

V.M. MANIKANDAN 

## REVERSIBLE DATA-HIDING-THROUGH-ENCRYPTION SCHEME USING ROTATED STREAM CIPHER

**Abstract** *Research into the domain of reversible data-hiding has received a great deal of attention in recent years due to its wide applications in medical image transmission and cloud computing. Reversible data-hiding during image encryption is a recently emerged framework for hiding secret data in an image during the image-encryption process. In this manuscript, we propose a new reversible data-hiding-through-encryption scheme that will ensure a high embedding rate without bringing any additional overhead of key handling. The proposed algorithm can use any secure symmetric encryption scheme, and the encryption and/or decryption key should be shared with the receiver for data extraction and image recovery. As per the proposed scheme, the data hider can hide three-bits of a secret message in an image block of a size of  $B \times B$  pixels. The data extraction and image recovery will be carried out by analyzing the closeness between adjacent pixels. The simulation of the new scheme carried out on the USC-SIPI dataset shows that the proposed scheme outperforms the well-known existing schemes in terms of embedding rates and bit error rates.*

**Keywords** reversible data-hiding, image encryption, image decryption, secure message transmission

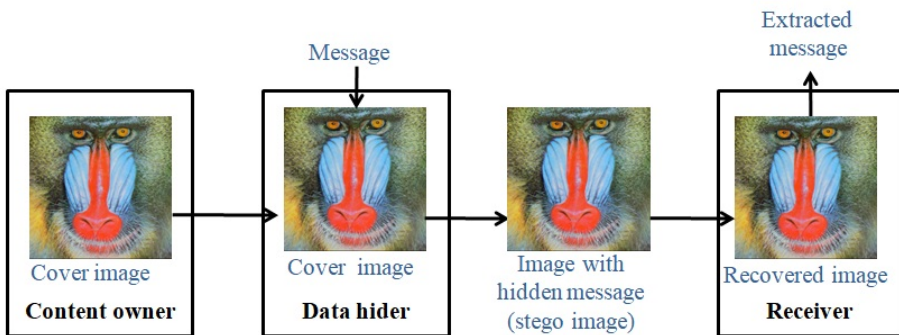
**Citation** Computer Science 22(2) 2021: 267–291

**Copyright** © 2021 Author(s). This is an open access publication, which can be used, distributed and reproduced in any medium according to the Creative Commons CC-BY 4.0 License.

## 1. Introduction

Data-hiding techniques have been widely used for secure message transmission over the last three decades [2, 6]. Reversible data-hiding is a recently explored area in the field of data-hiding [3]. In a reversible data-hiding scheme, a sender can embed some secret message on a cover medium, and a receiver can extract the secret message along with the recovery of the original cover medium. Digital images are widely used as a cover medium; hence, reversible data-hiding on images is one of the widely explored domains [18]. There are numerous applications for reversible data-hiding; one of the main uses is in medical image transmission, where the sender can embed patient details or diagnosis results in the medical image itself instead of sending it as a separate text file [17]. Reversible data-hiding schemes can also be used by cloud service providers to embed metadata on contents uploaded to cloud storage by cloud users [25]. Conventional data-hiding schemes cannot be used by cloud storage service providers, as all of these schemes permanently modify the original image pixels and no cloud service providers have permission to do any such kinds of modifications on its received data.

In general, three different roles will be played by the people in the reversible data-hiding framework: the *content owner*, *data hider*, and *receiver*. The *content owner* is the person who has the original image. The various roles in a reversible data-hiding framework are graphically shown in Figure 1.



**Figure 1.** Various roles in reversible data-hiding framework

In medical image transmission, a doctor who treats a patient and holds his/her medical images can be considered to be a content owner; in a cloud environment, users who want to upload images to the cloud can be treated as content owners. The *data hider* is responsible for embedding the desired data into the image. In the medical image-transmission environment, the doctor or his/her staff will act as the data hider. The cloud service provider may play the role of data hider in a cloud environment while embedding metadata on the images received from a user. The *receiver* is responsible for extracting the secret message from the stego image (an image that contains a secret

message) and recovering the original image. A specialist doctor at the receiving end may act as a receiver during medical image transmission. In a cloud environment, the cloud service provider itself may do the metadata extraction from the image, and the recovered image may be given back to the user without any damage. In reversible data-hiding through an encryption scheme, the key assumption is that the same person will play the role of both content owner and data hider.

The existing reversible data-hiding schemes can be broadly classified into three categories:

- Reversible data-hiding in natural images: In this, original image pixels will be updated by the data hider to embed secret message bits. Three main approaches are widely explored for doing reversible data-hiding on natural images: histogram shifting [15], difference expansion [9, 21], and lossless compression [3]. Later, histogram shifting and reversible data-hiding schemes are improved by exploring prediction error histograms and prediction-error expansion [4, 7, 16, 20].
- Reversible data-hiding in encrypted images: In this, the data hider will receive an encrypted image, and a secret message will be embedded on the image. The first scheme in this category was introduced in 2011 by Xinpeng Zhang [26]. In 2012, the same scheme was improved by bringing some side-match techniques to reduce the bit error rate during image recovery [8]. Later, a few more reversible data-hiding schemes were introduced for hiding data in encrypted images [23, 24, 27].
- Reversible data-hiding through encryption: This is a recently introduced technique for reversible data-hiding. The key idea of this scheme is that the reversible data-hiding will be carried out during the image-encryption phase itself. The reversible data-hiding schemes in this category are discussed in [12–14].

In this paper, we propose a new reversible data-hiding-through-encryption scheme that uses stream ciphers with matrix rotation and a sequence of predefined bit-wise XOR operations for image-encryption purposes. For a better understanding of the proposed algorithms, an overview of the existing reversible data-hiding-through-encryption schemes discussed in [12, 14] is shown in Figure 2.

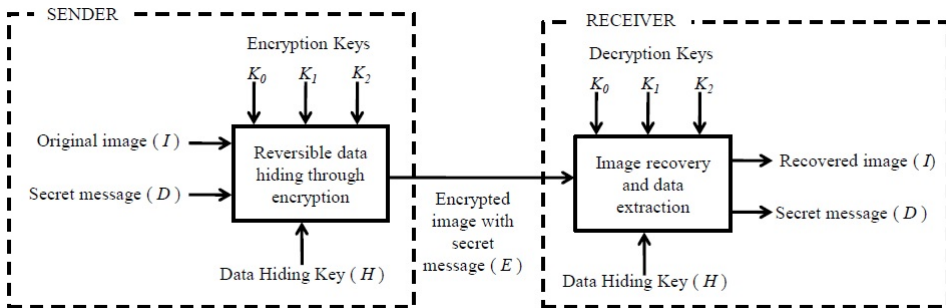


Figure 2. Overview of existing reversible data-hiding through encryption

In the existing reversible data-hiding-through-encryption schemes, the sender and receiver use three different encryption and/or decryption keys ( $K_0, K_1$ , and  $K_2$ ). The sender will initiate a block-wise image-encryption process, and the selection of the pseudo-random bytes for encrypting a selected block will be determined by the secret message bit that the sender wants to embed into it. A trained support vector machine model (SVM) is used in [12], and an entropy-based measure is used in [14] for the data-extraction/image-recovery process.

As mentioned earlier, we propose a new reversible data-hiding-through-encryption scheme that uses a rotated stream cipher for encryption purposes. The key properties of the proposed scheme are listed below:

- The sender and receiver should agree upon a single key; say,  $K$ . This reduces the overhead of the key-sharing process, which is a major concern in the existing reversible data-hiding-through-encryption scheme.
- Eight different stream ciphers are generated in the proposed scheme through a sequence of matrix rotation operations and a set of predefined bit-wise operations.
- By considering the adjacent pixels, a new similarity measure is introduced in this paper for data extraction and image recovery in order to reduce bit error rates.
- Since the proposed scheme uses a statistically computed measure, it does not require any training process to generate a trained model. Note that the schemes reported in [12,13] used a trained support vector machine (SVM) model for data extraction and image recovery.
- The sender can embed three bits in an image block of a size of  $B \times B$  pixels. So, the block size used in the proposed scheme will decide the embedding rate, and the embedding rate will be  $\frac{3}{B^2}$  bits per pixel.

