electronics

MDPI

Article

An Efficient Separable Reversible Data Hiding Using Paillier Cryptosystem for Preserving Privacy in Cloud Domain

Ahmad Neyaz Khan [1,*], Ming Yu Fan [1,*], Muhammad Irshad Nazeer [2], Raheel Ahmed Memon [2], Asad Malik [3] and Mohammed Aslam Husain [4]

[1] School of Computer Science and Engineering, UESTC, Chengdu 611731, China
[2] Department of Computer Science, Sukkur IBA University, Sukkur 65200, Pakistan; irshad.nazeer@iba-suk.edu.pk (M.I.N.); raheelmemon@iba-suk.edu.pk (R.A.M.)
[3] School of Information Science & Technology, SWJTU, Chengdu 614200, China; asad@my.swjtu.edu.cn
[4] EED, REC, Ambedkarnagar 224122, India; mahusain87@gmail.com
* Correspondence: ahmadnk500@gmail.com (A.N.K.); ff98@163.com (M.Y.F.); Tel.: +86-182-8024-9324 (A.N.K.)

Received: 23 April 2019; Accepted: 3 June 2019; Published: 17 June 2019

Abstract: Reversible data hiding in encrypted image (RDHEI) is advantageous to scenarios where complete recovery of the original cover image and additional data are required. In some of the existing RDHEI schemes, the image pre-processing step involved is an overhead for the resource-constrained devices on the sender's side. In this paper, an efficient separable reversible data hiding scheme over a homomorphically encrypted image that assures privacy preservation of the contents in the cloud environment is proposed. This proposed scheme comprises three stakeholders: content-owner, data hider, and receiver. Initially, the content-owner encrypts the original image and sends the encrypted image to the data hider. The data hider embeds the encrypted additional data into the encrypted image and then sends the marked encrypted image to the receiver. On the receiver's side, both additional data and the original image are extracted in a separable manner, i.e., additional data and the original image are extracted independently and completely from the marked encrypted image. The present scheme uses public key cryptography and facilitates the encryption of the original image on the content-owner side, without any pre-processing step involved. In addition, our experiment used distinct images to demonstrate the image-independency and the obtained results show high embedding rate where the peak signal noise ratio (PSNR) is $+\infty$ dB for the directly decrypted image. Finally, a comparison is drawn, which shows that the proposed scheme is an optimized approach for resource-constrained devices as it omits the image pre-processing step.

Keywords: reversible data hiding (RDH); image processing; cloud computing; public key cryptography (PKC); security

1. Introduction

Data hiding is one of the techniques used for securing data, apart from encrypting data. In data encryption technique, the original data are converted into a non-interpretable form so that adversary cannot extract any useful information. In data hiding, additional data are embedded into the carrier cover media (text, audio, video and image) in such a way that they remain concealed and can be extracted from the cover media later. Cover media is the original media that is used to carry additional data.

However, it is notable that, in data hiding, when data are extracted from the cover media, some form of distortion remains in the recovered cover media. In some scenarios (e.g., medical and satellite imagery), distortion in the cover image is inadmissible. That is, the image to be recovered on the receiver's side needs to be lossless. To cope with this problem, several reversible data hiding (RDH)

techniques have been proposed. RDH is a technique to manipulate pixel bits of the cover image to create some space for embedding the additional data into the cover image, where both the additional data and the original cover image can be recovered completely. This is done while maintaining the perceptible quality of the carrier media. RDH schemes can be broadly classified into three categories: difference expansion [1], lossless compression [2] and histogram shifting [3].

With the growth of cloud-based applications [4,5], data outsourcing is one of the fields where the users are dependent on cloud for processing and storage. Security concern of the data owners using cloud for the cover media is addressed using encryption for the cover image. For the sake of management (using timestamp, tagging, image source information, etc.) of the encrypted media, the data hider embeds some additional data into the encrypted image. To achieve security and reversibility, the technique of RDH is used in the encrypted domain.

Reversible data hiding in encrypted images (RDHEI) is a method where additional data are embedded by the data hider into the encrypted cover image (obtained from the content-owner) to obtain marked encrypted image. This marked encrypted image is sent to the receiver, where recovery of both additional data and the cover image from the marked encrypted image is made losslessly.

Many RDHEI schemes based on the symmetric key have been proposed until now. In symmetric key cryptography, there is only one secret-key for both encryption and decryption, and key management is needed to share the secret key between the sender and the receiver. Some of the RDH schemes based on symmetric key cryptography are discussed below.

In 2008, Puech et al. [6] proposed the first RDHEI scheme where the Advanced Encryption Standard (AES) is used for encryption. Here, the encrypted image is divided into non-overlapping blocks of n-pixels each, and each block is responsible to carry one bit of additional data. The local standard deviation of the marked encrypted image is used to retrieve additional data on the receiver's side. In 2011, Zhang [7] successfully recovered an image similar to the original image with a secret key, which is encrypted using a stream cipher. With the help of data hiding key and spatial co-relation in the image, additional data and original image are recovered losslessly. Embedding is done using a two-step block division of the encrypted original image, and using least significant bit (LSB) flipping to identify the type of embedded bit (0 or 1). Hong et al. [8] improved Zhang's scheme [7] by exploiting the correlations in neighboring border pixels, which were not taken into consideration by Zhang [7].

At the receiver side, additional data in RDHEI are broadly recovered in two ways, namely non-separable and separable techniques, which are respectively depicted in Figure 1a,b. To extract the additional data from the marked encrypted image using the aforementioned schemes, the receiver must have the data-hiding key and the secret key. As the additional data can only be extracted after image decryption, this type of schemes falls under the category of non-separable RDHEI, as depicted in Figure 1a, where the additional data cannot be extracted without decrypting the marked encrypted image.

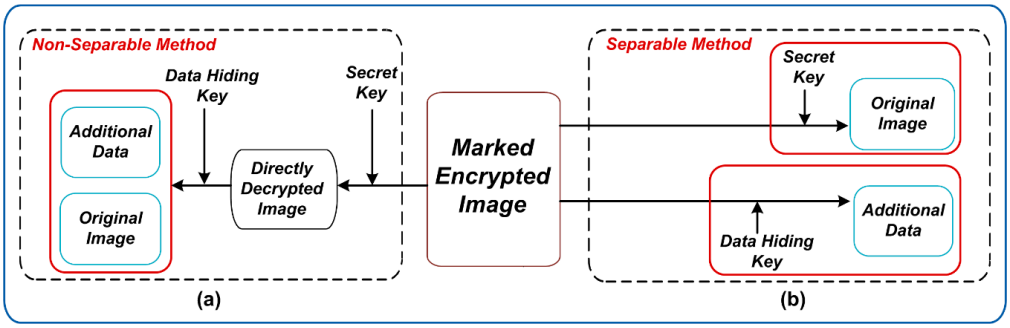

Figure 1. (**a**) Non-separable; and (**b**) separable methods in reversible data hiding.

Figure 1b depicts the separable method used in RDHEI, where the receiver can extract the additional data independent of image decryption, with the use of data hiding key only. The original image can be recovered by only using the secret key.

Zhang [9] proposed the first separable RDHEI method in 2012. RDHEI methods are fit for the cloud applications, where the data hider using the cloud can embed additional data for the sake of management. In addition, the receiver can extract additional data without knowing the content of the original image. The privacy of the cover image is still preserved as the additional data can be extracted without image decryption. Yin et al. [10] proposed a separable RDHEI scheme by breaking the cover image into non-overlapping blocks and the additional data are embedded into the blocks using block smoothness order and peak points.

