REVERSIBLE DATA HIDING ON ENCRYPTED IMAGES USING GINGERBREADMAN MAP AND RC5 STREAM CIPHER AND MULTI-LAYER EMBEDDING

Bethala Shirisha[1], Dr. V. Kamakshi Prasad[2]

Article History: Received: 12.12.2022 **Revised:** 29.01.2023 **Accepted:** 15.03.2023

Abstract

reversible data hiding, or RDH for short. Within the scope of this paper, we provide a high-capacity RDH-EI method that uses embedding on so many layers. In most cases, it involves three distinct players: the receiver, the info hider, and the material owner. The image has been scrambled inside its original form by the owner of both contents using data encryption. After encryption, there is still the potential for the one who hid the data or the cloud owner to insert extra data into it. The embedded information and the reference image could be retrieved without any loss on the receiving end. This part of the study proposes a method for encrypting images based on a rotation and substitution cipher. During the permutations step, the image's pixels are muddled using the Gingerbreadman map. During the replacement stage, the image's pixels are bit-wise XOR-ed with the output sequence generated by the Rivest Cipher 5 (RC5) stream cipher method. Unaware of the original image's information, the encryption process can insert private documents into image disparities reversibly via a modification to the two-dimensional difference histogram. Furthermore, the information collection and the image restoration processes are absolute, with no error. The outcomes of the investigations prove that the strategy can be implemented effectively and successfully. There are additional security evaluations discussed in this study, including key sensitivity.

Keywords*:* Reversible data hiding (RDH). Cloud computing. Gingerbreadman map, Stream cipher, RC5.

[1]Research Scholar, JNTU Hyderabad.
[2]Professor, JNTU Hyderabad

Email: shirishasai34@gmail.com

DOI: 10.31838/ecb/2023.12.s3.235

1. Introduction

Presently, digitizing data communication and ensuring the confidentiality of digital data transmitted through the internet has become exceedingly challenging for researchers. The cloud service provider, accountable for storing the data, doesn't have permission to access the original data (i.e., plain text). Still, in a few applications scenarios [1, 2], cloud-hosting or data managers may necessarily have to directly include a few additional messages, such as identification or notation data, inside encrypted data to determine if the data has changed or to show who owns the data. Reversible data hiding in encrypted images (RDH-EI) is a crucial branch of reversible data hiding which may embed hidden messages into images while also safeguarding the confidentiality, in the environment openly, of the image [3]. RDH-EI is also known as reverse data hiding in encrypted images. The cryptosystem of the public key is the cornerstone of the huge majority of the recently proposed RDH-EI techniques. However, the expansion rate of the public-key crypto-system is currently at an unacceptably high level, making it more difficult to transmit information. The broad adoption of cloud computing has led to an increase in the number of users who save their personal information on cloud storage [4–7]. The need to protect private information and safely send information has become a new barrier that has gotten much attention. Nevertheless, the present method that inserted external information into image encryption alongside mistake decompression might lead to leakage of image content and a not-so-high embedding rate. The above problem can be solved by developing new algorithms. This work demonstrates a poetry RDH strategy for use in encoded images. This scheme could give a more robust security level and a higher embedding rate. Figure 1 illustrates a cloud storage environment with a sample instance of an application scenario. An image depicts the user data. This image is encrypted on user 1, which could be a desktop computer; it is then sent to cloud server 1, where it is decrypted. Cloud server (i) inserts a label into the encryption algorithm so that it can be used once to verify the source of the file. The produced file may have to be transferred to a different server. That server will be referred to as a server. (ii) It's possible that at a later point in time, the user will demand the file to be downloaded from second server-2 to another device, i.e., device 2, such as the person's phone. Before providing the decrypted image to the user, the other server must retrieve the original encrypted image and find the hidden data. Device 2 can decrypt the image when it has received the recovered image. In this case, we need to use data hiding that can be reversible, i.e., RDH, in the encryption sector.

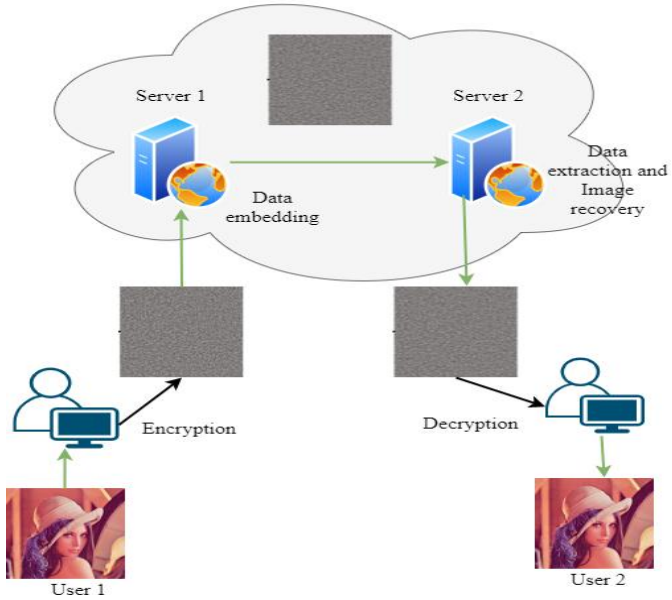


Figure 1 : Illustration of a cloud storage environment with a sample application scenario

Researchers ought to strike the ideal balance between the Embedding Rate (ER) and the PSNR (Peak Signal to Noise Ratio) of the image that has been directly decrypted to create a successful RDH-EI system. The following features of the implementation guidance should be brought to your attention immediately because of their necessity.

i) Integrating RC5 with the gingerbread man map, which would be regarded as a chaotic map, and the black-white multilayered embedding methodology is what we mean when we refer to our suggested RDH-EI scheme.

ii) The envisaged RDH-EI scheme falls into the reserving room before encryption (RRBE) category,

which indicates that the segmentation approach can be completed during the phase in which the content owner is implicated. Furthermore, the specific limit of multi-layer embedding, which further improves the embedding

capacity, indicates that the segmentation approach can be completed during the phase in which the content owner is implicated. Furthermore, the specific limit of multi-layer embedding further improves the embedding capacity during the transition stage in which the data has always been kept secret.

iii) Additionally, it demonstrates that the encryption function can attain improved protection effectiveness after a chosen plaintext attack has been implemented.

