Journal of Electrical Engineering & Technology (2020) 15:1413–1429
https://doi.org/10.1007/s42835-020-00393-x

ORIGINAL ARTICLE

# Colored Image Encryption and Decryption with a New Algorithm and a Hyperchaotic Electrical Circuit

Batuhan Arpacı[1] · Erol Kurt[2] · Kayhan Çelik[2] · Bünyamin Ciylan[3]

Received: 6 March 2019 / Revised: 5 October 2019 / Accepted: 24 February 2020 / Published online: 6 March 2020

## Abstract

In the present work, an encryption/decryption technique, using a new bit-level scrambling and a new diffusion algorithm is presented. The proposed system uses a modified Chua's circuit (MCC) for the chaotic number generation for the first time to our knowledge. In 2006, the MCC, which exhibited a hyper-chaotic behavior for a wide parameter regime due to its double frequency excitation feature was suggested by one of the authors of the present paper. However, it has not been used for secure communication issues. According to present technique, the generated data are transformed to the developed algorithm for the encryption and decryption purposes. Following the encryption procedure, the encrypted colored images are evaluated by a variety of tests including the analyses of secret key size, secret key sensitivity, histogram, correlation, differential attack, information entropy, and noise attack. The results prove that the suggested colored image encryption/decryption technique is satisfactory for the secure communication issues in terms of efficiency and speed.

**Keywords** Bit level scrambling · Chaotic sequence · Color image · Decryption · Encryption · Modified Chua's circuit

## 1 Introduction

In the present world, the information technologies rapidly grow. That reality enforces one to apply new methodologies on the image security in many fields from companies to the public services [1, 2]. Today, secure communication becomes very important issue for industrial production departments, defense industry and private usage [3, 4]. Especially, the image ciphering concept has become a vital task to prevent the information theft for important industrial projects and military applications.

For any secret communication issue, the techniques of the cryptography have taken attention of the community. However, traditional encryption methods such as DES, AES, and IDEA have some security flaws, because there are many tools to decrypt the images, which have been ciphered with conventional techniques [4–6]. Among them, some tools can be mentioned as correlation, histogram and bulky data [7, 8]. In order to avoid this insecure situation, improving innovative encryption techniques is a vital task [5, 6].

Among the secure communication issues, the encryption of color images stays in a special stage. In principle, there exist two main processes [9, 10] (i.e. permutation and diffusion). These processes can be used for image encryption, but the implementation of only one of these stages at a bit or pixel level will not provide required security. Thus, the encryption procedures should be improved further than any decryption techniques to avoid from the insecure image communication. For instance, applying only the exchange procedure in the bit level can give satisfactory results in both permutation and diffusion stages [6, 11]. According to the literature, [11, 12] those characteristics meet the basic requirements of any kind of image encryption system. Many scientists and researchers used chaos-based encryption systems to design and implement novel image encryption schemes [11–15]. Indeed, the random numbers received from any chaotic system have a great advantage for the encryption procedure. Therefore, it is not astonishing that many chaos-based random number generators exist in literature. The main characteristics for a chaos-based system

✉ Erol Kurt
ekurt52tr@yahoo.com

[1] Information Systems Department, Informatics Institute, Gazi University, Ankara, Turkey

[2] Department of Electrical and Electronics Engineering, Technology Faculty, Gazi University, Besevler, 06500 Ankara, Turkey

[3] Department of Computer Engineering, Technology Faculty, Gazi University, Besevler, 06500 Ankara, Turkey

is that the output data (i.e. functions) never repeat, thereby any external source cannot have the information to decrypt the random data. Strictly speaking, the chaotic circuits transmit the related data to encrypt the image to only a well-synchronized slave system. A slave circuit system can only decrypt the image for the desired aim [16].

The progress of the technology has enabled the transmission of large data over the network. Presently, multimedia data have become an important element for the use of the network communication. Especially, the spread of color image transmission has revealed security requirements [17–19], but the encryption algorithms designed for gray images generally remain bulky in the color images and also traditional encryption algorithms are poor for color images. In addition to that, in some algorithms developed for the color image encryption, RGB components of the image are encrypted independently of each fact which affects the system negatively in terms of [20, 21]. Color image encryption is usually realized at pixel level [22, 23]. In recent years, there are many bit level color image cipher schemes in the literature [24–26]. It is clear for many systems that only the permutation operation at a bit level gives quite satisfactory results for a ciphering process [6, 11, 27], whereas, since the data size in the color image is high, the design algorithm for a bit level encryption should be as optimized as possible so that it does not give any bad results in terms of speed.

In the present work, a new chaos-based algorithm is proposed for ciphering the color images. The novel feature of the paper comes from two parts, namely the algorithm itself and the modified Chua's circuit (MCC) in the processes of ciphering and deciphering. The proposed algorithm combines diffusion and permutation features for a bit level color image encryption. It has also been proven that the suggested system is resistant to any plain text attacks, since the key is built using the SHA-256 [28, 29] algorithm and plain image. The new system also reduces the correlation because of the mixture of three-color image components.

This paper is organized as follows: in Sect. 2, some related studies on the secure communication literature have been stated. In Sect. 3, the MCC system is described with the relevant system parameters. Some samples on the hyperchaotic results and Lyapunov exponents are also presented in this section. In Sect. 4, the proposed secure communication algorithm is discussed. The experimental findings are given in Sect. 5. The security tests and performance analyses are reported in Sect. 6. Consequently, the paper is closed with a brief conclusion section.

## 2 Related Work

In any chaotic image encryption system, there are two main issues: First is the chaotic system, which is used as a random number generator. Second is an encryption algorithm, which has an importance in terms of efficiency and speed [7]. In the literature, many chaotic systems have been used for image encryption [30–33]. The dynamic properties of these chaotic systems affect the quality of the encryption process [34]. Conditions such as the width of the parameter space, the strength of randomness, simple and usefulness play effective roles in the selection of the chaotic generator [35]. On the one hand, the optimal adjustment of color image encryption algorithm is very important in terms of efficiency and speed [36].

In this work, an electronic circuit, namely MCC having a hyperchaotic character is used for color image encryption for the first time to our knowledge. In addition, bit-level scrambling and a new diffusion algorithm are examined as the second innovative part. The safety and performance tests prove that the system works effectively and fast enough to apply the encryption scheme.

## 3 Modified Chua's Circuit and Its Hyperchaotic Feature

In this section, the formulation of the proposed modified Chua Circuit (MCC) and its hyperchaotic behavior are presented. The MCC is used as a random number generator for the encryption process in the present work. Therefore it is vital to encrypt any kind of image with a high efficiency. Initially the state equations of the MCC are shown as follows [37]:

$$\begin{cases} \dot{x} = y - bx - \frac{1}{2}(a-b)[\,|x + \sin(z)| - |x - \sin(z)|\,], \\ \dot{y} = -\beta(y + x) + f\sin(v), \\ \dot{z} = \phi, \\ \dot{v} = \omega \end{cases} \tag{1}$$

In Eq. (1), $a$, b, $\phi, \beta, \omega, f$ are the control parameters, which define the system dynamics and enable the system to produce different output data for the encryption.

The phase spaces from Eq. (1) are shown in Figs. 1 and 2 after applying the time integration by using the Runge–Kutta method in MatLab. These phase spaces prove that the MCC system can produce strange attractors with chaotic (see in

Fig. 1a–d) and hyperchaotic (see in Fig. 2a–d) ones in 2D and 3D representations.

Lyapunov exponents are very important for the identification of the characteristics of a dynamic system. Indeed, one can be assure on the chaotic behavior of any dynamics system after applying the Lyapunov calculation scheme [37, 38]. The formulation of the Lyapunov exponent is given in Ref. [37]. For instance, there must be one positive exponent for a chaotic system [39]. If there would be two positive exponents, the system exhibits a hyperchaotic character, which means that the system diverges in two dimensions. The Lyapunov exponents of Fig. 1a–d are shown in Fig. 3a, b. The signs of the plots are (0, 0, −, +), which yields a chaotic nature. However, the exponents in Fig. 3c, d give

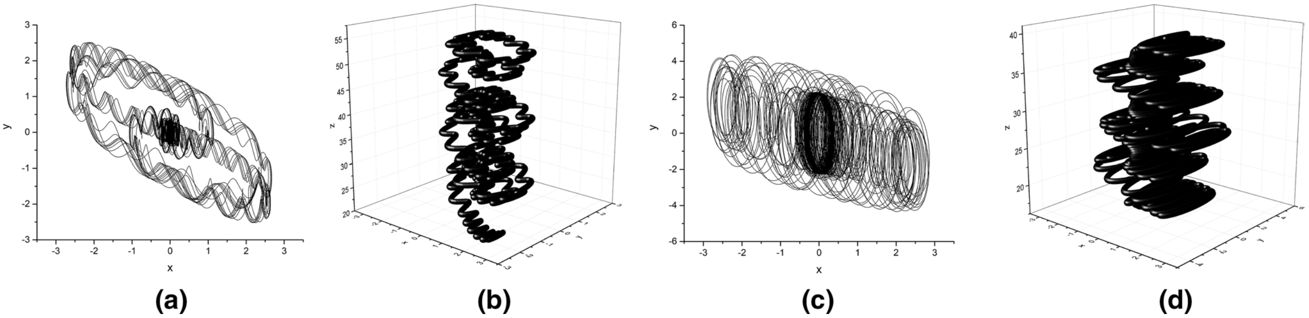

**Fig. 1** 2-D and 3-D representations of sample chaotic attractors with parameters: **a**, **b** $a = -1.17$, $b = -0.49$, $\beta = 0.55$, $f = 1.99$, $\phi = 0.2$ and $\omega = 6.4$, **c**, **d** $a = -1.17$, $b = -0.49$, $\beta = 0.55$, $f = 13.92$, $\phi = 0.2$ and $\omega = 6.4$, respectively