Generally, at the content-owner's side, creating space for embedding data is done mainly in two ways: Vacating room after encryption (VRAE) (Figure 2a) and vacating room before encryption (VRBE) (Figure 2b). In VRAE, the space to embed additional data by the data hider is created after the image encryption. However, in VRBE, the space to embed additional data by the data hider is created before the image encryption. The framework followed by the above schemes is VRAE.

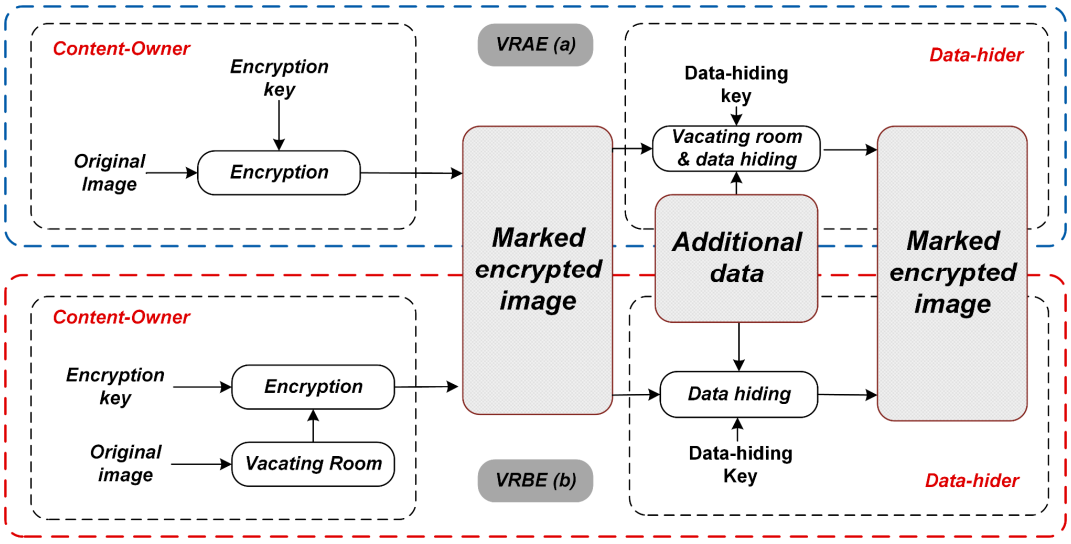

Figure 2. (**a**) Vacating room after encryption (VRAE); and (**b**) vacating room before encryption (VRBE).

After the image encryption, entropy rises, which allows less space for payload (additional data). To tackle this problem, Ma et al. [11] proposed the first VRBE scheme by preprocessing the cover image, using traditional RDH scheme. Zhang et al. [12] preprocessed the cover image by estimating some pixels in the cover image. Then, they embedded additional data by using histogram shifting for estimated prediction errors.

Qian et al. [13] significantly enhanced image quality and embedding rate using histogram modification, based on n-nary histogram intensities. However, the image histogram leakage reduced image security. Zhang et al. [14] embedded compressed additional data into an encrypted image using low-density parity check code. Zheng et al. [15] compressed the pixel LSBs, in the chaotic-encrypted image, using Hamming distance to embed data. Cao et al. [16] used patch level sparse representation to embed data, using the sparse coding technique. Self-embedding of leading residual errors and embedding over-complete dictionary (created by using sparse coefficients to represent cover image) into the encrypted image is used in this technique.

Recently, public key cryptography is used for efficient key management over cloud [17,18], specifically in RDHEI [19–24]. In asymmetric key (or public key) cryptography, the key for encryption (public-key) and the key for decryption (private-key) are different. In the context of RDHEI using public-key, Chen et al. [19] proposed the first signal based reversible data hiding, for multiple signals and data hiders, using Paillier cryptosystem [25]. Shiu et al. [20] improved the work of Chen et al. [19], having a drawback of inherent overflow, by grouping 64-pixels of the encrypted image followed by compression. Li et al. [21] used difference histogram shifting based on the additive homomorphic

property to embed additional data. Wei et al. [22] enhanced the work of Li et al. [21], by taking the cross-shaped division mask instead of block-based division mask to use all the embedding opportunities. Zhang et al. [23] proposed reversible and lossless RDHEI in public-key, with better performance in terms of PSNR of the directly decrypted image compared to some of the previous schemes.

Tai et al. [24] proposed RDHEI scheme based on Paillier's cryptosystem. Here, the image is preprocessed before encryption by dividing each pixel into two parts, called encrypted units: EU^1 and EU^2. The order of the two encrypted parts of a pixel, namely EU^1 and EU^2, is exploited to embed the bits of additional data in the data hiding phase. If the additional bit to be embedded is 1 and $EU^1 < EU^2$, then swap EU^1 and EU^2. If the additional bit to be embedded is 0 and $EU^1 > EU^2$, then swap EU^1 and EU^2. On the receiver side, the additive homomorphic property of Paillier's cryptosystem is used to recover the original image losslessly. Again, the order of the two encrypted parts, EU^1 and EU^2, is used to extract the bits of additional data from the encrypted image. In [24], the image is preprocessed before encryption and this makes the content-owner to send double the size of image to the data hider. This issue is the one addressed in our proposed scheme.

In this paper, an efficient separable reversible data hiding scheme for encrypted images is proposed. It enjoys the benefits of using Paillier cryptosystem [25], which provides privacy preserving advantage over the cloud. The usage of our scheme befits cloud domain. Thus, a real-life application scenario of the proposed scheme over cloud is vividly explained in Section 3.6. Paillier's cryptosystem is used to encrypt the original image with the public key. When the encrypted image is received by the data hider, the data-hiding key and the public key is used to embed additional data into the encrypted image to obtain marked encrypted image. The additional data and the original image are recovered losslessly at the receiver's side using respective data hiding and private keys, independent of each other. This is done in a separable manner, which means for embedded additional data extraction, encrypted image is not required to be decrypted and vice versa.

After the brief discussion of previous schemes, we move on to a brief introduction to Paillier's public key cryptosystem in Section 2. In Section 3, the proposed scheme is discussed with an example. Experimental results and discussions are covered in Section 4. Finally, a conclusion is drawn in Section 5.

2. Paillier Cryptosystem

Paillier cryptosystem [25] is one of the most widely used public key cryptosystems, based on homomorphic properties along with probabilistic properties. Our scheme uses Paillier cryptosystem [25] for the encryption, decryption and its homomorphic properties have been exploited for data embedding. Homomorphic implies that arithmetic operations will be preserved from plaintext space to ciphertext space. It has given a strong base for secure computing on cloud, as it deals with privacy-preserving concerns of data owners. This is due to its property of semantic security, i.e., the same plaintext gives different ciphertexts, which obviously can be recovered using the private key. It means one cannot distinguish between the different ciphertexts generated from the same plaintext.

