Signal Processing: *Image Communication* 41 (2016) 144–157

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

# A new chaos-based image encryption system

Safwan El Assad *, Mousa Farajallah

*Ecole polytech – University of Nantes-Rue Christian Pauc-44306, Nantes cedex 3, France*

ARTICLE INFO

*Article history:*
Received 6 July 2015
Received in revised form
18 September 2015
Accepted 20 October 2015
Available online 3 November 2015

*Keywords:*
Chaos-based cryptosystem
New formulation of the 2D-cat map
Binary diffusion matrix
Security analysis

ABSTRACT

During the last decade, a variety of chaos-based cryptosystems has been introduced to protect the content of the transmitted images. In this paper, we propose a new fast, simple, and robust chaos-based cryptosystem structure and we analyze its performances. The cryptosystem uses a diffusion layer followed by a bit-permutation layer, instead of byte-permutation, to shuffle the positions of the image pixels. Moreover, the permutation layer is achieved by a new proposed formulation of the 2D cat map that allows an efficient implementation, measured by the time complexity, in terms of arithmetic and logic operations, and also, in terms of clock cycles, of the key-dependent permutation process in comparison with the standard one. Hence, it provides a very fast diffusion process to spread the influence of a single bit over the others. The new cryptosystem includes a robust and uniform chaotic pseudo-random generator (a very simplified version of a generator published in our patent) to change the control parameters in each round of the encryption/decryption processes. The generator is highly nonlinear and produces robust sequences of discrete values having very long orbits. The proposed cryptosystem is defined on finite numbers, and its speed is faster than many chaos-based cryptosystems, while having a very high security level. The security analysis and the obtained simulation results show that the proposed cryptosystem is resistant to various types of attacks and it is efficient for hardware and software implementation.



## 1. Introduction

In any public communication network, such as satellite, mobile-phone, and the Internet, it is almost impossible to prevent unauthorized people from eavesdropping. To make use of the already existed public communication networks and to maintain the secrecy, cryptographic techniques are applied [1]. The security of image and video data has become increasingly important for many applications, including video conferencing, medical imaging, industrial, and military imaging systems. An increasing number of applications demand both real-time performance and security: private multimedia messages exchanged by portable devices over the wireless networks and sensitive data exchanged in wireless sensor networks. With a fixed block size, the advanced encryption standard (AES) [2–4] is not suitable for the above applications and for video encryption [5,6]. For example, in selective encryption we need to wait to have 128 bits before the starting of encryption process. This is an inconvenient for the latency of the system. Chaos-based cryptosystems are more flexible, whatever the block size.

The two important properties of confusion and diffusion for any good standard encryption algorithm [7] are also existed in chaotic systems, which are ergodic and extremely sensitive to the secret key, composed by the initial conditions and the system parameters. According to the classification of chaotic systems, the chaotic encryption schemes, which have been proposed, can be divided into analog chaotic cryptosystems utilizing continuous dynamical

* Corresponding author.

http://dx.doi.org/10.1016/j.image.2015.10.004
0923-5965/

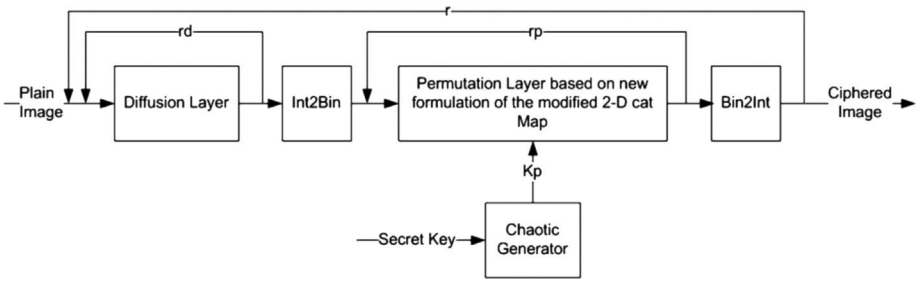


**Fig. 1.** Description of the encryption process.

systems [8] and digital chaotic cryptosystems utilizing discrete dynamical systems [9,10]. In addition, chaos based encryption algorithms are usually characterized by a low complexity of their structures. Therefore, in recent years, a number of chaos-based cryptosystems have been proposed [7,11–16]. Most of them are blocks ciphers and based on the substitution–permutation network (SPN). In a chaos-based encryption algorithm, the substitution and the permutation operations are done in accordance with chaotic sequences [17,18].

In [19], Fridrich proposed a chaos-based encryption scheme in which a process of permutation on the pixels, followed by a nonlinear process are iterated. The permutation is done using three 2D chaotic maps, namely the Backer map, the Cat map, and the Standard map. The nonlinear process is achieved by a nonlinear feedback register.

Masuda et al, considered in [20] two classes of chaotic finite-state maps: key-dependent chaotic S-boxes and chaotic mixing transformation. They proposed two chaotic block ciphers, i.e., uniform and Feistel, and they exactly estimate bounds for the differential probability and the linear probability to make their ciphers resistant to differential and linear cryptanalysis.

In [21], a fast and robust image encryption algorithm called Enhanced 1-D Chaotic Key-Based Algorithm for Image Encryption (ECKBA) was proposed. The algorithm performs $r$ rounds of an SP-network on each pixel. Two PWLCM maps are used, one in the substitution process (addition modulo 256 and bitwise operations) and the other one in the permutation process. The permutation is of degree 8, and its index in the full symmetric group S8 is sorted in lexicographical Cartesian of $i$ order (see appendix of reference [21]). The weakness of this algorithm is the error propagation caused by the used perturbation technique.

Yang et al, derived in [22] a fast image encryption and authentication scheme. A key Hash function is introduced to generate 128 bits hash value from both of the plain image and the secret hash keys. The hash value plays the role of the key for encryption and decryption, while the secret hash keys are used to authenticate the decrypted image. The permutation and substitution are performed in a single scan of the plain-image pixels. The permutation process is achieved by the modified standard map and the substitution process (based on a logistic map) is done in a way that the change made to a particular pixel depends on the accumulated effect of all previous pixel values. Chen et al. analyzed the period distribution of the generalized discrete cat map over the Galois ring. Used as a generator, the produced sequences by this map are modeled as a 2-dimensional LFSR sequences [23].

Zhang et al. [15] proposed a chaotic image encryption based on circular substitution box and key stream buffer. The encryption process consists of a substitution and diffusion layers to encrypt each pixel of the image using an encryption key that depends on the plain image. In this way, one encryption round is sufficient to achieve the security against the known and plain text attacks. However, this scheme is not a practical one, because, it is obligatory to send a new encryption key for each new plain image.

In this paper, we propose an efficient chaos-based cryptosystem formed by two layers: a diffusion layer, operating on the pixels, followed by a key-dependent chaotic permutation process, operating on the bits. The linear diffusion transformation is done by a binary matrix that gives high diffusion effects without using any multiplications. So, it provides a simpler implementation of bitwise operations. The permutation process is achieved by using a new proposed expression of the modified 2D cat map which is implemented with an efficient manner, as compared to the basic calculation. This proposed structure achieves a higher security level. Also, it is faster than many known chaos-based encryption algorithms. The paper is organized as follows. In Section 2, the proposed chaos-based cryptosystem is described. In Section 2.1, we describe in detail the encryption process. The diffusion layer is given in Section 2.1.1 and in Section 2.1.2, we study in detail the formulation and the time complexity in terms of arithmetic and logic operations, and also, in terms of clock cycles, of the key-dependent permutation process. The implemented chaotic generator and its performances are given in Section 2.2. Section 2.3 presents in detail the decryption process. In Section 3, we illustrate the obtained performance in terms of time consuming and security analysis of our proposed cryptosystem. Finally, Section 4 contains our conclusion and perspectives.

## 2. Proposed chaos-based cryptosystem

The structure of the proposed chaos-based cryptosystem is shown in Fig. 1, for the encryption part, and in Fig. 4, for the decryption part. The cryptosystem is a block cipher implemented in CBC mode. This new structure has efficient encryption features, i.e., robustness against the cryptanalysis and a high calculation speed. Moreover, it is adequate for a software and hardware implementations (using FPGA card or an ASIC).

### 2.1. Description of the encryption process

The encryption part of the proposed cryptosystem (see Fig. 1) consists, first of a diffusion layer based on a binary matrix of size $32 \times 32$. This process is repeated $r_d$ times. In the next step, the data is prepared for the permutation layer using the integer to binary number converter function *Int2Bin*(). The permutation layer is based on a new modified 2-D cat map, working at the data bit level, and it gets the dynamic parameters from a chaotic generator. It is iterated $r_p$ times. The advantage of the bit permutation layer is that it produces a higher security than both the permutation and the substitution layers based on bytes. Indeed, theoretically, when a bit permutation layer is applied on a block, it performs, on one scan, a substitution and a diffusion operations on the bytes. As a byte contains 8 bits, then, 8 bit permutations possibly will affect 8 bytes, and then each byte (8 bit permutations) possibly will transfer the effects of the confusion and of the diffusion to other 8 bytes. As a result, the effects of the confusion and of the diffusion are better than in the case of byte permutation followed by byte substitution, because in this last case one byte affects only one byte. In the proposed cryptosystem, 6 encryption rounds based on byte permutation and on the byte substitution are needed to produce the same level of security as a process of bit-permutation which is executed in only one round. At the end of the first iteration, a binary to integer convertor function *Bin2Int*() is used to prepare the data for the next iteration. The whole process is repeated $r$ times until it reaches the required avalanche effect and therefore the maximum security level.

#### 2.1.1. Diffusion layer

The diffusion layer works on blocks of 32 bytes each, it achieves a local diffusion, and it is defined by the following equation:

$$\begin{bmatrix} Od_0 \\ Od_1 \\ \vdots \\ Od_{31} \end{bmatrix} = [\mathbf{DM}] \odot \begin{bmatrix} O_0 \\ O_1 \\ \vdots \\ O_{31} \end{bmatrix} \tag{1}$$

where $Od_i, O_i \in [0, 255]$ are the output and the input bytes of the diffusion process, [**DM**] is the binary diffusion square matrix of $32 \times 32$, which must be invertible but not self-invertible. Indeed, the self-invertible matrix is one of the weak points of any cryptosystem (see Appendix A). We implemented the diffusion layer using the binary matrix proposed in [24]. The number of branches obtained from this matrix is 10 and therefore, the diffusion power is important. Actually, the maximum branch of $32 \times 32$ binary matrix is regarded to be 12, but those matrices of branch 12 do not meet the following criteria [24]: efficient implementation in 8-bit processor; secure against truncated and impossible differential attacks; the branch number is equal or bigger than 10. The branch number $B_{\mathbf{DM}}$ of the diffusion layer **DM** is defined by

$$B_{\mathbf{DM}} = \min_{x \neq 0}[w_H[\mathbf{x}] + w_H(DM[\mathbf{x}])] \tag{2}$$

where $w_H[\mathbf{x}] = card\{i / 0 \leq i < n, x_i \neq 0\}$ is the Hamming weight of $n$ dimensional vector $\mathbf{x}$ in Appendix B, we give the detailed equations of the diffusion and the inverse diffusion processes. In order to define the $\odot$ matrix operator in (1), we write the first output $Od_0$ in an equivalent extended form as: $Od_0 = O_0 \oplus O_2 \oplus O_3 \oplus O_4 \oplus O_5 \oplus O_8 \oplus O_9 \oplus O_{10} \oplus O_{12} \oplus O_{13} \oplus O_{17} \oplus O_{18} \oplus O_{19} \oplus O_{24} \oplus O_{25} \oplus O_{29} \oplus O_{31}$ where $\oplus$ is the bitwise operator (xor).

