computation

MDPI

Article

Three-Dimensional and Multiple Image Encryption Algorithm Using a Fractional-Order Chaotic System

Ghader Ghasemi *,†, Reza Parvaz *,† and Yavar Khedmati Yengejeh †

Department of Mathematics, University of Mohaghegh Ardabili, Ardabil 56199-11367, Iran; khedmati.y@uma.ac.ir

* Correspondence: ghasemi@uma.ac.ir or ghghasemi1402@gmail.com (G.G.); rparvaz@uma.ac.ir (R.P.)

† These authors contributed equally to this work.

Abstract: The rapid development of communication in the last decade has heightened the necessity to create a secure platform for transferring data, including images, more than in previous years. One of the methods of secure image transmission is the encryption method. In this work, an encryption algorithm for multiple images is introduced. In the first step of the proposed algorithm, a key generation algorithm based on a chaotic system and wavelet transform is introduced, and in the next step, the encryption algorithm is developed by introducing rearrange and shift functions based on a chaotic system. One of the most important tools used in the proposed algorithm is the hybrid chaotic system, which is obtained by fractional derivatives and the Cat map. Different types of tests used to study the behavior of this system demonstrate the efficiency of the proposed hybrid system. In the last step of the proposed method, various statistical and security tests, including histogram analysis, correlation coefficient analysis, data loss and noise attack simulations, have been performed on the proposed algorithm. The results show that the proposed algorithm performs well in secure transmission.

Keywords: chaotic system; three-dimensional images; multiple-image; fractional calculation; Atangana–Baleanu fractional derivative

check for updates

Academic Editor: Michele Bonnin

Received: 18 April 2025
Revised: 6 May 2025
Accepted: 8 May 2025
Published: 10 May 2025

Citation: Ghasemi, G.; Parvaz, R.; Khedmati Yengejeh, Y. Three-Dimensional and Multiple Image Encryption Algorithm Using a Fractional-Order Chaotic System. *Computation* **2025**, *13*, 115. https://doi.org/10.3390/computation13050115

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Images play a crucial role in maintaining security, especially considering that images can contain personal information. In previous decades, the tools for taking pictures were limited to analog cameras, and keeping private images was a relatively easy task, but in recent years, with the advancement of digital imaging technology, it is rare to find a person without a mobile phone capable of taking images. Analog images are stored on photographic film, and copying requires special equipment and can be hidden in a physical environment such as a safe. But digital images have a different nature and are stored on digital devices or can be copied by pressing a key. Digital images are also widely transmitted over the Internet. Due to the differences in their structures, these types of images present different security challenges. In this paper, we focus on the security of digital images. In some cases, it is necessary to create a secure environment for a sequence of images. For example, a medical image is taken from various parts of the body and creates a sequence of images or a three-dimensional image represents different layers of the image. An algorithm that encrypts these images collectively is more efficient in terms of both security and speed compared to an algorithm that encrypts each image individually. Various articles in the last few years tried to develop the multiple image encryption algorithm using means of various tools. In [1], blurring pixels with SHA-512 are employed to produce a random

key, and then by using a new one-dimensional chaotic systems, the encryption algorithm is introduced. In [2], domino encryption strategy is used in the encryption algorithm. Radon transform and ghost imaging are introduced in [3] for the proposed algorithm. Also, quaternion discrete fractional Tchebyshev moment transform with cross-coupling operation is proposed for the encryption algorithm in [4]. However, since security can be enhanced with advancements in science, researchers continually develop different algorithms. In recent years, neural networks have been of interest to researchers in the field of security, and this tool has been used to create a secure image environment. For example, in [5], the fuzzy method and neural network have been introduced to create a secure environment for transmitting medical images. Another crucial tool employed in encryption algorithms is the utilization of chaotic systems. The primary reason for this lies in the ability of these systems to generate a sequence of numbers with a random structure, which can be reconstructed given the initial information. Although chaotic systems are a useful tool in image encryption, classical maps such as the logistic and sine maps have limitations, including a restricted chaotic range and non-uniform distribution. To address these issues, researchers have conducted various studies using different methods and tools. In [6], the chaotic system based on a fractional seed chaos generator and fractal theory is introduced; a 2D cascaded logistic map is studied in [7], a new n-dimensional conservative chaos based on a generalized Hamiltonian system and a conservative hyperchaotic system is studied in [8–10]. In the present work, the chaos system is improved based on fractional calculations and a combination with the Cat map, and then an encryption algorithm based on this chaos system is presented. The general structure of this paper is as follows: In Section 2, a 4D chaotic system is introduced using the Atangana–Baleanu fractional derivative [11]. Subsequently, this system is combined with the Cat map, and a new 2D chaotic system is introduced. The purpose of this process is to create complexity in the generation of random numbers in order to reduce the possibility of a reproduction of numbers by attackers to a minimum. In Section 3, the encryption algorithm based on the proposed chaps system for the multiple images is presented. This algorithm includes two parts: key generation and encryption. The key generation algorithm utilizes wavelet transform to reduce calculation time. Also, the algorithm is designed in such a way that some parts of this algorithm can be done in parallel to increase the speed. Since a 3D image can be converted into a multiple images by separating its layers, the proposed algorithm can be easily applied to this type of images. Following the presentation of the proposed algorithm, in Section 4, the chaotic behavior of the proposed systems is investigated by various types of tests such as statistical tests. Additionally, the security of the encryption algorithm is evaluated against different types of attacks using a range of tests.

2. Fractional-Based Hybrid Chaotic System

The concept of expanding based on Atangana–Baleanu fractional derivatives for a hybrid system was developed in [12]. In the remainder of this paper, following the presentation of preliminary definitions, this concept is generalized to introduce a chaotic system for a four-dimensional system. Subsequently, a hybrid system is created by utilizing this system and the Cat map, with the aim of generating sequences with a chaotic structure. Readers can find more information about the definitions mentioned in the next section in references related to fractional calculus, such as [13,14].

2.1. 4D Fractional Chaotic System

Definition 1. *The left Atangana–Baleanu fractional derivative for $f(x) \in H^1(a,b)$, where $b > a$ of order $\alpha \in (0,1)$ by Caputo view, is defined as*

$$ {}^{ABC}_{a}D^{\alpha}_{t} f(t) = \frac{\beta(\alpha)}{1-\alpha} \int_a^t \frac{df(s)}{ds} \varepsilon_{\alpha,1}\big(-\frac{\alpha}{1-\alpha}(t-s)^{\alpha}\big)ds, \quad t \in (a,b), \tag{1} $$

where ε denotes Mittag–Leffler function that defined by using the Euler's gamma function (Γ) as

$$ \varepsilon_{a,b}(x) := \sum_{i=0}^{\infty} \frac{x^i}{\Gamma(ai+b)}, \ \ x \in \mathbb{C}. $$

Also, in this definition, $\beta(\alpha)$ is consider as a normalization function and obtained by

$$ \beta(\alpha) = 1 - \alpha + \frac{\alpha}{\Gamma(\alpha)}, \ \ \beta(0) = \beta(1) = 1. $$

Considering the information stated in Definition 1, the left Atangana–Baleanu fractional derivative in Riemann–Liouville sense can be stated similarly as below

$$ {}^{ABR}_{a}D^{\alpha}_{t} f(t) = \frac{\beta(\alpha)}{1-\alpha} \frac{d}{ds} \int_a^t f(s)\varepsilon_{\alpha,1}\big(-\frac{\alpha}{1-\alpha}(t-s)^{\alpha}\big)ds, \quad t \in (a,b). \tag{2} $$

Definition 2. *By the conditions stated in Definition 1, the left Atangana–Baleanu fractional integral is defined as*

$$ {}^{AB}_{a}I^{\alpha}_{t} f(t) = \frac{1-\alpha}{\beta(\alpha)} f(t) + \frac{\alpha}{\beta(\alpha)} {}^{RL}_{a}I^{\alpha}_{t} f(t), \ \ t \in (a,b), \tag{3} $$

where ${}^{RL}_{a}I^{\alpha}_{t} f(t)$ is used for the Riemann–Liouville fractional integral and defined as

$$ {}^{RL}_{a}I^{\alpha}_{t} f(t) = \frac{1}{\Gamma(\alpha)} \int_a^t f(s)(t-s)^{\alpha-1}ds. $$

By using the above two definitions, the relation between Atangana–Baleanu fractional derivatives (1) and (2) and integral (3) can be expressed as [15]

$$ \begin{aligned} &{}^{ABR}_{a}D^{\alpha}_{t}\big({}^{AB}_{a}I^{\alpha}_{t} f(t)\big) = {}^{AB}_{a}I^{\alpha}_{t}\big({}^{ABR}_{a}D^{\alpha}_{t} f(t)\big) = f(t), \\ &{}^{AB}_{a}I^{\alpha}_{t}\big({}^{ABC}_{a}D^{\alpha}_{t} f(t)\big) = f(t) - f(a). \end{aligned} \tag{4} $$

Now, we continue to introduce a chaotic system based on the discussed topics. Consider the following system of Atangana–Baleanu fractional differential equations with the initial conditions $x(0) = x_0, y(0) = y_0, z(0) = z_0$ and $k(0) = k_0$.

$$ \begin{cases} {}^{ABR}_{a}D^{\alpha}_{t} x(t) = \Phi_1\big(x(t), y(t), z(t), k(t)\big), \\ {}^{ABR}_{a}D^{\alpha}_{t} y(t) = \Phi_2\big(x(t), y(t), z(t), k(t)\big), \\ {}^{ABR}_{a}D^{\alpha}_{t} z(t) = \Phi_3\big(x(t), y(t), z(t), k(t)\big), \\ {}^{ABR}_{a}D^{\alpha}_{t} k(t) = \Phi_4\big(x(t), y(t), z(t), k(t)\big). \end{cases} \tag{5} $$

For the simplicity, system (5) is written as ${}^{ARB}_{a}D^{\alpha}_{t}X(t) = \Upsilon\big(X(t)\big)$, where $\Upsilon := \big(\Phi_1, \Phi_2, \Phi_3, \Phi_4\big)$ and $X(t) := \big(x(t), y(t), z(t), k(t)\big)$. In this system, the functions $\Phi_i \ (i = 1, \ldots, 4)$ are selected as the same maps that are introduced in [16] as follows

$$\Phi^r_1(x,y,z,k) := \begin{cases} \alpha^x_1 f^x_1 \circ F^x_1(r,x) + g^x_1(r,x,y,z,k) \\ \qquad + h^x_1\big(\frac{(\beta^x_1 - r)z}{2}\big) \mod 1, \ \textit{when } k < 0.5, \\ \\ \alpha^x_2 f^x_2 \circ F^x_2(r,x) + g^x_2(r,x,y,z,k) \\ \qquad + h^x_2\big(\frac{(\beta^x_2 - r)(1-z)}{2}\big) \mod 1, \ \textit{when } k \geq 0.5, \end{cases} \tag{6}$$

$$\Phi^r_2(x,y,z,k) := \begin{cases} \alpha^y_1 f^y_1 \circ F^y_1(r,y) + g^y_1(r,x,y,z,k) \\ \qquad + h^y_1\big(\frac{(\beta^y_1 - r)\zeta_1}{2}\big) \mod 1, \ \textit{when } \zeta_1 < 0.5, \\ \\ \alpha^y_2 f^y_2 \circ F^y_2(r,y) + g^y_2(r,x,y,z,k) \\ \qquad + h^y_2\big(\frac{(\beta^y_2 - r)(1-\zeta_1)}{2}\big) \mod 1, \ \textit{when } \zeta_1 \geq 0.5, \end{cases} \tag{7}$$

$$\Phi^r_3(x,y,z,k) := \begin{cases} \alpha^z_1 f^z_1 \circ F^z_1(r,z) + g^z_1(r,x,y,z,k) \\ \qquad + h^z_1\big(\frac{(\beta^z_1 - r)\zeta_2}{2}\big) \mod 1, \ \textit{when } \zeta_2 < 0.5, \\ \\ \alpha^z_2 f^z_2 \circ F^z_2(r,z) + g^z_2(r,x,y,z,k) \\ \qquad + h^z_2\big(\frac{(\beta^z_2 - r)(1-\zeta_2)}{2}\big) \mod 1, \ \textit{when } \zeta_2 \geq 0.5, \end{cases} \tag{8}$$

$$\Phi^r_4(x,y,z,k) := \begin{cases} \alpha^k_1 f^k_1 \circ F^k_1(r,k) + g^k_1(r,x,y,z,k) \\ \qquad + h^k_1\big(\frac{(\beta^k_1 - r)\zeta_3}{2}\big) \mod 1, \ \textit{when } \zeta_3 < 0.5, \\ \\ \alpha^k_2 f^k_2 \circ F^k_2(r,k) + g^k_2(r,x,y,z,k) \\ \qquad + h^k_2\big(\frac{(\beta^k_2 - r)(1-\zeta_3)}{2}\big) \mod 1, \ \textit{when } \zeta_3 \geq 0.5, \end{cases} \tag{9}$$

where parameter $r \in (0,4]$, $\zeta_i (i = 1,2,3)$ is chosen as x, y, z and F^{τ}_{ς} for $\tau = x, y, z, k$, $\varsigma = 1, 2$ are selected as Sin or Logistic maps. $\alpha^{\tau}_{\varsigma}, \beta^{\tau}_{\varsigma}$ for $\tau = x, y, z, k, \varsigma = 1, 2$ are constant numbers in $\mathbb{R}$, and $g^{\tau}_{\varsigma}, h^{\tau}_{\varsigma}$ are considered as sufficiently smooth functions. By taking $[0, +\infty) = \cup_{i=0}^{+\infty} I_i$, where $I_i := [ih, (i+1)h)$, $h > 0$, and using the method described in [17], the discrete solution of this system can be found as follows. In the first step, let ${}^{ABR}_{a}D^{\alpha}_{t}X(t) = \Upsilon\big(X(h\lfloor \frac{t}{h} \rfloor)\big)$, then for $t \in I_n$, we obtain

$${}^{ABR}_{a}D^{\alpha}_{t}X(t) = \Upsilon\big(X_n(nh)\big). \tag{10}$$

Assuming that $t = (n+1)h$ and using (4) and (10), the final solution is found as follows:

$$X_{n+1} = \Big(\frac{1-\alpha}{\beta(\alpha)} + \frac{h^{\alpha}}{\beta(\alpha)\Gamma(\alpha)}\Big)\Upsilon\big(X_n\big),$$

The behavior of this chaotic system is evaluated in Section 4.2. In the following, the output of this system for input values α and r is shown by Ψ_r^α.

