International Journal of Safety and Security Engineering
Vol. 14, No. 2, April, 2024, pp. 633-646
Journal homepage: http://iieta.org/journals/ijsse

# Image Encryption Based on Hybrid Parallel Algorithm: DES-Present Using 2D-Chaotic System

Suha Husam Jasim[1*], Haider Kadhim Hoomod, Khalid Ali Hussein

Computer Science, College of Education, University Mustansiriyah, Baghdad 10052, Iraq

Corresponding Author Email: suha_hussam@uomustansiriyah.edu.iq



https://doi.org/10.18280/ijsse.140229

**Received:** 24 January 2024
**Revised:** 13 March 2024
**Accepted:** 22 March 2024
**Available online:** 26 April 2024

***Keywords:***
*chaotic keys generation, Data Encryption Standard (DES), image, parallel computing, present light weight*

**ABSTRACT**

Image encryption algorithms have recently been developed to protect data from hackers and give recipients privacy. DES is a widely recognized block cypher that has certain vulnerabilities that make it susceptible to differential attacks. The present is a lightweight symmetric algorithm that provides privacy for transferring information over the network but has some drawbacks in that it is difficult to maintain an appropriate level of complexity. The study suggests that to encrypt and decrypt images as quickly as possible, the system uses parallel environments in algorithms (Present and DES). It also uses a 2D-Chaotic key generation system to make the system stronger against statistical, differential, and brute force attacks. Where the DES algorithm uses four rounds, within each one round from the des, the present algorithm executes only four rounds, and the same 2D-Chaotic System is used to generate the key. The keys and blocks are distributed to 4 cores, 5 cores, or 6 cores at the same time. The performance evaluation of the proposed algorithm is quantified by several metrics: All peak signal-to-noise ratio (PSNR) values are low, which means the quality image encryption is good. Unlike MSE, all the values are very high, which indicates that the image we have encrypted has no similarity to the encrypted image. The NPCR value of 99.6658% indicates a high degree of accuracy in changing pixel values. Additionally, a unified average changing intensity (UACI) that doesn't go over 30.90% shows that the algorithm is good at making big changes in pixel intensities. And the analysis speed of the proposed system based on the parallelism of the environment is faster than the sequence algorithms (DES-Present). The results demonstrate the algorithm's ability to encrypt color images, making it useful in applications that require strong data and image security.

## 1. INTRODUCTION

Image encryption has become a vital tool for maintaining privacy in a variety of contexts, especially those involving sensitive data [1]. By using encryption techniques, visual content is safe from unauthorized access [2]. Symmetric and asymmetric systems are the general categories into which cryptographic techniques fall. In symmetric key cryptography, the sender and recipient agree on which key will be used. Asymmetric key cryptography, on the other hand, gives each user a distinct set of public and private keys [3]. Within a variety of encryption algorithms, researchers have argued for the use of large keys in an attempt to increase data security. Although stronger data protection is often associated with larger key sizes, it is important to recognize the complexity that comes with key management, use hybrid algorithms to take advantage of the advantages of each algorithm, and use a parallel environment to get faster output [4, 5].

## 2. RELATED WORKS

Fernando et al. [6] suggested and tested AES and DES on Raspberry Pi minicomputers. I suggested finding time- and memory-efficient algorithms to encrypt and decrypt the same key. The study found that AES encrypted messages faster. Wu and Dai [7] suggested using a quantum genetic algorithm to optimize the DES S-box design. The 64 DES cipher texts and variable bits differ by 32 bits while encrypting accounting data. Laia et al. [8] suggested the successful operation of the DES algorithm and the Blum-Blum-Shub (BBS) algorithm to encrypt and decrypt messages to improve their security. Xing et al. [9] suggested testing serial AES and DES code on their experimental platform through task-based evaluations. Serial DES and AES were 40 and 72 times slower than parallel ones. Barhoush et al. [10] suggested increasing the size of the DES key to make it more secure without increasing the cost. Mihalkovic et al. [11] suggested discussing the new CBC symmetric cryptographic function and matrix power function. AES-128 versus Triple DES evaluates three 64-bit arithmetic ciphers. Yunus et al. [12] suggested using the Triple DES (3DES) algorithm to secure medical record data in Jember Family Health Home Clinic's electronic health record system, as shown in Table 1.

**Table 1.** Comparison between the proposed algorithm and related works

| References | Encryption Method | The Result |
|---|---|---|
| **[6]** | AES -DES algorithm test on Raspberry Pi | According to the findings of the study, AES encrypted messages more quickly than DES. AES-encrypted messages require more memory than DES-encrypted ones do. |
| **[7]** | The DES algorithm based on logistic discrete chaotic. | The chaotic system has good chaotic characteristics, but its system structure is simple, and its resistance to some differential attacks has not been fully considered. However, large data transmissions significantly reduce system performance. |
| **[8]** | examining force analysis attacks on BC3, AES, and DES | The success rate for analysis attacks to recover cryptographic device secret keys is one hundred percent for AES and seventy-five percent for DES. |
| **[9]** | Comparing Blowfish, DES, and AES on small and large data files. | When it comes to small text files that are less than one thousand bytes in size, DES performs better than both AES and Blowfish. |
| **[10]** | DES AES | To make DES more secure and faster than AES, the key size and permutation table were increased. The extended DES is DES22. The experiment shows that DES22 is faster and safer than AES. |
| **[11]** | AES and TDES | The encryption is approximately 47 times faster than TDES and roughly 1.5 times faster than AES-128. |
| **[12]** | EHR system and the Triple DES (3DES) algorithm. | The EHR system will have file security with the Triple DES (3DES) algorithm, which uses UML diagrams to protect critical medical record data. |
| **The proposed (DES-Presnt)** | | Key space ($2^{232.4}$) and pass all the test NIST. Higher speed (encryption/decryption). (4.434164762/ 3.40021944) mile/second. High Performance (speed 3.311008704, efficient 0.551834784, cost 25.80416393, overhead 11.5645287) that the algorithm can resist statistical attacks, differential attacks, and brute force attacks |

## 3. CONTRIBUTION

A hybrid algorithm has been proposed between the DES and Present algorithms using parallel execution of these algorithms. Depending on the specified number of cores, there can be four, five, or six types, and using the two-dimensional chaotic system to create the keys in a dynamic way. It is difficult to predict the key with each execution process, which increases the difficulty of guessing the keys. To an unauthorized person. There are numerous algorithms in IoT applications that are highly efficient but require significant time, processor power, and memory, which are limited resources. Hence, it is crucial to discover an algorithm that achieves a harmonious equilibrium between complexity, implementation time, resource utilization, and enhanced security. The algorithm DES, which exhibits sluggishness in the encryption process and possesses vulnerabilities that render it susceptible to attacks, the present lightweight algorithm's low complexity renders it susceptible to a wide range of attacks. The objective of this is to develop a highly efficient algorithm for encrypting color images. To achieve this, the DES algorithm will be combined with the Present lightweight and take advantage of the strengths of both algorithms to achieve a balance between strong security and fast encryption time between different encryption techniques. Additionally, efforts will be made to implement this algorithm in a parallel manner to maximize speed and efficiency.

### 3.1 Data Encryption Standard (DES)

The Data Encryption Standard (DES) block is a symmetric-key algorithm for the encryption of digital images. To meet the need for a standardized encryption algorithm, IBM and NIST.

Researchers created the 1970s-era DES [13]. A safe and effective way to encrypt and decrypt internet data was the goal. Many mathematical operations are performed on a user-provided initial key to create these keys. Key generation begins with the PC-1 permutation function compressing the 64-bit key into a 56-bit sub key. This sub key is split into 28-bit C0 and D0. Each round of encryption or decryption shifts Cn and Dn left by one or two bits, odd or even. After PC-2, the Cn+1 and Dn+1 halves form the 48-bit sub keys KN+1a and KN+1b [14]. In the next step, the initial permutation reorganizes input data before DES processing. One 64-position permutation table rearranges input bits. This first permutation has two goals. The diffusion mechanism spreads input-bit effects to output bits. Second, 1-bit plaintext changes affect multiple cipher text bits, complicating and securing DES [15]. Expanding DES diffusion and complexity require permutation. The expansion function expands the last round's 32-bit half-block to 48 bits. XOR add input block bits in E. expansion. Larger initial blocks increase the DES encryption and decryption error rooms. Horst DES relies on Feistel's network structure. Left and right are equal plaintext feistily networked parts. Functions use Ln-1 and round pre-processing sub key KN to calculate Rn. Each round, Rn-1 becomes Ln. Bitwise operations include XOR, permutations, and S-boxes. Repeat until L16=R16. Last Permutation Feistily network intermediate cipher texts L16R16 permute final or inverse initial after all rounds. This step reverses the first permutation and generates cipher text for transmission or storage. Removing intermediate cipher text patterns in the final permutation increases diffusion. Therefore, DES hopes an attacker without the right key will struggle to decipher the cipher text [16]. As illustrated in Figure 1, for a single round implementation [17].

### 3.2 Present lightweight algorithm

This block cipher-symmetric algorithm protects cyber data. Made for low-resource devices. Andrey Bogdanov and his team have been using it since 2007. Insurance, computer science, etc. use it. The fast, efficient solution maximizes data security [18]. The 31-round SP network and 64-bit blocks encrypt data. The key can be 128 or 80 bits. A 4-bit to 4-bit S-box (16 round parallel rounds) replaces an 8-bit S-box for nonlinear layers. Because of its adaptability and simplicity. Step-present algorithm [19], as shown in Figure 2.

