ISSN:2320-0790

An International Journal of Advanced Computer Technology

Reversible Data Hiding In Encrypted Images Using Improved Encryption Technique

Mr. Balika J. Chelliah[1], Dr. J. Jagadeesan[2], Sneha Mathur[3], Saurav Dutta[4]

[1]Assistant Professor, [2] Head of the Department, [3] Research Scholar, [4] Research Scholar,
SRM University, Chennai

Abstract: Recently more and more attention is paid to reversible data hiding (RDH) in encrypted images, since it maintains the excellent property that the original cover can be losslessly recovered after embedded data is extracted while protecting the image content's confidentiality. All previous methods embed data by reversibly vacating room from the encrypted images, which may be some errors on data extraction and/or image restoration. In this paper we propose a different scheme which attains real reversibility by reserving room before encryption with a traditional RDH algorithm, and then encrypting the data and embedding the data in the encrypted image, which is encrypted using a new proposed algorithm. The proposed method can achieve real reversibility that is data extraction and image recoveries are free of any error.

Keywords: Image Encryption, Image encryption key, Stream Cipher, RC4 algorithm, Data Encryption Key , Self Embedding, Data Hiding Key , Image Decryption , Data Decryption

I. INTRODUCTION

Reversible data hiding in images is a technique by which the original cover can losslossely recovered after the embedded messages are extracted .This important technique is widely used in medical imaginary ,military imaginary and law forensics ,where no distortion of original cover is allowed .since first introduced , RDH has attract3ed considerable research interest.

In practical aspects, many RDH techniques have emerged in recent years. A more popular method is based on difference expansion (DE)[5],in which the difference of each pixel group is expanded and the least significant bit (LSBs)of the differences are all 0s and can be used for embedding messages.

With regards to provide confidentiality for images, encryption is an effective and popular mean as it converts the original and meaningful contents to incomprehensible one. Although few RDH techniques in encrypted images have been published yet, there are some promising applications if RDH can be applied to encrypted images. In[13]Hwang et al. advocated a reputation –based trust management scheme enhanced with data coloring(a way of embedding data into covers) and software water marking in which data encryption and coloring offers possibilities for up holdings the content owners privacy and data integrity. Obviously, the clouded service provider has no right to introduce permanent distortion during data coloring and encrypted data. Thus a data coloring technique based on encryption data is preferred. Suppose a medical image database is stored in data center and a server in the data center can embedded notations into an encrypted version of an encrypted image through a RDH techniques. With the notation the server can manage an image or verify its integrity without having the original contents and thus the patient privacy is protected. On the other hand, a doctor having the cryptographic key can decrypt and restore the image in a reversible manner for the purpose of further diagnosing. The methods mentioned above rely on spatial correlation of original image to extract data. That is, the encrypted image should be decrypted first before data extraction.

To separate the data extraction from image decryption, emptied out space for data embedding following the idea of compressing encrypted images, Compression of encrypted data can be formulated as source coding with side information at the decoder, in which the typical method is to generate the compressed data in lossless manner by exploiting the syndromes of parity-check matrix of channel codes. The method in [18] compressed the encrypted LSBs to vacate room for additional data by finding syndromes of a parity-check matrix, and the side information used at the receiver side is also the spatial correlation of decrypted images. All the three methods try to vacate room from the encrypted images directly. However, since the entropy of encrypted images has been maximized, these techniques can only achieve small payloads [16], [17] or generate marked image with poor quality for large payload [18] and all of them are subject to some error rates on data extraction and/or image restoration. Al- though the methods in can eliminate errors by error- correcting codes, the pure payloads will be further consumed.

In the present paper, we propose a novel method for RDH in encrypted images, for which we do not "vacate room after encryption" as done in , but "reserve room before encryption". In the proposed method, we first empty out room by embedding LSBs of some pixels into other pixels with a traditional RDH method and then encrypt the image, so the positions of these LSBs in the encrypted image can be used to embed data. Not only does the proposed method separate data extraction from image decryption but also achieves excellent performance in two different prospects:

• Real reversibility is realized, that is, data extraction and image recovery are free of any error.