2.2. Hybrid Chaotic System

In the following, a hybrid chaotic system is introduced using the fractional chaotic system from the previous subsection and the Cat map. The most important reason for producing this hybrid system is to create more complexity on the sequence produced in the previous section. This method increases the complexity of the proposed image encryption algorithm. In the proposed system, the Cat map is used. This map has been studied in various papers, and various methods have been provided to calculate this map. For example, see [18–20]. The multidimensional generalization Cat map can be written as

$$X_{n+1} = C^{nD} X_n \mod 1, \tag{11}$$

where in (11)

$$C^{nD} = \begin{bmatrix} I & P_a \\ P_b & I + P_b P_a \end{bmatrix}.$$

In this matrix, $P_a \in \mathbb{R}^{m\times(n-m)}$, $P_b \in \mathbb{R}^{(n-m)\times m}$ and $I \in \mathbb{R}^{m\times m}$ are the identity matrices. By simple calculations, it can be seen that $\det(C^{nD}) = 1$. In the proposed hybrid chaos, C^{2D} is used as follows.

$$C^{2D} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

The sequence production steps are presented in the following steps.

Step 1. Consider $X_0, r, n, h, C^{2D}, i = 0$ and α as input values.

Step 2. Define $X_i^j := X_i$, $j = 1$ and run the following algorithm.

---

repeat
  $Y_j := C^{2D} X_i^j$;
  $Z_j := \Psi_r^{\frac{\alpha}{5}}\left(\Psi_r^\alpha([X_i^j, Y_j])\right)$;
  $X_i^j(1) = \mod\left(Z_j(1) + Z_j(2), 1\right)$;
  $X_i^j(2) = \mod\left(Z_j(3) + Z_j(4), 1\right)$;
  $j = j + 1$;
until $j \leq 2$

Step 3. Consider $i = i + 1$, $X_i = X_i^2$.

Step 4. If $i = n$, stop, orelse go to Step 2.

The effect of this algorithm on the fractional-based chaotic system and the analysis of the behavior of the hybrid system are evaluated in the next subsection. Additionally, a sequence of the proposed hybrid system with n elements is shown by

$$[X_0 \ \cdots \ X_n] = \Omega_{h,n}^{\alpha,r}(X_0).$$

The structure of the proposed hybrid system is shown in Figure 1.

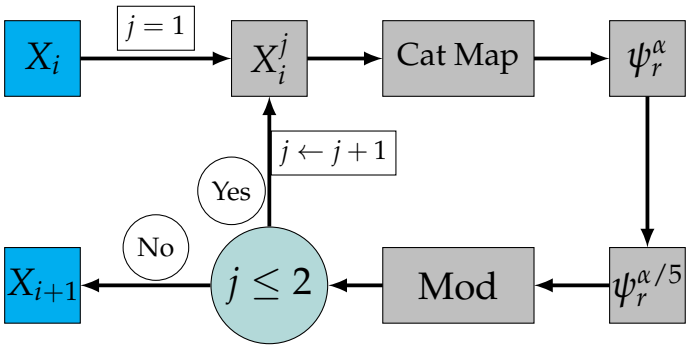


Figure 1. Flowchart of the proposed hybrid chaotic system.

3. 3D and Multiple Image Encryption Algorithm

To create a secure environment for image transfer, we need two components: one involving the key, and the other involving the transformation of the original data into unrecognizable data. In the discussion of data security, a secure path exists for the transmission of cryptographic keys. These keys are transferred between the sender and receiver of the message through this path. However, it is important to note that this path has a limited transfer volume and, as a result, cannot be utilized for transferring original data. In the following, first, an algorithm for generating the key space is introduced. The generated key space should not have a large volume to ensure easy transferability. Additionally, it should also be sensitive to small changes in the input image.

3.1. The Key of Encryption Algorithm

The keys utilized for encryption algorithms are categorized as symmetric and asymmetric. In the proposed algorithm, the symmetric key is used for encryption and decryption. The key used can be specified as a number of our own choice, and this fixed number can be used for encryption. However, one of the most secure methods is to choose the keys in a way that depend on the input data. This allows the key to be changed with small variations in the input data. Due to the fact that a large number of images are used in the proposed algorithm, the time taken to obtain the key should not be too long. Also, the keys should be produced as quickly as possible and should be sensitive to small changes. The flowchart of the key generation algorithm is illustrated in Figure 2. In the key space generation algorithm, some parts are designed in such a way that these parts can be calculated in parallel. If there is a parallel calculation system, the calculation time can be reduced. These specific parts are highlighted in the flowchart. The key generation algorithm is written in the following steps. Consider a sequence of images as $\{I_i\}_{i=1}^k$, where $I_i \in \mathbb{R}^{n_i \times m_i}$, and the input values α, r, and h for the chaotic system.

Step 1. In this step, in order to reduce the volume of calculations, we reduce the size of the image used in the calculations by wavelet transformation. For $i = 1, \cdots, k$, derive $I_i^{ll} \in \mathbb{R}^{n_i' \times m_i'}$ as the low–low pass filter of the Daubechies wavelet transform at level 5 (db5) for I_i.

Step 2. Obtain the sequence $\{(v_i^x, v_i^y)\}_{i=1}^{k}$ by using the following formulae.

$$v_i^x := \frac{4}{256 n_i' m_i'} \sum_{j=1}^{\lfloor \frac{n_i'}{2} \rfloor} \sum_{l=1}^{\lfloor \frac{m_i'}{2} \rfloor} I_i^{ll}(j,l),$$

$$v_i^y := \frac{4}{256 n_i' m_i'} \sum_{j=\lfloor \frac{n_i'}{2} \rfloor + 1}^{n_i'} \sum_{l=\lfloor \frac{m_i'}{2} \rfloor + 1}^{m_i'} I_i^{ll}(j,l).$$

Changes in the pixels of the input image cause the values of these parameters to vary, which affects other parts of the proposed algorithm and leads to changes in the final encrypted image.

Step 3. Define $\omega^x := \sum_{i=1}^{k} v_i^x$ and $\omega^y := \sum_{i=1}^{k} v_i^y$.

Step 4. Calculate the c^a and $c^b \in \mathbb{R}^{1\times 7}$ using the following algorithm.

$c^a = 0, c^b = 0$;
for $j = 1 : k$ **do**
  $a_{j,1}^x := v_j^x$, $a_{j,1}^y := v_j^y$;
  if $j = 1$ **then**
    $b_{j,1}^x := \omega^x$, $b_{j,1}^y := \omega^y$;
  else
    $b_{j,1}^x := a_{j-1,7}^x$, $b_{j,1}^y := a_{j-1,7}^y$;
  end if
  $[a_{j,1:7}^x; b_{j,1:7}^x] := \Omega_{h,7}^{\alpha,r}(a_{j,1}^x, b_{j,1}^x)$;
  $[a_{j,1:7}^y; b_{j,1:7}^y] := \Omega_{h,7}^{\alpha,r}(a_{j,1}^y, b_{j,1}^y)$;
  $c^a = \frac{1}{3}\left(c^a + a_{j,1:7}^x + b_{j,1:7}^y\right)$;
  $c^b = \frac{1}{3}\left(c^b + a_{j,1:7}^y + b_{j,1:7}^x\right)$;
end for

Step 5. By using the bitxor operator ($\oplus$), obtain $\{x_1, x_2, y_1, y_2\}$ as

$$[x_1, x_2] := \frac{1}{2\times 10^{15}}\Big[\bigoplus_{i=2}^{4} \lfloor c_i^a \times 10^{15} \rfloor, \ \bigoplus_{i=5}^{7} \lfloor c_i^a \times 10^{15} \rfloor\Big],$$

$$[y_1, y_2] := \frac{1}{2\times 10^{15}}\Big[\bigoplus_{i=2}^{4} \lfloor c_i^b \times 10^{15} \rfloor, \ \bigoplus_{i=5}^{7} \lfloor c_i^b \times 10^{15} \rfloor\Big].$$

After performing the above steps, the final key space for the encryption algorithm is considered as follows

$$\text{Key} = \{\alpha, r, h, x_1, x_2, y_1, y_2\}. \tag{12}$$

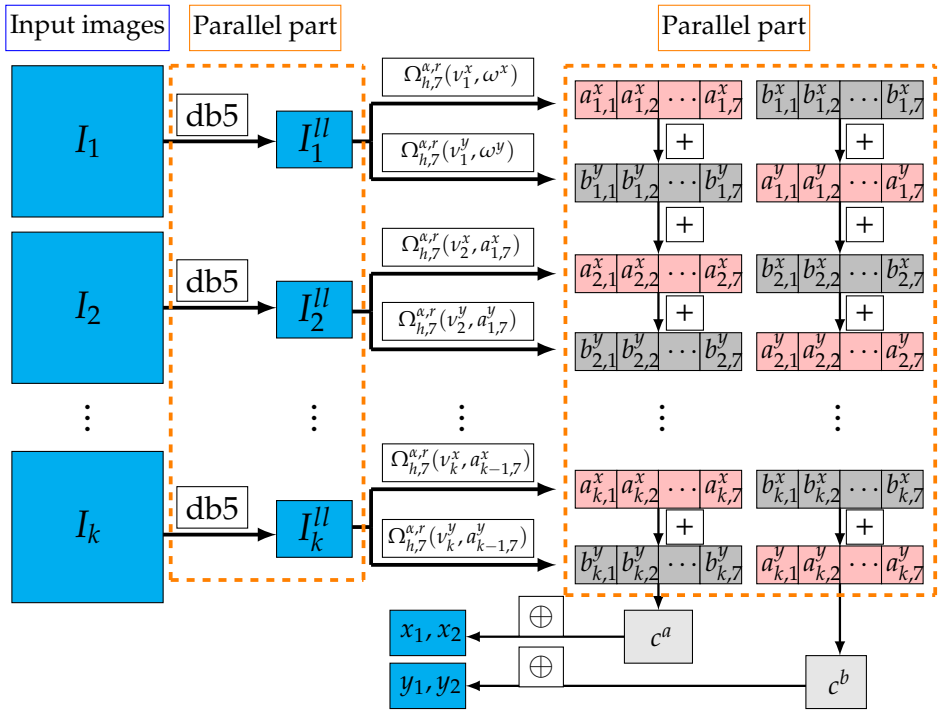


Figure 2. Flowchart of the proposed key generation algorithm.

3.2. Proposed Shift and Rearrange

Rearranging pixels of images at the appropriate time is a critical step in image encryption algorithms. In this subsection, we introduce two straightforward and efficient methods, one based on shifting image pixels and the other on swapping image pixels. For the first method, we begin by creating a matrix using a permutation that contains exactly one non-zero element (equal to one) in each row and column (Algorithm 1). Multiplying this matrix with any other matrix leads to shifting the elements of that matrix. In Algorithm 2, the pixels of each layer in the multilayer matrix undergo shifts using distinct permutations. It is noteworthy that in this context, the layer corresponding to each pixel remains unchanged. To illustrate the operation of this function, refer to Figures 3 and 4b.