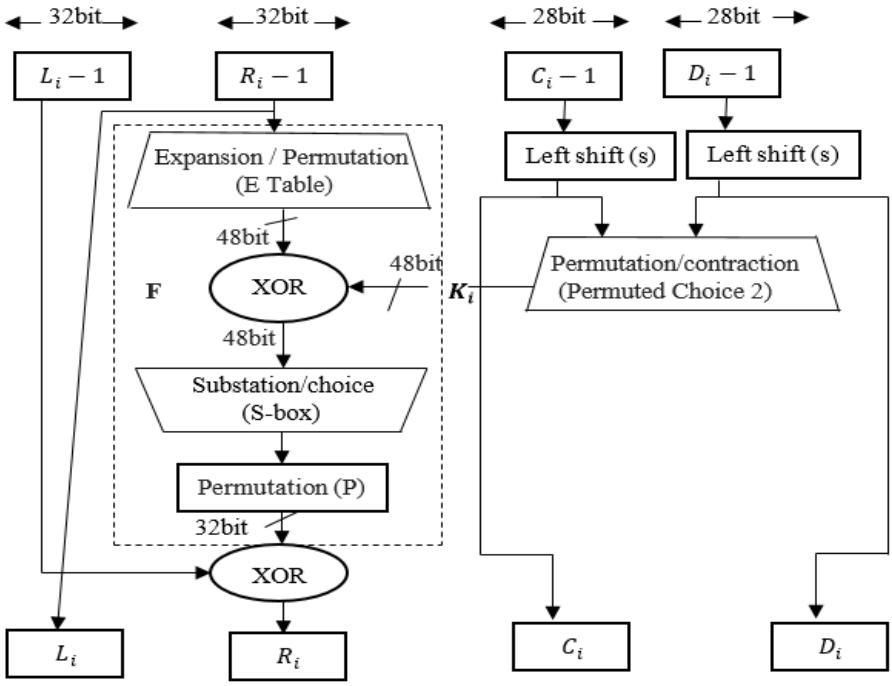



**Figure 1.** One round of DES algorithm [17]

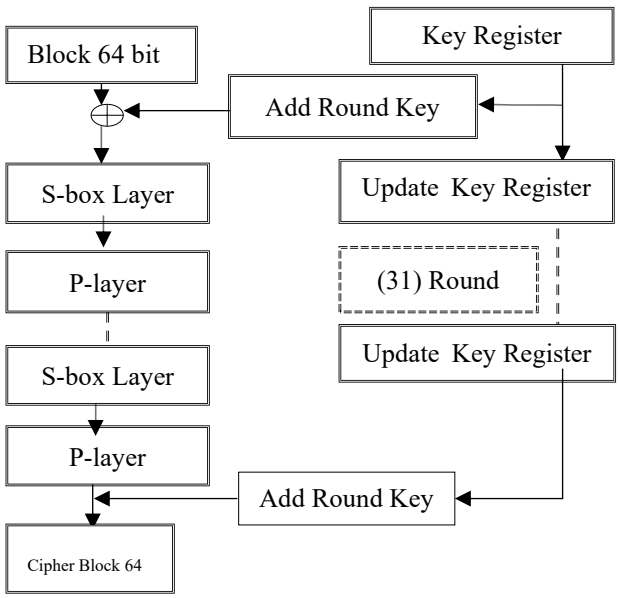



**Figure 2.** Present lightweight algorithm

Present algorithms include Round Key, S-Box, P-Layer, and Key Update. First, Status = Status XOR (round key). These use "S" containers. S-Boxes substitute four-bit inputs into outputs nonlinearly. The initial input and final output of an S-box circuit are shown. Next after the permutation layer operation P (m) [20, 21]. Provide key updates. From $K_{79}$ to K0, Register K stores user input permanently. In K78, K77, K76, etc.'s 64-bit round key, I am registering K's remaining 64 bits. Ki= ($K_{63}$ $K_{62}$ $K_{61}$...$K_0$) Eq. (1) explain [22].

$$[K_{79}K_{78} \ldots\ldots K_1K_0] = [K_{18}K_{17} \ldots\ldots K_{20}K_{19}].$$
$$\text{And } [K_{79}K_{78}K_{77}K_{76}] = S[K_{79}K_{78}K_{77}K_{76}].$$
$$\text{Finally}[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus Round\_Counter \quad (1)$$

### 3.3 Chaotic map generation

The suggested encryption algorithm generates chaotic map key streams. The chaotic system has two quadratic nonlinear equations, two initial values, and five control parameters. It has two equations, x and y. This map generation generates X, Y, and Z stream vectors. Image width and height are pixels, and each vector is (W×H×3), as in Eq. (2), as shown in Table 2.

$$X_{i+1} = a\,Y_i^2 - b\,X_i^2 - c$$
$$Y_{i+1} = d\,X_i\,Y_i - e\,X_i \quad (2)$$

where, a=4, b=1.1, c=4.4, d=0.1, and e=8. These values created chaotic phase portraits [23, 24].

**Table 2.** Two-dimensional chaotic key generation

| **Algorithm 1. Chaotic Keys Generation** |
|---|
| Input: a, b, c, d, e, $X_0$, $y_0$, W, H |
| Output: X, Y, Z // keys stream vectors of dimension (1 N) |
| start Algorithm |
| Processing Algorithm: |
| Step1: $X_1 = X_0$ , : $y_1 = y_0$ , : $z_{10} = X_1 \oplus y_1$ |
| Step2: N = W×H × 3 |
| Step3: Iterate (N-1) times |
| $X_{i+1} = a\,Y_i^2 - b\,X_i^2 - c$ // i= (2, 3.....N). |
| $Y_{i+1} = d\,X_i\,Y_i - e\,X_i$// i= (2, 3.....N). |
| $z_{i+1} = X_{i+1_1} \oplus Y_{i+1}$ . |
| $X_i$=$X_{i+1}$ |
| $Y_i$ =$Y_{i+1}$ |
| End Iteration |
| End Algorithm. |

### 3.4 Parallel computing

The general field that covers all facets of computing with parallelism—hardware, software, and algorithms—is called parallel computing. One important technique in parallel

computing is parallel processing, which is the act of carrying out multiple computations at once. An essential element of efficient parallel computing and processing is a parallel algorithm, which is a set of instructions created with the express purpose of being executed in parallel [25, 26].

## 4. PROPOSED ENCRYPTION ALGORIT

Data Encryption Standard (DES) and Present are two algorithms proposed to encrypt images in a parallel environment to ensure instantaneous transmission and reception of images. The two proposed algorithms are illustrated in Figure 3 and Figure 4. The first step is to download the image that needs to be encrypted from its storage location, then the image must be divided into three vectors called red, green, and blue, each of which is a two-dimensional matrix. After that, the vectors are summed into a single vector, and then the single vector must be divided into a block in order to apply the proposed algorithm. The DES algorithm is executed twice. A 128-bit block is inserted. Each block is divided into 64 bits to be an input to the DES algorithm. The 56-bit key is generated using the two-dimensional chaos system equation and four rounds instead of 16 rounds. This is considered an improvement to the DES algorithm, and within each round there is one round. From the Des algorithm, the present algorithm is executed only four rounds instead of 31 rounds. The present algorithm takes its input from the left side of the DES algorithm, where each side equals 32 bits. When the left sides of the algorithm are combined, it becomes 64 bits, and the key is 80 or 128 bits long, which is generated using the two-dimensional chaos system equation. After that, the sub-images and encryption keys are distributed across a number of cores (4, 5, or 6), which are synchronized depending on the parallel kernel environment. The encryption image is generated by initially resampling the coding matrix (consisting of red, blue, and green components) into RGB format and subsequently merging them into a unified RGB sequence. This process is iterated until the encrypted image is acquired, as shown in Tables 3-6.

**Table 3.** Preprocessing input image

| **Algorithm 2. preprocessing input file (image)** |
|---|
| Input: block size, RGB image<br>Output: number the blocks |
| Steps<br>1: begin.<br>2: Enter color image.<br>3: Get image size (rows, columns).<br>4: separated the color image into three distinct sections:<br>Red, green, and blue.<br>5: Each color channel should have its reshape converted<br>Into a two-dimensional matrix. It is red, green, and blue.<br>6: Three channels become one T channel.<br>7: Split the T channel into equal blocks of a certain size.<br>8: End Algorithm. |

**Table 4.** Key generation

| **Algorithm 3. Key sub generation for DES.** |
|---|
| Input: A, B, (c, d, e, f, and g), key size.<br>Output: $Key_{\mathrm{a}}$(i), $Key_{\mathrm{b}}$(i) |
| Steps<br>1: specifying general Number (key size).\\ Apply the<br>(DES-Present) Algorithm.<br>2: For i ← 0 To (key size).<br>$A_{(i+1)}$ ← $c\,B_i^2 - d\,A_i^2 - e$ .<br>$B_{(i+1)}$ ← $f\,A_i\,B_i - g\,A_i$ .<br>T1 Round ($A_{(i+1)} \times 10^5$) Mod (256).<br>$Key_a$(i) ← T1.<br>T2 ← Round ($B_{(i+1)} \times 10^5$) Mod (256).<br>$Key_{\mathrm{b}}$(i) ← T2.<br>3: For J ← 1 to 8. \\ for the DES, after the apply the table PC1<br>$column_1$ ← Convert to binary $Key_{\mathrm{a}}$(i).<br>$Row_1$ ← Get first seven bit of COL1.<br>$Key_{a1}$ ← to merge($COL_1$,$ROW_1$).<br>4: split $Key_{b1}$ into $C_0$ and $D_0$.<br>$C_0$ ← $Key_{a1}[0\ To\ 27]$.<br>$D_0$ ← $Key_{a1}[28\ To\ 56]$.<br>For K ← 1 To 4.<br>4:1 Apply shift operation on $C_0$ to find $C_1$, Apply shift operation on $D_0$ to find $D_1$.<br>4:2 $T_1$ ← to merge ( $C_1$, $D_1$.).<br>$T_2$ ← Apply table PC2 on the $T_1$, for the get The Key 48bit for Find $T_3$.<br>Sub key (k) ← $T_3$.<br>$C_0$ ← $T_1[0\ To\ 27]$<br>$D_0$ ← $T_1$ $[28\ To\ 56]$.<br>4:3 return Sub key (K)<br>5: End Algorithm. |

