International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

A Final Paper on Reversible Data Hiding in Encrypted Images by Reserving Room before Encryption

Miss Shruti Vilas Tilwant[1], Prof. S.T. Khandare[2]

[1]Department of CSE, Babasaheb Naik College of Engg, Pusad

(stilwant@gmail.com)

[2]Department of CSE, Babasaheb Naik College of Engg, Pusad

(khandare.shailesh@rediffmail.com)

Abstract:

Reversible data hiding (RDH) in encrypted images has recently gotten a lot of interest since it has the wonderful virtue of allowing the original cover to be retrieved losslessly when embedded data is removed while maintaining the image content's anonymity. All prior approaches embed data by reversibly removing room from encrypted images, which might lead to data extractionand/or image restoration issues. Experiments reveal that with the same image quality as earlier methods, such as PSNR 40 db, this unique method can incorporate more than 10 times as large payloads.

Index Terms—Reversible data hiding, image encryption, privacy protection, histogram shift.

Introduction:

In pictures, reversible data hiding (RDH) is a technique that allows the original cover to be retrieved without loss after the embedded message has been extracted. This crucial approach is commonly utilized in medical imaging, military imaging, and criminal forensics, where no alteration of the original cover is permitted.RDH has sparked a lot of scientific interest when it was initially presented.

Many RDH approaches have emerged in recent years from a practical standpoint. Fridrich et al. [4] developed a broad RDH framework. Spare space can be preserved for embedding supplementary data by first extracting compressible aspects of the original cover and then losslessly compressing them. A more prevalent method is difference expansion (DE) [5, in which the difference of each pixel group is increased, for example, by 2, and the least significant bits (LSBs) of the difference are all-zero, allowing for message embedding.

Histogram shift (HS) [6] is another interesting RDH approach in which space is saved for data embedding by shifting the bins of the histogram of grey values. To attain greater performance, state-of-the-art algorithms [7]–[11] commonly integrated DE or HS with image residuals, such as anticipated errors.

Encryption [12] is a popular and successful method for providing image confidentiality since it changes the original and meaningful content into an unintelligible one. Although few RDH algorithms in

encrypted photos have been reported to far, if RDH can be used to encrypted images, there are several intriguing applications.

There have been some efforts at RDH in encrypted photos. Zhang split the encrypted image into many blocks in [16].Room for the embedded bit can be created by flipping three LSBs of the half of pixels in each block. Finding which part of one block has been flipped is the first step in data extraction and image recovery. The spatial correlation in the decrypted image can be used to carry out this operation.

Zhang [18] cleared space for data embedding to isolate data extraction from picture decryption, based on the principle of compressing encrypted images [14], [15]. Compression of encrypted data can be expressed as source coding with side information at the decoder [14], with the traditional way being to generate compressed data in a lossless manner by utilizing parity-check matrix of channel codes syndromes.

The method in [18] compressed the encrypted LSBs to make place for more data by locating parity-check matrix syndromes, and the side information used on the receiver side is also the spatial correlation of decrypted images.

All three approaches attempt to immediately vacate the room from the encrypted photos. However, because the entropy of encrypted images has been increased, these techniques can only achieve modest payloads [16], [17], or generate marked images with poor quality for high payloads [18], and all of them are prone to certain data extraction and/or image restoration error rates. Although the procedures in [16], [17] use error correcting codes to reduce errors, the pure payloads will still be consumed.

In this study, we suggest a new approach for RDH in encrypted images, in which we "reserve room before encryption" rather than "vacate room after encryption" as in [16]–[18].

In the suggested method, we first empty out room by using a typical RDH method to embed LSBs of some pixels into other pixels, then encrypt the image so that the positions of these LSBs in the encrypted image can be utilized to embed data. Not only does the suggested technique decouple data extraction from image decryption, but it also outperforms the competition in two areas:

• True reversibility is achieved, i.e., data extraction and image recovery are both error free.

• The PSNRs of decrypted images including embedded data are dramatically enhanced for given embedding rates, and the range of acceptable embedding rates is greatly expanded.

Previous Art:

As shown in Fig. 1, the methods proposed in [16]–[18] can be summarized as the "vacating room after encryption (VRAE)" framework (a).

