating process of each sub-matrix. The general equation for our local rule structure is as follow.

$$CA_{LR} = \begin{cases} I_\alpha + I_\beta]mod255 + 1 & N = 1 \\ [(N-1)_\alpha + (N-1)_\beta]mod255 + 1 & N \neq 1 \end{cases} \tag{9}$$

Whereas $\alpha = Count\ 0^s \times 48$, and $\beta = Count\ 1^s \times 49$, ***I*** is initial configuration and ***N*** is the total number of sub-matrices of each window of an image. For example, let suppose we have initial configuration matrix $\boldsymbol{C^o}$ of size 4 × 8 as shown below.

As we know that the ASCII value of **0** is **48** and the ASCII value of **1** is **49**, so our local rule for the first sub-matrix will be as follow.

$$\begin{aligned} CA_{LR} &= [(16 \times 49) + (16 \times 48)]mod255 + 1 \\ \Rightarrow CA_{LR} &= 22 + 1 = 23 \end{aligned}$$

1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	1
0	1	1	0	0	0	1	0
0	1	0	1	1	0	1	0

Converting this decimal value $[23]_{10}$ into binary values $[00010111]_2$. As we can see only **4** bits are **1** So it means for the first sub-matrix only these **4** cells will participate in matrix update process. To find that particular **four** cells we put this value in the Eq. (8) as follow.

$$\begin{aligned} S_{DN(i,j)}^{t+1} = &(0 \times S_{i-1,j+1}^{t}) \oplus (0 \times S_{i+1,j+1}^{t}) \\ &\oplus (0 \times S_{i-1,j-1}^{t}) \oplus (1 \times S_{i+1,j-1}^{t}) \oplus (0 \times S_{i,j+1}^{t}) \\ &\oplus (1 \times S_{i,j-1}^{t}) \oplus (1 \times S_{i-1,j}^{t}) \oplus (1 \times S_{i+1,j}^{t}) \end{aligned} \tag{10}$$

So our Eq. (8) will finally be get simplified as follow.

$$S_{DN(i,j)}^{t+1} = S_{i+1,j-1}^{t} \oplus S_{i,j-1}^{t} \oplus S_{i-1,j}^{t} \oplus S_{i+1,j}^{t} \tag{11}$$

This updated Eq. (11) has four state variables that are **SE** = 1, **S** = 1, **W** = 1 and **E** = 1, that means only these particular four cells will take part in the matrix update process of particular sub-matrix. $S_{i+1,j-1}^{t}$ is actually ***SE***, $S_{i,j-1}^{t}$ is actually ***S***, $S_{i-1,j}^{t}$ is actually ***W***, while $S_{i+1,j}^{t}$ is actually ***E***. Therefore, by using these local rules, the encryption speed, and efficiency along with security can be improved enormously. Finally, this updated Moore neighborhood structure base initial configuration matrix will be XORed with the particular sub-matrix to get the updated form of that matrix as follow.

$$S_{U(i,j)}^{t+1} = S_{P(i,j)}^{t} \oplus S_{DN(i,j)}^{t+1} \tag{12}$$

Let for example our first sub-matrix $S_{1(i,j)}^{t}$ is as given below, while we suppose that the edge values are linked or connected.

$$S_{P(i,j))}^{t} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

So according to the local rule that generated by the formula of Eq. (9), its $S_{DN(i,j)}^{t+1}$ is also shown below:

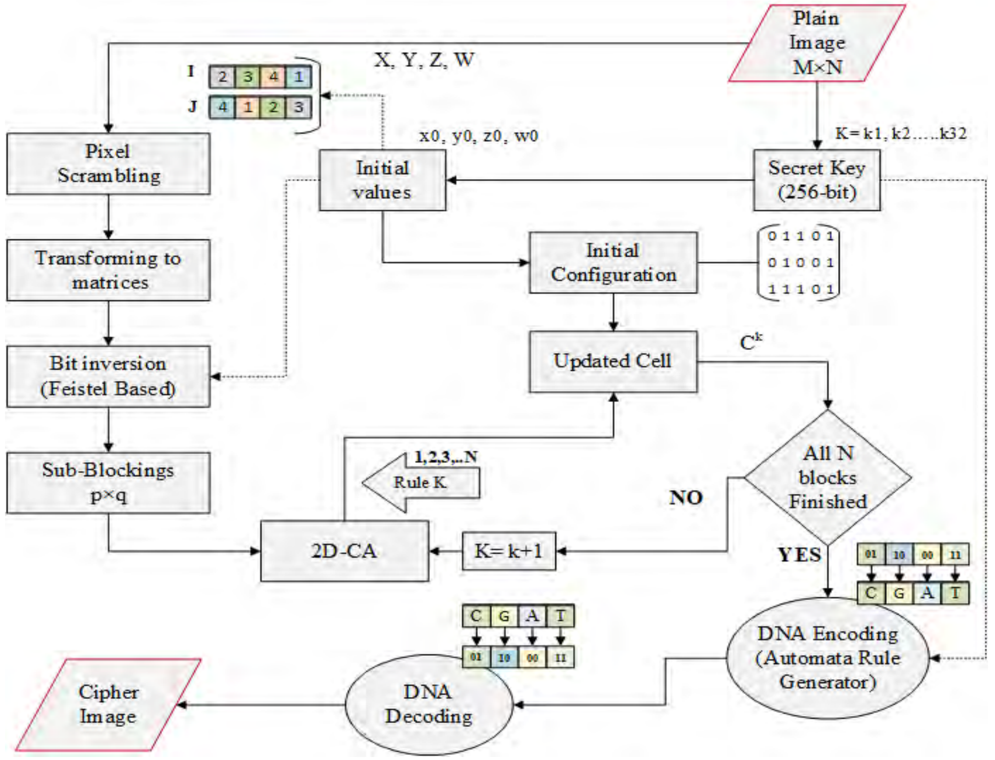


FIGURE 5. Block diagram of proposed encryption Scheme.

$$S^{t+1}_{DN(i,j))} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Hence, the first sub-matrix will be updated by XORing these two matrices as illustrated below.

$$S^{t+1}_{U(i,j)} = S^{t}_{P(i,j)} \oplus S^{t+1}_{DN(i,j)}$$

$$S^{t+1}_{P(i,j))} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (13)$$

Similarly, for the 2nd sub-matrix, the local rule will be based on the formula of the ASCII values of this 1st sub-matrix and so on for the 3rd sub-matrix, the local rule will be of the sum of the ASCII value of 2nd sub-matrix. Which means the local rule of each sub-matrix has a direct relation with the previous sub-matrix whereas, for the 1st sub-matrix we got the local rule from the initial configuration matrix C^o. While during decryption process in order to get back the original matrix the equation will be as follow.

$$S^{t}_{P(i,j)} = S^{t+1}_{U(i,j)} \oplus S^{t+1}_{DN(i,j)} \quad (14)$$

D. DNA SEQUENCE AND ALGEBRAIC OPERATION

Deoxyribonucleic acid or DNA is made of a molecule called nucleotide. The four nucleic bases comprised in DNA are; Adenine (**A**), Guanine (**G**), Cytosine (**C**), and Thymine (**T**). **A** and **T**, **C** and **G** are the complementary of each other because of their opposite binary value representation.

As in binary form **0** and **1** are complementary, so **00** and **11** are also complementary. Similarly, **01** and **10** are also complementary to each other, and this is known as Watson-Crick base pairing rule [33]. Table 1, illustrates the DNA encoding and decoding rules for a DNA sequence.

TABLE 1. DNA rules.

DNA Rule	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
C	01	10	00	11	00	11	01	10
G	10	01	11	00	11	00	10	01

For example, let suppose the pixel value of a grey scale image is **120**, the corresponding binary value is $(01111000)_2$, So DNA sequence according to the above **Rule1** or **R1** will be **CTGA**. Similarly, if the DNA sequence is **GTCA** its binary value obtained through **R7** will be $(10000111)_2$, and it's a binary form of decimal number **135**. So different rule can give birth to different binary sequence and in result corresponding different decimal values.

Like binary addition and subtraction DNA Exclusive-OR or (XOR) operation can also be implemented according to the traditional binary form. As we have eight kinds of DNA encoding rules, so that means we also have eight DNA-XOR rules. One sort of XOR-operation is shown in Table 2.

TABLE 2. DNA XOR rules.

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

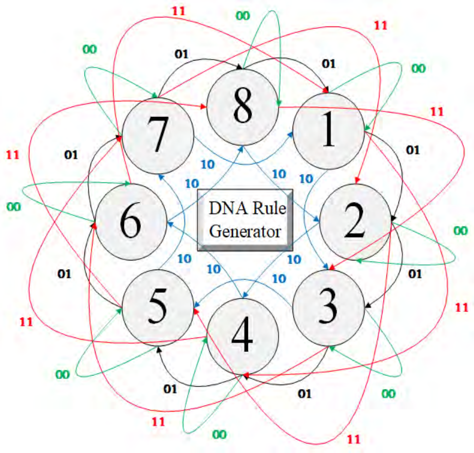


FIGURE 6. FSM based DNA rule generator.