2.1. Key Generation

Two large prime numbers p and q of equal length are randomly chosen, satisfying $gcd\ (pq, (p-1) \times (q-1)) = 1$. Subsequently, the message sender calculates $\boldsymbol{N}$ and λ using $\boldsymbol{N} = pq$ and $\lambda = lcm(p-1, q-1)$. Again, some integer g following $g \in \mathbb{Z}^*_{\boldsymbol{N}}$ is randomly selected, such that it satisfies $gcd(L(g^{\lambda} mod \boldsymbol{N}^2), \boldsymbol{N}) = 1$, where $L(x) = (x-1)/\boldsymbol{N}$. As a result, we get the public key $(\boldsymbol{N}, g)$ and private key (λ).

2.2. Encryption

Let m be the given message to be encrypted, where integer $m \in \mathbb{Z}_N$. Select an integer $r \in \mathbb{Z}^*_{\boldsymbol{N}}$ randomly. The corresponding ciphertext c can be obtained using:

$$c = E[m, r] = g^m \times r^N mod \boldsymbol{N}^2 \quad (1)$$

where $E[\cdot]$ represent encryption function having the property of Paillier cryptosystem, and the ciphertext c lies in the set $\mathbb{Z}^*_{N^2}$.

2.3. Decryption

Using the private key (λ) in decryption function $D[\cdot]$, the user can decrypt the ciphertext c to get original message m, using:

$$m = D[c] = \frac{L(c^{\lambda} mod \boldsymbol{N}^2)}{L(g^{\lambda} mod \boldsymbol{N}^2)} mod \boldsymbol{N}. \tag{2}$$

2.4. Homomorphic Property

The encryption function is additive homomorphic, i.e., the multiplication of two ciphertexts will decrypt to the sum of their corresponding plaintexts. For two plaintexts $m_1, m_2 \in \mathbb{Z}_N$ and randomly selected integers $r_1, r_2 \in \mathbb{Z}^*_N$, the corresponding ciphertexts $c_1, c_2 \in \mathbb{Z}^*_{N^2}$ can be calculated as $c_1 = E[m_1, r_1] = g^{m_1} \times r_1^N mod \boldsymbol{N}^2$ and $c_2 = E[m_2, r_2] = g^{m_2} \times r_2^N mod \boldsymbol{N}^2$. In addition, c_1 and c_2 satisfy Equations (3) and (4):

$$c_1 \times c_2 = g^{m_1+m_2} \times (r_1 \times r_2)^{\boldsymbol{n}} mod \boldsymbol{N}^2, \tag{3}$$

$$D[g^{m_1+m_2} \times (r_1 \times r_2)^N mod \boldsymbol{N}^2] = m_1 + m_2 mod \boldsymbol{N}^2. \tag{4}$$

Semantic security is assured with the homomorphic property of the Paillier cryptosystem, as shown for a message $m \in \mathbb{Z}_N$ in Equation (5), where $r_1, r_2 \in \mathbb{Z}^*_N$, $c_1 = E[m, r_1]$ and $c_1 = E[m, r_2]$.

$$c_1 \neq c_2 \tag{5}$$

3. Proposed Scheme

The idea of our scheme is based on the separable method, as illustrated in Figure 3. The notations used in the proposed scheme are denoted in Table 1. It comprises of three stakeholders: the content-owner, the data-hider, and the receiver.

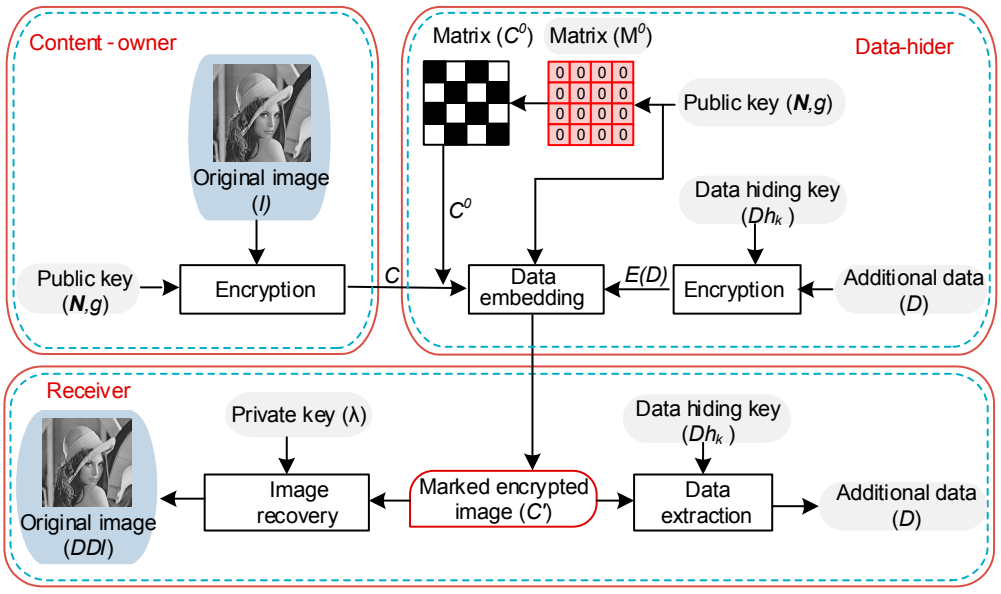

Figure 3. Working of the proposed scheme.

Table 1. Key notations used in the proposed scheme.

Notations	Description
$(\boldsymbol{N}, g)$	A public key for encryption
(Dh_k)	A data hiding key for hiding and recovery of additional data
λ	A private key possessed by the receiver for image recovery
I	An original image of size $L \times B$
k	Index for each pixel where $1 \leq k \leq L \times B$
I_k	kth pixel of the original image I
C_k	An encrypted value of I_k i.e., kth encrypted pixel of the encrypted image C
$E[\cdot]$	An encryption function
$D[\cdot]$	A decryption function
r_k	A randomly selected integer for each I_k such that $r_k \in \mathbb{Z}_N^*$
C	An encrypted image generated from all C_k achieved by pixel by pixel encryption
D	Additional data of size $L \times B$ bits to be embedded
$E(D)$	Encrypted additional data using (Dh_k)
M^0	A zero matrix of size $L \times B$ where all the elements are zero
C^0	A matrix resulting from encryption of matrix M^0
C_k^0	kth encrypted value of "0" from the matrix C^0
PU	A padded unit
PU_k	kth padded unit consisting of pair (C_k^0, C_k) where $1 \leq k \leq L \times B$
C'	A marked encrypted image *(MEI)* constituted from all the padded units (*PUs*)
DDI	A directly decrypted image
DDI_k	kth pixel of directly decrypted image (*DDI*)

Data-hider uses (Dh_k) to encrypt the additional data (D) and uses $(\boldsymbol{N}, g)$ to embed the encrypted additional data $E(D)$ into the encrypted image. At the receiver's end, additional data are recovered from the marked encrypted image C' using (Dh_k), and the original image I is completely recovered in a separable manner, using (λ) owned by the receiver.

3.1. Image Encryption

In this step, the content-owner scans each pixel of the original image I in the order from left to right and top to bottom, as shown in Figure 4 (for an image of size 5×5), to get the index k for each pixel. The size of I is $L \times B$, where $1 \leq k \leq L \times B$. This order is used throughout our proposed scheme to get the value at index k.

