[1]Bethala Shirisha*

[2]Dr. V. Kamakshi Prasad

Reversible Data Hiding in Encrypted Images Based on Hybrid Chaotic Logistic Map and Tent Map to Protect Privacy of User in Cloud

Abstract: **-** Reversible data hiding in encrypted images (RDHEI) serves a dual purpose in today's environment, where cloud storage and privacy protection are becoming increasingly crucial by acting as both a privacy safeguard and a means of transferring private information. This paper provides a Hamming Code according to our proposed prediction and detection methods. The image is encrypted first by the content owner, who employs a hybrid chaotic logistic and tent map function. A robust cryptosystem based on a 3D chaotic map for image encryption in secure cloud services. The proposed encryption method depends on a mix of 3D Tent and 3D Logistic chaotic maps. With this map, a nonlinear ciphering process is implemented for pixel value diffusion and position permutation. A chaotic sequence is used to generate the user-defined key. The data hider will then include the data through a technique known as histogram modification. We use histogram shifting. Based on the code, the user could either obtain the hidden data by image encrypting that contains protected private data or decode and recreate the secured personal mage. We get an average entropy of encrypted image 7.99, NPCR of 99.6% and UACI of 33.5%. Compared with previous studies, a novel method we created simplifies the scheme significantly while achieving better data processing performance loss and hiding capacity.

Keywords: RDHEI; Permutation and diffusion, Chaos; Logistic map, tent map

1. Introduction

Cloud computing is a rapidly evolving technology that offers users on-demand access to shared, dynamically modifiable resources via a computer network. [1]. Cloud computing services are growing in popularity among businesses because they are versatile, efficient, scalable, and often the most cost-effective option. [2]. One example of a cloud computing service that is presently available is Amazon Elastic Compute Cloud (Amazon EC2). This service promotes virtual information technology (Virtual IT) by allowing customers to rent virtual computers to execute their software applications. [3]. Amazon Elastic Compute Cloud (Amazon EC2) is a cloud computing service with scalability. [4]. Google App Engine is another example of a cloud computing service that facilitates application hosting. As cloud storage and computing technology advanced, local servers began to transfer data storage, image processing, and other tasks to cloud servers. A third-party-provided data source transfers a sizable amount of data to the cloud for simple sharing, quick processing, centralization, or verification. This can encrypt data before uploading it to the cloud service to safeguard the privacy of the user's information and the data stored there. For instance, to practice telehealth and protect patient privacy, a local hospital transmitting a patient's medical image to a different hospital via services must encrypt the image before sending it through a communication channel to the cloud. The purpose is to protect the patient's medical information, and the channel service provider may be required to include authentication information in the transmitted data. The image uploader, or content owner, may send encrypted images rather than raw before employing cloud platform services, as this example demonstrates [5].

[1] *Research Scholar, Dept. of CSE, JNTU Hyderabad, India shirishasai34@gmail.com

[2]Professor, Dept. of CSE JNTU Hyderabad,, India

Additionally, the hidden data may require the network provider to include specific data in the encrypted image for management purposes, including the encrypted content owner details, timestamp, tags, and source data. The user can obtain the original image, hidden text, and key by downloading image encryption, and then apply the RDH method to the encrypted image to utilize the RDHEI technique. The technique has the potential to secure secret communication while also allowing for its reversibility. Instead of using the plaintext image, the RDHEI method integrates the hidden message into the encryption image. This accurately retrieves hidden data and permits the undamaged restoration of original plaintext images. To ensure its privacy in multimedia sources, as well as the protection of copyright and content integrity, the RDHEI technology combines the benefits of RDH with encryption.

Research on RDHEI has been limited, with [6] using AES encryption for every extra bit. However, disrupting extra data results in a significant decrease in the decrypted image's PSNR. The study in [7] aimed to eliminate extraneous data by compressing the lower-left bit fields (LSBs) of the encrypted image. Data extraction can proceed independently of image decryption, allowing the receiver to restore the original image using the spatial correlation of decrypted images. The data hider used the LDPC algorithm to compress half of the fourth LSB in the encrypted image after first encrypting the original material [8]. The remaining half of the fourth LSB can be utilized for storing both compressed and extra data. To make the embedding rate faster, the authors of [9] used a block permutation and a stream cypher, which had an encryption method with bit substitution in the prediction error. The work in [10] suggests that RDHEI can be achieved by transferring unnecessary data from the unencrypted image to the encrypted one. Additionally, researchers have developed novel steganography algorithms, such as those found in [11–13]. However, there is still room for improvement in the embedding capability of the methods mentioned before. Traditional encryption approaches face significant challenges due to modern cyber threats and the exponential growth of computing power. Emerging as potential solutions to these problems are new techniques that make use of chaos theory. The chaotic-based cryptosystem offers a new choice for image encryption, addressing a gap discovered in previous work on chaos-based image encryption.

To begin, more security testing is required to ensure sufficient confidence in the encryption process, since the most recent published work [14, 15] uses a single chaotic map to encrypt images. The second point is that previous efforts to encrypt pictures used either a chaotic map alone or a blend of the two, such as the block cypher approach and a chaotic map [16-18]. A well-known chaotic system, the logistic map is known for both its complexity and its seeming simplicity. There is a lot of literature on its use in cryptography, especially for producing pseudo-random sequences [19]. The tent map, known for its simple dynamics and uniform distribution, is one such chaotic system that has proven useful in improving encryption methods [20]. The task aims to develop a more advanced and secure method for data hiding and encryption by integrating two maps into a three-dimensional chaotic system.

In order to make the RDHEI process more resilient and unpredictable, this study proposes a hybrid 3D chaotic system that uses the complicated dynamics of logistic and tent maps. This method guarantees the reversibility of the data hiding procedure, enhancing the security of the concealed data and enabling precise recovery of the original image after data extraction. In this study, we outline the mathematical foundation of the hybrid 3D chaotic system, detailing its construction and properties. We then discuss its application in RDHEI, focusing on its implementation within AWS cloud services to protect user privacy. Our experimental results demonstrate the system's efficacy in resisting common cryptographic attacks while maintaining computational efficiency.

The following objectives will enable us to overcome the previously mentioned limitation. The image encryption solution we recommend employs a chaotic map, namely the Logistic map, in combination with a tent map, resulting in a chaotic hybrid map. Second, we examine and contrast these three image-collect encryption approaches to find the most successful. We use four generally established criteria to evaluate the effectiveness of their job in the third part of the investigation: A visual evaluation, differential analysis, calculation speed, and statistics are all included in this paper.

2. Related works

Image encryption can be performed using a block cipher and a chaotic layout, which can reduce the number of encryption cycles. However, a thorough examination of these techniques is necessary to protect against cryptanalytic attacks.

In [21], the authors achieved a low embedding rate by combining the Paillier cryptosystem with the stream cipher in a hybrid encryption approach. Once they split the original image into non-overlapping chunks, they apply a hybrid encryption that has a low expansion rate. They identified these blocks as belonging to one of two types, then encrypted them. An adaptive and optimizing mechanism determines the embedding ability of these two different types of blocks. Private data will be added to image files by altering cipher texts through key data hiding. After recovering the embedded data during the decoding stage, they reconstructed the original image using an encrypted key and data hiding. Unlike other state-of-the-art methods, the experiment outcomes demonstrate a superior result at a comparatively high PSNR. In [22], the authors suggested a novel method they called "reversible data concealing in the encrypted image" (RDH-EI) using IWT (integer wavelet transform) and a chaotic system. This method delivered high-quality information along with the decoded image. This method uses an IWT transform to split the solute carrier image into wavelet elements. The chaotic system generates a location sequence, an encryption sequence, and a scrambled pattern for data concealment and image encryption, respectively. The positioning series scrambled the sensitive information in the diagonal component, then used encryption and a scrambled sequence to secure the data in the approximated parts, resulting in the final encrypted image. To solve the problem of losing pixels during the reconstruction step, secure a wavelet portion. The encryption technique concealed the image that accompanied the data, ensuring the solutions were both secure and carried a maximum payload. The various layers of security required maintenance, splitting the decryption key into two halves to extract the image's hidden information.