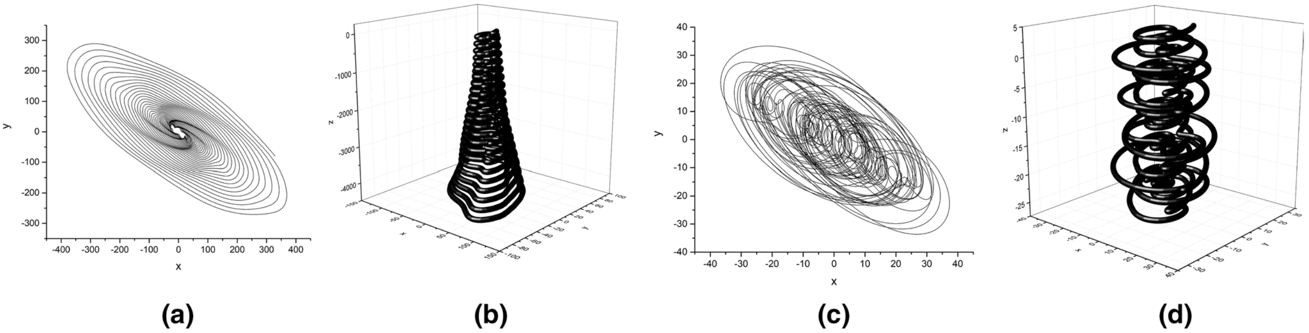

**Fig. 2** 2-D and 3-D representations of sample hyperchaotic attractors with parameters: **a**, **b** $a = -2.91$, $b = -0.56$, $\beta = 0.55$, $f = 12.99$, $\phi = -15.1$ and $\omega = 2.91$, **c**, **d** $a = -2.91$, $b = -0.56$, $\beta = 0.55$, $f = 9.01$, $\phi = -0.13$ and $\omega = 1.29$, respectively

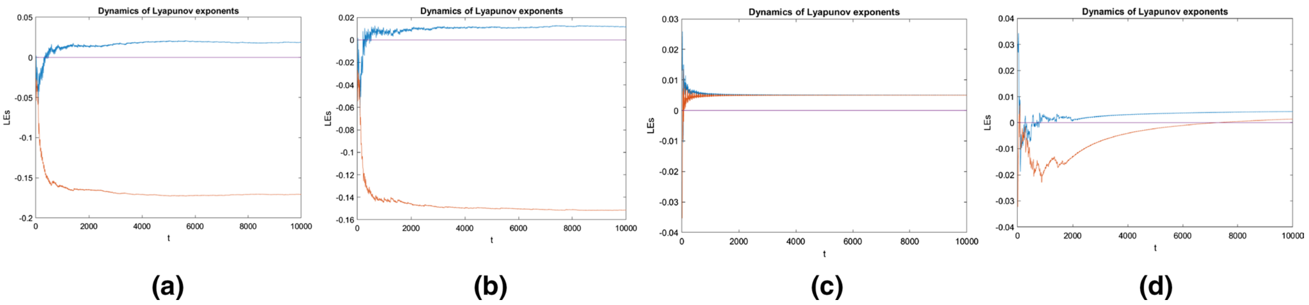

**Fig. 3** Lyapunov exponents with the parameters, **a** $a = -1.17$, $b = -0.49$, $\beta = 0.55$, $f = 1.99$, $\phi = 0.2$ and $\omega = 6.4$, **b** $a = -1.17$, $b = -0.49$, $\beta = 0.55$, $f = 13.92$, $\phi = 0.2$ and $\omega = 6.4$, **c** $a = -2.91$, $b = -0.56$, $\beta = 0.55$, $f = 12.99$, $\phi = -15.1$ and $\omega = 2.91$, and **d** $a = -2.91$, $b = -0.56$, $\beta = 0.55$, $f = 9.01$, $\phi = -0.13$ and $\omega = 1.29$, respectively

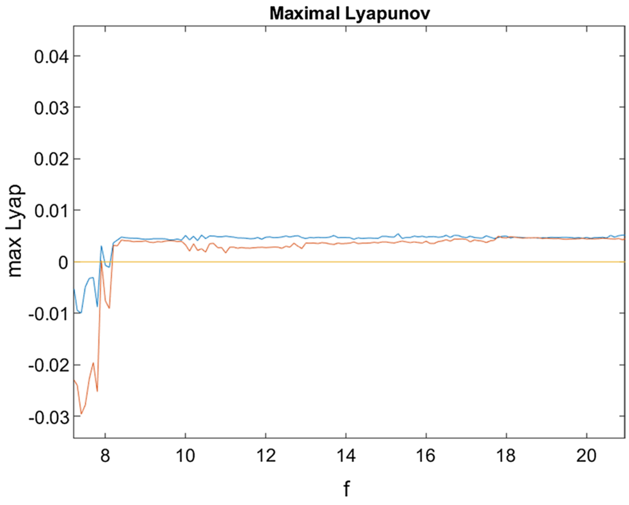

**Fig. 4** The variation of maximal Lyapunov exponents with respect to parameter $f$. Other parameters are $a = -2.91$, $b = -0.56$, $\beta = 0.55$, $\phi = -0.13$, and $\omega = 1.29$, respectively

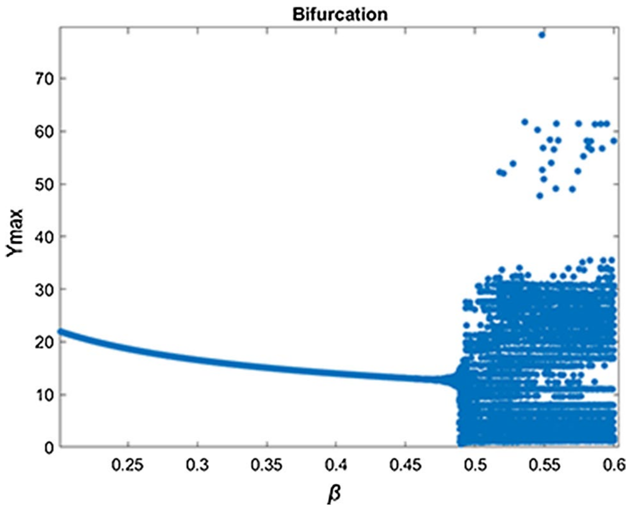

**Fig. 5** The bifurcation diagram for parameter $\beta$. The other parameters are $a = -1.17$, $b = -0.49$, $f = 13.92$, $\phi = 0.2$ and $f = 4.1$, respectively

two positive exponents for Fig. 2a–d with the signs of (0, 0, +, +), which denotes a hyperchaotic behavior form the relevant parameter set.

In order to show the dependence on the input parameter $f$, the maximal Lyapunov exponents are depicted in Fig. 4. It is obvious that the system reaches to a hyperchaotic regime beyond $f = 9$. By using another parameter of the system i.e. $\beta$, a bifurcation diagram has been produced in Fig. 5. It is obvious that the chaotic nature appears for larger $\beta$ values (i.e. 0.49). A periodic regime exists for low $\beta$ values.

## 4 Chaos Based Image Encryption Scheme

### 4.1 Secret Key Generation

SHA-256, which is a traditionally used cryptographic hash algorithm, produces a 256-bit hash value. And this value changes completely when there is a slight change in the input of the algorithm. The function is used to generate the keys of this cryptosystem. Indeed, first of all, a 48-bit digest output which is described as *PK* is obtained from the plain image for input to the SHA-256 function. On the other hand, random noise *RN* is generated at the beginning of each encryption process using *randi* function of the MatLab. Subsequently, a 256-bit digest hash value *SK* is generated by executing SHA-256 with the *PK* and *RN* input. Thus, the secret key produced is completely unique thanks to SHA-256, even if there is a slight change in the plain image, or even no changes at all. As a result, all of this indicates that our encryption system can be resistant to against chosen-plaintext, chosen-ciphertext and known-plaintext attacks. The secret key generation is explained in detail below by pseudocodes written in MatLab.

ImgA plain color image is presented as input to the following algorithms:

### Row Collection Algorithm

```
$\textit{function}\quad \text{rowCollection} = \text{rowCollect(ImgA)}$
$[\text{r, n, k}] = \textit{size}(\text{ImgA})$
$\textit{for } i = \ : r\ 1$
    $\textit{if } \ i == 1$
        $\text{rowCollection(i)} = \text{mod}(\textit{sum}(\text{ImgA}(i,:)),256);$
    $\textit{else}$
        $\text{rowCollection(i)} = \textit{bitxor}(\text{rowCollection}(i-1), \text{mod}(\textit{sum}(\text{ImgA}(i,:)),256));$
    $\textit{end}$
$\textit{end}$
$\textit{end}$
```

### Column Collection Algorithm

```
$\textit{function}\quad \text{columnColl ection} = \text{columnColl ect(ImgA)}$
$[\text{r, n, k}] = \textit{size}(\text{ImgA})$
$\textit{for } i = \ : r\ 1$
    $\textit{if } \ i == 1$
        $\text{columnColl ection (i)} = \text{mod}(\textit{sum}(\text{ImgA}(:,i)),256);$
    $\textit{else}$
        $\text{columnColl ection (i)} = \textit{bitxor}(\text{columnColl ection}(i-1), \text{mod}(\textit{sum}(\text{ImgA}(:,i)),256));$
    $\textit{end}$
$\textit{end}$
$\textit{end}$
```

### Color Components Collection Algorithm

```
$\textit{function}\quad [\text{rgCollection, rbCollection, bgCollection, rgbCollection}] = \text{rgbCollect(ImgA)}$
$\text{ImgAR} = \text{ImgA}(:,:,1)$
$\text{ImgAG} = \text{ImgA}(:,:,2)$
$\text{ImgAB} = \text{ImgA}(:,:,3)$
$\text{rgCollection} = \textit{bitxor}(\text{mod}(\textit{sum}(\text{ImgAR}(:)),256), \text{mod}(\textit{sum}(\text{ImgA}G(:)),256))$
$\text{rbCollection} = \textit{bitxor}(\text{mod}(\textit{sum}(\text{ImgAR}(:)),256), \text{mod}(\textit{sum}(\text{ImgAB}(:)),256))$
$\text{bgCollection} = \textit{bitxor}(\text{mod}(\textit{sum}(\text{ImgAB}(:)),256), \text{mod}(\textit{sum}(\text{ImgAG}(:)),256))$
$\text{rgbCollection} = \textit{bitxor}\begin{pmatrix} \textit{bitxor}(\text{mod}(\textit{sum}(\text{ImgAR}(:)),256), \text{mod}(\textit{sum}(\text{ImgAG}(:)),256)), \\ \text{mod}(\textit{sum}(\text{ImgAB}(:)),256) \end{pmatrix}$
$\textit{end}$
```