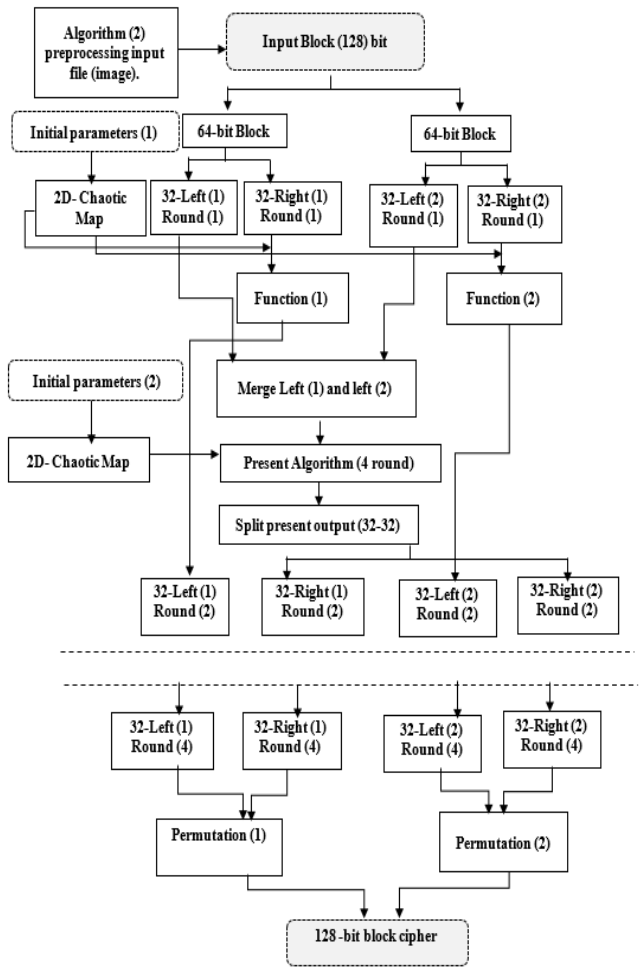



**Figure 3.** Proposed sequential algorithms

**Table 5.** Generating keys for the present algorithm

**Algorithm 4. Generating sub keys for the present algorithm.**

Input: Input: $Key_a(i)$, $Key_b(i)$
Output: list of 64-bit round keys

Steps
1: Start
2: initial parameters
round keys = [], S-Box-Layer // Hex-decimal
$Key_3($ ← combine ( $Key_1(i)$, $Key_2(i)$)
3: For i ← 1 to 4 // The block size is 64 bits
$[k_{79}k_{78}K_{77}K_{76}K_{75} \dots k_{16}]$ <<<16
$[k_{63}k_{62}\, K_{61} \dots]$ //Raw key extraction
$Key_3(i)$ ← $[k_{79}k_{78}K_{77}K_{76}K_{75} \dots k_{16}]$
Split $Key_3(i)$ into part
$L_1$= $Key_3(i)[1.61]$, $R_1$= $Key_3(i)$ [62.80]
$Key_{b1}$= swap [ $L_1$ , $R_1$] // (<<< 19) or (>>> 61)
4: $T_1$ = $Key_3(i)$ $[K_{79}K_{78}K_{77}K_{76}]$
$T_2$= Convert binary to Hex-decimal ($T_1$) //with 4 bit and the Worked 16 parallel.
$T_3$= S_box [$T_2$]
$T_4$= hex-decimal to Bin ($T_3$,4))//Convert hex-decimal to binary with 4 bit
5:$[K_{19}K_{18}K_{17}K_{16}K_{15}]$ ← $[K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus Round_i$
$Key_3(i$ ← $[K_{19}K_{18}K_{17}K_{16}K_{15}]$
6: return $Key_3(i)$
7: End Algorithm.

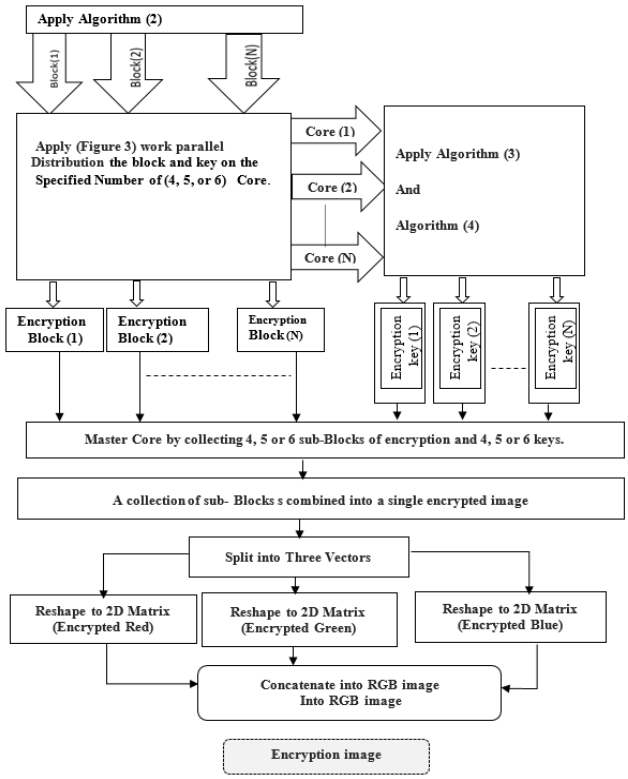



**Figure 4.** Proposed parallel algorithms

To decrypt the proposed algorithm on the encrypted image, the steps of the algorithm begin to be applied, reversing the steps, starting from the last round to the first round.

**Table 6.** Proposed algorithms (DES-present)

**Algorithm 4. Proposed algorithms**

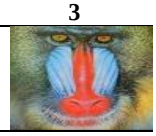
Input: block is 128 bits, $Key_a(i)$, $Key_b(i)$ //distribution the Block and key on the specified number of (4,5, or 6) Core.
Output: Cipher Block is 128 bits

steps
1: initial parameter, S-Box Table, P-player for present, Table Initial Permutation (IP), the Table Expansion box (E), Table, Permutation, Box (p) in f-function, Table Permutation box $IP^{-1}$.
2: Apply algorithm (2).
3: For i ← 1 to 4
3:1 Apply algorithm (3) and Apply Algorithm Encryption (DES)// Apply Algorithm twice.
3:2 Block presnt ← to merge (left DES one , left DES two) // block size 64 bit.
3:3 out put the presnt ← Apply algorithm (4) and Algorithm Encryption Present // Split block into two
3:4 Right one i ← $Right\ DES1 \oplus$
out put the presnt algorithm left one 32 bit.
3:5 Right two i ← $Right\ DES2 \oplus$
out put the presnt algorithm left two 32 bit .
3:6 End for i
4: Cipher1 ← Swap (out put the present left one 32 bit , Right one)//Apply Table IP^ (-1).
5: Cipher2 ← Swap (out put the present left two 32 bit , Right two)//Apply Table IP^ (-1).
6: Cipher Block ← Combine (Cipher 1, Cipher 2).
7: End Algorithm.

## 5. COMPARISON TEST RESULTS AND ANALYSIS

Results and analysis from comparative tests for a cryptosystem to be considered strong, it must be able to withstand all known types of attacks. This includes attacks targeting only cypher text as well as differential, statistical, and brute force attacks. Using the algorithms proposed for use in a parallel environment, we were able to encrypt and decrypt images quickly and securely. Python, the latest version of the programming language Visual Studio Code, is used to write all tests. Introduced by the processor is the Intel (R) CoreTM i7-10750H 2.60GHz/2.59GHz CPU with 16.0GB of RAM. Apply tests to the data set in Table 7 to find statistical, differential, and key space tests.

**Table 7.** Data set image

| Number Image | 1 | 2 | 3 |
|---|---|---|---|
| |  |  |  |
| **Number Image** | **4** | **5** | **6** |
| |  |  |  |

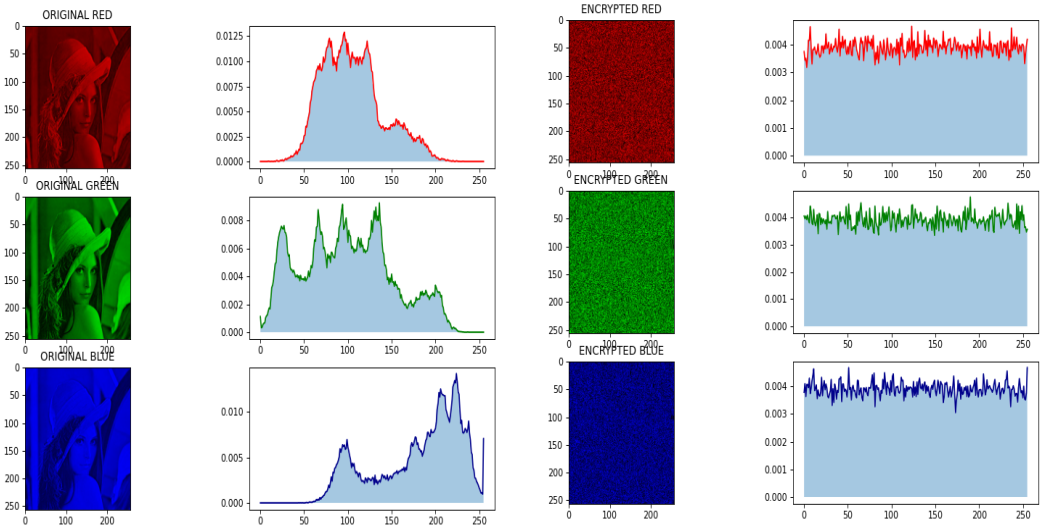
### 5.1 The examination of statistics

The test comprises entropy, correlation, and histogram tests for the lens image.

