2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

A New Reversible Data Hiding Algorithm in the Encryption Domain

Shuang Yi, Yicong Zhou*, Chi-Man Pun, C. L. Philip Chen
Department of Computer and Information Science
University of Macau, Macau, China 999078
Email: * yicongzhou@umac.mo

***Abstract*—This paper introduces a new reversible data hiding algorithm in the encryption domain. It integrates data hiding into the image encryption process to achieve different level of access right and security. Computer simulations and comparisons demonstrate that the proposed algorithm can withstand the differential attack and outperforms other existing methods in terms of security and the message embedding capacity that is 52% larger than the state-of-the-art method in the best scenario. The marked decrypted images of our proposed method show the best visual quality according to the PSNR results.**

***Index Terms*—Reversible data hiding, encryption domain, differential attack.**

I. INTRODUCTION

Data hiding is a branch of information security, it conceals the secrete data in the digital media (i.e., digital images) and send it to the receiver in an imperceptible way. Reversible Data Hiding (RDH) is a type of data hiding techniques that aim to fully recover the original content. It is useful at some applications in military, medical science or law enforcement, where the original content cannot to be damaged [1]–[5]. Nowadays, people show interests in hiding the secrete data in encrypted images such that both the cover images and secrete data can be protected.

The first method of Reversible Data Hiding in Encrypted Images (RDHEI) was proposed by Zhang [6]. It encrypts the original image by the XOR operation, separates the encrypted image into several non-overlapping blocks with a size of $a \times a$, and embeds one bit secrete data to each block by flipping 3 least significant bits (LSBs) of half pixels randomly chosen from the block. The data extraction is based on spatial correlations of the nature image. This RDHEI method may suffer from the incorrect extraction of the secrete data from the non-smoothness areas in nature images when block size is relatively small (i.e., 8×8). Later, Hong et al. [7] improved the data extraction accuracy by modifying the smoothness evaluation function. It reduced 1.21% of the data extraction error rate when the block size is 8. These two methods need the information of the original image to extract the secrete data. Therefore, they are not suitable for the scenario that the owner of the original image and the hider of the secrete data are different persons with different privileges.

In medical applications, we may want to protect the medical images and patient private information (i.e., personal information or medical record) individually. We may use the RDHEI methods to encrypt medical images while embedding the patient private information into the encrypted medical images. Therefore, the receivers with different privileges may extract different contents without any error. Several separable RDHEI methods were proposed in recent years [8]–[10]. Zhang et al. proposed the separable RDHEI methods to compress the encrypted images to accommodate secrete data and thus its data extraction process can be separated [8], [9]. However, these methods can only achieve small payloads and suffer from error rate in the data extraction and image recovering [10]. To overcome these problems, Ma et al. [10] proposed a method by vacating rooms before image encryption using traditional RDH methods. Zhang et al. [11] applied a histogram shifting method on estimating errors to empty out room for data hiding.

Previous arts [6]–[10] encrypt the original image by changing only their pixel values using the XOR operations while keeping pixel positions unchanged. This may suffers from the differential attack.

In this paper, we propose a novel RDH algorithm in the encryption domain. It compresses the original image to reserve space for data hiding, encrypts the image using a substitution process, embeds the secrete data into the reserved space, scrambles the entire image using a permutation process to obtain the marked encrypted image. Compared with existing RDHEI methods, the proposed algorithm has larger embedding capacity. It is able to protect the original images and secrete data with a high level of security, and to extract both of them individually.

The rest of this paper is organized as follows: Section II will briefly review two relevant techniques which will be used in the new data hiding algorithm introduced in Section III. Some implementation issues will be discussed in Section IV. Section V will provide several simulation results and compare the proposed algorithm with several existing methods. Section VI will draw a conclusion.

II. BACKGROUND

In this section, the Secure Hash Algorithm [12] and the Logistic-Sine System [13] are briefly reviewed as background.