Like addition and subtraction, DNA rules mostly remain the same during encryption or decryption of early encryption schemes. In this paper, we proposed a new concept of *Finite State Machine* **(FSM)** for generating a different rule for each sub-matrix as shown in Fig. 6. This **FSM** is a mathematical model of computation having **four** possible outputs based on **two bits** of input. This **FSM** can be used for any type of two bits of DNA rule structure but in this paper, we have used it for the rules that are listed in the above Table 1. This **FSM** based logic comprises of eight states that means eight rules and each state has **4** possible output states that lead to next rule state, so it means $\mathbf{8^4} = \mathbf{4096}$ different rules combination, while rule transition from one to another depends upon the following four combinations, either **00**, **01**, **10** or **11**. The general formula equation to determine the DNA rule is as follow.

$$DNA_{LR} = \begin{cases} [P1+P2]mod8+1 & K=1 \\ (K-1)_{f_b} \mid (K-1)_{l_b} mod8+1 & K\neq 1 \end{cases} \tag{15}$$

Whereas $\boldsymbol{f_{bit}}$ and $\boldsymbol{l_{bit}}$ are the first bit and last bit respectively of a particular sub-matrix, **K** is the total sub-block matrices of an image. Whereas ***P1*** and ***P2*** are the integer values of a particular pair of the hexadecimal form of the hash key string. For example, we have three sub-matrices SM_1, SM_2, and SM_3 as given below. Let suppose integer values of ***P1*** and ***P2*** of hash key-string through $int[K(0:2), 16]$ and $int[K(30:32), 16]$] comes **155** and **233** respectively. So from the formula, we can get the starting DNA rule for our first sub-matrix that is **5** as calculated below with Eq. (16). So the 1st sub-matrix will be encoded according to this **R5** provided in Table 1.

$$\begin{aligned} DNA_{SR} &= [P_1+P_2]mod8+1 \\ &\Rightarrow 388mod8+1=5 \end{aligned} \tag{16}$$

After getting the DNA rule for the 1st sub-matrix FSM starts and from 2nd to Nth sub-matrix it generates random rules on the basis of first and last bit of the previous sub-matrix.

$$SM_{1st}: \begin{pmatrix} 10 & 01 & 10 & 10 \\ 00 & 10 & 01 & 11 \\ 01 & 10 & 01 & 01 \\ 01 & 01 & 10 & 00 \end{pmatrix} \overrightarrow{R5}: \begin{pmatrix} A & T & A & A \\ C & A & T & G \\ T & A & T & T \\ T & T & A & C \end{pmatrix}$$

As first and last bit is **10**, so **FSM** takes **10** as an input and goes to state **7**, so **R7** is the rule for the 2nd sub-matrix.

$$SM_{2nd}: \begin{pmatrix} 11 & 11 & 11 & 10 \\ 10 & 10 & 00 & 00 \\ 01 & 00 & 01 & 01 \\ 11 & 01 & 11 & 11 \end{pmatrix} \overrightarrow{R7}: \begin{pmatrix} A & A & A & G \\ G & G & T & T \\ C & T & C & C \\ A & C & A & A \end{pmatrix}$$

First and last bit of the second sub-matrix is **11**, so **FSM** takes **11** as an input and goes to state **2** so **R2** is the rule for 3rd sub-matrix.

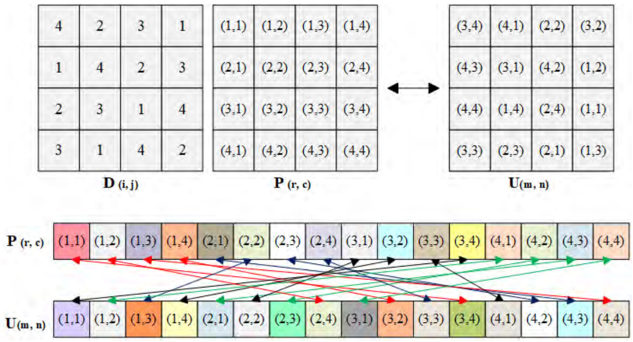
$$SM_{3rd}: \begin{pmatrix} 01 & 10 & 01 & 11 \\ 10 & 00 & 01 & 11 \\ 00 & 10 & 00 & 00 \\ 10 & 00 & 01 & 10 \end{pmatrix} \overrightarrow{R2}: \begin{pmatrix} G & C & G & T \\ C & A & G & T \\ A & C & A & A \\ C & A & G & C \end{pmatrix}$$

where ***SM*** stands for sub-matrix and so on for all the sub-matrices random DNA rules are generated, whereas the rule for **N$^{\text{th}}$** sub-matrix decided by **(N-1)$^{\text{th}}$** sub-matrix. Remember that each image has its own start and end rule based on the formula and the hash values of a particular image. So the whole process totally depends on particular two bits of an image sub-blocks. Thus, the FSM structure makes it very difficult to predict the next rule without knowing the working principle of **FSM-DNA** rule generator. Except that, which two bits out of **K** bits are taking part in the rule selection further increase the security and key-space. It also possesses the extensive complexity for hackers to guess or predict the working principle of FSM-DNA rule generator without getting enough prior information.

E. PIXEL SCRAMBLING

Scrambling is an efficient way to reduce the correlation among the pixels. In this paper, we have proposed a new scrambling algorithm for pixel shuffling that efficiently shuffles the pixels in a random way with respect to their previous position. It simultaneously alters the pixels position in rows and columns, thus increases the efficiency of the algorithm with respect to time while efficiently reducing the correlation between neighboring pixels.

Firstly, two random array sequences are generated by Logistic Sine system pseudo-random numbers generator **(LSS-PRNG)**, both are arranged in order and on the bases of first array sequence values, the 2nd array sequence values are shifted to get random matrix $\mathbf{D_{i,j}}$ and then according to the



4	2	3	1
1	4	2	3
2	3	1	4
3	1	4	2

D (i, j)

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

P (r, c)

(3,4)	(4,1)	(2,2)	(3,2)
(4,3)	(3,1)	(4,2)	(1,2)
(4,4)	(1,4)	(2,4)	(1,1)
(3,3)	(2,3)	(2,1)	(1,3)

U(m, n)

FIGURE 7. Pixel scrambling.

given algorithm shuffling of pixels took place. This scrambling technique can be demonstrated in term of $A\ \overbrace{Bijection}\ B$.

Suppose $\mathbf{P}_{(r,c)}$ is the pixel position in the original matrix **P** of an image, then $\mathbf{U}_{\mathbf{(m,\,n)}}$ will denote the scrambled location of a particular pixel in scrambled matrix **U**. It is one to one relation or bijection and can be described by the following formula.

$$P_{(r,c)} \mapsto U_{(m,n)}$$
$$\begin{cases} m = (i + D_{1,j} + j) modN + 1, & \text{for } 0 \le i \le N-1 \\ n = D_{m,j} \end{cases} \tag{17}$$

Whereas ***i=r*** and ***j=j***, while in the decryption process it is opposite, such as $B\ \overbrace{Bijection}\ A$ from $\mathbf{U}_{(m,n)}$ to $\mathbf{P}_{\mathbf{(r,\,c)}}$ and it can be described as follow.

$$U_{(m,n)} \mapsto P_{(r,c)}$$
$$\begin{cases} r = (i – D_{1,j} + j) modN + 1, & \text{for } 0 \le i \le N-1 \\ c = D_{r,j} \end{cases} \tag{18}$$

Fig. 7 gives a numerical and graphical explanation of our new scrambling algorithm while Algorithm.1 is the pseudo-code of a particular algorithm. As can be seen, no two pixels of the original and scrambled matrix is at the same position, even it's difficult to predict the relation between any two pixels.

As given below one to one mapping $(1, 1)$ of $\mathbf{P}_{\mathbf{(r,\,c)}}$ has the position of $(3, 4)$ in $\mathbf{U}_{\mathbf{(m,\,n)}}$.

Let's start from ***(i, j)*** that is (1,1), (1,2), (1,3) and (1,4). For **(1,1)** as according to algorithm $r = i = 1$ and $c = D_{(i,j)} = D_{(1,1)} = 4$, so according to the formula $m = (1 + D_{1,1} + 1) mod 4 + 1 = 2 + 1 = 3$, $n = D_{(3,1)} = 2$, hence $P_{(r,c)} = U_{(m,n)} \Rightarrow P_{(1,4)} = U_{(3,2)}$.

For **(1,2)**, $r = i = 1$, $c = D_{(i,j)} = D_{(1,2)} = 2$, So $m = (1 + D_{1,2} + 2) mod 4 + 1 = 1 + 1 = 2$ and $n = D_{(2,2)} = 4$ hence $P_{(r,c)} = U_{(m,n)} \Rightarrow P_{(1,2)} = U_{(2,4)}$.

For **(1,3)**, $r = i = 1$, $c = D_{(i,j)} = D_{(1,3)} = 3$, So $m = (1 + D_{(1,3)} + 3) mod 4 + 1 = 3 + 1 = 4$ and $n = D_{(4,3)} = 4$ hence $P_{(r,c)} = U_{(m,n)} \Rightarrow P_{(1,3)} = P_{(4,4)}$.