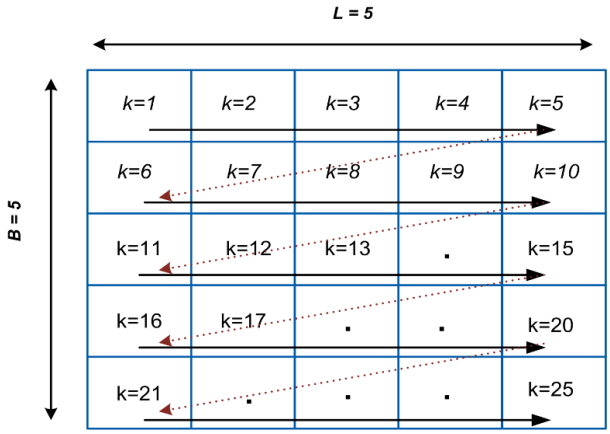

Figure 4. Scanning order to get index k for an image of size 5×5.

After scanning, the content-owner encrypts each I_k taken from original image I. Public key and Equation (1) are used to encrypt I_k to get the corresponding encrypted value C_k as follows:

$$C_k = E[I_k, r_k] = (g)^{I_k} \times (r_k)^N \ mod \ \boldsymbol{N}^2 \tag{6}$$

where $E[\cdot]$ is the encryption function and r_k is randomly selected integer for each I_k such that $r_k \in \mathbb{Z}_{\boldsymbol{N}}^*$. Step-wise encryption of the original image is as follows.

Step 1. Scan I for each pixel I_k in order from left to right and top to bottom (Figure 4).

Step 2. Encrypt each I_k using the public key $(\boldsymbol{N}, g)$ as in Equation (6) to get the encrypted pixel C_k.

Step 3. After encrypting all the k pixels, the encrypted image generated is C.

3.2. Data Embedding

In this step, initially, the data hider uses the data hiding key (Dh_k), to encrypt the additional data (D) to obtain $E(D)$. Here (also refer to Section 3.5), it is assumed that the number of bits in $E(D)$ is equal to the number of pixels in the encrypted image C. Before embedding, the data hider generates a separate zero matrix M^0 of size $L \times B$, where all the elements are zero. The data hider uses Equation (1) and the public key to encrypt M^0, to obtain the encrypted matrix C^0. Thus, after encryption the size of matrix C^0, is the same as the size of the encrypted image C. It can be noted that each element of the matrix C^0 is distinct but of the same size, by the property of semantic security (Equation (5)) inherent in Pallier cryptosystem.

In the next step, each bit of $E(D)$ is embedded into the corresponding C_k of the encrypted image C. That is, each encrypted pixel C_k is responsible for carrying one bit of $E(D)$. For embedding, the data hider has to pad kth encrypted value of "0" (i.e., C_k^0) from the matrix C^0, with the corresponding kth encrypted pixel (i.e., C_k) of C. This padding results into a matrix of $L \times B$ padded units (PUs). As each PU is composed of two encrypted values (C_k, C_k^0), $L \times B$ padding units make $2 \times L \times B$ encrypted values. This makes the size of the PU matrix equal to $2 \times L \times B$, which is double the size of the C i.e., $L \times B$. All the PUs constitute the encrypted image with additional data, i.e., the marked encrypted image C'.

Padding Procedure

In padding, four cases arise as depicted in Figure 5: two cases, if the bit of $E(D)$ is "0", and two cases, if the bit of $E(D)$ is "1". For embedding, we compare values C_k and C_k^0.

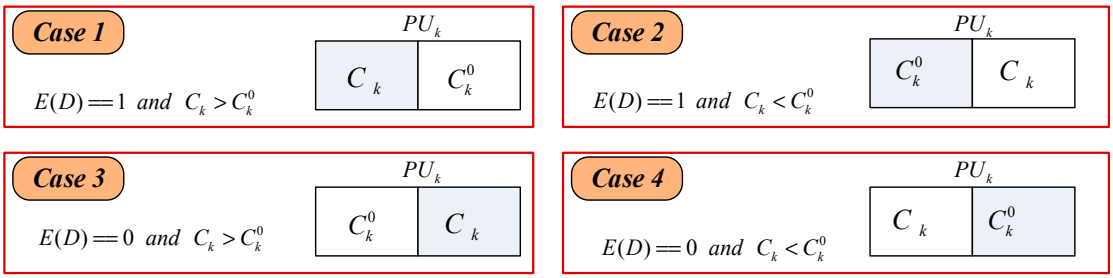

Figure 5. Four cases for padded unit (PU) construction.

When the bit of $E(D)$ is equal to 1, the padding is done such that the bigger value (between C_k and C_k^0) is at the first position in PU_k. Thus, if $C_k > C_k^0$, then C_k is placed first in the PU_k, as in Figure 5 (Case 1). If $C_k < C_k^0$, then C_k^0 is placed first in the PU_k, as in Figure 5 (Case 2).

When the bit of $E(D)$ is equal to 0, the padding is done in such a way that the bigger value (between C_k and C_k^0) will be at the second position in the PU_k. If $C_k > C_k^0$, then C_k is placed second in the PU_k, as in Figure 5 (Case 3). If $C_k < C_k^0$, then C_k^0 is placed second in the PU_k, as in Figure 5 (Case 4).

The step by step procedure for data embedding into the encrypted image can be understood from the following algorithm:

Let us assume C_k to be the encrypted image for the pixel k where $k \in [1, (L \times B)]$ and C_k^0 be the encrypted value of "0" for pixel k.

Step 1. With the help of data hiding key (Dh_k), bits of additional data (D) are encrypted in order to generate bits of encrypted additional data $E(D)$.

Step 2. If the bit of $E(D)$ to be embedded is 1:

If $C_k > C_k^0$ in the selected PU_k, then we rearrange the order in PU_k by appending C_k^0 after C_k.

(That is, the bigger value is first and the smaller value is second in the PU_k (Figure 5, Case 1).)

Otherwise, we append C_k after C_k^0.

(That is, the bigger value is first and the smaller value is second in the PU_k (Figure 5, Case 2).)

Step 3. If the bit of $E(D)$ to be embedded is 0:

If $C_k > C_k^0$ in the selected PU_k, then we rearrange the order in PU_k by appending C_k after C_k^0.

(That is, the smaller value is first and the bigger value is second in the PU_k (Figure 5, Case 3).)

Otherwise, we append C_k^0 after C_k.

(That is, the smaller value is first and the bigger value is second in the PU_k (Figure 5, Case 4).)

Step 4. After the encrypted image C has been embedded with the $E(D)$, the marked encrypted image C' is obtained with all the PU_k.

3.3. Data Extraction

After the receipt of the marked encrypted image C' on the receiver side, the embedded $E(D)$ is extracted using the data-hiding key (Dh_k). The step by step procedure for data extraction from C' is as follows:

Step 1. Scan the marked encrypted image C' in the same manner as used in the encryption and embedding phase, i.e., left to right and top to bottom (Figure 4).

Step 2. For each of the selected PU_k, Steps 3 and 4 are performed.

Step 3. If the first value of the pair (C_k^0, C_k) in the selected PU_k is bigger than the second value, then the embedded bit of $E(D)$ is "1".

In this case, "1" will be extracted.

Step 4. If the first value of the pair (C_k^0, C_k) in the selected PU_k is smaller than the second value, then the embedded bit of $E(D)$ is "0".

In this case, "0" will be extracted.

Step 5. After extracting all the bits, the encrypted additional data $E(D)$ is constituted.

Data hiding key (Dh_k) is used to regenerate the original additional data (D).