#### 2.1.2. Permutation layer

The permutation layer is the process of changing or reordering the bit/byte position on the image, without changing its value.

*Permutation Layer Based on the standard 2-D cat map*: The permutation process based on the standard 2-D cat map is given by (3) [19,25]. Notice that, the determinant of the Jacobian cat matrix is equal to one and then, the cat map is bijective. Hence, it is a one-to-one mapping and each position in the square matrix is transformed to another position uniquely:

$$\begin{bmatrix} i\_new \\ j\_new \end{bmatrix} = \begin{bmatrix} 1 & u \\ v & v \times u + 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} rx + ry \\ ry \end{bmatrix} Mod \begin{bmatrix} M \\ M \end{bmatrix} \tag{3}$$

where $[i,j]$ and $[i\_new, j\_new]$ are the original and the permuted bit positions of the *data_bit_matrix* of the size $M \times M$; where $M = (32 \times 8)^{0.5} = 16$. The parameters of the cat map $u, v, rx, ry$, in the range $[0, M-1]$, are supplied by the generator of chaotic sequences. They are dynamic, because they are changed each iteration and each block.

*Permutation layer based on the modified 2-D cat map*: The modified 2-D cat map is given by (4).

$$\begin{aligned} \mathbf{MI\_new} &= mod(\mathbf{MI} + u \times \mathbf{Mc} + \mathbf{Lc}, M) \\ \mathbf{Mc\_new} &= mod(v \times \mathbf{MI} + (v \times u + 1) \times \mathbf{Mc} + \mathbf{C}, M) \end{aligned} \tag{4}$$

where **MI**, **Mc**, **Lc**, and **C** are square matrices $(M \times M)$ given by the following forms:

$$\mathbf{MI} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ M & M & \cdot & M \end{bmatrix} \tag{5}$$

$$\mathbf{Mc} = \mathbf{MI}^{\mathsf{T}} \tag{6}$$

Assume $\mathbf{J_M}$ be a square matrix of size $M \times M$, whose elements are all 1, then

$$\mathbf{Lc} = (rl + rc) \times \mathbf{J_M} \tag{7}$$

$$\mathbf{C} = rc \times \mathbf{J_M} \tag{8}$$

**MI_new** and **Mc_new** are the permuted bit positions of the **MI** and **Mc** matrices.

*New formulation of the modified 2-D cat map for an adequate implementation in C code*: The mathematical form of (4) takes a lot of time in matrix multiplication to find the **MI_new** and **Mc_new** matrices, because it needs much more memory for matrix's data. For that, we derive a new formulation to reduce the consumed time and memory. The new formulation is derived from (4) and (5)–(8) as follows:

$$\mathbf{MI\_new} = mod\left( \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ M & M & \cdot & M \end{bmatrix} \right.$$

$$+\begin{bmatrix} 1\times u & 2\times u & \cdots & M\times u \\ 1\times u & 2\times u & \cdots & M\times u \\ \vdots & \vdots & \ddots & \vdots \\ 1\times u & 2\times u & \cdots & M\times u \end{bmatrix} + \mathbf{Lc}, M \Bigg) \tag{9}$$

Let us assume that

$$\mathbf{Mv} = [1 \;\; 2 \;\; \cdots \;\; M]$$
$$\mathbf{Mv} = [Mv[0] \;\; Mv[1] \;\; \cdots \;\; Mv[M-1]] \tag{10}$$

$$u\mathbf{Mv} = u \times \mathbf{Mv} = [u \;\; 2\times u \;\; \cdots \;\; M\times u]$$
$$u\mathbf{Mv} = [uMv[0] \;\; uMv[1] \;\; \cdots \;\; uMv[M-1]] \tag{11}$$

Then, (9) can be rewritten as

$$\mathbf{Ml\_new} = mod\Bigg(\begin{bmatrix} Mv[0] & Mv[0] & \cdots & Mv[0] \\ Mv[1] & Mv[1] & \cdots & Mv[1] \\ \vdots & \vdots & \ddots & \vdots \\ Mv[M-1] & Mv[M-1] & \cdots & Mv[M-1] \end{bmatrix} + \begin{bmatrix} uMv[0] & uMv[1] & \cdots & uMv[M-1] \\ uMv[0] & uMv[1] & \cdots & uMv[M-1] \\ \vdots & \vdots & \ddots & \vdots \\ uMv[0] & uMv[1] & \cdots & uMv[M-1] \end{bmatrix} + \mathbf{Lc}, M \Bigg) \tag{12}$$

Now, let us suppose that $u\mathbf{Mv}m$ is the modified of $u\mathbf{Mv}$:

$$u\mathbf{Mv}m = \begin{bmatrix} u+rl+rc & 2\times u+rl+rc & & \\ & \cdots & & M\times u+rl+rc \end{bmatrix} \tag{13}$$

Then

$$Ml\_new[i,j] = mod(Mv[i] + uMvm[j], M) \tag{14}$$

To save memory and time in the implementation, (14) can be equivalently written as

$$xrow = mod(Mv[i] + uMvm[j], M) \tag{15}$$

(*xrow* is a dummy variable used instead of the **Ml_new**) **Mc_new** is derived in a similar procedure as above:

$$\mathbf{Mc\_new} = mod\Bigg(\begin{bmatrix} 1\times v & 1\times v & \cdots & 1\times v \\ 2\times v & 2\times v & \cdots & 2\times v \\ \vdots & \vdots & \ddots & \vdots \\ M\times v & M\times v & \cdots & M\times v \end{bmatrix} + \begin{bmatrix} u\times v+1 & 2\times u\times v+2 & \cdots & M\times u\times v+M \\ u\times v+1 & 2\times u\times v+2 & \cdots & M\times u\times v+M \\ \vdots & \vdots & \ddots & \vdots \\ u\times v+1 & 2\times u\times v+2 & \cdots & M\times u\times v+M \end{bmatrix} + \mathbf{C}, M \Bigg) \tag{16}$$

Let us assume that

$$v\mathbf{Mv} = v \times \mathbf{Mv} = [v \;\; 2\times v \;\; \cdots \;\; M\times v]$$
$$v\mathbf{Mv} = [vMv[0] \;\; vMv[1] \;\; \cdots \;\; vMv[M-1]] \tag{17}$$

$$v\mathbf{Mv}m1 = (u\times v+1) \times \mathbf{Mv} = [u\times v+1 \;\; 2\times u\times v+2 \;\; \cdots \;\; M\times u\times v+M]$$
$$v\mathbf{Mv}m1 = [vMvm1[0] \;\; vMvm1[1] \;\; \cdots \;\; vMvm1[M-1]] \tag{18}$$

Also, (16) can be equivalently written as

$$\mathbf{Mc\_new} = mod\Bigg(\begin{bmatrix} vMv[0] & vMv[0] & \cdots & vMv[0] \\ vMv[1] & vMv[1] & \cdots & vMv[1] \\ \vdots & \vdots & \ddots & \vdots \\ vMv[M-1] & vMv[M-1] & \cdots & vMv[M-1] \end{bmatrix} + \begin{bmatrix} vMvm1[0] & vMvm1[1] & \cdots & vMvm1[M-1] \\ vMvm1[0] & vMvm1[1] & \cdots & vMvm1[M-1] \\ \vdots & \vdots & \ddots & \vdots \\ vMvm1[0] & vMvm1[1] & \cdots & vMvm1[M-1] \end{bmatrix} + \mathbf{C}, M \Bigg) \tag{19}$$

let us suppose that

$$v\mathbf{Mv}m2 = \begin{bmatrix} v\times u+1+rc & & 2\times u\times v+2+rc \\ \cdots & M\times u\times v+M+rc & \end{bmatrix} \tag{20}$$

Then

$$Mc\_new[i,j] = mod(vMv[i] + vMvm2[j], M) \tag{21}$$

To save memory and time in the implementation, (21) can be equivalently written as

$$ycol = mod(vMv[i] + vMvm2[j], M) \tag{22}$$

(*ycol* is a dummy variable used instead of the **Mc_new**) The pseudo C code implementation of (15) and (22) is given in Algorithm 1.

**Algorithm 1.** Algorithm in pseudo C code for the implementation of the new 2-D cat map formulation.

```
for i=0 to M−1
  Mv[i] = i+1
End i
for k=0 to r−1
for i=0 to M−1
  uMvm[i] = Mv[i] × u[k] + rl[k] + rc[k]
  vMv[i] = Mv[i] × v[k]
  vMvm2[i] = u[k] × vMv[i] + Mv[i] + rc[k]
End i
for i=0 to M−1
  for j=0 to M−1
    xrow = mod(Mv[i] + uMvm[j], M)
    ycol = mod(vMv[i] + vMvm2[j], M)
    data_bit_1[i,j] = data_bit_2[xrow, ycol]
  End j
End i
End k
```

where *data_bit_2*[*xrow*, *ycol*] is the source matrix (the one contains the data bits of the image after diffusion), and *data_bit_1*[*i*,*j*] is the destination matrix (the permuted data bits).

*Time complexity analysis*: We propose below a comparative theoretical analysis of the time complexity in terms of arithmetic, logic operations and clock cycles of our new formulation of the modified 2-D cat map and the standard cat.

Assume *X*1 is the addition operation, *X*2 is the multiplication operation and *X*3 is the modulus operation.

**Remark.**

1. The analysis is carried out for a block of size of $M \times M$ and for one encryption round.

2. The assign operation (see line 14 in Algorithm 1) is identical in both implementation. It requires $M^2$ assign operations and $M^2$ memory locations.

*Time complexity of Eq. (3).*

The time complexity of Eq. (3), ($TC_{Eq.(3)}$) is derived based on the following assumptions:

1. The $v \times u+1$ value is calculated one time per block and it is saved in a register ($Reg_1$) to be used.
2. The $rx+ry$ is calculated one time per block and it is saved in a register $Reg_2$.

The time complexity of Eq. (3) is given by

$$TC_{Eq(3)} = TC_{i_n} + TC_{j_n}$$

where $TC_{i_n}$ and $TC_{j_n}$ are the time complexity of $i_n$ and $j_n$ respectively:

$$TC_{i_n} = 2 \times M^2 \times X1 + M^2 \times X2 + M^2 \times X3$$

$$TC_{j_n} = 2 \times M^2 \times X1 + 2 \times M^2 \times X2 + M^2 \times X3$$

$$TC_{Eq(3)} = 4 \times M^2 \times X1 + 3 \times M^2 \times X2 + 2 \times M^2 \times X3$$

In terms of memory, Eq. (3) needs $M^2$ locations.

*Time complexity of the proposed implementation given by Eqs. (15) and (22)*

Based on the previous assumptions (1) and (2) and on the following two assumptions in Algorithm 1:

1. The array $Mv[i]$ of lines 6, 7, 8 and 12 is replaced by $Reg_1 = i+1$.
2. The value $rl[k]+rc[k]$ is calculated one time for the loop begins at line 5 and saved in a register ($Reg_2$).