A. *Secure Hash Algorithm (SHA)*

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) [12]. They are sensitive to input

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

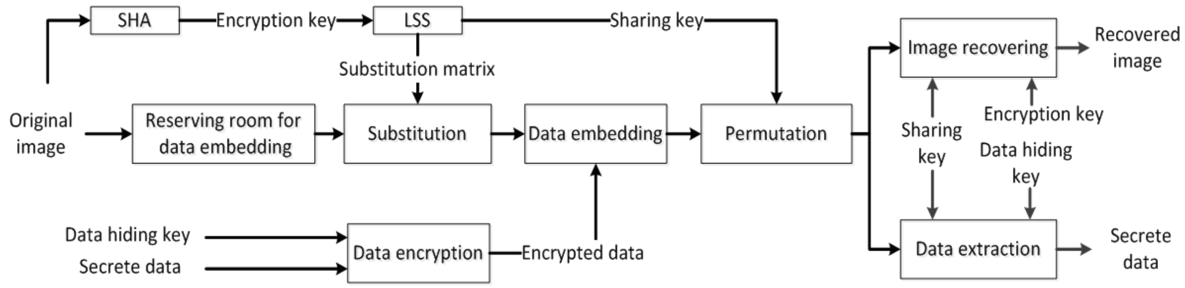


Fig. 1. The structure of RDHED

values, one bit change within the input signals result in a totally different output hash value. In this paper, we use it to initiate LSS.

B. Logistic-Sine System (LSS)

Logistic-Sine System [13] is a one dimension chaotic system defined by

$$X_{n+1} = (\gamma X_n(1 - X_n)) + (4 - \gamma)sin(\pi X_n/4)mod\ 1 \quad (1)$$

where $\gamma \in (0, 4]$.

LSS has a wide chaotic range and good chaotic behaviors. In this paper, we use it for image encryption.

III. Proposed Algorithm

In this section, we introduce the new reversible data hiding algorithm in the encryption domain, called RDHED. Its flowchart is shown in Fig. 1.

It is composed of three phases: reserving room, data embedding and data extraction/image recovering. The content owner first reserves rooms in the original image using the universal data embedding method [14], substitutes the image according to the encryption key. The data hider embeds the encrypted secrete data into the substituted image with the data hiding key. Finally a permutation process is utilized to generate the marked encrypted image based on the sharing key received from the content owner. Note that the encryption and data hiding keys are private keys while the sharing key is a public key shared by both the content owner and data hider. After receiving the marked encrypted image, the receiver can extract the secrete data, the original image, or both when the shared key is being utilized with the data hiding key, the encryption key, or both keys. Next, we discuss each phase in detail.

A. Reserving room

To obtain spaces for embedding the secrete data, reserving room first decomposes the original image into 8 binary bit planes, and utilizes 6 most significant bit (MSB) planes to accommodate the pixels from the other 2 LSB planes. The flowchart of reserving room is shown in Fig. 2.

Assume that the original image I is with a size of $M \times N$, each pixel value of I is within the range of [0, 255], the content owner decomposes the original image into 8 bit planes as defined by

$$I = \sum_{b=0}^{7} I_b \cdot 2^b \quad (2)$$

where I_b is the b^{th} bit plane of image I.

For each of the 6 MSB planes, calculate the capacity C_d $(2 \leqslant d \leqslant 7)$, if $C_d > 0$, orderly embed C_d bits of the LSB planes into the d^{th} bit plane. For better illustration, here we simply modify and demonstrate the process of the universal data embedding method [14].

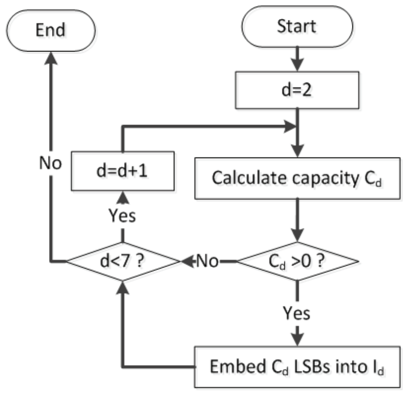


Fig. 2. Flowchart of reserving room

Bits in I_d $(2 \leqslant d \leqslant 7)$ are scanned in the inverse S order as shown in Fig. 3 to form a bit sequence $A_d = (a_1 a_2 a_3, ..., a_{MN})$. Separate A_d into λ (Eqn. 3) non-overlapping tuples with the length equal to 3 as defined by Eqn. 4.

