IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 39, No. 5, October, 2022, pp. 1453-1462
Journal homepage: http://iieta.org/journals/ts

A New Encryption Algorithm Based on Fibonacci Polynomials and Matrices

Orhan Dişkaya[1], Erdinç Avaroğlu[2*], Hamza Menken[1], Ahmet Emsal[2]

[1] Department of Mathematics, Faculty of Science, Mersin University, Mersin 33110, Turkey
[2] Computer Engineering Department, Faculty of Engineering, Mersin University, Mersin 33110, Turkey

Corresponding Author Email: eavaroglu@mersin.edu.tr

https://doi.org/10.18280/ts.390501

Received: 29 June 2022
Accepted: 2 October 2022

Keywords:
cryptography, Fibonacci polynomial matrix, image encryption, histogram analysis, differential attack

ABSTRACT

Confusion and diffusion features are two fundamental needs of encoded text or images. These features have been used in various encryption algorithms such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES). The AES adopts the S-box table formed with irreducible polynomials, while the DES employs the Feistel and S-box structures. This study proposes a new encryption algorithm based on Fibonacci polynomials and matrices, which meets the fundamental needs of image encryption and provides an alternative to other encryption algorithms. The success of the proposed method was tested on three different images, as evidenced by the histogram analysis results of the sample images, together with the number of changing pixel rate (NPCR) and the unified averaged changed intensity (UACI). In addition, the root mean squared error (RMSE) suggests that the decoded images are consistent with the original images. It can therefore be summarized that the proposed encryption algorithm is suitable for image encryption.

1. INTRODUCTION

Since ancient times, all of humanity has shared the commitment to protecting their right to privacy. For governments, organizations, and people in general, privacy protection is crucial. For all these parties, the common issue is to protect their privacy, and keep their conversations and data confidential, so that no attacker can obtain or read this information. Cryptography was used to encrypt the sent messages with an algorithm using a key that was only known to the sender and receiver and to guarantee their confidentiality.

Many encryption methods have been put forth through the years to safeguard privacy. The classical encryption techniques include Caesar, Hill, Vigenere, etc. But these techniques cannot meet the security needs in the fast-evolving modern society. This gives birth to modern encryption algorithms, such as Data Encryption Standard (DES), Advanced Encryption Standard (AES), Rivest–Shamir–Adleman (RSA) algorithm, and digital signature algorithm (DSA). Capable of applying to various areas and meeting security needs, modern encryption algorithms can be divided into symmetric encryption and asymmetric encryption.

DES, a secret-key symmetric (single-key) encryption algorithm, relies on block encryption to encrypt huge amounts of data. Despite being the most widely used encryption algorithm ever, it has been defeated by contemporary computers. The DES approach is intricate and incorporates both confusion and diffusion processes. It utilizes S-box and Fiestel architecture.

Another approach for data encryption is AES, which was developed after DES was cracked. The keys used for both encryption and decryption in this symmetric-key encryption process are connected to one another. Four steps make up AES, including substituting bits, shifting rows, mixing columns, and adding keys. AES often employs the same confusion and diffusion processes as DES. The difference is that AES operates on irreducible polynomials. Polynomials play a significant role in mathematics and are widely employed, particularly in the field of cryptology. One of these is the Fibonacci polynomial, which has been employed in a number of research that have been published in the field of cryptology [1-8]. In this paper, the literature review summarizes these investigations.

Fibonacci polynomials and matrices are used in several of the above studies. However, in the encryption phase, con and diff operations are not applied. This paper develops a new encryption technique, providing an alternative to the encryption algorithms that offer con and diff operations in the literature. Fibonacci polynomials and matrices are adopted in the suggested strategy. The next term in a series of numbers known as the Fibonacci numbers is determined by the sum of the two initial conditions. Three separate images were encrypted and decrypted using the suggested technique. The success of the proposed technique was demonstrated by the histogram analysis of the operations conducted. The value obtained from the mean squared error (MSE) was zero, indicating the consistency between the original and decoded images. The number of pixel change rate (NPCR) and the unified averaged changed intensity (UACI) were also provided to demonstrate the effectiveness of our approach.

This study makes the following contributions:

- Uniform distribution is achieved through encryption.
- Confusion and diffusion processes are provided by Fibonacci polynomial matrices.
- The Fibonacci polynomial matrices in confusion and diffusion are calculated differently in each cycle, while the matrix of the column shuffling step in the AES encryption algorithm is the same in every cycle.
- The proposed approach can be applied easily to both images and texts.

- The statistical analysis proves the good performance of our approach.

The remaining parts are organized as follows: Section 2 reviews the literature on Fibonacci polynomials and matrices; Section 3 explains the proposed method; Section 4 applies our approach on 3 different images, and presents the results; Section 4 discusses the results, and summarizes the strengths and defects of our approach.

2. LITERATURE REVIEW

According to a novel method put forth by Mukherjee and Samanta [1], a message can be both encoded and hidden in an image by using the Fibonacci series to encrypt it.

In their study, Uçar et al. [2] developed two new encoding/decoding methods that make use of Fibonacci Q-matrices and R-matrices. The models rely on blocked message matrices and key-based encryption for each message matrices. These new algorithms have great accuracy capabilities in addition to increasing information security.

Zou et al. [3] introduced a brand-new technique for digital image mixing based on the Fibonacci numbers. There have been concerns raised about the mixing transform's homogeneity and periodicity. The benefits of mixing transform are as follows: Encoding and decoding are both very easy to use in real - time scenarios. The mixing effect is excellent, as the image data is randomly redistributed across the image. The technique is resistant to common image attacks like noise, compression, and data packet loss.

Khadri et al. [4] dealt with text encoding in such a way that the receiver can only find out the original message without any data loss or shift or data leakage. Some possible approaches were presented to solve their identified problem, using Fibonacci series. The data was encrypted by combining the original data with Fibonacci numbers to get a cipher text, which is incomprehensible to any intruder. Only the receiver knows the logic of the way doing this.

Diskaya et al. [5] developed a classical Aes-like cryptology based on the Fibonacci polynomial matrix, using a certain irreducible polynomial. Asci and Aydinyuz [6] generalized the AES-like cryptology on 2×2 matrices with the elements of k-order Fibonacci polynomial series using a certain irreducible polynomial in the cryptology algorithm, which is called AES-like cryptology on the k-order Fibonacci polynomial matrix. Anderson [7] proposed a number of keystream generators based on Fibonacci series. Karacam et al. [8] examined the transmission of time and position variable cryptology in the Fibonacci and Lucas number series with music.