.

Related Works

This paper discusses reversible data hiding, often known as RDH, which is an effective method of data hiding and mentions a few significant and recent publications. Whenever it comes to encrypted images, one of the essential capabilities that RDH can provide is the embedding capacity. Several RDH algorithms already used to encrypt images still need to be optimized to the required embedding capacity.

In [8], the authors presented an encrypted image based on adaptive prediction error coding to solve the issue, which they described as a new RDH technique. In order to sustain the free correlation of the actual image within the encoded sector, the proposed RDH algorithm tends to use a block-based encryption scheme. Additionally, it uses a novel technique referred to as adaptive prediction error coding to free up space for data embedding. The adaptive estimated error coding is a significant contribution that the proposed RDH method offers. Being able to empty room from encoded images efficiently by adaptively coding prediction errors by the contents of the blocks, and as a result, it contributes to a large embedding capacity (EC). Several tests on benchmark image databases have been carried out to verify the effectiveness proposed by the RDH algorithm. The findings demonstrated that the mean embedding rates on the spatial databases of BOWS-2, BOSSBase, and UCID are respectively 1.7081, 2.4437, and 2.3083 bpp. The comparison outcomes demonstrate that the proposed RDH algorithm has a greater embedding capacity than many other state-of-the-art RDH algorithms.

In [9], the authors proposed a secure and confidential RDHEI process oriented on APL (adaptive prediction-error labeling). This method makes use of Laplacian-like distributions of prediction errors. Two different predictions that encryption and mix APL have been introduced to improve the quantity of reserved room before encryption. The first takes a pessimistic approach and reveals the shuffled labels in the encrypted images, whereas the second takes an optimistic approach and encrypts the labels. If the recipient owns the secret key, it is possible to completely and separately retrieve the original image and any other data that may have been stored with it. Most notably, the inherent confidentiality of RDHEI is highlighted, the invisibility of image content, and the un-detectability of extra data. This is an important advancement over previous method. According to the outcomes of the experiments, the technique is superior to the previous works in terms of its ability to achieve more robust hiding capacity and appropriate security.

In [10], the authors suggested using a hybrid encryption algorithm that combined the Paillier cryptosystem and the stream cipher to reach a low expansion rate. Hybrid encryption with a low expansion rate is applied to the original image after being cut into non-overlapping blocks. These blocks are then selected as belonging to one of two blocks: encrypted, the adaptable optimization algorithm gives the different types of blocks. During the decoding phase, it is possible to extract the data that has been embedded, and image data can be reconstructed using the encryption key and the data-hiding key. Updating cipher texts by the data hiding key allows for integrating hidden data into raw images. The research results demonstrate a performance achieved with a comparatively high PSNR compared to some technologies considered state-of-the-art approaches.

In (11), the authors proposed a reversible data hiding for encrypted images (RDHEI), which reduces assumption inaccuracies into block-based blocks with usable encrypted images. On the side of the data owner, the original image is cut up into various dimension blocks, and block-based signal amplification is used to encrypt each block individually. Upon that data hider's side, predictions are generated with the help of a successful block-based predictor. The Huffman coding method is presented to minimize prediction mistakes inside the blocks of space that can be utilized to contain a great deal of additional data. On the receiving end, the additional data may be extracted entirely using an info concealment key, and image data can be retrieved without any quality loss using an image encryption key. Each of these keys is used to encrypt the data. Our research proves the inserting rate of our suggested scheme is much greater than that of other acts that are considered to be state-of-the-art.

In [12], the authors presented an approach for encrypting an image by first dividing the initial image into four sub-images with the help of sampling, then scrambling the sub-images by employing the two secret keys and the Arnold transformation, and eventually reconstructing the shattered sub-images to form the encoded image. Afterward, when additional data was inserted into the image by altering the difference in color in neighboring images that were

directly adjacent to one another, the receiver can represent a decrypted solution by utilizing the private keys in conjunction with a compressed image that contains additional data. In the meantime, with the assistance of the decoding and the data concealment key, the recipient can pick out the data that has been hidden and reconstruct the initial image without making any errors. The results demonstrated that the proposed process could obtain a higher payload while retaining acceptable image quality.

2. Methods

In this paper, we have constructed a new method that allows embedding several multi-layer data in encrypted images using a technology known as black-and-white pixel value embedding. The idea can be divided into three primary stages: 1) the content owner, 2) a data hider, and 3) a receiver. Figure 2 depicts the proposed approach, in which the content or data owner first sets aside a good accessible place on the main image and then converts the image into its encoded counterpart using the encryption key. Now, embedding data in the encrypted image is reversible intrinsically. All the data hider must do is fit new data into the previously vacated space once the space is emptied. In this novel approach, we adhere to the traditional assumption that redundant image content should be lossless compressed (for example, using superior RDH techniques), and then it needs to be encrypted to ensure its confidentiality.

A. Content owner:

First, the content owner reserves sufficient room on the original image. Next, the information is transformed into its encrypted version using the appropriate encryption key.

Encrypted Image Generation:

The phase includes self-reversible embedding, image encryption, and image partition. First, the top photo is cut in half to generate two halves labeled *A* and *B*. Next, *A*'s least significant bits are transiently inserted into B. This allows least significant bits to be used for trying to accommodate emails of *A*. Eventually, the rearranged encryption is done to produce the final image copy.

Image partition: Consider original image *I* is an 8-bit image that is gray-scale with its size $M \times N$ and pixels $I_{i,j} \in [0, 255]$, $1 \le i \le M$, $1 \le j \le N$. The owner of the contents splits it into multiple image rows with blocks. Each block contains m = [l/N] row, in which *l* seems to be the length of the messages that should be inserted, and the quantity of blocks is ($n = M\text{-}m + 1$). The very 1st flatness can be calculated with each block using the function (1).

$$f = \sum_{u=2}^{m}\sum_{v=2}^{N-1}\left|I_{u,v} - \frac{I_{u-1,v} + I_{u+1,v} + I_{u,v-1} + I_{u,v+1}}{4}\right| \quad (1)$$

The individual block containing the highest *f* value is designated as *A* and placed at the beginning of the image[16]. After that, the part of image *B* that didn't record is put together, as shown in Figure 3.