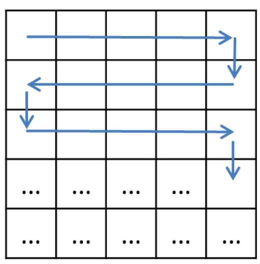


Fig. 3. Bit scanning order

$$\lambda = \lfloor \frac{M \times N}{3} \rfloor \tag{3}$$

$$\mathcal{D}(A_d, 3) = A_d^3 = (T_1T_2T_3, ..., T_\lambda) \tag{4}$$

where

$$T_j = (a_1^j a_2^j a_3^j), \quad (j \in [1, \lambda]) \tag{5}$$

Note that there are t ($1 \leqslant t \leqslant 8$) unique tuples due to the length of tuples equal to 3. Sort the t tuples in a descending order according to their frequency of occurrences. The sorted tuples are defined by

$$ST = (ST_1ST_2ST_3, ..., ST_t) \tag{6}$$

where $ST_g \in T_j$ ($1 \leqslant g \leqslant t, 1 \leqslant j \leqslant \lambda$). ST_1 and ST_t are the tuples with the most and least occurrence frequencies, respectively. Mapping ST into the Golomb-Rice codewords (GRC) which is defined by

$$GRC = (GRC_1GRC_2GRC_3, ..., GRC_t) \tag{7}$$

where

$$GRC_k = \begin{cases} 010 & \text{if } k = 1, \\ 011 & \text{if } k = 2, \\ 0 \uplus GRC_{k-2} & \text{if } k \geqslant 3, k \ mod \ 2 = 1, \\ 0 \uplus GRC_{k-2} & \text{if } k \geqslant 3, k \ mod \ 2 = 0. \end{cases} \tag{8}$$

and $\uplus$ is the string connection operation. From the definition, each GRC is in the following form

$$(qcr) = (0_10_2, ..., 0_z1r) \tag{9}$$

where q is the quotient part with length equals to z, in this case, $z \leqslant 4$; c is the terminal bit with a constant value 1; and r is the remainder part that belongs to $\{0, 1\}$.

Replace $T_j (1 \leqslant j \leqslant \lambda)$ with $GRC_k (1 \leqslant k \leqslant t)$ to generate a new set of tuples denoted by MA_d (Eqn. 10.) according to the mapping function as shown in Eqn. 11

$$MA_d = (\mathcal{T}_1\mathcal{T}_2\mathcal{T}_3, ..., \mathcal{T}_\lambda) \tag{10}$$

where $\mathcal{T}_p \in GRC$ ($1 \leqslant p \leqslant \lambda$).

$$GRC_g = f(T_j) \quad (\text{if } T_j = ST_g) \tag{11}$$

where $f(\cdot)$ is the multiple-to-one mapping function that maps $T_j (1 \leqslant j \leqslant \lambda)$ into $GRC_g (1 \leqslant g \leqslant t)$.

Trim each tuple $\mathcal{T}_p$ into two parts, $\mathcal{TQ}$ and $\mathcal{R}$, by the following equations

$$\mathcal{TQ} = q_1\overline{q_2}q_3\overline{q_4} \cdots \overline{q_\lambda} \tag{12}$$

$$\mathcal{R} = r_1r_2r_3 \cdots r_\lambda \tag{13}$$

where q_i and $r_i (1 \leqslant i \leqslant \lambda)$ denote the quotient and remainder parts of each tuple of MA, respectively; and $\overline{q_i}$ is the logic inverse of q_i.

Construct augmented payload P by

$$P = (p_1p_2p_3, ..., p_{2\lambda}) = (\mathcal{TQ} \uplus ST \uplus P_{LSBs}) \tag{14}$$

where P_{LSBs} denotes the bits that are orderly scanned from the 2 LSB planes.

The capacity C_d is the bit length of P_{LSBs}, which is calculated by

$$C_d = 2\lambda - L_{\mathcal{TQ}} - 3t \quad bits \tag{15}$$

where $L_{\mathcal{TQ}}$ is the length of $\mathcal{TQ}$. If $C_d \leqslant 0$, repeat previous processes by setting d to $d+1$ ($d < 7$); otherwise, replace A_d^3 by RA_d^3 with C_d bits (P_{LSBs}) embedded using Eqn. 16.

$$RA_d^3 = \{RT_1RT_2RT_3, ..., RT_\lambda\} \tag{16}$$

where

$$RT_j = (P_jr_j), \quad (1 \leqslant j \leqslant \lambda) \tag{17}$$

$$\mathcal{D}(P_d, 2) = P_d^2 = (P_1P_2, ..., P_\lambda) \tag{18}$$

$$P_j = (p_1^j p_2^j), \quad (1 \leqslant j \leqslant \lambda) \tag{19}$$

r_j is the j^{th} bit in $\mathcal{R}$ and P_j denotes 2 non-overlapping bits segmented from P.