Algorithm 1 Scrambling

```
**Input:** Generate random matrix $D(i, j)$ of size of Image matrix $P(r, c)$
**Output:** Scrambled matrix $U(m, n)$
$I$ = read Image $(r, c)$
$(i, j)$ = size of $(I)$
LSS-PRNG = (array$_1$, array$_2$)
Sort $array_1$ and get $array_1{}'$; $array_1{}' = array_{1I}$
Sort $array_2$ and get $array_2{}'$; $array_2{}' = array_{2J}$
SET $D \in \mathbb{Z}^{(M \times N)}$; $U \in \mathbb{Z}^{(M \times N)}$;
for $i = 1$ to $N$; do
    for $j = 1$ to $M$; do
        $r = i, c = D(r, j)$;
        $m = ((i + D(i, j) + j) modN) + 1$;
        $n = D_{(m,j)}$
    end for
end for
for $i = 1$ to $M$; do
    for $j = 1$ to $N$; do
        $m = i, n = D(m, j)$;
        $r = ((i – D(i, j) + j) modN) + 1$;
        $c = D_{(r,j)}$
    end for
end for
```

Lastly for **(1,4)**, $r = i = 1$, $c = D_{(i,j)} = D_{(1,4)} = 1$, So $m = (1 + D_{(1,4)} + 4) mod 4 + 1 = 2 + 1 = 3$ and $n = D_{(3,4)} = 4$, hence $P_{(r,c)} = U_{(m,n)} \Rightarrow P_{(1,1)} = U_{(3,4)}$. Similarly, for (2,1), (2,2), (2,3), (2,4) and all remaining pixels location can be calculated according to the given formula.

III. IMAGE ENCRYPTION SCHEME

The block diagram of our proposed encryption scheme is shown in Fig. 5. While the step by step demonstration is given below.

Step:1 Calculate the hash value of an image, take the double hash of this hash string and sort this hash string as a key **K** while dividing the string for the initial values $\boldsymbol{x}_0$, $\boldsymbol{y}_0$, z_0 and $\boldsymbol{w}_0$ of the logistic sine system as described below.

$$x_0 = \frac{(k_1 \oplus k_3 \oplus k_5 \oplus k_7) + (k_9 \oplus k_{11} \oplus k_{13} \oplus k_{15})}{2^8} \tag{19}$$

$$y_0 = \frac{(k_2 \oplus k_4 \oplus k_6 \oplus k_8) + (k_{10} \oplus k_{12} \oplus k_{14} \oplus k_{16})}{2^8} \tag{20}$$

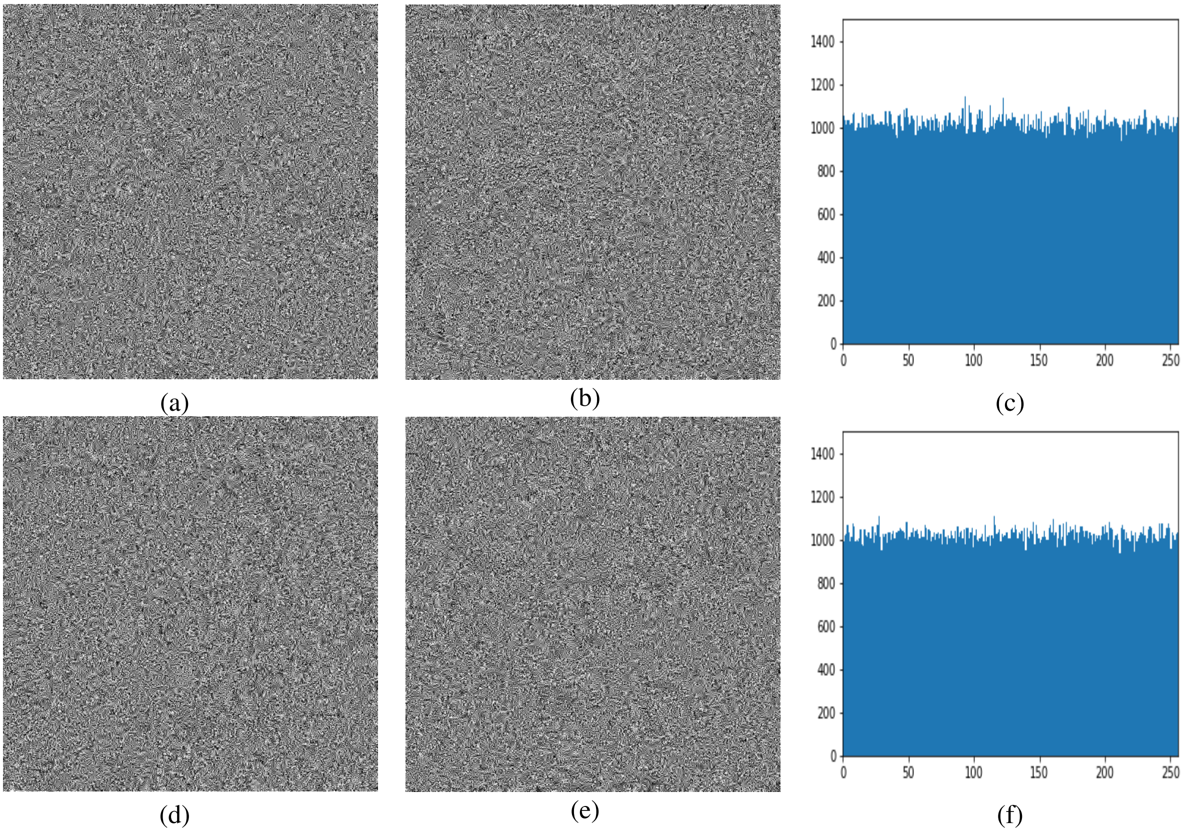
$$z_0 = \frac{(k_{17} \oplus k_{19} \oplus k_{21} \oplus k_{23}) + (k_{25} \oplus k_{27} \oplus k_{29} \oplus k_{31})}{2^8} \tag{21}$$

$$w_0 = \frac{(k_{18} + k_{20} + k_{22} + k_{24} + k_{26} + k_{28} + k_{30} + k_{32})}{(k_{18}, k_{20}, k_{22}, k_{24}, k_{26}, k_{28}, k_{30}, k_{32}) \times 2^5} \tag{22}$$

Step:2 Create two random sequence arrays through LSS-PRNG by using the following equation.

$$X_1 = x_0 \times (y_0 + z_0) mod 1 \tag{23}$$

$$R_1 = x_0 \times (y_0 + w_0) mod 4 \tag{24}$$



(a) (b) (c)

(d) (e) (f)

FIGURE 8. **Key sensitivity result with one bit changed key. (a) Encrypted with K2. (b) Subtraction. (c) Histogram of subtracted image. (d) Encrypted with K3. (e) Subtraction. (f) Histogram of subtracted image.**

Sort the values as a sequence in **I** and **J** arrays, get the index of the data and fill up the rows and columns of the matrix $\mathbf{D_{(i,j)}}$.

Step:3 Perform the shifting operation in such a way that each integer value of **J** shift towards the left according to the integer value of array **I**. Transformed the input image of size $M \times N$ into matrix form and performed scrambling according to the Eq. (17) of pixel scrambling section. While during decryption, Eq. (18) performed the reversed operation.

Step:4 Transform every pixel of the scrambled image $u \times v$ into binary and divide the image matrix into $p \times q$ size sub-blocks. Where $u \times p = M$ and $v \times q = N$.

Step:5 Pick up the 1st sub-block and divide it into two equal column blocks called left block and right block termed as $\mathbf{L_k}$ and $\mathbf{R_k}$ respectively. Get the starting bit for this first block through the double hash value of the hash key string of the image by using Eq. (2) and Eq. (3) described in Feistel section. Do this to all the sub-blocks of the specified image in a sequence.

Step:6 Assemble all the blocks and get the initial configuration matrix from the double hash key string **K** of the hash value of the image $M \times N$ as described in the pseudo-code of Algorithm.2.

Step:7 Get the local rule from this initial configuration matrix for the cellular part of the 1st sub-block with the following formula.

$$CA_{LR} = [(0^s \times 48) + (1^s \times 49)] mod 255 + 1 \tag{25}$$

Conversion of this value to binary will determine the values of **NW, NE, SW, SE, N, S, W, E** either **1** or **0**. Whereas the local rule for the last or N^th sub-block is gotten from **(N-1)^th** sub-matrix with the help of following formula.

$$LR = \begin{cases} I_a + I_b] mod 255 + 1 & N = 1 \\ [(N-1)_\alpha + (N-1)_\beta] mod 255 + 1 & N \neq 1 \end{cases} \tag{26}$$

where **a** and **b** are the $0^s \times 48$ and $1^s \times 49$ values respectively.

Step:8 Set $\boldsymbol{N = N + 1}$, do all respective steps in a loop until all the sub-block matrices of the particular window get updated through the local rules of Eq. (26).

Step:9 Update the initial configuration matrix according to local rules of step.8 for the respective sub-matrices and performed **XOR** operation with the particular sub-matrix of size $l \times m$ to get the updated sub-matrix as follow.

$$S^{t+1}_{P(i,j)} = S^t_{P(i,j)} \oplus S^{t+1}_{DN(i,j)} \tag{27}$$

Performed this state update process for all the sub-block matrices of size $l \times m$ of a particular window.

Step:10 Assemble all these updated matrices for DNA conversion, get the DNA rule for the first sub-matrix by the following formula:

$$DNA_{STR} = (P_1 + P_2) mod 8 + 1 \tag{28}$$

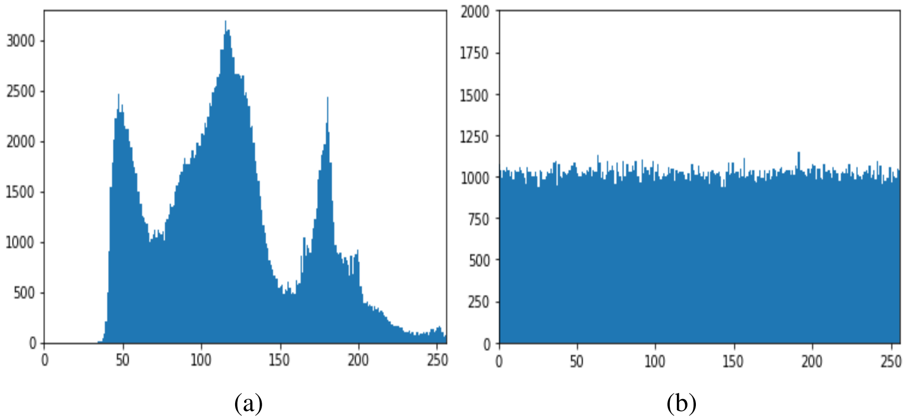


FIGURE 9. Histogram of plain opera image and ciphered image. (a) Plain image. (b) Ciphered image.