Algorithm 1 PermutationMatrix

Input: A permutation $P \in \mathbb{N}^{1\times n}$
Output: A permutation matrix $M \in \{0,1\}^{n\times n}$

$M = \text{zerose}(n)$
for $i = 1, \ldots, n$ **do**
    $M(i, P(i)) = 1$
end for
return the matrix M

Algorithm 2 Shift

Input: Matrices $A_1, \ldots, A_l \in \mathbb{R}^{n\times n}$ and permutations $P_1, \ldots, P_l \in \mathbb{N}^{1\times n}$
Output: Matrices $B_1, \ldots B_l \in \mathbb{R}^{n\times n}$

for $i = 1, \ldots, l$ **do**
    $M_l \leftarrow \text{PermutationMatrix}(P_l)$
    $B_l = M_l A_l M_l$
end for
return Matrices $B_1, \ldots B_l$

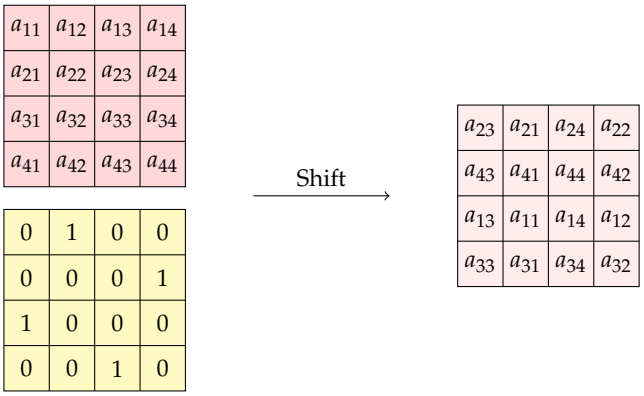


Figure 3. Illustration of Function 2 execution.

In the second method, pixel locations are altered based on permutations. As the first step, Algorithm 3 is defined. Then compared to the first method, this approach induces more disturbance in the neighborhood of pixels (Algorithm 4). To achieve a more favorable outcome, Algorithm 5 is introduced by incorporating the application of Algorithm 3 to combine the layers of a multilayer matrix. This involves not only changing the rows and columns within each layer but also modifying the arrangement of the layers themselves. To illustrate the operation of this function, refer to Figures 4c and 5. Figure 4d shows the optimal performance of Shift and Rearrange3D when applied consecutively. Note that the second step in Figure 5 is based on the following permutation matrices.

$$P_1 = \begin{bmatrix} 8 & 11 & 1 \\ 6 & 9 & 2 \\ 7 & 10 & 12 \\ 5 & 3 & 4 \end{bmatrix}, \qquad P_2 = \begin{bmatrix} 8 & 9 & 2 \\ 6 & 10 & 11 \\ 5 & 1 & 3 \\ 12 & 7 & 4 \end{bmatrix}.$$

Algorithm 3 RowCol

Input: Natural umbers n, m
Output: Natural numbers $i, j \in [1, n]$

$q = \text{floor}(m/n)$
$r = \text{mod}(m, n)$
if $r = 0$ **then**
    $(i, j) = (q, n)$
else
    $(i, j) = (q + 1, r)$
end if
return Natural numbers i, j

Algorithm 4 Rearrange

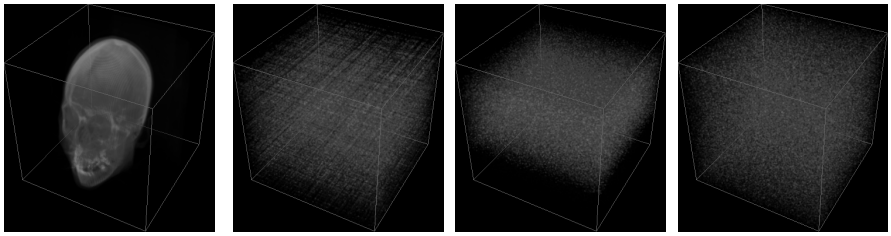
Input: A matrix $A \in \mathbb{R}^{n \times m}$ and a permutation $P \in \mathbb{N}^{n \times m}$
Output: A matrix $A_P \in \mathbb{R}^{n \times m}$

for $i = 1, \ldots, n$ **do**
    for $j = 1, \ldots, m$ **do**
        $(v, w) \leftarrow \text{RowCol}\,(m, P(i, j))$
        $A_p(i, j) = A(v, w)$
    end for
end for
return Matrix A_P

Algorithm 5 Rearrange3D

Input: A multilayer matrix $A \in \mathbb{R}^{n\times m\times l}$ and a permutation $P \in \mathbb{N}^{n\times l\times m}$
Output: A matrix $A_P \in \mathbb{R}^{n\times m\times l}$

for $k = 1, \ldots, m$ **do**
    $B(:,:,k) \leftarrow \text{Rerrange}\left(A(:,k,:), P(:,k,:)\right)$
end for
for $k = 1, \ldots, l$ **do**
    $A_p(:,:,k) = B(:,k,:)$
end for
return Matrix A_P



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

Figure 4. Execution of Shift and Rearrange on an image: (**a**) 3D plain image, (**b**) execution of Shift on the plain image, (**c**) execution of Rearrange3D on the plain image, (**d**) execution of Shift and Rearrange, consecutively.

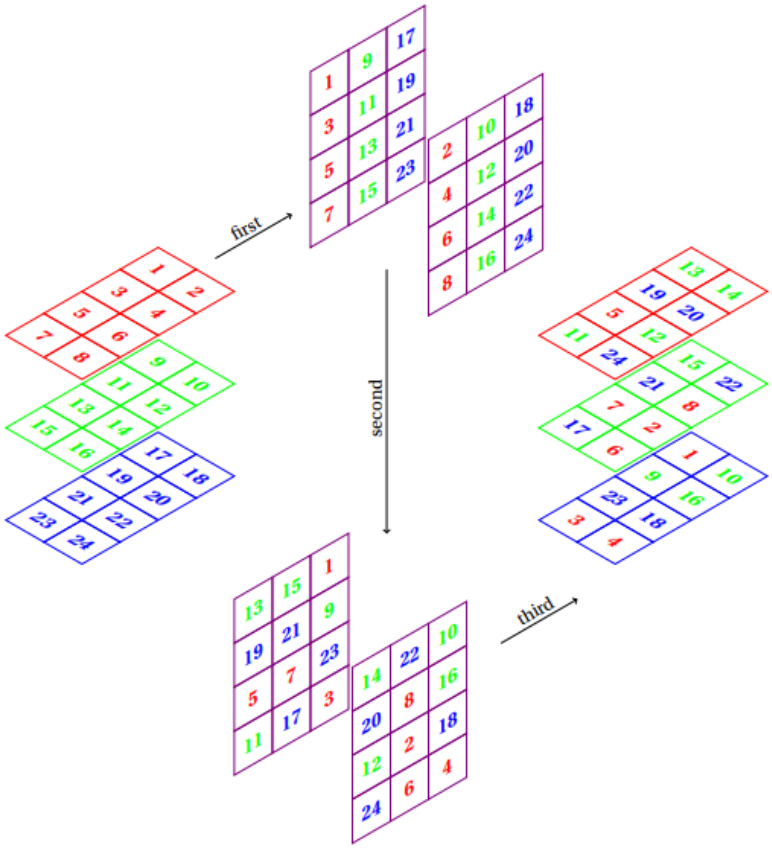


Figure 5. Illustration of Function Rearrange3D execution.

It is worth noting that both defined operators are invertible, and their inverses are denoted by adding "Inv" to the beginning of their titles.

3.3. Proposed Encryption Algorithm

In this subsection, an encryption algorithm for multiple images is presented using the concepts introduced in the previous sections. Before presenting the encryption algorithm, the following notation is introduced. Let

$$\begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix} = \Omega_{h,n}^{\alpha,r}([x_0; y_0]),$$

then $\overline{\Omega}_{h,n}^{\alpha,r}([x_0; y_0])$ is defined as

$$\overline{\Omega}_{h,n}^{\alpha,r}([x_0; y_0]) := [x_1 \ \ldots \ x_n \ y_1 \ \ldots \ y_n].$$

Now, the encryption algorithm is being introduced. Consider $\{I_i\}_{i=1}^k$ as a sequence of n-by-m input images. In the initial phase of the algorithm, these images enter the key generation algorithm, and the key space is considered as (12). Subsequent to this step, the remaining steps of the algorithm are delineated as follows:

Step 1. By using the input keys and hybrid chaotic system, obtain the vector $v^1 \in \mathbb{R}^{1\times 2\vartheta}$ with $\vartheta := \lfloor \frac{nm}{4} \rfloor$ as follows:

$$v^1 = \overline{\Omega}_{h,\vartheta}^{\alpha,r}([x_1; y_1]),$$

After performing these calculations, reshape the vector v^1 into an n-by-m matrix, denoted as V^1.

Step 2. Obtain the sequence $\{I_i^1\}_{i=1}^k$ as

$$I_i^1 = I_i \oplus V_i^c,$$

where

$$V_i^c := \begin{cases} V^1 & i = 1, \\ \text{circshift}\big(V^1, [v_i^1 \ \ v_{\vartheta+i}^1]\big) & i = 2, \ldots, k. \end{cases}$$

In the above formulas, "circshift" represents circular shifts.

Step 3.

(a) By using input keys x_2 and y_2 obtain $v^2 \in \mathbb{R}^{1\times 2\varrho}$ with $\varrho = \lfloor \frac{kn}{2} \rfloor$ as

$$v^2 = \overline{\Omega}_{h,\varrho}^{\alpha,r}([x_2; y_2]).$$

(b) Calculate $P^1 \in \mathbb{N}^{k\times n}$ as follows:

$$P^1(i,:) := \begin{cases} \text{Index}_{\text{sort}}\Big(v^2\big(1, (\frac{i}{2}-1)n+1 : \frac{i}{2}n\big)\Big), & \text{when } i \text{ is even}, \\ \text{Index}_{\text{sort}}\Big(v^2\big(1, (\frac{i+1}{2}-1)n+1 : \frac{i+1}{2}n\big)\Big), & \text{when } i \text{ is odd}. \end{cases}$$

In the above formula, $\text{Index}_{\text{sort}}$ represents a permutation of indices in $\mathbb{N}$, where the elements in this vector indicate the positions of the elements after sorting them from the smallest to the largest values.

(c) Consider $\{P^1(i,:)\}_{i=1}^k$ and $\{I_i^1\}_{i=1}^k$ as input values for shift Algorithm 2 and obtain $\{I_i^2\}_{i=1}^k$ as output of this function.

Step 4.

(a) Obtain $v^3 \in \mathbb{R}^{1\times 2\varrho}$ as

$$v^3 = \overline{\Omega}_{h,\varrho}^{\alpha,r}([x_1; y_2]).$$

(b) Define $P^2 \in \mathbb{N}^{1\times kn}$ as

$$P^2 = \text{Index}_{\text{sort}}(v^3).$$

and reshape the P^2 into an n-by-k matrix, denoted as V^2.

(c) Define $M^v \in \mathbb{N}^{m\times n\times k}$ as

$$M^v(i,:,:) := \begin{cases} V^2, & i = 1, \\ \text{circshift}\Big(M^v(i-1,:,:), \big[\lfloor v_i^3 \times 10^2 \rfloor \;\; \lfloor v_{i+1}^3 \times 10^2 \rfloor\big]\Big), & i = 2,\ldots,m. \end{cases}$$

(d) Define a multilayer matrix $A(:,:,i) := I_i^2$, $i = 1, \cdots, k$, and then consider M^v and A as input values for Rearrange3D Algorithm 5 and obtain final encrypted images $\{I_i^e\}_{i=1}^k$.

A flowchart illustrating the proposed encryption algorithm is presented in Figure 6. The decryption algorithm is precisely the inverse of the encryption algorithm; therefore, the details of the decryption algorithm are not studied. When considering color images, each color space is selected as a separate input image for the algorithm. If the input image is three-dimensional, it is initially divided into two-dimensional images corresponding to the slices of the image. Each of these images is then treated as an input for the algorithm. This process is implemented using the `Image3DSlices` command in Wolfram Mathematica software 9.0.1.0

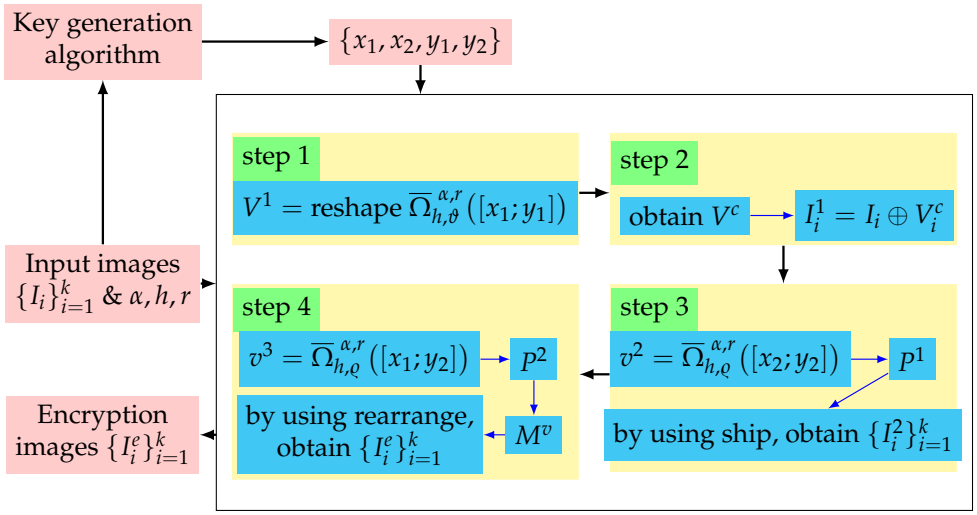


Figure 6. Flowchart of the proposed encryption algorithm.