Convert RA_d^3 into a bit sequence and reshape it with the size of $M \times N$ to obtain a new bit plane RI_d instead of I_d that with C_d LSBs embedded.

The final image with C bits of LSBs vacated is generated and denoted by RI, where C is calculated by

$$C = \sum_{d=2}^{7} C_d \quad (if \ C_d > 0) \tag{20}$$

B. Data hiding in encryption domain

Having the reserved room, we can embed the secrete data by encrypting RI. Firstly, a substituted image SI is generated by

$$SI_{(i,j,b)} = RI_{(i,j,b)} \oplus SM_{(i,j,b)}, \quad (0 \leqslant b \leqslant 7) \tag{21}$$

where $\oplus$ is the bit XOR operation; an $M \times N$ matrix SM with the data range of [0, 255] is generated by LSS. The LSS initial value X_0 and parameter γ are defined by the encryption key, which is generated from SHA.

After substitution, the bit plane index (BPI) with capacity larger than 0 and the number of LSBs (NB) embedded in each BPI plane are embedded in the last LP bits of the reserved area. LP is the bit length for accommodating BPI and NB. They will be used for the image recovering process.

Finally, we embed the size information C_m of the secrete data in the first NP LSBs to tell the data hider the length of LSBs can be modified. The C_m is calculated by

$$C_m = C - LP - NP \tag{22}$$

where

$$NP = \lfloor log_2(2MN) \rfloor \tag{23}$$

Once the data hider gets the substituted image SI and C_m extracted from the first NP LSBs, he/she can add secrete data m to the image using the LSB replacement without knowing the image contents. In order to achieve a higher level of security, the data hider encrypt secrete data before embedding using a data hiding key.

After embedding the secrete data, the data hider uses a scrambling method with the sharing key to permutate the image to generate the final marked encrypted image, denoted as I_E. The sharing key is received from content owner. Note that this step does not rely on any specific permutation method.

C. Data extraction and image recovering

Suppose the receiver has obtained I_E, he/she can extract the original image or/and secrete data using different security keys. The data extraction and image recovering are the reverse procedures of the embedding and encryption processes. Here, we briefly demonstrate the extraction/decryption processes.

1) Extract the secrete data: Using the sharing and data hiding keys, the receiver can extract the secrete data without knowing the original image content. Firstly, he applies the inverse permutation on I_E using the sharing key, extracts the encrypted secrete data from two LSB planes with NP and C_m, and recovers the secrete data using the data hiding key.

2) Extract the original image: If the receiver has both the encryption and sharing keys, he/she may extract two types of images: the marked decrypted image, which is similar like the original image with secrete data embedded, and the decrypted image, which is exactly the same as the original image.

For the marked decrypted image, the extraction process are following the steps:

- **Step 1.** Apply the inverse permutation to I_E using the sharing key and extract the parameter C_m from the first NP LSBs.
- **Step 2.** Apply the inverse substitution to the image except for the $C_m + NP$ LSBs to generate image $I^{'}$ using the encryption key.
- **Step 3.** For a bit plane $I_d^{'}$ (the initial value of d is 2) with capacity $C_d > 0$, convert it into a bit sequence $RA_d^{'}$ and sparse it using Eqn. 4. For each of the 3 bit tuples, obtain the first 2 bits to form a bit sequence $P^{'}$, and the remainder part denotes by $\mathcal{R}^{'}$. With the knowledge of NB, extract the trimmed quotient part $\mathcal{TQ}^{'}$, sorted tuples $\mathcal{ST}^{'}$ and the secrete data $P_{LSBs}^{'}$ from $P^{'}$. Reconstruct the bit sequence $A_d^{'}$ according to $\mathcal{R}^{'}$, $\mathcal{TQ}^{'}$, $\mathcal{ST}^{'}$ and GRC. Reshape $A_d^{'}$ to form the decrypted bit plane $I_d^{'}$.
- **Step 4.** Add d by 1. If $2 \leqslant d \leqslant 6$, repeat **Step 3** until all 6 MSB planes are recovered.

In order to obtain the decrypted image, after the previous steps, extract all $P_{LSBs}^{'}$ from 6 MSB planes and put them back to the original LSB positions.