3. MATERIALS AND METHODS

In mathematics, Fibonacci polynomials are considered a generalization of Fibonacci numbers. In 1883, Fibonacci polynomials $f_n(x)$ were examined, for the first time, by Belgian Mathematician Eugene C. Catalan. This work was followed by German mathematician Ernst Jacobsthal (1882-1965). In 1973, Hoggatt and Bicknell [9] defined Fibonacci polynomials and matrices.

The Fibonacci polynomials sequence can be defined by the recurrence relationship:

$$f_{n+2}(x) = xf_{n+1}(x) + f_n(x), \quad n \geq 0$$

with the initial conditions being $f_0(x)=0$ and $f_1(x)=1$.

The matrix of the Fibonacci polynomials $Q(x)$ can be defined as:

$$Q(x) = \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix},$$

and the general Fibonacci polynomials matrix $Q^n(x)$ can be expressed as:

$$Q^n(x) = \begin{pmatrix} f_{n+1}(x) & f_n(x) \\ f_n(x) & f_{n-1}(x) \end{pmatrix} \tag{1}$$

Much information about Fibonacci numbers can be found in the book Fibonacci and Lucas Numbers with Applications authored by Koshy [10]. According to this book, the Cassini's identity, an important identity of the Fibonacci numbers, can be explained as follows:

$$f_{n+1}(x) f_{n-1}(x) - f_n^2(x) = (-1)^n \tag{2}$$

Since the determinant of the Fibonacci polynomial matrix equals the Cassini's identity (2) and is non-zero, the Fibonacci polynomial matrix $Q^n(x)$ is invertible for all n values. This is an important feature for cryptology. The inverse of the Fibonacci polynomial matrix $Q^n(x)$ can be given as:

$$Q^n(x)^{-1} = \begin{pmatrix} f_{n-1}(x) & f_n(x) \\ f_n(x) & f_{n+1}(x) \end{pmatrix} \tag{3}$$

4. RESULT AND DISCUSSION

4.1 Our approach

The encryption algorithm involves three rounds, the initial round, the main round, and the final round. In the initial round, XOR operation is performed between the state matrix (clear text) and the key matrix (private key).

In the main round, the mixing process is applied in general. The confusion and diffusion in this round are created in the following steps:

- Traverse conversion
- Fibonacci s-box conversion
- Change row and column matrix
- Multiply matrix (multiplication by Fibonacci polynomial matrix)

The results obtained through con and diff operations and the key generated from the round key are subjected to XOR operation. The above operations in the main round are repeated twelve times.

In the final round, the matrix from the main part is once again processed through Fibonacci sbox conversion and the last key, before terminating the encryption process.

The process of our approach is detailed as follows:

Firstly, cross transform is applied on the add matrix, and the matrix is mixed using Fibonacci AES s-box [11].

Step 1 (Initial round): The 128-bit image (state matrix) and the polynomials in each cell of the key matrices are added as follows:

Suppose X_{ij}, Y_{ij}, and Z_{ij} are order 7 polynomials, where $1 \leq i$, $j \leq 4$,

$$Z_{ij} = X_{ij} + Y_{ij} = \left(a_7^{(ij)} + b_7^{(ij)}\right)x^7 + \mathrm{L} + a_0^{(ij)} + b_0^{(ij)},$$
$$a_k^{(ij)},\ b_k^{(ij)} \in \{0,1\},\ 0 \le k \le 7.$$

is obtained.

Step 2 (Main Round): According to mod 2, the order 7 polynomials are divided into two order 3 blocks, forming the 4th degree A and B polynomial elements. That is:

$$Z_{ij} = c_7^{(ij)}x^7 + c_6^{(ij)}x^6 + \mathrm{L} + c_0^{(ij)},\ c_k^{(ij)} \in \{0,1\},\ 0 \le k \le 7.$$

In the case that A_{ij}, B_{ij} are order 3 polynomials;

$$A_{ij} = c_7^{(ij)}x^3 + c_6^{(ij)}x^2 + c_5^{(ij)}x + c_4^{(ij)},\ c_k^{(ij)} \in \{0,1\},\ 4 \le k \le 7,$$
$$B_{ij} = c_3^{(ij)}x^3 + c_2^{(ij)}x^2 + c_1^{(ij)}x + c_0^{(ij)},\ c_k^{(ij)} \in \{0,1\},\ 0 \le k \le 3.$$

is obtained.

Then, the values marked in the traverse conversion matrix are multiplied by the first eight Fibonacci elements obtained using *mod 2* and the irreducible polynomial x^4+x+1.

The first eight Fibonacci elements are:

$$f_1(x) = 1,\ f_2(x) = x,\ f_3(x) = x^2 + 1,\ f_4(x) = x^3,\ f_5(x) = x^2 + x,$$
$$f_6(x) = x^2,\ f_7(x) = x^3 + x^2 + x,\ f_8(x) = x^3.$$

The multiplication goes as follows:

$$C_{ij} = A_{ij} \times f_{i+j} (\mathrm{mod}\ 2) \ \text{ or } \ C_{ij} = B_{ij} \times f_{i+j} (\mathrm{mod}\ 2)$$

If polynomials A_{ij}, B_{ij}, and C_{ij} of the obtained matrix are converted using the hexadecimal base against the Fibonacci s-box table, it is possible to obtain the Fibonacci s-box conversion matrix. The rows of this matrix are shifted by the first four terms of the Fibonacci numbers (i.e., 1, 1, 2 and 3), respectively. Then similar operations are applied to the columns, respectively:

- The first row of the matrix is shifted 1 bit left based on the value of the first term of the Fibonacci s-box table.
- The second row of the matrix is shifted 1 bit to the left based on the value of the second term of the Fibonacci s-box table.
- The third row of the matrix is shifted 2 bits to the left based on the value of the third term of the Fibonacci s-box table.
- The fourth row of the matrix is shifted 3 bits to the left based on the value of the fourth term of the Fibonacci s-box table.

The row changes result in a change row matrix. Then, the column change is applied to the change row matrix:

- The first column of the matrix is shifted up by 1 bit based on the value of the first term of the Fibonacci s-box table.
- The second column of the matrix is shifted up by 1 bit based on the value of the second term of the Fibonacci s-box table.
- The third column of the matrix is shifted up by 2 bits based on the value of the third term of the Fibonacci s-box table.
- The fourth column of the matrix is shifted up by 3 bits based on the value of the fourth term of the Fibonacci s-box table.