Algorithm 2 Initial Configuration

```
Input: 256-Hash key string
Output: $initial\ configuration = np.zeros((8, 8))$
$t = 0$
for $i$ in range (8); do
    $print, (K[t : 2]);$
    $keyy = bin(int(K[t : t + 2], 16));$
    $keyy = keyy[2 :]$
    if $len(keyy) < 8$ than
    $key = str('0') * (8 - len(keyy))$
    $keyy = key + keyy$
    $ky = j\ for\ j\ in\ keyy$
    $initial\ configuration\ [i, :] = ky$
    else
    $ky = j\ for\ j\ in\ keyy$
    $initial\ configuration\ [i, :] = ky$
    $t+ = 2$
end for
end for
```

Whereas $\mathbf{P}_1$ and $\mathbf{P}_2$ are gotten from the hash value of the image by $int[K(0 : 2), 16]$ and $int[K(30 : 32), 16]$ respectively. Follow **FSM-DNA** rule generator to get the DNA local rule for the next sub-matrix. Continue the process until all the sub-matrices are converted to DNA form.

Step:11 Rejoin all these DNA blocks and get the universal rule for decoding the DNA matrix by using above-mentioned formula and getting the value of **P** through the W_0 by $int[K(0 : 8), 16]$. Decode the whole DNA matrix back into binary and that will be our cipher image.

While decryption method is the inverse of encryption method so before decrypting the cipher image; the key must be transmitted to the receiver through the secure way, along with other information about which two pairs of the hash value are used to get the value of P1 and P2. In our way, as the actual key is the double hash value of plain image while we are transmitting the single hash value key, so its also less risk of being hacked will let any information to reveal as compared to transmitting original key.

IV. RESULTS AND DISCUSSION

This section provides the simulation result of our proposed encryption scheme. The experimental results are manipulated by **Python** 3.6.5 installed over personal computer having 3.1 GHz CPU and memory of 4GB, and with an operating system of Window 10 Enterprise.

Results, tables, and figures are presented in the next Security Analysis section. Table 3 contains the test parameter and other values that we have used as a test. The plain test image is Opera. The histogram of the plain image and the ciphered image is shown in Fig. 9(a) and Fig. 9(b), respectively. While the plain image, its ciphered form, and the retrieved image are shown in Fig. 9(a), (b), (c) respectively. Whereas Fig. 14 showing all the test images that we have used for the result part.

V. SECURITY ANALYSIS

A. KEY ANALYSIS

The security of an encryption algorithm ought to have vast key space more sensitive to the secret key to tackling a different kind of attacks such as statistical attacks, differential attack, known plain text attacks and exhaustive attacks [34], [35].

The large size of the key space also makes brute-force attacks infeasible. Larger and more sophisticated the secret key regarding a bit change; the more processing power and time are required to crack the encrypted information. The next sections discussed the security performance of the proposed algorithm. The integer value of **P1** and **P2** along with hexadecimal key strings for our test results are enlisted in Table 3.

B. EXHAUSTIVE ATTACK

In our proposed algorithm, the hash value **K** generated by the hash (SHA-256) function of the image, thus the hash value of key and integer values of **P1** and **P2** are termed as a secret key.

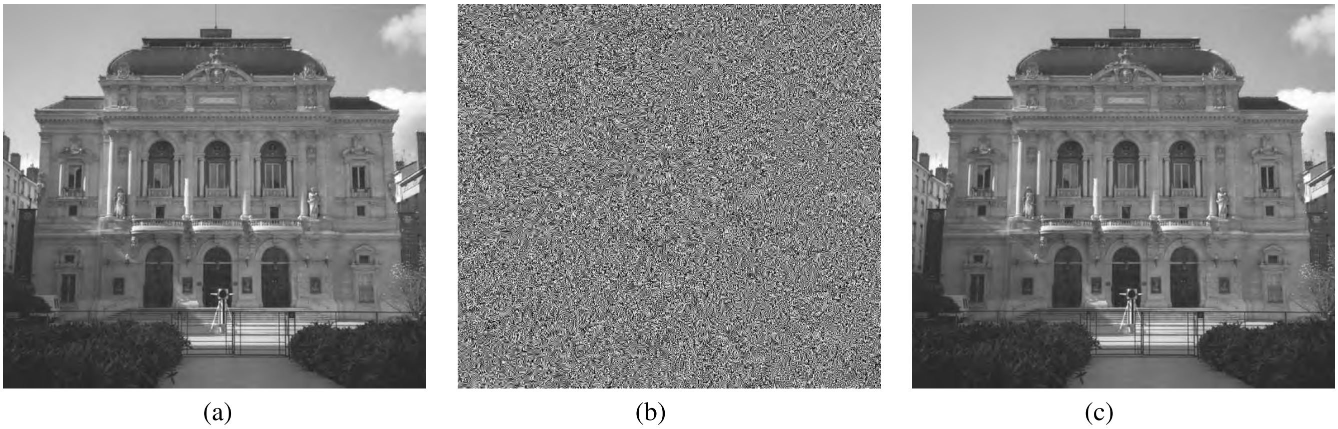


FIGURE 10. Plain image of opera and its corresponding cipher and decrypted image. (a) Plain image. (b) Ciphered image. (c) Decrypted image.

TABLE 3. Experiment parameters.

Items	System Parameters and Values
256-bit hexa-decimal key	DACBAE1DF7118AC54A2B3A6E5E2386338A4159 CB70DF86DFF4E0045005632C87
P_1	$int(K[0:2], 16) = 181$
P_2	$int(K[2:4], 16) = 247$

Except for that particular input two-bits combination of **FSM-DNA** rule generator is also termed as a key. It is kenned that the SHA-256 hash keyspace with the intricacy of the finest assailant is $\mathbf{2^{128}}$. That is more prodigiously colossal than $\mathbf{2^{100}}$, thus this designates our proposed algorithm is ample to obviate any brute force attack or exhaustive attacks.

C. KEY SENSITIVITY

The sensitivity of algorithms towards the secret key during encryption and decryption is the key point of the robustness of an encryption algorithm. Higher the sensitivity the more secure is the information because only a slight change in the key will lead towards an entirely different cipher image. That means no one can recover the original image except having the correct secret key. We tested the key sensitivity for both encryption and decryption part over the test image Opera of Fig. 10(a). While Fig. 10(b) and Fig. 10(c) is the corresponding cipher image and retrieved image, respectively. We have performed the key sensitivity test with different keys that are only one bit different from the correct encryption key. The original key ***K1*** and slight or a bit changed keys ***K2*** and ***K3*** respectively are given below.

K1 = DA CB AE 1D F7 11 8A C5 4A 2B 3A 6E 5E 23 86 33 8A 41 59 CB 70 DF 86 DF F4 E0 04 50 05 63 2C 87

K2 = DA CB AE 1D F7 11 8A C5 4A 2B 3A 6E 5E 23 86 33 8A 41 59 CB 70 DF 86 DF F4 E0 04 50 05 63 2C 88

K3 = CA CB AE 1D F7 11 8A C5 4A 2B 3A 6E 5E 23 86 33 8A 41 59 CB 70 DF 86 DF F4 E0 04 50 05 63 2C 87

Firstly we tested for encryption part; Fig. 8(a), is the cipher image encrypted with ***K2***, Fig. 8(b) is the subtraction of Fig. 8(a) and Fig. 9(b), while Fig. 8(c) is the histogram of the subtracted resultant image. Similarly, Fig. 8(d) is the encrypted image with ***K3*** and Fig. 8(e) is the corresponding subtracted image of Fig. 8(d) and Fig. 9(b). While Fig. 8(f) is the corresponding histogram of the subtracted image.