• For given embedding rates, the PSNRs of decrypted image containing the embedded data are significantly improved; for the acceptable PSNR, the range of embedding rates is greatly enlarged.

In all methods, the encrypted 8-bit gray-scale images are generated by encrypting every bit-plane with a stream cipher. The method in [16] segments the encrypted image into a number of non overlapping blocks sized by ; each block is used to carry one additional bit. To do this, pixels in each block are pseudo-randomly divided into two sets and according to a data hiding key. If the additional bit to be embedded is 0, flip the 3 LSBs of each encrypted pixel in, otherwise flip the 3 encrypted LSBs of pixels in. For data extraction and image recovery, the receiver flips all the three LSBs of pixels in to form a new decrypted block, and flips all the three LSBs of pixels in to form another new block; one of them will be decrypted to the original block. Due to spatial correlation in natural images, original block is presumed to be much smoother than interfered block and embedded bit can be extracted correspondingly. However, there is a risk of defeat of bit extraction and image recovery when divided block is relatively small (e.g.,) or has much fine detailed textures.

II. EXISTING SYSTEM

Since losslessly vacating room from the encrypted images is relatively difficult and sometimes inefficient, why are we still so obsessed to find novel RDH techniques working directly for encrypted images? If we reverse the order of encryption and vacating room, i.e., reserving room prior to image encryption content owner side, the RDH tasks in encrypted images would be more natural and much easier which leads us to the novel framework, "reserving room before encryption (RRBE)". Below is the comparison of our proposed system and the existing system-

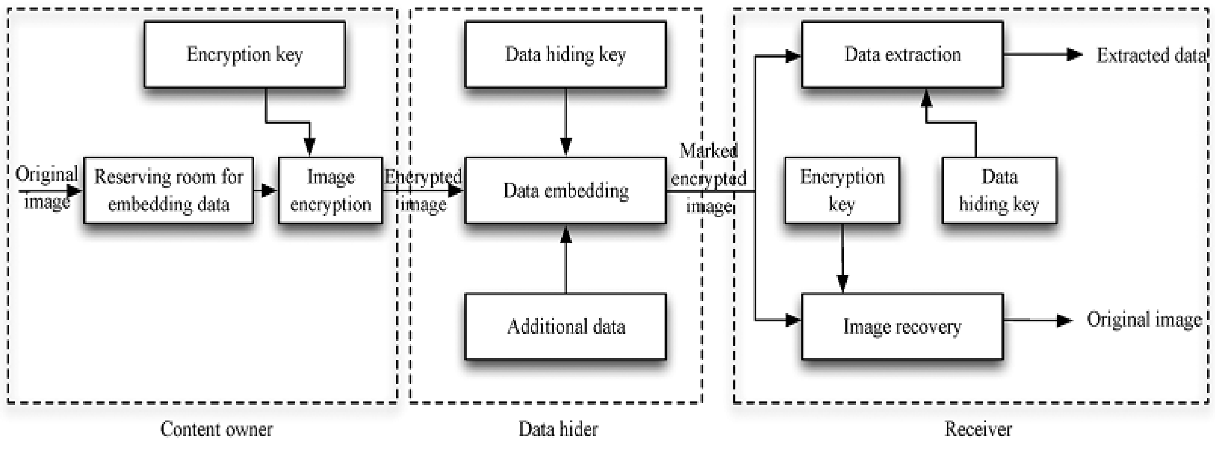

Fig. 1: a) Framework: "reserving room before encryption (RRBE)."

Actually, to construct the encrypted image, the first stage can be divided into three steps: image partition, self reversible embedding followed by image encryption. At the beginning, image partition step divides original image into two parts and ; then, the LSBs of are reversibly embedded into with a standard RDH algorithm so that LSBs of can be used for accommodating messages; at last, encrypt the rearranged image to generate its final version .

PROPOSED SYSTEM-

In the proposed system Fig 1(b), the content owner first reserves the enough space on original image into its encrypted version with encrypted key. Next the data is encrypted using RC4 algorithm and embedding process starts. The data embedding process in encrypted images is inherently for the Data hider needs to accommodate into the sparse space previous emptied out. The Data extraction and image recovery are identical to that of framework VRAE.