The remaining section of this paper is organized as follows: in Section 2, we discuss the proposed reversible data-hiding process and the data-extraction/image-recovery process. An illustrative example is given in Section 3, and in Section 4, we detailed the experimental study and result analysis. In Section 5, we conclude the paper with some future directions for work in this area.

## 2. Proposed scheme

The proposed reversible data-hiding-through-encryption scheme and data-extraction/image-recovery process are detailed in this section. Algorithm 1 describe the steps for reversible data-hiding during the image-encryption process. Similarly, Algorithm 2 describes the steps during the data-extraction/image-recovery process. Algorithm 2 uses a closeness measure (computed using Algorithm 3) for data extraction and image recovery.

Note that Algorithm 1 considers the original image ( $I$ ) with a size of  $R \times C$  pixels, the secret message bit sequence ( $S$ ), and the encryption key ( $K$ ) as inputs. The encryption key ( $K$ ) will be used, along with a stream-cipher generator (we used

an RC4 stream-cipher generator [10]; the scheme permits the use of any secure stream generators for encryption to generate a pseudo-random matrix ( $U$ ) of size  $R \times C$ . From  $U$  through a sequence of rotations and a sequence of bit-wise operations, the sender will generate eight different pseudo-random matrices (say,  $U_0, U_1, U_2, U_3, U_4, U_5, U_6$ , and  $U_7$ ). In addition to this, the original image and pseudo-random matrices will be divided into non-overlapping blocks of a size of  $B \times B$  pixels. The key idea behind the proposed scheme is that, to encrypt the  $X^{th}$  block from the original image ( $I_X$ ), we will consider any one of the  $X^{th}$  blocks from  $U_0, U_1, U_2, U_3, U_4, U_5, U_6$ , and  $U_7$  based on the three-bit secret message bits that the sender wishes to embed. Readers may note that eight possible combinations need to be considered while embedding three bits from the secret message; these are 000, 001, 010, 011, 100, 101, 110, and 111. The same process will be carried out on all the blocks of the image to obtain the final encrypted image with hidden secret message bits ( $E$ ).

---

**Algorithm 1:** Proposed reversible data-hiding-through-encryption scheme using rotated stream cipher

---

**Input:** Original image  $I$  having size of  $R \times C$  pixels, secret message bit sequence  $S$ , and encryption key  $K$

**Output:** Encrypted image  $E$  with hidden secret message  $S$

- 1 Generate pseudo-random sequence of bytes (stream cipher)  $U$  of size  $R \times C$  using stream-cipher generator by providing encryption key  $K$  // RC4 stream-cipher generator is used in proposed scheme
  - 2 Find pseudo-random byte matrices  $U_1, U_2, U_3, U_4$  in following way:
  - 3  $U_0 = U$
  - 4  $U_1$  is rotated version of  $U_0$  in 90-degree counterclockwise direction;
  - 5  $U_2$  is rotated version of  $U_1$  in 90-degree counterclockwise direction;
  - 6  $U_3$  is rotated version of  $U_2$  in 90-degree counterclockwise direction.
  - 7 Compute  $U_4$  by performing bit-wise XOR operation between  $U_0$  and  $U_1$ .
  - 8 Compute  $U_5$  by performing bit-wise XOR operation between  $U_1$  and  $U_2$ .
  - 9 Compute  $U_6$  by performing bit-wise XOR operation between  $U_2$  and  $U_3$ .
  - 10 Compute  $U_7$  by performing bit-wise XOR operation between  $U_3$  and  $U_4$ .
  - 11 Divide original image  $I$  into non-overlapping blocks of size in such way that blocks will have size of  $B \times B$  pixels.
  - 12 Initialize matrix of size  $R \times C$  to keep encrypted image.
  - 13 Also divide pseudo-random byte matrices  $U_0, U_1, U_2, U_3, U_4, U_5, U_6, U_7$  into non-overlapping blocks of size  $B \times B$ .
  - 14 Access  $X^{th}$  block from original image  $I_X$  and three consecutive bits from secret message  $S$  (say,  $S_P, S_Q, S_R$ ) that we want to hide next in image. //  $X$  can be identified using data-hiding key that will help provide better security. Here, we accessed blocks in row-wise linear order.
  - 15 Find integer  $T$  that corresponds to three-bit sequence  $S_P, S_Q, S_R$ .
  - 16 Now, use  $X^{th}$  block from  $U_T$  to encrypt  $I_X$  through bit-wise XOR operation. Keep encrypted block as  $X^{th}$  block in  $E$ .
  - 17 Repeat Steps 14 through 16 for encrypting all blocks in image  $I$ .
  - 18 Return encrypted image  $E$ .
-

---

**Algorithm 2:** Data extraction and image recovery using new closeness measure between adjacent pixels

---

**Input:** Encrypted image  $E$  that consists of  $R \times C$  pixels generated by following steps in Algorithm 1 and the decryption key  $K$

**Output:** Recovered image  $I$  and bits extracted from image  $D$

---

- 1 Generate pseudo-random sequence of bytes (stream cipher)  $U$  of size  $R \times C$  using stream-cipher generator by providing decryption key  $K$  // RC4 stream-cipher generator is used in proposed scheme; any other stream-cipher generator can be used
  - 2 Find pseudo-random byte matrices  $U_1, U_2, U_3, U_4$  in following way:
  - 3  $U_0 = U$
  - 4  $U_1$  is rotated version of  $U_0$  in 90-degree counterclockwise direction;
  - 5  $U_2$  is rotated version of  $U_1$  in 90-degree counterclockwise direction;
  - 6  $U_3$  is rotated version of  $U_2$  in 90-degree counterclockwise direction.
  - 7 Compute  $U_4$  by performing bit-wise XOR operation between  $U_0$  and  $U_1$ .
  - 8 Compute  $U_5$  by performing bit-wise XOR operation between  $U_1$  and  $U_2$ .
  - 9 Compute  $U_6$  by performing bit-wise XOR operation between  $U_2$  and  $U_3$ .
  - 10 Compute  $U_7$  by performing bit-wise XOR operation between  $U_3$  and  $U_4$ .
  - 11 Divide encrypted image  $E$  into non-overlapping blocks of size  $B \times B$  pixels.
  - 12 Initialize matrix  $I$  with size of  $R \times C$  to keep recovered image.
  - 13 Initialize an empty list  $D$  to keep extracted secret message bits.
  - 14 Also divide pseudo-random byte matrices  $U_0, U_1, U_2, U_3, U_4, U_5, U_6, U_7$  into non-overlapping blocks of size  $B \times B$ .
  - 15 Access  $X^{th}$  block from encrypted image  $E_X$  and find eight different versions of decrypted image block  $E_X$  in following way (Steps 16 through 23):
  - 16  $V_0 = E_X \oplus U_{0X}$
  - 17  $V_1 = E_X \oplus U_{1X}$
  - 18  $V_2 = E_X \oplus U_{2X}$
  - 19  $V_3 = E_X \oplus U_{3X}$
  - 20  $V_4 = E_X \oplus U_{4X}$
  - 21  $V_5 = E_X \oplus U_{5X}$
  - 22  $V_6 = E_X \oplus U_{6X}$
  - 23  $V_7 = E_X \oplus U_{7X}$
  - 24 Compute closeness measure  $M_P$  by calling Algorithm 3 from each version of decrypted image block  $V_P$  where  $P \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ .
  - 25 Find minimum value from  $\{M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7\}$  (say,  $M_Q$ ).
  - 26 Convert  $Q$  into corresponding three-bit binary sequence ( $S_P, S_Q, S_R$ ) and concatenate with  $D$ .
  - 27 Correctly recovered image block is  $V_Q$ ; keep it as  $X^{th}$  block of recovered image  $I$ .
  - 28 Repeat process (Steps 15 through 27) to extract all hidden bits and recover original image.
  - 29 Return recovered image  $I$  and extracted bit sequence  $D$ .
-

**Algorithm 3:** Closeness measure computation**Input:** Image block  $L$  with size of  $B \times B$  pixels**Output:** Closeness measure  $M$  computed from  $L$ 


---

```

1  $x = 1$ 
2  $y = 1$ 
3  $M = 0$ 
4 while  $x \leq B$  do
5   while  $y \leq (B - 1)$  do
6      $M = M + |L_{x,y} - L_{x,y+1}|$ 
7      $y = y + 1$ 
8    $x = x + 1$ 
9 Return closeness measure  $M$ .
```

---

At the receiver side, the data extraction and image recovery will be carried out by using a measure that is computed by considering the closeness between the adjacent pixels by accessing the pixels in a row-wise linear order. The receiver receives an encrypted image ( $E$ ) with a hidden message (the output of Algorithm 1). The receiver already knows the decryption key ( $K$ ). Now, the receiver generates a pseudo-random matrix ( $U$ ) that has a size of  $R \times C$ . Note that the received encrypted image also has a size of  $R \times C$  pixels.