The time complexity of Eqs. (15) and (22) is derived as follows:

1. Line 6: $M$ addition operations are needed to calculate $Mv[i]$, $M$ multiplication operations are needed to evaluate $Mv[i] \times u[k]$ and $M$ addition operations are needed to add the value of the $Reg_2$. The total required operations are: $TC_6 = 2 \times M \times X1 + M \times X2$.
2. Line 7: Only $M$ multiplication operations are needed to compute $Mv[i] \times v[k]$, because $Mv[i]$ was already calculated in the previous line. The total needed operations are: $TC_7 = M \times X2$.
3. Line 8: $M$ multiplication operations are needed to compute $u[k] \times vMv[i]$, and $2 \times M$ addition operations. The total required operations are: $TC_8 = 2 \times M \times X1 + M \times X2$.
4. Line 12: $M$ addition operations are needed to calculate $Mv[i]$, $M^2$ addition operations are needed to calculate $Mv[i]+uMvm[j]$, and $M^2$ modulus operations are needed. The total required operations are: $TC_{12} = M \times X1 + M^2 \times X1 + M^2 \times X3$.
5. Line 13: $M^2$ addition operations are needed to evaluate $vMv[i]+vMvm2[j]$, and $M^2$ modulus operations are needed. The total required operations are: $TC_{13} = M^2 \times X1 + M^2 \times X3$.

The time complexity $TC_{Proposed}$ is

$$TC_{Proposed} = TC_6 + TC_7 + TC_8 + TC_{12} + TC_{13}$$

$$TC_{Proposed} = (2 \times M^2 + 5 \times M) \times X1 + 3 \times M \times X2 + 2 \times M^2 \times X3$$

Table 1 presents the comparative of the time complexity in terms of arithmetic and logic operations. It is clear from Table 1 that the time complexity of the proposed implementation is less than the standard one given by Eq. (3). Indeed, for the proposed implementation, the required multiplication operations is $M$ times less than the standard implementation. And, the required addition operations is less than the standard implementation (for $M > 2$).

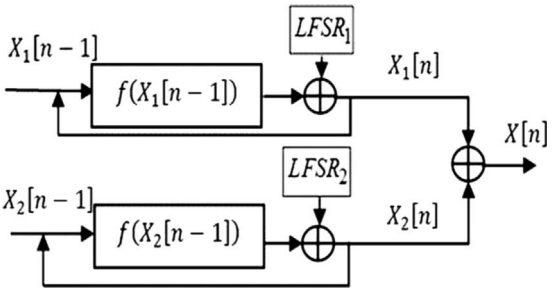


**Fig. 2.** Proposed chaotic sequence generator.

**Table 1**
Time complexity of arithmetic and logic operations.

| | $X1$ | $X2$ | $X3$ | $TC$ |
|---|---|---|---|---|
| Eq. (3) | $4 \times M^2$ | $3 \times M^2$ | $2 \times M^2$ | $4 \times M^2 \times X1 + 3 \times M^2 \times X2 + 2 \times M^2 \times X3$ |
| Ours | $2 \times M^2 + 5 \times M$ | $3 \times M$ | $2 \times M^2$ | $(2 \times M^2 + 5 \times M) \times X1 + 3 \times M \times X2 + 2 \times M^2 \times X3$ |

**Table 2**
Time complexity in clock cycles for all operations.

| | $X1$ | $X2$ | $X3$ | $R$ | $W$ | $Total$ |
|---|---|---|---|---|---|---|
| Eq. (3) | $2 \times M^2$ | $3 \times M^2$ | $2 \times M^2$ | 0 | 0 | $7 \times M^2$ |
| Ours | $M^2 + 2.5 \times M$ | $3 \times M$ | $2 \times M^2$ | $4 \times M$ | $3 \times M$ | $3 \times M^2 + 12.5 \times M$ |

Moreover, for a fair comparison, we give in Table 2, the time complexity in terms of clock cycles (*TCC*) for the operations: addition, multiplication, modulus, Read (R) and Write (W).

For each operation, the *TCC* is calculated according to the published manual of the instruction table in pages 159–160 of [26]. The Microprocessor version (Sandy Bridge Microprocessor code) of [26] is close to the used one in our simulation tests. The addition operation ($X1$) takes 0.5 clock cycle, the multiplication operation ($X2$) takes 1 clock cycle, the modulo operation ($X3$) takes 1 clock cycle, the read operation (R) takes 1 clock cycle, and the write operation (W) takes 1 clock cycle. Then, according to Table 2, for $M>3$, the proposed implementation consumes less number of cycles than the standard implementation. As an example, for $M=16$, the proposed implementation requires 968 cycles while Eq. (3) requires 1792 cycles. However, compared with the standard implementation, the proposed implementation needs of $M^2+3M$ locations of memories instead of $M^2$ and also, of 3 operations of writing (in lines, 6, 7 and 8) and 4 operations of reading (in lines 8, 12 and 13), of the Algorithm 1. In terms of the execution time, the average execution time of the modified 2D cat map is approximately 1.2 faster than the standard 2D cat map.

### 2.2. Implemented chaotic generator

To generate a chaotic sequences, a process of discretization is used. Therefore, a small periodicity of the trajectory could appear, depending on the initial conditions and the parameters of the generator. This is a dangerous weakness that must be avoided. The implemented generator of a discrete chaotic sequences in this cryptosystem is a very simplified version of the one proposed by El Assad and Noura in a patent [27]. It consists of two chaotic maps (i.e., Skew tent and PWLCM), connected in parallel as shown in Fig. 2. Each component generator is perturbed using a linear feedback shift register (LFSR). This ensures a very large periodicity for all generated sequences. The discrete Skew Tent Map and the Discrete Piece-wise Linear Chaotic Map (PWLCM) are defined in the following.

First, the discrete Skew Tent Map is defined as [20]

$$X[n]=F[X[n-1]]=\begin{cases}\left\lfloor\dfrac{2^N\times X[n-1]}{P}\right\rfloor & \text{if } 0<X[n-1]<P\\ 2^N-1 & \text{if } X[n-1]=P\\ \left\lfloor\dfrac{2^N\times(2^N-X[n-1])}{2^N-P}\right\rfloor & \text{if } P<X[n-1]<2^N\end{cases} \tag{23}$$

where $P$ is the control parameter and is ranging from 1 to $2^N-1$, and the finite precision $N=32$ bits.

Second, the PWLCM map is defined [28] as

$$X[n]=F[X[n-1]]=\begin{cases}\left\lfloor\dfrac{2^N\times X[n-1]}{P}\right\rfloor & \text{if } 0<X[n-1]<P\\ \left\lfloor\dfrac{2^N\times(X[n-1]-P)}{2^{N-1}-P}\right\rfloor & \text{if } P<X[n-1]<2^{N-1}\\ \left\lfloor\dfrac{2^N\times(2^N-X[n-1]-P)}{2^{N-1}-P}\right\rfloor & \text{if } 2^{N-1}\le X[n-1]<2^N-P\\ \left\lfloor\dfrac{2^N\times(2^N-X[n-1])}{P}\right\rfloor & \text{if } 2^N-P\le X[n-1]<2^N-1\\ 2^N-1 & \text{otherwise}\end{cases} \tag{24}$$

Here, the control parameter $P$ of PWLCM is ranging from 1 to $2^{(N-1)}-1$. The proposed chaotic generator has the following cryptographic properties: random pseudo mapping, delta-like auto-correlation, nearly zero cross correlation, uniform distribution, passing empirical statistic NIST 800-22 (National Institute of Standards and Technology) tests [29], and having a large size of the secret key. The size of the secret key is determined by four initial conditions: 2 values for the Skew tent and PWLCM maps of size $N$ and the other for the two LFSRs, and two control parameters, i.e., $P_1$ (for the Skew tent) and $P_2$ (for the PWLCM).

$$|K|=2\times N+|P_1|+|P_2|+|K_1|+|K_2|=169 \text{ bits} \tag{25}$$

with $N=32$, $|K_1|=23$, $|K_2|=19$, $|P_1|=32$, $|P_2|=31$. We have performed the NIST test (a battery of 188 tests) on 100 sequences, each containing one million bits. In Fig. 3, we show the obtained proportion value of sequences passing a test, versus the indice of the test (from 1 to 188).

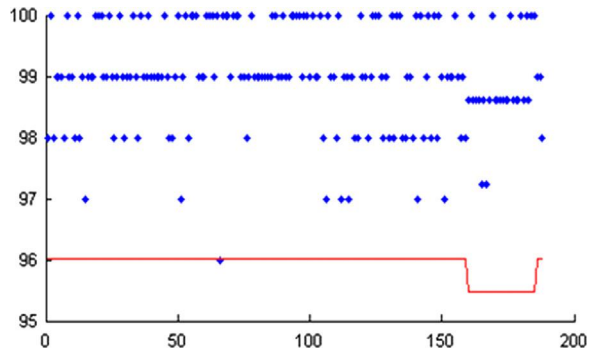


**Fig. 3.** Proportion values of NIST test versus the Indice of the test.

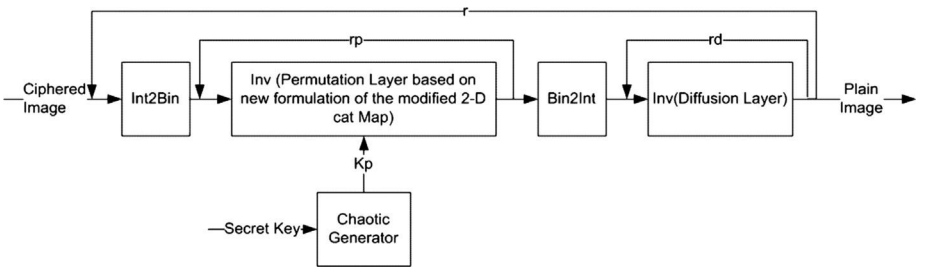


**Fig. 4.** Description of the decryption process.

**Table 3**
Average encryption time of the proposed algorithm, and some known algorithms (in milli-second).

| Algorithm | 128 | 256 | 512 | 1024 Gray |
|---|---|---|---|---|
| Proposed | 2.24 | 8.38 | 31.72 | 41.87 |
| Socek et al. [21] | | | | 1570 |
| Lian et al. [31] | | | 349 | |
| Yang et al. [22] | | | 93.8 | |
| Wong et al. [32] | | | 95.6 | |
| Zhang et al. [15] | | 7.5 | 30 | |

**Table 4**
Average decryption time of the proposed algorithm, and some known algorithms (in milli-second).

| Algorithm | 128 | 256 | 512 | 1024 Gray |
|---|---|---|---|---|
| Proposed | 2.34 | 8.48 | 32.17 | 42.27 |
| Socek et al. [21] | | | | 1570 |
| Lian et al. [31] | | | 358 | |
| Yang et al. [22] | | | 102.6 | |
| Wong et al. [32] | | | 104.31 | |
| Zhang et al. [15] | | 7.5 | 30 | |

**Table 5**
Encryption throughput and number of cycles per byte of the proposed algorithm and some known algorithms.

| Algorithm | ET (MBps) | Cycles per Byte |
|---|---|---|
| Proposed | 22.70 | 130 |
| Socek et al. [21] | 0.48 | 2595 |
| Lian et al. [31] | 2.14 | 754 |
| Yang et al. [22] | 7.66 | 274 |
| Wong et al. [32] | 7.54 | 398 |
| Zhang et al. [15] | 25 | 122 |

As we can see, the proposed generator passes all the tests and therefore, it is statistically secure. In addition, the implemented generator, combines two non-linearity functions (Skew tent map, PWLCM map), and it is not possible to analyze each map separately and to guess the secret key (2 parameters and 4 initial conditions). This is equivalent to the ciphertext attack or to the exhaustive attack. Moreover, a chaotic generator has only an output, there is no input text, except the secret key, then, the plaintext, the known-plaintext, and the chosen-plaintext can be assimilated to the key sensitivity attack. Any chaotic generator can succeed this test, because the main property of a chaotic system is its extreme sensibility to even one bit change of the secret key. As all previous attacks are ineffective, then the proposed generator can be used in secure communication systems.