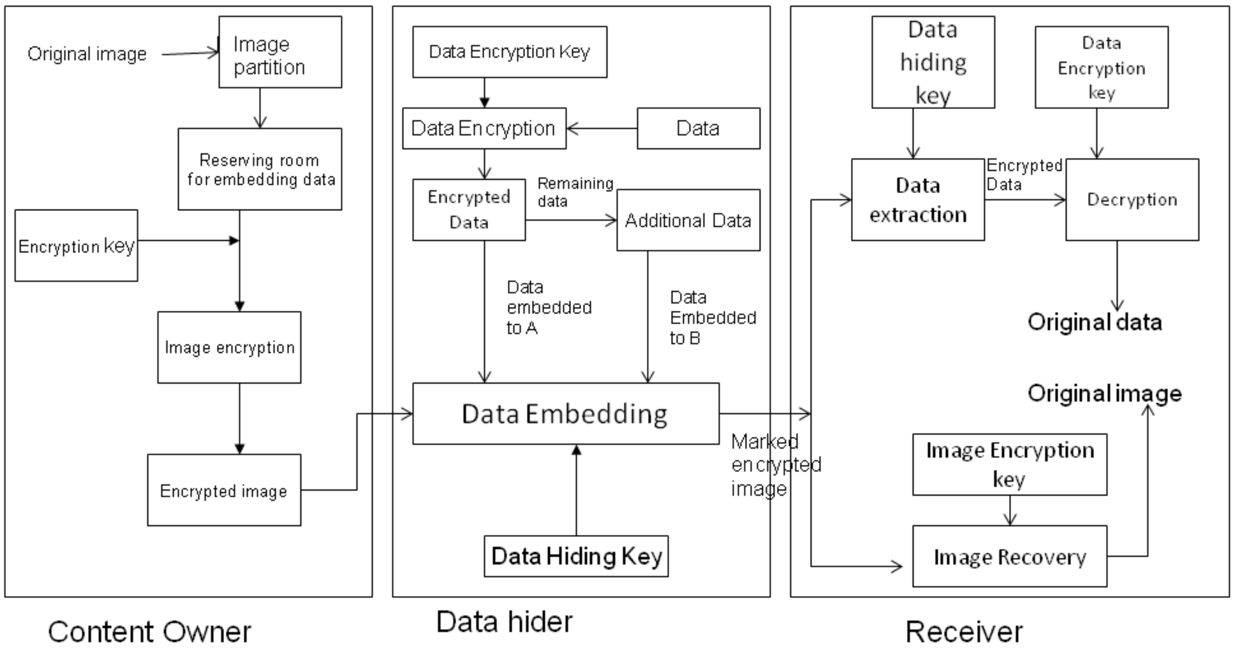

Fig. 1. b) Framework: "reserving room before encryption (RRBE). With advance features."

A. *Generation of encrypted Image*

The first stage can be divided into three steps: Image partition , self reversible embedding and image encryption. At the beginning image partition step divides original image into two parts A and B ; then the LSBs of A are reversibly embedded into B with standard RDH algorithm so that LSBs of A can be used for accommodating messages; Finally some bits are substituted in the image to generate an encrypted image.

1) Image Partition: The goal of image partition is to construct a smoother area B, on which standard RDH algorithm such as [10],[11] can achieve better performance. To do that , without loss of To do that, without loss of generality, assume the original image is an 8 bits gray-scale image with its size $M \; X \; N$ and pixels $C_{i,j}$ ϵ [0,255], $1 \le i \le M$, $1 \le j \le N$. First, the content owner extracts from the original image, along the rows, several overlapping blocks whose number is determined by the size of to-be-embedded messages, denoted by s . In detail, every block consists of m rows, where $m = [s/N]$, and the number of blocks can be computed through $n = M - m + 1$. An important point here is that each block is overlapped by pervious and/or sub sequential blocks along the rows.