A content owner encrypts the original image using a standard cipher and an encryption key in this framework. The content owner turns over the encrypted image to a data hider (e.g., a database manager), who can embed some auxiliary data into the encrypted image by losslessly vacating some space according to a data concealing key. Then, using the data concealing key, a receiver, such as the content owner or an

authorized third party, can extract the embedded data and, using the encryption key, restore the original image from the encrypted version.

The encrypted 8-bit gray-scale images are formed in all methods of [16]–[18] by encrypting every bit-plane with a stream cipher. The approach described in [16] divides the encrypted image into a number of no overlapping blocks, each of which is utilized to carry one extra bit. Pixels in each block are pseudo-randomly separated into two groups S1 and S2 based on a data hiding key to accomplish this. Flip the 3 LSBs of each encrypted pixel in S1 if the additional bit to be inserted is 0; otherwise flip the 3 encrypted LSBs of pixels in S2.

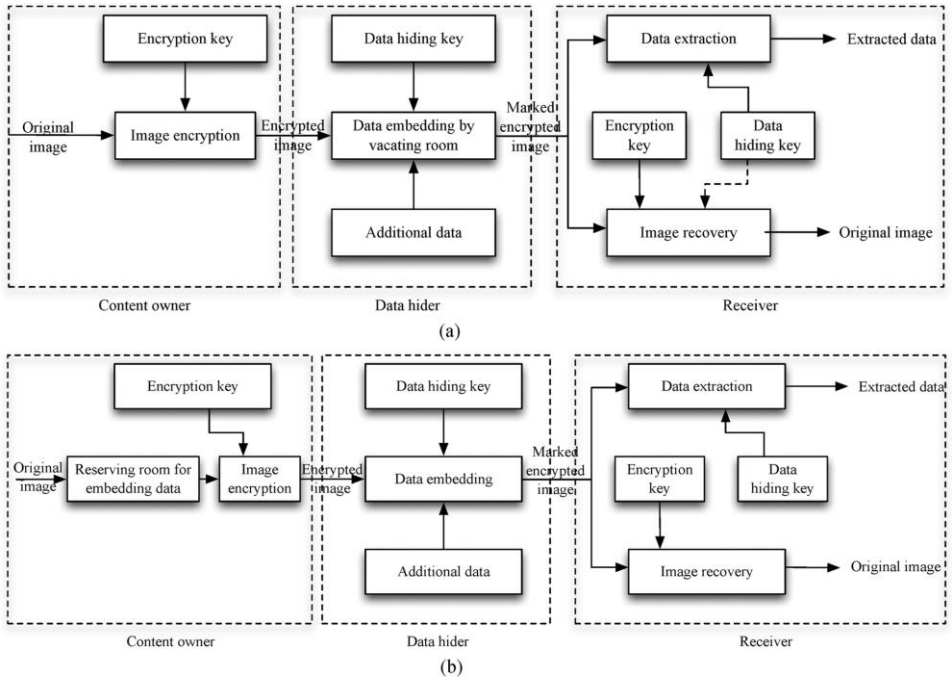
The receiver flips all three LSBs of pixels in S1 to generate a new encrypted block, and flips all three LSBs of pixels in S2 to form another new block for data extraction and picture recovery; one of these will be decrypted to the original block. Because natural images have spatial correlation, the original block is assumed to be considerably smoother than the interfering block, and the embedded bit can be retrieved accordingly. When the divided block is relatively small (e.g., a=8) or has a lot of fine-detailed textures, there is a chance of bit extraction and picture recovery failing.

Proposed Method:

Why are we still so fascinated with finding novel RDH approaches that work directly for encrypted photos, despite the fact that losslessly vacating room from encrypted images is rather difficult and sometimes inefficient? The RDH chores in encrypted images would be more natural and much easier if we reverse the sequence of encryption and vacating room, i.e., reserving room prior to image encryption at the content owner side, which leads to the unique framework, "reserving room before encryption (RRBE)".The content owner first reserves adequate space on the original image, as shown in Fig. 1(b), and then turn the image into its encrypted version using the encryption key.

The data embedding procedure in encrypted images is now intrinsically reversible, because the data hider merely needs to accommodate data into the previously vacated vacant space. Framework VRAE uses the same data extraction and picture recovery methods. Standard RDH algorithms are clearly the best operator for reserving room before encryption, and they can be simply implemented to Framework RRBE to provide higher performance than Framework VRAE approaches.