In [23], the authors chose four sub-images from a sampling of the original image. They used the Arnold transformation and two secret keys to jumble the sub-images before reconstructing them into a secure image. Next, they updated the encrypted image by changing the difference value between two nearby pixels. To recover a decrypted image, the receiver can use the decryption keys and an image with additional encryption information. The user can retrieve the hidden data and accurately reconstruct the first image using the decryption and data-hiding keys. The experiment's findings demonstrate that the suggested strategy can acquire a greater payload while maintaining a practical image. In [24], the authors presented the adaptive gradient prediction (AGP) approach. Using a challenging and effective local complexity measuring technique, the AGP makes predictions. These projections are based on pixel changes in the areas surrounding the prediction pixel in all directions (horizontal, vertical, and diagonal). The experimental results clearly show that the AGP-RDHEI approach provides measurable embedding rate gains. In [25], the authors proposed, based on signal reconstruction, the concept of “reversible data hiding in the encrypted domain” (RDH-ED). The concept entails making an encrypted signal using non-expandable symmetric encryption (NSE). NSE employs homomorphic encryption with a short-bit key during signal reconstruction to ensure the integrity of the original signal. Homomorphism makes data hiding simple to implement, as it allows for the construction of the decrypted password in a homomorphic form. The suggested method rebuilds an NSE-based signal using either homomorphic symmetric encryption or homomorphic public-key encryption. In contrast, homomorphism can exist in a variety of configurations, including multiplicative homomorphism and additive homomorphism. Additionally, they utilize multiplication to construct symmetric public-key addition homomorphisms.

In [26], the authors developed a method that simultaneously uses several chaotic techniques, streams, and hash functions. The SHA-1 hash method encrypts key inputs to enhance operational security. It can also produce more dynamic keys during the chaotic and stream encryption phases. Initially, they break up the image into small chunks, then give each encrypted block a unique key pattern before encrypting it with chaotic keys which refer to as chaos encryption. The remaining step involves reformatting all blocks into whole images, enabling encryption using the stream technique. They conducted numerous experiments using RGB images to assess encryption strength using various methods such as entropy, histogram analysis, UACI, NPCR, SSIM, PSNR, and avalanche phenomenon. As a result, the average entropy value of 7.9996 and the avalanche effect value of 50.0366 demonstrate the recommended method's resistance to a variety of attacks, including statistical attacks. In [27], the authors used a structure using diffusion, permutation, and diffusion. Using 3D cues, this work defines and applies the view planes of color images in both diffusion phases. They suggested using a 3D Zigzag transformation to break the association among R, G, and B components during the permutation step. For the purpose of creating a novel PRNG, they also merge two chaotic systems. The suggested method offers satisfactory performance and excellent security, according to experiments and algorithm evaluations.

In [28], the authors suggested a 3D chaotic map-based encryption technique. This map employs a nonlinear ciphering procedure to achieve pixel value diffusion and position permutation. The proposed cryptosystem performs five operations on the medical pictures to ensure a high degree of security. A few examples of these operations include XOR, row and column rotation, 3D chaos creation, and chaos histogram equalisation. They examined numerous medical images, each with different features, to verify the proposed cryptosystem. In addition, it provides healthcare, IoMT, and cloud application platforms with high levels of resilience and recommended security. The proposed 3D chaotic cryptosystem achieves an average entropy of 7.95 and NPCR averages 99.62%, both nearing their maximum values of 8 and 99.60%, respectively.

3. Materials and methods

The RDH-EI approach in the proposed system typically involves three different parties, as shown in Figure 1. The data hider, the receiver, and the content owner. The image can be encrypted and transmitted to the data hider only after the content owner pre-processes it to reserve room space in the RRBE scheme. Now, the data hider will encrypt secret data into the images. Last, the recipient can get private information using the hiding key K_h *and* a first image using an encryption key. K_e. Alternatively, the receiver can hold two keys at the same time to perform both actions. The momentum of this stage can be divided into two separate arguments. First, the proposed approach should provide greater security. At this time, the information previously visible in the image will have been extracted, making it imperceptible, and preserving the confidentiality of the data included in the original image. The second thing to remember is to make better use of redundant space. At this point, the redundant space in the initial image will be converted to segmented blocks, allowing this redundancy to be used more efficiently. To encrypt the image, this work used two chaotic maps: a tent map and a 3D logistic map. Fig. 1 depicts the general encryption process. The 3D logistic map and tent map are introduced first, followed by the suggested encryption and decryption technique.

Block-based scrambling

Block-based scrambling is a technique used in image encryption to rearrange the positions of image pixels. It works by dividing the image into fixed-size blocks and then shuffling them according to a specific algorithm. The source image is converted to a pixel array using appropriate operations. The introduced two-pass Block-XOR procedure can be applied row-wise or column-wise. First, a two-pass block scrambling algorithm named Block-XOR is performed on the pixel array column-wise, as shown in Figure 2. The pixel array is partitioned into data and key columns during the initial iteration. It is often written as a set of *M* data columns and one key column. In a 512 × 512 image, there are 4 * 128 column blocks, i.e., (3 + 1) * 128, where 3 and 1 represent the number of data columns and key columns, respectively. An XOR operation is applied between the data and key column values. In Figure 2, the Block-XOR algorithm considers a portion of the 6x6-pixel area of the input image. The chosen pixels are divided into 3 × 2 column blocks, or (2+1) × 2 column blocks, each of which has two data columns and one key column. The italic and highlighted columns are the key columns, and the rest of the columns are data columns. The output of pass-1 operations shows that all the data column values have become ciphers.