Thereby, a change column matrix is obtained, and then divided into blocks to be multiplied by the 2×2 Fibonacci polynomial matrix (1). The generated new 2×2 type matrices adopt mode 2 and irreducible polynomials x^4+x+1, respectively. On this basis, the multiply matrix can be obtained by:

$$\begin{bmatrix} K_{22} & L_{22} \\ K_{33} & L_{33} \end{bmatrix}\begin{bmatrix} f_{n+1}(x) & f_n(x) \\ f_n(x) & f_{n-1}(x) \end{bmatrix} = \begin{bmatrix} D_{11} & E_{11} \\ D_{21} & E_{21} \end{bmatrix},$$
$$\begin{bmatrix} K_{23} & L_{23} \\ K_{34} & L_{34} \end{bmatrix}\begin{bmatrix} f_{n+1}(x) & f_n(x) \\ f_n(x) & f_{n-1}(x) \end{bmatrix} = \begin{bmatrix} D_{12} & E_{12} \\ D_{22} & E_{22} \end{bmatrix}, \cdots,$$
$$\begin{bmatrix} K_{21} & L_{21} \\ K_{32} & L_{32} \end{bmatrix}\begin{bmatrix} f_{n+1}(x) & f_n(x) \\ f_n(x) & f_{n-1}(x) \end{bmatrix} = \begin{bmatrix} D_{34} & E_{34} \\ D_{44} & E_{44} \end{bmatrix}.$$

and $\{n=2+t : t.\text{round}=t\}$ where for $1 \le i,j \le 4$, K_{ij}, L_{ij}, D_{ij}, and E_{ij} are order 4 polynomials.

The *n* value is updated for these operations each round. The add matrix is created in the final phase of the main round by adding the multiply matrix and round key 0. The round key is added to the key created using the prior key in each loop.

Step 3 (Final Round): The final round takes place after the image has been encrypted in the main round. The image that needs to be encrypted will undergo multiply matrix operations in the final round, be added with round key 13, putting an end to the encryption process.

Figure 1 depicts the flow of the encryption process. Figure 2 presents the flow of the encryption algorithm.

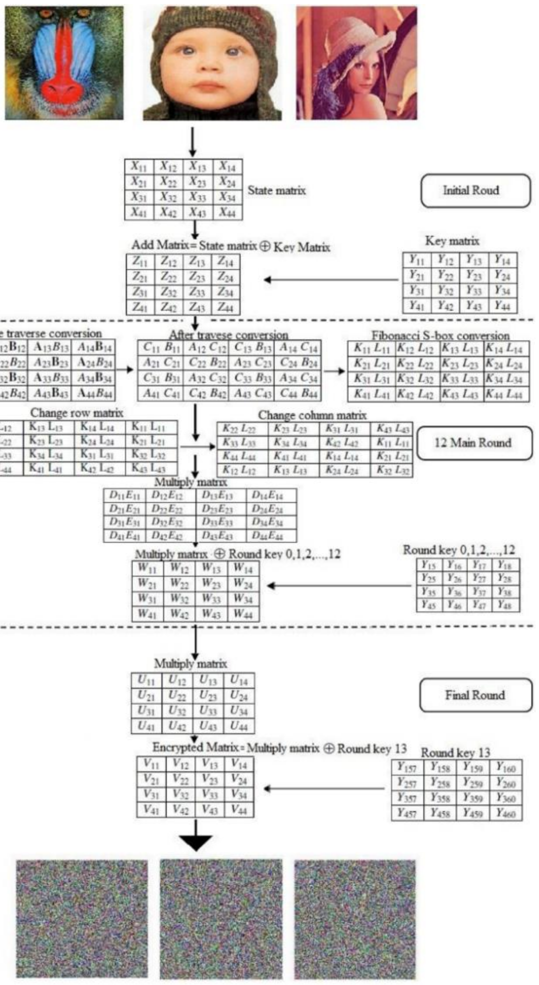
State matrix
Initial Roud
Add Matrix= State matrix ⊕ Key Matrix
Key matrix
Before traverse conversion
After traverse conversion
Fibonacci S-box conversion
Change row matrix
Change column matrix
12 Main Round
Multiply matrix
Multiply matrix ⊕ Round key 0,1,2,...,12
Round key 0,1,2,...,12
Multiply matrix
Final Round
Encrypted Matrix= Multiply matrix ⊕ Round key 13
Round key 13