4. Simulation Results and Security Analysis

In this section, the security of the algorithm presented in the previous section is investigated by means of different types of security tests.

4.1. Simulation Infrastructure

As mentioned in the previous sections, the algorithm is designed to operate on multiple images. The number of images it can process depends on the available computational resources. On systems with higher processing power and more memory, the proposed algorithm is capable of handling a larger number of images simultaneously. To generate numerical results, a Windows-10-based system equipped with an Intel(R) Core(TM)

i3-5005U CPU running at 2.00 GHz and 4.00 GB RAM is employed. The simulations are conducted in the MATLAB R2014b and Wolfram Mathematica 9 programming environments. Additionally, the images utilized in this section were sourced from USC-SIPI (available at http://sipi.usc.edu/database/(accessed on 18 April 2025)), and the information-hiding homepage (available at https://www.petitcolas.net/steganography/index.html (accessed on 18 April 2025)), and internal 3D images were sourced from Wolfram Mathematica software.

4.2. Analysis of the Proposed Systems

There are various tests to investigate the behavior of a chaotic system. In this paper, the Lyapunov exponent, bifurcation diagram, histogram analysis and sensitivity to the initial point are used to investigate the behavior of the chaotic system. The proposed 4D hybrid chaotic system with the following input values is used in the relevant calculations in this paper.

- $\{\alpha_1^x, \alpha_2^x, \alpha_1^y, \alpha_2^y, \alpha_1^z, \alpha_2^z, \alpha_1^k, \alpha_2^k\} = \{30, 20, 20, 20, 10, 20, 10, 20\}$.
- $\{\beta_1^x, \beta_2^x, \beta_1^y, \beta_2^y, \beta_1^z, \beta_2^z, \beta_1^k, \beta_2^k\} = \{6, 9, 4, 20, 4, 7, 4, 7\}$.
- $\{\zeta_1, \zeta_2, \zeta_3\} = \{x, y, z\}$.
- $f_1^x(p) = \cosh(p)$, $f_2^x(p) = f_1^y(p) = f_1^z(p) = f_2^k(p) = p$, $f_2^y(p) = \cos(p)$, $f_2^z(p) = f_1^k(p) = \sin(p)$.
- $g_1^x = 2\sin(rx) + 2\cosh(20rz + k)$, $g_2^x = -3\sinh(\pi rx) + \cos(xyz)$, $g_1^y = 2\cos(rx + y) - \exp(y + z^2) + k$, $g_2^y = z^2 + \sin(k) + 2\sinh(\pi ry)$, $g_1^z = 2\exp(rz + y + z + k)$, $g_2^z = 2ry + \exp(2rxyk)$, $g_1^k = 2\exp(rz + y + z) + \cos(rk)$, $g_2^k = \sin(2rx + y)$.
- $h_1^x(p) = -2p, h_2^x(p) = h_2^y(p) = 4p, h_1^y(p) = h_1^z(p) = h_1^k(p) = 2p$, $h_2^z(p) = \cosh(4p)$, $h_2^k(p) = \exp(4p)$.

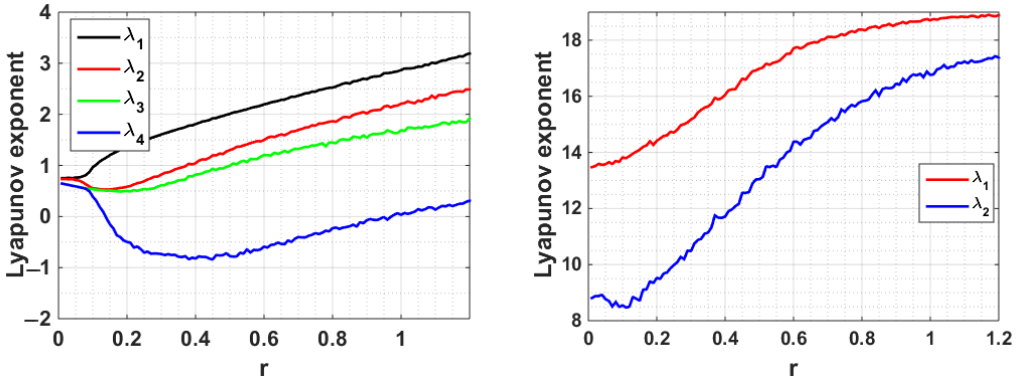
The Lyapunov exponent for a chaotic system as F is obtained by

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |F'(x_i)|.$$

For this value, three possible states occur: greater than zero, equal to zero or less than zero. By utilizing each of these states, the chaotic behavior of the system can be analyzed. If the Lyapunov exponent is greater than zero, then the system is chaotic. If the Lyapunov exponent is less than zero, then the orbit is periodic. When this value is equal to zero, a bifurcation occurs [21]. The graph of this value for two systems is given in Figure 7. According to the results in Figure 7a,b, it can be seen for $\alpha = 0.7$ that some of the Lyapunov exponents of the 4D fractional chaotic system are negative. After using the proposed hybrid chaotic system on the 4D fractional chaotic system with $\alpha = 0.7$ in Figure 7b, it can be seen that all the Lyapunov exponent values are positive. Therefore, the hybrid chaotic system improves the chaotic region of the 4D fractional chaotic system. Also, according to the results, it can be seen that the value of Lyapunov exponent for hybrid system is greater than this value for the 4D fractional chaotic system.

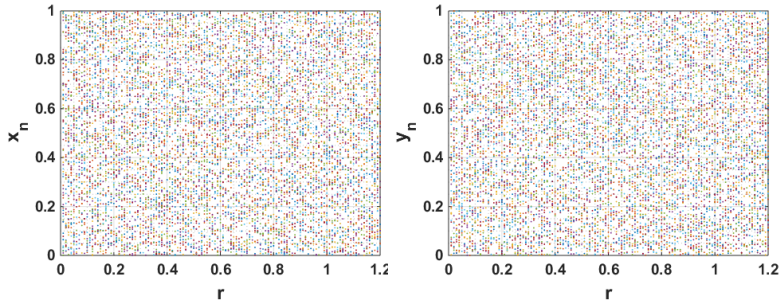
Another test that is used to study the behavior of the proposed systems is to draw the Bifurcation diagram. To draw this diagram, the first step is to divide the domain of r into several points. Then, for each of these numbers, the sequence produced by the chaotic system after a great repetition is plotted along the vertical line. In the event of convergence, the output sequence will approach a fixed point for specific points. Conversely, in the case of divergence, the output sequence will spread out. Bifurcation diagrams for the hybrid chaotic system with $\alpha = 0.3$ are shown in Figure 8. The results of this diagram show the divergence of the sequences produced by the chaotic system. Another diagram used to investigate the chaotic behavior of a system is the cobweb plot. The output results of this

plot for the proposed system are given in Figure 9. This figure shows that the outputs of the hybrid chaotic system do not approach a fixed point and represent chaotic orbits for the proposed systems.



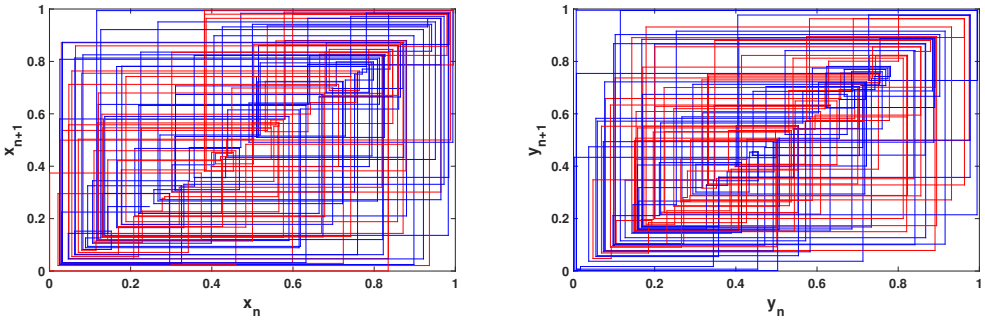
(**a**) $\alpha = 0.7$ (**b**) $\alpha = 0.7$

Figure 7. Lyapunov exponent results with $\alpha = 0.7$ for (**a**) 4D fractional chaotic system and (**b**) hybrid chaotic system.



(**a**) (**b**)

Figure 8. Bifurcation diagram with $\alpha = 0.3$ for (**a**,**b**) hybrid chaotic system.



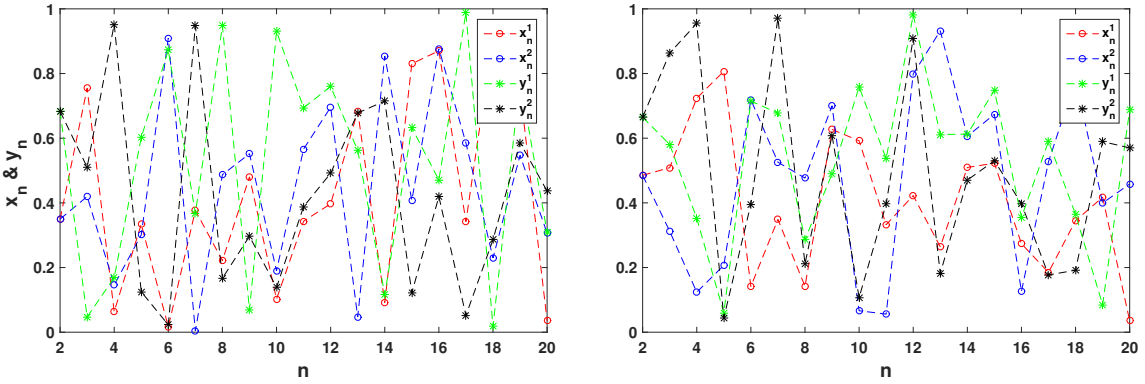
(**a**) (**b**)

Figure 9. Cobweb plots with $\alpha = 0.3$, $r = 0.2$ (red) and $r = 0.8$ (blue) for (**a**,**b**) hybrid chaotic system.

A paramount feature that a chaotic system should have is the sensitivity to the initial point. To study this feature, the outputs of the proposed systems with $\alpha = 0.3$ and $\alpha = 0.7$, $r = 0.7$ and different values of the initial point with a small difference, i.e., $X_0^1 = (0.1, 0.4)$ and $X_0^2 = (0.1, 0.4 + 10^{-15})$ are drawn in Figure 10. According to the results, it can be seen that although the change in the initial point is small, the outputs of the system are different. Consequently, minor changes in the input would result in modifications to the production sequence.

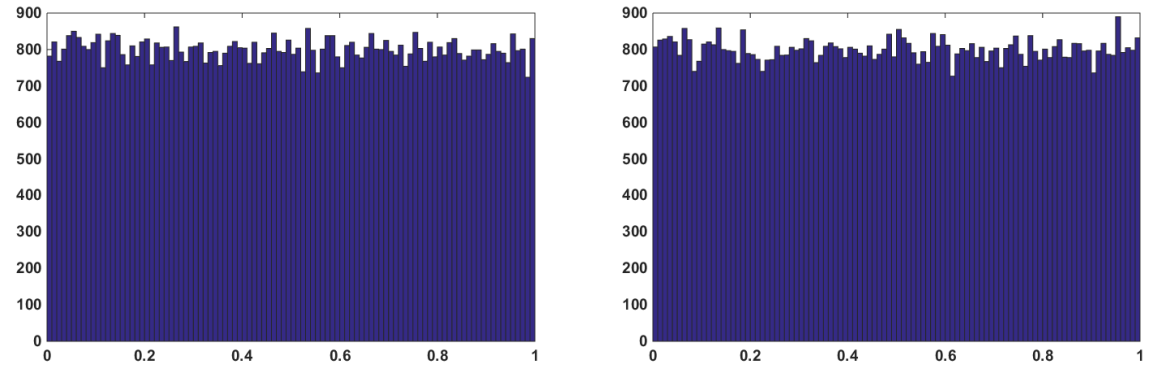
The nonuniform distribution can be considered as one of the reasons for the weakness of the encryption algorithm in dealing with statistical attacks. Since the distribution of the

chaotic system affects the distribution of the encrypted image, the uniform distribution of the chaotic system used in encryption can be effective in enhancing the security of the encryption algorithm. The histogram of hybrid chaotic system is shown in Figure 11. By using this figure, we can see that the histogram plot of output values is flat.