**PK Pre-Key Generate Algorithm**

```
$function \quad pkGeneration = pkGenerate(\mathrm{ImgA})$
$[r\ \ n,k],= size(\mathrm{ImgA})$
$rowCollectionBin = de2bi(rowCollection(r)\,8)$ ,
$columnCollectionBin = de2bi(columnCollection(n)\,8)$ ,
$rgCollectionBin = de2bi(rgCollection\ 8)$ ,
$rbCollectionBin = de2bi(rbCollection\ 8)$ ,
$bgCollectionBin = de2bi(bgCollection\ 8)$ ,
$rgbCollectionBin = de2bi(rgbCollection\ 8)$ ,
$prekeyBin = \begin{bmatrix} rowCollectionBin, columnCollectionBin, rgCollectionBin, rbCollectionBin, \\ bgCollectionBin, rgbCollectionBin \end{bmatrix}$
$PK = binaryVectorToHex(prekeyBin)$
$end$
```

**One-time Secrey Key Generate Algorithm**

```
$function \quad \mathrm{SK} = oneTimeSecretKeyGenerate$
$randomBin = randi([0\,1],16,1)'$
$RN = binaryVectorToHex(randomBin)$
$SK = sha256(strcat(PK, RN))$
$end$
```

Supposing that a hexadecimal number as $h_i$, secret key SK can be defined as the hexadecimal number array as follows:

$$SK = \left[h_1, h_2, \ldots, h_{64}\right],\ \exists i \in [1, 2, \ldots, 64] \quad \forall h_i \in [0-9], [A-F] \tag{2}$$

## 4.2 Obtaining the Initial Values of the Chaotic Equation from the Secret Key

The fact that chaotic systems are sensitive to initial values makes these systems very important for encryption systems. In this study, we have tried to reflect the slightest change in the secret key to the initial values. So, we get the values $a_1, a_2, b_1$ and $b_2$ to make sure that the slightest change in secret key causes changes in all initial values.

The initial values $x_1, y_1, z_1, v_1$ and the initial parameter $f$ for Eq. (1) can be derived as follows:

$$\begin{cases} x_1' = \left(hex2de(subset(1,\ 10, SK))10^{-11}\right) \\ \quad + \left(hex2de(subset(11,\ 16, SK))10^{-14}\right) \\ y_1' = \left(hex2de(subset(17,\ 26, SK))10^{-11}\right) \\ \quad + \left(hex2de(subset(27,\ 32, SK))10^{-14}\right) \\ z_1' = \left(hex2de(subset(33,\ 42, SK))10^{-11}\right) \\ \quad + \left(hex2de(subset(43,\ 48, SK))10^{-14}\right) \\ v_1' = \left(hex2de(subset(49,\ 58, SK))10^{-11}\right) \\ \quad + \left(hex2de(subset(59,\ 64, SK))10^{-14}\right) \end{cases} \tag{3}$$

Here $hex2de(.)$ function converts the secret key from hexadecimal number to a decimal number, $subset(i, j, K)$ returns elements between the $i$th index and $j$th index of the $K$ 1-D array.

In Eq. (3), we determine the multiplications $10^{11}$ and $10^{14}$ in order to adjust the relevant decimals of the $x_1, y_1, z_1$ and $v_1$.

The values of $a_1, a_2, b_1$ and $b_2$ are calculated according to the program schemes in Figs. 6 and 7 using *subset*, *concat*, *roundD*, *hex2de* and *sum* functions. The *subset* and *hex2de* functions are mentioned above. The *concat*(./.) function concatenate the values given into it. The *roundD*(.) returns the decimal portion of the given decimal number and *sum*(.) is aggregate function.

$$\begin{cases} x_1 = x_1'(2 - a_1) \\ y_1 = y_1'(2 - a_2) \\ z_1 = z_1'(2 - b_1) \\ v_1 = v_1'(2 - b_2) \\ f = 9.1 + a_1 \end{cases} \tag{4}$$

where the number 9.1 refers to the lower chaotic parameter for *f*. Since $a_{1,2}$ and $b_{1,2}$ refers to the numbers lower than 1, we make the multiplication higher by using the term $(2 - (a, b)_{1,2})$.

### 4.3 Encryption Algorithm

Figure 8 represents the overall flow chart of the encryption procedure. The steps of the flow chart are explained as the following:

Input: Plain image *P*, secret key *SK*

Output: Cipher image *C*

If we assume that the horizontal and vertical magnitudes are *W* and *H* respectively, the size of the color plain image is $W \times H \times 3$. Then the total size of the colored image is as follows:

$$s = W \times H \times 3 \tag{5}$$

Step 1. Take the initial values $(x_1, y_1, z_1, v_1)$ and the initial parameter *f* of the chaotic system using Eqs. (3, 4).

Step 2. With the help of the iteration method, generate chaotic numbers array *CN* whose size are $(s \times 4) + 5000$ by solving this time-continuously chaotic system. And then, remove the first thousand chaotic values that could adversely affect the encryption system.

$$\begin{aligned} &n = s \times 4, \\ &cs = n + 4000; \end{aligned} \tag{6}$$

Step 3. With the help of these numbers *CN* produced by chaotic generator, generate the key matrix *KM* to be applied in the diffusion and scrambling stages.

$$\begin{aligned} &for\, i = 1 : cs \\ &CN(i) = abs\big((CN(i) - round(CN(i), 6) \times 10\text{^}6)\big); \\ &end \\ &CN' = unique(CN); \\ &CN'' = subset(1, s \times 4, CN'); \\ &KM = sort(CN'') \end{aligned} \tag{7}$$

Here, *CN*(*i*) means that the *i*th element of *CN,* which is a 1-D array. The *round* function rounds the entered decimal number to the nearest number. The second parameter of the function determines the decimal point of the number to be rounded. The *abs* function takes the absolute value of the entered number. '^' is the exponent operator that we know.

The *unique* function deletes repetitive elements of an array. The *sort* function returns the new index numbers of the array by sorting the array from small to large. The *subset*(*i*,*j*, *K*) returns elements between the *i*th index and *j*th index of the K. As a result, we can define the *KM* matrix as follows:

$$\begin{aligned} &\forall i, j \in [1, 2, \dots, n], \\ &KM = [km_1, km_2, \dots km_n], km_i \in [1, n], Z^+, \\ &\forall i \neq j \Rightarrow \forall km_i \neq km_j \end{aligned} \tag{8}$$

Step 4. Resize the plain image *P* for each pixel by starting from component R sequentially from upper point to bottom point, then left to right, with components G and B. After that, convert each pixel into a 8-digit binary format. As a result, a matrix of *s* rows and 8 columns is obtained. The *getBinimage* function applies all these operations to give the *PB* matrix.

$$PB = getBinimage(P) \tag{9}$$

The first column of the *PB* matrix corresponds to the first bit in the binary format of the decimal values corresponding to each row in this matrix. The same logic is used from 1th column to the 8th column.

Separate the first 4 columns and the last 4 columns of the binary matrix. Indeed, vertically divide the matrix *PB* in half. The matrix containing the first four columns of *PB* is $PB_1$, the other matrix is called $PB_2$.

Step 5. Perform the mapping method to the matrix $PB_2$ by using the *KM* key matrix.

$$\begin{aligned} &PB_2' = reshape(PB_2, n, 1) \\ &PB_2'' = PB_2'(KM) \\ &PB_2 = reshape(PB_2'', s, 4) \end{aligned} \tag{10}$$