*Structure and size of the dynamic keys*: The structure and the size of the dynamic keys used for the permutation and reverse permutation layers, are given below:

$$Kp = [Kp_1 \| Kp_2 \| \cdots \| Kp_r]$$
$$Kp_j = \left[Kp_{j,1} \| Kp_{j,2} \| \cdots \| Kp_{j,rp}\right] \quad j = 1, 2, \ldots r$$
$$Kp_{j,k} = [u_{j,k} \| v_{j,k} \| rl_{j,k} \| rc_{j,k}] \quad k = 1, 2, \ldots rp \tag{26}$$

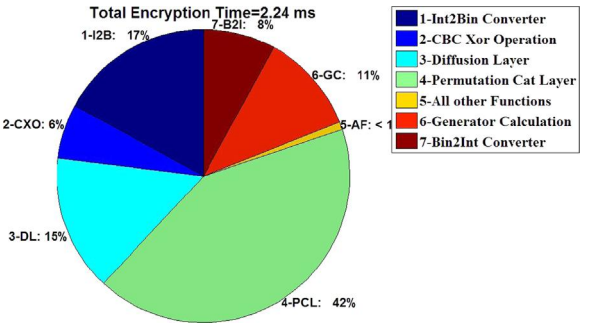


**Fig. 5.** Average encryption percentage time of each component of the proposed algorithm for lena.bmp of size $128 \times 128 \times 3$.

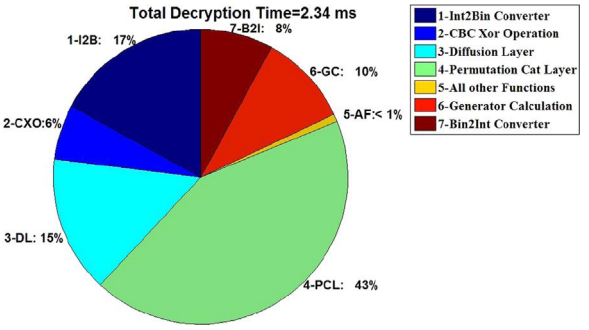


**Fig. 6.** Average decryption percentage time of each component of the proposed algorithm for lena.bmp of size $128 \times 128 \times 3$.

In (26), $r$ is the number of iterations for each block, and $rp$ is the number of rounds for each iteration $j$. So, the number of keys $Kp$ for one block of the cryptosystem is $Nk_p = r \times rp$.

The parameters of the cat map are in the range $0 \le u, v, rl, rc \le M-1$. Then, the necessary number of bits to represent each parameter is $q = \lceil \log_2(M) \rceil$, where $\lceil z \rceil$ denotes the ceil $(z)$. So, the size of the dynamic key for one round $(rp = 1)$, is $|Kp_(j,k)| = 4 \times q$; the size of the dynamic key for one iteration $(r = 1)$ is $|Kp_j| = 4 \times q \times rp$, and the size of the dynamic key, used to encrypt one block from the image, is $|Kp| = 4 \times q \times rp \times r$. The previous analysis is needed for the calculation of the necessary samples generated by the chaotic generator, for each iteration.

Example: $M = 16$, so $q = 4$ and $|Kp| = 16 \times rp \times r$. In our experiments, the required security level is reached for $rp = r = 1$ (see Fig. 7), then $|Kp| = 16$ bits. This means that one sample of the implemented chaotic generator (quantified on 32 bit) is sufficient to supply the dynamic keys for each block.

### 2.3. Description of the decryption process

The decryption process is almost similar to the encryption one. From Fig. 4, first of all, the bytes in a ciphered block are converted into a stream of bits using an Int2Bin() function. Then the inverse permutation layer is applied in reverse order from the last pair of bits into the first pair. The Bin2Int() function is used to convert the resultant bits from permutation layer into a stream of bytes. Finally, the inverse diffusion layer, based on the inverse matrix, is used.

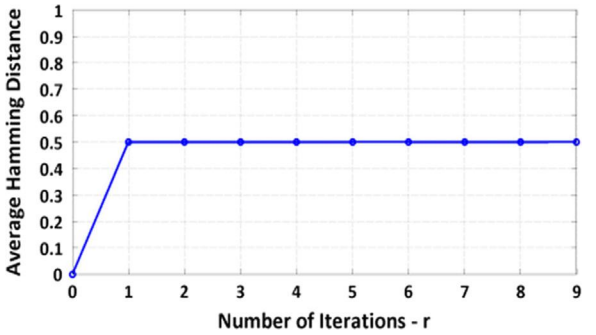


**Fig. 7.** Average Hamming distance versus number of iterations $r$.

The whole previous processes are repeated $r$ times. Furthermore, the decryption process uses the same chaotic generator of the encryption part.

#### 2.3.1. Reverse of the new formulation based on the modified 2-D cat map

The cat map is a non-invertible function because of the modulus operation, but it is a reversible one. For that, all the parameters of the dynamic keys $Kp = u, v, rl, rc$ must be generated and stored for all iterations and then used in reverse order to retrieve the original matrix. The reverse permutation based on the derived formulas, is implemented using the pseudo C code as in Algorithm 2. As we can see, the first two parts of the code are similar to those given by the permutation layer of Algorithm 1, while the last part (nested loops) is achieved in reverse order.

**Algorithm 2.** Algorithm in pseudo C code of the reverse operation of the new permutation formula.

```
for i=0 to M−1
  Mv[i] = i+1
End i
for k = r−1 to 0
for i=0 to M−1
  uMvm[i] = Mv[i] × u[k]+rl[k]+rc[k]
  vMv[i] = Mv[i] × v[k]
  vMvm2[i] = u[k] × vMv[i]+Mv[i]+rc[k]
End i
for i = M−1 to 0
  for j = M−1 to 0
    xrow = mod(Mv[i]+uMvm[j], M)
    ycol = mod(vMv[i]+vMvm2[j], M)
    data_bit_2[xrow, ycol] = data_bit_1[i, j]
  End j
End i
End k
```

#### 2.3.2. Inverse diffusion layer

The inverse diffusion layer is achieved by the inverse binary matrix as

$$\begin{bmatrix} O_0 \\ O_1 \\ \vdots \\ O_{31} \end{bmatrix} = [\mathbf{DM}]^{-1} \odot \begin{bmatrix} Od_0 \\ Od_1 \\ \vdots \\ Od_{31} \end{bmatrix} \tag{27}$$

where $Od_i, O_i \in [0, 255]$ and $[\mathbf{DM}]^{-1}$ is the inverse of the binary diffusion square matrix of size $32 \times 32$ [30]. We develop the calculation of the first byte $O_0$ from (27) as: $O_0 = Od_0 \oplus Od_8 \oplus Od_9 \oplus Od_{10} \oplus Od_{16} \oplus Od_{17} \oplus Od_{19} \oplus Od_{20} \oplus Od_{22} \oplus Od_{23} \oplus Od_{24} \oplus Od_{25} \oplus Od_{27} \oplus Od_{29} \oplus Od_{31}$.

## 3. Performance in terms of time consuming and security analysis

### 3.1. Time consuming

The proposed algorithm is tested using a C compiler, a PC with 3.1 GHz processor Intel Core ¢ i3-2100 CPU, 4 GB RAM, and Windows 7, 32-Bit Operation System. The proposed algorithm was applied to the Lena.bmp image of different sizes ($128 \times 128 \times 3$, $256 \times 256 \times 3$, $512 \times 512 \times 3$ and $1024 \times 1024 \times 1$). The average encryption/decryption times, the average encryption throughput and the average number of cycles per byte of the proposed algorithm, and some known chaos-based cryptosystems are given in Tables 3–5. The average time is calculated as follows: we executed our algorithm 100 times and for each time we computed the encryption/decryption times for the whole image, with $r=1$ and 2. Then, we calculated the average consuming time, the average encryption throughput and the average number of cycles per byte.

$$ET = \frac{Image_{Size}(MByte)}{Encryption_{Time}(second)} \tag{28}$$

$$\text{Number of cycles per Byte} = \frac{\text{CPUSpeed}}{ET} \tag{29}$$

The following tables present the simulation results of the above mentioned experiments, with $r_p = r_d = 1$. As we can see the proposed algorithm is faster than all cited algorithms in these tables, except Zhang et. al., algorithm [15], which has little bit better computational performance.

In Figs. 5 and 6, we also give the average time percentage of each cryptosystem component. The test was performed on the Lena image of size $128 \times 128 \times 3$ and $r=1$. The encryption/decryption times were calculated 1000 times, each time using a new random secret key, and then, we considered the average time percentage of each component. We can observe that the permutation layer consumes practically 43% of the encryption/decryption times. This result is valid for any value of $r$.

### 3.2. Security analysis

A cryptosystem should be computationally infeasible to break with available resources, either current or future. And this, even if the cryptanalyst has complete details on the cryptosystem and its implementation. According to Kerckhoffs, this means that the secrecy must reside entirely in the secret key. The attacker is assumed to have complete access to the communication between the sender and the receiver, and try to recover the plaintext without access to the secret key (by finding an equivalent algorithm to decrypt any new messages encrypted with the same key) or to recover the secret key. For that purpose, he can realize the following known types of attacks and cryptanalysis, ordered, for an attacker, from the hardest type to the easiest: (1) Ciphertext only: the attacker has the ciphertext of several

**Table 6**
NPCR and UACI for the plaintext sensitivity test.

| Algorithm | Image | Image size | UACI | NPCR |
|---|---|---|---|---|
| Proposed Algorithm | Lena | $512 \times 512 \times 3$ | 33.462 | 99.609 |
| ECKBA [21] | Lena | $512 \times 512 \times 3$ | 33.36 | 99.612 |
| Proposed Algorithm | Lena | $512 \times 512 \times 1$ | 33.463 | 99.607 |
| Ahmed et al. [37] | Lena | $512 \times 512 \times 1$ | 33.4 | 99.6 |
| Yang et al. [22] | Lena | $512 \times 512 \times 1$ | 33.479 | 99.618 |
| Wong et al. [32] | Lena | $512 \times 512 \times 1$ | 33.427 | 99.609 |
| Lian et al. [31] | Lena | $512 \times 512 \times 1$ | 33.419 | 99.587 |
| Proposed Algorithm | Barbara | $512 \times 512 \times 1$ | 33.463 | 99.607 |
| Chen et al. [38] | Barbara | $512 \times 512 \times 1$ | 25.200 | – |
| Proposed Algorithm | Barbara | $256 \times 256 \times 1$ | 33.452 | 99.597 |
| Behnia et al. [39] | Barbara | $256 \times 256 \times 1$ | 33.25 | 0.41962 |
| Proposed Algorithm | Boat | $256 \times 256 \times 1$ | 33.448 | 99.596 |
| Song et al. [40] | Boat | $256 \times 256 \times 1$ | 33.453 | 99.625 |
| Akhshani et al. [36] | Boat | $256 \times 256 \times 1$ | 33.200 | – |
| Zhang et al. [15] | Barbara | $512 \times 512 \times 1$ | 33.475 | 99.663 |