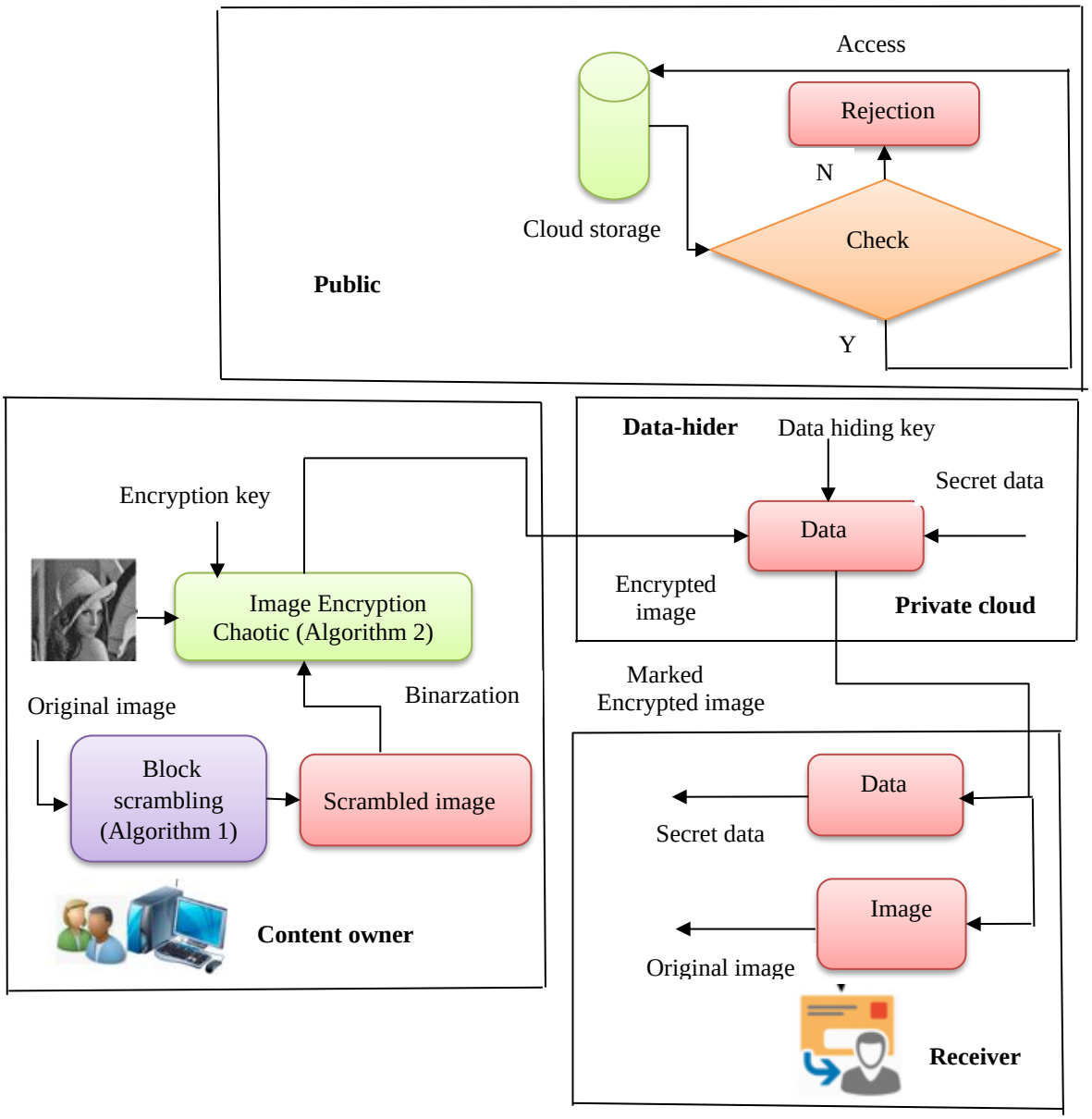

Figure 1: Framework of the proposed method

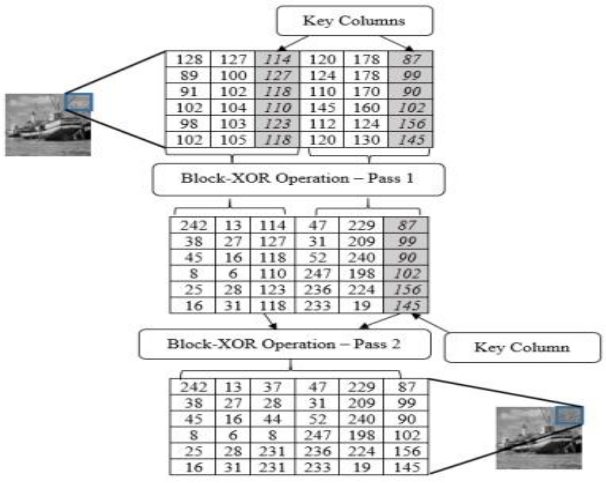

Fig 2. Two-passBlock-XOR Operation

In pass-2, the same procedure is applied to the key columns by considering any key column as the key column. Figure 3 explains the reverse Block-XOR operation. The second stage of block scrambling is swapping the pixels in an anti-clockwise crisscross manner within each block, which is explained in Algorithm 1. The objective is to break the correlations between adjacent pixels and obtain an image that does not resemble the original. Thus, getting the input image without the corresponding unscrambling method is impossible.

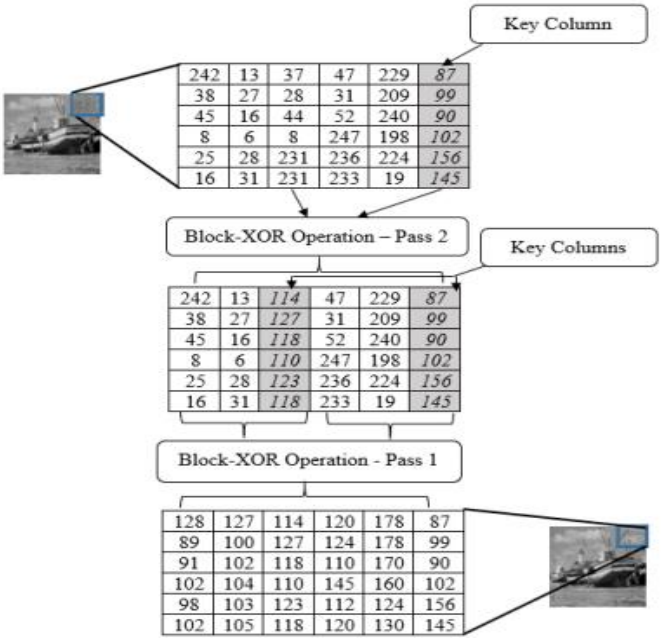

Fig 3. Two-passBlock-XOR reverse operation

Algorithm 1: Image Scrambling

Input: Image P

Output: Scrambled image P1

begin
width, height = size(P);

xres=width;
yres=height;
BLKSZ=width/2

for i=2 to BLKSZ+1 do

for j=0 to (xres / i) do

for k=0 to (yres / j)

rot (arr, i, j*i, k*i) //function to rotate array

end
end
end
for *i*=3 to BLKSZ+1 do

for j=0 to (BLKSZ+2-i) do

for k=0 to (BLKSZ+2-i)

rot (arr, BLKSZ+2-i, j* BLKSZ+2-i, k* BLKSZ+2-i)

end
end

end
P^1=arr;
returnP1;
end

Proposed Image encryption

The image obtained from the scrambling process is converted into a pixel array, then encrypted using a mix of 3DTent map and 3D logistic map. The key is a chaotic sequence produced by the logistic map, converted to a binary sequence. There are two stages first one Confusion process, which can to permutate pixel positions, and the Diffusion process it encrypt pixel values. After creating the restricted image, the content owner will encrypt it utilizing logistic tent map encryption and an encryption key. This will happen after the reserved image is generated.

Confusion process: It's a technique that rearranges the order of pixels in an image based on a shuffling pattern. However, chaotic maps like the 3D tent map can be used to generate these shuffling patterns. Tent map [11] ranks among the most studied and commonly used chaotic maps for generating pseudo-random numbers in a variety of applications, including secure encryption:

$$u(n+1) = \begin{cases} \dfrac{u(n+1)}{\mu}; \text{if } u(n) < a \\ \dfrac{1-u(n+1)}{1-\mu}; \text{ if } u(n) \geq a \end{cases} \quad (1)$$

Where *a* is the control parameter, and when is between (0.4 and 0.5) and [0, 1], the state of the generated sequence [0, 1] has a tendency to be somewhat chaotic. This holds true for all of the created series. The 3D tent map is defined by the equations:

$$u(n+1) = \begin{matrix} \dfrac{u(n)}{a} & \text{if } u(n) < a \\ \dfrac{1-u(n+1)}{1-a} & \text{if } u(n) \geq a \end{matrix}$$

$$v(n+1) = \begin{matrix} \dfrac{v(n)}{b} & \text{if } v(n) < b \\ \dfrac{1-v(n+1)}{1-b} & \text{if } v(n) \geq b \end{matrix} \quad (2)$$

$$w(n+1) = \begin{matrix} \dfrac{w(n)}{c} & \text{if } w(n) < c \\ \dfrac{1-w(n+1)}{1-c} & \text{if } w(n) \geq c \end{matrix}$$

Here, *a*, *b*, and *c* are parameters between 0 and 1.

Step 1: Define the image dimensions (width and height).

Step 2: Define the size of the sub-blocks for shuffling

Step 3: Initialize three chaotic variables (*u*, *v*, *w*) for the 3D tent map with initial values between 0 and 1.

Step 4: Iterate through the image, shuffling sub-blocks at a time.

Step 5: For each sub-block:

- Use the chaotic variables (*u*, *v*, *w*) to generate a random permutation that defines the new order of pixels within

the sub-block.

- Update the chaotic variables using the 3D tent map equation.
- Apply the permutation to shuffle pixel positions within the sub-block.

Repeat steps 4 and 5 for all sub-blocks in the image.

Diffusion process: Pixel diffusion alters the pixel values in an image based on a diffusion process. The 3D logistic map can be used to generate pseudo-random values that influence these changes.

The logistic map is one of the most basic but also one of the most interesting maps. Consider the mathematical equation to be a representation of a polynomial mapping.

$$x(n+1) = rx(n)\big(1 - x(n)\big) \qquad (3)$$

where *x* is the map input variable *x*(0) acts as its initial condition, l is the system parameter ϵ (0 < *r* < 4), and '*n*' is several iterations that need to be applied. Depending on the value of l, the logistic map depicts the path toward freedom. When l is less than 3, the value of x, instead of exhibiting chaotic behaviours, emerges in a stable phase of cycle-2 after reaching fixed positions after several repeats. The diagram reveals chaotic dynamics for 3.57 < l < 4 and x(n)˛ (0, 1) for every n if we keep increasing the value of l. The secret key can be expressed by the beginning values l and x(0) when encoding images using the logistic map. Thus, recipient to effectively decode the message, they need to understand the precise values of these two parameters, l and x(0). As a result, the method used for encryption becomes fully key-dependent, making it exceedingly difficult for an attacker or middleman to extract information from encrypted photos. The bifurcation graphic in Figure 4 clearly illustrates this chaotic state.