This is because, in this new framework, we apply the traditional approach of losslessly compressing redundant picture content (e.g., using superior RDH algorithms) before encrypting it for privacy reasons.



Then, using the Framework "RRBE," we develop a realistic technique that consists of four stages: picture production, data hiding in encrypted image, data extraction, and image recovery. It's worth noting that the proposed solution uses a typical RDH strategy for reserving.

A. Generation of Encrypted Image:

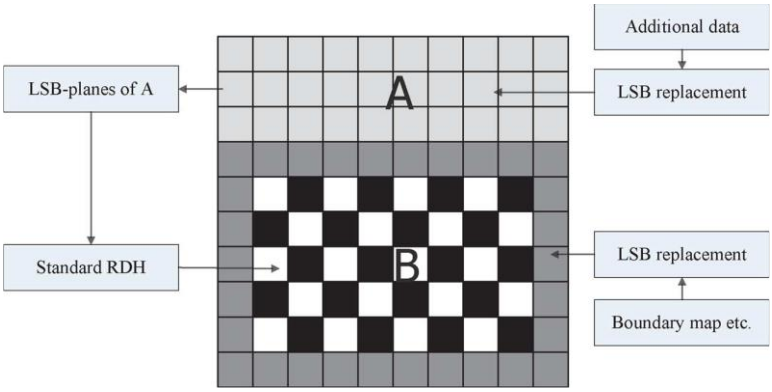
In fact, the initial stage of creating the encrypted image may be broken down into three steps: image partition, self-reversible embedding, and image encryption. The image partition step divides the original image into two parts A and B, after which the LSBs of A are reversibly embedded into B using a standard RDH algorithm so that the LSBs of A can be used for accommodating, and finally, the rearranged image is encrypted to generate its final version.

1) Image Partition:

As the operator for reserving room before encryption is a regular RDH approach, the purpose of image partition is to build a smoother region on which standard RDH algorithms like [10], [11] can perform better. Assume the original image is an 8-bit gray-scale image with a size of M * N and pixels to do so without losing generality. C_{ij} Ɛ [0-255].$1 \leq i \leq M$,$1 \leq j \leq N$.

First, the content owner takes multiple overlapping blocks from the original image along the rows, the number of which is defined by the size of the to-be-embedded messages, denoted by. Each block is made up of m rows, where m=[*l*/N] and the number of blocks may be calculated using [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

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



The term "higher" refers to blocks with more intricate textures. As a result, the content owner chooses the block with the highest to be and places it in front of the image, which is then concatenated by the rest of part B, which has fewer textured areas, as seen in Fig. 2.

The preceding discussion is predicated on the fact that just one LSB-plane of A has been recorded. It is simple for the content owner to embed two or more LSB-planes of A into B, resulting in a half- or more-than-half reduction in the size of A. However, after data embedding in the second step, the performance of A in terms of PSNR degrades dramatically as the number of bit-planes exploited increases. As a result, in this work, we analyze cases in which just three LSB-planes of A are used, and in the following section, we empirically calculate the number of bit-planes for various payloads.

2) Self-Reversible Embedding:

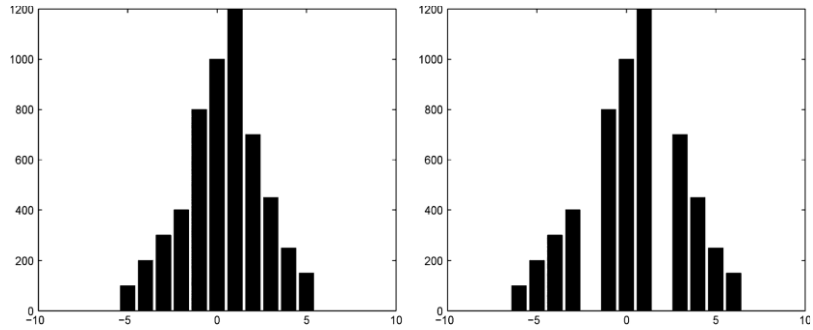
The purpose of self-reversible embedding is to use typical RDH techniques to embed the LSB-planes of A into B. To demonstrate the process of self-embedding, we simplify the method described in [10]. It's worth noting that this phase isn't dependent on any certain RDH method.