**Table 7**
NPCR and UACI for the key sensitivity test.

| Algorithm | Image | Image size | UACI | NPCR |
|---|---|---|---|---|
| Proposed Algorithm | Barbara | $512 \times 512$ | 33.46 | 99.609 |
| Chen et al. [38] | Barbara | $512 \times 512$ | – | 99.610 |
| Proposed Algorithm | Lena | $128 \times 128$ | 33.46 | 99.611 |
| Zhao et al (average) [41] | Lena | $128 \times 128$ | – | 99.568 |
| Proposed Algorithm | Airplane | $512 \times 512$ | 33.46 | 99.610 |
| Ahmed et al. [37] | Airplane | $512 \times 512$ | 33.41 | 99.598 |
| Proposed Algorithm | Lena | $256 \times 256$ | 33.46 | 99.609 |
| Song et al. [40] | Lena | $256 \times 256$ | – | 99.610 |
| Proposed Algorithm | Lena | $512 \times 512$ | 33.46 | 99.609 |
| Yang et al. [22] | Lena | $512 \times 512$ | – | 99.62 |

messages; (2) Known plaintext attack: the attacker has access to the ciphertext of several messages and their corresponding plaintext; (3) Chosen plaintext attack: the attacker has obtained temporary access to the encryption machinery, and then he can choose a specific plaintext to encrypt and obtain the corresponding ciphertext; (4) Chosen ciphertext attack: the attacker has obtained temporary access to the decryption machinery, and then he can choose a specific ciphertext to decrypt and obtain the corresponding plaintext [2,33]. The last two attacks are possible, when the cryptosystem, whose secret key is fixed by the manufacturer and unknown to the attacker, is embedded in a device: such as mobile phone SIM (subscriber identity module) cards; POST (point of sale terminals) machines; or web application session token encryption, which the attacker can freely manipulate, [34]. Besides, if a cryptosystem is able to resist chosen plaintext attack (in that case, it is also resistant to known plaintext and ciphertext only attacks) and chosen ciphertex attack, then it is computationally secure. The proposed cryptosystem can resist chosen plaintext attack, ciphertext attack, and brute-force attack. Indeed, it has two nonlinear components: the chaotic generator that produces keys dependent to supply the encryption primitives. Therefore, the relationship between the dynamic keys and the ciphertext is more complicated than a conventional key usage, because the permutation processes are dynamically changed for each round and for each block. This means also that, even if an adversary is able to crack a dynamic key, all

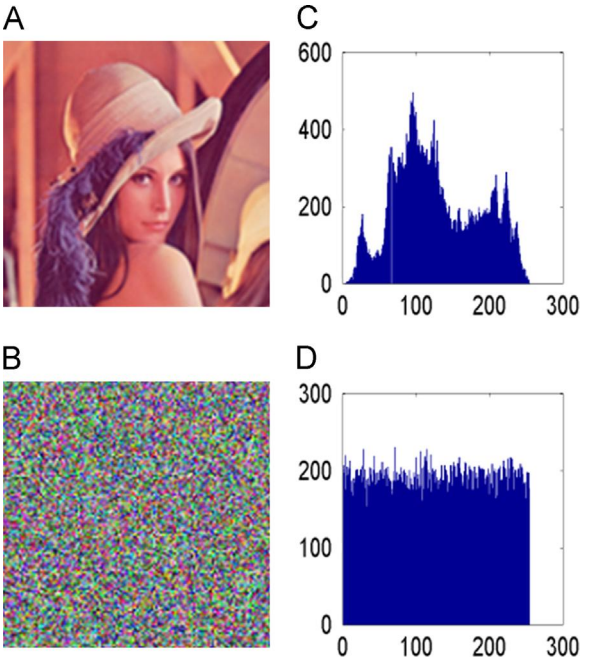


**Fig. 8.** Lena image and ciphered Lena and their Histograms. (a) Plain Lena image, (b) Ciphered Lena image, (c) Histogram of the plain Lena image and (d) Histogram of the ciphered Lena image.

**Table 8**
Chi-square test of histograms.

| Ciphered image | Lena | Baboon | Peppers |
|---|---|---|---|
| Chi-square | 252.1 | 256.1 | 253.5 |

other dynamic keys are still safe [35]. The second nonlinear component is the Int2Bin converter, it protects at least against the linear attack. In addition, with the influence of the other components (efficient binary diffusion, bit-permutation, Bin2Int converter), the proposed structure resists the chosen plaintext attack and the chosen ciphertext attack. Indeed, the very good obtained diffusion–confusion effects by our scheme, are demonstrated and explained below, by the plaintext sensitivity attack for the diffusion effect and by the histogram and correlation analysis for the confusion effect.

*3.2.1. Chosen plain-text sensitivity attack: diffusion property*

In order to resist the chosen plaintext attack and the differential attacks [36], any secure cryptosystem should be highly sensitive to one bit change in the plaintext. This property of diffusion assesses how a change in the plaintext affects the ciphered one. In a chosen plaintext attack, more than one plaintext (with one-bit changes between them) are selected to analyse the difference between their corresponding ciphertext. Our proposed cryptosystem achieves the avalanche effect and then, it overcomes the plaintext sensitivity attacks. This means that if we encrypt two plain-images $I_1$ and $I_2$, which differ only by one bit (randomly), the Hamming distance (in bits) between the corresponding ciphered images $C_1$ and $C_2$ must be close to 50% (probability

**Table 9**
Correlation coefficient values of two adjacent pixels in the plain and the cipher images.

| Algorithm name | Image name | Image size | H Plain | V Plain | D Plain | H Ciphered | V Ciphered | D Ciphered |
|---|---|---|---|---|---|---|---|---|
| Proposed Algorithm | Barbara | 512 Gray | 0.89538 | 0.95887 | 0.88304 | 0.00155 | 0.00163 | 0.001485 |
| Chen et al. [38] | Barbara | 512 Gray | 0.91765 | 0.95415 | 0.90205 | 0.01183 | 0.00016 | 0.01480 |
| Proposed Algorithm | Lena | 512 Gray | 0.98071 | 0.98214 | 0.96547 | 0.00131 | 0.00121 | 0.00117 |
| Proposed Algorithm | Lena | 128 Gray | 0.91019 | 0.96104 | 0.87166 | −0.00622 | 0.00611 | −0.006269 |
| Zhao et al. [41] | Lena | 128 Gray | 0.94800 | 0.88510 | 0.85460 | 0.02480 | −0.00940 | −0.01830 |
| Proposed Algorithm | Barbara | 256 Gray | 0.92287 | 0.95024 | 0.91523 | 0.00317 | −0.00326 | −0.00309 |
| Behnia et al. [39] | Barbara | 256 Gray | 0.95740 | 0.93990 | 0.91830 | 0.00380 | 0.00230 | 0.00040 |
| Proposed Algorithm | Lena | 256 Gray | 0.97165 | 0.98730 | 0.95440 | 0.00312 | −0.00317 | −0.00310 |
| Song et al. [40] | Lena | 256 Gray | 0.96592 | 0.94658 | 0.92305 | 0.00550 | 0.00411 | 0.00021 |
| Proposed Algorithm | Boat | 256 Gray | 0.94417 | 0.95263 | 0.90701 | 0.00320 | −0.00309 | −0.00306 |
| Akhshani et al. [36] | Boat | 256 Gray | 0.95160 | 0.94470 | 0.90590 | 0.00650 | 0.00550 | 0.00820 |
| Proposed Algorithm | Lena | 512 Color | 0.99233 | 0.99649 | 0.98712 | −0.00158 | 0.00159 | −0.00147 |
| Yang et al. [22] | Lena | 512 Color | 0.98022 | 0.98663 | 0.96468 | −0.00209 | −0.01618 | 0.01780 |
| Wong et al. [32] | Lena | 512 Color | 0.97510 | 0.98892 | 0.96704 | 0.00681 | 0.00782 | 0.00323 |
| Proposed Algorithm | Airplane | 512 Color | 0.96606 | 0.96384 | 0.93674 | 0.00157 | −0.00151 | −0.00158 |
| Ahmed et al. [37] | Airplane | 512 Color | 0.96775 | 0.95753 | 0.93002 | 0.00247 | 0.00182 | 0.00038 |
| Zhang et al. [15] | Barbara | 512 Gray | 0.86061 | 0.95982 | 0.87741 | −0.00824 | −0.00036 | 0.00128 |

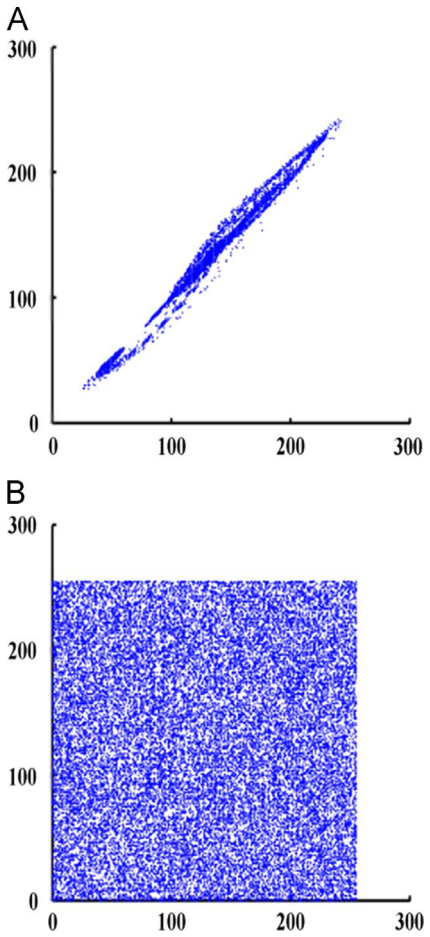


**Fig. 9.** Correlation Analysis of lena and it is ciphered image in three directions. (a) Horizontal correlation of the plain image, and (b) Horizontal correlation of the ciphered image.

of bit changes). Therefore, the previous attacks would become ineffective. This test gives also the minimum number of rounds $r$, needed to overcome the plaintext sensitivity attacks. To do this test we executed our proposed algorithm for 2500 random secret keys. For each execution, we calculated the Hamming distance value as a function of $r$, between the two ciphered images $C_1$ and $C_2$, which resulted from the two chosen plain-images $I_1$ and $I_2$:

$$d_{Hamming}(C_1, C_2) = \frac{1}{|Ib|}\sum_{K=1}^{|Ib|}(C_1[K] \oplus C_2[K]) \tag{30}$$

Fig. 7 shows the average values of the Hamming distance over the 2500 secret keys, versus the number of rounds $r$. The plain image under the test was lena.bmp image of size $|Ib| = 128 \times 128 \times 3 \times 8 = 393,216$ bits. We can observe that, with $r \geq 1$, the Hamming distance between $C_1$ and $C_2$ is 0.4999 and it is clear this value is very close to the optimal value of 50%. Hence, a high security level is reached.

The above test is also used to measure the resistance of the cryptosystem against the differential attacks introduced by Eli Biham and Adi Shamir [42]. In order to test the plaintext sensitivity and the key sensitivity attacks, researchers usually use two security analysis parameters: the Number of Pixels Change Rate (NPCR) and the Unified Average Changing Intensity (UACI). These parameters are calculated in bytes. A secure cryptosystem should have high sensitivity for one bit change in the secret key or in the plain image, in both processes of enciphering and deciphering. In particular, when an image is encrypted, a tiny change of the encryption key/plain image gives a completely different ciphered image. Therefore, when an image is decrypted, a tiny change of the decryption key can cause a failure of the deciphering. So, let us denote the ciphered-images, before and after one bit change in the plain-image, or one bit change of the encryption key, as $C_1$ and $C_2$, respectively. The NPCR is used to measure the number of elements that differ between two images, matrixes, or arrays. Let $C_1[i,j,p]$ and $C_2[i,j,p]$ be the $i$th row, the $j$th column, and the $p$th plane of the two ciphered images $C_1$ and $C_2$, respectively. The NPCR is defined as

below:

$$NPCR = \frac{1}{L \times C \times P} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} D[i,j,p] \times 100\% \quad (31)$$

$$D[i,j,p] = \begin{cases} 0, & \text{if } C_1[i,j,p] = C_2[i,j,p] \\ 1, & \text{if } C_1[i,j,p] \neq C_2[i,j,p] \end{cases} \quad (32)$$

The UACI measures the average intensity differences between the two ciphered images and is defined as

$$UACI = \frac{1}{L \times C \times P \times 255} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} |C_1 - C_2| \times 100\% \quad (33)$$

The optimal NPCR value is almost 99.61%, and the optimal UACI value is almost 33.46% [43,44].