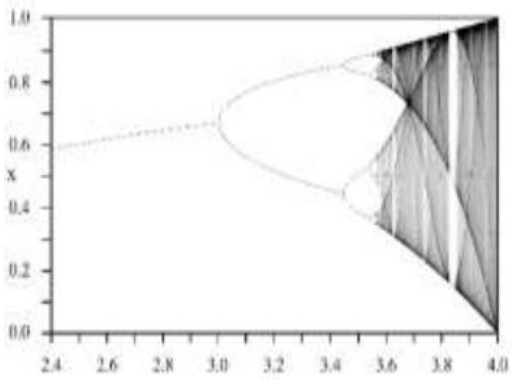

Figure 4: Logistic map

The advanced model of the 1D-logistic map in 3D-LCM is more secure than the first dimension map [37] which extends the various chaotic maps as follows:

$$x(n+1) = rx(n)\big(1 - x(n)\big)$$

$$y(n+1) = ry(n)(1 - y(n) + kz(n)) \qquad (4)$$

$$z(n+1) = rz(n)\big(1 - z(n)\big)$$

k is a constant value used to couple the equations (obtained from *k*1).

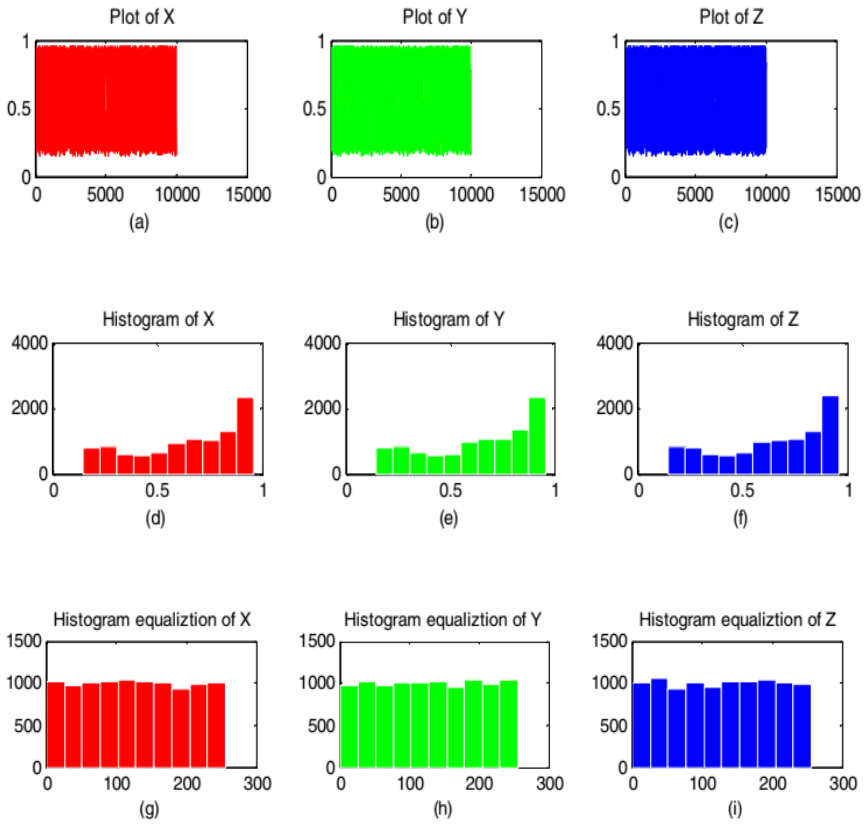

Fig 5: Histogram equalization of 34D chaos

In our study, we observe the chaotic behavior of the extended version derived from the one-dimensional chaotic map in Eqs. (3). The control parameters for the 3D chaotic map are denoted as "r", typically in the range between 3.57 and 4. These control parameters are the primary values, referred to as "r", utilized in our proposed method. The next step involves histogram equalization of the generated keys, as they are initially non-uniform, as illustrated in Fig. 5. This process results in uniformly distributed keys, as indicated in Fig. 3, and is crucial for achieving enhanced privacy by equalizing the histograms of the keys. To generate uniformly-distributed keys for an image of size M × N, we employ the following equations.

$$x = \big(integer(x \times N_2)\big) mod N \qquad (5)$$

$$y = \big(integer(y \times N_4)\big) mod M \qquad (6)$$

$$z = \big(integer(z \times N_6)\big) mod 256 \qquad (7)$$

where *N*6, *N*4, and *N*2 are large arbitrary numbers generally greater than 10,000 (Wen et al. 2016). For the simplicity of the suggested cryptosystem, the *N*2, *N*4, and *N*6 are selected to be equal. 3D logistic map and Tent map include the mixed encryption for the proposed model these steps are as follows:

Step 1: To obtain an extra scrambled image for iteration =10 to set the parameter of Tent map this map applied color image (gray scale) square input with size (M,N) to shuffle portion of the pixel location to Eq. (8). For new pixel location are at u_{new}, v_{new}, z_{new} .

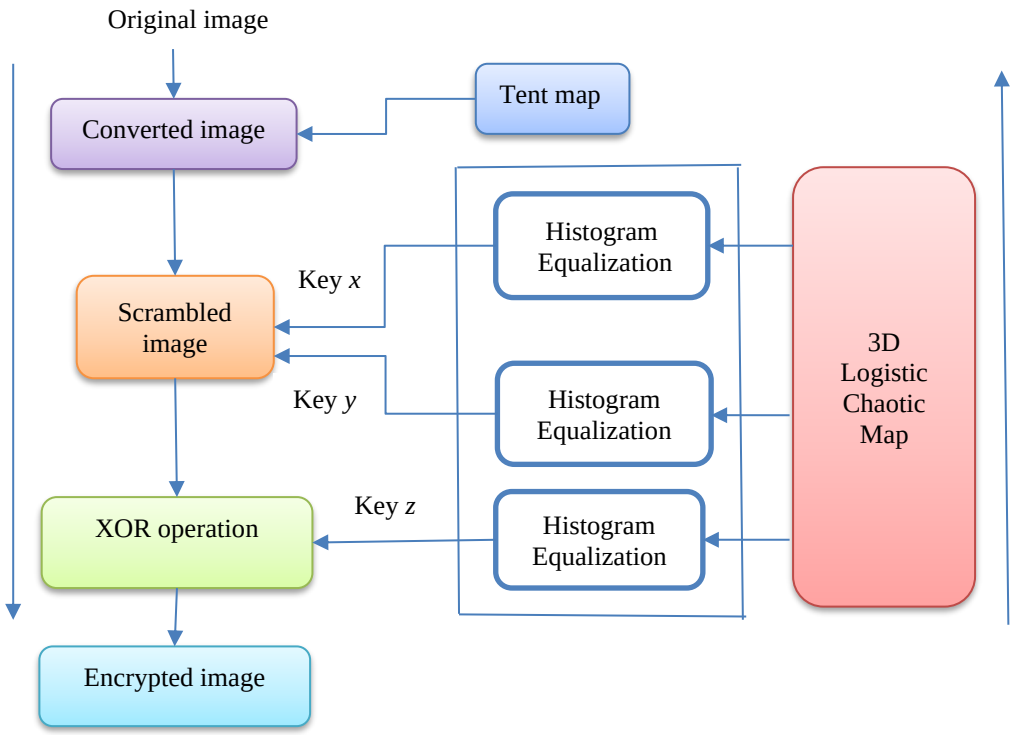

Fig. 6 Proposed 3D chaotic encryption-based image cryptosystem

Step 2: The 3D logistic map produces three chaotic sequences (x, y, z) by utilizing the parameters in Eq. (5) to (7) as mentioned above. In the pixel permutation process, circular row and column rotation is carried out. Large random numbers are generated for K1 and K3 to determine the indexing of row circular right rotation and column circular down rotation respectively. The rotation number aligns with the x chaos sequence and y chaos sequence, thus enabling the permutation of image pixels along x and y.

Step 3: It takes a further step to change the pixel value for improved encryption. Without understanding the ordered key of the chaotic process, the XOR operation alters the pixel values in a way that makes it difficult to undo. As the index for this process, a big random number, K5, is created. The sequence number (z), which is obtained by Eq. (7), and the converted (permuted and transformed) picture to one pair are subjected to an XOR operation.

Decryption **Method**

To retrieve the original image, effectively in the encryption process to apply the XOR operation, inverse Tent map, and 3D chaotic map.