Self-Reversible Embedding: The fact that the capability of the hidden message will increase directly proportionate to the height of the peak point is not something that should come as a surprise to anyone. Since the histogram shift approach is preferred in terms of both the throughput and the grade of the steganography it creates, we use it to conserve space while integrating data. If this technique is used, it is possible to include the least relevant features (LSBs) of part *A* into the discrete points of the histogram of part *B*. This is achieved by moving the zero points, which increases available space. Pixels $I_{i,j} \in [0, 255]$ ($1 \le i \le M$, $1 \le j \le N$) in *B* are initially divided into 2 groups depending on the rows. These two groups are as follows: even-numbered rows whose row number I fulfills mod (*i*, 2) = 0 and odd-numbered rows whose row number I satisfies mod (*i*, 2) = 1.

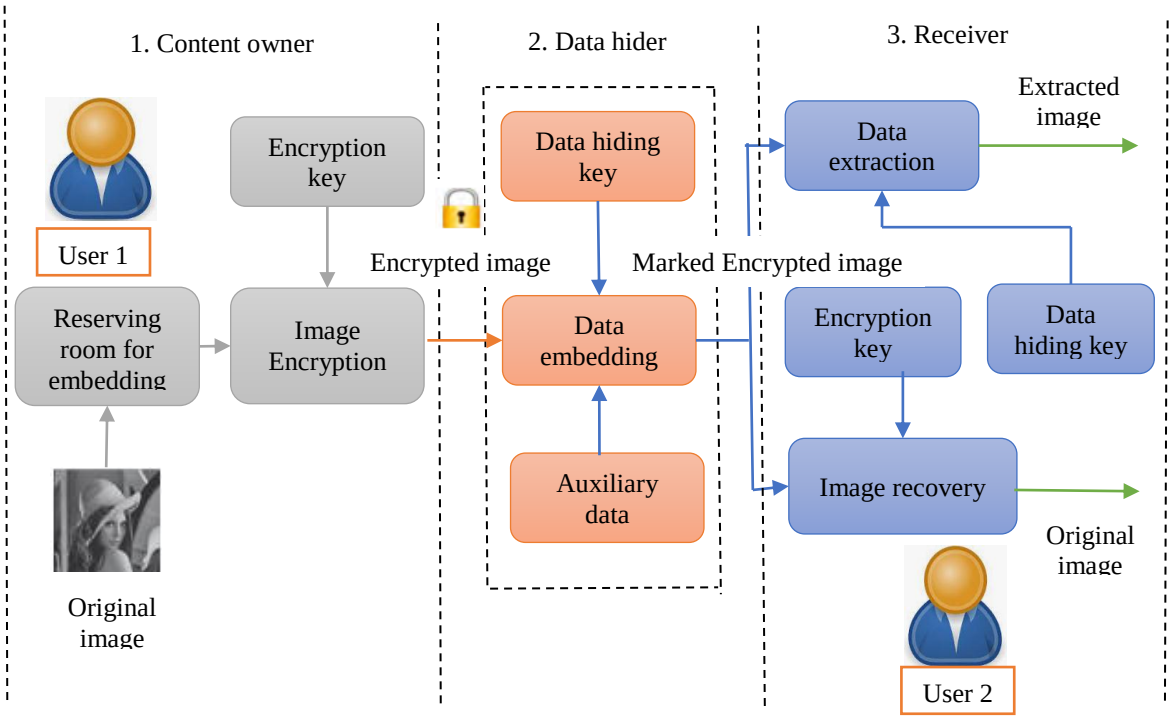


Figure 2: The framework of proposed scheme

In Figure 4a, the white rows depict the rows with even row numbers. Figure 4b indicates that the letter W is produced when all of the rows that have even values are joined together, but the letter *O* represents the rows that do not have even values and are the ones that are left behind [17]. Within the dashed box of *W*, Figure 4c depicts the adjusted schematic. The writing examples demonstrate the original pixels and the grey points indicate the interpolation pixels. We got this result by using the bicubic approximation formula to *W*.

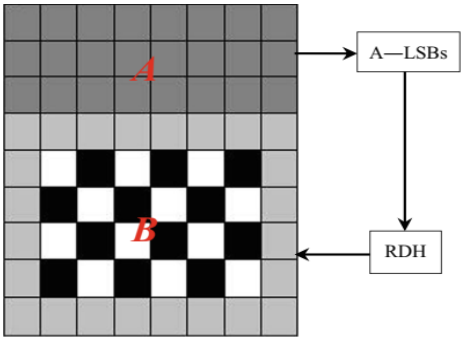


Figure 3: Illustration of image partition and embedding process

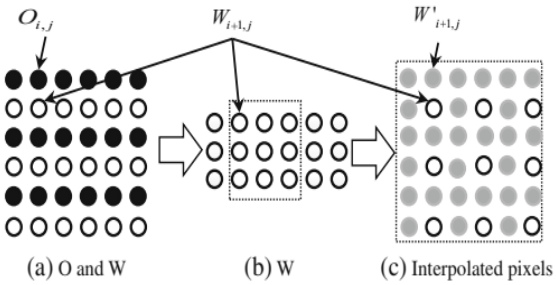


Figure 4: Sketch map of using the interpolated pixels as the predicted value

Due to pixel $O_{i,j}$ in O, by using interpolated pixels W' i + 1,j (in W) as the expected value of $O_{i,j}$, also the estimating error is found via

$$e_{i,j} = O_{i,j} - W'_i = 1, j \quad (2)$$

This obtain non-even-numbered rows of the estimating error matrix e of by executing for all pixels in O and:

$$e = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,N} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,N} \\ \vdots & \cdots & \cdots & \vdots \\ e_{[M/2],1} & e_{[M/2],2} & \cdots & e_{[M/2],} \end{bmatrix} \quad (3)$$

Vacancy can be reserved inside the error matrix using a partitioned local histogram shift. The loading is as follows:

Sourcing in every point for the peak value point, denoted as RP and *LP*, and searching for the 0 points in every part, represented as *LZ* and *RZ*, by dividing the histogram of assuming errors into the left part and right part. Then, the predicted error matrix is partitioned into *k* by *k* blocks. Each block requires looking for numbers equivalent to *LP*, *RP*, *LZ*, and *RZ*. After that, the same embedding process must embed a message in the maximum point points RP and LP, represented by the value "1."