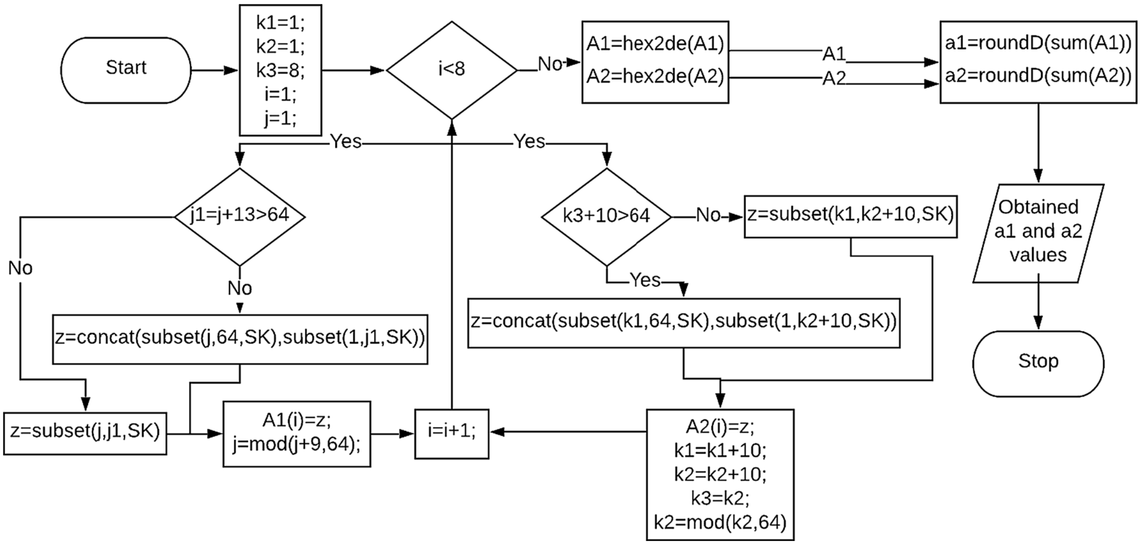

**Fig. 6** Flow diagram where $a_1, a_2 \in [0, 1]$ decimal values are obtained

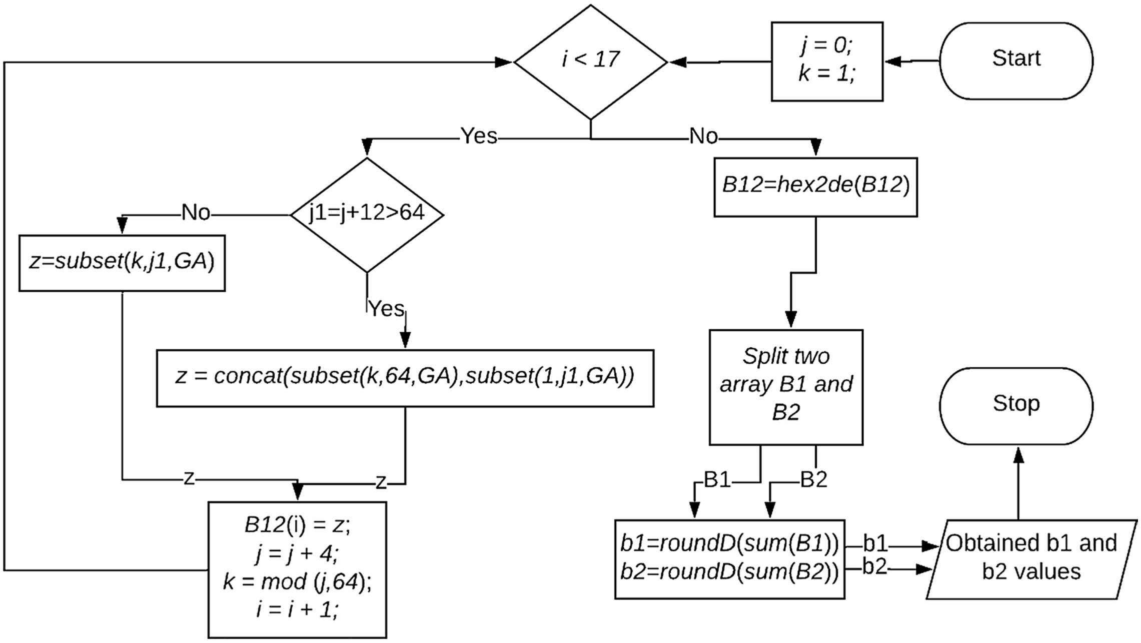

**Fig. 7** Flow diagram where $b_1, b_2 \in [0, 1]$ decimal values are obtained

Here, *reshape* resizes any matrix according to the values given. The $PB_2'(KM)$ operation scrambles the values in $PB_2'$ to different positions according to $KM$. Indeed, the operation can be explained as follows:

Let's assume $A$ is an array. In that case, when $I$ is a subset of the positive integers,$A(I)$ is a subset of the $A$. We define it as follows:

$$\begin{aligned}
&A = \left\{a_1, a_2, \dots, a_n\right\},\ I = \left\{i_1, i_2, \dots i_m\right\} :\\
&A(I) = \left\{a_{i_1}, a_{i_2}, \dots, a_{i_m}\right\} = \left\{A\left(i_1\right), A\left(i_2\right), \dots, A\left(i_m\right)\right\},\\
&i \in I \Rightarrow A(i) = a_i
\end{aligned} \tag{11}$$

Step 6. Perform diffusion method to matrices $PB_1$ and $PB_2$ by using the key matrix $KM$.

$$\begin{aligned}
&KM' = reshape(KM, s, 4);\\
&PD_1 = bi2de\left(PB_1\right);\\
&PD_2 = bi2de\left(PB_2\right);\\
&for\, i = 1 : s\\
&sm1 = \bmod\left(PD_1(i), 4\right);\\
&if\ sm1 == 0\ sm1 = 4; end\\
&PD_2(i,:) = bitxor\left(\begin{matrix} PD_2(i), \\ \bmod\left(KM'(i, sm1), 15\right)\end{matrix}\right);\\
&sm2 = \bmod\left(PD_2(i), 4\right);\\
&if\ sm2 == 0\ sm2 = 4; end\\
&PD_1(i,:) = bitxor\left(\begin{matrix} PD_1(i), \\ \bmod\left(KM'(i, sm2), 15\right)\end{matrix}\right);\\
&end\\
&CB_1 = de2bi\left(PD_1\right);\\
&CB_2 = de2bi\left(PD_2\right);
\end{aligned} \tag{12}$$

Here, while the function *bitxor* applies bitwise *xor* logical operation, the *bi2de* function converts the number from the binary format to decimal one. The *de2bi* is the opposite *bi2de* and the *mod* function is the standard modulus operator. $KM'(i,j)$ denotes the element of $i$th row and $j$th column in the $KM'$ matrix.

Step 7. In contrast to the separation in step 4, combine the matrices $CB_1$ and $CB_2$. Convert binary matrix to decimal matrix. Finally, get the encrypted image by converting this matrix to the original dimensions of the image.

$$\begin{aligned}
&CD = bi2de(CB);\\
&C = reshape(CD, W, H, 3);
\end{aligned} \tag{13}$$

It will be shown that the algorithm above has certain superiorities on the other algorithms in the literature. Initially, the present algorithm gives good results for all security tests. It is not time-consuming and complicated. Indeed, it uses the image in 2 half parts, which are regarded as important and unimportant parts as in Fig. 8.

### 4.4 Decryption Algorithm

The cipher image $C$ is the input data for this process and the deciphered $P$ is denoted as "output", as the inverse of the encryption process.

Step 1. Obtain the key matrix $KM$ by applying steps 1, 2 and 3 of the above encryption process exactly.

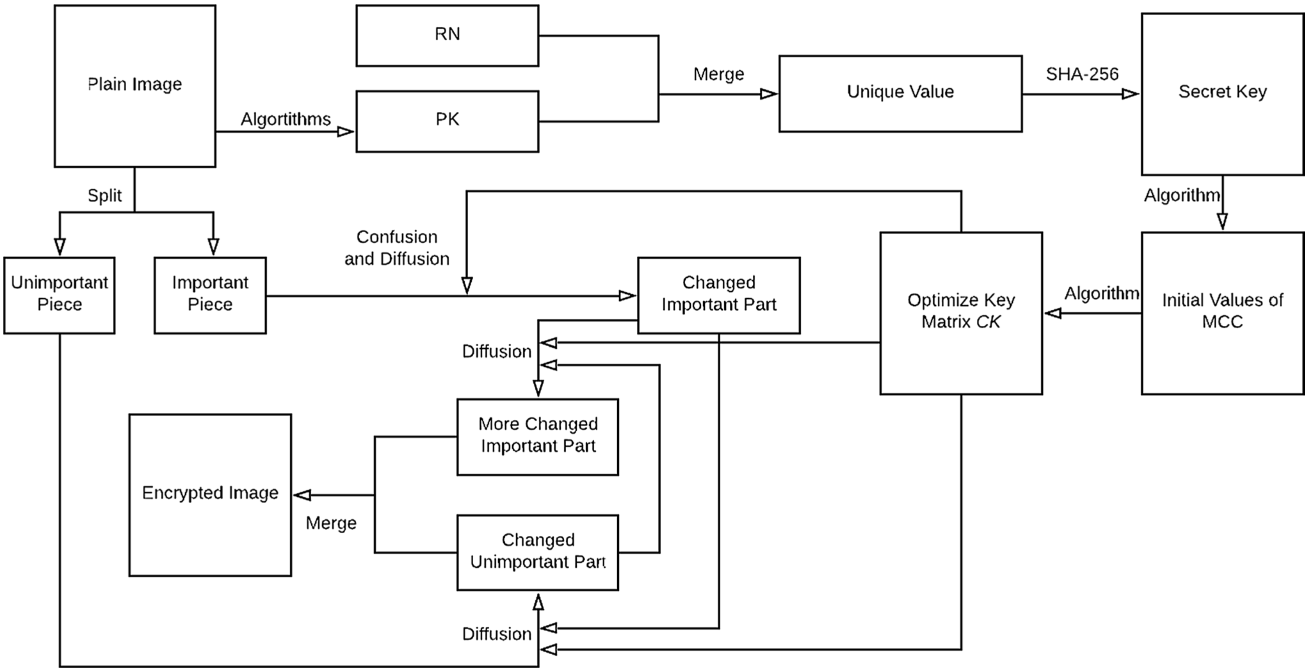

**Fig. 8** The flow chart of the encryption process

Step 2. Similar to step 5 in the encryption scheme, obtain the $CB_1$ and $CB_2$ matrices using the encrypted image instead of the plain image this time.

Step 3. Perform diffusion method to $CB_1$ and $CB_2$ matrices using $KM$ matrix.

$$\begin{aligned}
&KM' = reshape(KM, s, 4);\\
&CD_1 = bi2de\left(CB_1\right);\\
&CD_2 = bi2de\left(CB_2\right);\\
&for\, i = 1 : s\\
&sm2 = \mathrm{mod}\left(CD_2(i), 4\right);\\
&if\; sm2 == 0\; sm2 = 4; end\\
&CD_1(i, :) = bitxor\left(\begin{matrix} CD_1(i), \\ \mathrm{mod}\left(KM'(i, sm2), 15\right)\end{matrix}\right);\\
&sm1 = \mathrm{mod}\left(CD_1(i), 4\right);\\
&if\; sm1 == 0\; sm1 = 4; end\\
&CD_2(i, :) = bitxor\left(\begin{matrix} CD_2(i), \\ \mathrm{mod}\left(KM'(i, sm1), 15\right)\end{matrix}\right);\\
&end\\
&PB_1 = de2bi\left(CD_1\right);\\
&PB_2 = de2bi\left(CD_2\right);
\end{aligned} \tag{14}$$

The functions used here are defined in the encryption process in the previous section.

Step 4. Perform mapping method to the $CB_2$ matrix using the $KM$ key matrix.

$$\begin{aligned}
&PB_2' = reshape\left(PB_2, n, 1\right)\\
&PB_2''(CK) = PB_2'\\
&PB_2 = reshape\left(PB_2'', s, 4\right)
\end{aligned} \tag{15}$$

Step 5. Obtain the decoded $P$ matrix from the $PB_1$ and $PB_2$ matrices, similar to step 7 in the encryption algorithm.