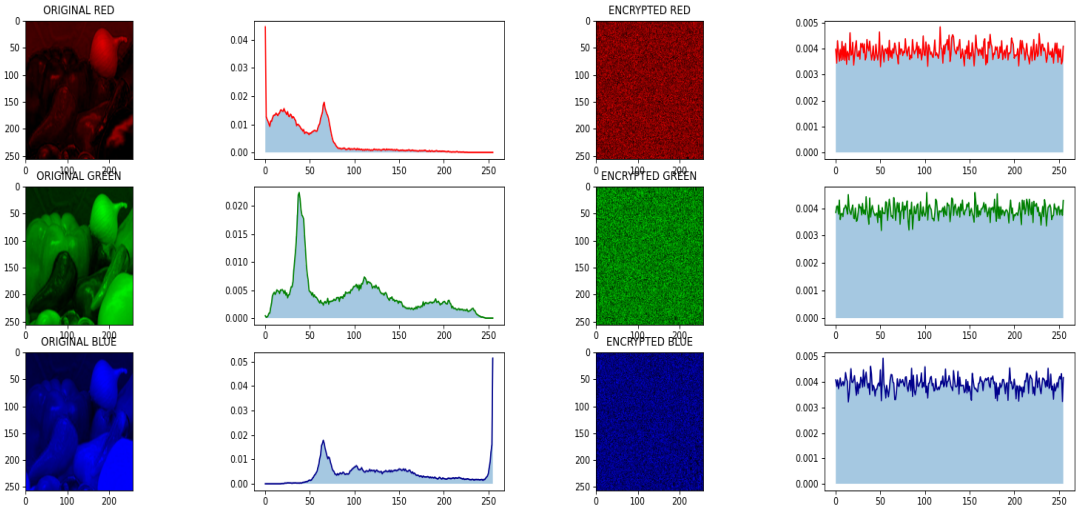
1. Histogram Analysis: One important statistic to consider when evaluating the suggested system is the histogram analysis. Figure 5 shows the histograms of the two images.

The result, the original image exhibits prominent, abrupt increases followed by rapid decreases, while the encrypted image displays a consistent distribution that deviates significantly from the original image and lacks any discernible statistical resemblance in terms of visual appearance.

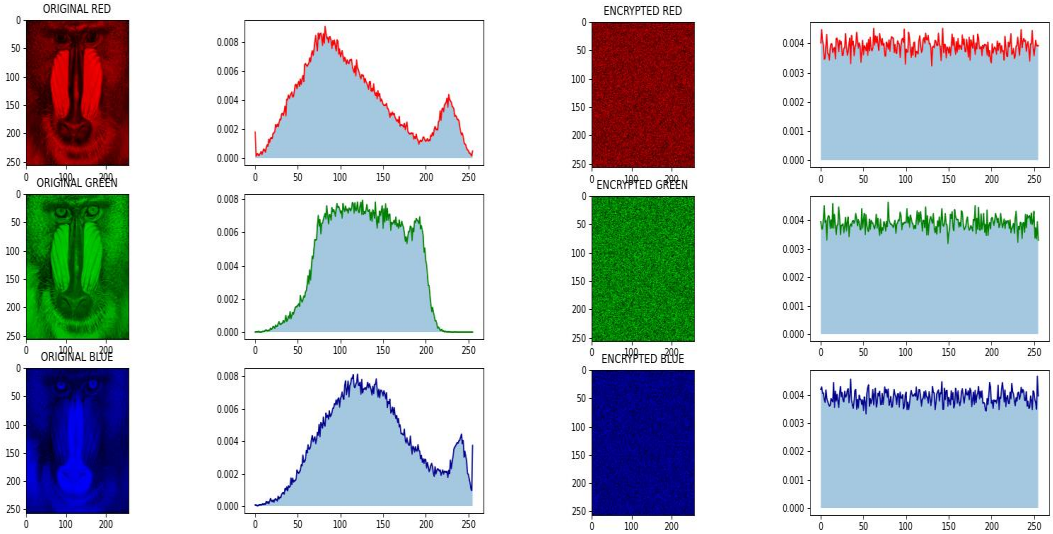
**Original image (1) Encryption image (1)**