Algorithm 2: chaotic map (*I*)

Input: Input: Any Square image M × N

Output: New encrypted image

1 *num* n, *row* r, *col* c

2 *Ia* TentMap(I)

3 *I*1(*m*, *m*) Shuffle(I)

4 (*C*1, *C*2) LogisticMap()

5 (*Sx*, *Sy*) (*Sort*(*C*1), *Sort*(*C*2))

6 for *i* = 2 to *m* **do**

7 for j = 0 to m **do**

8 if ($mod(j, 2) == 0$) **then**

9 I ' '$(i, j) = I'(i - 1, j - 1)(XOR)i_x(k)$, k = 1, 2, 3. . . . . ..(m/2);

10 else

11 $I'' (i, j) = I'(i - 1, j - 1)(XOR)i_y(k)$, k = 1, 2, 3. . . . . ..(m/2);

12 end if

13 end for

14 end for

15 return *Ie*

Data Hiding

Because shuffling-based encryption doesn't change the original image's histogram, the histogram-shifting technique helps hide data within an encrypted image. The method works like the following:

1. A pseudorandom permutation of the hidden key was used to encrypt the image pixels. Apply the same process used to encrypt the image for this step.
2. The initial few pixels are designated for embedding in headers that will be helpful throughout the retrieval portion of the procedure.
3. The maximum grey level has been introduced to the header. Transform a grey level to an 8-bit binary sequence, then use a single plain LSB substitution to embed it in the first eight pixels of the image.
4. A histogram of the remaining pixels is produced, and its grey level is the highest among the total pixels chosen for embed.
5. When a constant unity accumulation is applied to the pixel grey values over the entire array, from the highest to the lowest pixel count, the histogram shifts (zero is not considered in this counting process). The strategy will create a blank space in the histogram for the following grey level, corresponding to a grey level with the most images.
6. Transform the information into the correct format to enter the binary order.
7. Search for the maximum pixel count's pixel grey levels and change the pixel's grey level with a data stream in binary numbers intended to remain hidden. If a binary bit is "1", the grey pixel level should increase by one; otherwise, it should remain constant.
8. Go to the following step for all pixels that fulfil the criteria given in the previous step.
9. To recover the encrypted version of the image, perform an inverse shuffle on it.

Image and data recovery

On the receiver side, we investigated two techniques depending on the key supplied, and the third one might be constructed by combining the first two.

Data hidden key only:

The hidden data can be retrieved only when the user has been providing information hiding the key to a designated encrypted image; however, the encrypted cover cannot be decrypted without the data hiding the key. To retrieve the data, the receiver must follow the same steps as the data-hiding module but in reverse order. The first receiver creates the chaotic sequence using the data-hiding key provided to them and the method described in the image encryption module. Do pixel shuffling on the marked stego-cover using the chaotic sequence to acquire its header parameter and retrieve the hidden data. This will provide access to the header parameter. By reading the least significant bits of the beginning 8 bits, we may establish the level of grey used for information hiding. The next

step is to do a sequential scan within the unnecessary pixels for header embedding for the produced grey level in the manner described above and the unity addition grey level of the obtained value. When a maximum pixel count grey level is encountered, the value '0' is taken from the bit, and when the maximum pixel count unity accreted grey level is met, the value '1' is taken from the bit. These extracted bits are stored in a buffer until they can be retrieved and utilised to generate meaningful embedded data. Apply a unity decrement to the pixels in the range of the grey equivalent with the greatest pixel count down to the grey levels corresponding to the minimum number of pixels when data extraction is complete. This will bring back the dislocated histogram. The block should then be reshuffled in the opposite direction to obtain an encryption cover.

The encryption key alone is provided:

The encryption module's current key calculation procedure and encryption approaches have been used. Suppose a user is given the encrypting keys to be informed that the wrap is a steganography attempt. In that case, they will be ready to decrypt the image like the original cover, even though they cannot study its data. Construct a chaotic sequence (*1*) with the approach already provided. Arrange pixels in the image using the map of the location developed when sorting through the chaotic sequence. This process should be able to return all pixels to their original locations. A decrypted image has only one pixel of distortion, a difference of 1 in the grayscale value of the pixels used for data hiding.

4. Results and Discussion

In this part of the paper, we will review the efficacy of the proposed technique by running experiments on a collection of standard gray-scale images. As shown in Figure 7, six test images from the USC-SIPI image collection were used as the original protected images: Lena, Goldhill, Airplane, Man, and Boat. These are 512×512 pixels in size. Tiffany, Lena, the Man, and the Aircraft are among the test images.

(a) Lena (b) Goldhill (c) Airplane

(d) Man (e) Boat (f) Tiffany

Figure 7: Six original protected images.

The three essential factors that need to be studied in our suggested system are the assessment of the structural similarity index (SSIM), peak signal-to-noise ratio (PSNR), & embedding capacity (EC). A PSNR, which is determined by Equation (3), and the SSIM were used to assess the image's quality. When the PSNR value increases, the image's quality level increases higher.

$$PSNR = 10log_{10}\frac{255^2}{MSE} \tag{3}$$

$$MSE = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}(D(x,y)^2 \tag{4}$$

Here $M \times N$ are the image sizes & $D()$ is the pixel value comparison matrix between the original and recovered image. The reconstructed image is larger than the original. SSIM mimics a human visual system (HVS) to determine image quality by evaluating brightness, contrast, and overall structure. In general, if the value of SSIM is close to 1 rather than some other number, the stego-image will have a greater perceived quality.

The system's embedding capacity was measured in bits and defined as follows:

$$EC = [\beta \times B^2]\frac{M \times N}{B^2}\ (bits) \qquad (5)$$

where β measures the compression ratio, B indicates the block dimensions, and [] specifies that a value will be rounded to the nearest whole number. Most embedding capacities are customizable and can be changed by the compression ratio. Table 1 lists the experimental data for six different test images from dataset.

Table 1: Relevant experimental data for six different test images

Test Image	PSNR (dB) Baseline	PSNR (dB) Our work	(SSIM) Baseline	(SSIM) Our work	Embedding Capacity (EC) (Bit)
Lena	51.24	52.24	0.9845	0.9910	65,536
Goldhill	49.58	49.98	0.9784	0.9845	65,536
Airplane	49.65	49.88	0.9457	0.9532	65,536
Man	48.57	49.10	0.9841	0.9873	65,536
Boat	49.57	49.73	0.9547	0.9674	65,536
Tiffany	48.97	49.22	0.9684	0.9714	65,536

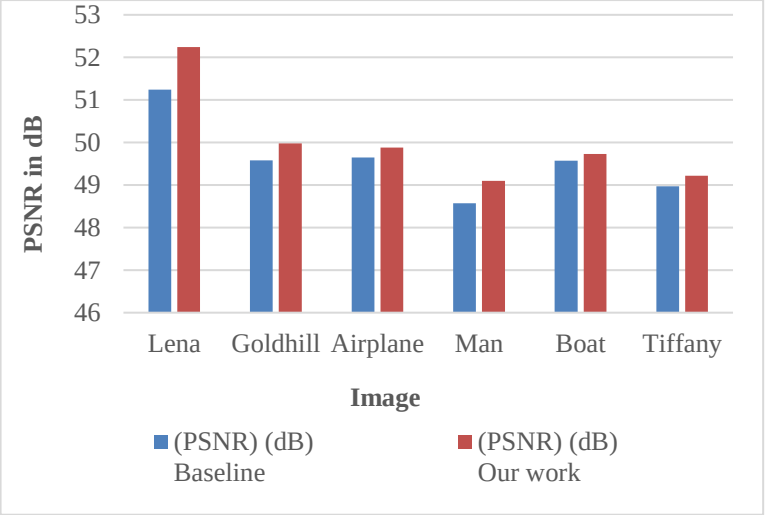

Figure 8. Comparison of PSNR dB on six test images

Figure 8 shows that our proposed work has the highest PSNR of 52.24 dB for the Lena image and the lowest PSNR of 49.10 dB for the Boat image, respectively. Also, our proposed approach has outperformed all six test images compared to the Baseline approach.