(**a**) (**b**)

Figure 10. Two outputs of the proposed hybrid chaotic system with (**a**) $\alpha = 0.3, r = 0.7$, (**b**) $\alpha = 0.7$, $r = 0.7$, and $X_0^1 = (0.1, 0.4)$ (red, green), $X_0^2 = (0.1, 0.4 + 10^{-15})$ (blue, black).



(**a**) (**b**)

Figure 11. Histogram of hybrid chaotic system with $\alpha = 0.7$ and $r = 0.5$: (**a**) x_n, (**b**) y_n.

The last test that is studied to check the behavior of the proposed systems is the $0-1$ test. The reader can find more information about this test in reference [22,23]. In this test for the time series as T_n, the following systems is obtained

$$p_{n+1} = p_n + T_{n+1}\cos(cn),$$
$$q_{n+1} = q_n + T_{n+1}\sin(cn),$$

where $c \in (0, 2\pi)$ and considered as fixed number. Then, the following growth rate is defined

$$K = \lim_{n\to\infty} \frac{\log M_n}{\log n},$$

where $M_n = \lim_{N\to\infty} \sum_{j=1}^{N} (p_{j+n} - p_j)^2 + (q_{j+n} - q_j)^2$. Expected ideal values of K for a chaotic system are one and zero for a non-chaotic system. The plot of c versus K is drawing in Figure 12. The results of this output show that the calculated values of K are close to ideal value one. Also, in this test, the plot of p versus q is drawn in Figure 13. According to these results, irregularity and diffusion can be observed in the output sequences.

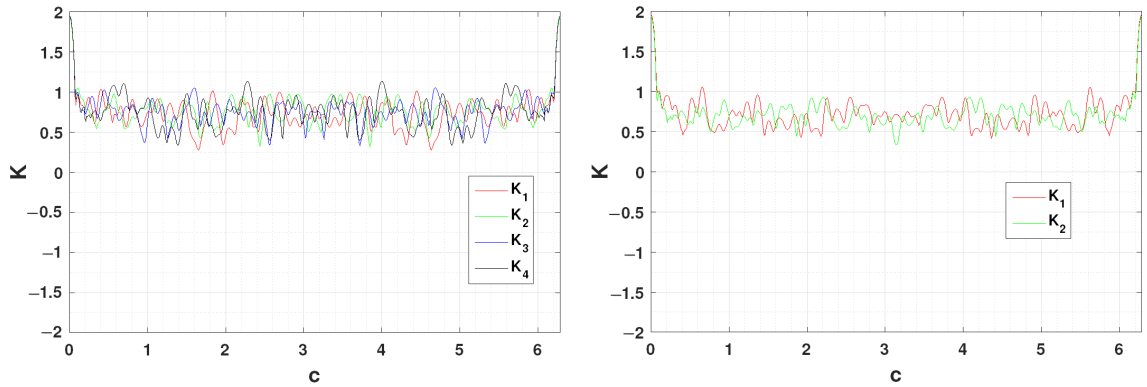


Figure 12. Plot of c versus K with $\alpha = 0.3$ and $r = 0.75$ for (**a**) 4D fractional chaotic system and (**b**) hybrid chaotic system.

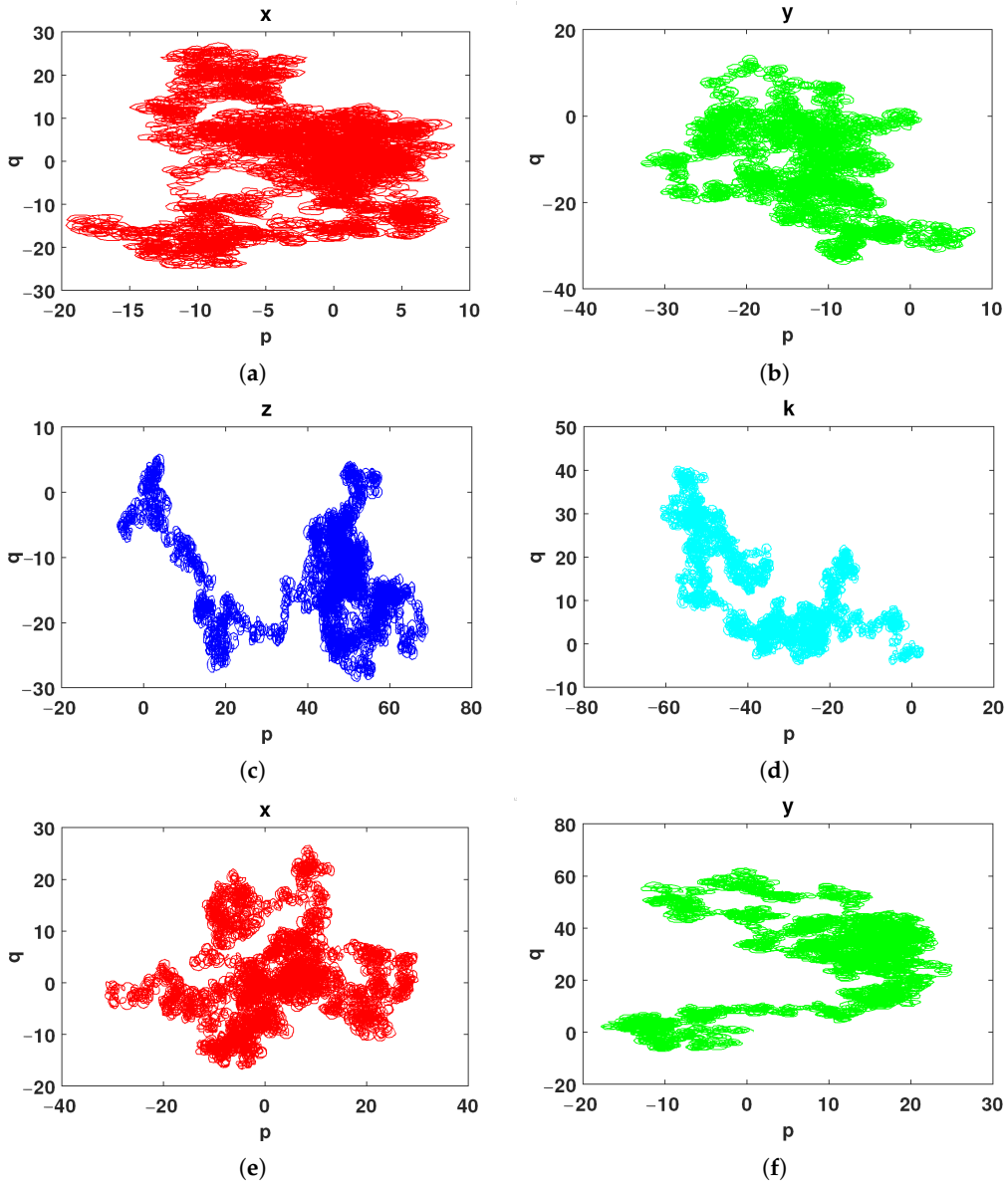


Figure 13. Plot of p versus q with $\alpha = 0.3$ and $r = 0.75$ for (**a**–**d**) 4D fractional chaotic system and (**e**,**f**) hybrid chaotic system.

4.3. Key Space Analysis

In this subsection, the analysis of the key generation algorithm is discussed. The key space consists of two components: a set of fixed keys and a set of keys dependent on input images. This dual nature of key selection arises from practical limitations in securely transferring keys. To overcome this challenge, the sender can initiate the process by transmitting a subset of fixed keys. Subsequently, the variable keys are sent in correspondence with a sequence of different images. As an initial study, the sensitivity of the variable key space to minor changes in the input data is studied. For this purpose, the fixed key space equal to $\{\alpha, r, h\} = \{0.2, 1.1, 0.001\}$ is considered. Also, four sequences of images are selected as

- $\mathrm{I}^1 = \{\mathrm{I}_1^1 = \texttt{Aerial 5.1.10}(256 \times 256),\ \mathrm{I}_2^1 = \texttt{Airplane 5.1.11}(256 \times 256),\ \mathrm{I}_3^1 = \texttt{Chemical plant 5.1.14}(256 \times 256),\ \mathrm{I}_4^1 = \texttt{Clock 5.1.12}(256 \times 256)\}$.
- $\mathrm{I}^2 = \{\mathrm{I}_1^2 = \texttt{Female 4.1.02}(256 \times 256 \times 3),\ \mathrm{I}_2^2 = \texttt{Female 4.1.03}(256 \times 256 \times 3),\ \mathrm{I}_3^2 = \texttt{Jelly beans 4.1.08}(256 \times 256 \times 3),\ \mathrm{I}_4^2 = \texttt{House 4.1.05}(256 \times 256 \times 3)\}$.
- $\mathrm{I}^3 = \{\mathrm{I}_1^3 = \texttt{Airplane 5.1.11}(256 \times 256),\ \mathrm{I}_2^3 = \texttt{Clock 5.1.12}(256 \times 256),\ \mathrm{I}_3^3 = \texttt{Jelly Jellybeans 4.1.08}(256 \times 256 \times 3),\ \mathrm{I}_4^3 = \texttt{House 4.1.05}(256 \times 256 \times 3)\}$.
- $\mathrm{I}^4 = \{\mathrm{I}_1^4 = \texttt{Watch}(512 \times 512),\ \mathrm{I}_2^4 = \texttt{Lena}(512 \times 512),\ \mathrm{I}_3^4 = \texttt{Peppers}(512 \times 512),\ \mathrm{I}_4^4 = \texttt{Fishing Boat}(512 \times 512)\}$.

Also, the Engine image is chosen as a 3D image. The applied changes are as follows:

Case i. Without changing the input data.

Case ii. The following changes have been applied to images $\mathrm{I}^1, \mathrm{I}^2, \mathrm{I}^3$ and I^4: $\mathrm{I}_1^1(20,20) = \mathrm{I}_1^1(20,20) + 1, \mathrm{I}_1^2(20,20,1) = \mathrm{I}_1^2(20,20,1) + 1, \mathrm{I}_1^3(20,20) = \mathrm{I}_1^3(20,20) + 1$ and $\mathrm{I}_1^4(20,20) = \mathrm{I}_1^4(20,20) + 1$.

Case iii. α has been changed to $0.2 + 10^{-15}$.

Case iv. The value of r has been updated to $1.1 + 10^{-15}$.

Case v. The new value assigned to h is $0.001 + 10^{-15}$.

The results related to the input changes of the key generation algorithm mentioned above are given in Table 1. Based on this table, it is evident that the key generation algorithm is sensitive to very small changes, and any change in the input of the algorithm creates a new key space.

The next aspect to analyze is the size of the key space. The size of the keys used should not be too small to be easily detected by a key recovery attack, such as the brute force attack, nor should it be too large that it cannot be transferred in a secure environment. In relation to the subjects discussed in source [24], it is noted that the algorithm can withstand the brute force attack if the key size is a minimum of 2^{100}. Assuming that the calculation accuracy in the program execution environment has an accuracy of 10^{-15}, the size of the key space will be equal to $10^{105} \approx 2^{348}$. Therefore, the size of the key space is large enough to resist this attack. In Table 2, the size of the key space of the proposed algorithm is compared with the size of the key space of other papers. It should be noted that this table is given only for comparison, and the magnitude of the key size does not necessarily indicate the superiority of the algorithm. As mentioned earlier, a key value greater than 2^{100} enhances the algorithm's resistance against attacks. However, if the secure transmission path is constrained, the size of the key space may become problematic.

As the last study in this subsection, the effect of a small change in the key space in the encryption and decryption algorithms is investigated. For this purpose, the sequence of images I^1 and the following keys are considered.

$$K_1 := \{\alpha, r, h, x_1, x_2, y_1, y_2\} = \{0.2, 1.1, 0.001, 0.3699, 0.3310, 0.3302, 0.3041\},$$
$$K_2 := \{\alpha, r, h, x_1, x_2, y_1, y_2\} = \{0.2, 1.1, 0.001, 0.3699 + 10^{-15}, 0.3310, 0.3302, 0.3041\}.$$

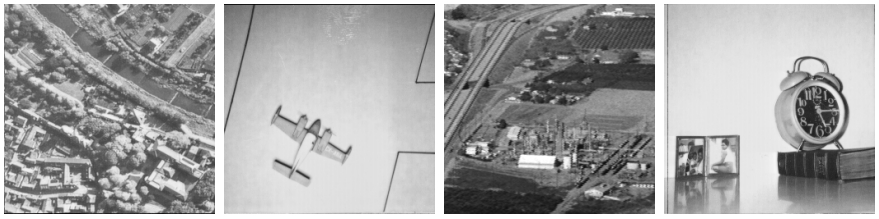
It can be seen that these two key spaces exhibit very little difference. In the first case, the sequence of images I^1 is encrypted by the key K_1, and then the sequence of decryption images is checked by the key spaces K_1 and K_2. The results of this process are illustrated in Figure 14. It can be seen that the original images cannot be recovered using a key space with only minor differences from the original key. The sequence of images encrypted by these two keys creates different encrypted images even though they are slightly different. In order to show the difference between these two sequences, pixel-by-pixel subtraction and its histogram are shown in Figure 15, and these results show their difference. Consequently, without comprehensive knowledge of the key space, image recovery is not feasible.