IV. Implementation issues

In this section, we discuss some implementation issues of the proposed algorithm.

A. Left bits

In Eqn. 4, we separate the bit sequence into λ parts with the bit length equal to 3. If $(M \times N) mod\ 3 > 0$, there will be 1 or 2 bits left. In this case, only previous 3λ bits are processed and the left bits are added to the end of the bit sequence to reconstruct the bit plane RI_d. The same processes required for the image recovering phase.

B. Choice of the number of LSB planes

When the original image is an all-black or all-white image, it reaches the maximum capacity $\mathcal{C}$.

$$\begin{aligned} \mathcal{C} &= 6(\lfloor (MN)/3 \rfloor - 3) - LP - NP \\ &\leqslant 2MN - 18 - LP - NP \end{aligned} \tag{24}$$

Therefore, 2 LSB planes are enough to embed into 6 MSB planes for reserving room.

V. Simulations and comparisons

In this section, we present several simulation results and comparisons with several existing methods.

A. Histogram analysis

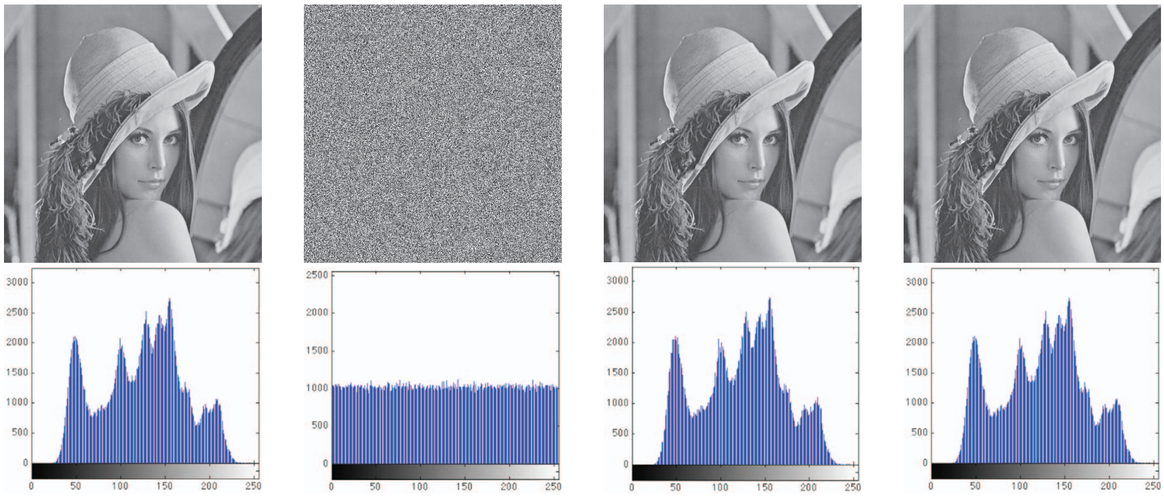
Fig. 4 shows the simulation results of the proposed algorithm. The original image is the grayscale Lena image (Fig. 4(a)) with a size of 512×512. The marked encrypted image is embedded with secrete data with a embedding rate of 0.7 bpp (bit per pixel). From the results, the marked encrypted image has a uniform histogram distribution (Fig. 4(b)). This ensures the hackers' difficulty to extract any useful information. The original image can be perfectly recovered without any error as can be seen in Fig. 4(d). There is no visual difference among the original, marked decrypted and decrypted images (Figs. 4(a), (c) and (d)).

B. Differential analysis

Fig. 5 demonstrates the differential analysis of the proposed algorithm. Figs. 5(a) and (b) are two Lena images with one bit difference as shown in Fig. 5(c). The pixel value at position (5, 5) is 165 in Fig. 5(a) while 164 in Fig. 5(b). Encrypting these two images with the same secrete data (the embedding rate is 0.701 bpp) and the security keys generates the marked encrypted images as shown in Figs. 5(d) and (e). Fig. 5(f) shows the difference between Figs. 5(d) and (e). The results indicate that even one bit difference in the original image will result in a totally different marked encrypted images. This is because the SHA and LSS in the proposed algorithm are sensitive to the change of the original image.