Pixels in the rest of image B are first categorized into two sets: white pixels with its indices i and j and satisfying (i+j) mod 2=0 and black pixels whose indices meet (i+j) mod 2=1, as shown in Fig. 2. Then, each white pixel, B_{ij} is estimated by the interpolation value obtained with the four black pixels surrounding it as follows

$$B'_{i,j} = w_1 B_{i-1,j} + w_2 B_{i+1,j} + w_3 B_{i,j-1} + w_4 B_{i,j+1},$$

Where the weight *Wi*, $1 \leq i \leq 4$ is calculated using the method provided in [10]. The estimating error is determined using the formula $e_{ij} = B_{ij} - B'_{ij}$, and then certain data can be embedded into the estimating error sequence via histogram shift, as discussed below. Following that, we calculate the estimating errors of black pixels using nearby white pixels that may have been updated. After that, a new estimated error sequence is created that can also accommodate messages. In addition, we can create a multilayer embedding strategy by treating the updated B as the "original" one as necessary. In summary, in every single-layer embedding procedure, two estimating error sequences are built for embedding messages in order to exploit all pixels of B.

Some messages can be placed on each error sequence using bidirectional histogram shift. To put it another way, divide the histogram of estimating errors into two sections, left and right, and look for the highest point in each portion, designated by LM and RM, respectively. LM =-1 and RM=0 for normal pictures. Additionally, look for the zero point in each section, which is symbolized by the letters LN and RN. To embed messages into places with an estimating error equal to RM, shift all error values one step to the right between RM+1 and RN-1, and then represent bit 0 with RM and bit 1 with RM+1.



The embedding method in the left part is similar, only the shifting direction is left, and the shift is accomplished by subtracting 1 from the pixel values.

Let's say we need to repeat the embedding procedure x times to accommodate more data. Peak points of two error sequences were picked and used to embed messages in the previous x-1 single-layer embedding rounds, as indicated above. Only a small part of messages remain to be embedded in the th single-layer embedding, hence it is unwise to fit such little data at the expense of shifting all error values between peak points and their corresponding zero points.

When natural border pixels shift from 255 to 256 or from 0 to -1, the overflow/underflow problem occurs, as it does with other RDH methods. To avoid this, we only embed data into estimating error with a pixel value of 1 to 254 as its corresponding pixel. When non boundary pixels are modified from 1 to 0 or from 254 to 255 during the embedding process, however, uncertainties persist. Pseudo-boundary pixels are the border pixels that are formed during the embedding process. As a result, in the extraction process, a boundary map is used to determine if border pixels in a marked image are natural or faux.

Bit "0" indicates a genuine border pixel, while bit "1" indicates a pseudo-boundary pixel. Because the estimating errors of B's marginal area cannot be estimated using (2), we choose the marginal area displayed in Fig. 2 to position the boundary map and embed it using LSB replacement. Messages, i.e. LSB-planes of,A and reversibly implanted into B, are combined with the original LSBs of marginal area.

Even with a high embedding rate, the length of the border map is usually quite short; consequently, the marginal area of B is sufficient to contain it. In the meantime, numerous parameters in the original image, such as LN RN LM RM LP RP, payloads embedded into the estimating errors of black pixels Rb, total embedding rounds x, start row SR, and end row ER of A, are similarly embedded into the marginal region. These variables are crucial in the data extraction and picture recovery procedure.

3. Image Encryption:

We can encrypts to construct the encrypted image, denoted by after the rearranged self-embedded image, designated by is formed. The encryption version of is easily acquired using a stream cypher. A grey value Xij spanning from 0 to 255, for example, can be represented by 8 bits, resulting :

$\mathbf{X}_{i,j}(k) =$ [xij/ 2^k] $k = 0, 1, \ldots, 7.$

The encrypted bits Eij (k) can be calculated through exclusive or operation

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

Where $r_{i,j}(k)$ is created using a conventional stream cypher based on the encryption key. Finally, in the encrypted version of A, we embed 10 bits information into the LSBs of the first 10 pixels to tell the data hider the number of rows and bit-planes he can embed information into. Note that without the encryption key, the data hider or a third party cannot access the content of the original image, ensuring that the content owner's privacy is preserved.

B) Data Hiding in Encrypted Image:

The data hider can insert data into the encrypted image E once he obtains it, but he does not have access to the original image. The process of embedding begins with locating the encrypted form of A, represented by Ae. The data hider can easily extract 10 bits information in LSBs of the first 10 encrypted pixels since Ae has been relocated to the top of its E.