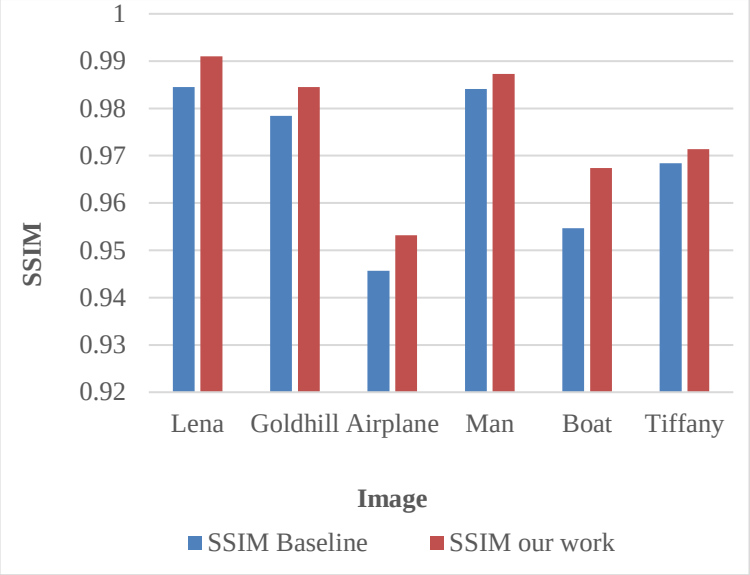

Figure 9. Comparison of SSIM on six test images

Figure 9 shows that our proposed work has the highest SSIM of 0.9910 for the Lena image and the lowest SSIM of 0.9532 for the Airplane image, respectively. Also, our proposed approach has outperformed all six test images compared to the Baseline approach.

Security Evaluation of the Suggested RDHEI Scheme

Table 2 displays the PSNR (peak signal-to-noise ratio) for each encrypted image and a correlation coefficient that describes the connection between that image and the original matching picture. After completing the encryption process, it is clear that the PSNR of each encrypted image is less than 10. It is the common finding in all encrypted images. As a result, decrypting the original image data, I_o obtained from the encrypted image I_e, is a complex procedure to implement. A statistical approach is the correlation coefficient which determines how closely two variables are related based on how their relative positions vary over time. This is accomplished by comparing how the relative positions of the variables change. We employ the correlation coefficient derived in Equation (6) to demonstrate how the proposed RDHEI system exerts its influence.

$$\rho = \frac{N\sum_{i=1}^{N}(x_i \times y_i) - \sum_{i=1}^{N} x_i \times \sum_{i=1}^{N} y_i}{\sqrt{(N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2) \times (N\sum_{i=1}^{N} y_i^2 - (\sum_{i=1}^{N} y_i)^2)}} \quad (6)$$

Here, x and y are the grayscale value of two pixels that have the similar values chosen between I_o and I_e, and N is image's total pair of pixels.

Table 2: PSNR of encrypted images, labelled encrypted images, and correlation with original images.

Images	**PSNR (Marked Encrypted Image**	**Correlation Coefficient**	**PSNR (Encrypted Image)**	**Correlation Coefficient**
Lena	9.3321	−0.0009	9.4214	−0.0008
Goldhill	9.5614	-0.0001	9.6124	-0.0006
Airplane	8.1235	0.0024	8.1142	0.0042
Man	8.1240	-0.0003	8.2147	0.0002
Boat	8.5417	-0.0007	8.4567	-0.0008
Tiffany	8.8414	-0.0020	8.7891	-0.0021

Comparisons with Similar State-of-the-Art Techniques

First, the max & averaged ER of each approach are compared in Table 3. The embedding rates of designated encrypted pictures are then averaged.

Table 3: Comparisons of max ER (embedding rate) of several pictures utilising the proposed approach and state-of-the-art techniques.

Images	Puteaux et al. [17]	Yi et al. [18]	Chen et al. [19]	Our work
Lena	0.977	2.014	1.944	2.489
Goldhill	0.983	2.457	2.338	2.654
Airplane	0.839	0.462	0.535	1.987
Man	0.990	2.442	2.648	2.741
Boat	0.976	2.147	1.871	2.412
Tiffany	0.996	2.474	2.110	2.547

Figure 10 shows that our proposed work has the highest embedding rate of 2.489 for the Lena , 2.654 for the Goldhill, 1.987 for the Airplane, 2.741 for the Man, 2.412 for the Boat, and 2.547 for the Tiffany images, respectively. Also, our proposed approach has outperformed all six test images compared to the Baseline approach.

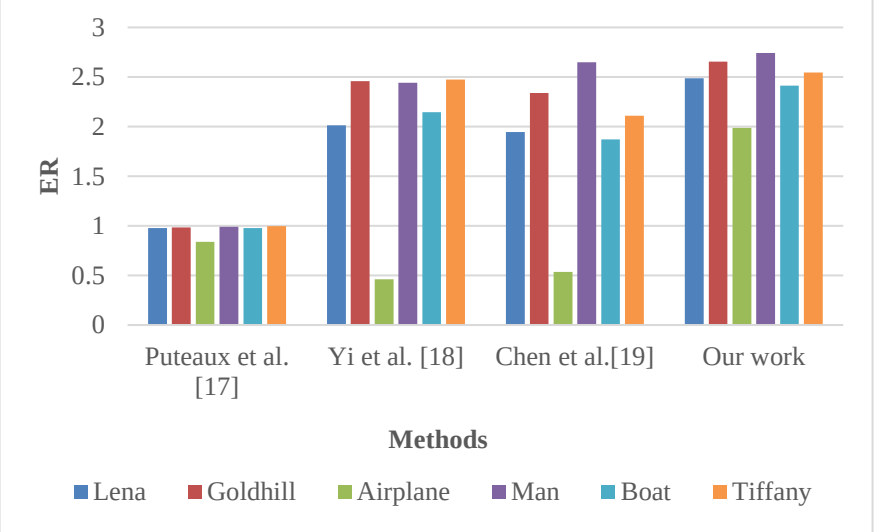

Figure 10. Comparison of 4 methods for ER of 6 test images

A. The Encryption Example

In order to confirm the algorithm's validity, the experiment has been taken. Set an image of size 256× 256 and the initial keys are: $x(1)$=0.2350; $y(1)$=0.3500; $z(1)$=0.7350; α=0.0125; β=0.0157; γ=3.7700, $N2$=N4=$N6$=100000, $N1$=5000, $N3$=6000, $N4$=7000.

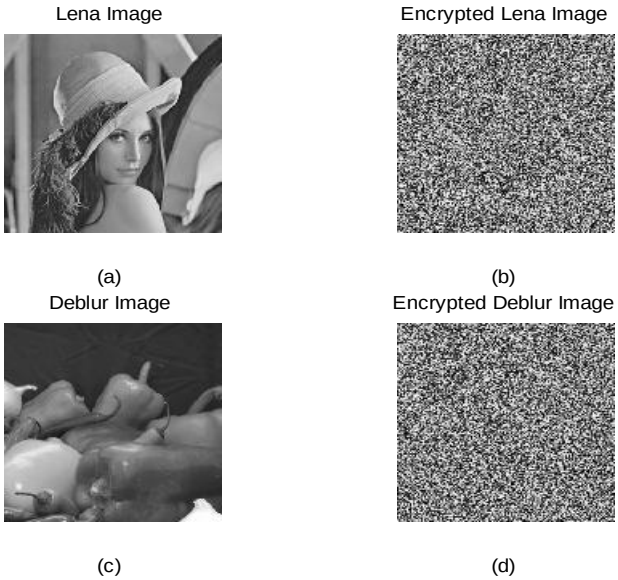

Figure 11. Encryption for Lena and Deblur image.

In Fig.11 shows the encryption example. Among them (a) Original Lena image, (b) Encrypted Lena image, (c) Original Deblur image and (d) Encrypted Deblur image. From the figure we can see that pixels are diffused properly and completely different from original image.

B. Statistical Analysis

It is clear from examining the strong correlation value between neighboring pixels that statistical attacks are a serious risk for image encryption. To demonstrate the superior confuse and spread features of the encryption, which act as a strong defense against such attacks, a thorough statistical study has been carried out. Histograms of the original and encrypted image are shown in Figure 12, with the original Lena image labeled as 12(a) and the encrypted Lena image as 12(b). In addition, the Deblur image's histograms are shown in 12(c) and 12(d), and the Mandrill image's histograms are shown in 12(e) and 12(f). The encrypted image's histogram shows a consistent distribution of pixel values and is free of any data that may be used by unauthorized individuals.