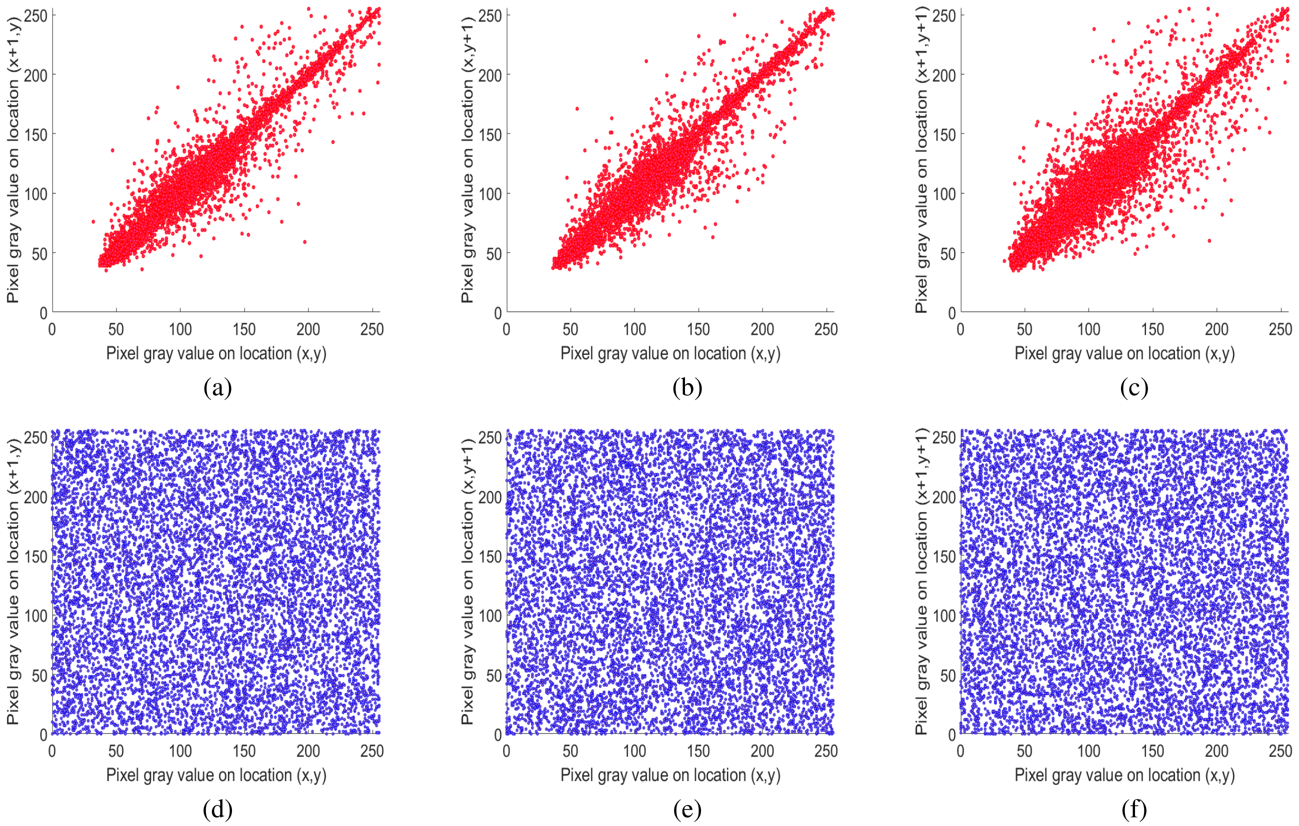
These test results show that our algorithm has strong key sensitivity. The key sensitivity also evaluated for the decryption process. We tried to decrypt the original cipher image with ***K2*** and ***K3*** respectively. It has clearly shown in Fig. 12 (a) and Fig. 12(c) while the histogram of Fig. 12(a) and Fig. 12(c) is shown in Fig. 12(b) and Fig. 12(e) respectively. We can see that the retrieved images are looks like noise and no information is visible though having only one-bit change between original key ***K1*** and slightly changed keys ***K2*** and ***K3***. This verified that our proposed algorithm is too sensitive to the secret key.

D. STATISTICAL ATTACKS OR HISTOGRAM ANALYSIS

An unvarying and rosiness dissemination is necessary for any good encryption algorithm otherwise; an approximate degree of data can be seeped by common statistical attacks. The histogram is a common approach to get the distribution of an image pixel values. Fig. 8(a) and Fig. 8(b) are the histograms of the plain image and ciphered image. It is clear that the histogram of a ciphered image is almost flat and uniform which leads to statistical attacks invalid. In addition, histogram variances mainly used to quantitatively examine the uniformity of an image. Lower variance means higher uniformity of an image, alternatively the better security of the particular algorithm. Variance can be obtained through the following formula.

$$var(I) = \frac{1}{k^2}\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{1}{2}(I_i - I_j)^2 \tag{29}$$

Here $I = I_1, I_2, I_3, \ldots I_{256}$ represents the vector of histogram values and $\mathrm{I_i}$ and $\mathrm{I_j}$ denote the number of pixels where gray value is equivalent to i and j respectively. For the test images Lena (256×256), cameraman (256×256) and baboon (256 × 256) the variance of plain images and corresponding



(a) (b) (c)

(d) (e) (f)

FIGURE 11. Pixel correlation in plain image (a)–(c) and corresponding direction in cipher image (d)–(f).

TABLE 4. The variance of histogram of test images comparison.

Algorithms	Plain Image	Ours	Ref. [36]	Ref. [37]	Ref. [38]
image variance	Lena 256 × 256 38952.86	Cipher image 226.209	244.31	276.39	260.71
	Cameraman 256 × 256 161272.17	Cipher image 233.281	-	-	-
	Baboon 256 × 256 62914.39	Cipher image 229.59	-	-	-

ciphered images are given in Table 4. The variance of plain Lena image is **38952.8671** while the variance of our ciphered image is **226.2091** which is lower than the Ref. [36], Ref. [37] and Ref. [38]. While the variance of cameraman and baboon are **233.281** and **229.59** respectively. The result shows our encryption scheme is better and more efficient, that gives better results as compared to others.

E. CORRELATION COEFFICIENT ANALYSIS

For correlation among the adjacent pixels test, we selected randomly **8K** sets of adjacent pixels in **XYZ** plane i.e. horizontal, vertical and diagonal direction respectively of both the plain and its corresponding cipher image.

The correlation coefficient $\boldsymbol{C}_{r(x,y)}$ of two neighboring pixels calculated rendering to the subsequent formulas.

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{30}$$

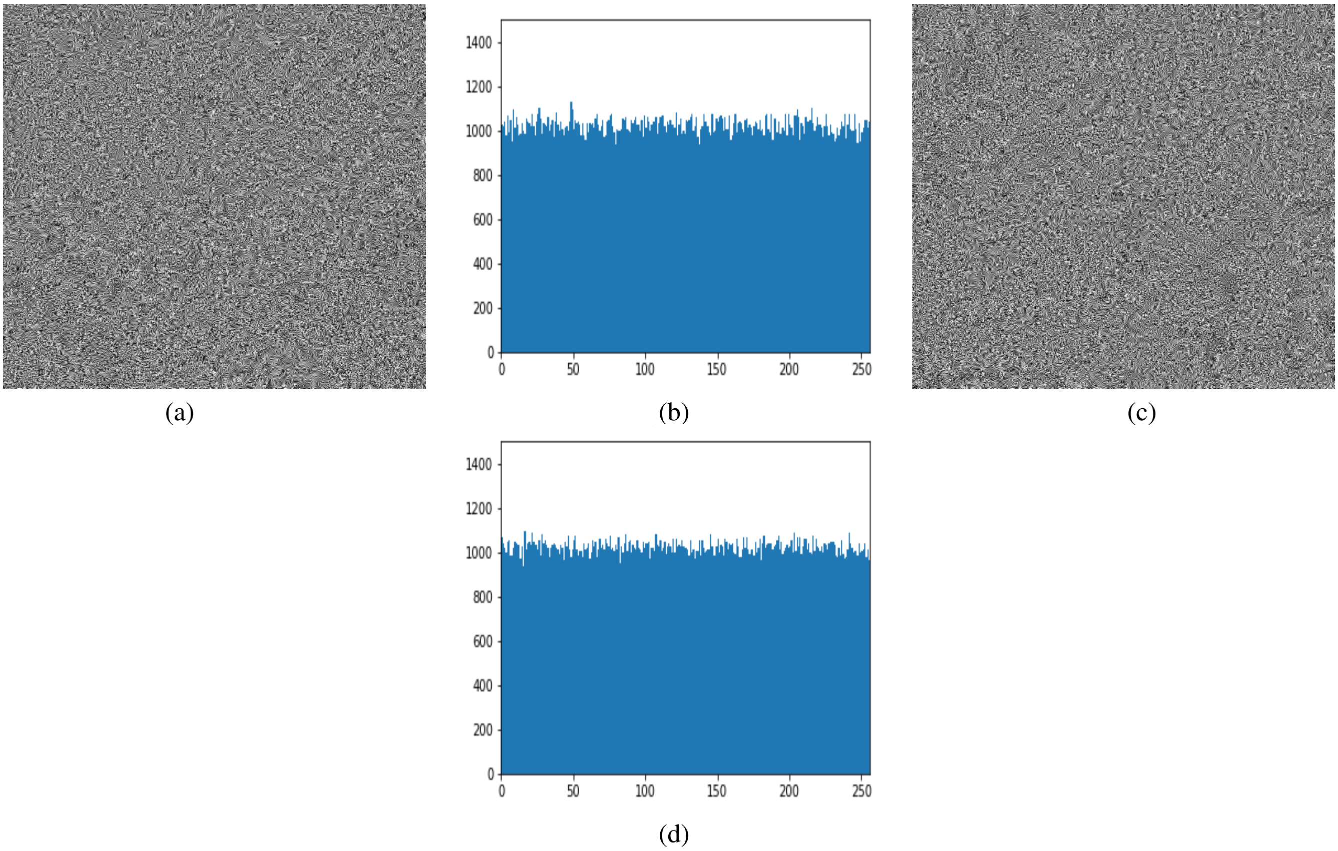
$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \tag{31}$$

$$C_{ovariance}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)) \tag{32}$$

$$C_{r(x,y)} = \frac{C_{ovariance}(x, y)}{\sqrt{D(x) \times D(y)}} \tag{33}$$

In this formula ***x,y*** is the gray values of neighboring pixels, ***N*** is the total number of pixels selected from the image, $\boldsymbol{C}_{ovariance}(x, y)$ is the covariance, while ***D(x)*** is the variance and ***E(x)*** is the mean. Table 5 shows the correlation values of our algorithm for different test images. While Fig. 11(a), (b), (c) is the plot of correlation of adjacent pixels in the horizontal, vertical and diagonal direction of plain test image Opera. While Fig. 11(d),(e),(f) is the corresponding pixels location in the ciphered image.