For each block, define a function to measure its first-order smoothness

Higher relates to blocks which contain relatively more complex textures. The content owner, therefore, selects the particular block with the highest to be A, and puts it to the front of the image concatenated by the rest part B with fewer textured areas, as shown in Fig. 2

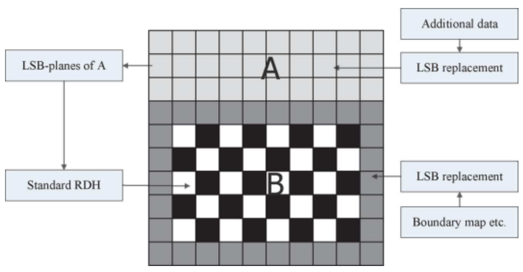

Fig. 2: Illustration of image partition and embedding process.

The above discussion implicitly relies on the fact that only single LSB-plane of A is recorded. It is straight forward that the content owner can also embed two or more LSB-planes of A into B, which leads to half, or more than half, reduction in size of A. However, the performance of A, in terms of PSNR, after data embedding in the second stage decreases significantly with growing bit planes exploited. Therefore, in this paper, we investigate situations that at most three LSB-planes of A are employed and determine the number of bit-plane with regard to different payloads experimentally in the next section.

2. Self-Reversible Embedding: The goal of self reversible embedding is to embed the LSB-planes of A into B by employing traditional RDH algorithms. For illustration, we simplify the method in [10] to demonstrate the process of self-embedding. Note that his step does not rely on any specific RDH algorithm. Pixels in the rest of image B are first categorized into two sets: white pixels with its indices *i* and *j* satisfying $(i+j) \bmod 2 = 0$ and black pixels whose indices meet $(i+j) \bmod 2 = 1$, as shown in figure. Then, each white pixel, $B_{i,j}$ is estimated by the interpolation value obtained with the four black pixels surrounding it as follows

$$B_{i,j} = w_1 B_{i-1,j} + w_2 B_{i+1,j} + w_3 B_{i,j-1} + w_4 B_{i,j+1}, \qquad (2)$$

Where the weight w1, $1 \leq i \leq 4$, is determined by the same method as proposed in [l0]. The estimating error is calculated via $e_{ij} = B_{i,j} - B`_{i,j}$ and then some data can be embedded into the estimating error sequence with histogram shift, which will be described later. After that, we further calculate the estimating errors of black pixels with the help of surrounding white pixels that may have been modified. Then another estimating error sequence is generated which can accommodate messages as well. Furthermore, we can also implement multilayer embedding scheme by considering the modified B as "original" one when needed. In summary, to exploit all pixels of B, two estimating error sequences are constructed for embedding messages in every single-layer embedding process.

By bidirectional histogram shift, some messages can be embedded on each error sequence. That is, first divide the histogram of estimating errors into two parts, i.e., the left part and the right part, and search for the highest point in each part, denoted by *SM* and *TM*, respectively. For typical images, *SM* =-1 and *TM* = 0. Furthermore, search for the zero point in each part, denoted by *SN* and *TN*. To embed messages into positions with an estimating error that is equal to *TM*, shift all error values between TM+1 and TN-1 with one step toward right, and then, we can represent the bit 0 with *TM* and the bit 1 with TM + 1. The embedding process in the left part is similar except that the shifting direction is left, and the shift is realized by subtracting 1 from the corresponding pixel values.
Suppose we should implement the embedding scheme *x-1* times to accommodate additional data. In the previous *x* - 1 single-layer embedding rounds, peak points of two error sequences are selected and utilized to embed messages as above mentioned. When it comes to the *x*th single-layer embedding, only a small portion of messages is left to be embedded, so it is unadvisable to accommodate such little data at the expense of shifting all error values between peak points and their corresponding zero points. To deal with this issue, we can either exploit only part of error sequences which has enough peak points to embed the remaining messages while leaving the rest error sequences unchanged, or find two proper points, denoted by *SP* and *TP*, whose sum is larger, however closest to, the size of remaining messages. By shifting error values between *SP*, *TP* and their corresponding zero points, messages can be embedded into *SP* and *TP* instead of peak points.

$$f = \sum_{u=2}^{m} \sum_{v=2}^{N-1} \left| C_{u,v} - \frac{C_{u-1,v} + C_{u+1,v} + C_{u,v-1} + C_{u,v+1}}{4} \right| .$$

Fig. 3 illustrates the idea of selecting proper Points.