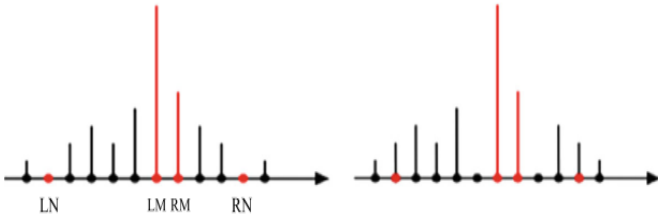


Figure 5 (a) All error values between RM + 1 and RN-1 shifted by adding 1 as well the result as shown in Figure 5 (b) is the error values between *LN* + 1 and *LM*-1 by subtracting 1

When there are no values such as this, the block does not embed any information and is instead indicated with a "0." Because once compared to the full estimating error sequence, which shifts all error values between RN-1 and RM + 1 by adding 1 and shifts error values among both LM-1 and LN + 1 by deducting 1, as shown in Figure 5, the distance between the highest points and 0 points within a block is lowed, which means that the number of changed pixels is reduced. Figure 5: Comparing the entire estimating error sequence to the partial estimating error sequence by adding each error between RN-1 and RM + 1, following the fulfillment of unfilled posts, some facts may be incorporated into the error matrix used for estimating. After that, we evaluate the estimating error of something like the pixels in the even-numbered rows with the assistance of the approximated pixels in the rows odd numbered that have some data encoded inside them. Following, the estimated error matrix consisting of even numbers rows is produced. This matrix can accommodate extra information if it is required.

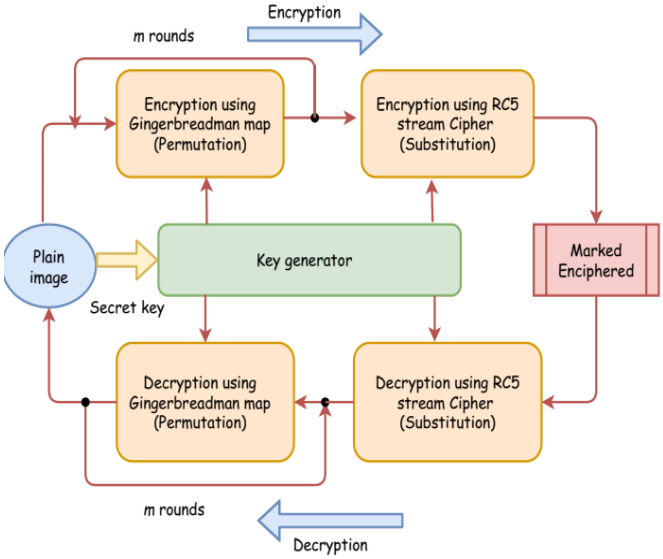


Figure 6: Encryption process

Image Encryption:

Figure 6 depicts the encryption system employed for the proposed architectural design. This scheme is divided into two main parts. The encryption and erratic key generation phases use RC5 data encryption. The chaotic public key stage is predicated on the Gingerbreadman map.

Gingerbreadman map:

For certain initial conditions and initial parameters, the map is chaotic. It resembles a gingerbread man. It is given by the piecewise linear transformation on plotting the set of chaotic solutions of this map.

$$a_{i+1} = 1 - b_i + |a_i|, i\epsilon N \cup \{0\} \quad (4)$$

$$b_{i+1} = a_i$$

where a_0 and b_0 are initial parameters

RC5 algorithm:

The RC5 cipher illustrates a symmetric block cipher, which means that the decryption and encryption procedures use the same cryptographic key. It is a parameterization technique, and its note is RC5-w/r/b, in which w presents the magnitude of the word, r for the no. of rounds, and b for the number of bytes enclosed in the personal key. The parameters, together with their values, are presented in Table 1. The RC5 algorithm can be broken into three distinct phases: the expansion of the key, the data's encryption, and the data's decryption; as part of the method for decoding, simple math operations like addition, subtraction, exclusive OR, and rotation were used.

Table 1: RC5-w/r/b parameters

Parameters	Description	Values
w	Word size bits	16,32, 64
r	Total no. of rounds	0,1,….255
b	No.of bytes in a secret key	0,1,….255

Following the generation of the rearranged soul image, which I indicate, we can encrypt to build the encrypted image, which IE shows. A content owner encrypts a self-embedded image in this section using a standard cipher (Rivest Chiper (RC5) and Gingerbreadman) with an encryption key. It's a type of encryption that encodes understandable text bit by bit, and all you have to do to use it is XOR the bits with the output bits of the pseudo-random bit generator. In addition, it is a symmetric encryption since the operation known as XOR, which stands for exclusive OR, is a symmetric operation. The prediction model comprises two parts: the permutation and the substitution.

$$P_w = odd[(e - 2) * 2^w] \quad (5)$$

$$Q_w = odd[(\Phi - 1) - 2^w] \quad (6)$$

In that, '*e*' is the Eulers, and 8 is the arum ratio (nearly 1.618). Table 2 displays the P and Q magic variables for the RC5 algorithm in their octal equivalents correspondingly. Figure 7 shows that the first growth phase comprises three parts: conversion, initialization, and mixing.