## 5 Experimental Results

For the experiments, many parameter sets can be used. As a sample parameter set, we have considered the parameters of the chaotic circuit as $a = -2.91, b = -0.56, \beta = 0.55$ $, \phi = -0.13$ and $\omega = 1.29$. Because, the dynamic system exhibits a hyperchaotic behavior with these first parameters for $f \geq 9.1$. Along with plain image and random noise, the secret key produced with the help of the *SHA-256* function is 2A8649DDF54B044DC1A50329C54B4960010066BA8FD-005D4392B536545B04ECE. Then, the initial state variables and driving amplitude $f$ of MCC are obtained from this secret key.

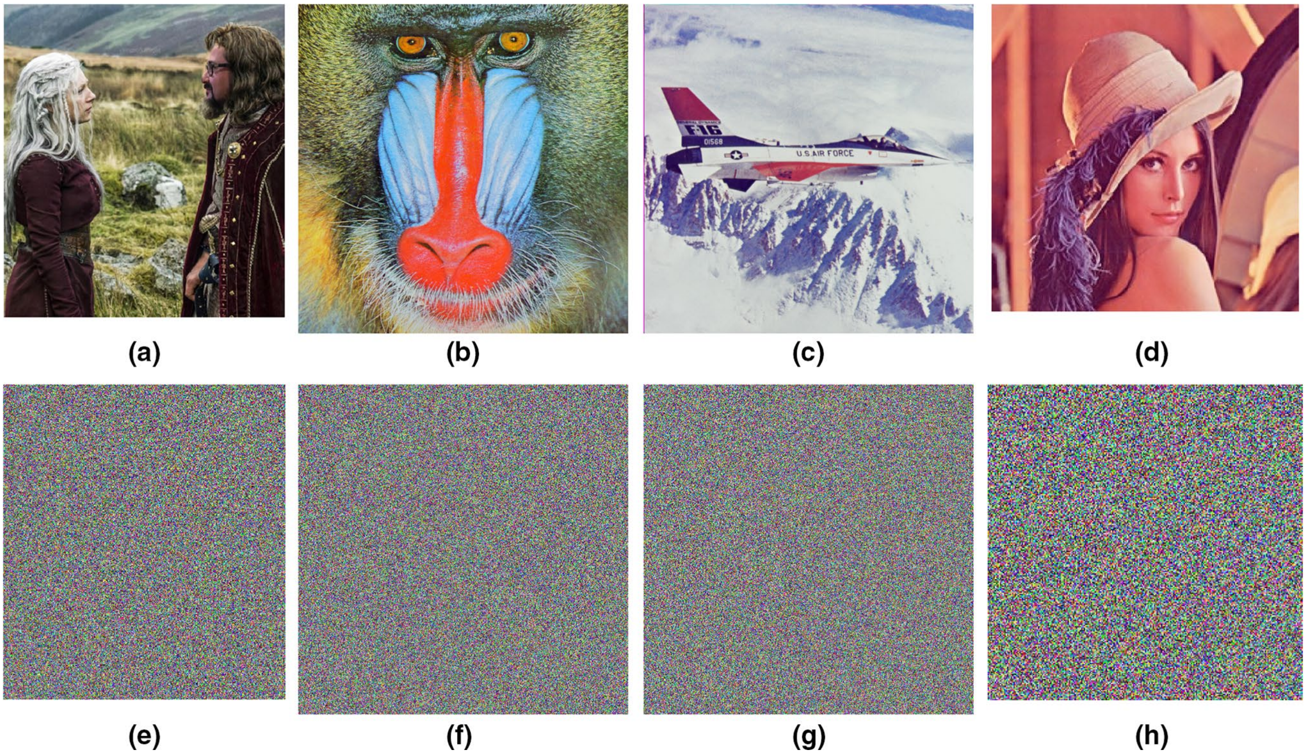

**Fig. 9** The plain images and their corresponding encoded results. **a** Vikings, **e** encrypted Vikings, **b** Baboon, **f** encrypted Baboon, **c** Airplane, **g** encrypted Airplane, **d** Lena and **h** encrypted Lena, respectively

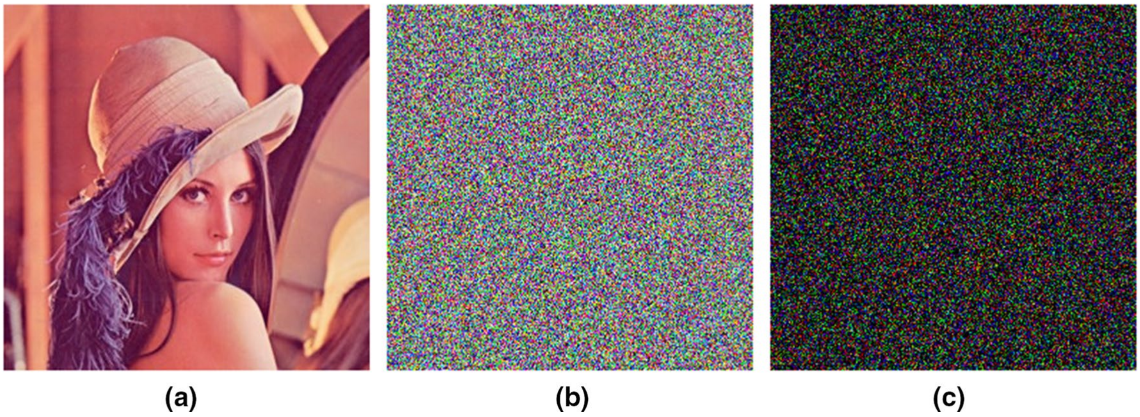

**Fig. 10** **a** One bit-modified version of Fig. 9d, **b** encrypted image of **a**, **c** the difference between Fig. 9b, h

**Table 1** Minimum, maximum and average UACI(%) values

| Image | *R* | | | *G* | | | *B* | | |
|---|---|---|---|---|---|---|---|---|---|
| | Max | Mean | Min | Max | Mean | Min | Max | Mean | Min |
| Vikings | 33.4156 | 33.3658 | 33.3479 | 33.3825 | 33.3482 | 33.3108 | 33.4182 | 33.3716 | 33.3556 |
| Baboon | 33.4684 | 33.4426 | 33.4281 | 33.5922 | 33.5371 | 33.4915 | 33.5163 | 33.4994 | 33.4635 |
| Airplane | 33.4428 | 33.4195 | 33.3841 | 33.4087 | 33.3856 | 33.3692 | 33.3867 | 33.3421 | 33.3242 |
| Lena | 33.5830 | 33.4926 | 33.4471 | 33.4826 | 33.4620 | 33.4483 | 33.5683 | 33.4961 | 33.4102 |
| Lena Ref. [45] | | 33.45 | | | 33.38 | | | 33.46 | |
| Lena Ref. [46] | | 33.48 | | | 33.46 | | | 33.42 | |
| Lena Ref. [47] | | 33.43 | | | 33.46 | | | 33.62 | |

**Table 2** Minimum, maximum and average NPCR(%) values

| Image | *R* | | | *G* | | | *B* | | |
|---|---|---|---|---|---|---|---|---|---|
| | Max | Mean | Min | Max | Mean | Min | Max | Mean | Min |
| Vikings | 99.6226 | 99.6118 | 99.6012 | 99.6095 | 99.6001 | 99.5944 | 99.6193 | 99.6167 | 99.5963 |
| Baboon | 99.6184 | 99.6021 | 99.5753 | 99.6028 | 99.5934 | 99.5812 | 99.6482 | 99.6216 | 99.6081 |
| Airplane | 99.6229 | 99.6148 | 99.5916 | 99.6156 | 99.5962 | 99.5894 | 99.6149 | 99.6032 | 99.5926 |
| Lena | 99.6218 | 99.6069 | 99.5945 | 99.6423 | 99.6102 | 99.5982 | 99.6193 | 99.5921 | 99.5736 |
| Lena Ref. [45] | | 99.59 | | | 99.59 | | | 99.60 | |
| Lena Ref. [46] | | 99.61 | | | 99.61 | | | 99.61 | |
| Lena Ref. [47] | | 99.57 | | | 99.58 | | | 99.57 | |

**Table 3** Information entropies of the cipher images

| Image | R | G | B |
|---|---|---|---|
| Vikings | 7.9978 | 7.9974 | 7.9971 |
| Baboon | 7.9997 | 7.9991 | 7.9992 |
| Airplane | 7.9985 | 7.9988 | 7.9987 |
| Lena | 7.9995 | 7.9988 | 7.9991 |
| Lena Ref. [45] | 7.9993 | 7.9993 | 7.9994 |
| Lena Ref. [47] | 7.9814 | 7.9810 | 7.9816 |

The sizes of plain images are 456 × 408, 512 × 512, 512 × 512, and 256 × 256 for the Vikings, Baboon, Airplane and Lena plain images, respectively (Fig. 9a–d). The encrypted versions of these images are given in Fig. 9e–h, respectively.