Let us assume that  $U_0$  is a matrix that consists of the same values from  $U$ . Further, three different matrices, say  $U_1$ ,  $U_2$ , and  $U_3$  will be generated through a sequence of the counter clock-wise rotation operation in 90 degrees. Further, through a pre-defined pair-wise bit-wise XOR operation four more matrices will be generated, say  $U_4$ ,  $U_5$ ,  $U_6$ , and  $U_7$ . Then, both the original image and the pseudo-random matrices will be divided into non-overlapping blocks of size  $B \times B$ . The blocks of the original image will be accessed in the same order that we used during the data-hiding process. To decide on the block positions, both the sender and receiver can use a data-hiding key. The usage of the data-hiding key will also improve the security of the proposed scheme.

To extract the data from the  $X^{th}$  block of  $E$  (say,  $E_X$ ), we must consider the  $X^{th}$  blocks from  $U_0, U_1, U_2, U_3, U_4, U_5, U_6$ , and  $U_7$  and do the bit-wise XOR operations to generate eight different versions of the image blocks ( $V_0, V_1, V_2, V_3, V_4, V_5, V_6$ , and  $V_7$ ). One of the blocks from  $\{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7\}$  will be the correctly recovered image block. To identify the recovered image block, a new measure is introduced in this manuscript that will consider the absolute difference between adjacent pixel pairs while accessing in a row-wise linear order. The computation of the closeness measure is described in Algorithm 3.

The pre-assumption is that the pixels in the correctly recovered image block will be highly correlated as compared to the other blocks. Now, if  $V_Q$  (where  $V_Q \in \{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7\}$ ) is the image block with the lowest measure (as per Algorithm 3), then it will be considered to be the recovered image block, and the three-bit binary that corresponds to  $Q$  will be the extracted secret message from

the  $K^{th}$  block. This process will be continued for all of the blocks in the image ( $I$ ) to generate the final encrypted image ( $E$ ). If an original image block consists of the same pixel values, the closeness measure will be 0.

### 3. Illustrative example

In Section 2, the proposed algorithms are detailed. For a better understanding of the proposed scheme, an illustrative example is discussed in this section.

#### 3.1. Proposed reversible data-hiding through encryption

Let us assume that we are considering an original image  $I$  with a size of  $8 \times 8$  pixels and a block size of  $4 \times 4$  pixels that is used during the reversible data-hiding. Note that the image and block sizes that are considered during simulation are different from the details mentioned here. Figure 3 shows a sample original image as well as the four non-overlapping blocks of a size of  $4 \times 4$  pixels.

156	157	158	159	158	157	155	155
157	157	158	158	158	158	156	156
159	158	156	155	156	157	158	158
159	158	156	155	156	157	158	158
157	157	158	158	158	157	155	155
157	157	158	158	158	157	155	155
159	158	156	155	157	157	157	157
160	158	154	153	156	158	158	158

Figure 3. Sample original image

The next step is to generate a pseudo-random matrix ( $U$ ) of size  $8 \times 8$  by providing an encryption key to a stream generator (like RC4). Let us assume that the pseudo-random matrices that are generated by using encryption key  $K$  are given in Figure 4.

139	35	207	45	27	190	73	46
71	147	44	96	98	161	218	61
109	228	209	2	10	149	249	12
216	54	70	65	228	6	226	129
2	48	111	203	251	54	92	96
32	28	240	4	16	139	153	152
172	57	209	153	228	197	91	161
211	250	86	154	148	64	87	37

Figure 4. Pseudo-random matrix generated using encryption key  $K$

Then, the eight pseudo-random matrices ( $U_0, U_1, U_2, U_3, U_4, U_5, U_6,$  and  $U_7$ ) are generated from  $U$  based on the sequence of rotations (in a counterclockwise direction) and through bit-wise XOR the operations between the pairs of matrices. The matrices that we will get through this operation are shown in Figure 5.



139	35	207	45	27	190	73	46
71	147	44	96	98	161	218	61
109	228	209	2	10	149	249	12
216	54	70	65	228	6	226	129
2	48	111	203	251	54	92	96
32	28	240	4	16	139	153	152
172	57	209	153	228	197	91	161
211	250	86	154	148	64	87	37

$U_0$

46	61	12	129	96	152	161	37
73	218	249	226	92	153	91	87
190	161	149	6	54	139	197	64
27	98	10	228	251	16	228	148
45	96	2	65	203	4	153	154
207	44	209	70	111	240	209	86
35	147	228	54	48	28	57	250
139	71	109	216	2	32	172	211

$U_1$

37	87	64	148	154	86	250	211
161	91	197	228	153	209	57	172
152	153	139	16	4	240	28	32
96	92	54	251	203	111	48	2
129	226	6	228	65	70	54	216
12	249	149	10	2	209	228	109
61	218	161	98	96	44	147	71
46	73	190	27	45	207	35	139

$U_2$

211	172	32	2	216	109	71	139
250	57	28	48	54	228	147	35
86	209	240	111	70	209	44	207
154	153	4	203	65	2	96	45
148	228	16	251	228	10	98	27
64	197	139	54	6	149	161	190
87	91	153	92	226	249	218	73
37	161	152	96	129	12	61	46

$U_3$

165	30	195	172	123	38	232	11
14	73	213	130	62	56	129	106
211	69	68	4	60	30	60	76
195	84	76	165	31	22	6	21
47	80	109	138	48	50	197	250
239	48	33	66	127	123	72	206
143	170	53	175	212	217	98	91
88	189	59	66	150	96	251	246

$U_4$

11	106	76	21	250	206	91	246
232	129	60	6	197	72	98	251
38	56	30	22	50	123	217	96
123	62	60	31	48	127	212	150
172	130	4	165	138	66	175	66
195	213	68	76	109	33	53	59
30	73	69	84	80	48	170	189
165	14	211	195	47	239	143	88

$U_5$

246	251	96	150	66	59	189	88
91	98	217	212	175	53	170	143
206	72	123	127	66	33	48	239
250	197	50	48	138	109	80	47
21	6	22	31	165	76	84	195
76	60	30	60	4	68	69	211
106	129	56	62	130	213	73	14
11	232	38	123	172	195	30	165

$U_6$

118	178	227	174	163	75	175	128
244	112	201	178	8	220	18	73
133	148	180	107	122	207	16	131
89	205	72	110	94	20	102	56
187	180	125	113	212	56	167	225
175	245	170	116	121	238	233	112
216	241	172	243	54	32	184	18
125	28	163	34	23	108	198	216

$U_7$

**Figure 5.** Pseudo-random matrices  $U_0, U_1, U_2, U_3, U_4, U_5, U_6, U_7$  generated using matrix rotation and predefined pair-wise bit-wise bit-XOR operation

Note that, in this example, we have considered the original image with a size of  $8 \times 8$  pixels and a block-size of  $4 \times 4$  pixels. So, in the original image, there will be four image blocks, and in each image block, we can embed three bits of secret data. Since there are 4 image blocks in the original image, we can embed 12 secret message bits in this image. Let us assume that, in the original image, the blocks are numbered 0, 1, 2, and 3 (row-wise linear order) and the secret message bit sequence is  $D = (1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0)$ . The following three-bit combinations will be embedded in each of the blocks.

- block 0: 1, 1, 0,
- block 1: 0, 1, 1,
- block 2: 0, 1, 1,
- block 3: 0, 1, 0.

The blocks and their corresponding numbers in the original image are shown in Figure 6.