The data hider simply uses LSB replacement to replace the available bit-planes with new data m after determining how many bit-planes and rows of pixels he can edit. Finally, the data hider adds a label to the end of m to indicate the conclusion of the embedding operation, and then encrypts m using the data hiding key to create the marked encrypted picture denoted by E'. The additional data could not be extracted by anyone who did not have the data hiding key.

C. Data Extraction and Image Recovery:

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

1) **Extracting Data From Encrypted Images**: An inexperienced database manager may only have access to the data hiding key and must modify data in the encrypted domain to manage and update personal information of images that are encrypted to safeguard clients' privacy. The order of data extraction before image decryption ensures that our job will be feasible in this circumstance.

When the database manager has the data hiding key, he can decrypt Ae's LSB-planes and read the decrypted version straight to extract the additional data m. When the database manager receives a request to change encrypted image information, the database management replaces the LSB and encrypts the revised information using the data hiding key all over again. Because the entire procedure takes place on an encrypted domain, there is no risk of original content being leaked.

2) **Extraction of Data from Decrypted Images**: In Case 1, both data embedding and extraction are done in an encrypted domain. On the other hand, there is a scenario in which the user wishes to decode the image first and then extract the data from the decrypted image as needed. An application for such a scenario can be found in the following example. Assume Alice has outsourced her photographs to a cloud server and that the images have been encrypted to secure the information they contain.

To manage the encrypted photographs, the cloud server embeds some information into the encrypted images, including the identity of the images' owner, the identity of the cloud server, and time stamps.

It's important to note that the cloud server has no authority to permanently alter the photos. Now the photos have been downloaded and decoded by Bob, an authorised user who has been given the encryption key and the data hiding key. Bob wanted to obtain tagged decrypted images, that is, decrypted images that retain the notation and may be used to trace the data's source and history.

PSNR results (dB)

embedding rate (bpp)		0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	1 LSB-plane	67.16	63.44	55.46	52.33	49.07	45.00	40.65	35.84
	2 LSB-planes	66.48	62.65	54.69	51.55	48.39	45.10	42.56	39.46
	3 LSB-planes	64.41	60.94	52.95	49.96	46.79	43.98	41.91	39.53
Airplane	1 LSB-plane	65.94	63.18	57.02	54.20	50.98	48.26	44.67	40.78
	2 LSB-planes	65.48	62.33	55.91	53.05	49.87	48.10	45.05	42.73
	3 LSB-planes	63.47	60.97	53.87	50.79	47.65	45.79	43.88	42.19
Barbara	1 LSB-plane	65.39	62.56	55.56	51.46	47.68	43.56	39.24	34.80
	2 LSB-planes	65.00	61.85	54.72	50.71	47.25	43.70	40.78	37.53
	3 LSB-planes	63.33	60.50	53.16	49.36	45.98	42.81	40.34	37.58
Baboon	1 LSB-plane	57.49	55.71	50.19	46.17	40.68	35.87	31.16	25.92
	2 LSB-planes	57.43	55.47	49.87	45.92	40.41	36.47	33.08	29.85
	3 LSB-planes	57.10	55.13	49.23	45.40	40.09	36.33	32.96	30.19
Peppers	1 LSB-plane	63.77	61.30	54.17	51.02	46.00	42.08	36.91	——
	2 LSB-planes	63.67	60.53	53.50	50.50	46.16	42.65	39.47	35.76
	3 LSB-planes	62.34	59.54	52.22	49.18	45.43	42.10	39.40	36.87
Boat	1 LSB-plane	67.22	64.13	56.75	52.62	49.10	45.21	41.24	35.99
	2 LSB-planes	66.72	63.26	55.75	51.71	48.40	44.98	42.46	39.98
	3 LSB-planes	64.57	61.34	53.73	50.02	46.71	43.81	41.70	39.46

D) Generating the Marked Decrypted Image:

The content owner should take the following two procedures to create the indicated decrypted picture X," which is made up of A" and B."

• Step 1: The content owner decrypts the image, except for the LSB-planes of Ae, using the encryption key. The embedded data can be computed using the decrypted version of E'.

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

And

$$\mathbf{X}''_{i,j} = \sum_{k=0}^{7} \mathbf{X}''_{i,j}(k) \times 2^k,$$

where $\mathbf{E}'_{i,j}(k)$ and $\mathbf{X}''_{i,j}(k)$ are the binary bits of $\mathbf{E}'_{i,j}$ and$\mathbf{X}''_{i,j}$, 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.