## 6 Security and Performance Analyses

### 6.1 Key Space Analysis

All the chaotic systems have a common feature: They are very dependent on the initial values. In other words, if any slight change occurs in the initial values of the functions, the functions produce entirely different result after sufficiently

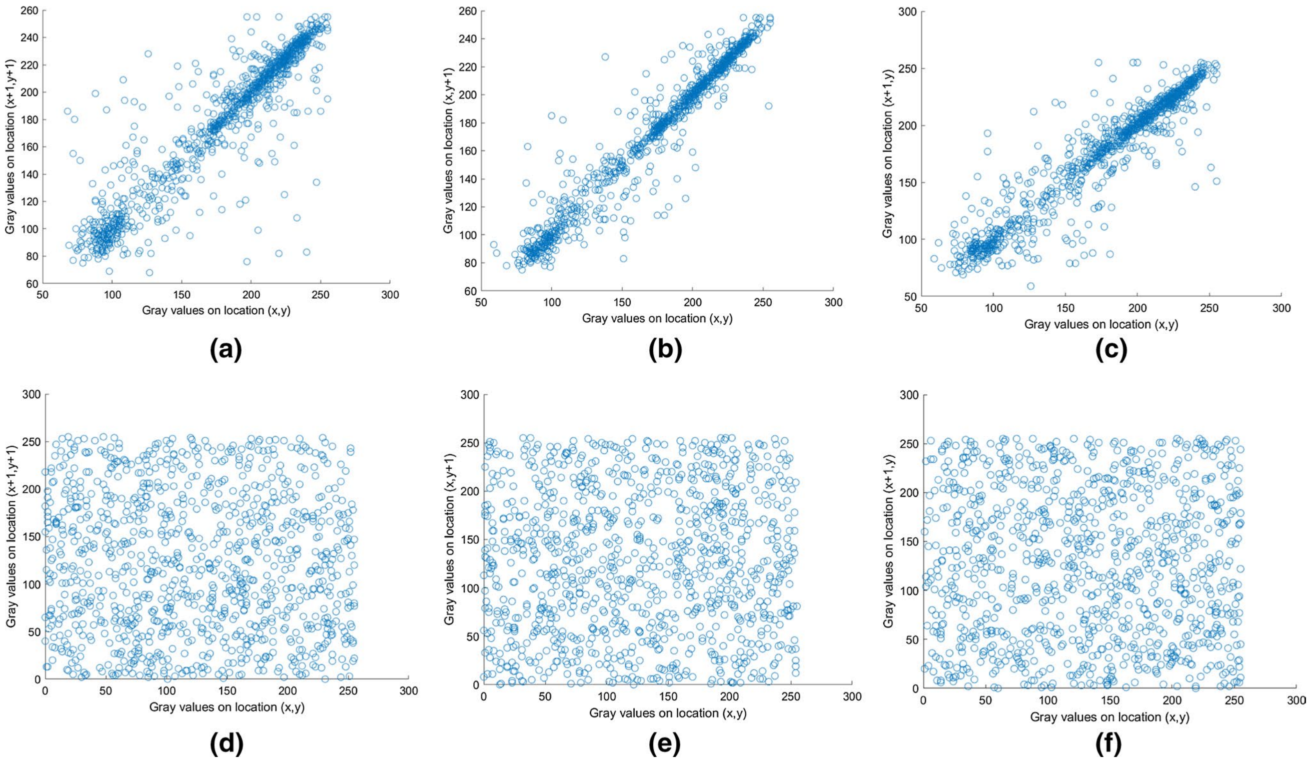

**Fig. 11** Distributions of the correlations between the plain and the encoded images. **a**, **c**, **e** are the diagonal, vertical and horizontal of plain image, **b**, **d**, **f** are the diagonal, vertical and horizontal of cipher image respectively

**Table 4** Correlation coefficients for adjacent pixels in the original images and their cipher images

| Images | Directions | Original | | | Encrypted | | |
|---|---|---|---|---|---|---|---|
| | | R | G | B | R | G | B |
| Vikings | Diagonal | 0.9409 | 0.9498 | 0.9569 | 0.0062 | − 0.0039 | − 0.0036 |
| | Vertical | 0.9777 | 0.9804 | 0.9668 | 0.0021 | − 0.0106 | 0.0036 |
| | Horizontal | 0.9647 | 0.9711 | 0.9677 | 0.0185 | 0.0140 | − 0.0165 |
| Baboon | Diagonal | 0.8527 | 0.7124 | 0.8415 | − 0.0120 | − 0.0106 | 0.0065 |
| | Vertical | 0.8683 | 0.7782 | 0.8790 | − 0.0113 | − 0.0175 | 0.0165 |
| | Horizontal | 0.9253 | 0.8626 | 0.9113 | − 0.0267 | − 0.0249 | 0.0006 |
| Airplane | Diagonal | 0.9112 | 0.9449 | 0.8952 | 0.0103 | 0.0043 | − 0.0007 |
| | Vertical | 0.9670 | 0.9612 | 0.9530 | − 0.0133 | − 0.0193 | 0.0004 |
| | Horizontal | 0.9762 | 0.9685 | 0.9689 | 0.0340 | 0.0114 | 0.0023 |
| Lena | Diagonal | 0.9303 | 0.9287 | 0.8760 | − 0.0125 | 0.0184 | 0.0096 |
| | Vertical | 0.9661 | 0.9673 | 0.9530 | 0.0118 | 0.0221 | 0.0030 |
| | Horizontal | 0.9461 | 0.9268 | 0.9147 | 0.0009 | − 0.0041 | − 0.0009 |
| Lena Ref. [1] | Diagonal | 0.9587 | 0.9412 | 0.8625 | − 0.0002 | − 0.0006 | − 0.0101 |
| | Vertical | 0.9728 | 0.9596 | 0.8797 | − 0.0002 | 0.0139 | 0.0011 |
| | Horizontal | 0.9819 | 0.9705 | 0.9203 | 0.0363 | − 0.0008 | − 0.0092 |
| Lena Ref. [2] | Diagonal | 0.9696 | 0.9555 | 0.9182 | 0.0009 | − 0.0014 | − 0.0019 |
| | Vertical | 0.9893 | 0.9824 | 0.9575 | − 0.0002 | 0.0018 | 0.0002 |
| | Horizontal | 0.9797 | 0.9690 | 0.9328 | − 0.0005 | − 0.0013 | − 0.0002 |

large time duration. The key space should be capable of neutralizing brute-force attacks for the encryption algorithm designs with a sufficient reliability. The encryption system key includes the initial values ($x_1$, $y_1$, $z_1$, $v_1$) and initial parameter of $f$. In general, for systems with chaotic features, the precision of the initial conditions should be

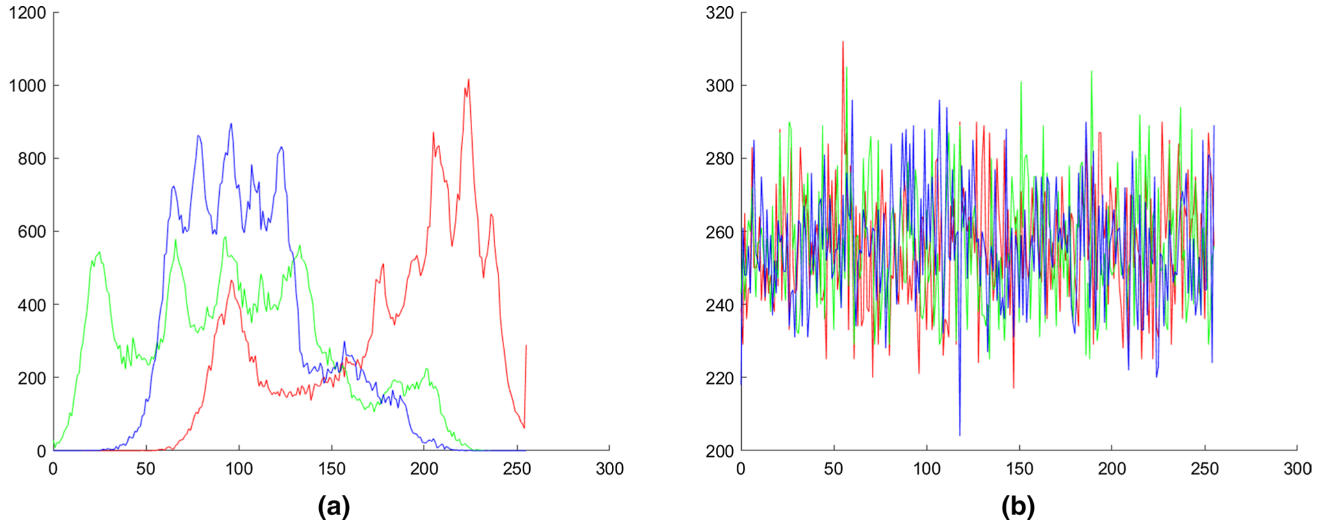

**Fig. 12** Histogram of plain and encrypted images of Lena respectively

**Table 5** Quantitative results of resisting noise attack

| Image | Density | MSE | | | PSNR | | | Correlation | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | R | G | B | R | G | B | R | G | B |
| Vikings | 0.01 | 107 | 109 | 107 | 27.8260 | 27.7520 | 27.8289 | 0.9936 | 0.9941 | 0.9905 |
| | 0.05 | 545 | 550 | 551 | 20.7660 | 20.7287 | 20.7171 | 0.9687 | 0.9694 | 0.9654 |
| | 0.1 | 1076 | 1085 | 1096 | 17.8138 | 17.7772 | 17.7335 | 0.9274 | 0.9390 | 0.9180 |
| Baboon | 0.01 | 109 | 111 | 111 | 27.7630 | 27.6874 | 27.6879 | 0.9936 | 0.9954 | 0.9912 |
| | 0.05 | 543 | 536 | 539 | 20.7802 | 20.8372 | 20.8176 | 0.9657 | 0.9764 | 0.9660 |
| | 0.1 | 1080 | 1075 | 1082 | 17.7975 | 17.8151 | 17.7902 | 0.9309 | 0.9516 | 0.9337 |
| Airplane | 0.01 | 106 | 105 | 111 | 27.8645 | 27.9097 | 27.6741 | 0.9944 | 0.9933 | 0.9938 |
| | 0.05 | 546 | 549 | 558 | 20.7587 | 20.7343 | 20.6663 | 0.9667 | 0.9717 | 0.9719 |
| | 0.1 | 1080 | 1099 | 1079 | 17.7948 | 17.7210 | 17.7999 | 0.9438 | 0.9396 | 0.9498 |
| Lena | 0.01 | 108 | 107 | 112 | 27.8058 | 27.8307 | 27.6265 | 0.9934 | 0.9920 | 0.9914 |
| | 0.05 | 551 | 536 | 541 | 20.7226 | 20.8430 | 20.7987 | 0.9635 | 0.9639 | 0.9541 |
| | 0.1 | 1094 | 1056 | 1047 | 17.7401 | 17.8934 | 17.9298 | 0.9345 | 0.9260 | 0.9100 |

as high as possible such as 14 or 15 digits after the comma [5], so that the key space can reach at $10^{70}$. The key space is $S = 10^{70} \cong 2^{232} > 2^{100}$ [40], so that the cryptosystem can resist to brute-force attacks.