Generally speaking, two solutions can gain significantly improvement in terms of PSNR when the length of data is relatively short, i.e., when *x* = 1. And the superiority of one solution over the other depends highly on statistics of natural image itself which will be discussed in the next section.

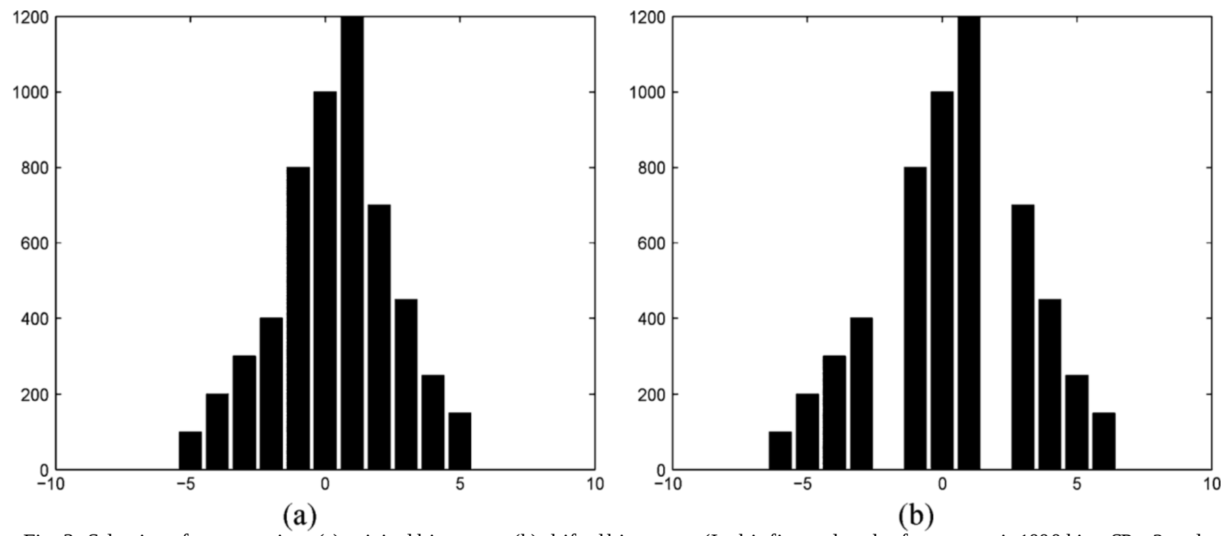
Fig. 3: Selection of proper points. (a) original histogram, (b) shifted histogram. (In this figure, length of messages is 1000 bits, SP=-2 and TP=2.)

The same with other RDH algorithms, overflow/underflow problem occurs when natural boundary pixels change from 255 to 256 or from 0 to -1. To avoid it, we only embed data into estimating error with its corresponding pixel valued from 1 to 254. However, ambiguities still arise when no boundary pixels are changed from I to 0 or from 254 to 255 during the embedding process. These created boundary pixels in the embedding process are defined as pseudo-boundary pixels. Hence, a boundary map is introduced to tell whether boundary pixels in marked image are natural or pseudo in extracting process. It is a binary sequence with bit "0" for natural boundary pixel, bit" I" for pseudo-boundary pixel. Since estimating errors of marginal area of B cannot be calculated via (2), to make the best use of B we choose its marginal area shown in Fig. 2 to place the boundary map, and use LSB replacement to embed it. The original LSBs of marginal area is assembled with messages, i.e., LSB-planes of A, and reversibly embedded into B. In most cases, even with a large embedding rate, the length of boundary map is very short; thus, the marginal area of B is enough to accommodate it. Meanwhile, several parameters such as *SN, TN, SM, TM, SP, TP*, payloads embedded into the estimating errors of black pixels *Rb,* total embedding rounds *x*, start row *SR* and end row *ER* of A in original image, are embedded into marginal area in a similar way. These parameters play an important role in data extraction and image recovery process.