156	157	158	159	158	157	155	155
157	158	158	158	158	157	155	156
157	158	155	155	156	157	158	158
157	157	158	158	158	157	155	155
157	158	158	155	155	157	157	157
160	158	154	153	156	158	158	158

**Figure 6.** Block numbers

During the proposed reversible data-hiding through the encryption process, the following sequence steps will be followed:

- Step 1: Block 0 of original will be encrypted using  $0^{th}$  block from  $U_6$ , as three-bit data bit sequence that we need to embed is  $(110)_2 = (6)_{10}$ .
- Step 2: Block 1 of original will be encrypted using  $1^{st}$  block from  $U_3$ , as three-bit data bit sequence that we need to embed is  $(011)_2 = (3)_{10}$ .
- Step 3: Block 2 of original will be encrypted using  $3^{rd}$  block from  $U_3$ , as three-bit data bit sequence that we need to embed is  $(011)_2 = (3)_{10}$ .
- Step 4: Block 3 of original will be encrypted using  $3^{rd}$  block from  $U_2$ , as three-bit data bit sequence that we need to embed is  $(010)_2 = (2)_{10}$ .

The final encrypted image that is obtained after the block-wise image-encryption process is shown in Figure 7.

The randomness of the pixel values in the encrypted image will be purely dependent on the pseudo-random generator that we use, and the proposed scheme is compatible with any pseudo-random generator.

106	102	254	9	70	240	220	16
198	255	71	74	168	122	15	191
81	214	231	228	218	76	178	81
101	91	174	171	221	159	254	179
9	121	142	101	223	219	173	67
221	88	21	168	156	76	127	246
200	197	5	199	253	177	14	218
133	63	2	249	177	81	189	21

Figure 7. Final encrypted image obtained after data-hiding

### 3.2. Data extraction and image recovery

At the receiver side, the data extraction and image recovery are carried out through the image-decryption process. We assumed that the receiver who supposedly extracts the secret message knows that the decryption key is  $K$ . For image recovery and data extraction, the receiver will generate a pseudo-random matrix ( $U$ ) using a pseudo-random generator by providing decryption key  $K$ .

From the pseudo-random  $U$ , eight different matrices will be generated through the matrix rotation and paired bit-wise XOR operation. The matrices are named  $U_0, U_1, U_2, U_3, U_4, U_5, U_6$ , and  $U_7$  (this will be the same those given in Figure 5). Note that the size of random matrix  $U_P$  where  $P \in 0, 1, 2, 3, 4, 5, 6, 7$  is the same as the size of the encrypted image  $E$  that is denoted by  $R \times C$ . Image  $E$  and all of the  $U_P$  random matrices will be divided into non-overlapping blocks of size  $B \times B$  and the blocks will be processed in row-wise linear order.

Considering the  $X^{th}$  block (Block  $X$ ) from the encrypted image, eight versions of the decrypted image blocks (say,  $V_0, V_1, V_2, V_3, V_4, V_5, V_6$ , and  $V_7$ ) will be generated through bit-wise XOR operations with the  $X^{th}$  block from random matrix  $U_P$ , where  $P \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ . Then, we must compute a closeness measure by considering the pixels in  $V_P$  to get  $M_P$  by calling on Algorithm 3. It might be noted that one of the image blocks from  $\{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7\}$  will be the correctly recovered image block. To identify the correctly decrypted version, a new pixel-closeness measure can be used. In a correctly recovered image block, the closeness measure will be very low as compared to the other versions of the image blocks. For illustration, the first image block from the encrypted image and the eight different versions of the decrypted image blocks are shown in Figure 8.

From Figure 8, it can be seen that the correctly decrypted block was selected as  $V_6$ , since we got the smallest closeness measure (the pixel values are very close). Since  $V_6$  is the correctly decrypted block, the binary value that corresponds to 6 (which is 110) will be extracted from the block. It may be noted that, during the data-hiding process, we embedded 110 in this block. The same process will be continued for all of the remaining three blocks to get back the original image and to extract the 12-bit secret message hidden in the image.

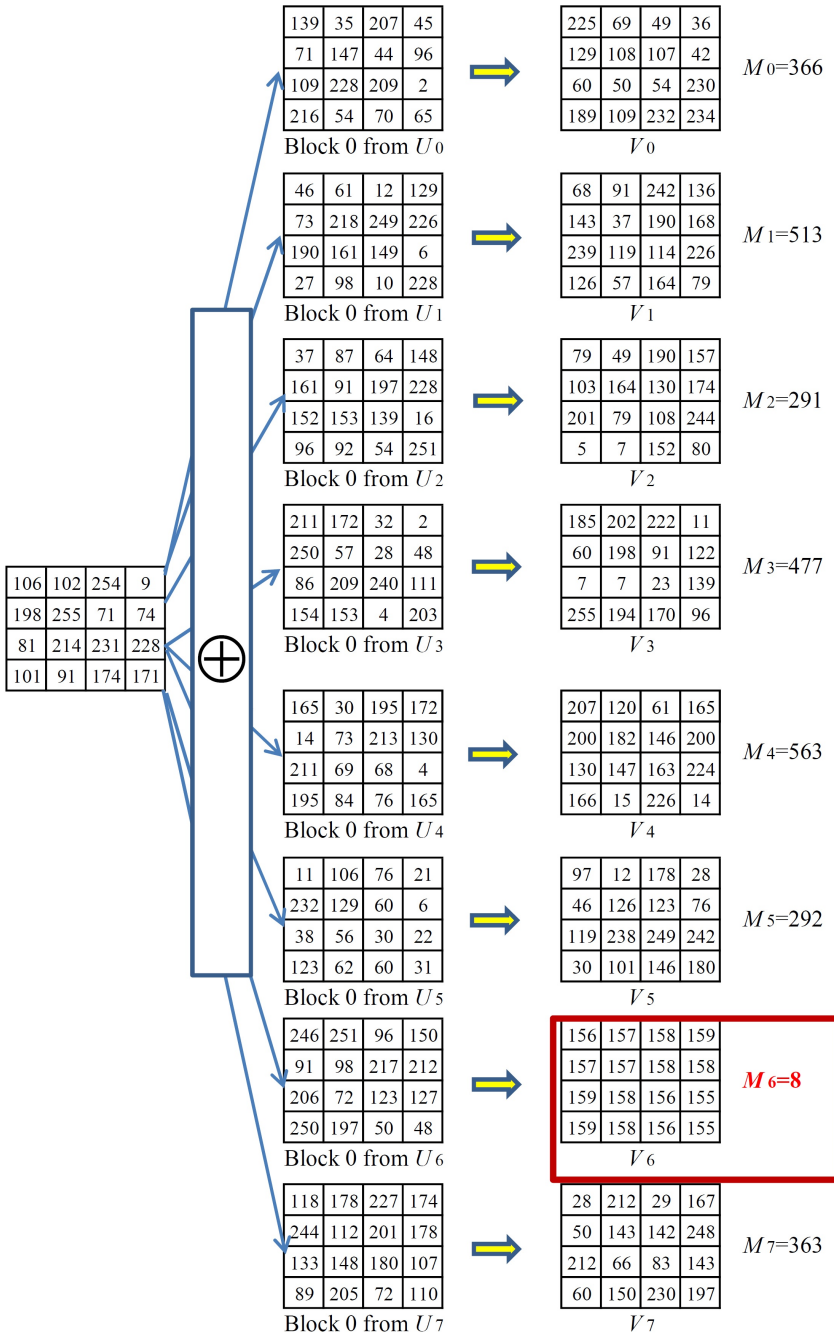


Figure 8. Final encrypted image obtained after data-hiding

The closeness measure obtained from all of the eight versions of the decrypted blocks is given in Table 1. From each block, minimum measure  $M_Q$  will be identified, and the three-bit binary that corresponds to  $Q$  will be the extracted bit sequence from that block.