### 6.2 Key and Plain Image Sensitivity Analyses

It should be pointed out that any small modification at the initial values of the chaotic system would yield entirely different outputs. The key of the Modified Chua crypto system is a 'nonce', based on the hash value generated by the plain image and a random sequence. Thus, if the startup conditions are changed slightly, this would cause to generate different encrypted images. In the MCC system, it is observed that the algorithm is very delicate to the slightest variation in the key after applying the experiments.

Figure 10a is a one bit modified version of the Lena image and its encrypted state is given in Fig. 10b. The differences between Figs. 9h and 10b is also given in Fig. 10c. From this point of view, results of the encryptions are also divergent from each other.

### 6.3 Resistance to Known Plaintext and Chosen Plaintext Attacks

According to the proposed algorithm, the key strongly depends on the hash value of the original image. Therefore, different keys would be produced for different kind of images. Any attacker cannot decipher a particular image with a key used from another image. To conclude, the implemented software may be resistant to both the known—plaintext and chosen—plaintext attacks.

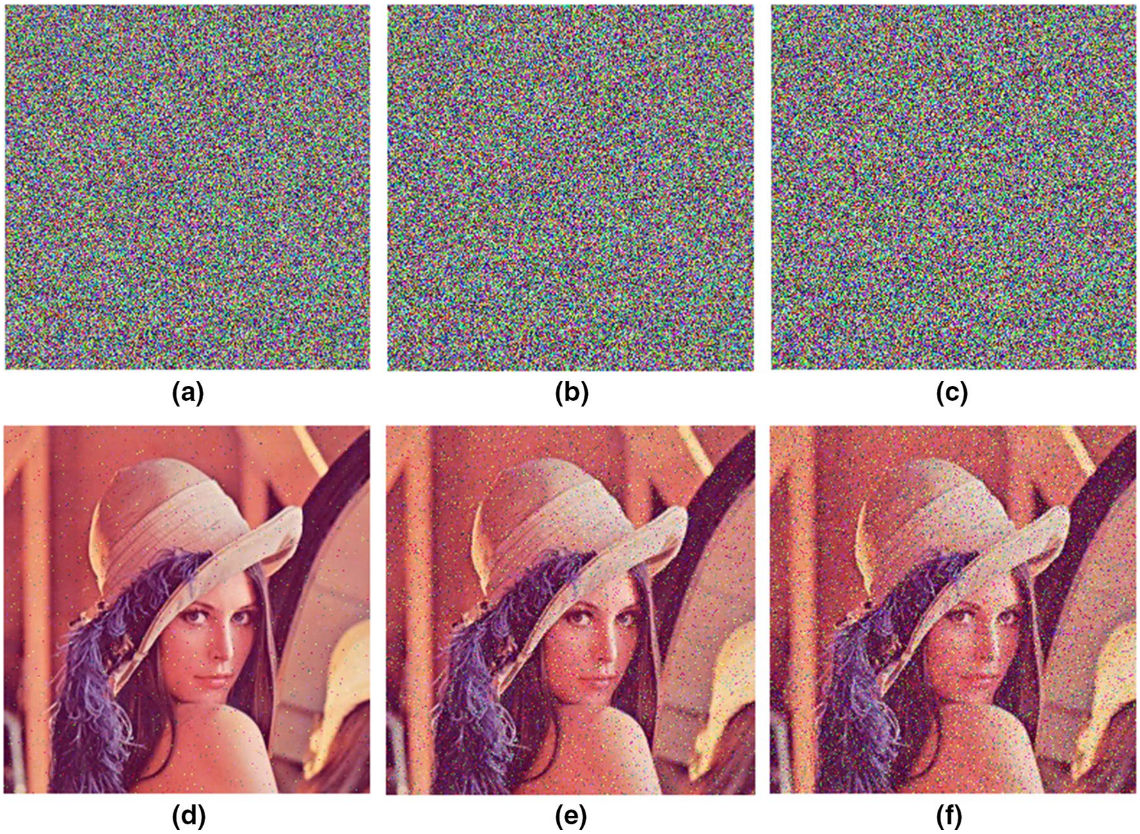

**Fig. 13** The cipher images with salt and pepper noise and their deciphered forms **a**, **d** noise with d = 0.01. **b**, **e** Noise with d = 0.05. **c**, **f** Noise with d = 0.1

### 6.4 Differential Attacks

Typically, in an image encryption unit, it is considered that the encrypted media should differ from its unencrypted version. To determine such a difference between the versions, the criteria NPCR [41] and UACI [42] are generally used in the literature.

In other words, the crypto system, which is recommended here should guarantee that the encrypted versions of two images become different to each other, when one bit modification is made into one of them. Tables 1 and 2 show the NPCR and UACI test findings for 1500 randomly selected pairs. The findings are satisfactory and the software is found to be robust against differential attacks.

### 6.5 Information Entropy Analysis

Information entropy is used for the measurement of an arbitrary distribution in a media file. The formulation of this operation is presented as follows [43]:

$$H(m) = \sum_{i=0}^{2^n-1} p\left(m_i\right) \log_2 \frac{1}{p\left(m_i\right)} \tag{16}$$

The information entropy for an encrypted version should be as high as possible, indeed it should be 8 for ideal results as in Ref. [44]. That makes the information difficult to expose. Here, Table 3 gives the information entropy results of three pieces of the encrypted image by using the Eq. (16). It is found that the results are close to 8.

### 6.6 Correlation Coefficient Analysis

There exists a relationship between neighboring pixels in any original image. In order to counteract statistical attacks for this relationship, the correlation on the neighboring pixels in an encrypted image should be minimal. The following formulation can be applied to calculate this correlation value between two adjacent pixels [48].

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{17}$$

$$\mathrm{cov}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - E(x)\right)\left(y_i - E(y)\right), \tag{18}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i,\ D(x) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - E(x)\right)^2. \tag{19}$$

Figure 11 shows the correlation distributions of two horizontally, vertically and diagonal adjacent pixels in the plain and ciphered Lena images. It is clear that the correlation between the neighboring pixels decreases substantially.

Table 4 gives the correlation values between the plain images and their encrypted versions. The test results prove that the correlation between the adjacent pixels of the encoded image version is very low, whereas the correlation between the plain images exists quite high. This ensures that the encryption performed here is effective.

### 6.7 Histogram Analysis

The histogram of an image provides information about the distribution of its pixel values and represents this image. As seen in Fig. 12, the histogram of the original image has several peaks while the encrypted image has a nearly constant distribution.

### 6.8 Resisting Noise Attack Analysis

The encoded image version is inevitably exposed to different types of noises, when the data passes through a real communication channel. This noise can cause problems during the acquisition of the original image. Therefore, the algorithm should be noise resistant, so that the encryption scheme can be valid. The Peak Signal-to-Noise Ratio (PSNR) is used to measure the quality of the decoded image after the attacks. For the image components, PSNR can be obtained by the following formulation [49]:

$$PSNR = 10 \times \log_{10}\left(\frac{255 \times 255}{MSE}\right)(\mathrm{dB}) \tag{20}$$

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{i=1}^{n}\left\|I_1(i,j) - I_2(i,j)\right\|^2 \tag{21}$$

MSE is the mean square error between the original and recovered images and is represented as $I_1(i,j)$ and $I_2(i,j)$ respectively, with the size of $mxn$. Figure 13 shows the encrypted image Lena exposed to the Salt Pepper noise with different density of this and its deciphered ones. The MSE and PSNR of these decoded images are shown in Table 5. From this Table 5 and Fig. 13, we can understand that the original image is entirely obtained again, which is noticeable, the PSNR value is about 30 dB, and the decoded images are highly correlated. This means that the decoded images are very close to the original image. Thus, it can be said that the proposed algorithm is resistant to resisting noise attacks to some degree.

### 6.9 Speed Analysis

The encryption speed is one of the key issues for the secure communication. Some precautions have been taken in order to speed up the encryption/decryption in the system. Initially, time-consuming operations were not used in the algorithm. For instance xor operation has been used to save computer time. Besides, the data obtained from the chaotic system is used for both diffusion and penetration process, thereby time is saved for the data production scheme, too. For instance, when Matlab R2017b is used in a PC with Intel Core i7-6700 CPU @3.4GHZ, 8 GB memory operating under Windows 10, the averaged time for the encryption of Lena image is 0.14 s, which is a sufficient value.

## 7 Conclusions

An original encryption/decryption algorithm has been developed for the encryption and the decryption of the images by using the modified Chua's circuit (MCC) system, which exhibits a hyperchaotic behavior for a large parameter regime due to the double frequency dependent nature. The Lyapunov spectrum has been found to characterize the hyperchaotic regime of the data. To our knowledge, the MCC system has been used for the first time for such an encryption study. Besides, the scrambling feature, which is implemented at a bit level and novel diffusion system using the MCC has been applied in the algorithm.

Following the encryption procedure, the encrypted colored image has been tested by a variety of tests including the secret key size and secret key sensitivity, histogram analysis, correlation analysis, differential analysis and information entropy analysis. The results of the analysis prove that the proposed algorithm is quite effective and provides an efficient technique for the color image encryption/decryption in the area of secure communication. The hyperchaotic MCC data give sufficient input to the algorithm to fulfill the security requirements. In addition to the security test results, the speed analyses give sufficient results. For instance, it gives 0.14 s for the encryption of colored image Lena.

**Acknowledgements** This paper has been devoted to the brave students and lecturers, who protect their labs and classrooms on 4th Oct. 2019 against the Gazi University Rectorship, which has decided to apply a unilateral decision on Gazi University Technology Faculty buildings.