Table 1. The output of the key generation algorithm for different inputs.

Images	Key	Case i	Case ii	Case iii	Case iv	Case v
I^1	x_1	0.3699	0.3607	0.0053	0.2009	0.1936
	y_1	0.3302	0.3437	0.2468	0.2260	0.3965
	x_2	0.3310	0.1241	0.5127	0.3521	0.2145
	y_2	0.3041	0.3462	0.4062	0.1765	0.1814
I^2	x_1	0.4680	0.1866	0.1718	0.1439	0.1860
	y_1	0.3314	0.3966	0.1107	0.3376	0.3431
	x_2	0.3057	0.4664	0.2168	0.1822	0.3739
	y_2	0.2933	0.1508	0.4643	0.1746	0.3616
I^3	x_1	0.2307	0.1978	0.4691	0.4069	0.2901
	y_1	0.1547	0.1771	0.1116	0.3248	0.3889
	x_2	0.4242	0.2972	0.3734	0.3846	0.2588
	y_2	0.4377	0.5214	0.2964	0.3770	0.3176
I^4	x_1	0.2633	0.2028	0.2784	0.2385	0.3297
	y_1	0.0873	0.3831	0.3515	0.1748	0.0777
	x_2	0.1848	0.3284	0.5258	0.2342	0.2752
	y_2	0.2583	0.3127	0.4522	0.2607	0.2202
Engine	x_1	0.5381	0.3147	0.3814	0.2705	0.0153
	y_1	0.2818	0.5031	0.3812	0.2064	0.1799
	x_2	0.2737	0.1674	0.2143	0.0793	0.2559
	y_2	0.1766	0.4007	0.0623	0.0133	0.5334

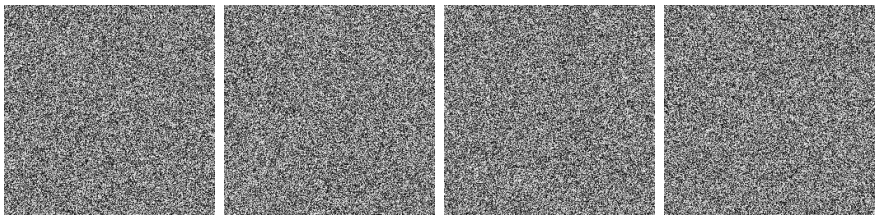
Table 2. Key space size for different algorithms.

Algorithm	Proposed	Ref. [25]	Ref. [26]	Ref. [27]	Ref. [28]
Key size	$10^{105} \approx 2^{348}$	1.936×10^{59}	10^{60}	4.34×10^{96}	10^{75}
Algorithm	Ref. [29]	Ref. [30]	Ref. [31]	Ref. [32]	Ref. [33]
Key size	10^{56}	5.841×10^{135}	1.2446×2^{327}	1.157×10^{203}	10^{56}



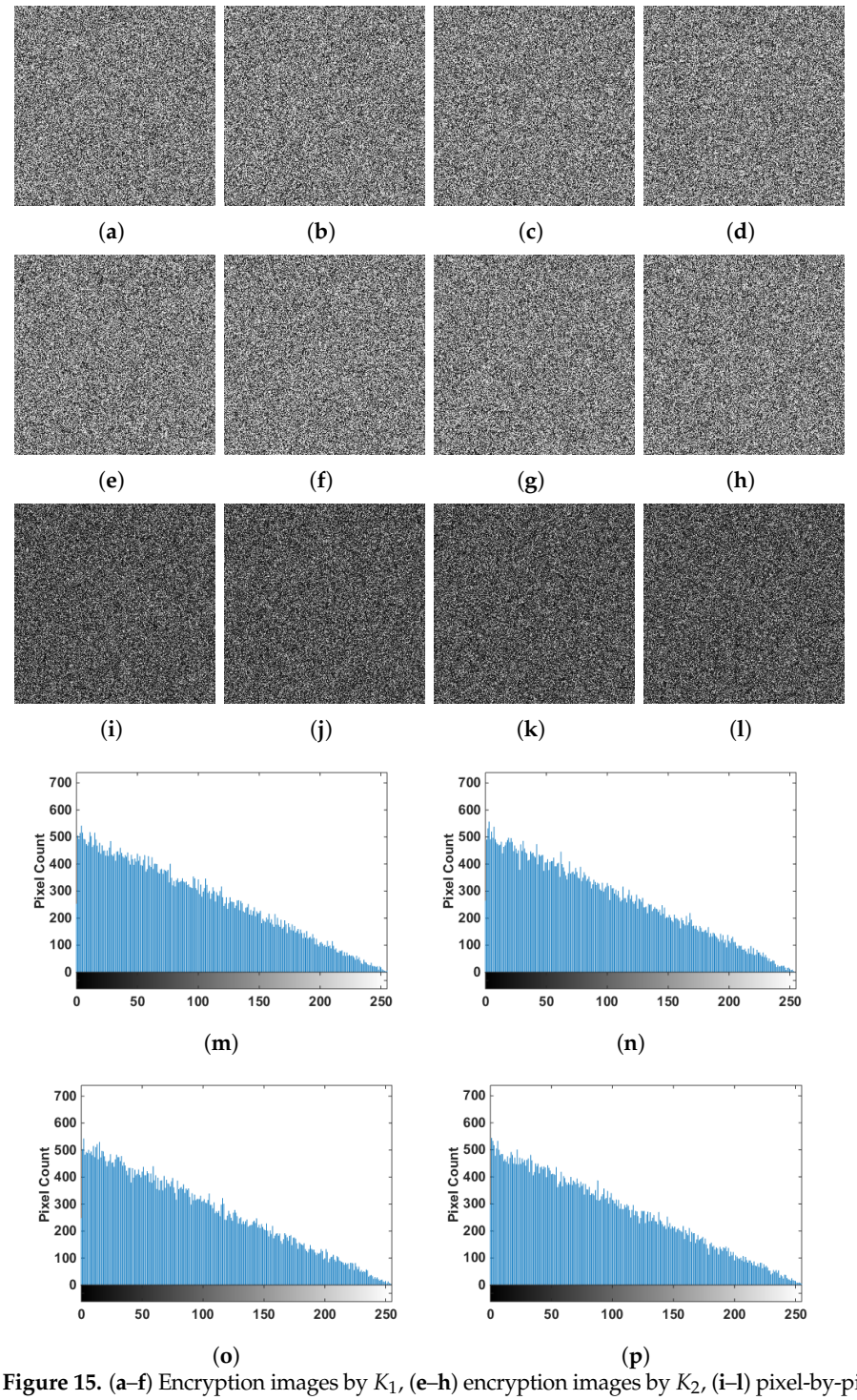
(**a**) (**b**) (**c**) (**d**)

Figure 14. *Cont.*



(**e**) (**f**) (**g**) (**h**)

Figure 14. (**a**–**d**) Decrypted images by K_1 and (**e**–**h**) decrypted images by K_2.



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

(**e**) (**f**) (**g**) (**h**)

(**i**) (**j**) (**k**) (**l**)

(**m**) (**n**)

(**o**) (**p**)

Figure 15. (**a**–**f**) Encryption images by K_1, (**e**–**h**) encryption images by K_2, (**i**–**l**) pixel-by-pixel subtraction of the encryption images and (**m**–**p**) histogram of the pixel-by-pixel subtraction of the encryption images.

4.4. Analysis of Program Execution Time

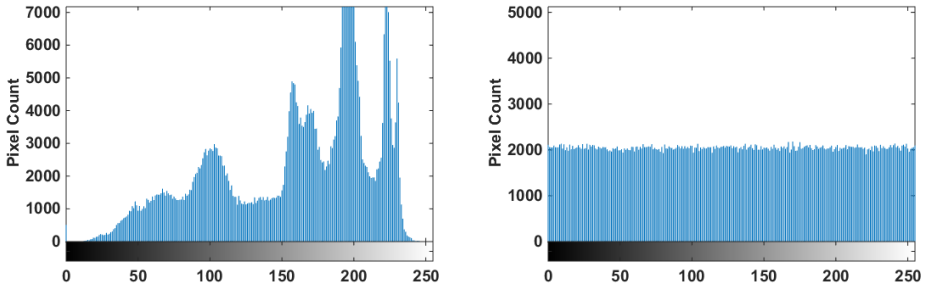
In addition to ensuring secure data transmission, an encryption algorithm must also be efficient in terms of its execution time to be suitable for simulation and widespread use. To evaluate the execution time of the proposed algorithm, the execution times for some of the image sequences are listed in Table 3. According to the reported times, it can be concluded that the calculation time for the proposed algorithm is efficient.

Table 3. Program execution time (in seconds).

Images	Key Generation	Encryption	Decryption
I^1	0.2084	3.7334	3.8196
I^2	0.2677	6.6504	6.4169
I^3	0.2648	5.1215	5.0604
I^4	0.3203	14.1428	14.1293

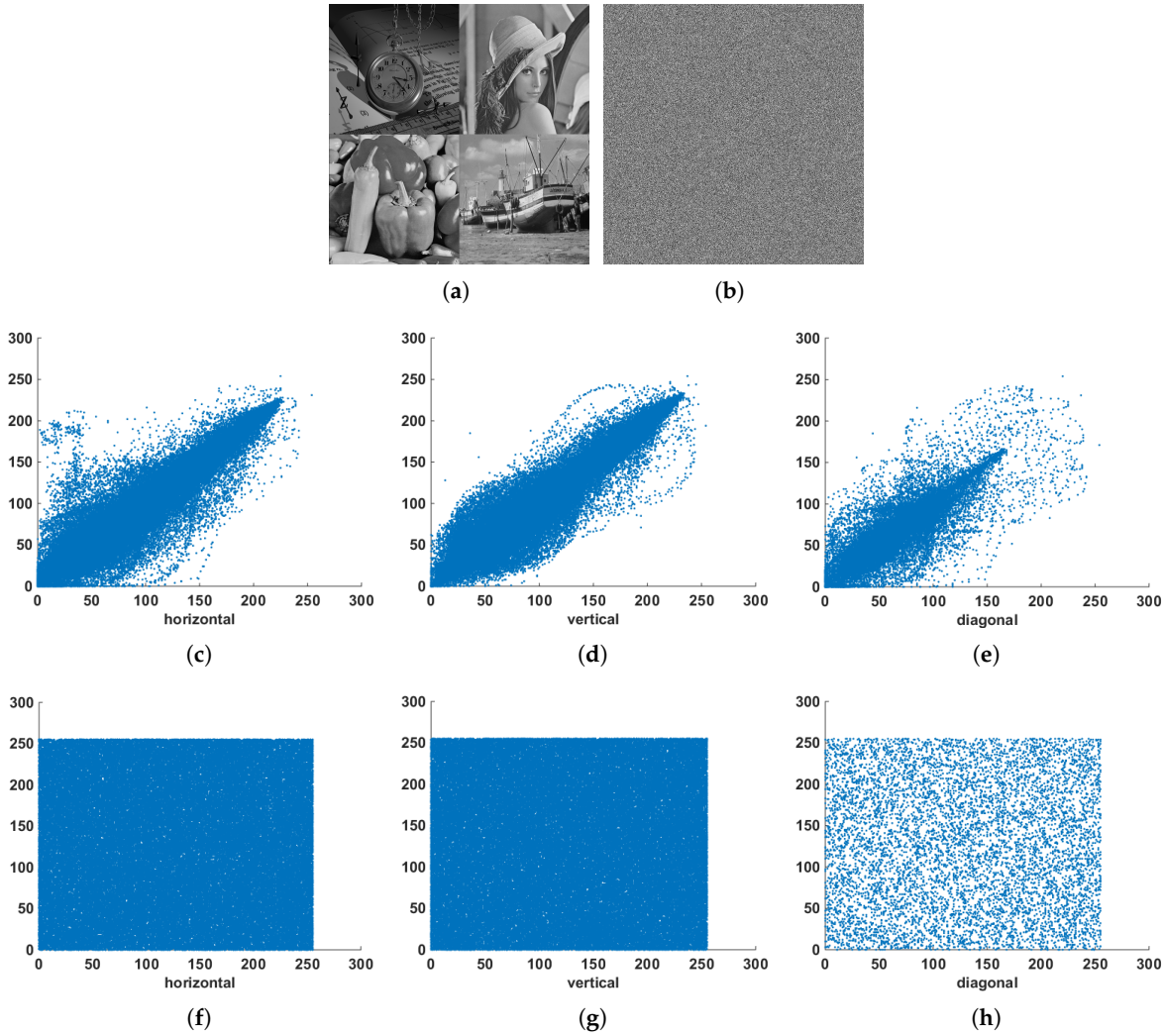
4.5. Histogram and Correlation Coefficients Analysis