Table 2: Hexadecimal values of magic constants for RC5 key extension

Size of word	16-bit	32-bit	64-bit
P_w	0xB&E1	0xB3E15163	0xB7E151628AED2A6B
Q_w	0x9E37	0x9E3779B9	0x9E3779B97F4A7C15

During the 1st stage, the sender's confidential key *k*[0... *b*-1] is transferred to the array, which is new, of bytes, *L*[0... *c*-1], where *c* = [*b*/*u*] is the number of words /bytes. This is the beginning of the decryption process. All empty bytes of *L* is filled with zeros to conclude the procedure. The second stage is the adoption of the '*s*' array to a fixed order-less bit arrangement. This is done by selecting a mathematical sequence mod 2*w*, and it is doable for it to be confirmed by the two magic constants '*Pw*' and '*Qw*' defined for an arbitrary '*w*' parameter. The 3rd step entails blending the sender's private key with the subordinate array key in 3 independent passes. Addition and removal of words, marked by the symbol *C*=-, bit-wise XOR *C*, and right and left shift, signified by the notation >>>, are the three operations utilized in the RC5 cryptographic algorithms. During encrypting text, the plain text is entered as 2*w* bits. The letter *r* presents the iterative quantity of rounds, and the subkey is *S*[2*rC*2].

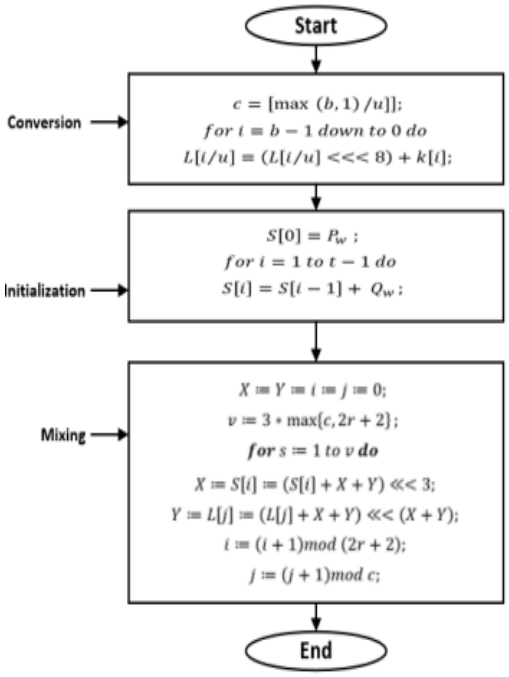


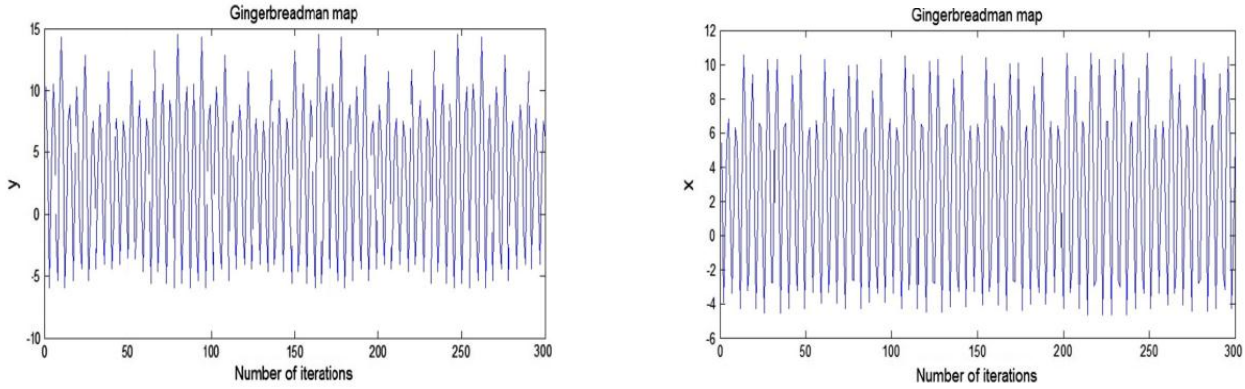
Figure 7: RC5 key generation phase.

The input block of data is given to two w-bit registers named *X* and *Y*, and the corresponding output is placed in registers *X* and *Y*. The RC5 encryption algorithm is given as follows:

Algorithm 1 Input (*X*, *Y*)
$X = X + S[0]$;
$Y = Y + S[1]$;
for *i* = 1 to *r* **do**
$X_{i+1} := \left((X_i \oplus Y_i) <<< Y_i\right) + [2i]$;
$Y_{i+1} := \left((Y_i \oplus X_i) <<< X_i\right) + [2i + 1]$;
$Outpt(X_{i+1}, Y_{i+1})$
The decryption process is a reverse of encryption and is given in Algorithm 2.

Algorithm 2 Input (X_{i+1}; Y_{i+1})
for *i* = *r* down to **do**
$Y_{i-1} = \left((Y_i - S[2i + 1) >>> X_i\right) \oplus X_i$;
$X_{i-1} = \left((X_i - S[2i]) >>> Y_i\right) \oplus Y_{i-1_i}$;
$X = X - S[0]$;
$Y = Y - S[1]$;
$Outpt(X, Y)$

The current RC5 key process is updated by including a 2D chaotic-based key distribution phase to raise the algorithm's sensitivity to randomness and increase its initial value. This is done to increase the algorithm's initial value. This disorderly map is divided into six hexagonal portions and illustrates random behavior in all currently filled sections. Figures 8(a) and 8(b) show that the remaining fixed hexagon sections have flow connecting their units. This chaotic map demonstrates random behavior in some regions while others remain stable.



(b) (a)

Figure 8: Gingerbreadman chaotic map (a) Randomness in *x* rection (b) Randomness in *y* rection

Data hider:

Following the initial phase, the encrypted copy of the original image that includes information is delivered to the data hider. Although the data hider does not know what the image or encryption key (Key1) contained initially, the data hider can insert more data into the encrypted image. To successfully embed hidden data into an encrypted image, the data hider must first collect 10-bits from the least significant bits of the 1st ten encoded pixels. And then, he/she knows the data about the inserted position. Later, the data hider encrypts the confidential message s by using the Eq. (7) and data hiding key K2 to create encrypted secret message sE, K2 is a binary series with equal length as *s*. In the end, the data hider only uses LSB re-placement to swap out the bits of part *A* that are accessible with the ones that contain the secret data s_E. The image that has been encrypted and holds the secret message sE is called IEM.

$$s_E = s \oplus K_2 \quad (7)$$

Receiver:

When the receiver receives such an image marked for encryption, the receiver can either recover the image using the available keys or extract the extra data from the tagged image. The receiver can recover the embedded data if he possesses the data-hiding key supplied by the data hider. Suppose the receiver has the encryption key that the content owner provided. In that case, they can retrieve the original image without losing quality.