**Table 1**

Closeness measure obtained from each block while attempting image recovery

	$M_0$	$M_1$	$M_2$	$M_3$	$M_4$	$M_5$	$M_6$	$M_7$
Block 0	366	513	291	477	563	292	<b>8</b>	363
Block 1	458	568	506	<b>5</b>	671	661	377	293
Block 2	564	351	374	<b>11</b>	279	474	444	553
Block 3	563	589	<b>6</b>	405	307	564	342	475

From Table 1, the following points can be observed:

- Block 0: the minimum value is  $M_6 = 8$ , so we will extract 110 from this block.
- Block 1: the minimum value is  $M_3 = 5$ , so we will extract 011 from this block.
- Block 2: the minimum value is  $M_3 = 11$ , so we will extract 011 from this block.
- Block 3: the minimum value is  $M_2 = 6$ , so we will extract 010 from this block.

So, the final extracted message will be the concatenated sequence of all of the three-bit sequences that are obtained from all of the four blocks, which will be (110011011010); this is the same as the embedded secret message bit sequence.

#### 4. Experimental study and result analysis

The experimental study of the proposed scheme was carried out on standard images that were downloaded from the USC-SIPI dataset, which is managed by the University of Southern California [22]. The USC-SIPI dataset consists of four different categories of images: *textures*, *aerials*, *sequences*, and *miscellaneous*. The miscellaneous category consists of well-known images like *a baboon*, *peppers*, *an airplane*, *a boat*, etc. During the experimental study, we converted all of the images into 8-bit grayscale images of a size of  $512 \times 512$  pixels. The implementation of the proposed scheme was done using Matlab2020 in a 64-bit Windows machine with an Intel (R) Core (TM) i5-8250U CPU with 8GB RAM.

The efficiency of a reversible data-hiding scheme is determined by four parameters: embedding rate, bit error rate, peak signal-to-noise ratio, and structural similarity index.

- Embedding rate. The embedding rate is the ratio between the total number of bits that are embedded in the image and the total number of pixels in the image. The embedding rate will be measured as *bits per pixel* (bpp). The embedding rate decides the amount of data that we can hide in an image using a reversible data-hiding scheme (and we prefer a reversible data-hiding scheme with a higher embedding rate).

- Bit error rate (BER). The number of bits that are incorrectly extracted from the image at the receiver side will determine the bit error rate. The BER is defined as follows:

$$BER = \frac{N_W}{N_{Tot}} \quad (1)$$

where  $N_W$  is the total number of bits that are incorrectly recovered, and  $N_{Tot}$  is the total number of bits that are extracted. Ideally, the bit error rate should be 0 for a reversible data-hiding scheme.

- Peak signal-to-noise ratio (PSNR). The PSNR between the original image and the recovered image is a parameter that helps us measure the efficiency of the image-recovery process. The PSNR between original image  $O$  and recovered image  $I$  of a size of  $R \times C$  pixels is defined as follows:

$$MSE = \frac{\sum_{x=1}^R \sum_{y=1}^C [O_{x,y} - I_{x,y}]^2}{(R \times C)} \quad (2)$$

the  $MSE$  is needed to compute the PSNR value.

$$PSNR = 10 \cdot \log_{10} \left( \frac{255^2}{MSE} \right) \quad (3)$$

If an original image and recovered image are the same, then the PSNR value will be  $\infty$ . In a reversible data-hiding scheme, we expect a PSNR value of  $\infty$  between the original image and the recovered image.

- Structural similarity index (SSIM). The SSIM is another measure for finding the efficiency of the image-recovery process in a reversible data-hiding process. The SSIM checks the structural similarity between two images; these values fall within a range 0 to 1. If two images are the same, the SSIM measure will be 1. In reversible data-hiding, we expect an SSIM value of 1 when comparing an original image with a recovered image.

#### 4.1. Analysis of image recovery

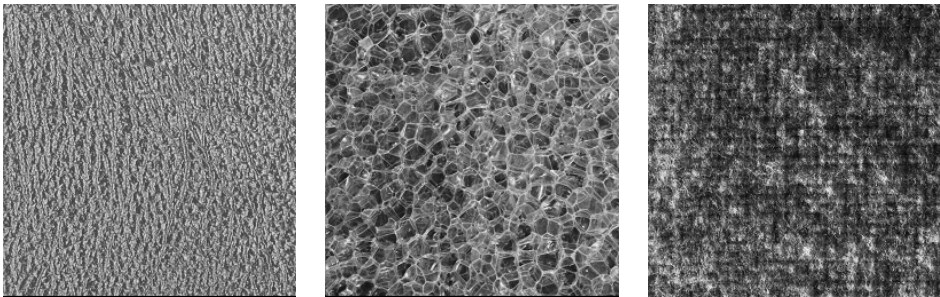
The primary focus of any reversible data-hiding scheme is the recoverability of the original image during data extraction. In the proposed scheme, the recovery of the images during data extraction is influenced by the block size that we are using. A smaller block may lead to an error during image recovery; larger blocks will help us improve the image recovery capability of the proposed scheme. The number of images recovered correctly by the proposed scheme while using different block sizes is shown in Table 2.

From Table 2, it can be seen that all of the images were recovered perfectly while we utilized a block size of  $32 \times 32$  pixels. It may also be noted that many images were recovered correctly while we used a block size of  $8 \times 8$  pixels. In addition, it can be seen that a lower number of texture images were recovered correctly (67.18%) while we used a block size of  $8 \times 8$  pixels.

**Table 2**  
Number of images recovered while using different block sizes

Image category	Total number of images	$4 \times 4$	$8 \times 8$	$16 \times 16$	$32 \times 32$
Miscellaneous	39	1	37	39	39
Arial	38	0	38	38	38
Sequences	69	0	69	69	69
Textures	64	0	43	60	64

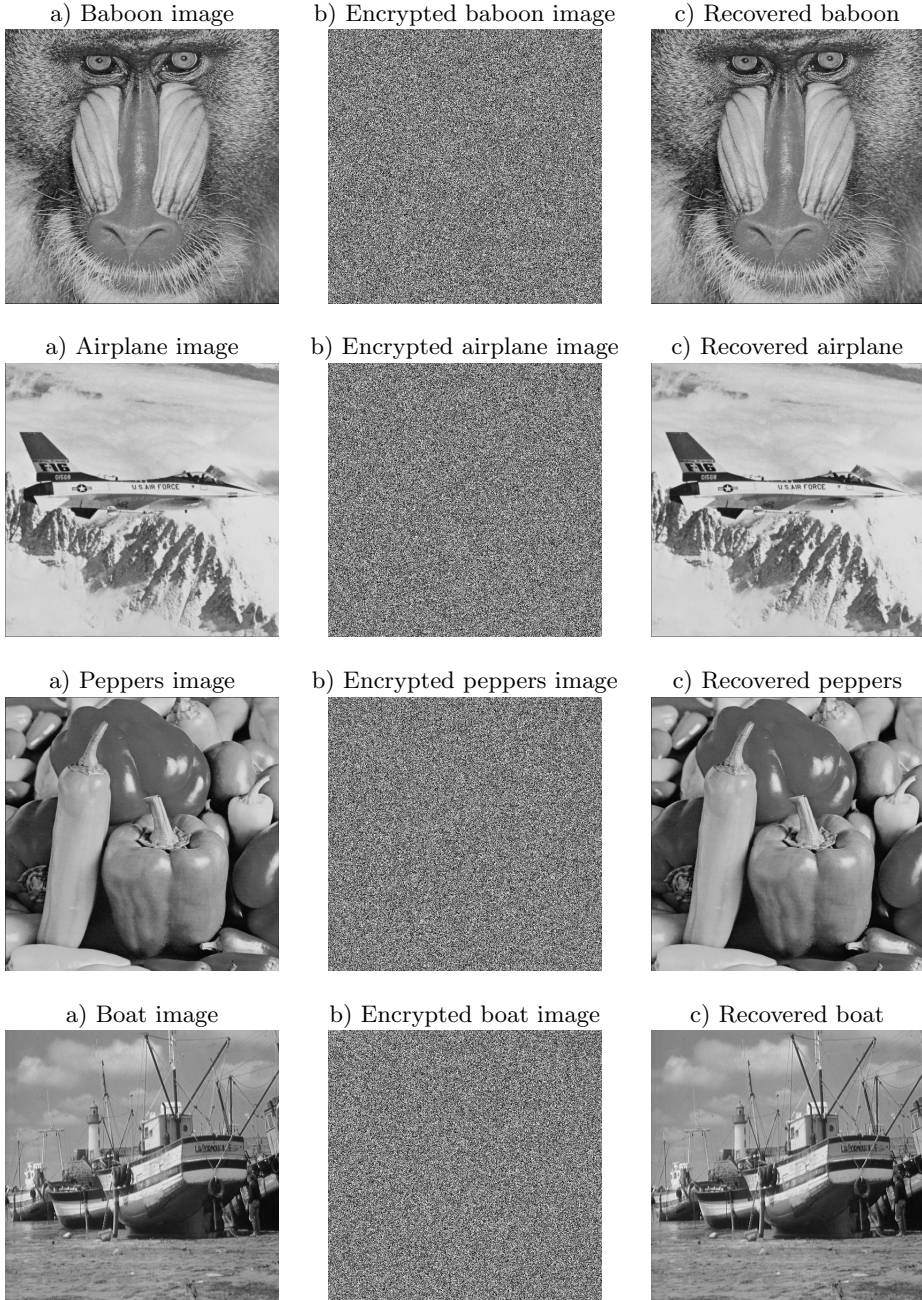
The texture images have a random pixel distribution, which leads to low image recoverability. We analyzed the closeness between the pixels in the images during data extraction and image recovery, but the texture images themselves are not smooth in nature. A few sample images from the texture-image category are shown in Figure 9.