Figure 1. The encryption process

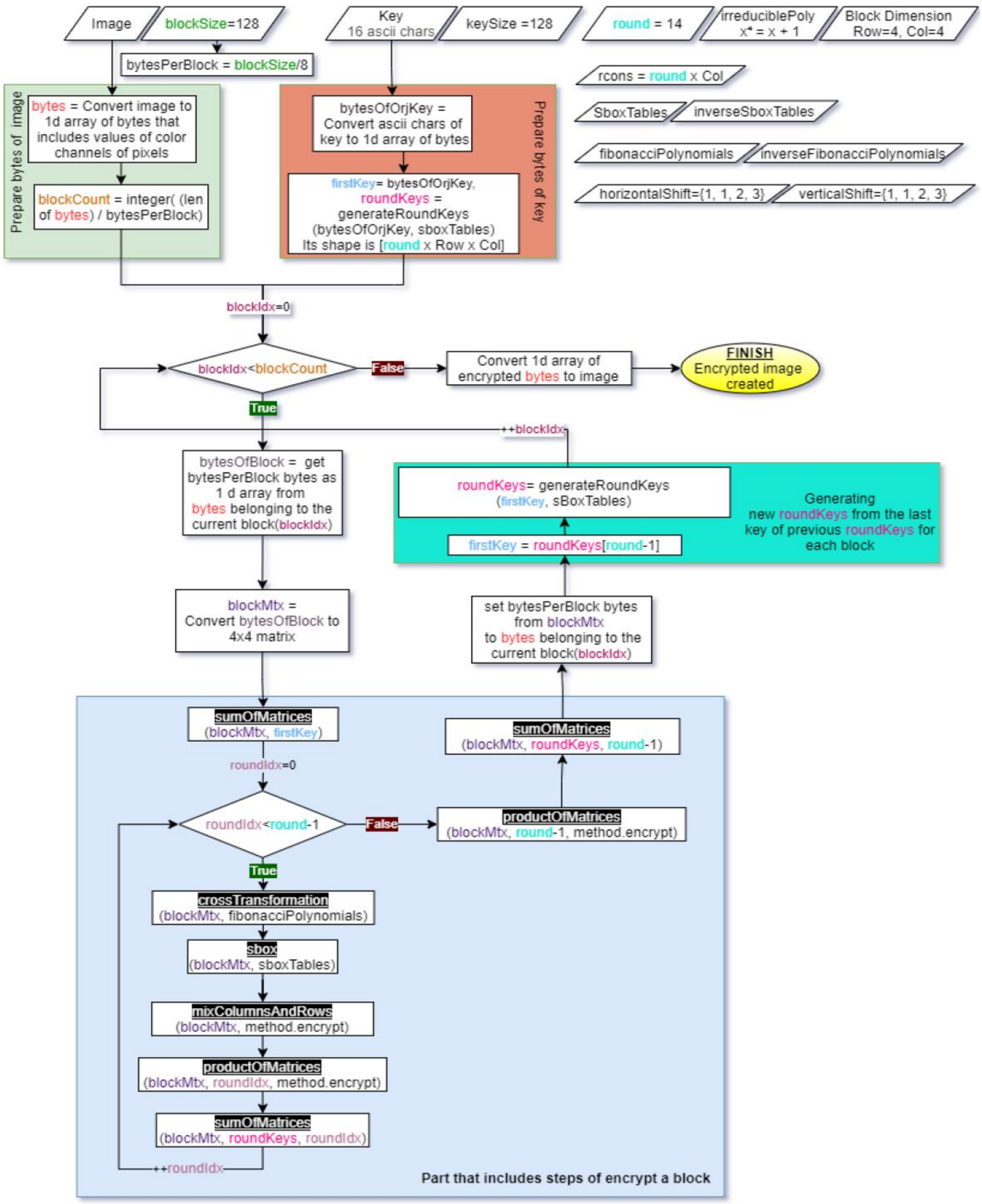

Image
blockSize=128
bytesPerBlock = blockSize/8
Key 16 ascii chars
keySize =128
round = 14
irreduciblePoly $x^4 = x + 1$
Block Dimension Row=4, Col=4
rcons = round x Col
SboxTables
inverseSboxTables
fibonacciPolynomials
inverseFibonacciPolynomials
horizontalShift={1, 1, 2, 3}
verticalShift={1, 1, 2, 3}
Prepare bytes of image
bytes = Convert image to 1d array of bytes that includes values of color channels of pixels
blockCount = integer((len of bytes) / bytesPerBlock)
Prepare bytes of key
bytesOfOrjKey = Convert ascii chars of key to 1d array of bytes
firstKey= bytesOfOrjKey, roundKeys = generateRoundKeys (bytesOfOrjKey, sboxTables) Its shape is [round x Row x Col]
blockIdx=0
blockIdx<blockCount
False
True
Convert 1d array of encrypted bytes to image
FINISH Encrypted image created
++blockIdx
bytesOfBlock = get bytesPerBlock bytes as 1 d array from bytes belonging to the current block(blockIdx)
roundKeys= generateRoundKeys (firstKey, sBoxTables)
firstKey = roundKeys[round-1]
Generating new roundKeys from the last key of previous roundKeys for each block
blockMtx = Convert bytesOfBlock to 4x4 matrix
set bytesPerBlock bytes from blockMtx to bytes belonging to the current block(blockIdx)
sumOfMatrices (blockMtx, firstKey)
sumOfMatrices (blockMtx, roundKeys, round-1)
roundIdx=0
roundIdx<round-1
False
True
productOfMatrices (blockMtx, round-1, method.encrypt)
crossTransformation (blockMtx, fibonacciPolynomials)
sbox (blockMtx, sboxTables)
mixColumnsAndRows (blockMtx, method.encrypt)
productOfMatrices (blockMtx, roundIdx, method.encrypt)
sumOfMatrices (blockMtx, roundKeys, roundIdx)
++roundIdx
Part that includes steps of encrypt a block

Figure 2. Flow of the encryption algorithm

Tables 1-5 display the nominal codes of cross transformation, s-box function, mix column and row, product of matrix and sum of matrix stages in the flowchart, respectively.

4.2 Practical applications of the proposed method and analysis results

This section discusses the practical application and test results of the proposed encryption algorithm. Specifically, our approach was applied to Lena, baboon and baby images, which are widely used in the literature. All tests were carried out on a computer with Intel Core i5 (7300HQ) quad-core CPU, 8 GB DDR4 RAM, 1 TB HDD and Windows 10. All images were processed by MATLAB R2019b. The application effect was measured by histogram analysis, root mean square error (RMSE) and differential attack analysis.

Table 1. The croos transformation

```
Function CrosTransformation (Inputs: blockMtx, Fibonacci
polynomials)
{
        byte dec, a, b, c;
        Polynomial poly, polyF, polyRes;
        bool direction;
        for (byte i = 0, j; i < Faes.rowSize; i++)
        {
          for (j = 0; j < Faes.columnSize; j++)
          {
            dec = blockMtx[i, j];
            a = (byte)(dec >> 4);
            b = (byte)(dec & 15);
            direction = ((i + j) & 1) == 0;
            if (direction) //Left
            {
              poly = Polynomial.WithDec(a);
               polyF = fibonacciPolynomials[
                Convert.ToByte(((i + 1) * (j + 1)) &
15) ];
            }
            else
            {
              poly = Polynomial.WithDec(b);
              polyF = fibonacciPolynomials[
                Convert.ToByte(((i + 1) * (j + 1)) & 15)
              ];
            }
             polyRes = (poly * polyF).reduction().mod2();
            c = Convert.ToByte(polyRes.evaluate());
            blockMtx[i, j] = direction
              Convert.ToByte((c << 4) + b) :
              Convert.ToByte((a << 4) + c);
          } } }
```

Table 2. SBOX function

```
Function sbox (Inputs: blockMtx, sboxTables)
{
        for (byte i = 0, j, dec; i < Faes.rowSize; i++)
        {
          for (j = 0; j < Faes.columnSize; j++)
          {
            dec = blockMtx[i, j];
            blockMtx[i, j] = sBoxTables[dec >> 4, dec &
15];
          } } }
```