The correlation table results show that the correlation among the adjacent pixels in the plain image is higher that is nearly equal to **1** in all direction. While in cipher images all are less than **0.0025**, which depict that our proposed



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

FIGURE 12. Decrypted images with K2, K3 and their corresponding histograms. (a) Decrypted with K2. (b) Histogram of (a). (c) Decrypted with K3. (d) Histogram of (c).

TABLE 5. Pixel correlation coefficient values.

Image Dimension	types	Correlation Coefficient		
		Horizontal	Vertical	Diagonal
Opera	Plain	0.9755	0.9767	0.9478
512 × 512	Cipher	-0.0016	-0.0024	-0.0010
Lena	Plain	0.9023	0.9492	0.8862
512 × 512	Cipher	-0.0015	-0.0023	-0.0061
New-York	Plain	0.9550	0.9541	0.9224
512 × 512	Cipher	-0.0077	-0.0032	-0.0021
Pepper	Plain	0.9810	0.9837	0.9555
512 × 512	Cipher	-0.0010	-0.0019	-0.0012
Baboon	Plain	0.8760	0.7752	0.7384
512 × 512	Cipher	-0.0046	-0.0014	-0.0036
Light-House	Plain	0.9610	0.9497	0.9297
800 × 600	Cipher	-0.0018	0.0039	-0.0016
Pentagon	Plain	0.9604	0.9558	0.9251
1200 × 800	Cipher	0.0046	0.0020	-0.0139

algorithm has efficiently reduced the correlation between the adjacent pixels and its hard for hackers to get useful information from our cipher image through pixel correlation method.

F. DIFFERENTIAL ATTACK

Typically, hackers create small amendment in the original image and then use the encryption methodology to encrypt the alike images before and after these minor changes. Through this technique, they try to crack out the connection between plain and cipher images. Therefore, to measure the robustness of our proposed algorithms against such attacks we employed the number of pixel change rate (**NPCR**) and unified average changing intensity (**UACI**) for our algorithm. Both **NPCR** and **UACI** can be calculated by the following formulas.

$$NPCR_{U_1,U_2} = [\frac{\sum_{i,j} I(i,j)}{H \times M}] \times 100 \tag{34}$$

$$UACI_{U_1,U_2} = \frac{1}{H \times M}[\sum_{i,j} \frac{|U_1(i,j) - U_2(i,j)|}{2^8 - 1}] \times 100 \tag{35}$$

where $H \times M$ is the height and width of the image, while $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are two different ciphered images before and after one pixel of the plain image is changed. Whereas ***I(i,j)*** is defined as

$$I(i,j) = \begin{cases} 0 & U_1(i,j) \neq U_2(i,j) \\ 1 & otherwise \end{cases} \tag{36}$$

In the above equation, ***I*** depict the difference between $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. **NPCR** and **UACI** are presented in Table 7, While Table 8 depicts the values when pixel value has changed at a particular position that is mentioned in the table.

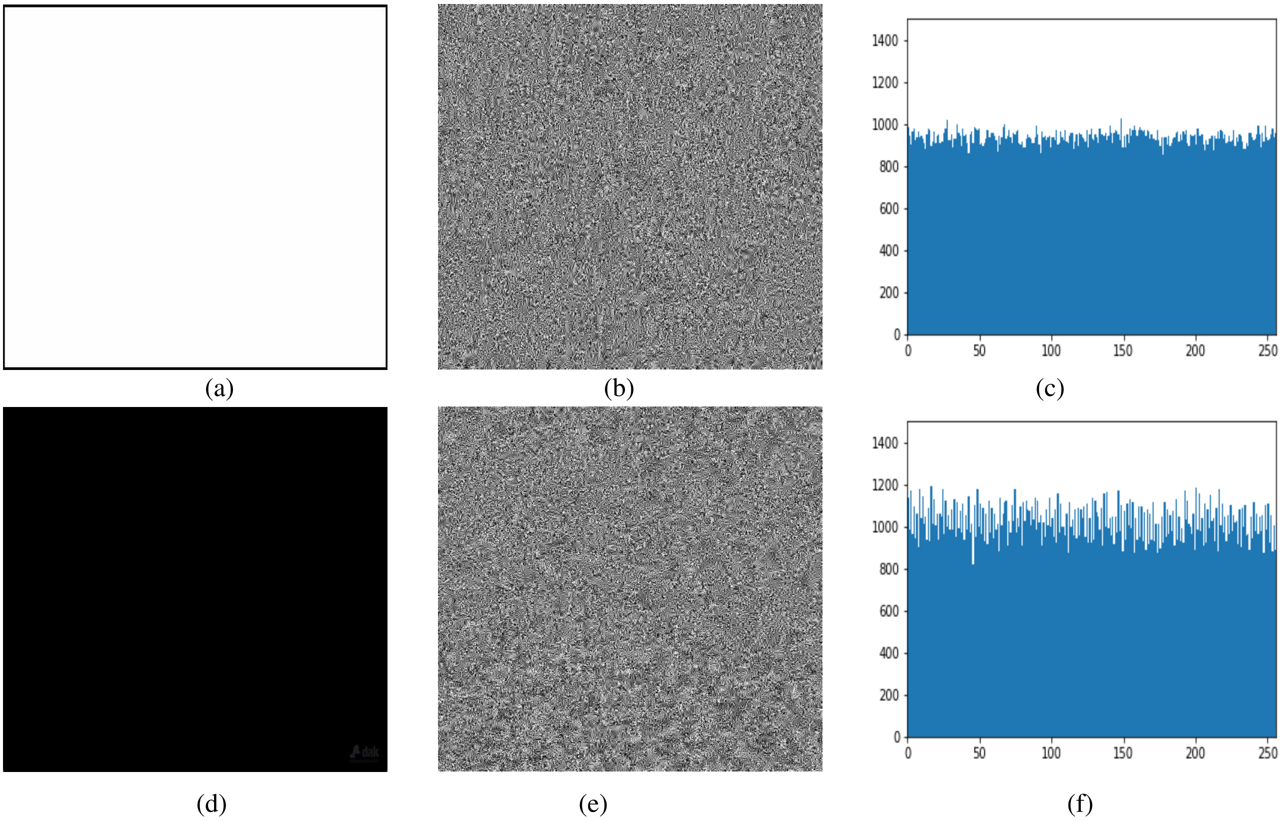


FIGURE 13. Cipher images and corresponding histogram of all white and black image. (a) All white. (b) Ciphered image. (c) Histogram of (b). (d) Black. (e) Ciphered image. (f) Histogram of (e).

TABLE 6. Comparison between our algorithm and others for test image cameraman of size 256 × 256.

S.No	NPCR	UACI	Entropy	Corr. Coefficient		
				Horiz-ontal	Vert-ical	Diag-onal
Camera	-	-	6.904	0.899	0.954	0.915
Ours	99.99	0.334	7.997	0.001	-0.001	0.001
Ref. [36]	99.60	0.334	7.996	-0.008	0.011	0.012
Ref. [37]	99.66	0.332	7.997	-0.016	-0.002	0.010
Ref. [39]	99.62	0.334	7.997	-0.008	-0.021	0.006
Ref. [40]	99.62	0.334	7.996	-0.005	0.008	0.010

G. INFORMATION ENTROPY

In information theory, entropy is the most significant feature of the disorder. We can say numerical property reflecting the randomness associated unpredictability of an information source called entropy [41].

Let *v* be an information source; then the entropy computing formula will be as follow.

$$H(v) = \sum_{i=0}^{2^n-1} \rho(v_i) log_{10} \frac{1}{\rho(v_i)} \quad (37)$$

Here $\rho(v_i)$ denotes the probability of symbol $\boldsymbol{v}_i$. The ideal entropy value for a random image with a gray level of 2^8 is **8** [42]. Which means the closer the entropy value is, the more is the haphazardness of an image, conclusively less information disclosed by the encryption scheme. Table 9 and Table 10 shows the entropy values of different test images. We can see that entropy values for all the ciphered images generated by our algorithm are almost equal to ideal value **8**. That means our algorithm performance is highly satisfactory and ciphered image could hardly leak any useful information to the hackers.

Also, Table 6, listed the entropy, **NPCR** and correlation comparison for the test image cameraman (256 × 256) between our algorithm and the Ref. [36], Ref. [37], Ref. [39] and Ref. [40]. The entropy values, as well as correlation and **NPCR** values of our algorithm, are better than others.

TABLE 7. NPCR and UACI for different test images.

Images	Opera	Lena	Baboon	New-York
NPCR	99.996	99.996	99.999	99.776
UACI	>0.3343	>0.3355	>0.3350	>0.3336
Images	Pentagon	Light-house	Peppers	Camera-man
NPCR	99.986	99.987	99.984	99.996
UACI	>0.3351	>0.3352	>0.3343	>0.3351

TABLE 8. NPCR and UACI of opera image 512 × 512 for different positions.

Position	(1, 1)	(25, 35)	(100, 190)	(230, 250)
NPCR	99.996	99.929	99.993	99.649
UACI	>0.334	>0.333	>0.335	>0.332

FIGURE 14. All test images. (a) Lighthouse. (b) New-York. (c) Lena. (d) Pentagon. (e) Baboon. (f) Peppers. (g) Cameraman. (h) Bridge.

TABLE 9. Information entropy of different test images.