3.4. Image Recovery

In this step, if the receiver wants to recover the original image I, he must own the private key (λ). After receiving the marked encrypted image C', the receiver applies homomorphic multiplication on each pair of (C_k^0, C_k) in PU_k of C', to get the corresponding C_k, and then, decrypts each C_k to get kth pixel of the directly decrypted image DDI_k with private key (λ) using:

$$DDI_k = D[C_k] = \frac{L((C_k)^{\lambda} mod N^2)}{L(g^{\lambda} mod N^2)} mod \boldsymbol{N}, \tag{7}$$

where $D[\cdot]$ is the decryption function, λ is the private key and DDI is the directly decrypted image. In our scheme, DDI results in the completely recovered original image I, i.e., no post-processing on DDI is further required. Step-wise procedure used to recover the original image using the private key (λ) is as follows:

Step 1. Scan the marked encrypted image C' in the same manner as used in the encryption and embedding phase, i.e., left to right and top to bottom (Figure 4).

Step 2. For each selected PU_k.

Step 3. Apply homomorphic multiplication ($\times$) to each pair (C_k^0, C_k) in PU_k, to obtain C_k, such that $C_k = C_k^0 \times C_k$.

(The order of C_k^0 and C_k does not affect the result.)

Step 4. Each C_k is decrypted using the private key (λ) to give the corresponding DDI_k.

Step 5. DDI is obtained constituting all the DDI_k.

It can be noted that, in Step 3., the original values (unencrypted values) of C_k^0 and C_k are "0" and I_k respectively. When homomorphic multiplication is applied to (C_k^0, C_k), it means internally zero is added to the value I_k. Thus, we get encrypted value C_k (i.e., $C_k = (C_k^0 \times C_k)$ = encrypted value of $(0 + I_k)$).

3.5. Exemplifying Our Proposed Scheme

Figure 6 shows the working of the proposed scheme at the data-hider side. It is supported by the following example: our example shows the working for the 1st pixel value $I_1 = 65$, of the original image. Let the public key = $(1763,\ 94)$ and the private key = (840). Using the public key and the encryption function $E[\cdot]$, we get $E\ (65) = C_1 = (184,481)$ and $E(0) = C_1^0 = (304,186)$. Let the bit to be embedded be 1, i.e., $E(D) = 1$. For embedding, we compare C_1 and C_1^0. According to Step 2 of Section 3.2 (Figure 5, Case 2), when $C_1^0 > C_1$, we put C_1 after C_1^0 in PU_1.Thus, here, $PU_1 = (304,186;184,481)$, i.e., the bigger value is first and the smaller value is second. The receiver extracts the first bit of $E(D)$, by reading the order in PU_1. In $(304,186;184,481)$, the first value is bigger than the second value, so the embedded bit of $E(D)$ is 1. However, this bit will be further decrypted using data-hiding key (Dh_k) to get D Furthermore, the receiver having the private key (840) gets the first pixel value DDI_1, for the directly decrypted image DDI, from the $PU_1 = (304,186;184,481)$ by using homomorphic multiplication. This is done as: $\lambda(PU_1) = \lambda((C_1^0 \times C_1) mod N^2) = \lambda((304,186 \times 184,481) mod 1763^2) = \lambda(0 + 65) = 65$. Thus, the pixel value DDI_1 is same as the original pixel value $I_1 = 65$.

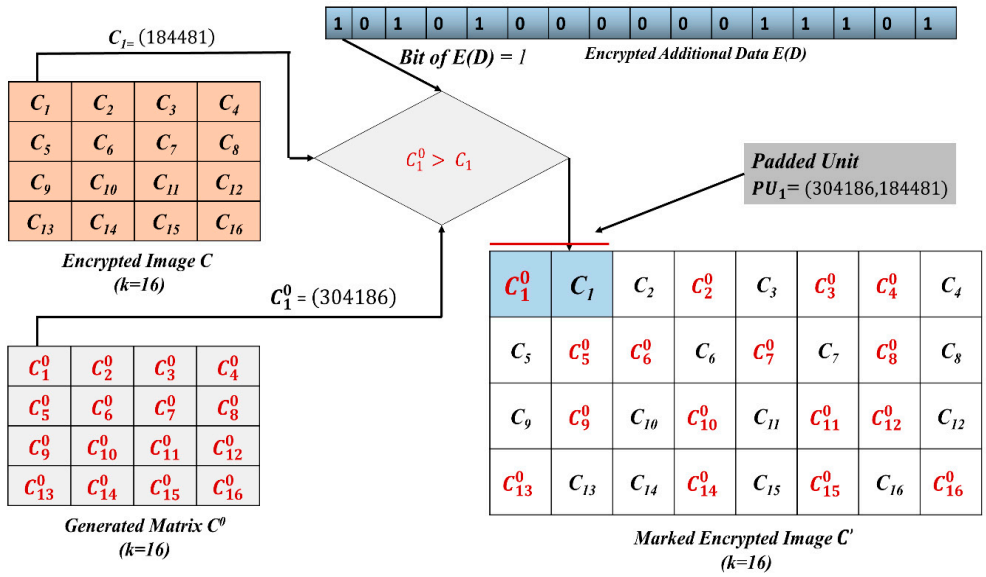

Figure 6. Data embedding process in our proposed scheme after getting an encrypted image C of size (4×4) from the content-owner. C_1^0 and C_1 are the values for index $k = 1$ of C^0 and C, respectively.

3.6. Proposed Scheme in Cloud Domain

There are a number of resource constrained devices based on cloud services; to show the process flow, involved hardware and software services we take an example of closed circuit television (CCTV) cameras installed on roads for monitoring the traffic against any traffic law violation. As shown in Figure 7, the workflow of the given scenario can be divided into following three phases:

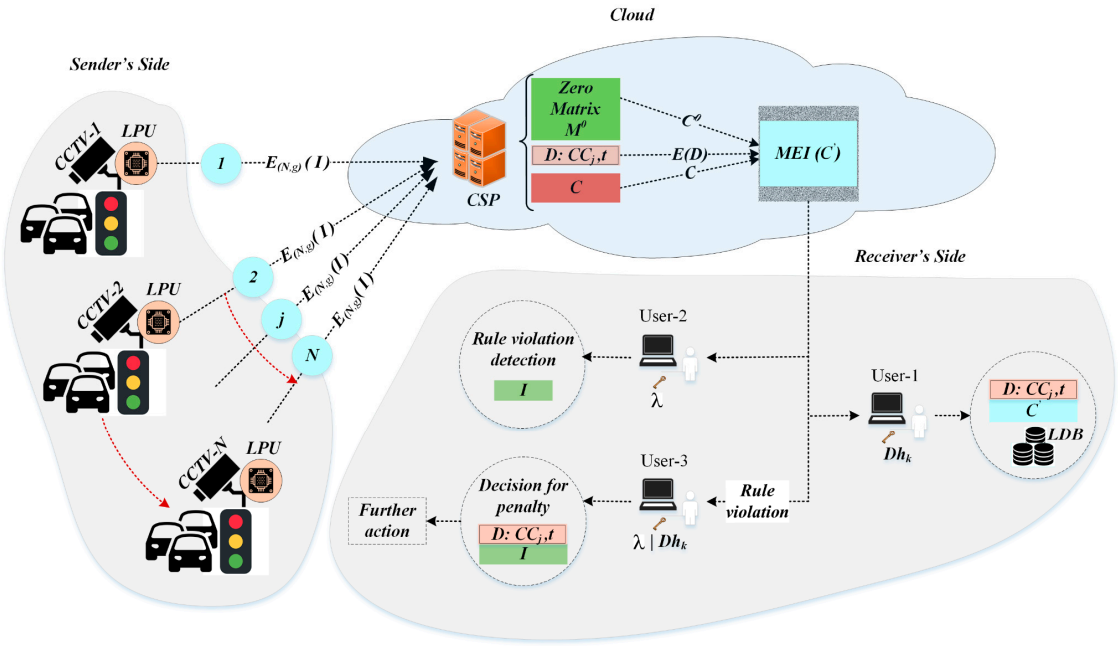

Figure 7. Of the proposed scheme in the cloud scenario.