Two measures are utilized to quantitatively evaluate the impact of two marked encrypted images with tiny changes in the original image. They are the number of pixel change rate (NPCR) and the unified average changing intensity (UACI) as defined by Eqn. 25 and Eqn. 27.

$$NPCR = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D(i,j)}{M \times N} \times 100\% \tag{25}$$



(a) (b) (c) (d)

Fig. 4. Histogram analysis of the original Lena image and its marked encrypted and decrypted images. (a) The original image and its histogram; (b) the marked encrypted image and its histogram; (c) the marked decrypted image (embedding rate 0.7 bpp) and its histogram; (d) the decrypted image and its histogram.

where

$$D(i,j) = \begin{cases} 0 & \text{if } (I_E^1 = I_E^2)\ , \\ 1 & \text{if } (I_E^1 \neq I_E^2)\ . \end{cases} \quad (26)$$

$$UACI = \frac{1}{M \times N}\left[\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{|I_E^1 - I_E^2|}{255}\right] \times 100\% \quad (27)$$

where I_E^1 and I_E^2 are two marked encrypted images.

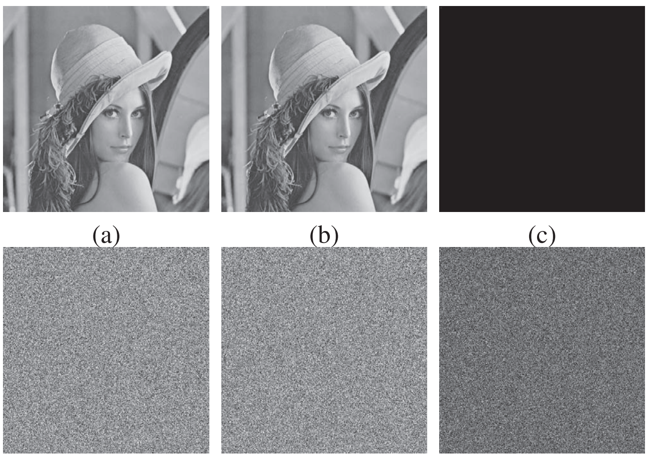


Fig. 5. Differential analysis. (a) The original image I^1; (b) the original image I^2, which is obtained from (a) with one bit difference; (c) the difference between (a) and (b), $|I^1 - I^2|$; (d) the marked encrypted image I_E^1 of (a); (e) the marked encrypted image I_E^2 of (b); (f) the difference between (d) and (e), $|I_E^1 - I_E^2|$.

The NPCR and UACI results obtained from two marked encrypted images in Fig. 5 are 99.344% and 33.544%, which are extremely close to the expected NPCR and UACI values (99.609% and 33.464%) proved in [15]. Therefore, our proposed algorithm has good diffusion properties and can withstand the differential attack.

C. Embedding capacity comparison

We compared the embedding capacity of our proposed algorithm with several existing methods in 5 standard images (512×512). The results are shown in Table I. For methods in [6] and [7], we use the block size of 8×8 to do the simulations, the pure capacity is calculated by $Cap(1-H(\rho))$ rather than Cap, where the $H(\rho)$ indicates the binary entropy function with error rate ρ. From the results, the capacity of these two methods are relatively small because one block is utilized to accommodates only 1 bit secrete data. Methods in [8] and [9] moderately improved the embedding capacity by compressing the encrypted image for embedding the secrete data and we list the capacities under the case of fully recovering the original images. For Ma's method in [10], we use the modified RDH method in [16] for experiments. The proposed algorithm achieves the maximum embedding capacity among these methods in most of the cases. Its embedding capacity is 52% larger than the best result of these existing methods in the best scenario.

TABLE I
CAPACITY (.DPP) COMPARISON OF DIFFERENT METHODS.

Capacity	Lena	Man	Peppers	Barbara	Copter
RDHED	0.701	1.105	0.810	1.392	1.455
Zhang's [6]	0.013	0.014	0.013	0.004	0.061
Hong's [7]	0.014	0.015	0.015	0.010	0.062
Zhang's [8]	0.033	0.025	0.032	0.027	0.035
Zhang's [9]	0.101	0.076	0.116	0.091	0.132
Ma's [10]	0.996	0.975	0.636	0.915	0.998