Table 3. Mix column and row

```
Function mixColumnsAndRows (Inputs: blockMtx,
method)
{
        List<byte> row = new List<byte>(new
byte[Faes.columnSize]);
        List<byte> column = new List<byte>(new
byte[Faes.rowSize]);
        byte r, i, b, l;
        if (method == Method.decrypt)
        {
          for (r = 0, l =
Convert.ToByte(Faes.horizontalShift.Length); r < l; r++)
          {
            for (i = 0; i < Faes.columnSize; i++) row[i] =
blockMtx[r, i];
            for (i = 0; i < Faes.horizontalShift[r]; i++)
            {
              b = row[0];
              row.RemoveAt(0);
              row.Add(b);
            }
            for (i = 0; i < Faes.columnSize; i++)
blockMtx[r, i] = row[i];
          }
          for (r = 0, l =
Convert.ToByte(Faes.verticalShift.Length); r < l; r++)
          {
            for (i = 0; i < Faes.rowSize; i++) column[i] =
blockMtx[i, r];
            for (i = 0; i < Faes.verticalShift[r]; i++)
            {
              b = column[0];
              column.RemoveAt(0);
              column.Add(b);
            }
            for (i = 0; i < Faes.rowSize; i++) blockMtx[i,
r] = column[i];
          }
        }
        else
        {
          for (r = 0, l =
Convert.ToByte(Faes.verticalShift.Length); r < l; r++)
          {
            for (i = 0; i < Faes.rowSize; i++) column[i] =
blockMtx[i, r];
            for (i = 0; i < Faes.verticalShift[r]; i++)
            {
              b = column[Faes.rowSize - 1];
              column.RemoveAt(Faes.rowSize - 1);
              column.Insert(0, b);
            }
            for (i = 0; i < Faes.rowSize; i++) blockMtx[i,
r] = column[i];
          }
          for (r = 0, l =
Convert.ToByte(Faes.horizontalShift.Length); r < l; r++)
          {
            for (i = 0; i < Faes.columnSize; i++) row[i] =
blockMtx[r, i];
            for (i = 0; i < Faes.horizontalShift[r]; i++)
            {
              b = row[Faes.columnSize - 1];
              row.RemoveAt(Faes.columnSize - 1);
              row.Insert(0, b);
            }
            for (i = 0; i < Faes.columnSize; i++)
blockMtx[r, i] = row[i];
          } } }
```

Table 4. Product of matrix

```
Function productOfMatrices (Inputs: blockMtx, roundIdx,
method)
{
        Polynomial[,] group = new Polynomial[2, 2];
        Polynomial[,] fibonacciMatrix;
        {
           int round = roundIndex + 1,
              multipier = method == Method.decrypt ? 1 : -
1;
           fibonacciMatrix = new Polynomial[2, 2]{
                 {this.fibonacciPolynomials[round + 1 *
multipier] , this.fibonacciPolynomials[round] },
                 {this.fibonacciPolynomials[round] ,
this.fibonacciPolynomials[round - 1 * multipier] }, };
      }
        Polynomial res;
        uint d;
        for (
           byte k = 0, l, i, j, m, dec;
           k < 2;
           k++
        )
        {
           for (l = 0; l < 4; l++)
           {
              for (i = 0; i < 2; i++)
              {
                 group[i, 0] = Polynomial.WithDec(
                    Convert.ToByte(blockMtx[(k << 1) + i,
l] >> 4)
                 );
                 group[i, 1] = Polynomial.WithDec(
                    Convert.ToByte(blockMtx[(k << 1) + i, l]
& 15)
                 );
              }
              for (i = 0; i < 2; i++)
              {
                 dec = 0;
                 for (j = 0; j < 2; j++)
                 {
                    res = new Polynomial();

      for (m = 0; m < 2; m++)
                       res.summation(group[i, m] *
fibonacciMatrix[m, j]);
                    d = res.reduction()
                       .mod2()
                       .evaluate();
                    dec += (byte)(j == 0 ? d << 4 : d);
                 }
                 blockMtx[(k << 1) + i, l] = dec;
              }        }        }     }
```

Table 5. Sum of matrix

```
Function sumOfMatrices (Inputs: blockMtx, key)
  {
           for (byte i = 0, j; i < Faes.rowSize; i++)
              for (j = 0; j < Faes.columnSize; j++)
                 blockMtx[i, j] ^= key[i * Faes.columnSize + j];
        }
        private void sumOfMatrices(ref byte[,] blockMtx, ref
 byte[,,] roundKeys, byte roundIndex)
        {
           for (byte i = 0, j; i < Faes.rowSize; i++)
              for (j = 0; j < Faes.columnSize; j++)
                 blockMtx[i, j] ^= roundKeys[roundIndex, i,
 j];       }
```

4.2.1 Histogram analysis

For an image with brightness of [0, L − 1], the histogram can be described by a discrete function:

$$H(rk)= nk,$$

where, rk is the brightness intensity; nk is the number of image pixels with brightness intensity of rk [12]. For the target image of histogram analysis, the number of pixels with brightness of 0, 1, 2 … L-1 is calculated. The results are placed on the vertical axis. The histogram provides useful statistics about the image, which facilitates image compression, and segmentation. The histogram of encrypted images must be uniform [13, 14]. Figures 3-5 present the original, encrypted and decrypted states of the sample images, respectively. The R, G, B values are given on the right of these states. A uniform distribution has been obtained for all values, as shown by the coded histograms in the images. The histogram of the original images and the encrypted images are very different from one another. A uniform distribution makes it challenging to draw statistical inferences and makes statistical attacks on the suggested encryption technique more difficult.

4.2.2 Root Mean Square Error (RMSE)

This section investigates whether there is a difference between the original image and the decoded image, using the RMSE defined in formula (4). This metric measures the magnitude of error in quadric terms. As the standard deviation of the estimation errors, it is often used to find the distance between the predicted and true values. The value range of RMSE is 0 to ∞. The smaller the RMSE, the better the prediction. If RMSE equals zero, the model must have made no error [15].

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n} e_j^2}{n}} \tag{4}$$

where, A_j and P_j are the actual and predicted values, respectively; e_j=A_j-P_j is the error; n is the size of the dataset.

The magnitude of the error, absolute error, and squared error can be respectively expressed as $D1$=A_j-P_j=e_j, $D2 = |A_j - P_j| = |e_j|$, and $D3 = (A_j - P_j)^2 = (e_j)^2$, respectively.

The results in Table 6 demonstrate that no loss arises from encryption in the original images, for the decoded image that was acquired from the encrypted image has no difference. The histogram density graphs of the encrypted and decoded images are identical, as can be seen in Figures 3, 4, and 5.