Discussion on Boundary Map:

In this research, the boundary map is utilised to discriminate between natural and faux boundary pixels, and the size of the boundary map is essential to the practical application of the suggested approach. The size of six standard pictures' boundary maps. In most circumstances, there is no requirement for a

boundary map. Even for Peppers, the highest size is 1741 bits (with a large embedding rate of 0.4 bpp and embedding technique 4 rounds), and the marginal area (512*4*4=8912 bits) is large enough to handle it.

EXPERIMENTS AND COMPARISONS:

To demonstrate the feasibility of the suggested technique, we use the standard image Lena (see Fig. 5(a)). The encrypted image with embedded messages is shown in Fig. 5(b), while the decrypted version with messages is shown in Fig. 5(c) (c).

The recovery version, shown in Fig. 5(d), is identical to the original image.

The proposed approach was compared to state-of-the-art works [16]–[18]. As previously stated in Section I, all of the methods in [16]–[18] may introduce mistakes in data extraction and/or image restoration, whereas the suggested method is error-free for all types of photographs.

PSNR is used to compare the quality of tagged decrypted pictures. The PSNR findings of distinct marked decrypted images under varied embedding rates are presented in Fig. 6. To be fair, we replace the error-correcting codes in [16], [17] with error-correcting codes. The pure payload of [16], [17] is decreased from [16], [17] to [16], [17] by inserting error-correcting codes cap by cap to $\mathrm{Cap}(1H(\rho))$, where $H(\rho)$ is the binary entropy function with error rate p. Take test image Baboon for instance.

. If each embedding block is sized of 8 * 8 with error rate 15.55% [16], then the pure payload is 1543 bits rather than 4096 bits. As for the method in [18], we only choose those results with a significantly high probability of successful data extraction and perfect image recovery to draw the curves.

It can be seen that our solution outperforms state-of-the-art RDH algorithms in encrypted images over a wide range of embedding rates and in all circumstances.

At the embedding rate range that the approaches in [16]–[18] can reach, the benefit in terms of PSNR is considerably high.

Another benefit of our method is that it allows for a considerably wider range of embedding rates for acceptable PSNRs. In fact, compared to the approaches in [16]–[18], the suggested method can incorporate more than 10 times as large payloads with the same acceptable PSNR (e.g., PSNR=40 dB), implying a high potential for practical applications.

Conclusion:

Because of the privacy-preserving requirements of cloud data management, reversible data hiding in encrypted images is a novel issue gaining traction. Previously, RDH in encrypted photos was implemented by vacating room after encryption, whereas we advocated reserving room before encryption. As a result, the data hider can take advantage of the excess space created in the previous stage to make the data concealing process much easier.

For plain photos, the suggested method can take use of all traditional RDH techniques and achieve good performance while maintaining full secrecy. Furthermore, this unique method can achieve true reversibility, distinct data extraction, and a significant improvement in the quality of encrypted photos that have been tagged.

References:

[1] T. Kalker and F.M.Willems, “Capacity bounds and code constructions
for reversible data-hiding,” in *Proc. 14th Int. Conf. Digital Signal Processing (DSP2002)*, 2002, pp. 71–76.

[2] W. Zhang, B. Chen, and N. Yu, "Capacity-approaching codes for reversible data hiding," in *Proc 13th Information Hiding (IH'2011), LNCS 6958*, 2011, pp. 255–269, Springer Verlag.
[3] W. Zhang, B. Chen, and N. Yu, "Improving various reversible data hiding schemes via optimal codes for binary covers," *IEEE Trans. Image Process.*, vol. 21, no. 6, pp. 2991–3003, Jun. 2012.
[4] J. Fridrich and M. Goljan, "Lossless data embedding for all image formats," in *Proc. SPIE Proc. Photonics West, Electronic Imaging, Security and Watermarking of Multimedia Contents*, San Jose, CA, USA, Jan. 2002, vol. 4675, pp. 572–583.
[5] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.
[6] Z. Ni, Y. Shi, N. Ansari, and S. Wei, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.
[7] D.M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 721–730, Mar. 2007.
[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.
[19] Miscelaneous Gray Level Images [Online]. Available: http://decsai. ugr.es/cvg/dbimagenes/g512.php