In unencrypted images, in most cases, the numerical values of the pixels are not uniformly distributed, and the number of the values of pixels varies according to the type of image. These differences contribute to the image's meaningfulness. For example, in an image taken from the night sky, black color will be more prevalent than other colors. Additionally, in a meaningful, unencrypted image, the relationship between adjacent pixels is often significant. The relationship between pixels is analyzed using the statistical concept of correlation coefficients. Therefore, if an image is not well encrypted, the attacker may continue to decrypt it based on the numerical values of some pixels or the relationship between pixels. In practice, some attacks are designed to exploit this weak point, such as statistical attacks. In this subsection, the histogram and correlation coefficient are studied. The histogram plots for all images in sequences I^2 and the sequence of encrypted images of these sequences are drawn in Figure 16. The results of this figure show that the encrypted image, unlike the original image, has a uniform distribution. In the calculations, the Pearson correlation coefficient is used to measure correlation. The value of this test for an encrypted image should be close to zero. A value near zero indicates that the pixels are uncorrelated, which is desirable for encryption. Conversely, a value close to one suggests a strong correlation between pixels, implying insufficient encryption To check the correlation coefficients, the sequence of images I^4 is selected. Also, to avoid the need for calculating the correlation coefficients for each image, the images are placed side by side, and this value is computed. In Figure 17, the correlation distributions for different directions are illustrated. Additionally, the correlation coefficients are obtained and compared with other algorithms in Table 4. For another example, the sequence of images $I^5 = \{\texttt{Barbara}(512 \times 512), \texttt{Airplane}(512 \times 512), \texttt{Fishing boat}(512 \times 512), \texttt{Watch}(512 \times 512)\}$ is considered, and then, using the fixed key $\{\alpha = 0.2, r = 1.1, h = 0.001\}$, the sequence of encrypted images is obtained. In Table 5, after obtaining the sequence of encrypted images, the correlation coefficients for each image are calculated and compared by their average value. From these results, it can be observed that the values are close to zero, indicating no discernible relationship between the pixel intensities of adjacent pixels in various directions.



(**a**) (**b**)

Figure 16. Histogram plot of the sequence of the images: (**a**) I^2 and (**b**) encrypted I^2.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**) (**g**) (**h**)

Figure 17. (**a**) Input images, (**b**) encrypted images and correlation of neighborhood pixels at different directions (**c**–**e**) before encryption and (**f**–**h**) after encryption.

Table 4. Comparison of correlation coefficients for I^4.

	Input images	Proposed	Ref. [34]
Horizontal	0.9684	0.0001	−0.0003
Vertical	0.9783	0.0010	0.0011
Diagonal	0.9558	0.0003	0.0013
	Ref. [35]	Ref. [36]	Ref. [37]
Horizontal	−0.0036	0.0538	0.0034
Vertical	0.0016	0.0389	0.0015
Diagonal	0.0058	0.0307	0.0008

Table 5. Comparison of correlation coefficients for I^5.

	Input images	Proposed	Ref. [38]
Horizontal	0.9272	−0.0001	0.0058
Vertical	0.9599	0.0014	0.0048
Diagonal	0.9070	0.0001	−0.0006
	Ref. [39]	Ref. [40]	Ref. [41]
Horizontal	0.0012	0.0256	−0.0010
Vertical	0.0018	0.0116	−0.0009
Diagonal	0.0014	0.0180	0.0006

4.6. NPCR, UACI and Information Entropy Analysis

NPCR (Number of Pixel Change Rate), UACI (Unified Average Changing Intensity) and information entropy are commonly used metrics for performance evaluation and analysis of image encryption algorithms. The tests NPCR and UACI, as their names suggest, calculate the amount of pixel changes. Since a suitable encryption algorithm should be able to detect most of the changes for small changes in the input images, these tests are used to calculate the sensitivity of the changes in the algorithm. For this purpose, the following formulas are used. Suppose the image is encrypted by the algorithm for the first time and denoted as d_1, then a small change is made to the image, which is subsequently encrypted by the algorithm and denoted as d_2. NPCR and UACI are defined as

$$\text{NPCR} = \frac{\sum_{i,j} \delta_{i,j}}{n \times m} \times \%100, \qquad \text{UACI} = \frac{\sum_{i,j} |d_1 - d_2|}{255 \times n \times m} \times \%100,$$

where n, m are the size of input image, and $\delta_{i,j}$ is defined as

$$\delta_{i,j} := \begin{cases} 1, & \text{if } d_1(i,j) \neq d_2(i,j), \\ 0, & \text{if } d_1(i,j) = d_2(i,j). \end{cases}$$

The values 100% and $33.\overline{33}$ are considered as ideal values for NPCR and UACI, respectively. Considering that in practice it is not possible for an algorithm to achieve the ideal value, a critical range for these values has been introduced in [42]. This range is checked in Table 6 for various sequences of images. Also, the results of the proposed algorithm are compared with those of another algorithm in Table 7. As the last result for this test, the average value of NPCR and UACI values for the sequence images I^5 have been compared with other encryption algorithms in Table 8. By the results of these tests and comparing it with the ideal values, it can be seen that the reported values are close to the ideal values.

Table 6. Numerical results of UACI and NPCR for various sequences of images.

		UACI Critical Values [42]				NPCR Critical Values [42]		
Image	**UACI**	$\mathbf{u}^{*-}_{0.05}$ = 33.2824 $\mathbf{u}^{*+}_{0.05}$ = 33.6447	$\mathbf{u}^{*-}_{0.01}$ = 33.2255 $\mathbf{u}^{*+}_{0.01}$ = 33.7016	$\mathbf{u}^{*-}_{0.001}$ = 33.1594 $\mathbf{u}^{*+}_{0.001}$ = 33.7677	**NPCR**	$\mathbf{N}^{*}_{0.05}$ = 99.5693	$\mathbf{N}^{*}_{0.01}$ = 99.5527	$\mathbf{N}^{*+}_{0.001}$ = 99.5341
I^1	33.3277	Pass	Pass	Pass	99.6063	Pass	Pass	Pass
I^2	33.4920	Pass	Pass	Pass	99.6190	Pass	Pass	Pass
I^3	33.4685	Pass	Pass	Pass	99.6002	Pass	Pass	Pass
Engine	33.4090	Pass	Pass	Pass	99.6107	Pass	Pass	Pass

Table 7. Comparison of NPCR, UACI and information entropy values.

	Proposed	**Ref. [34]**	**Ref. [37]**	**Ref. [43]**
NPCR	99.6041	99.6060	99.1841	99.6534
UACI	33.4587	33.5126	33.5284	33.6772
Information entropy	7.9998	7.9998	7.9996	7.9994

Table 8. Comparison of NPCR and UACI values for I^5.

	Proposed	**Ref. [38]**	**Ref. [39]**	**Ref. [40]**
NPCR	99.6162	99.6075	99.6100	99.6125
UACI	33.4485	33.4100	33.5450	33.4150

A measure of the randomness or unpredictability of pixel values is studied by the information entropy. If l and p represent the gray level and the probability, and I_i are the possible pixel values, this value is calculated by the formula:

$$H = -\sum_{i=0}^{l-1} p(I_i) log_2\big(p(I_i)\big).$$

The ideal value of information entropy is 8. The numerical results for this test are presented in Tables 7 and 9. Also, the numerical results for the average value of information entropy of the encrypted sequence images have been compared with the results of other papers in Table 10 for I^5. These results show that the value obtained from the proposed method is close to the ideal value.

Table 9. Numerical results of information entropy.

	$\mathbf{I^1}$	$\mathbf{I^2}$	$\mathbf{I^3}$	**Engine**
Information entropy	7.9992	7.9997	7.9996	7.9959

Table 10. Comparison of information entropy for I^5.

	Input Images	Proposed	Ref. [38]
Information entropy	7.1252	7.9992	7.9983
	Ref. [39]	Ref. [40]	Ref. [41]
Information entropy	7.9974	7.9420	7.9984

4.7. Data Loss and Noise Attacks

Because the route of encrypted data transmission lacks high security, if an attacker cannot decode the encrypted images, they may attempt to disrupt the receiver from obtaining information by introducing noise or destroying parts of the image. A suitable encryption algorithm should be able to prevent the loss of all the sent information after

encryption, despite the loss of part of the information. The results of this test are shown in Figures 18–20. According to the results, it can be seen that although part of the data is lost after decoding the images, the reconstructed images remain recognizable and are not entirely lost.

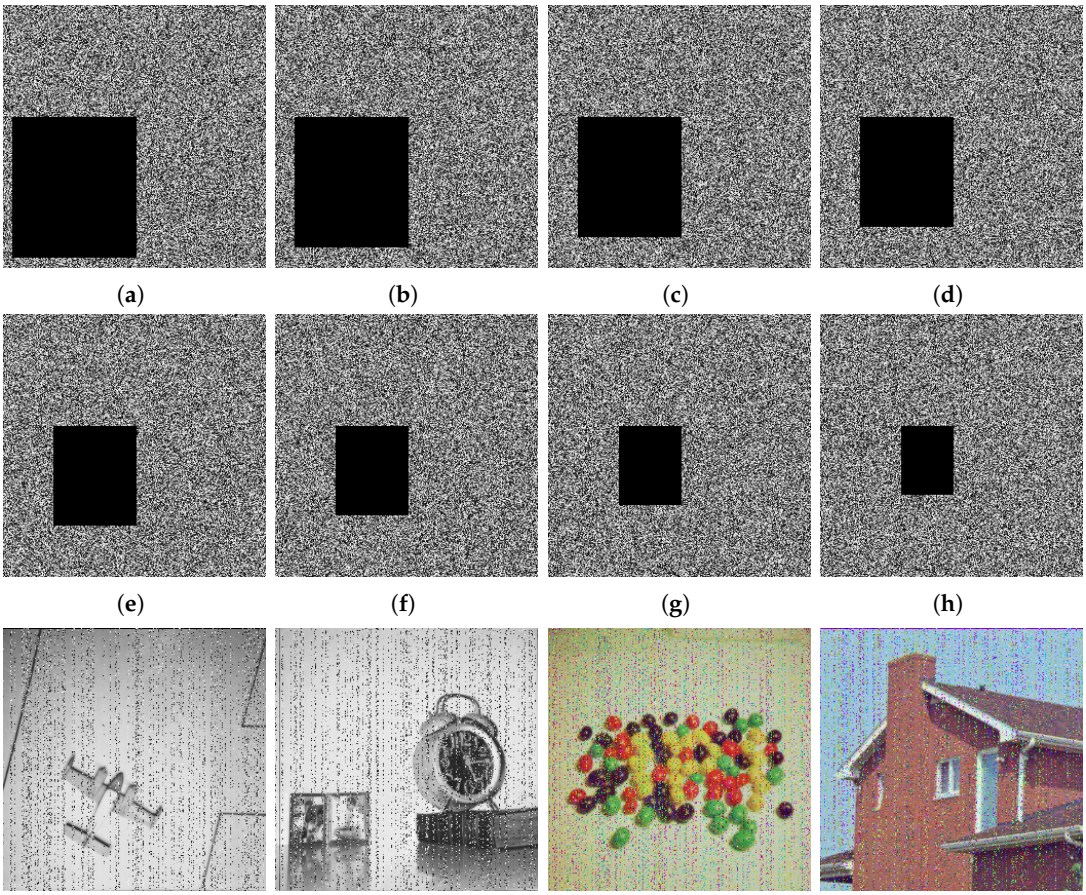


Figure 18. (**a**–**h**) Sequence of encrypted images destroyed by data loss attack and (**i**–**l**) decrypted images after data loss attack.