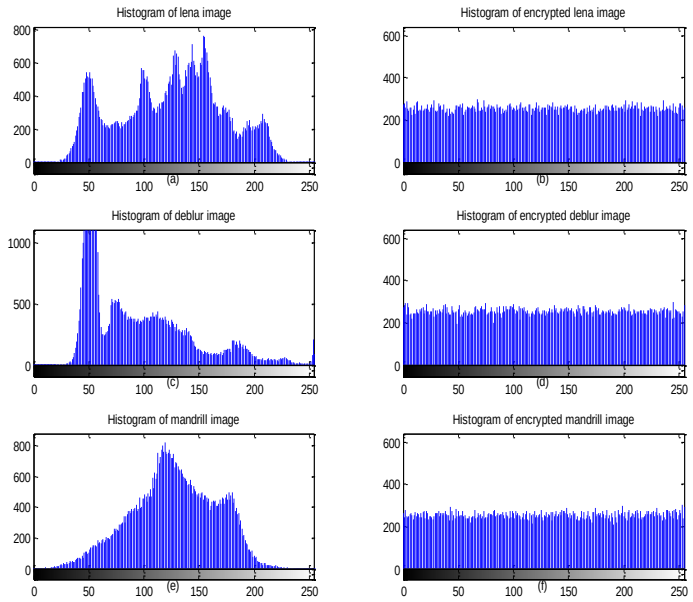

Figure 12. Histogram of original and Encrypted Lena, Deblur, and Mandrill image

C. Key Sensitivity Analysis

A single change of key in the description is very sensitive to secure the encryption algorithm. Highly sensitive encryption algorithms have the key of encryption to decryption.

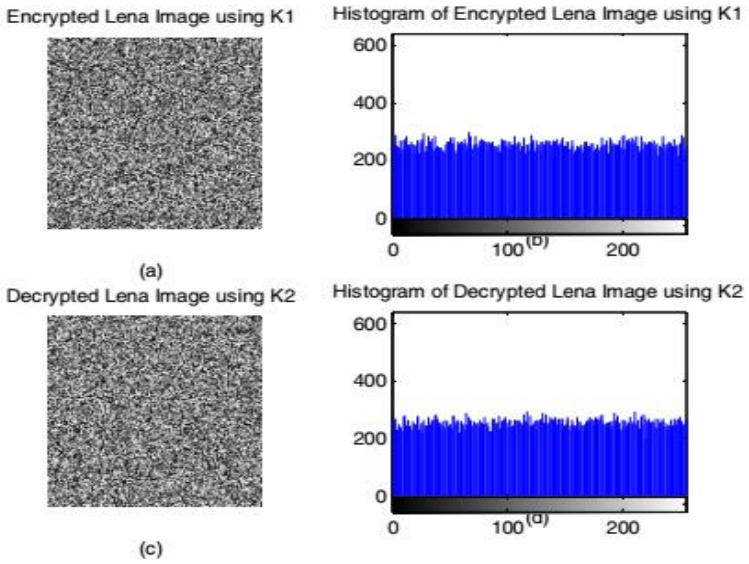

Figure 13. Key sensitivity of proposed algorithm

We encrypt the Lena image (K1) for the test of sensitivity, decode it with a slightly modified key (K2), and plot their histogram, which is shown in Fig. 13. The key we utilized for the key sensitivity analysis appears in Table 4.

Table 4. Sensitivity analysis key list

Key1	Key2
x(1)=0.2350 y(1)=0.3500 z(1)=0.7350 α=0.0125 β=0.0157 γ=3.7700 N2=N4=N6=100000 N1=5000 N3=6000 N4=7000	x(1)=0.2350+1×10^{-17} y(1)=0.3500 z(1)=0.7350 α=0.0125 β=0.0157 γ=3.7700 N2=N4=N6=100000 N1=5000 N3=6000 N4=7000

D. Information and Entropy Analysis

The entropy *H* of a symbol source *S* can be calculated by following the equation.

$$H(S) = -\sum_{i=0}^{N-1} P(S_i)\, log_2 P(S_i)$$

When the entropy is given in bits and P (Si) denotes the probability of symbol Si. The ideal value of entropy for message source S is represented by H(S) = 8, which corresponds to a genuine random source if the source S emits 28 symbols with equal chance, i.e. S = {s1, s2... s256}. The information entropy increases with the degree of uniformity in the gray value distribution. An encrypted picture may become predictable, endangering the security of the image, if its information entropy is far lower than the optimal value of 8. Nonetheless, the information entropy values that the suggested technique yielded for the case of encrypted photos are quite near to the optimal value of 8, the entropy values of the encrypted images are shown in Table 5.

Table 5. Information entropy of encrypted images for various test image

Lena		**Peppers**		**Mandrill**		**Deblur**	
Plain Image	Cipher Image	Plain Image	Cipher Image	Plain Image	Cipher Image	Plain Image	Cipher Image
7.4467	**7.9890**	**7.5553**	**7.9896**	**7.2636**	**7.9895**	**7.0207**	**7.9887**

E. Plain -text Sensitivity Analysis

When the plaintext is altered, it's critical to ensure the cipher image remains unaffected. Analyzing the sensitivity of the plaintext attacks involves using two criteria, NPCR (Number of Pixel Change Rate) and UACI (Unified Average Changing Intensity). NPCR measures the percentage of different pixel numbers between two cipher images, while UACI calculates the average intensity of differences between the two cipher images of M×N as described in the following:

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

$$UACI = \frac{1}{M \times N} \sum_{i,j} \frac{\lfloor C1(i,j) - C2(i,j) \rfloor}{255} \times 100\%$$

Where, *C1* and *C2* are two different cipher images encrypted by using a different keys, where *D*(*i* , *j*) is defined as follows:

$$D(i,j) = \begin{cases} 1 \; if \; C1(i,j) \neq C2(i,j) \\ 0 \; if \; C1(i,j) = C2(i,j) \end{cases}$$

Based on the calculations, the Average NPCR and UACI have been determined and are presented in Table-6 for various test images. Upon reviewing Table-6, it is evident that the NPCR stands at approximately 99.6%, with the lowest value recorded at 99.5880%, and the UACI is approximately 33.5%, with the poorest value being 33.5044%. These results are deemed satisfactory for image encryption.

Table 6. Plain text sensitivity analysis for various test image

Parameter	**Lena**	**Peppers**	**Mandrill**	**Deblur**
NPCR (%)	**99.6048**	**99.5972**	**99.6368**	**99.5880**
UACI (%)	**33.5044**	**33.5189**	**33.6354**	**33.5170**

F. Correlation Coefficient Analysis

In order to evaluate the encryption quality of the proposed encryption algorithm, the correlation coefficient is used calculate the correlation coefficients between two vertically, horizontally adjacent pixels of an encrypted image, the following equation is used.

$$\gamma = \frac{Cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}$$

$$D(x) = \frac{1}{M} \sum_{i=1}^{M} (x - \bar{x})^2$$

$$Con(x,y) = \frac{1}{M} \sum_{i=1}^{M} (x - \bar{x})\,(y - \bar{y})$$

where x, y are the values of the pair of randomized images and M is the number of randomized pairs. The results shown in Table 6 show how well the suggested strategy randomized the pixels. The correlation for a 256x256 Deblur picture appears in Fig. 14. Among those are (a) the encrypted picture (b) the original image's horizontal correlation (c) and (d) the vertical correlation. Figure 4 illustrates how, despite the original image's strong association with its neighbouring pixels and values being dispersed close to the center, the correlation value is lowered following encryption when pixel values become equally distributed.