3) Image Encryption:

After rearranged self-embedded image, denoted by X, is generated, we can encrypts X to construct the encrypted image, denoted by E. With a stream cipher, the encryption version of X is easily obtained. For example, a gray value X_{ij} ranging from 0 to 255 can be represented by 8 bits, $X_{i,j}(0)$, $X_{i,j}(1)$, ... , $X_{i,j}(7)$, such that

$$X_{ij}(k) = [\frac{X_{ij}}{2^k}] \bmod 2, \qquad k = 0,1,....7.$$

The encrypted bits *Ei,j(k)* can be calculated through exclusive or operation

$E_{i,j}(k) = X_{i,j}(k) \oplus r_{i,j}(k)$,

where $r_{i,j}(k)$ is generated via a standard stream cipher determined by the encryption key. Later on whole in each pixels we substitute some additional bit generated by permutation method to increase the security. Finally, we embed 10 bits information into LSBs of first 10 pixels in encrypted version of A to tell data hider the number of rows and the number of bit-planes he can embed information into. Note that after image encryption, the data hider or a third party can not access the content of original image without the encryption key, thus privacy of the content owner being protected.

B. Data Hiding in Encrypted Image

There are again two steps in Data hiding i.e before hiding, the original data is encrypted using RC4 Encryption technique and then hide it.

1) Data encryption using RC4 algorithm –

RC4:

The RC4 is remarkably simple .A variable length key of from 1 to 256 bytes is used to initialize a 256 byte state vector S with element 0 through 255.For Encryption and Decryption a byte k is generated from s by selecting one of the 255 entries in a systematic fashion. As each value of ‘K’ is generated .The entries are once again permuted.

INITIALIZATION OF S:

To begin the entries of S are set equal to values from 0 to 255 in an ascending order that is S[0]=0,S[1]=1,s[255]=255. A temporary vector t is also created. If the length of key is 256 bytes then K is transferred to T otherwise mod of keylen.

```
/* Initialization */

For i=0 to 255 do

S[i] =i;

T[i] = K[I mod keylen];
```

Next we use T to produce initial permutation of S. This involves swapping S[i] according to a scheme of T[i].

```
/*Initialization Permutation*/

j=0;

For i=0 to 255 do

i= (j+t[i]+s[i]) mod 256;

Swap(s[i].s[j]);
```

STREAM GENEATION:

Once the S vector is initialized the input key is no longer used. Stream generator involves starting with S [0] and going through S [255], and for each S[i].swapping with another bytes .After S [255] is reached the process continues starting over again at S [0].

```
/*Stream Generation*/

i,j=0;

While (true)

i= (i+1) mod 256;

j= (j+s[i]) mod 256;

Swap (s[i].s[j]);

t=(s[i]+s[j])mod 256;

k=s[t];
```

To encrypt XOR the value of K with next byte of plaintext. To decrypt the value of K with next byte of cyphertext.

The generated cyphertext is denoted by M.

2) *Data Hiding in encrypted image-*

Once the data hider acquires the encrypted image E, he can embed encrypted data M into it, although he does not get access to the original image. The embedding process starts with locating the encrypted version of A, denoted by A_E. Since AE has been rearranged to the top of E, it is effortless for the data hider to read 10 bits information in LSBs of first 10 encrypted pixels. After knowing how many bit-planes and rows of pixels he can modify, the data hider simply adopts LSB replacement to substitute the available bit-planes with additional data m. Finally, the data hider sets a label following m to point out the end position of embedding process and further encrypts m according to the data hiding key to formulate marked encrypted image denoted by E`. Anyone who does not possess the data hiding key could not extract the additional data.

C. Data Extraction and Image Recovery

Since data extraction is completely independent from image decryption, the order of them implies two different practical applications.