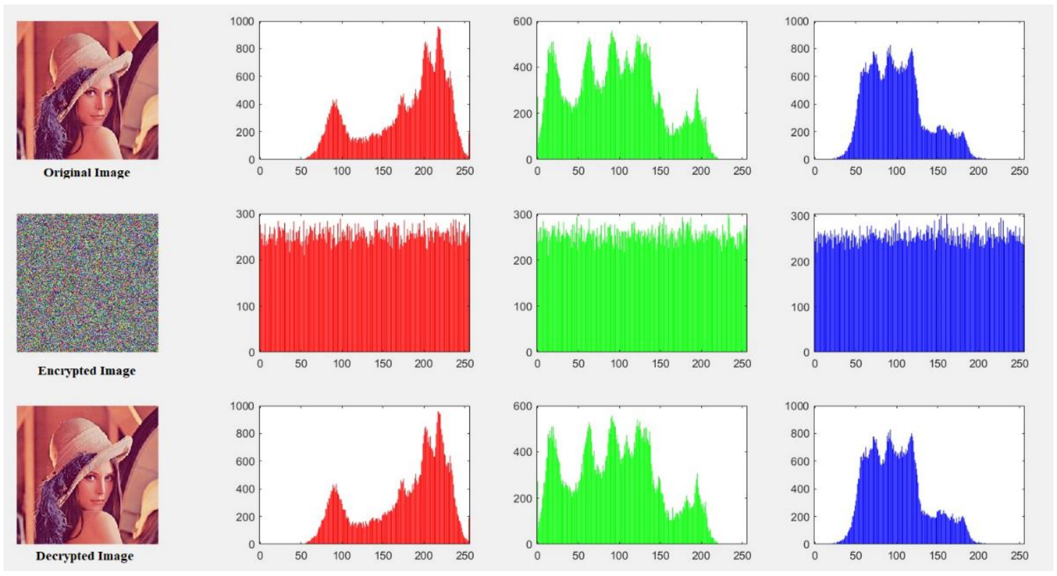

Figure 3. Histogram of the Lena image

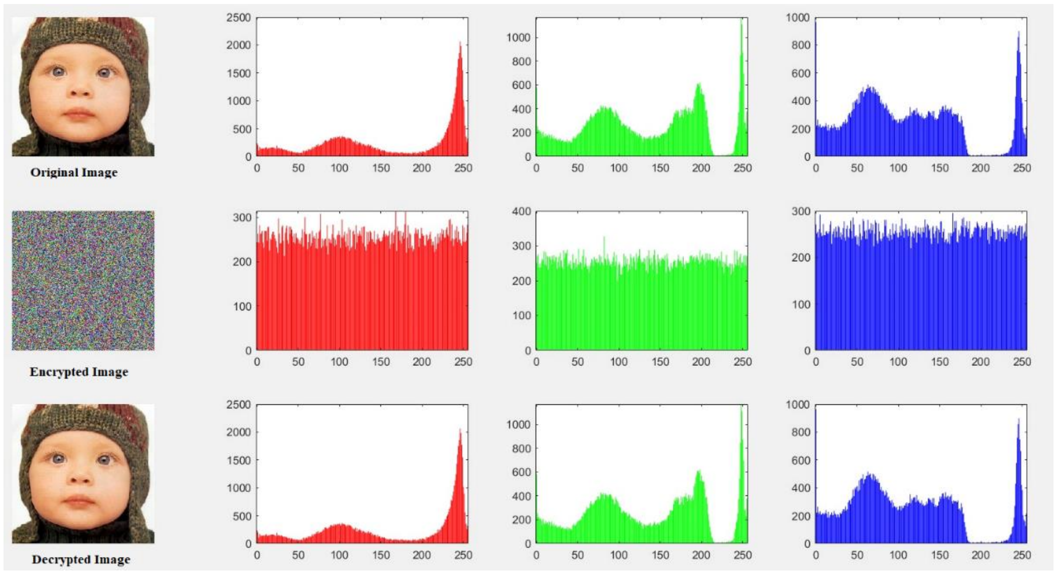

Figure 4. Histogram of the baby image

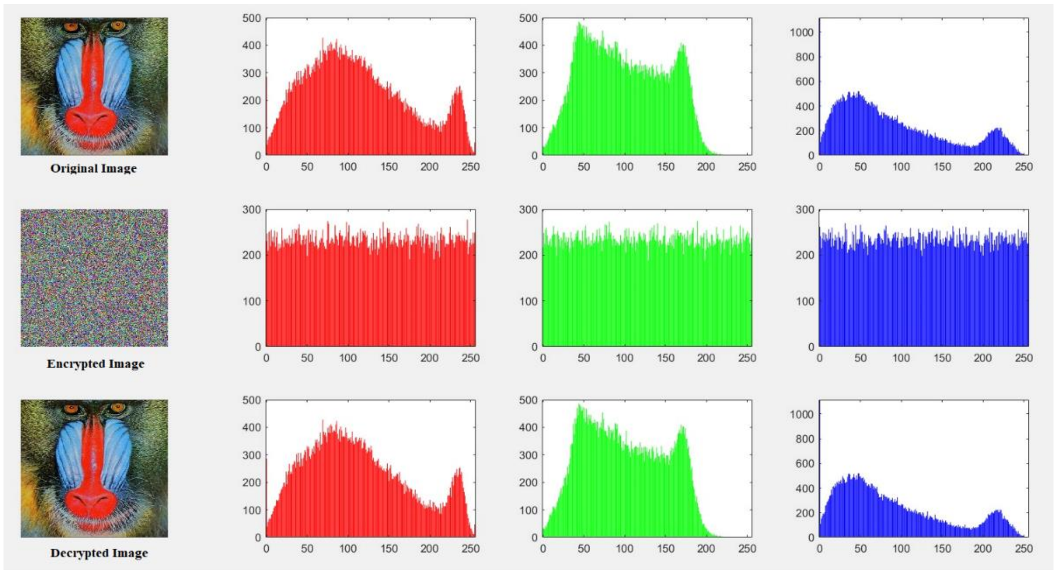

Figure 5. Histogram of the baboon image

Table 6. RMSE values of images

Sample pictures	RMSE
Lena	0
Baboon	0
Baby	0

4.2.3 Differential attack analysis

By using an encryption algorithm with a fixed key, differential attacks attempt to alter the original text pair in some way. The impact on the original text can be assessed by analyzing how much the encrypted output differ from the original input [16, 17]. The avalanche effect is being strengthened in cryptographic systems in an attempt to improve their defenses against differential attacks. This is a useful feature in cryptographic algorithms, for it causes minor changes in input to result in significant changes in output [18].

NPCR is one of the criteria for the effect of differential attacks on image encryption. NPCR evaluates an algorithm's sensitivity to minute alterations in the original image. First, the original image is subjected to the C1 cryptographic technique in order to determine the NPCR. Then, a randomly chosen pixel from the original image is altered. After that, the encryption algorithm is applied to the image of which a pixel has been modified again, yielding a second encrypted image, denoted C2. The following formulae are used to calculate the NPCR value [19, 20].