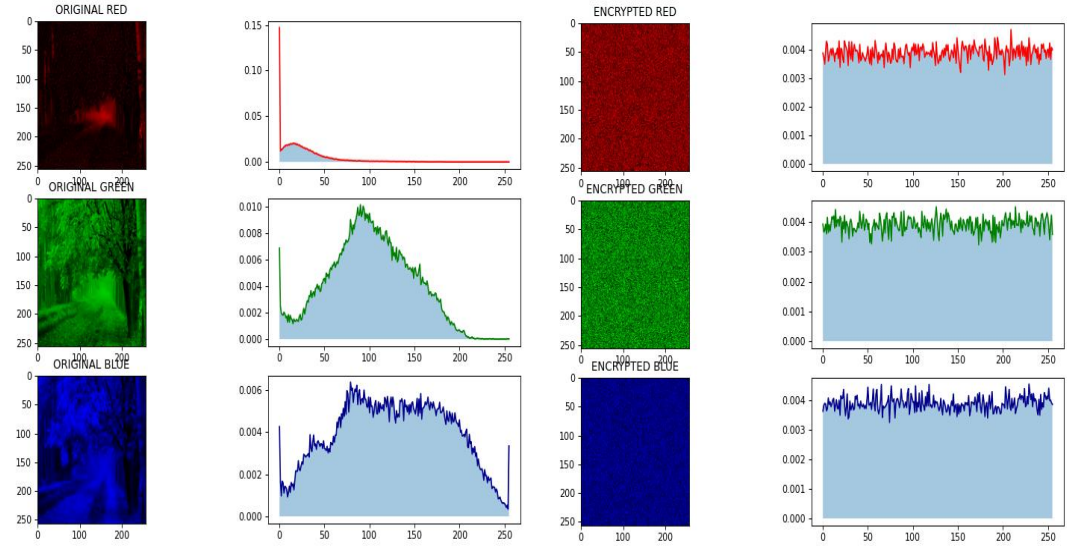

**Original image (2) Encryption image (2)**



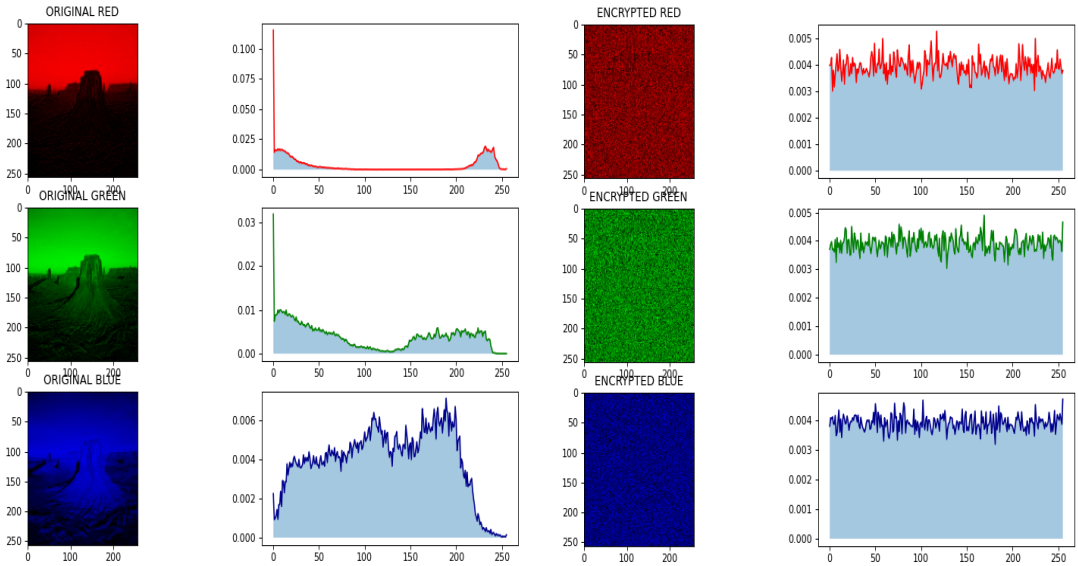

**Original image (3) Encryption image (3)**



**Original image (4) Encryption image (4)**




**Original image (5) Encryption image (5)**




**Original image (6) Encryption image (6)**

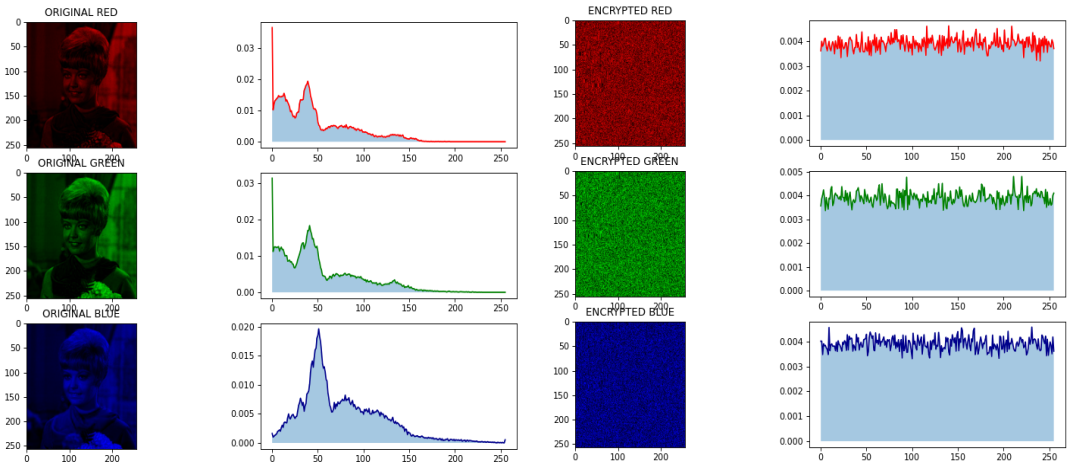


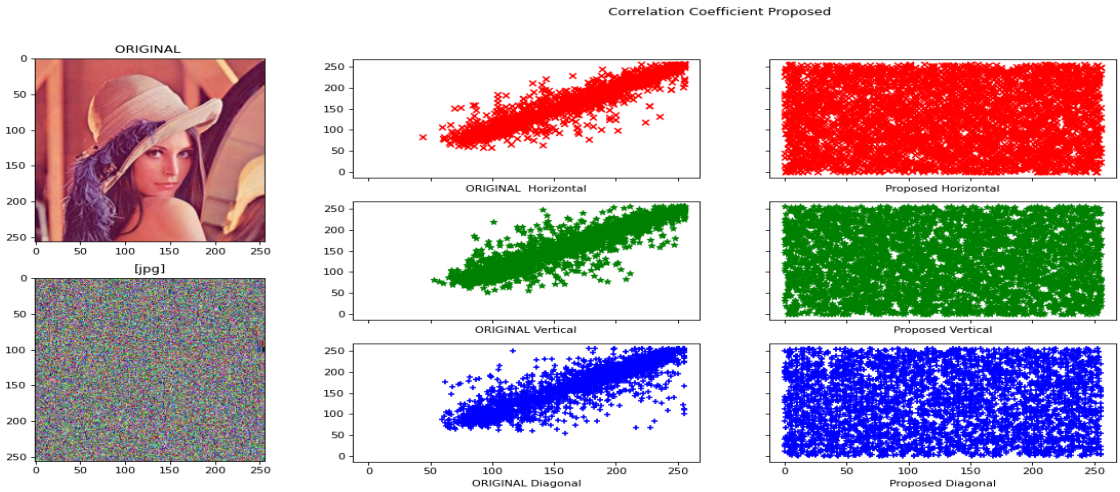

**Figure 5.** Histogram analysis

2. Correlation coefficient analysis: Specifies the relationship between the pixels of the original image and the pixels of the encryption image. The analysis comprises horizontal, diagonal, and vertical elements. The CC scale can have both negative and positive values. By applying Eq. (3), we obtained the Table 8, as shown in Figure 6, which is used to evaluate the algorithm design.

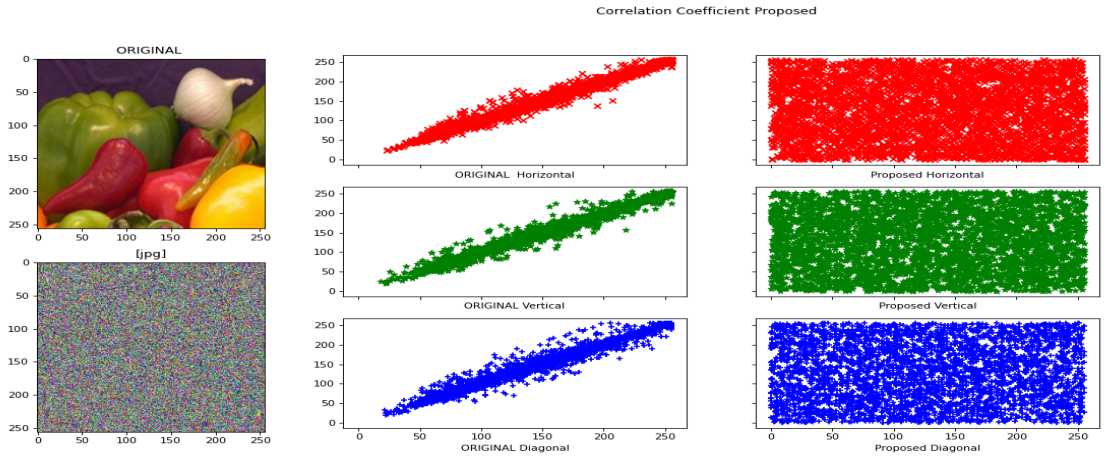
$$Cc=\frac{\Sigma_{im}\Sigma_{jn}(AA_{ijn}-\overline{AA})(BB_{mij}-\overline{BB})}{\sqrt{(\Sigma_{im}\Sigma_{jn}\,(AA_{ijn}-\overline{AA}\,)^2(\Sigma_{im}\Sigma_j(BB_{ijn}-\mathrm{B}\,\overline{\mathrm{B}})^2)}} \quad (3)$$

where, (A, B) are matrices of comparable dimensions, where ($\overline{A}$=mean (A), $\overline{B}$=mean (B)).

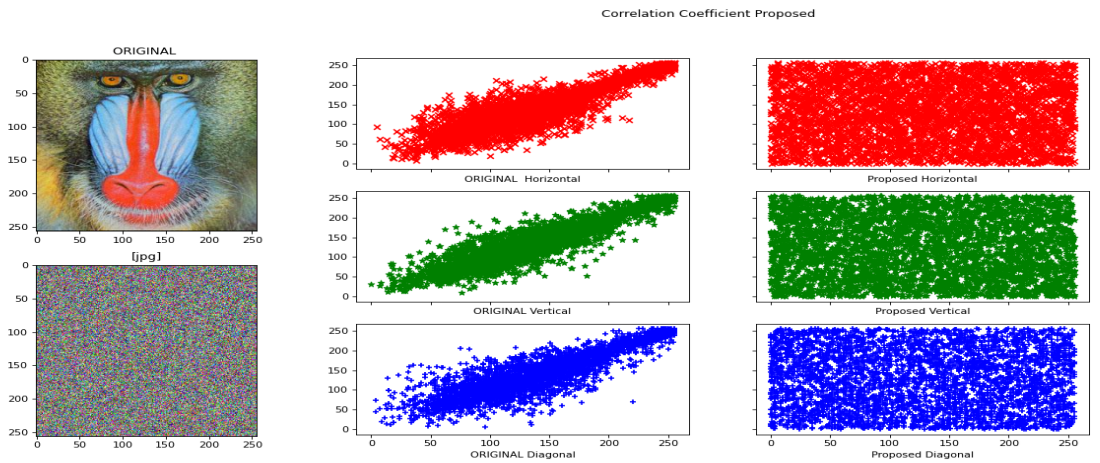
**Original image (1) Encryption image (1)**