## References

1. Gan Z, Chai X, Zhang M, Lu Y (2018) A double color image encryption scheme based on three-dimensional brownian motion. Multimed Tools Appl 77(21):27919–27953

2. Sahari ML, Boukemara I (2018) A pseudo-random numbers generator based on a novel 3D chaotic map with an application to color image encryption. Nonlinear Dyn 94(1):723–744
3. Alvarez G, Li S (2006) Some basic cryptographic requirements for chaos-based cryptosystems. Int J Bifurc Chaos 16(08):2129–2151
4. Liu H, Kadir A, Niu Y (2014) Chaos-based color image block encryption scheme using S-box. AEU Int J Electron Commun 68(7):676–686
5. Fridrich J (1998) Symmetric ciphers based on two-dimensional chaotic maps. Int J Bifurc Chaos 8(06):1259–1284
6. Kiraz MS, Uzunkol O (2016) Efficient and verifiable algorithms for secure outsourcing of cryptographic computations. Int J Inf Secur 15(5):519–537
7. Stinson DR (2005) Cryptography: theory and practice. CRC Press, Boca Raton
8. Fu C, Lin BB, Miao YS, Liu X, Chen JJ (2011) A novel chaos-based bit-level permutation scheme for digital image encryption. Opt Commun 284(23):5415–5423
9. Zhu ZL, Zhang W, Wong KW, Yu H (2011) A chaos-based symmetric image encryption scheme using a bit-level permutation. Inf Sci 181(6):1171–1186
10. Guan ZH, Huang F, Guan W (2005) Chaos-based image encryption algorithm. Phys Lett A 346(1–3):153–157
11. Xiao D, Liao X, Wei P (2009) Analysis and improvement of a chaos-based image encryption algorithm. Chaos Solitons Fractals 40(5):2191–2199
12. Wang Y, Wong KW, Liao X, Xiang T, Chen G (2009) A chaos-based image encryption algorithm with variable control parameters. Chaos Solitons Fractals 41(4):1773–1783
13. Celik K, Kurt E (2016) A new image encryption algorithm based on lorenz system. In: IEEE 8th international conference on electronics, computers and artificial intelligence (ECAI). Ploiesti, Romania, pp 1–6. https://doi.org/10.1109/ECAI.2016.7861097
14. Celík K, Kurt E, Stork M (2017) Can non-identical josephson junctions be synchronized? In: IEEE 58th international scientific conference on power and electrical engineering of Riga Technical University (RTUCON). Riga, Latvia, pp 1–5. https://doi.org/10.1109/RTUCON.2017.8124771
15. Abuturab MR (2012) Color image security system using double random-structured phase encoding in gyrator transform domain. Appl Opt 51(15):3006–3016
16. Lian S, Sun J, Wang Z (2005) Security analysis of a chaos-based image encryption algorithm. Phys A 351(2–4):645–661
17. Li C, Li S, Chen G, Halang WA (2009) Cryptanalysis of an image encryption scheme based on a compound chaotic sequence. Image Vis Comput 27(8):1035–1039
18. Liu H, Wang X (2010) Color image encryption based on one-time keys and robust chaotic maps. Comput Math Appl 59(10):3320–3327
19. Mazloom S, Eftekhari-Moghadam AM (2009) Color image encryption based on coupled nonlinear chaotic map. Chaos Solitons Fractals 42(3):1745–1754
20. Huang CK, Nien HH (2009) Multi chaotic systems based pixel shuffle for image encryption. Opt Commun 282(11):2123–2127
21. Chen L, Zhao D (2006) Optical color image encryption by wavelength multiplexing and lensless Fresnel transform holograms. Opt Express 14(19):8552–8560
22. Liu H, Wang X (2011) Color image encryption using spatial bit-level permutation and high-dimension chaotic system. Opt Commun 284(16–17):3895–3903
23. Zhang W, Wong KW, Yu H, Zhu ZL (2013) A symmetric color image encryption algorithm using the intrinsic features of bit distributions. Commun Nonlinear Sci Numer Simul 18(3):584–600
24. Zhou Y, Bao L, Chen CP (2014) A new 1D chaotic system for image encryption. Signal Process 97:172–182
25. Zhang Y, Xiao D (2014) An image encryption scheme based on rotation matrix bit-level permutation and block diffusion. Commun Nonlinear Sci Numer Simul 19(1):74–82
26. https://en.wikipedia.org/wiki/SHA-2. Accessed 15 Mar 2019
27. https://www.movable-type.co.uk/scripts/sha256.html. Accessed 15 Mar 2019
28. Kurt E (2006) Nonlinearities from a non-autonomous chaotic circuit with a non-autonomous model of Chua's diode. Phys Scr 74(1):22
29. Liu H, Wang X, Kadir A (2014) Chaos-based color image encryption using one-time keys and Choquet fuzzy integral. Int J Nonlinear Sci Numer Simul 15(1):1–10
30. Volos CK (2013) Image encryption scheme based on coupled chaotic systems. J Appl Math Bioinform 3(1):123
31. Yalcin ME, Suykens JA, Vandewalle J (2004) True random bit generation from a double-scroll attractor. IEEE Trans Circuits Syst I Regul Pap 51(7):1395–1404
32. Volos CK, Kyprianidis IM, Stouboulos IN (2009) Image encryption process based on a chaotic true random bit generator. IEEE 16th international conference on digital signal processing. Santorini-Hellas, Greece, pp 1–4. https://doi.org/10.1109/ICDSP.2009.5201107
33. Mirzaei O, Yaghoobi M, Irani H (2012) A new image encryption method: parallel sub-image encryption with hyper chaos. Nonlinear Dyn 67(1):557–566
34. Rhouma R, Meherzi S, Belghith S (2009) OCML-based colour image encryption. Chaos Solitons Fractals 40(1):309–318
35. Chai X, Gan Z, Zhang M (2017) A fast chaos-based image encryption scheme with a novel plain image-related swapping block permutation and block diffusion. Multimed Tools Appl 76(14):15561–15585
36. Chen G, Mao Y, Chui CK (2004) A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitons Fractals 21(3):749–761
37. Pareek NK, Patidar V, Sud KK (2006) Image encryption using chaotic logistic map. Image Vis Comput 24(9):926–934
38. Li C, Luo G, Qin K, Li C (2017) An image encryption scheme based on chaotic tent map. Nonlinear Dyn 87(1):127–133
39. Peng G, Min F (2017) Multistability analysis, circuit implementations and application in image encryption of a novel memristive chaotic circuit. Nonlinear Dyn 90(3):1607–1625
40. Liu W, Sun K, Zhu C (2016) A fast image encryption algorithm based on chaotic map. Opt Lasers Eng 84:26–36
41. Li Y, Wang C, Chen H (2017) A hyper-chaos-based image encryption algorithm using pixel-level permutation and bit-level permutation. Opt Lasers Eng 90:238–246
42. El Assad S, Farajallah M (2016) A new chaos-based image encryption system. Signal Process Image Commun 41:144–157
43. Vaidyanathan S, Akgul A, Kaçar S, Çavuşoğlu U (2018) A new 4-D chaotic hyperjerk system, its synchronization, circuit design and applications in RNG, image encryption and chaos-based steganography. Eur Phys J Plus 133(2):46
44. Akgul A, Pehlivan I (2016) A new three-dimensional chaotic system without equilibrium points, its dynamical analyses and electronic circuit application. Tech Gaz 23(1):209–214
45. Liu H, Wen F, Kadir A (2019) Construction of a new 2D Chebyshev-Sine map and its application to color image encryption. Multimed Tools Appl 78(12):15997–16010
46. Wang X, Guo K (2014) A new image alternate encryption algorithm based on chaotic map. Nonlinear Dyn 76(4):1943–1950
47. El-Latif AAA, Li L, Zhang T, Wang N, Song X, Niu X (2012) Digital image encryption scheme based on multiple chaotic systems. Sens Imaging Int J 13(2):67–88
48. Norouzi B, Seyedzadeh SM, Mirzakuchaki S, Mosavi MR (2015) A novel image encryption based on row-column, masking and

main diffusion processes with hyper chaos. Multimed Tools Appl 74(3):781–811

49. Salleh M, Ibrahim S, Isnin IF (2003) Image encryption algorithm based on chaotic mapping. Jurnal Teknologi 39(1):1–12

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Batuhan Arpacı** was born in Ankara, Turkey, in 1992. He received the bachelor's degree from the Department of Mathematics, Hacettepe University, Turkey. He has just copmpleted the master's degree in Information Systems at Gazi University. His main research direction is chaotic image encryption and decyrption.

**Erol Kurt** took his M. Sc. degree from the Institute of Science & Technology of Gazi University in Ankara, Turkey in 2001. He was awarded by an European Graduate College Grant during his Ph. D study at the Institute of Physics & Mathematics of Bayreuth University in Germany. He completed his Ph. D. degree in 2004 on the instabilities of rotating magnetic fluids. Then, he worked at Turkish Atomic Energy Authority R&D Department for 3 years. Beginning from the middle of 2009, he was assigned to the position of Associate Professor at Technology Faculty of Gazi University in Ankara. His main teaching and research areas include nonlinear phenomena in electrical/electronic circuits, electric machine design, mechanical vibrations, chaos, plasmas, fusion and magneto hydrodynamics. He has authored and co-authored many scientific papers. He is the founder chairman to the serial conference European Conference on Renewable Energy Systems (ECRES) and the guest editor for several reputable special issue journals. He is the Editor-in-Chief to Journal of Energy Systems (dergipark.gov.tr/jes). He is the member of Turkish Science Research Foundation (TUBAV).

**Kayhan Çelik** was born in Kayseri, Turkey in 1988. He received the B.S. degree in Electrical and Electronics Engineering from Erciyes University, Kayseri, Turkey, in 2011 and the M.S. degree from Gazi University, Ankara, Turkey in 2015. He is currently working toward the Ph.D. degree at the same university. His research interests include energy harvesting, chaotic image encryption and antenna design for energy harvesting applications.

**Bünyamin Ciylan** currently Associate Professor at Gazi University Faculty of Technology Computer Engineering. His research interests are in the areas of privileged account management, network forensics, ICS cyber security, information security standards, information technology law, modeling secure networks.