D. PSNR comparisons

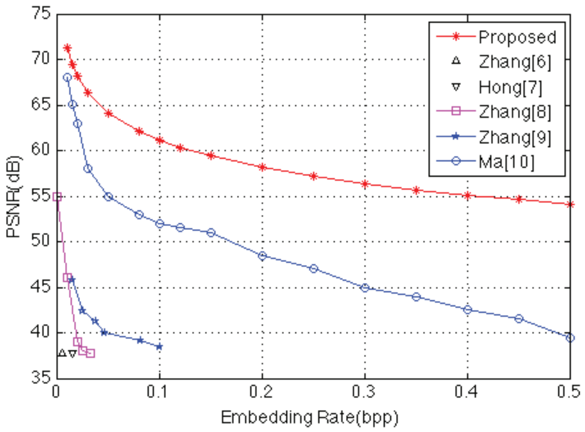
We use peak signal to noise ratio (PSNR) to compare the quality of the marked decrypted image by different methods. The PSNR is defined by

$$PSNR = 10 \times log_{10} \frac{255^2}{MSE} \quad (28)$$

where

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_{(i,j)} - I'_{(i,j)})^2 \quad (29)$$

$I_{(i,j)}$ is the original image and $I'_{(i,j)}$ is the marked decrypted version of $I_{(i,j)}$. Fig. 6 plots the PSNR results of the marked decrypted Lena image under different embedding rates. The results demonstrate that our proposed algorithm has the highest PSNR result under all embedding rates. This is because the proposed algorithm embeds the additional data only in the LSBs of the original image while other methods use the higher bit planes for secrete data embedding.



(a)

Fig. 6. PSNR comparisons with different methods of Lena image.

VI. Conclusion

In this work, we proposed a new reversible data hiding algorithm in the encryption domain. It integrates data embedding into the image encryption process. Both the original image and secrete data are protected. The image recovering and data extraction can be accomplished individually by using different security keys. Experimental results and comparisons have demonstrated that our proposed algorithm effectively improves the state-of-the-art methods in embedding capacity and PSNR.

Acknowledgment

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/017/2012/A1 and by the Research Committee at University of Macau under Grants MYRG2014-00003-FST, MRG017/ZYC/2014/FST, MYRG113(Y1-L3)-FST12-ZYC and MRG001/ZYC/2013/FST.

References

[1] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 3, pp. 354–362, March 2006.

[2] J. Tian, "Reversible data embedding using a difference expansion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 8, pp. 890–896, Aug 2003.

[3] V. Sachnev, H.-J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 7, pp. 989–999, July 2009.

[4] H.-C. Huang, F.-C. Chang, and W.-C. Fang, "Reversible data hiding with histogram-based difference expansion for qr code applications," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 2, pp. 779–787, May 2011.

[5] M. Fallahpour, D. Megias, and M. Ghanbari, "Reversible and high-capacity data hiding in medical images," *Image Processing, IET*, vol. 5, no. 2, pp. 190–197, March 2011.

[6] X. Zhang, "Reversible data hiding in encrypted image," *Signal Processing Letters, IEEE*, vol. 18, no. 4, pp. 255–258, April 2011.

[7] W. Hong, T.-S. Chen, and H.-Y. Wu, "An improved reversible data hiding in encrypted images using side match," *Signal Processing Letters, IEEE*, vol. 19, no. 4, pp. 199–202, April 2012.

[8] X. Zhang, "Separable reversible data hiding in encrypted image," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 2, pp. 826–832, April 2012.

[9] X. Zhang, Z. Qian, G. Feng, and Y. Ren, "Efficient reversible data hiding in encrypted images," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 322 – 328, 2014.

[10] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 3, pp. 553–562, March 2013.

[11] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Processing*, vol. 94, pp. 118 – 127, 2014.

[12] [Online]. Available: http://en.wikipedia.org/wiki/Secure_Hash_Algorithm

[13] Y. Zhou, L. Bao, and C. P. Chen, "A new 1D chaotic system for image encryption," *Signal Processing*, vol. 97, pp. 172 – 182, 2014.

[14] M. S. A. Karim and K. Wong, "Universal data embedding in encrypted domain," *Signal Processing*, vol. 94, pp. 174 – 182, 2014.

[15] C. Fu, J.-j. Chen, H. Zou, W.-h. Meng, Y.-f. Zhan, and Y.-w. Yu, "A chaos-based digital image encryption scheme with an improved diffusion strategy," *Optics express*, vol. 20, no. 3, pp. 2363–2378, 2012.

[16] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *Information Forensics and Security, IEEE Transactions on*, vol. 5, no. 1, pp. 187–193, March 2010.