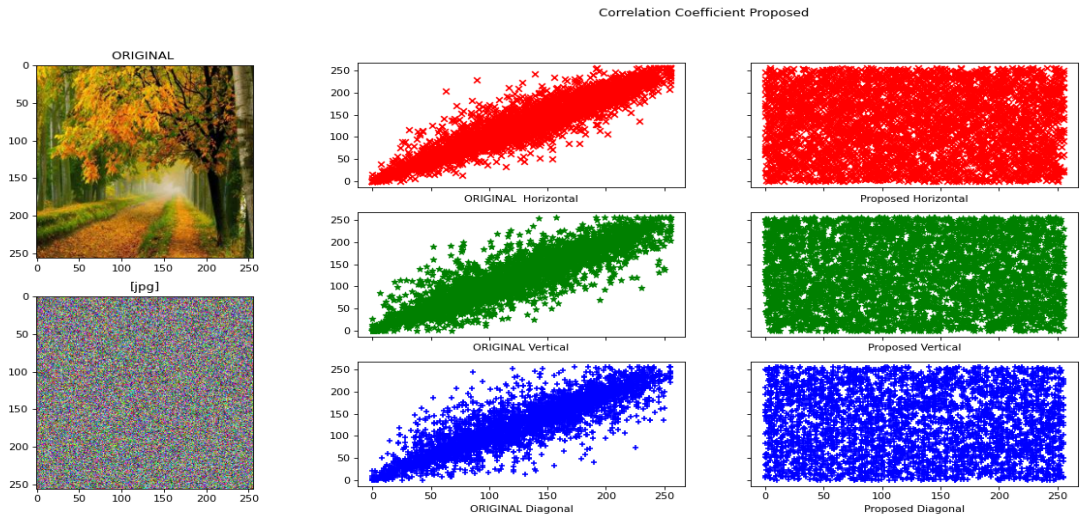

**Original image (2) Encryption image (2)**



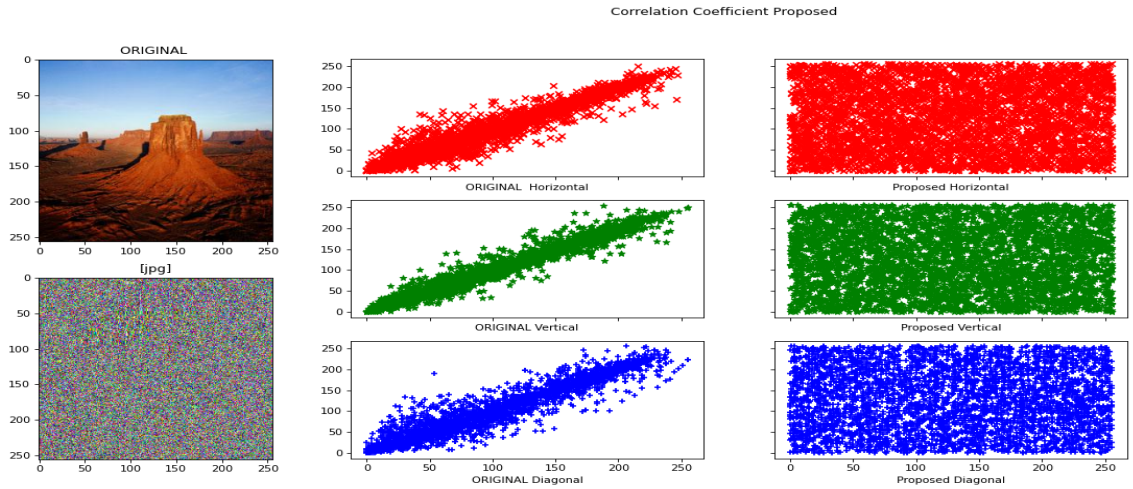

**Original image (3) Encryption image (3)**



**Original image (4) Encryption image (4)**




**Original image (5) Encryption image (5)**




**Original image (6) Encryption image (6)**

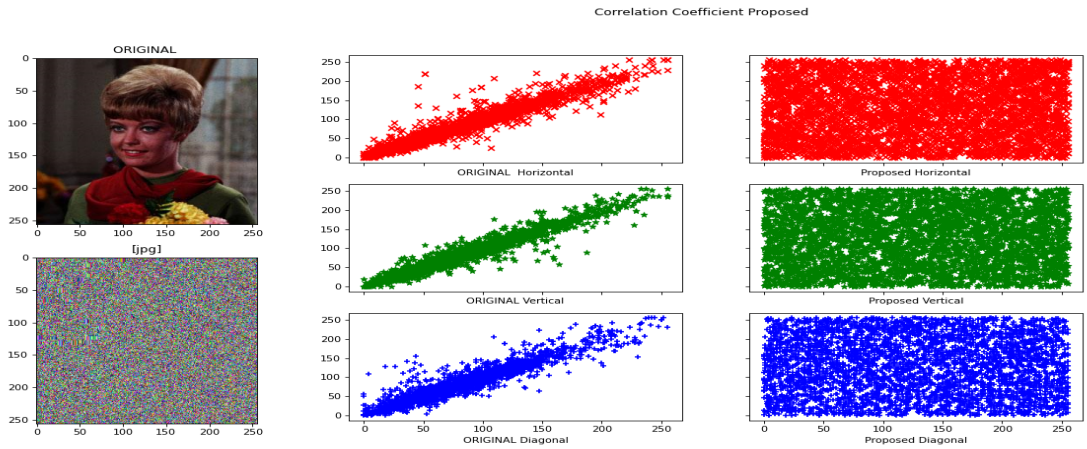



**Figure 6.** Correlation coefficient analysis

The result, a value very close to zero in all directions—horizontal, vertical, and diagonal.

The figure shows that the correlation coefficient analysis of the encryption image is close to zero for all three coordinates, which is an excellent result for resisting statistical attacks.

3. Entropy analysis: is a critical test to determine how random an image is. As Table 9 shows, after applying Eq. (4), the substitution and transposition were verified because all the values of the encrypted image that were output were from 7.99917 to 7.998684, which are close to the color scale and very close to the typical value of "8".

$$H(m) = -\sum_{i=0}^{N-1} P(m_i) log\,[p(m_i)] \quad (4)$$

where, H (m) stands for the entropy of a message, while p (mi) stands for the likelihood that the symbol will appear.

**Table 8.** Correlation coefficient analysis

| Image | Horizontal Correlation | Vertical Correlation | Diagonal Correlation |
|---|---|---|---|
| **1** | 0.001684 | -0.025513 | 0.017005 |
| **2** | -0.031362 | -0.009107 | 0.021371 |
| **3** | 0.004486 | -0.024106 | -0.044117 |
| **4** | -0.010733 | 0.014443 | -0.023371 |
| **5** | 0.019779 | -0.008594 | -0.014960 |
| **6** | 0.041261 | 0.020068 | -0.018614 |

**Table 9.** Entropy analysis

| Image | Entropy Decryption | Entropy Encryption |
|---|---|---|
| 1 | 7.773088 | 7.998954 |
| **2** | 7.698067 | 7.998684 |
| **3** | 7.69907 | 7.99917 |
| **4** | 7.625165 | 7.998988 |
| **5** | 7.684684 | 7.998184 |
| **6** | 7.179296 | 7.999037 |

The tests (histogram, entropy, and correlation coefficient) demonstrate the superior performance of the proposed algorithm.

### 5.2 Difference analysis

A good encryption method should ensure that any small update in the original image causes a noticeable difference in the encrypted image to prevent the differential attack. This paper uses NPCR (number of pixels change rate) and UACI to evaluate and analyses differential attacks. The NPCR measurement tests how one pixel change affects the entire image. Based on the math formula below: That is, as shown in Table 10 after apply the Eqs. (5) and (6).

$$\text{NPCR}=\frac{\Sigma_{ij} I(i,j)}{M \times H} \times 100\% \quad (5)$$

$$\text{UACI} =\frac{1}{M\times H}\left[\frac{\Sigma_{ij} D(i,j)-D'(i,j)}{255}\right] \times 100\% \quad (6)$$

where, before and after a single pixel change, encrypted images D and D' are presented, where L denotes the maximum supported value and T signifies the total number of pixels.

**Table 10.** Difference analysis

| Image | NPCRT | UACI |
|---|---|---|
| **1** | 99.4965% | 22.0046% |
| **2** | 99.51481% | 24.7059% |
| **3** | 99.4156% | 19.9347% |
| **4** | 99.46589% | 22.9132% |
| **5** | 99.6155% | 29.7902% |
| **6** | 99.6658% | 30.909% |

The result, UACI values range from 19.9347% to 30.909%, and NPCR values range from 99.4156% to 99.6658%, indicating a high level of sensitivity to changes in the pixels. The results of the tests show that the system is resistant to differential attacks.

### 5.3 Mean squared error (MSE) and (PSNR test) the peak signal-to-noise ratio test

Verifying error values that differentiate encrypted and unencrypted images The Mean Squared Error (MSE) can take values from 0 to infinity. PSNR assess the level of excellence in unaltered images in comparison to their encrypted equivalents. The range of Peak Signal-to-Noise Ratio (PSNR) is measured in decibels (dB) and extends from zero to infinity (∞), as shown in Table 11 after apply the Eq. (7) and (8). (En) Indicates Encryption, (De) Indicates Decryption:

$$\text{MSE}=\frac{1}{M\times N}\Sigma_{i=0}^{m-1}\Sigma_{j=0}^{n-1}[\text{A (i, j)} - \text{B (i, j)}]^2 \quad (7)$$

where, the encrypted and unencrypted pictures are denoted by A and B, respectively. Pixels in an image with dimensions m×n and coordinates (i, j).

$$\text{PSNR}=\frac{10\times \log_{10}(2xx-1)^2}{\text{MSE}} \quad (8)$$

where, the (X) represents the bit allocation per pixel.

**Table 11.** Mean squared error, the peak signal-to-noise ratio test

| Image | MSE (En) | PSNR (En) | MSE (De) | PSNR (De) |
|---|---|---|---|---|
| **1** | 8945.715332 | 8.614653 | 0 | ∞ |
| **2** | 11233.940943 | 7.625482 | 0 | ∞ |
| **3** **4** | 8212.705699 | 8.985941 | 0 | ∞ |
| **4** | 11247.550588 | 7.620224 | 0 | ∞ |
| **5** | 13062.049627 | 6.97069 | 0 | ∞ |
| **6** | 12100.111196 | 7.30291 | 0 | ∞ |

From Table 11, we notice that the MSE value is higher to get the encryption image; the value is between 11233.940943 and 13062.049627, which indicates that the restored image is difference from the original image, and the PNSR values are low. The security of an encryption system improves as the PSNR value decreases; the value between 6.97069 and 8.614653 indicates that it has better encryption quality.

(MSE) Between the original images and their decrypted counterparts. All the results indicate are "zeros", suggesting that the proposed system successfully decrypts the images without any errors.

To determine the similarity between explicit images and their corresponding decrypted images. The results for all the samples indicate a value of positive infinity, as there is no small error observed between them. The fact that equation 10 calculates the mean squared error (MSE) to be 0 supports this.

### 5.4 Analysis of key spaces

Testing is needed because the interceptor will use math and computers to find the key. Since everyone knows the algorithms, quickly generating the key breaks encryption. By complicating key finding, encryption is unbreakable. The proposed system has two symmetric keys and five parameters located 14 places after the comma of a 2D matrix.

Key Space= $\text{Possible Values}^{\text{Number of Variables}}$.

Key Space= $(10^{14})^{5}$,

Key Space= $10^{70}$.

The key space for five variables variables (c, d, e, f, and g), each with a total of $10^{14}$ potential values, is equal to $10^{70}$. This number is extremely large, indicating a wide range of possible keys. The number of bits required to represent $\log_2(10^{70}) \approx 2^{232.4}$.4 Due to its considerable length, it is impossible to employ brute force to crack this key.