(1) Image capturing and encryption at the sender's end
(2) Generating MEI over the cloud
(3) Authorized complete recovery of the D and I

(1) Image capturing and encryption: As shown in Figure 7, at the sender's end, there can be a number of CCTVs $(1, 2, 3, \ldots, j, \ldots, N)$, all of which are programmed to capture the images with a fixed time interval. To obtain the encrypted image C, the local processing unit (LPU) encrypts the image (I) captured from jth CCTV. This is done using public key $(N, \boldsymbol{g})$ and the encryption algorithm $E[\cdot]$. Once the image is encrypted, it is sent to the cloud service provider (CSP) acting as the data hider, for marking.

(2) Generating marked encrypted image (MEI): The data hider embeds the additional data D, such as the time (t) of the captured image or CCTV camera-id (CC_j). To make D secret, it is encrypted with the data hiding key (Dh_k) to obtain $E(D)$. A zero matrix M^0 equal to the size of the original image (I), is encrypted with public key $(N, \boldsymbol{g})$, to obtain C^0. C^0 is used, to embed $E(D)$ into the encrypted image C, to obtain C' as MEI.

(3) Authorized access and recovery: There are three type of end users with different access policies. The first category of users hold the data-hiding key (Dh_k) These users are authorized to access additional data (e.g., the camera-id CC_j, capturing time t, etc.) from the MEI. This type of data can be used to categorize and store the MEI in a local data base (LDB). The second type of users hold private key (λ) for retrieving original image. This type of authorized users exploit the original image for detecting any traffic rule violation such as wrongful crossing, incorrect overtake, etc. The third type of users hold both data hiding (Dh_k) and private (λ) keys. This enables users to access both additional data and the original image. Thus, for any rule violation, the image can be crosschecked with the corresponding additional data (time t, camera-id CC_j, location, etc.) for validity. Reasonable penalties can be applied to the subjects flouting the rules.

4. Experimental Results

To evaluate our proposed scheme, we used miscellaneous dataset [26] of gray-scale images, each having size 512×512. For simulation purpose, MATLAB 2015 b was used. We measured our results on the basis of embedding rate in terms of bit per pixel (bpp), visual quality in terms of peak signal noise

ratio (PSNR) and structural similarity index matrix (SSIM). The embedding rate (ER) can be calculated using Equation (8) as follows:

$$Embeding\ Rate(ER) = \frac{Number\ of\ total\ embedded\ bits}{Number\ of\ total\ pixels\ of\ the\ image}. \tag{8}$$

To calculate PSNR, Equation (9) is used

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE}(dB), \tag{9}$$

where mean square error (MSE) is calculated as follows:

$$MSE = \frac{1}{L \times B} \sum_{l=0}^{L-1} \sum_{j=0}^{B-1} (I(i,j) - R(i,j))^2. \tag{10}$$

Structural similarity index matrix (SSIM) is a method to measure the structural similarity between the recovered and reference images, where the reference image is the original image. It is used as a measure of perceived degradation in structural information of the recovered image with respect to the original image. The range of SSIM is $[-1,\ 1]$, where 1 shows that the images are identical. For original image I and the recovered image R, SSIM is calculated as follows

$$\mathrm{SSIM(I,R)} = \frac{(2\mu_I\mu_R + c_1)(2\sigma_{IR} + c_2)}{(\mu_I^2 + \mu_R^2 + c_1)(\sigma_I^2 + \sigma_R^2 + c_2)}. \tag{11}$$

where μ_I, μ_R are the mean of images I and R, respectively, and σ_I^2, σ_R^2 are the variance of images I and R, respectively. σ_{IR} is the covariance of I and R, respectively. Here, $c_1 = (k_1L)^2$,$c_2 = (k_2L)^2$, where L is the dynamic range of values of pixel and $k_1 = 0.01$, $k_2 = 0.03$ by default.

4.1. Results Showing Independence of the Proposed Scheme for Different Images

From the selected database [26] we experimented on four standard gray-scale images, as depicted in Figure 8a–d (Lena, Baboon, Boat and Airplane, respectively) with size 512×512 each. The respective images when decrypted directly from the marked encrypted image are depicted in Figure 8e–h. The embedding rate is 1 for all the four images, using Equation (8).

In the proposed scheme, for each pixel. one padded unit (PU) is constructed and each PU is responsible to carry 1-bit of additional data. This results in embedding rate of 1-bit per pixel, i.e., 1 bpp as each pixel is responsible to carry one bit of additional data. PSNR for directly decrypted images, calculated using Equation (9), was found to be $+\infty$ dB for all four images. It implies that the recovery of all the four original images was complete. This also shows that no post-processing on the directly decrypted image was needed to completely recover the original image. Using Equation (11), SSIM value for the four images was 1. This means that the structural similarity index of the recovered images against the original images was the same. Values for PSNR and SSIM support that the perceptual quality of the directly decrypted image with respect to the original image was the same. Table 2 shows the results for the metrics embedding rate, PSNR, and SSIM for all four images. It is inferred that the embedding rate, PSNR and SSIM are independent of the texture of chosen images, as these metric values remained constant for the four distinct images. This implies that, for a complex image, the embedding rate will be the same as that for a smooth image for our proposed scheme.

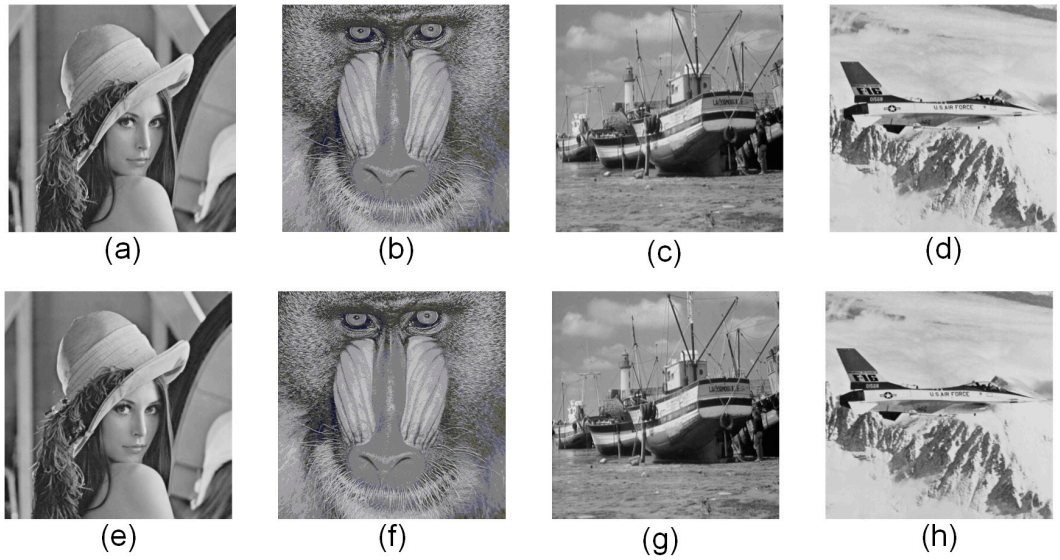

Figure 8. (**a**–**d**) The four original standard gray-scale images of size 512 × 512; and (**e**–**h**) the directly decrypted images (DDI) to check embedding rate, peak signal noise ratio (PSNR), structural similarity index matrix (SSIM) for different images in our scheme.