Values	Images	Baboon	Lena	Pentagon	Camera-man
Information entropy	plain	7.3595	7.4750	7.1451	6.9046
	cipher	7.9993	7.9976	7.9998	7.9972

It proves our algorithm has better performance as compared to others.

Recently, in [43] a new image uncertainty measure introduced using Shannon entropy over native image blocks. The proposed local measure of Shannon entropy overcome numerous flaws of universal Shannon entropy measures. Discriminating randomness comparison between different size images, inadequacy to discriminate image randomness afore and afterward shuffling, a probably erroneous score for integrated images.

This local Shannon entropy is employed to compute the unpredictability of our cryptography algorithm. We can calculate the (n, P_b) local Shannon entropy measurement by using the following method for local image blocks.

Firstly, select randomly non-overlapping image blocks like $\boldsymbol{S_1, S_2, S_3, \ldots, S_n}$ with P_b pixels within test image $M \times N$ of intensity scale $\boldsymbol{Q}$. Secondly, calculate the Shannon entropy $H(S_i)$ for all $i \in 1, 2, 3, 4, \ldots, n$ via Eq. (37). Thirdly, compute the sample mean of Shannon entropy over these particular **n** image blocks through the following equation.

$$\bar{H}_{N,P_b}(m) = \sum_{1}^{N} \frac{H(S_i)}{n} \tag{38}$$

TABLE 10. Information entropy of different test images.

Values	Images	Opera	Light house	New-York	Peppers
Information	plain	7.3596	7.4559	6.9315	7.5831
entropy	cipher	7.9992	7.9996	7.9994	7.9993

TABLE 11. Local entropy values of different test images.

Values	Images type	Cameraman	Opera	New-York	Peppers	Pentagon
Local	plain	5.557	5.687	5.450	6.095	6.140
entropy	cipher	7.905	7.908	7.906	7.908	7.907

TABLE 12. Correlation and entropy values of all white and black images.

Type	Image Size	Entropy values	Correlation Coefficient		
			Horizontal	Vertical	Diagonal
Full White	800×600	0	-	-	-
Cipher	-	7.99	-0.007	0.007	-0.008
≈ Black	512×512	0.060	0.885	0.889	0.885
cipher	-	7.99	-0.009	0.033	-0.008

In our experiments we have taken random blocks of 44 × 44, the result is shown in Table 11. From the results, we can say that our proposed encryption algorithm has good local randomness because the local entropy value of cipher image generated by our algorithm is satisfactory and ≥ 7.905 (see Table 11).

H. KNOWN PLAIN-TEXT ATTACKS

High sensitivity towards the single pixel change of plain image is the key point of the robustness of any encryption scheme. Hackers used known the plain text and chosen plain attacks to get useful information about the working of an encryption scheme. So for this purpose, they used the fully white or full black image to make the diffusion or permutation process ineffective and then attempt to get useful information.

In our proposed algorithm Hash function for the plain image, **Feistel** based bit diffusion makes such attacks ineffective. Except for that **2D-CA** Moore neighborhood-based local rule and **FSM** based DNA random rule generator make our algorithm robust against these known plain text and chosen plain text attacks.

Table 12 listed the entropy values and pixel correlation values of test image All white (800 × 600) and black image (512 × 512). Fig. 13(a), (d) is the all white and black test images, Fig. 13(b), (e) is the corresponding cipher images of full white and black images, respectively. While Fig. 13(c), (f) is the corresponding histograms of ciphered images. Histogram clearly showing that our algorithm can effectively withstand against these known plain text and chosen plain text attacks.

I. PERFORMANCE COMPARISON

Performance and speed are vital characteristics of any encryption algorithm after the satisfactory security level of a particular algorithm. Chaos-based schemes mostly carry diffusion and permutation steps. So round numbers directly influence the speed of the particular encryption scheme.

TABLE 13. Performance comparison between different methods having satisfactory security level.

Algorithms	NPCR	UACI	Total Number of Rounds		
			Image Scanning	Permutation	Diffusion
Ours	>0.998	>0.333	1	1	2
Ref. [15]	>0.996	>0.333	6	3	3
Ref. [36]	>0.996	>0.333	4	2	2
Ref. [45]	>0.996	>0.333	2	2	2
Ref. [46]	>0.996	>0.333	4	4	2
Ref. [47]	>0.996	>0.333	6	3	3

Table 13 enlisted comparison of our algorithm with the Ref. [15], Ref. [36], Ref. [45], Ref. [46], and Ref. [47] that posses some satisfactory security level. It is clear from the table that our algorithm gives satisfactory results **NPCR ≥ 99.8** and **UACI ≥ 0.333** with minimum rounds. So in this prospect, our proposed algorithm has better efficiency in both running time and security as compared to the Refs. [15], [36], [45], [46] and [47].

VI. CONCLUSION AND FUTURE WORK

We have proposed a new image encryption algorithm for classified or personal images to protect them from being misused or stolen. Even if hackers succeed to steal them Our algorithm promise high security of image and differs from others in the following ways. Firstly, Image binary form is divided into sub-blocks and a new Feistel based fast bit inversion method is employed to change the pixels values and reduce the correlation effect. Secondly, after analysis of earlier research work what we found is that the DNA encoding rules mostly remain fixed are sometimes taken as a key (changing from 1 to 8) which is actually a weak point of any algorithm because useful information from the ciphered image can be extracted faster than the algorithm in which rules are not fixed. However, in our algorithm for each sub-matrix **FSM** based DNA rule generator efficiently generate random rules for each sub-matrix based on its first and last bit combination. This provides extra security except for long keyspace because which two bits along with what order are taking part as input required more time to crack the DNA rule. In other way, we can say that **FSM** enhanced the key sensitivity because each sub-matrix has its own DNA rule and any slight change in key or any sub-matrix will lead to completely different rules structures for subsequent matrices. Thirdly, in 2D-CA part Moore neighborhood local rule structure is adopted that means for each matrix; a different combination of neighbors, this increases the performance and reduces the complexity. Besides that, local rules (LR) based strategy decide how many, which and whom cells will take part in the next matrix update process. As **CA-LR** is decided by the previous block that means each sub-block has a direct relation with the previous block and initial configuration matrix. Lastly, secure hash (**SHA-256**) algorithm has been used to get the secret key from the plain image, while the double hash function provides us benefit to send the secret key through normal channel because actual key, initial values for the **LSS-PRNG** part and the initial configuration matrix for 2D-CA, all linked with the double hash value of the plain image. Hence, our proposed encryption scheme is more secure and highly sensitive to the plain image. While our algorithm has some limitation also, like its work for gray-scale images so for future keeping aside the complexity of dealing with RGB channel of the colored image, it can be implemented for the color images too.

ACKNOWLEDGMENT

The authors are very grateful to our information security lab team members including our colleagues of the computer science department and other fellows for their valuable suggestions. They also would like to thank the anonymous referees for their valuable suggestions to improve the quality of this paper.

REFERENCES

[1] O. A. S. Alkishriwo, "An image encryption algorithm based on chaotic maps and discrete linear chirp transform," *CoRR*, vol. abs/1807.02647, 2018.

[2] X. Huang and G. Ye, "An efficient self-adaptive model for chaotic image encryption algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 19, no. 12, pp. 4094–4104, 2014.

[3] Q. Liu, P.-Y. Li, M. C. Zhang, Y.-X. Sui, and H.-J. Yang, "A novel image encryption algorithm based on chaos maps with Markov properties," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 20, no. 2, pp. 506–515, 2015.

[4] M. Salleh, S. Ibrahim, and I. F. Isnin, "Image encryption algorithm based on chaotic mapping," *J. Teknologi*, vol. 39, no. 13, pp. 1–12, 2003.

[5] S. C. Satapathy, S. K. Udgata, and B. N. Biswal, *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)* (Advances in Intelligent Systems and Computing), vol. 247. Odisha, India: Springer, 2014.

[6] N. K. Pareek, V. Patidar, and K. K. Sud, "Diffusion–substitution based gray image encryption scheme," *Digit. Signal Process.*, vol. 23, no. 3, pp. 894–901, 2013.

[7] A. Jolfaei, X.-W. Wu, and V. Muthukkumarasamy, "Comments on the security of 'Diffusion–substitution based gray image encryption' scheme," *Digit. Signal Process.*, vol. 32, pp. 34–36, Sep. 2014.

[8] X. W. Li, D. H. Kim, S. J. Cho, and S. T. Kim, "Integral imaging based 3-D image encryption algorithm combined with cellular automata," *J. Appl. Res. Technol.*, vol. 11, no. 4, pp. 549–558, 2013.

[9] R. I. Al-Khalid, R. A. Al-Dallah, A. M. Al-Anani, R. M. Barham, and S. I. Hajir, "A secure visual cryptography scheme using private key with invariant share sizes," *J. Softw. Eng. Appl.*, vol. 10, no. 1, pp. 1–10, 2017.

[10] I. Ozturk and I. Sogukpinar, "Analysis and comparison of image encryption algorithms," *Int. J. Inf. Technol.*, vol. 1, no. 2, pp. 108–111, 2004.

[11] N. E. Robbins, C. Trontin, L. Duan, and J. R. Dinneny, "Beyond the barrier: Communication in the root through the endodermis," *Plant Physiol.*, vol. 166, no. 2, pp. 551–559, 2014.

[12] R. Rizk and Y. Alkady, "Two-phase hybrid cryptography algorithm for wireless sensor networks," *J. Elect. Syst. Inf. Technol.*, vol. 2, no. 3, pp. 296–313, 2015.