### 5.5 Randomness NIST tests

After passing all 15 NIST tests, the proposed algorithm's generated sequence displayed a high level of randomness. Table 12 displays the results of each of the 15 statistical tests conducted by the National Institute of Standards and Technology (NIST). it has been determined that the sequence ratio is randomly distributed or independent of the significance value (α), with a default value of 0.01.

### 5.6 Performance measures for parallel algorithms

The degree to which a computational task or system makes use of multiple processing units or cores to accomplish tasks simultaneously is known as "parallelism," or parallel computing degree. The number of processors (speedup) equals the amount of time needed for the parallel proposed algorithm to run. Divided by the amount of time needed for the parallel algorithm to run. Speedup $S_{N.P}=T_{SA}/T_{PA}/T_{PA}$. The cost of the algorithm multiplied by the quantity of processors ($P$) determines how long it will take to execute a parallel proposed algorithm. Cost $C_P=P\times T_{PA}$. The percentage of parallel runtime that the parallel system performs well is known as the measure of efficiency or performance $E=T_{PA}/S_{N.P}$. Measure parallel overheads; $T>0$ in real life determines the parallel load "$T$." $T=P\times T_{PA}-T_{SA}$. Processors will communicate and synchronize with other processors until the parallel algorithm runs. Refers to time in parallel. $T_{PA}$ refers to time in parallel. $T_{SA}$ refers to time in sequence. As the Tables 13, 14 and 15.

**Table 12.** Randomness NIST tests

| Type of Test | P-Value | Conclusion |
|---|---|---|
| Frequency (Monobit) Test | 0.3893007386 | Random |
| Frequency Test within a Block | 0.0800080819 | Random |
| Test for the Longest Run of Ones | 0.5719362875 | Random |
| Runs Test | 0.7069189407 | Random |
| Approximate Entropy Test | 0.5595070741 | Random |
| Discrete Fourier Transform (Spectral) | 0.4117700649 | Random |
| Overlapping Template Matching Test | 0.4385673019 | Random |
| Non-overlapping Template Matching | 0.4214529724 | Random |
| linear Complexity Test | 0.2519033436 | Random |
| Serial Test | 0.8704929752 | Random |
| Cumulative Sums Test (Backward) | 0.4581922633 | Random |
| Binary Matrix Rank Test | 0.3286521908 | Random |
| Cumulative Sums Test (Forward) | 0.5115883838 | Random |
| Random Excursions Test (+1) | 0.2800034417 | Random |
| Random Excursions Variant | 0.0934478221 | Random |

The sequence is considered random because the ($P$) value is greater than 0.01, which is the significance limit.

Execution time of the proposed algorithm the parallel method on six cores takes much less time than the sequential method. From Table 13, 14, and 15. The result is that the greater the number of cores needed to run the proposed algorithm, the greater the speed, efficiency, and overhead for executing operations simultaneously and achieving optimal performance while consuming fewer resources. Performance comparisons with typical encryption algorithms [9] Our parallelized DES and AES algorithms outperform sequential processing techniques by almost 40 and 72 times, respectively.

**Table 13.** 4Core for (DES-present)

| Image | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $T_{PA}$ | 5.158788204 | 5.068117142 | 5.122981787 | 5.051019669 | 4.984307528 | 5.067356586 |
| $T_{SA}$ | 14.29676986 | 14.23963523 | 14.60504103 | 14.49023342 | 14.55853653 | 14.43378544 |
| Speedup Up | 2.771342667 | 2.80964998 | 2.850886775 | 2.868773905 | 2.920874454 | 2.848385582 |
| Efficiency | 0.692835667 | 0.806858043 | 0.712721694 | 0.717193476 | 0.730218614 | 0.712096396 |
| Cost | 20.63515282 | 17.64825344 | 20.49192715 | 20.20407868 | 19.93723011 | 20.26942634 |
| Overhead | 6.338382959 | 17.64825344 | 20.49192715 | 5.713845255 | 5.378693582 | 20.26942634 |

The time to execute the algorithm proposed in Core 4 is less than the time to execute the algorithm sequentially.

**Table 14.** 5Core for (DES-present)

| Image | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $T_{PA}$ | 4.449367285 | 4.41206336 | 4.434164762 | 4.516383648 | 4.49899292 | 4.277481556 |
| $T_{SA}$ | 14.29676986 | 14.23963523 | 14.60504103 | 14.49023342 | 14.55853653 | 14.43378544 |
| Speed up | 3.213214136 | 3.227432171 | 3.293752445 | 3.208370801 | 3.235954532 | 3.374365324 |
| Efficiency | 0.642642827 | 0.662201741 | 0.658750489 | 0.64167416 | 0.647190906 | 0.674873065 |
| Cost | 22.24683643 | 21.50346995 | 22.17082381 | 22.58191824 | 22.4949646 | 21.38740778 |
| Overhead | 7.950066568 | 21.50346995 | 22.17082381 | 8.091684819 | 7.93642807 | 21.38740778 |

The time to execute the algorithm proposed in Core 5 is less than the time to execute the algorithm sequentially.

**Table 15.** 6Core for (DES-present)

| Image | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $T_{PA}$ | 4.111061573 | 4.300693989 | 4.111302376 | 4.412790298 | 4.049250126 | 4.056567192 |
| $T_{SA}$ | 14.29676986 | 14.23963523 | 14.60504103 | 14.49023342 | 14.55853653 | 14.43378544 |
| Speed Up | 3.477634573 | 3.311008704 | 3.552412275 | 3.283689558 | 3.595366074 | 3.558128032 |
| Efficiency | 0.579605762 | 0.468274997 | 0.592068713 | 0.547281593 | 0.599227679 | 0.593021339 |
| Cost | 24.66636944 | 30.40870285 | 24.66781426 | 26.47674179 | 24.29550076 | 24.33940315 |
| Overhead | 10.36959958 | 30.40870285 | 24.66781426 | 11.98650837 | 9.736964226 | 24.33940315 |

**Table 16.** Time decryption for (DES-present)

| Image | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $T_{PA}$ (4 core) | 4.858917236 | 4.59931469 | 5.030154228 | 4.357462645 | 4.407291174 | 4.40695858 |
| $T_{PA}$ (5 core) | 3.903854847 | 3.780691385 | 3.876887083 | 3.864917755 | 3.851115465 | 3.821103096 |
| $T_{PA}$(6 core) | 3.667105436 | 3.40021944 | 3.438830376 | 3.659237385 | 3.547070742 | 3.539570332 |
| $T_{SA}$(sequence) | 14.44032121 | 14.35260201 | 14.61880398 | 14.40747929 | 14.5317452 | 14.42234516 |

### 5.7 Time decryption

The decryption time of the algorithm refers to the time it takes for authorized people to use the key to convert the encrypted image to the original image using the proposed decryption algorithm. As shown in Table 16, we conclude that the decryption time in a parallel environment is less than the decryption time for the sequential algorithm.

### 5.8 Time and space complexity of data structures and algorithms

Effective problem-solving requires understanding the time and space complexities of data structures and algorithms. Performance evaluation, scalability, optimization, and resource planning require data structure, algorithm time, and space complexity analysis. Analyzing these complexities helps us choose algorithms, optimize, and plan resources.
Time complexity is an algorithm's runtime as a function of input size. It shows how algorithm runtime increases with input size. Time complexity is usually expressed in Big O notation, which limits algorithm growth. The proposed algorithm is O ($n^2$/m), where m is the number of cores. Space complexity is the amount of memory an algorithm needs based on the size of its input. It shows how the algorithm's memory usage increases with input size. O($n^2$/m) space complexity for the proposed algorithm. While Memory usage for the algorithm as the Table 17.

**Table 17.** Memory for (DES-present)

| Image | 1 | 3 | 2 | 6 | 5 | 4 |
|---|---|---|---|---|---|---|
| **Memory** | 45.5 | 46.9 | 46.2 | 46.8 | 46.4 | 46.5 |

## 6. COMPARATIVE RESULTS

The suggested system outperforms the sequential hybrid algorithm that combines DES and Present in terms of speed thanks to its utilization of parallel processing. Because of this, it is a superior option for uses where the speed of encryption and decryption is paramount. In addition, it has been confirmed that the environment of rapid parallel processing does not introduce any new security vulnerabilities.

Comparisons with the study of Barhoush et al. [10] the rate of encryption for a changeable quantity of data units 256 represents 4 K units (256 bits). The DES algorithm takes 41 time milliseconds, DES2 (33) time milliseconds, and Parallel DES (11) time milliseconds. Furthermore, the proposed algorithm has a higher speed compared to the original DES algorithm and the original present algorithm. As in Tables 18, 19.

The encryption and decryption time for the DES algorithm takes between 23.3444781 and 24.99960899. It needs more time compared to the proposed algorithm.

The encryption and decryption time for the presnt algorithm takes between 9.39930510520 and 9.85665106773.