Using different algorithms applied to different images, we almost made a complete comparison of NPCR and UACI values for the plaintext sensitivity test in Table 6, and for the key sensitivity test in Table 7 (all used image in the Table 7 are gray scale). As we can see from these tables, the obtained values of NPCR and UACI for the proposed algorithm are closer to the optimal values for $r=1$.

### 3.2.2. Statistical analysis: confusion property

Confusion serves to make the relationship between the plaintext, the ciphertext and the key as complex as possible. It measures how a change in the secret key affects the ciphered text [2]. Good confusion makes the relationship statistics so complicated that, even if a large number of plaintext–ciphertext pairs produced by the same encryption key are available for an attacker, it is very difficult to recover the secret key. This means also that the cryptosystem should be secure against known plaintext and ciphertext only [35]. To show the very good confusion property of the proposed cryptosystem, we realize two types of statistical analysis: histogram and correlation.

*Histogram analysis*: To resist the statistical attacks, at least the histogram of the encrypted image should be uniformly distributed. Fig. 8 shows (a) the plain-image, (b) the corresponding cipher image, (c) the histogram of the plain-image, and (d) its corresponding cipher image, after applying the proposed cryptosystem on the test image lena.bmp of size $512 \times 512 \times 3$. It can be observed that the histogram of the encrypted image is very close to a uniform distribution. To ensure this uniformity, we applied the chi-square test given below:

$$\chi^2_{exp} = \sum_{i=0}^{Nv-1} \frac{(o_i - e_i)^2}{e_i} \quad (34)$$

where $Nv$ is the number of levels (here 256), $o_i$ are the observed occurrence frequencies of each color level (0–255) in the histogram of the ciphered image, and $e_i$ is the expected occurrence frequency of the uniform distribution, given here by $e_i = (L \times C \times P)/256$. We present in Table 8 the calculation results of the Chi square test of the histogram for three ciphered images of different natures (i.e., Lena, Peppers, and Baboon), all of them having the size of $128 \times 128 \times 3$, with a significant level of 0.05. From the obtained values, we can observe that $x^2_{exp} < x^2_{th}(255, 0.05) = 293$. So, it is clear that the distribution of the tested histograms is uniform and does not reveal any information for the statistical analysis.

*Correlation analysis*: As a general requirement for all image encryption schemes, the encrypted image should be greatly different from its original form. The correlation analysis is one of the usual ways to measure this property. Indeed, it is well-known that adjacent pixels in the plain images are very redundant and correlated.

To test the correlation between two adjacent pixels, the following procedure was carried out. Firstly, 80,000 pairs of two adjacent pixels are selected randomly in vertical, horizontal, and diagonal directions from the original and the encrypted images. And then, the correlation coefficient was computed according to the following formulas:

$$\rho_{xy} = \frac{\sum_{i=1}^{N} ([x_i - \bar{x}][y_i - \bar{y}])}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}} \quad (35)$$

In (35), the following notations are used: $N=80{,}000$ is the sample size, while $x$ and $y$ are the gray-level values of the two adjacent pixels in the image. The obtained results are shown in Table 9 and Fig. 9 indicate that the correlation coefficient, in all directions, of plaintext images is close to one, and the correlation coefficient of the encrypted images is close to zero. This means that no detectable correlation exists between the original and its corresponding ciphered image. All tested algorithms give approximately the same results.

The proposed nonlinear structure offers an efficient encrypted data: robustness against the cryptanalysis (known plaintext, chosen plaintext, chosen ciphertext attacks) and the high speed of calculation. Moreover, it is adequate for a software and hardware implementations (FPGA card or an ASIC).

## 4. Conclusion and future work

We proposed a new fast, simple, and robust structure of a chaos-based cryptosystem for secure image transmissions. The proposed structure is formed by a diffusion layer achieved by a binary matrix of size 32 by 32, followed by a bit permutation layer, achieved by a new formulation of the modified 2-D cat map. The new formulation of the modified 2-D cat map achieves a high performance in comparison to the standard 2-D cat map. Also, the proposed chaos-based cryptosystem is faster than many chaos-based algorithms except Zhang et al. [15], however, their scheme needs for each new plain image a new encryption key, and then repeat the protocol to shared the encryption key. The security analysis of the obtained experimental results shows that the proposed cryptosystem is resistant to all attacks identified in the literature. One of the striking results is that the proposed cryptosystem requires only one iteration of permutation to provide extreme sensitivity to one bit change in the plain text and in the secret key. The correlation values between adjacent pixels in the ciphered image are almost zero. Finally, the proposed algorithm is also suitable for hardware implementation. Our future works will focus on real time selective video encryption using a chaotic system for the scalable HEVC (High Efficiency Video Coding).

## Acknowledgment

This work is supported by the European Celtic-Plus project 4KREPROSYS - 4 K ultraHD TV wireless REmote PROduction SYStems, 2014–2017.

## Appendix A. The mathematical proof of the weakness for algorithms that use self invertible matrix during the encryption process

Let us denote by $P$ the self invertible matrix (i.e., $P=P^{-1}$), $Y$ is the output vector of the diffusion process, and $X$ is the input vector of the diffusion process. Then, applying the diffusion process more than one time can be written as

$$Y_1 = P \times X$$
$$Y_2 = P \times Y_1$$
$$Y_3 = P \times Y_2$$
$$\vdots$$
$$Y_n = P \times Y_{n-1}$$

but

$$Y_n = P \times P \times Y_{n-2}$$
$$Y_n = P \times P \times P \times Y_{n-3}$$
$$Y_n = P \times P \times P \times Y_{n-4}$$
$$Y_n = \overbrace{P \times P \times \cdots P}^{n-1 \ times} \times Y_1$$

but

$$Y_1 = P \times X$$

then

$$Y_n = \overbrace{P \times P \times \cdots P}^{n \ times} \times X$$
$$Y_n = P^n \times X$$

but

$P^n = P$ if $n$ odd
$P^n = I$ if $n$ even

where $I$ is the identity matrix and So, if $n$ is even $Y_n = I \times X = X$ else if $n$ is odd then $Y_n = I \times P \times X = P \times X$.

## Appendix B. Diffusion matrix and its inverse

In the encryption part, the calculations of $Y_{(i)}, i = 0, 1, 31$ are given by

$$Y_0 = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_8 \oplus X_9 \oplus X_{10} \oplus X_{12} \oplus X_{13} \oplus X_{17} \oplus X_{18} \oplus X_{19} \oplus X_{24} \oplus X_{25} \oplus X_{29} \oplus X_{31}$$
$$Y_1 = X_0 \oplus X_1 \oplus X_5 \oplus X_6 \oplus X_9 \oplus X_{10} \oplus X_{11} \oplus X_{13} \oplus X_{14} \oplus X_{16} \oplus X_{18} \oplus X_{19} \oplus X_{25} \oplus X_{26} \oplus X_{28} \oplus X_{30}$$
$$Y_2 = X_0 \oplus X_1 \oplus X_2 \oplus X_6 \oplus X_7 \oplus X_8 \oplus X_{10} \oplus X_{11} \oplus X_{14} \oplus X_{15} \oplus X_{16} \oplus X_{17} \oplus X_{19} \oplus X_{26} \oplus X_{27} \oplus X_{29} \oplus X_{31}$$
$$Y_3 = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_7 \oplus X_8 \oplus X_9 \oplus X_{11} \oplus X_{12} \oplus X_{15} \oplus X_{16} \oplus X_{17} \oplus X_{18} \oplus X_{24} \oplus X_{27} \oplus X_{28} \oplus X_{30}$$
$$Y_4 = X_0 \oplus X_1 \oplus X_7 \oplus X_9 \oplus X_{12} \oplus X_{14} \oplus X_{15} \oplus X_{20} \oplus X_{22} \oplus X_{27} \oplus X_{28}$$
$$Y_5 = X_1 \oplus X_2 \oplus X_4 \oplus X_{10} \oplus X_{12} \oplus X_{13} \oplus X_{15} \oplus X_{21} \oplus X_{23} \oplus X_{24} \oplus X_{29}$$
$$Y_6 = X_2 \oplus X_3 \oplus X_5 \oplus X_{11} \oplus X_{12} \oplus X_{13} \oplus X_{14} \oplus X_{20} \oplus X_{22} \oplus X_{25} \oplus X_{30}$$
$$Y_7 = X_0 \oplus X_3 \oplus X_6 \oplus X_8 \oplus X_{13} \oplus X_{14} \oplus X_{15} \oplus X_{21} \oplus X_{23} \oplus X_{26} \oplus X_{31}$$
$$Y_8 = X_0 \oplus X_1 \oplus X_2 \oplus X_5 \oplus X_8 \oplus X_9 \oplus X_{10} \oplus X_{12} \oplus X_{15} \oplus X_{17} \oplus X_{25} \oplus X_{26} \oplus X_{27}$$
$$Y_9 = X_1 \oplus X_2 \oplus X_3 \oplus X_6 \oplus X_9 \oplus X_{10} \oplus X_{11} \oplus X_{12} \oplus X_{13} \oplus X_{18} \oplus X_{24} \oplus X_{26} \oplus X_{27}$$
$$Y_{10} = X_0 \oplus X_2 \oplus X_3 \oplus X_7 \oplus X_8 \oplus X_{10} \oplus X_{11} \oplus X_{13} \oplus X_{14} \oplus X_{19} \oplus X_{24} \oplus X_{25} \oplus X_{27}$$
$$Y_{11} = X_0 \oplus X_1 \oplus X_3 \oplus X_4 \oplus X_8 \oplus X_9 \oplus X_{11} \oplus X_{14} \oplus X_{15} \oplus X_{16} \oplus X_{24} \oplus X_{25} \oplus X_{26}$$
$$Y_{12} = X_0 \oplus X_1 \oplus X_4 \oplus X_6 \oplus X_7 \oplus X_8 \oplus X_{11} \oplus X_{13} \oplus X_{14} \oplus X_{15} \oplus X_{16} \oplus X_{17} \oplus X_{20} \oplus X_{21} \oplus X_{28} \oplus X_{29} \oplus X_{30}$$
$$Y_{13} = X_1 \oplus X_2 \oplus X_4 \oplus X_5 \oplus X_7 \oplus X_8 \oplus X_9 \oplus X_{12} \oplus X_{14} \oplus X_{15} \oplus X_{17} \oplus X_{18} \oplus X_{21} \oplus X_{22} \oplus X_{29} \oplus X_{30} \oplus X_{31}$$
$$Y_{14} = X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_9 \oplus X_{10} \oplus X_{12} \oplus X_{13} \oplus X_{15} \oplus X_{18} \oplus X_{19} \oplus X_{22} \oplus X_{23} \oplus X_{28} \oplus X_{30} \oplus X_{31}$$
$$Y_{15} = X_0 \oplus X_3 \oplus X_5 \oplus X_6 \oplus X_7 \oplus X_{10} \oplus X_{11} \oplus X_{12} \oplus X_{13} \oplus X_{14} \oplus X_{16} \oplus X_{19} \oplus X_{20} \oplus X_{23} \oplus X_{28} \oplus X_{29} \oplus X_{31}$$
$$Y_{16} = X_1 \oplus X_2 \oplus X_3 \oplus X_9 \oplus X_{12} \oplus X_{13} \oplus X_{19} \oplus X_{23} \oplus X_{25} \oplus X_{26} \oplus X_{28}$$
$$Y_{17} = X_0 \oplus X_2 \oplus X_3 \oplus X_{10} \oplus X_{13} \oplus X_{14} \oplus X_{16} \oplus X_{20} \oplus X_{26} \oplus X_{27} \oplus X_{29}$$
$$Y_{18} = X_0 \oplus X_1 \oplus X_3 \oplus X_{11} \oplus X_{14} \oplus X_{15} \oplus X_{17} \oplus X_{21} \oplus X_{24} \oplus X_{27} \oplus X_{30}$$
$$Y_{19} = X_0 \oplus X_1 \oplus X_2 \oplus X_8 \oplus X_{12} \oplus X_{15} \oplus X_{18} \oplus X_{22} \oplus X_{24} \oplus X_{25} \oplus X_{31}$$
$$Y_{20} = X_4 \oplus X_6 \oplus X_{12} \oplus X_{13} \oplus X_{19} \oplus X_{20} \oplus X_{22} \oplus X_{23} \oplus X_{24} \oplus X_{25} \oplus X_{28} \oplus X_{30} \oplus X_{31}$$
$$Y_{21} = X_5 \oplus X_7 \oplus X_{13} \oplus X_{14} \oplus X_{16} \oplus X_{20} \oplus X_{21} \oplus X_{23} \oplus X_{25} \oplus X_{26} \oplus X_{28} \oplus X_{29} \oplus X_{31}$$
$$Y_{22} = X_4 \oplus X_6 \oplus X_{14} \oplus X_{15} \oplus X_{17} \oplus X_{20} \oplus X_{21} \oplus X_{22} \oplus X_{26} \oplus X_{27} \oplus X_{28} \oplus X_{29} \oplus X_{30}$$
$$Y_{23} = X_5 \oplus X_7 \oplus X_{12} \oplus X_{15} \oplus X_{18} \oplus X_{21} \oplus X_{22} \oplus X_{23} \oplus X_{24} \oplus X_{27} \oplus X_{29} \oplus X_{30} \oplus X_{31}$$
$$Y_{24} = X_0 \oplus X_1 \oplus X_7 \oplus X_9 \oplus X_{10} \oplus X_{11} \oplus X_{17} \oplus X_{18} \oplus X_{20} \oplus X_{21} \oplus X_{25} \oplus X_{26} \oplus X_{27}$$
$$Y_{25} = X_1 \oplus X_2 \oplus X_4 \oplus X_8 \oplus X_{10} \oplus X_{11} \oplus X_{18} \oplus X_{19} \oplus X_{21} \oplus X_{22} \oplus X_{24} \oplus X_{26} \oplus X_{27}$$
$$Y_{26} = X_2 \oplus X_3 \oplus X_5 \oplus X_8 \oplus X_9 \oplus X_{11} \oplus X_{16} \oplus X_{19} \oplus X_{22} \oplus X_{23} \oplus X_{24} \oplus X_{25} \oplus X_{27}$$
$$Y_{27} = X_0 \oplus X_3 \oplus X_6 \oplus X_8 \oplus X_9 \oplus X_{10} \oplus X_{16} \oplus X_{17} \oplus X_{20} \oplus X_{23} \oplus X_{24} \oplus X_{25} \oplus X_{26}$$
$$Y_{28} = X_1 \oplus X_3 \oplus X_4 \oplus X_{12} \oplus X_{13} \oplus X_{14} \oplus X_{16} \oplus X_{20} \oplus X_{22} \oplus X_{23} \oplus X_{28} \oplus X_{29} \oplus X_{30}$$
$$Y_{29} = X_0 \oplus X_2 \oplus X_5 \oplus X_{13} \oplus X_{14} \oplus X_{15} \oplus X_{17} \oplus X_{20} \oplus X_{21} \oplus X_{23} \oplus X_{29} \oplus X_{30} \oplus X_{31}$$
$$Y_{30} = X_1 \oplus X_3 \oplus X_6 \oplus X_{12} \oplus X_{14} \oplus X_{15} \oplus X_{18} \oplus X_{20} \oplus X_{21} \oplus X_{22} \oplus X_{28} \oplus X_{30} \oplus X_{31}$$
$$Y_{31} = X_0 \oplus X_2 \oplus X_7 \oplus X_{12} \oplus X_{13} \oplus X_{15} \oplus X_{19} \oplus X_{21} \oplus X_{22} \oplus X_{23} \oplus X_{28} \oplus X_{29} \oplus X_{31}$$