Data Extraction and Image Recovery:

The data extraction in this module is in no way reliant on the decryption of images; hence, the order of these operations implies two different approaches to real-world applications, such as

1) Extracting Data from Encrypted Images:

An incompetent database manager may only have access to the info concealing key and be required to change data inside the encrypted set to maintain and update the personal details of photographs that have been encrypted to protect clients' privacy. When a request is made to the database management system to update the information in protected images, the database management first changes the information via LSB re-placement and then encodes the newly upgraded information using the info hiding a secret. After the database manager has obtained the key for the secret data, he will be able to decrypt and get the additional information by directly accessing the file's decrypted version. It keeps the original content to avoid leakage because the whole process is done in an encrypted environment from start to finish.

2) Extracting Data from Decrypted Images:

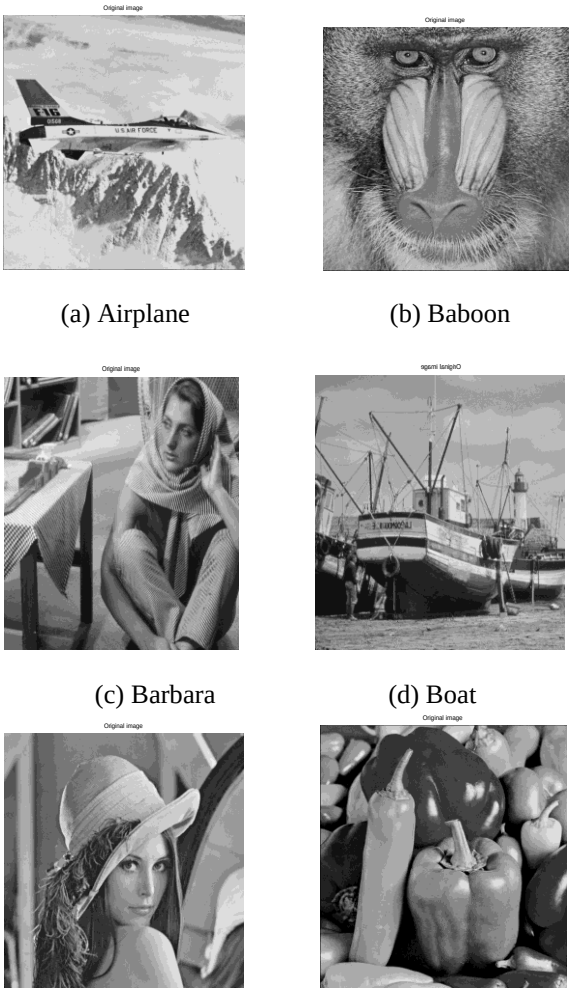
Intially, the user wishes to begin by decrypting the image, and then when the data is required, they will extract it from the image once it has been decrypted. The following example is an instance of how to implement such a scenario. Assume that user 1 uploaded her photographs to a remote server, and the server encrypts the images to keep the contents of the images secure. The cloud server embeds some notation into the encrypted images to handle the encrypted images. The such notation could include the identity of the owner of the images, the identity of the cloud server, or time stamps. Note that the cloud server does not have the authority to cause irreversible harm to the images. An authenticated user known as user 2 downloaded and decrypted the images. This user has access to the encryption key and the key to concealing the data. User 2 had high hopes of getting marked decrypted images, which have been decrypted but still contain the notation. These images may be utilized to track the origin of the data as well as its history. The sequence of decrypting the image before or without first extracting the data works well for this particular scenario. To be more explicit, the distortion is introduced by two different techniques: first embedding process, which modifies the LSB-planes of, and the self-reversible anchoring process, which embeds LSB planes of into. Both of these methods are part of the embedding. The first half of the distortion is well controlled by only utilizing the LSB

planes, while the second part can profit from the great performance of current RDH methods.

Data Extraction and Image Restoration:
Following the generation of the marked decrypted image within this module, the user content owner can extract the data and restore the original image. The hiding technique known as reversible hiding makes retrieving both the original host signal and the embedded message possible. Two crucial features must be fulfilled before reversible data concealing techniques may be utilized: the embedding capacity must be substantial, and distortion should be kept to a minimum. Both of these requirements are incompatible with one another. In most cases, a higher embedding capability will lead to greater distortion. A more efficient method can use less deformation to store the same amount of information or vice versa.

3. Results

In this part, we conduct experiments on various grayscale photographs that are considered standard to evaluate the effectiveness of the suggested technique. All the images are downloaded from the database [13]. Each testing set has a resolution of 512×512 pixels. The visual quality of the rebuilt image is evaluated by utilizing Peak Signal-to-Noise Ratio (PSNR), Bit Error Rates (BER), and payload. We worked on a personal computer with an Intel Core(TM) i5-4300U CPU operating at 1.9GHz and 4GB of RAM. The software used to implement all methods was MATLAB 2016a, whereas the operating system utilized was Windows Professional 64-bit.



(a) Airplane (b) Baboon

(c) Barbara (d) Boat

(e) Lena (f) Peppers

Figure 9: Original images.

Performance evaluation parameters:
In this section, we perform DHT evaluating metrics sing this study introduces the parameters.

Bit Error Rate (BER): This is the proportion of bits in the stego image IS with brightness values different from those of the cover object IC. The BER may be calculated using the equation (8) below.

$$Biterror\ rate = \frac{B_E}{B_C} \quad (8)$$

where B_C is the total number of bits in the grayscale cover image. And the total amount of bits different from the stego image is denoted by B_E.

$$B_E = \sum_{i=1}^{n} |I_{Cbin} - T_{Sbin}| \quad (9)$$

The BER can range anywhere from 0 to 1. where, the cover image's binary counterpart, I_{Cbin}, and the stego image's binary partner, I_{Sbin}, are denoted by the notation. The value 0 indicates that the stego image and the cover image have equal intensities, whereas the value 1 shows that the stego intensity differs from the cover image.