Table 7. UACI and NPCR test results

Sample pictures	NPCR	(UACI)
Lena	0.99619	0.335
Baboon	0.99602	0.334
Baby	0.99603	0.333

Table 8. Comparison of the proposed method with the studies in the literature

Reference	Method	Image	Parameter
Artuğer and Özkaynak [21]	Random (chaotic) Sbox	Baboon	NPCR= 0.9960 UACI:0.3343
Zhu et al. [22]	1D chaotic map	Lena	NPCR= 0.9963 UACI:0.3347
Benlashram et al. [23]	3D chaotic map and Pixel shuffiling	Lena	NPCR= 0.9965 UACI:0.3360
		Baboon	NPCR= 0.9965 UACI:0.3367
Shariatzadeh et al. [24]	Dynamic AES	Lena	NPCR= 0.9999 UACI:0.3364
		Baboon	NPCR= 0.9999 UACI:0.3361
El-Latif et al. [25]	Arnold cat map	Lena	NPCR= 0.9921 UACI:0.3343
		Baboon	NPCR= 0.9915 UACI:0.3338
Norouzi et al. [26]	hyper-chaotic system	Lena	NPCR= 0.9957 UACI:0.3347
		Baboon	NPCR= 0.9961 UACI:0.3357
Ge and Ye [27]	3D cat map	Lena	NPCR= 0.9961 UACI:0.3349
Sayed et al. [28]	2D Affine Transformation	Lena	NPCR= 0.9960 UACI:0.3348
		Baboon	NPCR= 0.9961 UACI:0.3345
Zhang et al. [29]	3D Chaotic map and DNA coding	Lena	NPCR= 0.9960 UACI:0.3346
		Baboon	NPCR= 0.9960 UACI:0.3345
Nematzadeh et al. [30]	Deoxyribonucleic Acid (DNA) sequence and Binary Search Tree (BST)	Lena	NPCR= 0.9962 UACI:0.3354
		Baboon	NPCR= 0.9936 UACI:0.3350
Zhang and Wang [31]	Deoxyribonucleic acid (DNA) encoding and chaotic system	Lena	NPCR= 0.9961 UACI:0.3345
		Baboon	NPCR= 0.9962 UACI:0.3343
Suseela et al. [32]	Torus Automorphism and Rubik's cube	Lena	NPCR= 0.9984 UACI:0.3687
		Baboon	NPCR= 0.9982 UACI:0.3612
Proposed method	Fibonacci polynomial matrix	Lena	NPCR= 0.9961 UACI:0.335
		Baboon	NPCR= 0.99602 UACI:0.334
		baby	NPCR= 0.99603 UACI:0.333

$D(i, j) = \{ 1 \text{ if } C_1(i, j) \neq C_2(i, j) \; 0 \text{ if } C_1(i, j) = C_2(i, j)$

$$NPCR = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i, j) \times \%100 \qquad (5)$$

As shown in formula (5), the ideal value for the NPCR value is 100%. According to the results in Table 1 about the proposed approach, the NPCR value is very close to 100%, indicating that the approach is extremely precise to the smallest changes in the input. Hence, changing a pixel value in the original image causes all pixels in the encrypted image to change.

UACL is another metric of image encryption efficiency. This metric is a yardstick of the avalanche effect:

$$UACI = \left(\sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|C_1(i, j) - C_2(i, j)|}{255 \times M \times N} \right) \times \%100 \qquad (6)$$

Eq. (6) is used to calculate the UACI.

The UACI value for successful image encryption is close to 33%. As shown in Table 7, our encryption algorithm achieved an UACI very close to 33%.

Table 8 compares the proposed approach with the methods in literature, using Lena and baboon images and chaotic systems. It is clear that our approach achieved the desired results in NPCR and UACI, surpassing the performance of the contrastive methods. This is attributable to the use of Fibonacci polynomials matrix, which has an important place in the field of mathematics.

There are different-based image encryption studies in the literature [33, 34].

5. CONCLUSION

Confusion and diffusion features are two basic requirements for encrypted text or images, and have been studied extensively in history. The most well-known solution is AES, which adopts irreducible polynomials. In the AES encryption algorithm, the same matrix is used for column shuffling in every cycle. By contrast, the proposed approach solves the Fibonacci polynomial matrices used in confusion and diffusion differently in each cycle. This helps to deal with more complex scenarios.

This study mainly presents a new encryption algorithm based on Fibonacci polynomials and matrices, providing an alternative to other encryption algorithms designed to meet the needs of confusion and diffusion features. The success of our algorithm was evaluated through encryption and decryption of 3 different images. The results of histogram analysis show that the encryption results in a uniform distribution. Besides, the RMSE (which is 0 for the 3 images) suggests no difference between the original and decoded images. Further, NPCR (approximate 100%) and UACI (approximate 33%) test results prove that our algorithm works successfully.

The proposed system was demonstrated to be practicable. By making the system more complex, the system security is greatly increased. The future research will propose novel encryption techniques using polynomials and matrices of number series like Lucas, Pell, Jacobsthal, Padovan, and Perrin. New S-box tables can also be created as an alternative to the s-box tables of AES.

REFERENCES

[1] Mukherjee, M., Samanta, D. (2014). Fibonacci based text hiding using image cryptography. Acharya Institute of Technology. Department of MCA, Bangalore, India, 2(2): 172-176. https://doi.org/10.12720/lnit.2.2.172-176

[2] Ucar, S., Tas, N., Yılmaz, N.Ö. (2019). A new application to coding theory via Fibonacci and Lucas numbers. Mathematical Sciences and Applications E-Notes, pp. 62-70. https://doi.org/10.36753/mathenot.559251

[3] Zou, J., Ward, R.K., Qi, D. (2004). A new digital image scrambling method based on Fibonacci numbers. In 2004 IEEE International Symposium on Circuits and Systems (ISCAS), 3: III-965. https://doi.org/10.1109/ISCAS.2004.1328909

[4] Khadri, S.K.A., Samanta, D., Paul, M. (2014). Approach of message communication using Fibonacci series: In cryptology. Lecture Notes on Information Theory, pp. 168-171. https://doi.org/10.12720/lnit.2.2.168-171

[5] Diskaya, O., Avaroglu, E., Menken, H. (2020). The classical AES-like cryptology via the Fibonacci polynomial matrix. Turkish Journal of Engineering, 4(3): 123-128. https://doi.org/ 10.31127/tuje.646926

[6] Asci, M., Aydinyuz, S. (2022). k-Order Fibonacci polynomials on AES-like cryptology. CMES-Computer Modeling in Engineering & Sciences, 131(1): 277-293. https://doi.org/10.32604/cmes.2022.017898

[7] Anderson, R. (1994). On Fibonacci keystream generators. In International Workshop on Fast Software Encryption, pp. 346-352. https://doi.org/10.1007/3-540-60590-8_26

[8] Karaçam, C., Algül, F.N., Tavit, D. (2021). Transmission of time and position variable cryptology in Fibonacci and Lucas number series with music. Journal of Mathematical Sciences and Modelling, 4(1): 38-50. https://doi.org/10.33187/jmsm.885876

[9] Hoggatt, V.E., Bicknell, M. (1973). Generalized Fibonacci polynomials. Fibonacci Qartelly, 11: 457-465. https://www.fq.math.ca/Scanned/11-5/hoggatt.pdf.