Case 1: *Extracting Data From Encrypted Images:*

To manage and update personal information of images which are encrypted for protecting clients' privacy, an inferior database manager may only get access to the data hiding key and have to manipulate data in encrypted demain. The order of data extraction before image decryption guarantees the feasibility of our work in this case.

When the database manager gets the data hiding key, he can decrypt the LSB-planes of A_E and extract the additional data m by directly reading the decrypted version. When requesting for updating information of encrypted images, the database manager, then, updates information through LSB replacement and encrypts updated information according to the data hiding key all over again. As the whole process is entirely operated on encrypted domain, it avoids the leakage of original content.

Case 2: *Extracting Data From Decrypted Images:*

In Case 1, both embedding and extraction of the data are manipulated in encrypted domain. On the other hand, there is a different situation that the user wants to decrypt the image first and extracts the data from the decrypted image when it is needed. The following example is an application for such scenario. Assume Alice out- sourced her images to a cloud server, and the images are encrypted to protect their contents. Into the encrypted images, the cloud server marks the images by embedding some notation, including the identity of the images' owner, the identity of the cloud server and time stamps, to manage the encrypted images. Note that the cloud server has no right to do any permanent damage to the images. Now an authorized user, Bob who has been shared the encryption key and the data hiding key, downloaded and decrypted the images. Bob hoped to get marked decrypted images, i.e., decrypted images still including the notation, which can be used to trace the source and history of the data. The order of image decryption before/without data extraction is perfectly suitable for this case. Next, we describe how to generate a marked decrypted image.

a) *Generating the Marked Decrypted Image:* To form the marked decrypted image X" which is made up of A" and B", the content owner should do following two steps.

- Step 1 : With the encryption key, the content owner decrypts the image except the LSB-planes of A_E. The decrypted version of E' containing the embedded data can be-calculated by

$X''_{i,j}(k) = E''_{i,j}(k)\, r_{i,j}(k)$

And

$$X''_{i,j} = \sum_{k=0}^{7} X''i.j \; x2^{k}$$

where $E'_{i,j}(k)$ and $X'_{i,j}(k)$ are the binary bits of $E'_{i,j}$ and $X''_{i,j}$ obtained via (3) respectively.

- Step 2 : Extract *SR* and *ER* in marginal area of B". By rearranging *A"* and B" to its original state, the plain image containing embedded data is obtained.

As can be seen, the marked decrypted image X" is identical to rearranged X except LSB-planes of A. At the meantime, it keeps perceptual transparency compared with original image C. More specifically, the distortion is introduced via two separate ways: the embedding process by modifying the LSB-planes of A and self-reversible embedding process by embedding LSB- planes of A into B. The first part distortion is well controlled via exploiting the LSB-planes of A only and the second part can benefit from excellent performance of current RDH techniques.

b) *Data Extraction and Image Restoration:*

After generating the marked decrypted image, the content owner can further extract the data and recover original image. The process is essentially similar to that of traditional RDH methods [10], [11].

The following outlines the specific steps:

- *Step 1.* Record and decrypt the LSB-planes of A" according to the data hiding key; extract the data until the end label is reached.
- *Step 2:* Extract *SN, TN, SM, TM, SP, TP, Ri; x* and boundary map from the LSB of marginal area of B". Then, scan B" to undertake the following steps.
- *Step 3:* If *Rb* is equal to 0, which means no black pixels participate in embedding process, go to Step 5.
- *Step 4:* Calculate estimating errors $e'_{i,j}$ of the black pixels $B''_{i,j}$. If $B''_{i,j}$ belongs to [1, 254], recover the estimating error and original pixel value in a reverse order and extract embedded bits when $e'_{i,j}$ is equal to *SN, TM(or TP), SM* (or *TP)* and *TN.* Else, if $B''_{i,j} \in \{0, 255\}$, refer to the corresponding bit *b* in boundary map. If *b* = 0, skip this one, else operate like $B''_{i,j} \in [1, 254]$. Repeat this step until the part of payload *Rb* is extracted. If extracted bits are LSBs of pixels in marginal area, restore them immediately.
- *Step 5:* Calculate estimating errors $e'_{i,j}$ of the white pixels $B''_{i,j}$ and extract embedded bits and recover white pixels in the same manner with Step 4. If extracted bits are LSBs of pixels in marginal area, restore them immediately.
- *Step 6 :* Continue doing Step 2 to Step 5 *x-1* rounds on *B"* and merge all extracted bits to form LSB-planes of A. Until now, we have perfectly recover B.
- *Step7:*Replace marked LSB-planes of A" with its original bits extracted from B" to get original cover image C.