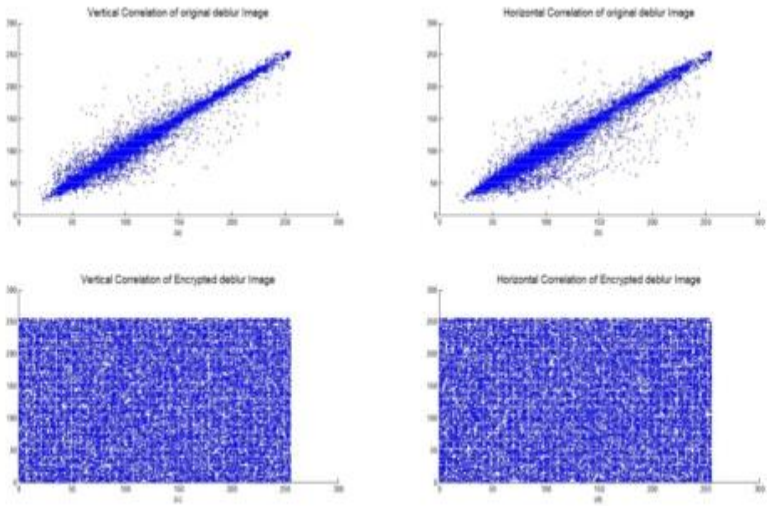

Figure 14. Correlation of original and encrypted Deblur image

5. Conclusion

This article presents a high-reversible hidden data approach based on encrypted format, using the Hamming Code to implement an LSB plane into the image's various planes. The study compares coupled chaotic encryption cryptographies for larger key spaces and higher attacking resistance. The 3DTent and Logistics Map combination performed the best in various test images. The proposed scheme uses a 1-dimensional sequence for chaotic maps and a 2-dimensional sub-chaotic matrix to generate an encrypting matrix, which is used to execute an XOR operation with the original pixel matrix. The major contributions of this paper include an automated selection mechanism, known as the sliding pixel window, which affects the system's performance and efficiency. With the help of this technique, it is possible to decrypt high-resolution photos quickly and ultimately achieve high-level encryption. In the sliding pixel window, the diffusion stage eliminates every potential pattern. In the first- and second-level encryption operations, two hyperchaotic systems with a logistic map generate pseudorandom numbers to hide the original data of each sub plain picture independently. The suggested picture encryption scheme offers a high level of security and renders brute-force assaults impossible. The technique may be applied in real-time and further refined, with potential applications in communication systems. Subsequent investigations may use an additional layer to generate appropriate starting circumstances for automated correction of chaotic iteration. The proposed approach makes data processing, storage, and transfer of user data inside cloud computing more convenient and secure.

References

[1] Saini, Dinesh & Kumar, Krishan & Gupta, Punit. (2022). Security Issues in IoT and Cloud Computing Service Models with Suggested Solutions. Security and Communication Networks. 2022. 10.1155/2022/4943225.

[2] Reza, Hassan & Sonawane, Madhuri. (2016). Enhancing Mobile Cloud Computing Security Using Steganography. Journal of Information Security. 07. 245-259. 10.4236/jis.2016.74020.

[3] Mehrotra, Piyush & Djomehri, Jahed & Heistand, Steve & Hood, Robert & Jin, Haoqiang & Lazanoff, Arthur & Saini, Subhash & Biswas, Rupak. (2013). Performance evaluation of Amazon Elastic Compute Cloud for NASA high-performance computing applications. Concurrency and Computation: Practice and Experience. 28. 10.1002/cpe.3029.

[4] Milhem, Hind & Harrison, Neil. (2022). The Quality Attibutes and Architectural Tactics of Amazon Web Services (AWS). 1-6. 10.1109/IETC54973.2022.9796821.
[5] Liu, Dandan & Shen, Jian & Xia, Zhihua & Xingming, Sun. (2017). A Content-Based Image Retrieval Scheme Using an Encrypted Difference Histogram in Cloud Computing. Information. 8. 96. 10.3390/info8030096.
[6] W. Puech, M. Chaumont, and O. Strauss, "A Reversible Data Hiding Method for Encrypted Images", Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, Proc. SPIE, 6819, Feb, 2008.
[7] X. Zhang, "Separable reversible data hiding in encrypted image", IEEE Trans. Inf. Forensics Security, vol. 7, no. 2, pp. 826–832, Apr. 2012.
[8] X. Zhang, Z. Qian, G. Feng, and Y. Ren, "Efficient reversible data hiding in encrypted images", J. Vis. Commun. Image Represent., vol. 25, no. 2, pp. 322–328, Feb. 2014.
[9] Q. Li, B. Yan, H. Li, N. Chen, "Separable reversible data hiding in encrypted images with improved security and capacity", Multimed Tools Appl., vol. 77, no. 23, pp. 30749-30768, Dec. 2018.
[10] Z. Liu, and C. –M. Pun, "Reversible Data-hiding in Encrypted Images by Redundant Space Transfer", Inf. Sci., vol. 433-434, pp. 188-203, Apr. 2018.
[11] X. Liao, J. Yin, M. Chen, and Z. Qin, "Adaptive payload distribution in multiple images steganography based on image texture features", IEEE Trans. Depend. Sec. Comput., DOI: 10.1109/TDSC.2020.3004708.
[12] Mohammadi, M. Nakhkash, and M. A. Akhaee, "A high-capacity reversible data hiding in encrypted images employing local difference predictor", IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 8, pp. 2366–2376, Aug. 2020.
[13] F. Chen, Y. Yuan, H. He, M. Tian, and H. Tai, "Multi-MSB compression based reversible data hiding scheme in encrypted images", IEEE Trans. Circuits Syst. Video Technol., DOI: 10.1109/TCSVT.2020.2992817.
[14] Pak, C.; Huang, L. A new color image encryption using combination of the 1D chaotic map. Signal Process. 2017, 138, 129–137.
[15] Wu, Y.; Noonan, J.P.; Yang, G.; Jin, H. Image encryption using the two-dimensional logistic chaotic map. J. Electron. Imaging 2012, 21, 013014.
[16] Arab, A.; Rostami, M.J.; Ghavami, B. An image encryption method based on chaos system and AES algorithm. J. Supercomput. 2019, 75, 6663–6682.
[17] Abdullah, H.N.; Abdullah, H.A. Image encryption using hybrid chaotic map. In Proceedings of the 2017 International Conference on Current Research in Computer Science and Information Technology (ICCIT), Dhaka, Bangladesh, 26–27 April 2017; pp. 121–125.
[18] Chen, C.; Zhang, H.; Wu, B. Image Encryption Based on Arnod Transform and Fractional Chaotic. Symmetry 2022, 14, 174.
[19] Pareek, N. K., Patidar, V., & Sud, K. K. (2006). Image encryption using chaotic logistic map. Image and Vision Computing, 24(9), 926-934. https://doi.org/10.1016/j.imavis.2006.02.021
[20] Patidar, V., Pareek, N. K., & Sud, K. K. (2009). A new substitution–diffusion based image cipher using chaotic standard and logistic maps. Communications in Nonlinear Science and Numerical Simulation, 14(7), 3056-3075. https://doi.org/10.1016/j.cnsns.2008.10.005.
[21] Chang, Qi. (2020). Reversible Data Hiding in Encrypted Images Based on Hybrid Cryptosystem. IETE Technical Review. 38. 1-7. 10.1080/02564602.2020.1757521.
[22] Meng, Lingzhuang & Liu, Lianshan & Wang, Xiaoli & Tian, Gang. (2022). Reversible data hiding in encrypted images based on IWT and chaotic system. Multimedia Tools and Applications. 81. 1-29. 10.1007/s11042-022-12415-z.
[23] Wu, Dan. (2018). Reversible Data Hiding for Encrypted Image Based on Arnold Transformation. MATEC Web of Conferences. 173. 03088. 10.1051/matecconf/201817303088.
[24] Qin, Jiaohua & He, Zhibin & Xiang, Xuyu & Tan, Yun. (2022). Reversible Data Hiding in Encrypted Images Based on Adaptive Gradient Prediction. Security and Communication Networks. 2022. 10.1155/2022/8757594.
[25] Chen, Bing & Yin, Xiaolin & Wei, Zhuo & Ren, Honglin. (2022). Reversible data hiding in encrypted domain by signal reconstruction. Multimedia Tools and Applications. 1-20. 10.1007/s11042-022-13266-4.

[26] Budiman, Fikri & Setiadi, De Rosal Ignatius Moses. (2020). A Combination of Block-Based Chaos with Dynamic Iteration Pattern and Stream Cipher for Color Image Encryption. International Journal of Intelligent Engineering and Systems. 13. 131-141. 10.22266/ijies2020.1231.12.

[27] Zhang, Xiaoqiang & Gong, Zhengjun. (2022). Color image encryption algorithm based on 3D Zigzag transformation and view planes. Multimedia Tools and Applications. 81. 10.1007/s11042-022-13003-x.

[28] El-Shafai, Walid & Khallaf, Fatma & El-Rabaie, El-Sayed & Abd El-Samie, Fathi. (2022). Proposed 3D chaos-based medical image cryptosystem for secure cloud-IoMT eHealth communication services. Journal of Ambient Intelligence and Humanized Computing. 15. 10.1007/s12652-022-03832-x.