In the decryption part, the calculations of $X_{(i)}, i = 0, 1, 31$ are given by

$$X_0 = Y_0 \oplus Y_8 \oplus Y_9 \oplus Y_{10} \oplus Y_{16} \oplus Y_{17} \oplus Y_{19} \oplus Y_{20} \oplus Y_{22} \oplus Y_{23} \oplus Y_{24} \oplus Y_{25} \oplus Y_{27} \oplus Y_{29} \oplus Y_{31}$$
$$X_1 = Y_1 \oplus Y_9 \oplus Y_{10} \oplus Y_{11} \oplus Y_{16} \oplus Y_{17} \oplus Y_{18} \oplus Y_{20} \oplus Y_{21} \oplus Y_{23} \oplus Y_{24} \oplus Y_{25} \oplus Y_{26} \oplus Y_{28} \oplus Y_{30}$$
$$X_2 = Y_2 \oplus Y_8 \oplus Y_{10} \oplus Y_{11} \oplus Y_{17} \oplus Y_{18} \oplus Y_{19} \oplus Y_{20} \oplus Y_{21} \oplus Y_{22} \oplus Y_{25} \oplus Y_{26} \oplus Y_{27} \oplus Y_{29} \oplus Y_{31}$$
$$X_3 = Y_3 \oplus Y_8 \oplus Y_9 \oplus Y_{11} \oplus Y_{16} \oplus Y_{18} \oplus Y_{19} \oplus Y_{21} \oplus Y_{22} \oplus Y_{23} \oplus Y_{24} \oplus Y_{26} \oplus Y_{27} \oplus Y_{28} \oplus Y_{30}$$
$$X_4 = Y_5 \oplus Y_{12} \oplus Y_{13} \oplus Y_{15} \oplus Y_{16} \oplus Y_{19} \oplus Y_{22} \oplus Y_{23} \oplus Y_{25} \oplus Y_{26} \oplus Y_{27}$$
$$X_5 = Y_6 \oplus Y_{12} \oplus Y_{13} \oplus Y_{14} \oplus Y_{16} \oplus Y_{17} \oplus Y_{20} \oplus Y_{23} \oplus Y_{24} \oplus Y_{26} \oplus Y_{27}$$
$$X_6 = Y_7 \oplus Y_{13} \oplus Y_{14} \oplus Y_{15} \oplus Y_{17} \oplus Y_{18} \oplus Y_{20} \oplus Y_{21} \oplus Y_{24} \oplus Y_{25} \oplus Y_{27}$$
$$X_7 = Y_4 \oplus Y_{12} \oplus Y_{14} \oplus Y_{15} \oplus Y_{18} \oplus Y_{19} \oplus Y_{21} \oplus Y_{22} \oplus Y_{24} \oplus Y_{25} \oplus Y_{26}$$
$$X_8 = Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_{10} \oplus Y_{12} \oplus Y_{13} \oplus Y_{24} \oplus Y_{26} \oplus Y_{27} \oplus Y_{28} \oplus Y_{31}$$
$$X_9 = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_{11} \oplus Y_{13} \oplus Y_{14} \oplus Y_{24} \oplus Y_{25} \oplus Y_{27} \oplus Y_{28} \oplus Y_{29}$$
$$X_{10} = Y_0 \oplus Y_2 \oplus Y_3 \oplus Y_8 \oplus Y_{14} \oplus Y_{15} \oplus Y_{24} \oplus Y_{25} \oplus Y_{26} \oplus Y_{29} \oplus Y_{30}$$
$$X_{11} = Y_0 \oplus Y_1 \oplus Y_3 \oplus Y_9 \oplus Y_{12} \oplus Y_{15} \oplus Y_{25} \oplus Y_{26} \oplus Y_{27} \oplus Y_{30} \oplus Y_{31}$$
$$X_{12} = Y_4 \oplus Y_5 \oplus Y_7 \oplus Y_8 \oplus Y_9 \oplus Y_{15} \oplus Y_{20} \oplus Y_{21} \oplus Y_{23} \oplus Y_{24} \oplus Y_{26} \oplus Y_{27} \oplus Y_{29} \oplus Y_{30} \oplus Y_{31}$$
$$X_{13} = Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_9 \oplus Y_{10} \oplus Y_{12} \oplus Y_{20} \oplus Y_{21} \oplus Y_{22} \oplus Y_{24} \oplus Y_{25} \oplus Y_{27} \oplus Y_{28} \oplus Y_{30} \oplus Y_{31}$$
$$X_{14} = Y_5 \oplus Y_6 \oplus Y_7 \oplus Y_{10} \oplus Y_{11} \oplus Y_{13} \oplus Y_{21} \oplus Y_{22} \oplus Y_{23} \oplus Y_{24} \oplus Y_{25} \oplus Y_{26} \oplus Y_{28} \oplus Y_{29} \oplus Y_{31}$$
$$X_{15} = Y_4 \oplus Y_6 \oplus Y_7 \oplus Y_8 \oplus Y_{11} \oplus Y_{14} \oplus Y_{20} \oplus Y_{22} \oplus Y_{23} \oplus Y_{25} \oplus Y_{26} \oplus Y_{27} \oplus Y_{28} \oplus Y_{29} \oplus Y_{30}$$
$$X_{16} = Y_0 \oplus Y_1 \oplus Y_3 \oplus Y_4 \oplus Y_7 \oplus Y_{17} \oplus Y_{20} \oplus Y_{21} \oplus Y_{22} \oplus Y_{24} \oplus Y_{26}$$
$$X_{17} = Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_4 \oplus Y_5 \oplus Y_{18} \oplus Y_{21} \oplus Y_{22} \oplus Y_{23} \oplus Y_{25} \oplus Y_{27}$$
$$X_{18} = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_5 \oplus Y_6 \oplus Y_{19} \oplus Y_{20} \oplus Y_{22} \oplus Y_{23} \oplus Y_{24} \oplus Y_{26}$$
$$X_{19} = Y_0 \oplus Y_2 \oplus Y_3 \oplus Y_6 \oplus Y_7 \oplus Y_{16} \oplus Y_{20} \oplus Y_{21} \oplus Y_{23} \oplus Y_{25} \oplus Y_{27}$$
$$X_{20} = Y_0 \oplus Y_2 \oplus Y_3 \oplus Y_6 \oplus Y_7 \oplus Y_{12} \oplus Y_{13} \oplus Y_{15} \oplus Y_{16} \oplus Y_{17} \oplus Y_{18} \oplus Y_{20} \oplus Y_{28} \oplus Y_{30} \oplus Y_{31}$$
$$X_{21} = Y_0 \oplus Y_1 \oplus Y_3 \oplus Y_4 \oplus Y_7 \oplus Y_{12} \oplus Y_{13} \oplus Y_{14} \oplus Y_{17} \oplus Y_{18} \oplus Y_{19} \oplus Y_{21} \oplus Y_{28} \oplus Y_{29} \oplus Y_{31}$$
$$X_{22} = Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_4 \oplus Y_5 \oplus Y_{13} \oplus Y_{14} \oplus Y_{15} \oplus Y_{16} \oplus Y_{18} \oplus Y_{19} \oplus Y_{22} \oplus Y_{28} \oplus Y_{29} \oplus Y_{30}$$
$$X_{23} = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_5 \oplus Y_6 \oplus Y_{12} \oplus Y_{14} \oplus Y_{15} \oplus Y_{16} \oplus Y_{17} \oplus Y_{19} \oplus Y_{23} \oplus Y_{29} \oplus Y_{30} \oplus Y_{31}$$
$$X_{24} = Y_0 \oplus Y_1 \oplus Y_3 \oplus Y_5 \oplus Y_6 \oplus Y_7 \oplus Y_8 \oplus Y_{10} \oplus Y_{11} \oplus Y_{12} \oplus Y_{14} \oplus Y_{15} \oplus Y_{16} \oplus Y_{18} \oplus Y_{27}$$
$$X_{25} = Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_4 \oplus Y_6 \oplus Y_7 \oplus Y_8 \oplus Y_9 \oplus Y_{11} \oplus Y_{12} \oplus Y_{13} \oplus Y_{15} \oplus Y_{17} \oplus Y_{19} \oplus Y_{24}$$
$$X_{26} = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_7 \oplus Y_8 \oplus Y_9 \oplus Y_{10} \oplus Y_{12} \oplus Y_{13} \oplus Y_{14} \oplus Y_{16} \oplus Y_{18} \oplus Y_{25}$$
$$X_{27} = Y_0 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_9 \oplus Y_{10} \oplus Y_{11} \oplus Y_{13} \oplus Y_{14} \oplus Y_{15} \oplus Y_{17} \oplus Y_{19} \oplus Y_{26}$$
$$X_{28} = Y_1 \oplus Y_3 \oplus Y_8 \oplus Y_{11} \oplus Y_{13} \oplus Y_{14} \oplus Y_{15} \oplus Y_{20} \oplus Y_{22} \oplus Y_{23} \oplus Y_{30}$$
$$X_{29} = Y_0 \oplus Y_2 \oplus Y_8 \oplus Y_9 \oplus Y_{12} \oplus Y_{14} \oplus Y_{15} \oplus Y_{20} \oplus Y_{21} \oplus Y_{23} \oplus Y_{31}$$
$$X_{30} = Y_1 \oplus Y_3 \oplus Y_9 \oplus Y_{10} \oplus Y_{12} \oplus Y_{13} \oplus Y_{15} \oplus Y_{20} \oplus Y_{21} \oplus Y_{22} \oplus Y_{28}$$
$$X_{31} = Y_0 \oplus Y_2 \oplus Y_{10} \oplus Y_{11} \oplus Y_{12} \oplus Y_{13} \oplus Y_{14} \oplus Y_{21} \oplus Y_{22} \oplus Y_{23} \oplus Y_{29}$$