**Figure 9.** Sample texture images that are not smooth (high smoothness measure)

The sample results obtained during the experimental of the proposed scheme on the well-known images such as *baboon*, *airplane*, *peppers*, and *boat* are shown in Figure 10.

The bit error rate, PSNR, and SSIM values obtained from the selected images are given in Table 3. It can be seen that all of the images were recovered correctly when we used a block size of  $8 \times 8$  pixels or more. So, the embedding rate for all of these images was  $\frac{3}{64} = 0.0468$  bpp. Another important property we can see from Table 3 is that a higher bit error was reported from the baboon and boat images than the other two images while we used a block size of  $4 \times 4$  pixels. The reason for this is that the baboon and boat images are highly textured as compared to the pepper and airplane images.



**Figure 10.** Sample images considered and corresponding encrypted images and recovered images obtained during experimental study (block size =  $8 \times 8$  pixels)



**Table 3**  
Bit error rate, PSNR, and SSIM measure obtained for well-known images  
from US-SIPI dataset

Image name	Block size	Bit error rate (BER)	PSNR [dB]	SSIM
Baboon	$4 \times 4$	0.0324	23.60	0.925
	$8 \times 8$	0	$\infty$	1
	$16 \times 16$	0	$\infty$	1
Airplane	$4 \times 4$	0.0143	26.08	0.959
	$8 \times 8$	0	$\infty$	1
	$16 \times 16$	0	$\infty$	1
Peppers	$4 \times 4$	0.0099	27.25	0.965
	$8 \times 8$	0	$\infty$	1
	$16 \times 16$	0	$\infty$	1
Boat	$4 \times 4$	0.0228	24.89	0.942
	$8 \times 8$	0	$\infty$	1
	$16 \times 16$	0	$\infty$	1

## 4.2. Analysis of embedding rate

The embedding rate from the proposed scheme is purely dependent on the block size that we use during the data-hiding/image-encryption process. A small block size will lead to a greater number of blocks in the cover image, and this will lead to a high embedding rate. Since the block size also affects the bit error rate, we cannot utilize overly small blocks to improve the embedding rate. We know that we expect a bit error rate of 0 from a reversible data-hiding scheme. From Table 2, it can be seen that, while we utilize a block size of  $8 \times 8$  pixels, all three categories of the images (*miscellaneous*, *aerial*, and *sequences*) were recovered correctly. In such cases, the embedding rate from all of the images from all of these categories will be  $\frac{3}{(8 \times 8)} = 0.0468$  bpp. From the *Texture* image category, all of the images were recovered when we used a block size of  $16 \times 16$  pixels, which yielded an embedding rate of  $\frac{3}{(16 \times 16)} = 0.0117$  bpp.

The closeness between the adjacent pixels in the *texture* image category is much less, which yields a high measure from the correctly recovered image blocks. From some blocks, the closeness measure from a correctly recovered block is greater than the measure from incorrectly recovered image blocks.

## 4.3. Analysis of time complexity

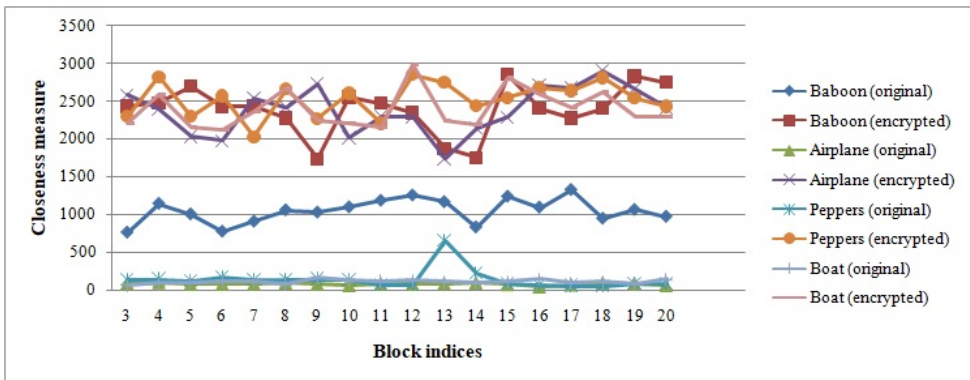
The theoretical time complexity of the algorithms can be considered as one of the efficiency parameters of a reversible data-hiding scheme. Let us assume that an original image has a size of  $N = R \times C$  pixels. Now, during the reversible data-hiding/image-encryption process, all of the pixels of the images want to be processed once at most for image encryption. Note that, during image encryption, each pixel (from selected blocks) will be XORed with a pseudo-random integer that is generated through the

RC4 stream generator. Note that the pseudo-random byte generation and the rotation of the stream ciphers are bounded by  $O(N)$ ; hence, the time complexity of the proposed reversible data-hiding-through-encryption scheme is also  $O(N)$ .

Similarly, during data extraction and image recovery, eight different versions of the image block is to be generated by XORing the blocks from the encrypted image with the stream ciphers that are generated. This operation is purely dependent on image size; at most, each pixel needs to be accessed eight times. Next, the main operation that we need to carry out at the receiver side is the closeness measure computation from each of the decrypted image blocks. From eight versions of image blocks, we want to compute the smoothness measure. This can also be carried out over a constant time since the block size is constant. So, the overall process required to carry out data extraction and image recovery also depends on the total number of pixels in an image. So, the time complexity of the algorithms for data extraction and image recovery is also  $O(N)$ , where  $N = R \times C$ .

#### 4.4. Justification to use new closeness measure

In the proposed scheme, we have introduced a new closeness measure to analyze the closeness between the pixels of an image block. Our key assumption is that the pixels in a natural image block will be very close, and the pixels in an encrypted block (decrypted with the wrong stream ciphers) will have a high difference between adjacent pixels. With this assumption only, we introduced Algorithm 3 to identify the correctly recovered image block. The proposed scheme may face some kind of challenge during recovery, when the original image itself may contain some random textures. The closeness measure observed from the first 20 blocks of 4 images (*a baboon*, *an airplane*, *peppers*, and *a boat*) in their original forms and their encrypted forms is shown in Figure 11.



**Figure 11.** Closeness measure obtained from original image blocks and encrypted image blocks

From Figure 11, it can be seen that the closeness measure from the encrypted images is very high as compared to the closeness measure from the original image blocks. Since the baboon image is highly textured, the closeness measure that is obtained from the original baboon image is slightly high as compared to the closeness measure obtained from the other images. But still, the closeness measures from all of the original images are less than the closeness measures from the corresponding encrypted image blocks. We utilized this property for data extraction and image recovery.

## 4.5. Security analysis

For encryption purposes, the proposed scheme can use any secure image-encryption technique. During our experimental study, we used the well-known RC4 data-encryption technique for image encryption. The RC4 algorithm will generate a sequence of pseudo-random bytes based on a given key value. While considering the security of the proposed scheme, it may be more secure than with conventional RC4 image encryption (or at least the same level of security).

The security analysis of the proposed scheme was carried out by analyzing the entropy and histogram of each encrypted image.