Peak Signal to Noise Ratio (PSNR): In most cases, the PSNR determines how good a stego image is in decibels (dB). The equation for PSNR is given by Equation (10), and it is essential to note that I_{Cmax} refers to the maximum pixel value of the cover image, while MSE refers to the average square error.

$$PSNR = 10log_{10}\left(\frac{I_{Cmax}^2}{MSE}\right) dB \quad (10)$$

$$MSE = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}(I_{Sxy} - I_{Cxy})^2 \quad (11)$$

The image coordinates are *x* and *y*, the dimensions of the image *are M* and *N*, the generated stego-image is I_{Sxy}, and the cover images I_{Cxy}.
Table 3 lists PSNR and Payload of proposed method and state of art methods in similar work.

Table 3: The comparison results of PSNR and Payload of proposed method and the methods in [13, 14]

Image name	**Method in [13]**		**Method in [14]**		**Our work**	
	PSNR in (dB)	Payload in (bpp)	PSNR in (dB)	Payload in (bpp)	PSNR in (dB)	Payload in (bpp)
Airplane	38.33	0.001	39.84	0.500	54.96	0.55
Baboon	38.37	0.004	39.84	0.500	54.89	0.54
Barbara	37.97	0.006	39.85	0.500	59.91	0.51
Boat	37.93	0.001	39.81	0.500	55.02	0.50
Lena	38.08	0.005	39.38	0.500	55.10	0.51
Peppers	38.05	0.005	39.83	0.500	54.99	0.53

Figure 10 shows the performance comparison of the payload for the proposed work and state-of-the-art methods. Evidently, for all six images, the proposed work outperforms in the case of payload as related to other methods in [13] and methods in [14].

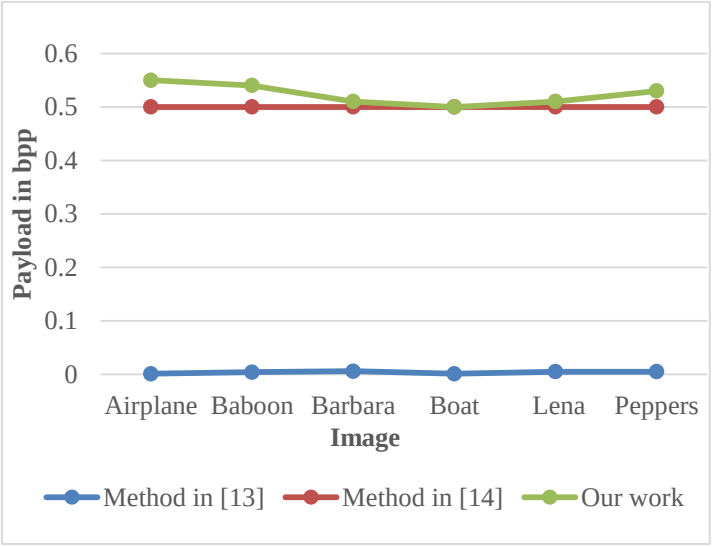


Figure 10: Performance comparison of Payload for proposed work and state-of-the-art methods

Figure 11 shows the performance comparison of PSNR for the proposed work and state-of-the-art methods. It is evident that for all six images, the proposed work outperforms in the case of PSNR as related to other methods in [13] and methods in [14]..

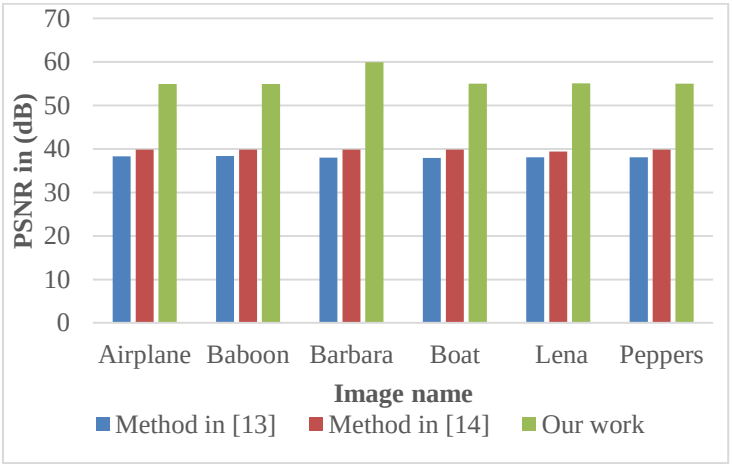


Figure 11: Performance comparison of PSNR for proposed work and state-of-the-art methods

Performance analysis on embedding rate and PSNR:

In this the proposed method again allows for encoding three bits. One of the benefits of using one-layer histogram shifting is that it makes it possible to reach a larger capacity. When two layers of embedding are employed, there is a significant drop in the image's fidelity. Figures 12 and 13 compare the different anchoring rates regarding their effects on efficiency, using Lena and boats as examples. When the embedding capacity is more than the maximum capability that can be achieved using a single-layer embedding strategy, it is clear that the suggested method can improve performance.

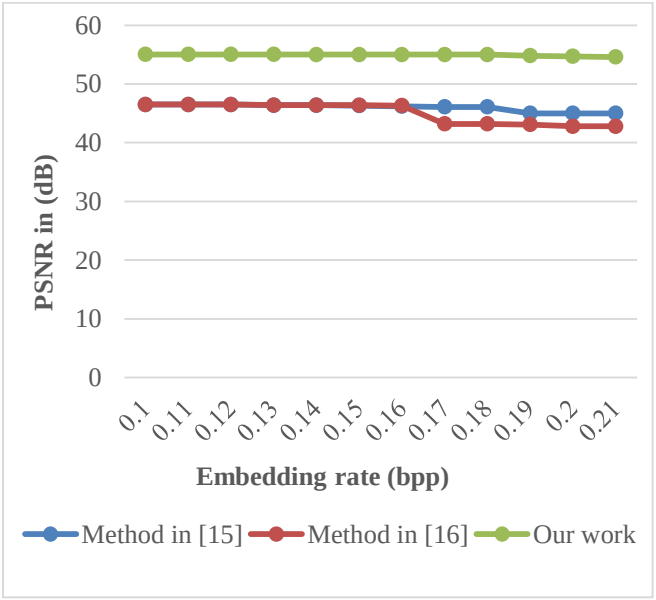


Figure 12: The performance comparison of different embedding rates for Boats image