Table 2. Embedding rate, peak signal noise ratio (PSNR), structural similarity index matrix (SSIM) for four distinct standard test images (Figure 8a–d) in the proposed scheme.

Test Images	Embedding Rate (bpp)	PSNR	SSIM
Lena, Baboon, Boat and Airplane	1.0	$+\infty$	1

4.2. Comparative Analysis with Other Standard Schemes in RDHEI

To compare the proposed scheme with other schemes [9–12,16,24], we used the test image Lena (Figure 9a) from the database [26]. Figure 10 shows the comparison on the basis of PSNR of the directly decrypted image and their corresponding embedding rate.

Figure 9. The test image Lena: (**a**) Original; and (**b**) directly decrypted image.

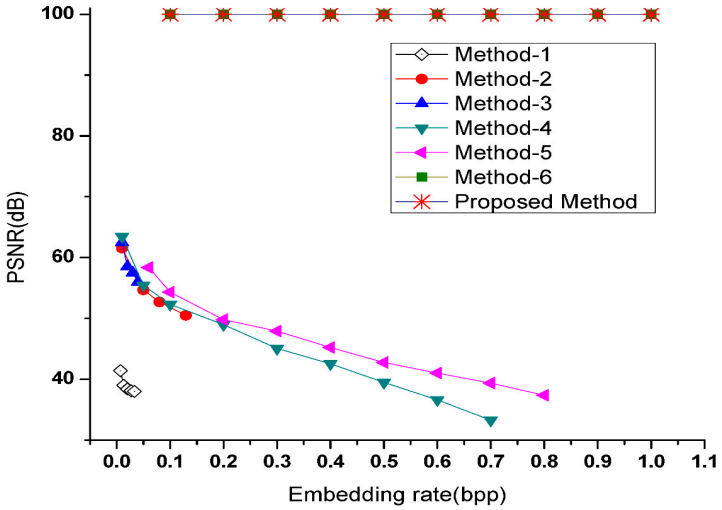

Figure 10. Performance comparison on the test image Lena for compared schemes Method-1 [9], Method-2 [10], Method-3 [12], Method-4 [11], Method-5 [16], Method-6 [24].

For Methods-1–5 [9–12,16], PSNR decreased with the increase in embedding rate. For Method-6 [24], PSNR was $+\infty$ dB and was independent of the embedding rate with maximum embedding rate of 1 bpp. It is notable that the maximum PSNR for any of the highest embedding rate for the directly decrypted image in Methods-1–5 [9–12,16] was less than 55.34 dB, showing that recovery of the directly decrypted image is incomplete without post-processing. However, in Method-6 [24] and the proposed scheme, the directly decrypted image is the same as the original image and no post-processing is required.

In the proposed scheme, the PSNR for the directly decrypted image (Figure 9b) was independent of the embedding rate and the maximum embedding rate is 1 bpp. The image quality of other compared schemes (except Method-6 [24]) was significantly less as compared to our proposed scheme when PSNR for the directly decrypted image was taken into account.

Table 3 shows a property-wise comparison of different schemes [9–12,16,24]. The maximum embedding rate for Zhang's scheme [9] was 0.033 bpp with PSNR = 38.0 dB, which decreased with the increase in the embedding rate. The maximum embedding rate for Zhang et al.'s scheme [12] was 0.04 bpp with PSNR = 55.34 dB. For Yin et al.'s scheme [10], maximum embedding rate is 0.1294 bpp with PSNR = 50.51 dB.

Table 3. Scheme-wise property comparison.

Schemes	Image Pre-Processing	Encryption	Receiver	Maximum Embedding Rate (bpp)	PSNR (dB) of Directly Image	Data Expansion
Zhang [9]	No	Stream cipher	Separable	0.033	38.0	No
Zhang et al. [12]	Yes	Stream cipher	Separable	0.04	55.34	No
Yin et al. [10]	No	Stream cipher	Separable	0.1294	50.51	No
Ma et al. [11]	Yes	Stream cipher	Separable	0.7	33.273	No
Cao et al. [16]	Yes	Stream cipher	Separable	0.8	37.375	No
Tai et al. [24]	Yes	Public key	Separable	1.0	$+\infty$	Yes
Proposed	No	Public key	Separable	1.0	$+\infty$	Yes

The maximum embedding rate for schemes of Ma et al. [11], Cao et al. [16] and Tai et al. [24] are 0.7 bpp, 0.8 bpp and 1.0 bpp with PSNR equal to 33.273 dB, 37.375 dB and $+\infty$ dB, respectively. For

our proposed scheme, the maximum embedding rate was 1 bpp with PSNR = $+\infty$ dB. Here, PSNR is $+\infty$ dB was irrespective of the embedding rate.

The schemes compared in Table 3 achieved complete recovery of image Lena (Figure 9a) at the receiver's side after post-processing. All schemes including the proposed scheme are separable. A stream cipher is used for encryption in the schemes in [9–12,16], while public key cryptography is used for encryption in the scheme in [24] and the proposed scheme. The public-key cryptosystem used in our scheme is responsible for the inevitable data expansion caused due to homomorphic properties unlike in the schemes in [9–12,16]. However, with this disadvantage comes an advantage that the property of homomorphic addition inherent in the Paillier cryptosystem used in our scheme makes it suitable for privacy-preserving environment needed for cloud computing.

Sharing the similarity of using Paillier public-key cryptosystem with Tai's scheme [24], our scheme is also a separable reversible data hiding scheme, where the data hiding key is used by the receiver to extract the additional data from the marked encrypted image. Although image pre-processing is not done in the schemes in [9,10], the quality of the directly decrypted image is lesser as compared to our scheme. In the scheme in [24], the image preprocessing step needed for data hiding is an unnecessary overhead for the content-owner. This overhead is gracefully transferred to the data hider's side using the benefits of cloud in our scheme, which enhances the efficiency in the resource-constrained environment on the content-owner's end because the size of the pre-processed image (in the scheme in [24]) to be encrypted and sent is reduced to half in our scheme (see Table 4).

Table 4. Comparative analysis of our scheme in terms of bit-size for image ($L \times$ B) with Tai et al. [24].