### 4.5.1. Analysis of entropy

The entropy is a statistical measure for measuring the randomness of the pixels in a given image, and the method of computing entropy  $E_P$  from an image is defined in Equation (4):

$$E_P = - \sum_{i=0}^{n-1} P_i \log_2 P_i \quad (4)$$

where  $P_i$  is the probability of the occurrence of pixel  $i$  in the image.

From the encrypted image, we expect an entropy measure that is close to 8 for a grayscale image that is represented using 8 bits. The average entropy measure obtained from the well-known original images and the corresponding encrypted images are given in Table 4.

**Table 4**

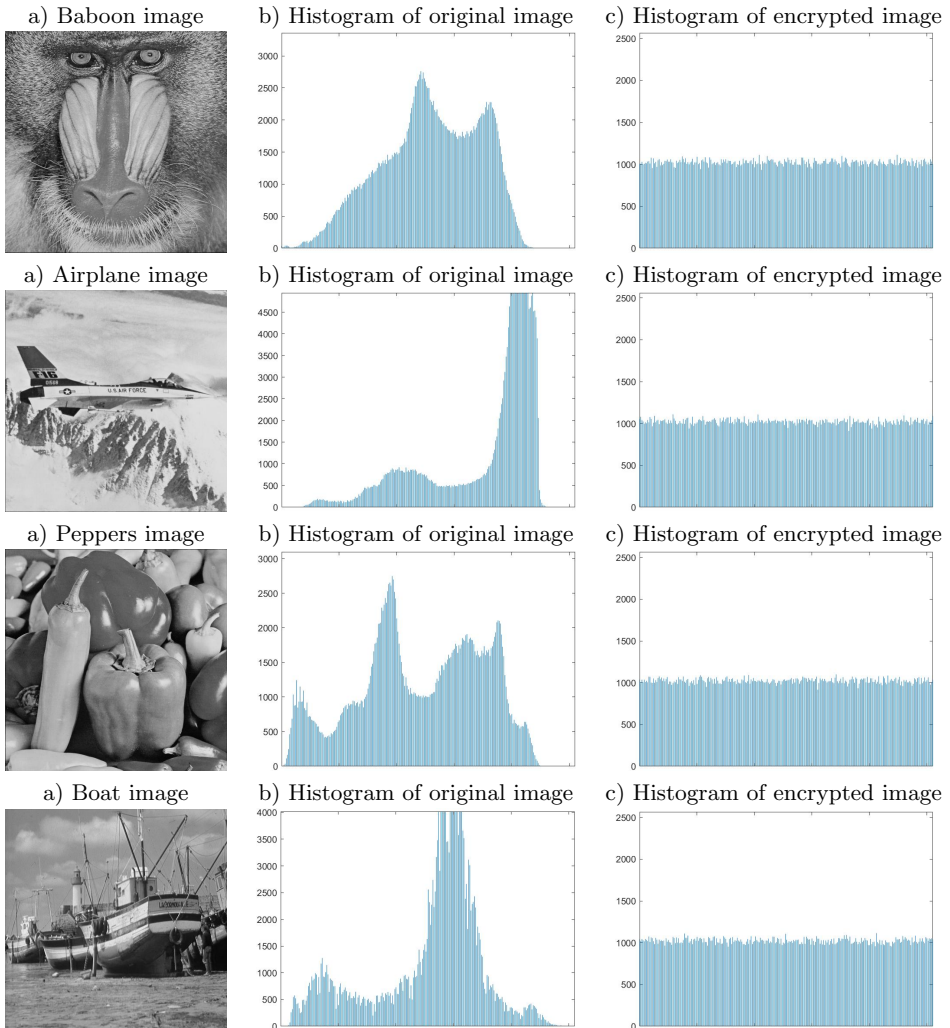
Entropy measure from original image ( $E_1$ ) and corresponding encrypted image ( $E_2$ )

Image name	$E_1$	$E_2$
Baboon	7.35	7.99
Airplane	6.70	7.99
Peppers	7.59	7.99
Boat	7.19	7.99

From Table 4, it can be observed that the entropy from all of the encrypted images is very close to 8; this indicates that the encrypted image pixels have the maximum amount of randomness.

#### 4.6. Analysis of histogram

Histogram analysis is another method that can be used to evaluate the efficiency of an encrypted image. The pixels in an encrypted image are expected to be uniformly distributed, which yields a histogram with a flat shape. The histogram obtained from an original image and its corresponding encrypted image is given in Figure 12.



**Figure 12.** Sample images considered and the corresponding encrypted images and recovered images obtained during experimental study (block size =  $8 \times 8$  pixels)

From Figure 12, it can be seen that the histogram obtained after image encryption is flat irrespective of the shape of the histogram of the original image. This indicates that the encrypted image pixels are uniformly distributed and difficult to do some cryptanalysis to recover the original image by an unauthorized person.

In addition to the experimental analysis of the proposed scheme, we have analyzed the encryption efficiency using theoretical analysis. Let us assume that some intruder tries to recover the original image from the encrypted image through a brute-force approach. Since we have considered a block size of  $8 \times 8$  pixels, there are 64 pixels in each image block. For each pixel in the image block, the intruder attempts a decryption with all of the pseudo-random numbers from 0 to 255. So, 256 attempts are required to acquire the correctly decrypted pixel. The probability to acquire the correct image pixel at location  $(x, y)$  (say,  $P(I_{x,y})$ ) can be defined as follows:

$$P(I_{x,y}) = \left( \frac{1}{256} \right) \quad (5)$$

Now, this attempt should be carried out for all of the pixels in a given block of a size of  $8 \times 8$  pixels. So, the probability of recovering the  $K^{th}$  original image block block (say  $P(I_K)$ ) is:

$$P(I_K) = \left( \frac{1}{256} \right)^{(8 \times 8)} \quad (6)$$

The probability of recovering the original image by the intruder again depends on the number of blocks in an image. If the original image consists of  $R \times C$  pixels and the block size is  $B \times B$  pixels, then the total image will have  $\lfloor \frac{R}{B} \rfloor \times \lfloor \frac{C}{B} \rfloor$  blocks. Hence, the probability of recovering the original image as it is by an intruder (say,  $P(I)$ ) is defined as follows:

$$P(I) = \left( \frac{1}{256} \right)^{(8 \times 8) \times (\lfloor \frac{R}{B} \rfloor \times \lfloor \frac{C}{B} \rfloor)} \approx 0 \quad (7)$$

When we consider an image of a size of  $512 \times 512$  pixels, it will have 4,096 blocks; in such a case, the probability of recovering the original image by an intruder is given below:

$$P(I) = \left( \frac{1}{256} \right)^{262144} \approx 0 \quad (8)$$

From the above discussions, it is clear that it is almost impossible for an intruder to recover the original image. To provide security for the hidden message, we shall introduce a data-hiding key to select the image blocks in pseudo-random order. The data-hiding key will generate a pseudo-random sequence of numbers within a range of 1 to  $(\lfloor \frac{R}{B} \rfloor \times \lfloor \frac{C}{B} \rfloor)$ ; the image blocks can be processed in this order instead of row-wise linear accessing. In an image that has a size of  $512 \times 512$ , there will be 4,096 number

of  $8 \times 8$  image blocks. Note that these 4,096 blocks can be accessed in 4,096 different ways; the blocks can be accessed when we use a data-hiding key, and 4,096 is a very large number.

The theoretical analysis proves that the security of the proposed scheme is good, as it is almost impossible for an intruder to break the scheme.

#### 4.7. Comparative study

Most of the reversible data-hiding schemes embed secret messages on natural images or in encrypted images. However, the proposed scheme will do reversible data-hiding during the image-encryption process. A comparison of the embedding rate from recent reversible data-hiding schemes in encrypted images [1, 5, 8, 11, 19, 26–28] with the proposed scheme is given in Table 5.