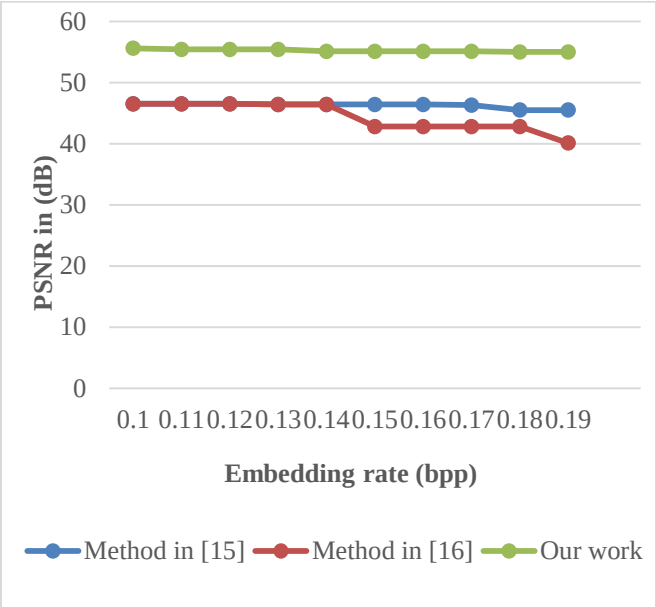


Figure 13: The performance comparison of different embedding rates for Lena image

4. Conclusion

This paper provides an algorithm for reversibly embedding secret data in encrypted images. The authors of this study created the algorithm. Even though he is unaware of the content of the original image, the data hider is enabled to include the private information into the encrypted image through 2D histogram modification thanks to the retained correlation. The system protects the secrecy of content since the embedding procedure is carried out on data that has been encrypted. Image archiving in cloud computing, an area where high image resolution and reproducibility are highly sought, is one of the possible applications of this technology. In this study, we develop a new way to hide data in an encoded image, making it easier to reverse with highest PSNR and Payload values.

5. References

Golovkova, Elena & Kulygin, Valentin & Pozdeev, Evgeniy. (2021). OPPORTUNITIES FOR USING CLOUD SERVERS IN EDUCATION. Modern Technologies and Scientific and Technological Progress. 1. 117-118. 10.36629/2686-9896-2021-1-1-117-118.

Dwivedi, Anirudh. (2019). Fog Computing Application for Cloud Servers. International Journal for Research in Applied Science and Engineering Technology. 7. 3146-3151. 10.22214/ijraset.2019.5520.

Xiang, S.-J & Luo, X.-R. (2016). Reversible data hiding in encrypted image based on homomorphic public key cryptosystem. 27. 1592-1601. 10.13328/j.cnki.jos.005007.

Waseem, Mohammad & Lakhan, Abdullah & Jamali, Irfan Ali. (2016). Data Security of Mobile Cloud Computing on Cloud Server. OALib. 03. 1-11. 10.4236/oalib.1102377.

Singh, Surendra. (2016). Cloud Computing: Comparative Study Own Server vs Cloud Server. 590-598. 10.1142/9789814704830_0056.

Xia, Ping. (2020). Data Security Risk and Preventive Measures of Virtual Cloud Server Based on Cloud Computing. 10.1007/978-3-030-43306-2_6.

Priya, A.Anusha & R., Gunasundari. (2017). Securing Data on the Cloud Server by the User Authentication and Data Security Techniques. International Journal of Computer Applications. 165. 6-12. 10.5120/ijca2017913844.

Tang, Zhenjun & Chunqiang, Mingyuan & Guijin, Yu & Zhang, Fan. (2021). Reversible data hiding for encrypted image based on adaptive prediction error coding. IET Image Processing. 2021. 1-13. 10.1049/ipr2.12252.

Wu, Xiaoshuai & Qiao, Tong & Xu, Ming & Zheng, Ning. (2021). Secure reversible data hiding in encrypted images based on adaptive prediction-error labeling. Signal Processing. 188. 108200. 10.1016/j.sigpro.2021.108200.

Chang, Qi. (2020). Reversible Data Hiding in Encrypted Images Based on Hybrid Cryptosystem. IETE Technical Review. 38. 1-7. 10.1080/02564602.2020.1757521.

Wang, Xu & Li, Li-Yao & Chang, Ching-Chun & Chen, Chih-Cheng. (2021). High-Capacity Reversible Data Hiding in Encrypted Images Based on Prediction Error Compression and

Block Selection. Security and Communication Networks. 2021. 1-12. 10.1155/2021/9606116.

Wu, Dan. (2018). Reversible Data Hiding for Encrypted Image Based on Arnold Transformation. MATEC Web of Conferences. 173. 03088. 10.1051/matecconf/201817303088.Cusack,

Kim HJ, Sachnev V, Shi YQ, Nam J, Choo HG (2008) A novel difference expansion transform for reversible data embedding. IEEE Trans Inf Forensics Secur 3(3):456–465

D. W. Xu, K. Chen, R. D. Wang, and S. B. Su, "Completely separable reversible data hiding in encrypted images," in International Workshop on Digital-forensics and Watermarking (IWDW 2015), Tokyo, Japan, vol. 9569 of LNCS, pp. 365–377, 2016.

M. Li, D. Xiao, Y. Zhang, and H. Nan, "Reversible data hiding in encrypted images using cross division and additive homomorphism," Signal Processing: Image Communication, vol. 39, pp. 234–248, 2015.

Narasimha, V., Dhanalakshmi, M. (2022). Severity and Risk Predictions of Diabetes on COVID-19 Using Machine Learning Techniques. In: Garcia Diaz, V., Rincón Aponte, G.J. (eds) Confidential Computing. Advanced Technologies and Societal Change. Springer, Singapore. https://doi.org/10.1007/978-981-19-3045-4_21.

Narasimha, Vadthe, and M. Dhanalakshmi. "Detection and Severity Identification of Covid-19 in Chest X-ray Images Using Deep Learning." *International Journal of Electrical and Electronics Research* 10 (2022).