[13] Gaytri, S. Suri, and R. Vijay, "An implementation and performance evaluation of an improved chaotic image encryption approach," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Jaipur, India, Sep. 2016, pp. 1458–1463.

[14] Z. Hua, S. Yi, and Y. Zhou, "Medical image encryption using high-speed scrambling and pixel adaptive diffusion," *Signal Process.*, vol. 144, pp. 134–144, Mar. 2018.

[15] Y. Zhang and D. Xiao, "An image encryption scheme based on rotation matrix bit-level permutation and block diffusion," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 19, no. 1, pp. 74–82, 2014.

[16] N. Zhou, S. Pan, S. Cheng, and Z. Zhou, "Image compression–encryption scheme based on hyper-chaotic system and 2D compressive sensing," *Opt. Laser Technol.*, vol. 82, pp. 121–133, Aug. 2016.

[17] E. Y. Xie, C. Li, S. Yu, and J. Lü, "On the cryptanalysis of Fridrich's chaotic image encryption scheme," *Signal Process.*, vol. 132, pp. 150–154, Mar. 2017.

[18] L. Y. Zhang, Y. Liu, F. Pareschi, Y. Zhang, K.-W. Wong, R. Rovatti, and G. Setti, "On the security of a class of diffusion mechanisms for image encryption," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1163–1175, Apr. 2018.

[19] J.-M. Guo, D. Riyono, and H. Prasetyo, "Improved beta chaotic image encryption for multiple secret sharing," *IEEE Access*, vol. 6, pp. 46297–46321, 2018.

[20] C. Li, Y. Liu, T. Xie, and M. Z. Q. Chen, "Breaking a novel image encryption scheme based on improved hyperchaotic sequences," *Nonlinear Dyn.*, vol. 73, no. 3, pp. 2083–2089, 2013.

[21] N. H. UbaidurRahman, C. Balamurugan, and R. Mariappan, "A novel DNA computing based encryption and decryption algorithm," *Procedia Comput. Sci.*, vol. 46, pp. 463–475, Dec. 2015. doi: 10.1016/j.procs.2015.02.045.

[22] B. Norouzi and S. Mirzakuchaki, "An image encryption algorithm based on DNA sequence operations and cellular neural network," *Multimedia Tools Appl.*, vol. 76, no. 11, pp. 13681–13701, 2017.

[23] X. Wang and C. Liu, "A novel and effective image encryption algorithm based on chaos and DNA encoding," *Multimed. Tools Appl.*, vol. 76, no. 5, pp. 6229–6245, 2017.

[24] X. Chai, Z. Gan, K. Yuan, Y. Chen, and X. Liu, "A novel image encryption scheme based on DNA sequence operations and chaotic systems," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 219–237, 2017.

[25] Y. Zhang, "The image encryption algorithm based on chaos and DNA computing," *Multimedia Tools Appl.*, vol. 77, no. 16, pp. 21589–21615, 2018.

[26] R. Guesmi, M. A. B. Farah, A. Kachouri, and M. Samet, "A novel chaos-based image encryption using DNA sequence operation and Secure Hash Algorithm SHA-2," *Nonlinear Dyn.*, vol. 83, no. 3, pp. 1123–1136, 2016.

[27] H. Liu, X. Wang, and A. Kadir, "Image encryption using DNA complementary rule and chaotic maps," *Appl. Soft Comput.*, vol. 12, no. 5, pp. 1457–1466, 2012. doi: 10.1016/j.asoc.2012.01.016.

[28] J. Jin, "An image encryption based on elementary cellular automata," *Opt. Lasers Eng.*, vol. 50, no. 12, pp. 1836–1843, 2012.

[29] R.-J. Chen and J.-L. Lai, "Image security system using recursive cellular automata substitution," *Pattern Recognit.*, vol. 40, no. 5, pp. 1621–1631, May 2007.

[30] X. Zhang, C. Wang, S. Zhong, and Q. Yao, "Image encryption scheme based on balanced two-dimensional cellular automata," *Math. Problems Eng.*, vol. 2013, Sep. 2013, Art. no. 562768.

[31] M. Li, D. Lu, W. Wen, H. Ren, and Y. Zhang, "Cryptanalyzing a color image encryption scheme based on hybrid hyper-chaotic system and cellular automata," *IEEE Access*, vol. 6, pp. 47102–47111, 2018.

[32] Y. Zhou, L. Bao, and C. P. Chen, "A new 1D chaotic system for image encryption," *Signal Process.*, vol. 97, no. 11, pp. 172–182, 2014.

[33] A. Rehman, X. Liao, A. Kulsoom, and S. A. Abbas, "Selective encryption for gray images based on chaos and DNA complementary rules," *Multimedia Tools Appl.*, vol. 74, no. 13, pp. 4655–4677, 2015.

[34] S. Lian, J. Sun, and Z. Wang, "Security analysis of a chaos-based image encryption algorithm," *Phys. A, Statist. Mech. Appl.*, vol. 351, nos. 2–4, pp. 645–661, 2005.

[35] J. Peng, S. Jin, Y. Liu, Z. Yang, M. You, and Y. Pei, "A novel scheme for image encryption based on piecewise linear chaotic map," in *Proc. IEEE Conf. Cybern. Intell. Syst. (ICCIS)*, Sep. 2008, pp. 1012–1016.

[36] X. Chai, K. Yang, and Z. Gan, "A new chaos-based image encryption algorithm with dynamic key selection mechanisms," *Multimedia Tools Appl.*, vol. 76, no. 7, pp. 9907–9927, 2017.

[37] C. Zhu, S. Xu, Y. Hu, and K. Sun, "Breaking a novel image encryption scheme based on Brownian motion and PWLCM chaotic system," *Nonlinear Dyn.*, vol. 79, no. 2, pp. 1511–1518, 2014.

[38] X. Chai, Z. Gan, K. Yang, Y. Chen, and X. Liu, "An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and DNA sequence operations," *Signal Process., Image Commun.*, vol. 52, pp. 6–19, Mar. 2017.

[39] C. E. Shannon, "Communication theory of secrecy systems," *MD Comput.*, vol. 15, no. 1, pp. 57–64, 1998.

[40] O. Mirzaei, M. Yaghoobi, and H. Irani, "A new image encryption method: Parallel sub-image encryption with hyper chaos," *Nonlinear Dyn.*, vol. 67, no. 1, pp. 557–566, 2012.

[41] X. Chai, "An image encryption algorithm based on bit level Brownian motion and new chaotic systems," *Multimedia Tools Appl.*, vol. 76, no. 1, pp. 1159–1175, 2017.

[42] C. Li, D. Lin, B. Feng, J. Lü, and F. Hao, "Cryptanalysis of a chaotic image encryption algorithm based on information entropy," *IEEE Access*, vol. 6, pp. 75834–75842, 2018.

[43] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Inf. Sci.*, vol. 222, pp. 323–342, Feb. 2013.

[44] C.-Y. Chen, C.-H. Chen, C.-H. Chen, and K.-P. Lin, "An automatic filtering convergence method for iterative impulse noise filters based on PSNR checking and filtered pixels detection," *Expert Syst. Appl.*, vol. 63, pp. 198–207, Nov. 2016.

[45] Y. Wang, K.-W. Wong, X. Liao, and G. Chen, "A new chaos-based fast image encryption algorithm," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 514–522, 2011.

[46] S. Xu, Y. Wang, J. Wang, and Y. Guo, "A fast image encryption scheme based on a nonlinear chaotic map," in *Proc. 2nd Int. Conf. Signal Process. Syst. (ICSPS)*, vol. 2, Jul. 2010, pp. V2-326–V2-330.

[47] S. Deng, Y. Zhan, D. Xiao, and Y. Li, "Analysis and improvement of a hash-based image encryption algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 16, no. 8, pp. 3269–3278, 2011.

SAJID KHAN received the B.Sc. degree (Hons.) in telecommunication from the University of Engineering and Technology, Pakistan, in 2012, and the master's degree in computer science from Preston University, Pakistan, in 2015. He is currently pursuing the Ph.D. degree in computer science and technology with the Information Security Lab, Huazhong University of Science and Technology, China. His research interests include network security, image encryption, and cryptography.

LANSHENG HAN received the bachelor's degree in mathematics from Lanzhou University, in 1995, and the master's and Ph.D. degrees in information security from the Huazhong University of Science and Technology (HUST), China, in 2001 and 2006, respectively. He is currently a Professor of information security with HUST. His main research interests include security of networks, malicious codes, and security in big data.

HONGWEI LU received the D.Eng. degree. He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST). He has presided over five cooperation projects along with a research institute. He has more than ten other horizontal projects. His research interests include information security and the Internet of Things.

KHUSHBU KHALID BUTT received the master's degree in computer science from the Huazhong University of Science and Technology, China, where she is currently pursuing the Ph.D. degree in computer science. Her research interests include cloud computing, machine learning, and big data security.

GUEHGUIH BACHIRA received the B.S. and master's degrees in computer science from Mohamed Seddik Ben Yahia University, Algeria, in 2014 and 2016, respectively. She is currently pursuing the Ph.D. degree in computer science with the Huazhong University of Science and Technology. Her research interests include information security, big data, and artificial intelligence.

NAIMAT-ULLAH KHAN received the B.S. degree in computer science from the Kohat University of Science and Technology and the M.S. degree in computer science from Preston University, Islamabad, in 2014. He is currently pursuing the Ph.D. degree in communication and information systems with Shanghai University, China. His research interests include artificial intelligence, machine learning, and big data analysis.