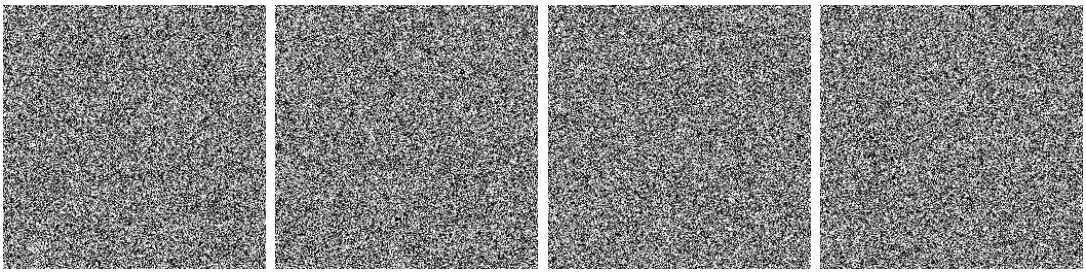
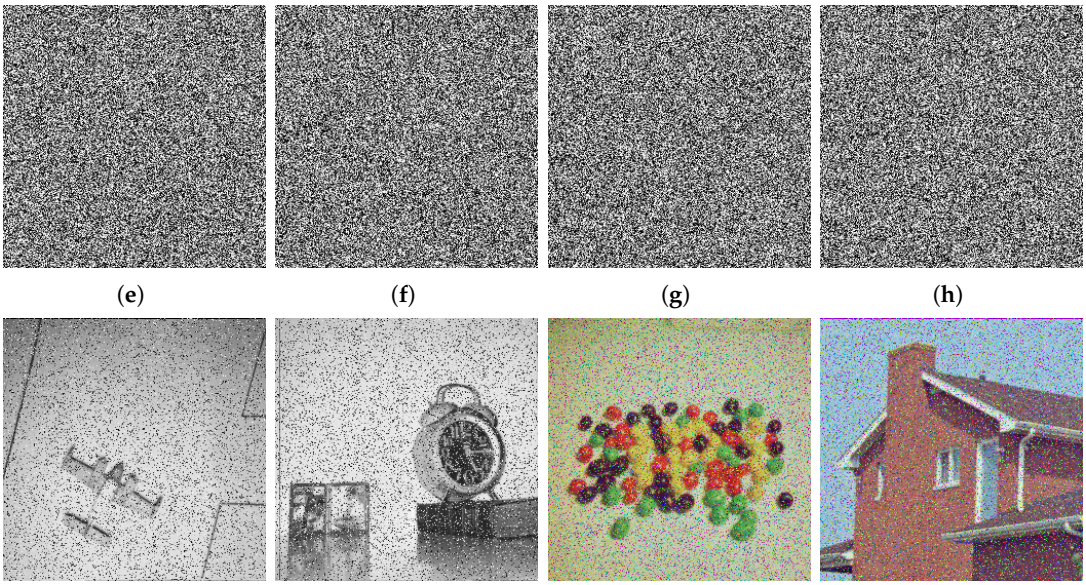


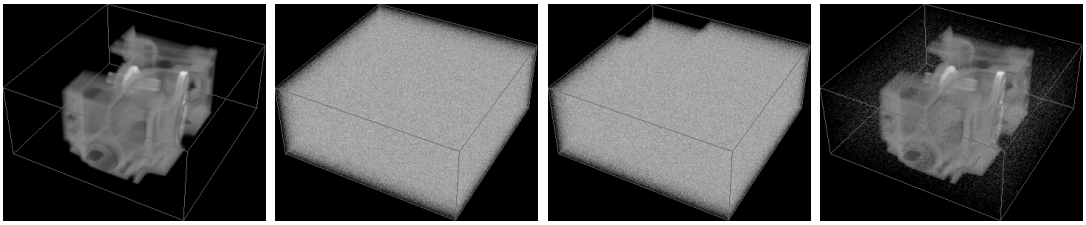
Figure 19. *Cont.*



(**e**) (**f**) (**g**) (**h**)

(**i**) (**j**) (**k**) (**l**)

Figure 19. (**a**–**h**) Sequence of encrypted images destroyed by salt and pepper noise attack with noise density = 0.2 and (**i**–**l**) decrypted images after noise attack.



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

Figure 20. (**a**) Original 3D image of engine, (**b**) encrypted image, (**c**) destroyed image by data loss attack and (**d**) decrypted images after data loss attack.

5. Conclusions

In this work, we address a critical issue in image transmission: ensuring image transmission security. In the first part, one of the most important tools used in the proposed algorithm, the chaotic system, has been introduced. For this purpose, first the 4D chaotic system is calculated by the Atangana–Baleanu fractional derivative, and then a hybrid chaotic system is introduced by combining this system with the Cat map. The chaotic behavior of these systems has been studied by various types of tests, and the results demonstrate the appropriate behavior of the proposed systems. In the second part, the encryption algorithm is designed for multiple images that can be used for 3D images. In the simulation section, the security of the algorithm is studied through different types of statistics tests, the results of which indicate the efficiency of the proposed algorithm.

Author Contributions: Methodology, G.G., R.P. and Y.K.Y.; Software, R.P. and Y.K.Y.; Validation, G.G., R.P. and Y.K.Y.; Writing—Original Draft, R.P.; Writing—Review and Editing, G.G., R.P. and Y.K.Y.; Supervision, G.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No associated data included in the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, X.; Liu, H. Cross-plane multi-image encryption using chaos and blurred pixels. *Chaos Solitons Fractals* **2022**, *164*, 112586. [CrossRef]
2. Sangavi, V.; Thangavel, P. An exquisite multiple image encryption harnessing multi-scroll Lu–Chen and Chua chaotic systems employing domino strategy. *J. Inf. Secur. Appl.* **2023**, *72*, 103408. [CrossRef]
3. Zhang, L.; Wang, Y.; Zhang, D. Research on multiple-image encryption mechanism based on Radon transform and ghost imaging. *Opt. Commun.* **2022**, *504*, 127494. [CrossRef]
4. Zhou, N.R.; Tong, L.J.; Zou, W.P. Multi-image encryption scheme with quaternion discrete fractional Tchebyshev moment transform and cross-coupling operation. *Signal Process.* **2023**, *211*, 109107. [CrossRef]
5. Pham, D.H.; Huynh, T.T.; Lin, C.M.; Giap, V.N.; Vu, V.P. Secure transmission of medical image using a wavelet interval type-2 TSK fuzzy brain-imitated neural network. *Soft Comput.* **2025**, *29*, 2311–2329. [CrossRef]
6. Chang, H.; Wang, E.; Liu, J. Research on image encryption based on fractional seed chaos generator and fractal theory. *Fractal Fract.* **2023**, *7*, 221. [CrossRef]
7. Setiadi, D.R.; Rijati, N. An image encryption scheme combining 2D cascaded logistic map and permutation-substitution operations. *Computation* **2023**, *11*, 178. [CrossRef]
8. Liu, X.; Tong, X.; Wang, Z.; Zhang, M. A new n-dimensional conservative chaos based on Generalized Hamiltonian System and its' applications in image encryption. *Chaos Solitons Fractals* **2022**, *154*, 111693. [CrossRef]
9. Liu, X.; Tong, X.; Zhang, M.; Wang, Z. A highly secure image encryption algorithm based on conservative hyperchaotic system and dynamic biogenetic gene algorithms. *Chaos Solitons Fractals* **2023**, *171*, 113450. [CrossRef]
10. Liu, X.; Tong, X.; Wang, Z.; Zhang, M. A novel hyperchaotic encryption algorithm for color image utilizing DNA dynamic encoding and self-adapting permutation. *Multimed. Tools Appl.* **2022**, *81*, 21779–21810. [CrossRef]
11. Dutta, H.; Akdemir, A.O.; Atangana, A. (Eds.) *Fractional Order Analysis: Theory, Methods and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2020.
12. Zarebnia, M.; Parvaz, R. Image encryption algorithm by fractional based chaotic system and framelet transform. *Chaos Solitons Fractals* **2021**, *152*, 111402. [CrossRef]
13. Gomez-Aguilar, J.F.; Atangana, A. (Eds.) *Applications of Fractional Calculus to Modeling in Dynamics AND Chaos*; CRC Press: Boca Raton, FL, USA, 2022.
14. Anastassiou, G.A. *Abstract Fractional Monotone Approximation, Theory and Applications*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
15. Gómez, J.F.; Torres, L.; Escobar, R.F. *Fractional Derivatives with Mittag-Leffler Kernel*; Springer International Publishing: Cham, Switzerland, 2019.
16. Parvaz, R.; Yengejeh, Y.K.; Behroo, Y. A new 4D chaos system with an encryption algorithm for color and grayscale images. *Int. J. Bifurc. Chaos* **2022**, *32*, 2250214. [CrossRef]
17. Agarwal, R.P.; El-Sayed, A.M.; Salman, S.M. Fractional-order Chua's system: Discretization, bifurcation and chaos. *Adv. Differ. Equations* **2013**, *2013*, 320. [CrossRef]
18. Falcioni, M.; Palatella, L.; Pigolotti, S.; Vulpiani, A. Properties making a chaotic system a good pseudo random number generator. *Phys. Rev. E—Stat. Nonlinear Soft Matter Phys.* **2005**, *72*, 016220. [CrossRef]
19. Wu, Y.; Hua, Z.; Zhou, Y. *n*-dimensional discrete cat map generation using Laplace Expansions. *IEEE Trans. Cybern.* **2015**, *46*, 2622–2633. [CrossRef]
20. Bao, J.; Yang, Q. Period of the discrete Arnold cat map and general cat map. *Nonlinear Dyn.* **2012**, *70*, 1365–1375. [CrossRef]
21. Van Opstall, M. Quantifying chaos in dynamical systems with Lyapunov exponents. *Furman Univ. Electron. J. Undergrad. Math.* **1998**, *4*, 1–8.
22. Rana, S.M. Bifurcation Analysis and 0–1 Chaos Test of a Discrete T System. *Chaos Theory Appl.* **2023**, *5*, 90–104. [CrossRef]
23. Bernardini, D.; Litak, G. An overview of 0–1 test for chaos. *J. Braz. Soc. Mech. Sci. Eng.* **2016**, *38*, 1433–1450. [CrossRef]
24. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
25. Shao, Z.; Liu, X.; Yao, Q.; Qi, N.; Shang, Y.; Zhang, J. Multiple-image encryption based on chaotic phase mask and equal modulus decomposition in quaternion gyrator domain. *Signal Process. Image Commun.* **2020**, *80*, 115662. [CrossRef]
26. Zhang, X.; Wang, X. Multiple-image encryption algorithm based on mixed image element and permutation. *Opt. Lasers Eng.* **2017**, *92*, 6–16. [CrossRef]
27. Yu, C.; Li, J.; Li, X.; Ren, X.; Gupta, B.B. Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram. *Multimed. Tools Appl.* **2018**, *77*, 4585–4608. [CrossRef]
28. Patel, A.; Parikh, M. Multiple image encryption using chaotic map and DNA computing. *Int. J. Sci. Res. Sci. Eng. Technol.* **2018**, *4*, 1395–1400.

29. Zhang, X.; Wang, X. Multiple-image encryption algorithm based on mixed image element and chaos. *Comput. Electr. Eng.* **2017**, *62*, 401–413. [CrossRef]
30. Tang, Z.; Song, J.; Zhang, X.; Sun, R. Multiple-image encryption with bit-plane decomposition and chaotic maps. *Opt. Lasers Eng.* **2016**, *80*, 1–11. [CrossRef]
31. Patro, K.A.; Soni, A.; Netam, P.K.; Acharya, B. Multiple grayscale image encryption using cross-coupled chaotic maps. *J. Inf. Secur. Appl.* **2020**, *52*, 102470. [CrossRef]
32. Hanif, M.; Naqvi, R.A.; Abbas, S.; Khan, M.A.; Iqbal, N. A novel and efficient 3D multiple images encryption scheme based on chaotic systems and swapping operations. *IEEE Access* **2020**, *8*, 123536–123555. [CrossRef]
33. Zhang, X.; Wang, X. Multiple-image encryption algorithm based on DNA encoding and chaotic system. *Multimed. Tools Appl.* **2019**, *78*, 7841–7869. [CrossRef]
34. Zhang, X.; Hu, Y. Multiple-image encryption algorithm based on the 3D scrambling model and dynamic DNA coding. *Opt. Laser Technol.* **2021**, *141*, 107073. [CrossRef]
35. Sahasrabuddhe, A.; Laiphrakpam, D.S. Multiple images encryption based on 3D scrambling and hyper-chaotic system. *Inf. Sci.* **2021**, *550*, 252–267. [CrossRef]
36. Hasheminejad, A.; Rostami, M.J. A novel bit level multiphase algorithm for image encryption based on PWLCM chaotic map. *Optik* **2019**, *184*, 205–213. [CrossRef]
37. Ye, H.S.; Zhou, N.R.; Gong, L.H. Multi-image compression-encryption scheme based on quaternion discrete fractional Hartley transform and improved pixel adaptive diffusion. *Signal Process.* **2020**, *175*, 107652. [CrossRef]
38. Zhang, X.; Liao, J. Multiple-image encryption algorithm based on 3D-LWT and dynamic stereo S-box. *Multimed. Tools Appl.* **2024**, *83*, 16337–16362. [CrossRef]
39. Tian, Y.; Lu, Z. Novel permutation-diffusion image encryption algorithm with chaotic dynamic S-box and DNA sequence operation. *AIP Adv.* **2017**, *7*, 085008. [CrossRef]
40. Zhang, X.; Liao, J. Multiple-image encryption algorithm based on the dice-rotation model and improved chaos. *J. Electron. Imaging* **2022**, *31*, 043047. [CrossRef]
41. Zhu, H.; Dai, L.; Liu, Y.; Wu, L. A three-dimensional bit-level image encryption algorithm with Rubik's cube method. *Math. Comput. Simul.* **2021**, *185*, 754–770. [CrossRef]
42. Wu, Y.; Noonan, J.P.; Agaian, S. NPCR and UACI randomness tests for image encryption. *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT)* **2011**, *1*, 31–38.
43. Naseer, Y.; Shah, T.; Javeed, A. Advance image encryption technique utilizing compression, dynamical system and S-boxes. *Math. Comput. Simul.* **2020**, *178*, 207–217. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.