We note that if the content owner wants to retrieve his image in Case 1, the procedures are exactly t same in Case 2. Thus, it is omitted in Case 1 for simplicity.

Once the Data is extracted it is a cypertext which is again decrypted using encryption key to generated the original Data.

III. CONCLUSION

Reversible data hiding in encrypted images is a new topic drawing attention because of the privacy-preserving requirements from cloud data management. Previous methods implement
RDH in encrypted images by vacating room after encryption, as opposed to which we proposed by reserving room before encryption. Thus the data hider can benefit from the extra space emptied out in previous stage to make data hiding process effortless. The proposed method can take advantage of all traditional RDH techniques for plain images and achieve excellent performance without loss of perfect secrecy. Furthermore, this novel method can achieve real reversibility, separate data extraction and greatly improvement on the quality of marked decrypted images.

IV. REFERENCES

[1] Kede Ma, Weiming Zhang, Xianfeng Zhao, Nenghai Yu, and Fenghua Li ," Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption" IEEE *Transaction on Information Forensics And Security , Vol. 8, NO. 3, March 2013.*

[2] Parag Kadam, Akash Kandhare, Mangesh Nawale and Mukesh Patil,"Separable Reversible Encrypted Data Hiding in Encrypted Image Using AES algorithm and Lossy Technique" *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME) February 21-22.*

[3] Rintu Jose and Gincy Abraham, "A Separable Reversible Data Hiding in Encrypted Image with Improved Performance" International *Conference on Microelectronics, Communication and Renewable Energy (ICMiCR-2013).*

[4] *Masaaki FUJIYOSHI "*Separable Reversible Data Hiding in Encrypted Image with Histogram Permutation" *Department of Information and Communication Systems, Tokyo Metropolitan University 6–6 Asahigaoka, Hino-shi, Tokyo 191–0065, Japan.*

[5] Xinpeng Zhang "Reversible Data Hiding in Encrypted Image" *IEEE Signal Processing Letters , Vol. 18, NO. 4, April 2011*

[6] Xinpeng Zhang "Separable Reversible Data Hiding in Encrypted Image" *IEEE Transaction on Information Forensics And Security, VOL. 7, NO. 2, APRIL 2012*

[7] S. Imaculate Rosaline and C. Rengarajaswamy " Reversible Data Hiding Technique for Stream Ciphered and Wavelet Compressed Image" *Proceedings of the International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME) February 21-22, 2013*

[8] X. L. Li, B. Yang, and T. Y. Zeng, "Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3524–3533, Dec. 2011.

[9] P. Tsai, Y. C. Hu, and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process.*, vol. 89, pp. 1129–1143, 2009.

[10] L. Luo *et al.*, "Reversible imagewatermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security,* vol. 5, no. 1, pp. 187–193, Mar. 2010

[11] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 989–999, Jul. 2009.

[12] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC, 1996.

[13] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," *IEEE Internet Comput.*, vol. 14, no. 5, pp. 14–22, Sep./Oct. 2010.

[14] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K.Ramchandran, "On compressing encrypted data," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.

[15] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted grayscale images," *IEEE Trans. Image Process.*, vol. 19, no. 4,pp. 1097–1102, Apr. 2010.

[16] X. Zhang, "Reversible data hiding in encrypted images," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.

[17] W. Hong, T. Chen, and H.Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Process. Lett.*, vol.19, no. 4, pp. 199–202, Apr. 2012.

[18] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security,* vol. 7, no. 2, pp. 826–832, Apr. 2012.