**Table 18.** Encryption/ decryption for original DES algorithm

| Image | Encryption | Decryption |
|---|---|---|
| **1** | 24.3756251335 | 24.191036939 |
| **2** | 25.3167097568 | 23.802267552 |
| **3** | 24.9996089935 | 24.324954986 |
| **4** | 24.2519631385 | 23.344478130 |
| **5** | 24.1495168209 | 23.813992738 |
| **6** | 24.6943865299 | 24.667293977 |

**Table 19.** Encryption / decryption for original present

| Image | Encryption | Decryption |
|---|---|---|
| **1** | 9.39930510520935 | 9.326847553253174 |
| **2** | 9.364143133163452 | 9.336862897872925 |
| **3** | 9.806873559951782 | 9.423346996307373 |
| **4** | 9.856651067733765 | 9.708191871643066 |
| **5** | 9.644702911376953 | 9.303942918777466 |
| **6** | 9.678985118865967 | 9.528454542160034 |

## 7. CONCLUSIONS

This paper demonstrates its efficacy in ensuring more secure, fast encryption and decryption color images using a parallel environment for different cores. The algorithm uses a two-dimensional chaotic system for key generation dynamism, unpredictability, and randomness. The key space($2^{232.4}$) is so big and passes all of NIST's tests that brute force can't be used to break it. The image quality is evaluated using the following standards: The NPCR values ranged from 99.4156% to 99.6658%. These values indicate a large change in the number of pixels that occurs between the original image and the encrypted image caused by the proposed encryption algorithm. A UACI value below 30.909% it indicates that the average pixel density is low between the original image and the encrypted image. All test values result Correlation coefficient analysis Close to zero value for all correlations; for example, image number 2: horizontal: -0.031362, vertical: -0.009107, diagonal: 0.021371. Encryption works well when the correlation coefficient is small, close to zero. Our encryption algorithm exhibits a significant degree of data randomness and unpredictability, which is a desirable characteristic of encryption. The increase in information entropy from the original to the encrypted image from 7.773088 to 7.998954 serves as evidence for this. The histogram of the encrypted image appears at one frequency, while the original image shows highs and lows. This is evidence of the complete difference between the original image and the encrypted image. The MES has high values between 11233.940943 and 13062.049627, which means the restored image is different from the original, and the PNSR has low values between 6.97069 and 8.614653, indicating encryption color image quality. With respect to the execution time, the result is that the encryption and decryption times of the proposed algorithm decrease with the increase in the number of cores, compared to the decryption and encryption times of the sequential algorithm (DES-Present), the original DES algorithm, and the

Present algorithm. The findings demonstrate the efficacy and feasibility of our encryption algorithm for real-world implementations. Essential for the purpose of insurance correspondence. The algorithm's encryption and decryption times are sufficiently rapid, thereby making it suitable for situations that require processing in real time or very close to real time.

## ACKNOWLEDGMENT

Al Mustansiriyah University's computer science department and college of education provided support for this research.

## REFERENCES

[1] Mehdi, S.A. (2021). Image encryption algorithm based on a novel 4D chaotic system. International Journal of Information Security and Privacy (IJISP), 15(4): 118-131. https://doi.org/10.4018/IJISP.2021100107

[2] Mehdi, S.A., Kadhim, A.A. (2019). Image encryption algorithm based on a new five dimensional Hyperchaotic system and Sudoku matrix. In 2019 International Engineering Conference (IEC), Erbil, Iraq, pp. 188-193. https://doi.org/10.1109/IEC47844.2019.8950560

[3] Mansour, I.A., Mehdi, S.A., Kuffi, E.A. (2021). On the complex SEE change and systems of ordinary differential equations. International Journal of Nonlinear Analysis and Applications, 12(2): 1477-1483. https://doi.org/10.22075/ijnaa.2021.5267

[4] Jebur, S.A., Hussein, K.A., Hoomod, H.K., Alzubaidi, L., Santamaría, J. (2022). Review on deep learning approaches for anomaly event detection in video surveillance. Electronics, 12(1): 29. https://doi.org/10.3390/electronics12010029

[5] Jasim, O.A., Hussein, K.A. (2021). A hyper-chaotic system and adaptive substitution box (S-Box) for image encryption. In 2021 International Conference on Advanced Computer Applications (ACA), Maysan, Iraq, pp. 144-149. https://doi.org/10.1109/ACA52198.2021.9626793

[6] Fernando, E., Agustin, D., Irsan, M., Murad, D.F., Rohayani, H., Sujana, D. (2019). Performance comparison of symmetries encryption algorithm AES and DES with raspberry pi. In 2019 International Conference on Sustainable Information Engineering and Technology (SIET), Lombok, Indonesia, pp. 353-357. https://doi.org/10.1109/SIET48054.2019.8986122

[7] Wu, Y.H., Dai, X.Q. (2020). Encryption of accounting data using DES algorithm in computing environment. Journal of Intelligent & Fuzzy Systems, 39(4): 5085-5095. https://doi.org/10.3233/JIFS-179994

[8] Laia, O., Zamzami, E.M., Sutarman. (2021). Analysis of combination algorithm Data Encryption Standard (DES) and Blum-Blum-Shub (BBS). Journal of Physics: Conference Series, 1898: 012017. https://doi.org/10.1088/1742-6596/1898/1/012017

[9] Xing, B., Wang, D.D., Yang, Y.Q., Wei, ZQ., Wu, J.J., He, C.H. (2021). Accelerating DES and AES algorithms for a heterogeneous many-core processor. International Journal of Parallel Programming, 49: 463-486. https://doi.org/10.1007/s10766-021-00692-4

[10] Barhoush, M., Abed-Alguni, B., Hammad, R., Al-Fawa, M., Hassan, R.N. (2022). DES22: DES-based algorithm with improved security. Jordanian Journal of Computers and Information Technology, 8(1): 18-32. https://doi.org/10.5455/jjcit.71-1632868199

[11] Mihalkovich, A., Levinskas, M., Makauskas, P. (2022). MPF based symmetric cipher performance comparison to AES and TDES. Mathematical Models in Engineering, 8(2): 15-25. https://doi.org/10.21595/mme.2022.22517

[12] Yunus, M., Sakkinah, I.S., Rahmawati, U.E., Deharja, A., Santi, M.W. (2023). File security design in electronic health record (EHR) system with triple DES algorithm (3DES) at Jember family health home clinic. International Journal of Health and Information System, 1(1): 1-8. https://doi.org/10.47134/ijhis.v1i1.2

[13] Alabdulrazzaq, H., Alenezi, M.N. (2022). Performance evaluation of cryptographic algorithms: DES, 3DES, Blowfish, Twofish, and Threefish. International Journal of Communication Networks and Information Security, 14(1): 51-61. https://doi.org/10.17762/ijcnis.v14i1.5262

[14] Akbar, M., Ahmad, I., Regula, T. (2021). Study and improved data storage in cloud computing using cryptography. International Research Journal on Advanced Science Hub, 3(2S): 94-99. https://doi.org/10.47392/irjash.2021.046

[15] Chowdhury, D., Dey, A., Garai, R., Adhikary, S., Dwivedi, A.D., Ghosh, U., Alnumay, W.S. (2023). DeCrypt: A 3DES inspired optimised cryptographic algorithm. Journal of Ambient Intelligence and Humanized Computing, 14(5): 4745-4755. https://doi.org/10.1007/s12652-022-04379-7

[16] Manohar, H. (2022). Design of distributed database system based on improved DES algorithm. Distributed Processing System, 3(4): 19-27. https://doi.org/10.38007/DPS.2022.030403

[17] Soe, T., Mon, S.S., Thu, K.A. (2019). Performance analysis of Data Encryption Standard (DES). International Journal of Trend in Scientific Research and Development, 3(5): 1439-1443. https://doi.org/10.31142/ijtsrd26650

[18] Hoomod, H.K., Naif, J.R., Ahmed, I.S. (2020). A new intelligent hybrid encryption algorithm for IoT data based on modified PRESENT-Speck and novel 5D chaotic system. Periodicals of Engineering and Natural Sciences, 8(4): 2333-2345. https://doi.org/10.21533/pen.v8i4.1738

[19] Katuk, N., Chiadighikaobi, I.R. (2022). An enhanced block pre-processing of PRESENT algorithm for fingerprint template encryption in the internet of things environment. International Journal of Communication Networks and Information Security (IJCNIS), 13(3). https://doi.org/10.17762/ijcnis.v13i3.5101

[20] Lam, T.N., Le, D.H., Cao, T.B.T. (2022). Implementation of lightweight cryptography core PRESENT and DM-PRESENT on FPGA. In 2022 International Conference on Advanced Technologies for Communications (ATC), Ha Noi, Vietnam, pp. 104-109. https://doi.org/10.1109/ATC55345.2022.9942995

[21] Kubba, Z.M.J., Hoomod, H.K. (2019). A hybrid modified lightweight algorithm combined of two cryptography algorithms PRESENT and Salsa20 using chaotic system. In 2019 First International Conference of Computer and Applied Sciences (CAS), Baghdad, Iraq, pp. 199-203. https://doi.org/10.1109/CAS47993.2019.9075488

[22] Chen, H. (2021). An enhanced encryption algorithm with key update scheme for Internet of Things. Journal of Physics: Conference Series, 1757: 012144. https://doi.org/10.1088/1742-6596/1757/1/012144

[23] Mahmood, S.A., Hussein, K.A. (2019). A parallel programming for robust chaotic map generation based on two and dimensional equation system. Journal of Engineering and Applied Sciences, 14(11): 3741-3745. https://doi.org/10.36478/jeasci.2019.3741.3745

[24] Mahmood, S.A., Hussein, K.A., Jurn, Y.N., Albahrani, E.A. (2019). Parallelizable cipher of color image based on two-dimensional chaotic system. Indonesian Journal of Electrical Engineering and Computer Science, 18(1): 101-111. http://doi.org/10.11591/ijeecs.v18.i1.pp101-111

[25] Hussein, K.A., Mehdi, S.A., Hussein, S.A. (2019). Image Encryption based on parallel algorithm via zigzag Manner with a new Chaotic system. Journal of Southwest Jiaotong University, 54(4). https://doi.org/10.35741/issn.0258-2724.54.4.29

[26] Rashid, A.A., Hussein, K.A. (2023). A lightweight image encryption algorithm based on elliptic curves and a 5D Logistic Map. Iraqi Journal of Science, 64(11): 5985-6000. https://doi.org/10.24996/ijs.2023.64.11.41