Schemes	Size (in bits)		
	Content-Owner	**Data-Hider**	**Receiver**
Tai et al. [24]	$2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$	$2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$	$2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$
Proposed	$1 \times L \times B(\lfloor \log_2 N^2 \rfloor + 1)$	$2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$	$2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$

In Table 4, a test image of size $L \times$ B is taken for comparison between our proposed scheme and the scheme in [24]. It is inferred that the proposed scheme has an edge over the scheme in [24] as we reduced the image preprocessing step on the sender's side, which is an additional step in the scheme in [24]. The size of the encrypted image is $L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$ bits in our proposed method. For the same image, the size of the encrypted image in the scheme in [24] is $2 \times L \times B \times (\lfloor \log_2 N^2 \rfloor + 1)$ bits. This is because the scheme in [24] involves a preprocessing step of dividing the original image into two parts and then encrypting it on the content-owner side. This can be also be seen as a reduction in the encryption cost to half on the content-owner side because, in the scheme in [24], the original image pixel is divided into two parts, each having the size of the original pixel. Thus, for encryption of a single image of size $L \times B$, the scheme in [24] has to encrypt $L \times B$ two times, i.e., for the two preprocessed parts of the same image. In the case of our scheme, for encrypting the image of size $L \times B$, encryption has to be done only once. It is a boon for those resource constrained devices on the sender side which entrust cloud for storage and processing. On the cloud side, the processing power is extremely large, thus attaching an equal payload to embed the additional data and its transmission is a trivial task.

5. Conclusions

In this work, an efficient separable reversible data hiding framework in encrypted images is proposed where Paillier cryptosystem is adequately used to preserve the privacy of the content in the cloud environment. Here, data extraction is accomplished in a separable manner, where the embedded additional data can be extracted using data hiding key and an independent complete image recovery is achieved using the private key.

In addition, the cost for encryption is reduced on the content owner's side (using limited processing and memory) with respect to Tai et al.'s scheme [24] by gracefully transferring the pre-processing step to the data hider side using cloud. Moreover, this proposed scheme exploits vast storage and memory resources available on the cloud. The proposed scheme was well explained with a real life application over cloud. Future works may include improving the efficiency of our scheme using new techniques in RDHEI.

Author Contributions: Conceptualization, A.N.K.; Data curation, A.N.K., M.I.N., A.M. and M.A.H.; Formal analysis, A.N.K. and A.M.; Funding acquisition, M.A.H.; Investigation, R.A.M.; Methodology, A.N.K.; Supervision, M.Y.F.; Validation, A.N.K.; Writing—original draft, A.N.K.; Writing—review & editing, A.N.K.

Funding: This research was funded by TEQIP-III of REC Ambedkar Nagar, grant number-TEQIP 3-RECABN and the APC was funded by TEQIP-III of REC Ambedkar Nagar.

Acknowledgments: The support of TEQIP-III of REC Ambedkar Nagar for this work is highly acknowledged.

Conflicts of Interest: The authors declare no conflict of interests.

References

1. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
2. Celik, M.; Sharma, G.; Tekalp, A.M.; Saber, E. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process.* **2005**, *14*, 253–266. [CrossRef] [PubMed]
3. Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
4. Malik, A.; Wang, H.; Wu, H.; Abdullahi, S.M. Reversible Data Hiding with Multiple Data for Multiple Users in an Encrypted Image. *Int. J. Digit. Crime Forensics* **2019**, *11*, 46–61. [CrossRef]
5. Shi, Y.-Q.; Li, X.; Zhang, X.; Ma, B.; Wu, H. Reversible Data Hiding: Advances in the Past Two Decades. *IEEE Access* **2016**, *4*, 1. [CrossRef]
6. Puech, W.; Chaumont, M.; Strauss, O. A reversible data hiding method for encrypted images. *Proc. SPIE* **2008**, *6819*, 68191E.
7. Zhang, X. Reversible Data Hiding in Encrypted Image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
8. Hong, W.; Chen, T.-S.; Wu, H.-Y. An Improved Reversible Data Hiding in Encrypted Images Using Side Match. *IEEE Signal Process. Lett.* **2012**, *19*, 199–202. [CrossRef]
9. Zhang, X. Separable Reversible Data Hiding in Encrypted Image. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 826–832. [CrossRef]
10. Yin, Z.; Luo, B.; Hong, W. Separable and Error-Free Reversible Data Hiding in Encrypted Image with High Payload. *Sci. World J.* **2014**, *2014*, 1–8. [CrossRef]
11. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 553–562. [CrossRef]
12. Zhang, W.; Ma, K.; Yu, N. Reversibility improved data hiding in encrypted images. *Signal Process.* **2014**, *94*, 118–127. [CrossRef]
13. Qian, Z.; Han, X.; Zhang, X. Separable Reversible Data hiding in Encrypted Images by n-nary Histogram Modification. In Proceedings of the 3rd International Conference on Multimedia Technology (ICMT 2013), Guangzhou, China, 29 November–1 December 2013; pp. 201–204.
14. Zhang, X.; Qian, Z.; Feng, G.; Ren, Y. Efficient reversible data hiding in encrypted images. *J. Vis. Commun. Image Represent.* **2014**, *25*, 322–328. [CrossRef]
15. Zheng, S.; Li, D.; Hu, D.; Ye, D.; Wang, L.; Wang, J. Lossless data hiding algorithm for encrypted images with high capacity. *Multimed. Tools Appl.* **2016**, *75*, 13765–13778. [CrossRef]
16. Cao, X.; Du, L.; Wei, X.; Meng, D.; Guo, X. High Capacity Reversible Data Hiding in Encrypted Images by Patch-Level Sparse Representation. *IEEE Trans. Cybern.* **2016**, *46*, 1. [CrossRef] [PubMed]
17. Kuribayashi, M.; Tanaka, H. Fingerprinting protocol for images based on additive homomorphic property. *IEEE Trans. Image Process.* **2005**, *14*, 2129–2139. [CrossRef] [PubMed]
18. Wu, H.-T.; Cheung, Y.-M.; Huang, J. Reversible data hiding in Paillier cryptosystem. *J. Vis. Commun. Image Represent.* **2016**, *40*, 765–771. [CrossRef]

19. Chen, Y.-C.; Shiu, C.-W.; Horng, G. Encrypted signal-based reversible data hiding with public key cryptosystem. *J. Vis. Commun. Image Represent.* **2014**, *25*, 1164–1170. [CrossRef]
20. Shiu, C.-W.; Chen, Y.-C.; Hong, W. Encrypted image-based reversible data hiding with public key cryptography from difference expansion. *Signal Process. Image Commun.* **2015**, *39*, 226–233. [CrossRef]
21. Li, M.; Xiao, D.; Zhang, Y.; Nan, H. Reversible data hiding in encrypted images using cross division and additive homomorphism. *Signal Process. Image Commun.* **2015**, *39*, 234–248. [CrossRef]
22. Liu, W.-L.; Leng, H.-S.; Huang, C.-K.; Chen, D.-C. A Block-Based Division Reversible Data Hiding Method in Encrypted Images. *Symmetry* **2017**, *9*, 308. [CrossRef]
23. Zhang, X.; Long, J.; Wang, Z.; Cheng, H.; Wang, J. Lossless and Reversible Data Hiding in Encrypted Images with Public Key Cryptography. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1. [CrossRef]
24. Tai, W.-L.; Chang, Y.-F. Separable Reversible Data Hiding in Encrypted Signals with Public Key Cryptography. *Symmetry* **2018**, *10*, 23. [CrossRef]
25. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of the Advances in Cryptology—EUROCRYPT '99, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
26. CVG-UGR—Image Database. Available online: http://decsai.ugr.es/cvg/dbimagenes/g512.php (accessed on 21 April 2019).