## References


[1] D. Schonberg, S.C. Draper, C. Yeo, K. Ramchandran, Toward compression of encrypted images and video sequences, IEEE Trans. Inf. Forensics Secur. 3 (4) (2008) 749–762.

[2] B. Schneier, Applied cryptography. Protocols, algorithms, and source code in c/bruce schneier, 1996.

[3] R.L. Rivest, M.J. Robshaw, R. Sidney, Y.L. Yin, The rc6 block cipher, in: In First Advanced Encryption Standard (AES) Conference, Citeseer, 1998.

[4] M. François, T. Grosges, D. Barchiesi, R. Erra, A new image encryption scheme based on a chaotic function, Signal Process.: Image Commun. 27 (3) (2012) 249–259.

[5] F. Peng, X.-w. Zhu, M. Long, A roi privacy protection scheme for h. 264 video based on fmo and chaos, IEEE Trans. Inf. Forensics Secur. 8 (10) (2013) 1688–1699.

[6] W. Amidouche, M. Farajallah, M. Raulet, O. Deforges, S. El-Assad, Selective video encryption using chaotic system in the shvc extension, in: In 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP, Brisbane Australia), IEEE, 2015.

[7] S. Rakesh, A.A. Kaller, B. Shadakshari, B. Annappa, Image encryption using block based uniform scrambling and chaotic logistic mapping, Int. J. Cryptogr. Inf. Secur. (IJCIS) 2 (1) (2012) 49–57.

[8] C.K. Volos, I.M. Kyprianidis, I.N. Stouboulos, Image encryption process based on chaotic synchronization phenomena, Signal Process.

[9] M. Baptista, Cryptography with chaos, Phys. Lett. A 240 (1) (1998) 50–54.

[10] M. Farajallah, S. El Assad, M. Chetto, Dynamic adjustment of the chaos-based security in real-time energy harvesting sensors, in: Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, 2013, pp. 282–289. http://dx.doi.org/10.1109/GreenCom-iThings-CPSCom.2013.65.

[11] S. Al-Maadeed, A. Al-Ali, T. Abdalla, A new chaos-based image-encryption and compression algorithm, J. Electr. Comput. Eng. 2012 (2012) 15.

[12] R. Ye, H. Zhao, An efficient chaos-based image encryption scheme using affine modular maps, Int. J. Comput. Netw. Inf. Secur. (IJCNIS) 4 (7) (2012) 41.

[13] Y. Wang, J. Wang, A new image encryption algorithm based on compound chaotic sequence, in: 2012 International Conference on Measurement, Information and Control (MIC), vol. 2, IEEE, 2012, pp. 962–966.

[14] M. Farajallah, Z. Fawaz, S. El Assad, O. Deforges, Efficient image encryption and authentication scheme based on chaotic sequences, in: SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies, 2013, pp. 150–155.

[15] X. Zhang, Z. Zhao, J. Wang, Chaotic image encryption based on circular substitution box and key stream buffer, Signal Process.: Image Commun. 29 (8) (2014) 902–913.

[16] Y. Zhang, D. Xiao, Y. Shu, J. Li, A novel image encryption scheme based on a linear hyperbolic chaotic system of partial differential equations, Signal Process.: Image Commun. 28 (3) (2013) 292–300.

[17] X. Tong, Y. Liu, M. Zhang, Y. Chen, A novel chaos-based fragile watermarking for image tampering detection and self-recovery, Signal Process.: Image Commun. 28 (3) (2013) 301–308.

[18] H. Zhu, C. Zhao, X. Zhang, A novel image encryption—compression scheme using hyper-chaos and chinese remainder theorem, Signal Process.: Image Commun. 28 (6) (2013) 670–680.

[19] J. Fridrich, Symmetric ciphers based on two-dimensional chaotic maps, Int. J. Bifurc. Chaos 8 (06) (1998) 1259–1284.

[20] N. Masuda, G. Jakimoski, K. Aihara, L. Kocarev, Chaotic block ciphers: from theory to practical algorithms, IEEE Trans. Circuits Syst. I: Regul. Pap. 53 (6) (2006) 1341–1352.
[21] D. Socek, S. Li, S.S. Maglivera, B. Furht, Short paper: enhanced 1-d chaotic key based algorithm for image encryption, September, 2005.
[22] H. Yang, K.-W. Wong, X. Liao, W. Zhang, P. Wei, A fast image encryption and authentication scheme based on chaotic maps, Commun. Nonlinear Sci. Numer. Simul. 15 (11) (2010) 3507–3517.
[23] F. Chen, K.-W. Wong, X. Liao, T. Xiang, Period distribution of generalized discrete arnold cat map for, IEEE Trans. Inf. Theory 58 (1) (2012) 445–452.
[24] B.W. Koo, H.S. Jang, J.H. Song, On constructing of a $32 \times 32$ binary matrix as a diffusion layer for a 256-bit block cipher, in: Information Security and Cryptology–ICISC 2006, Springer, 2006, pp. 51–64.
[25] R.L. Tataru, D. Battikh, S.E. Assad, H. Noura, O. Déforges, Enhanced adaptive data hiding in spatial lsb domain by using chaotic sequences, in: 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), IEEE, 2012, pp. 85–88.
[26] A. Fog, Lists of Instruction Latencies, Throughputs and Micro-Operation Breakdowns for Intel, amd and via cpus, Copenhagen University College of Engineering, 2012.
[27] S. EL ASSAD, H. NOURA, Generator of chaotic sequences and corresponding generating system, wO Patent 2,011,121,218 October 7, 2011.
[28] S. Lian, J. Sun, J. Wang, Z. Wang, A chaotic stream cipher and the usage in video protection, Chaos, Solitons & Fractals 34 (3) (2007) 851–859.
[29] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, J. Dray, S. Vo, et al., Statistical test suite for random and pseudorandom number generators for cryptographic applications, nist special publication.
[30] R. Li, B. Sun, C. Li, Impossible differential cryptanalysis of spn ciphers, IET Inf. Secur. 5 (2) (2011) 111–120.
[31] S. Lian, J. Sun, Z. Wang, A block cipher based on a suitable use of the chaotic standard map, Chaos, Solitons & Fractals 26 (1) (2005) 117–129.
[32] K.-W. Wong, B.S.-H. Kwok, W.-S. Law, A fast image encryption scheme based on chaotic standard map, Phys. Lett. A 372 (15) (2008) 2645–2652.
[33] D.R. Stinson, Cryptography Theory and Practice, Chapman Hall/CRC, 2006.
[34] G. Alvarez, S. Li, Some basic cryptographic requirements for chaos-based cryptosystems, Int. J. Bifurc. Chaos 16 (08) (2006) 2129–2151.
[35] Y. Wu, Y. Zhou, J.P. Noonan, S. Agaian, Design of image cipher using latin squares, Inf. Sci. 264 (2014) 317–339.
[36] A. Akhshani, A. Akhavan, S.-C. Lim, Z. Hassan, An image encryption scheme based on quantum logistic map, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4653–4661.
[37] A.A. Abd El-Latif, X. Niu, M. Amin, A new image cipher in time and frequency domains, Optics Commun.
[38] G. Chen, Y. Mao, C.K. Chui, A symmetric image encryption scheme based on 3d chaotic cat maps, Chaos, Solitons & Fractals 21 (3) (2004) 749–761.
[39] S. Behnia, A. Akhshani, H. Mahmodi, A. Akhavan, A novel algorithm for image encryption based on mixture of chaotic maps, Chaos, Solitons & Fractals 35 (2) (2008) 408–419.
[40] C.-Y. Song, Y.-L. Qiao, X.-Z. Zhang, An image encryption scheme based on new spatiotemporal chaos, Opt.—Int. J. Light Electron Opt.
[41] L. Zhao, A. Adhikari, D. Xiao, K. Sakurai, On the security analysis of an image scrambling encryption of pixel bit and its improved scheme based on self-correlation encryption, Commun. Nonlinear Sci. Numer. Simul. 17 (8) (2012) 3303–3327.
[42] E. Biham, A. Shamir, Differential cryptanalysis of des-like cryptosystems, J. CRYPTOL. 4 (1) (1991) 3–72.
[43] Y. Wu, J.P. Noonan, S. Agaian, Npcr and uaci randomness tests for image encryption, Cyber J.: Multidiscip. J. Sci. Technol., J. Sel. Areas Telecommun. (JSAT), 2011, pp. 31–38.
[44] F. Maleki, A. Mohades, S.M. Hashemi, M.E. Shiri, An image encryption system by cellular automata with memory, in: Third International Conference on Availability, Reliability and Security, 2008. ARES 08, IEEE, 2008, pp. 1266–1271.