**Table 5**  
Comparison of embedding rate for well-known images

Scheme	Airplane	Peppers	Boat	Baboon
Scheme in [26]	0.0039	0.0015	0.0039	0.0009
Scheme in [8]	0.0039	0.0039	0.0039	0.0009
Scheme in [27]	0.0300	0.0300	0.0300	0.0100
Scheme in [5]	0.0020	0.0020	0.0020	0.0020
Scheme in [19]	0.0040	0.0040	0.0040	0.0040
Scheme in [28]	0.0080	0.0080	0.0080	0.0080
Scheme in [11]	0.0080	0.0080	0.0080	0.0080
Scheme in [1]	0.0039	0.0039	0.0039	0.0039
Proposed Scheme	0.0468	0.0468	0.0468	0.0468

Note that the proposed scheme successfully recovers the original image with an embedding rate of 0.0468 bpp. The results obtained from the proposed scheme are far better than the results from the existing schemes considered here. The key observations from the experimental study and comparative result analysis are listed below.

- The proposed scheme is a reversible data-hiding scheme that is capable of carrying out the data-hiding during the image-encryption process. The proposed scheme will be useful when we want to protect the confidentiality of an image's contents and if there is also a need for sending an additional secret message along with the image.
- The proposed scheme works perfectly fine when the images are smooth (fewer texture variations).
- Most of the well-known images like a baboon, an airplane, a boat, peppers, etc. are perfectly recovered while we utilize a block size of  $8 \times 8$  pixels; this leads to an embedding rate of 0.0468 bpp.

## 5. Conclusion

A reversible data-hiding-through-encryption scheme is introduced in this paper with the help of a rotated stream cipher. The reversible data-hiding scheme discussed in this manuscript will be useful in medical image transmission for sending electronic patient records along with medical images. Similarly, the proposed scheme will be useful for embedding metadata or authentication information in multimedia data that is stored in cloud storage. Since the proposed scheme combines both image encryption and data-hiding into a single process, the proposed scheme is computationally efficient; the scheme provides a good embedding rate of 0.0468 bits per pixel. The comparative study shows that the embedding rate obtained from the proposed scheme outperforms the embedding rate from the well-known existing reversible data-hiding schemes. The data extraction/image recovery process is carried out using a new smoothness measure that considers the absolute difference between adjacent pixels. The proposed scheme only involves the concern of sharing an encryption key with a receiver; the same key can be used for transmitting any number of images. Note that key sharing is always present in any cryptosystem, and our scheme does not involve any other overhead. Future work can be carried out to define a new smoothness measure that will also work on highly textured images.

## References


- [1] Agrawal S., Kumar M.: Mean value based reversible data hiding in encrypted images, *Optik*, vol. 130(2017), pp. 922–934, 2017.
- [2] Bender W., Butera W., Gruhl D., Hwang R., Paiz F.J., Pogreb S.: Applications for data hiding, *IBM Systems Journal*, vol. 39(3.4), pp. 547–568, 2000.
- [3] Celik M.U., Sharma G., Tekalp A.M., Saber E.: Reversible data hiding. In: *International Conference on Image Processing*, pp. 157–160, IEEE, 2002.
- [4] Chen X., Sun X., Sun H., Zhou Z., Zhang J.: Reversible watermarking method based on asymmetric-histogram shifting of prediction errors, *Journal of Systems and Software*, vol. 86(10), pp. 2620–2626, 2013.
- [5] Chen Y.C., Shiu C.W., Horng G.: Encrypted signal-based reversible data hiding with public key cryptosystem, *Journal of Visual Communication and Image Representation*, vol. 25(5), pp. 1164–1170, 2014.
- [6] Cox I., Miller M., Bloom J., Fridrich J., Kalker T.: *Digital Watermarking and Steganography*, Morgan Kaufmann, 2007.
- [7] Fu D.S., Jing Z.J., Zhao S.G., Fan J.: Reversible data hiding based on prediction-error histogram shifting and EMD mechanism, *AEU – International Journal of Electronics and Communications*, vol. 68(10), pp. 933–943, 2014.
- [8] Hong W., Chen T.S., Wu H.Y.: An Improved Reversible Data Hiding in Encrypted Images Using Side Match, *IEEE Signal Processing Letters*, vol. 19(4), pp. 199–202, 2012.

- [9] Kim H.J., Sachnev V., Shi Y.Q., Nam J., Choo H.G.: A Novel Difference Expansion Transform for Reversible Data Embedding, *IEEE Transactions on Information Forensics and Security*, vol. 3(3), pp. 456–465, 2008.
- [10] Klein A.: RC4 and Related Ciphers. In: *Stream Ciphers*, pp. 183–228, Springer, London, 2013.
- [11] Li M., Li Y.: Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding, *Signal Processing*, vol. 130, pp. 190–196, 2017.
- [12] Manikandan V.M., Masilamani V.: Reversible Data Hiding Scheme During Encryption Using Machine Learning, *Procedia Computer Science*, vol. 133, pp. 348–356, 2018.
- [13] Manikandan V.M., Masilamani V.: An Improved Reversible Data Hiding Scheme Through Novel Encryption. In: *2019 Conference on Next Generation Computing Applications (NextComp)*, pp. 1–5, IEEE, 2019.
- [14] Manikandan V.M., Masilamani V.: A Novel Entropy-based Reversible Data Hiding during Encryption. In: *2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP)*, pp. 1–6, IEEE, 2019.
- [15] Ni Z., Shi Y.Q., Ansari N., Su W.: Reversible data hiding, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16(3), pp. 354–362, 2006.
- [16] Ou B., Zhao Y., Ni R.: Reversible watermarking using optional prediction error histogram modification, *Neurocomputing*, vol. 93, pp. 67–76, 2012.
- [17] Parah S.A., Ahad F., Sheikh J.A., Bhat G.M.: Hiding clinical information in medical images: A new high capacity and reversible data hiding technique, *Journal of Biomedical Informatics*, vol. 66, pp. 214–230, 2017.
- [18] Shi Y.Q., Li X., Zhang X., Wu H.T., Ma B.: Reversible data hiding: Advances in the past two decades, *IEEE Access*, vol. 4, pp. 3210–3237, 2016.
- [19] Shiu C.W., Chen Y.C., Hong W.: Encrypted image-based reversible data hiding with public key cryptography from difference expansion, *Signal Processing: Image Communication*, vol. 39, pp. 226–233, 2015.
- [20] Thodi D.M., Rodríguez J.J.: Expansion Embedding Techniques for Reversible Watermarking, *IEEE Transactions on Image Processing*, vol. 16(3), pp. 721–730, 2007.
- [21] Thodi D.M., Rodriguez J.J.: Prediction-error based reversible watermarking. In: *2004 International Conference on Image Processing, ICIP'04*, vol. 3, pp. 1549–1552, IEEE, 2004.
- [22] USC-SIPI: image database, <http://sipi.usc.edu/database/database.php?volume=misc>. Accessed: 10-12-2016.
- [23] Xiong L., Xu Z., Shi Y.Q.: An integer wavelet transform based scheme for reversible data hiding in encrypted images, *Multidimensional Systems and Signal Processing*, vol. 29(3), pp. 1191–1202, 2018.
- [24] Yin Z., Abel A., Tang J., Zhang X., Luo B.: Reversible data hiding in encrypted images based on multi-level encryption and block histogram modification, *Multimedia Tools and Applications*, vol. 76(3), pp. 3899–3920, 2017.



- [25] Zhang W., Wang H., Hou D., Yu N.: Reversible Data Hiding in Encrypted Images by Reversible Image Transformation, *IEEE Transactions on Multimedia*, vol. 18(8), pp. 1469–1479, 2016.
- [26] Zhang X.: Reversible Data Hiding in Encrypted Image, *IEEE Signal Processing Letters*, vol. 18(4), pp. 255–258, 2011.
- [27] Zhang X.: Separable Reversible Data Hiding in Encrypted Image, *IEEE Transactions on Information Forensics and Security*, vol. 7(2), pp. 826–832, 2011.
- [28] Zhang X., Long J., Wang Z., Cheng H.: Lossless and Reversible Data Hiding in Encrypted Images With Public-Key Cryptography, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26(9), pp. 1622–1631, 2015.

## Affiliations

V.M. Manikandan 

SRM University-AP, Andhra Pradesh, India, manikandan.v@srmmap.edu.in,  
ORCID ID: <https://orcid.org/0000-0001-6903-7563>

**Received:** 04.06.2020

**Revised:** 04.01.2021

**Accepted:** 15.02.2021