[10] Koshy, T. (2019). Fibonacci and Lucas Numbers with Applications, John Wiley & Sons.

[11] Mohamed, K., Ali, F.H.H.M., Ariffin, S., Zakaria, N.H., Pauzi, M.N.M. (2018). An improved AES S-box based on Fibonacci numbers and prime factor. International Journal of Network Security, 20(6): 1206-1214. https://doi.org/10.6633/IJNS.201811 20(6).21

[12] Neri, E., Caramella, D., Bartolozzi, C. (2008). Image processing in radiology. Medical Radiology. Diagnostic Imaging, Springer, Berlin. https://doi.org/10.1007/978-3-540-49830-8

[13] Amin, M., Faragallah, O. S., Abd El-Latif, A.A. (2010). A chaotic block cipher algorithm for image cryptosystems. Communications in Nonlinear Science and Numerical Simulation, 15(11): 3484-3497. https://doi.org/10.1016/j.cnsns.2009.12.025

[14] Chen, H.C., Guo, J.I., Huang, L.C., Yen, J.C. (2003). Design and realization of a new signal security system for multimedia data transmission. EURASIP Journal on Advances in Signal Processing, 2003(13): 1-15. https://doi.org/ 10.1155/S1110865703309011

[15] https://veribilimcisi.com/2017/07/14/mse-rmse-mae-mape-metrikleri-nedir, accessed on May 15, 2022.

[16] Biham, E., Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology, 4(1): 3-72. https://doi.org/10.1007/BF00630563

[17] Chen, L., Ma, B., Zhao, X., Wang, S. (2017). Differential cryptanalysis of a novel image encryption algorithm based on chaos and Line map. Nonlinear Dynamics, 87(3): 1797-1807. https://doi.org/10.1007/s11071-016-3153-y

[18] Hussain, I., Shah, T. (2013). Literature survey on nonlinear components and chaotic nonlinear components of block ciphers. Nonlinear Dynamics, 74(4): 869-904. https://doi.org/10.1007/s11071-013-1011-8

[19] Zhu, S., Zhu, C., Wang, W. (2018). A new image encryption algorithm based on chaos and secure hash SHA-256. Entropy, 20(9): 716. https://doi.org/10.3390/e20090716

[20] Alawida, M., Samsudin, A., Teh, J.S., Alkhawaldeh, R.S. (2019). A new hybrid digital chaotic system with applications in image encryption. Signal Processing, 160: 45-58. https://doi.org/10.1016/j.sigpro.2019.02.016

[21] Artuğer, F., Özkaynak, F. (2021). An effective method to improve nonlinearity value of substitution boxes based on random selection. Information Sciences, 576: 577-588. https://doi.org/10.1016/j.ins.2021.07.036

[22] Zhu, C., Wang, G., Sun, K. (2018). Improved cryptanalysis and enhancements of an image encryption scheme using combined 1D chaotic maps. Entropy, 20(11): 843-843. https://doi.org/10.3390/e20110843

[23] Benlashram, A., Al-Ghamdi, M., AlTalhi, R., Laabidi, P.K. (2020). A novel approach of image encryption using pixel shuffling and 3D chaotic map. In Journal of Physics: Conference Series, 1447(1): 012009-012009. https://doi.org/10.1088/1742-6596/1447/1/012009

[24] Shariatzadeh, M., Rostami, M.J., Eftekhari, M. (2021). Proposing a novel dynamic AES for image encryption using a chaotic map key management approach. Optik, 246: 167779. https://doi.org/10.1016/j.ijleo.2021.167779

[25] El-Latif, A., Ahmed, A., Li, L., Zhang, T., Wang, N., Song, X., Niu, X. (2012). Digital image encryption scheme based on multiple chaotic systems. Sensing and

Imaging: An International Journal, 13(2): 67-88. https://doi.org/10.1007/s11071-012-0409-z

[26] Norouzi, B., Mirzakuchaki, S., Seyedzadeh, S.M., Mosavi, M.R. (2014). A simple, sensitive and secure image encryption algorithm based on hyper-chaotic system with only one round diffusion process. Multimedia Tools and Applications, 71(3): 1469-1497. https://doi.org/10.1007/s11042-012-1292-9

[27] Ge, M., Ye, R. (2019). A novel image encryption scheme based on 3D bit matrix and chaotic map with Markov properties. Egyptian Informatics Journal, 20(1): 45-54. https://doi.org/10.1016/j.eij.2018.10.001

[28] Sayed, W.S., Radwan, A.G., Fahmy, H.A., Elsedeek, A. (2021). Trajectory control and image encryption using affine transformation of Lorenz system. Egyptian Informatics Journal, 22(2): 155-166. https://doi.org/10.1016/j.eij.2020.07.002

[29] Zhang, Q., Han, J., Ye, Y. (2021). Multi-image encryption algorithm based on image hash, bit-plane decomposition and dynamic DNA coding. IET Image Processing, 15(4): 885-896. https://doi.org/10.1049/ipr2.12069

[30] Nematzadeh, H., Enayatifar, R., Yadollahi, M., Lee, M., Jeong, G. (2020). Binary search tree image encryption with DNA. Optik, 202: 163505. https://doi.org/10.1016/j.ijleo.2019.163505

[31] Zhang, X., Wang, X. (2019). Multiple-image encryption algorithm based on DNA encoding and chaotic system. Multimedia Tools and Applications, 78(6): 7841-7869. https://doi.org/10.1007/s11042-018-6496-1

[32] Suseela, G., Kumari, N., Phamila, Y.A.V. (2016). Secured Image Compression using Wavelet Transform. Indian Journal of Science and Technology, 9(33): 1-6. https://doi.org/10.17485/ijst/2016/v9i33/92311

[33] Guler, H. (2021). Development of real-time fuzzy synchronization of chaos based system for image encryption. Traitement du Signal, 38(5): 1461-1467. https://doi.org/10.18280/ts.380521

[34] Cai Q.R. (2019). A secure image encryption algorithm based on composite chaos theory, Traitement du Signal, 36(1): 31-36. https://doi.org/10.18280/ts.360104