Reversible Data Hiding for Compression-friendly Image Encryption Method

KokSheik Wong* and Hitoshi Kiya†
* School of Information Technology, Monash University Malaysia, Selangor, Malaysia
E-mail: wong.koksheik@monash.edu
† Faculty of System Design, Tokyo Metropolitan University, Tokyo, Japan
E-mail: kiya@tmu.edu.jp

***Abstract*—This paper aims to add the data hiding capability to a compression-friendly image encryption method [1]. The data hiding method is designed to be *commutative* with the encryption method [1], and the resulting joint method is separable. Unlike the conventional histogram based methods where the image is processed entirely, the histogram of each non-overlapping image block is considered, where the block size coincides with the encryption block size. In addition, two ordering functions are proposed to facilitate the data hiding and extraction processes. Further constrains are imposed to ensure correct data extraction and reversibility. Experiment results suggest that the resulting joint method retains the compression-friendly feature of [1], with insignificant file size increment across all quality factors considered. In the best case scenario, 45765 bits could be hidden into the test image Airplane.**

***Index Terms*—Data hiding, reversible, joint-method, commutative, separable**

I. Introduction

In recent years, many researchers are aiming to achieve joint utilization of data hiding and encryption, focusing mainly on image / video [2]. Joint method is attractive, particularly in the era of cloud computing where mobile device users tend to store their contents / media in the cloud [3]. Specifically, the encryption module ensures privacy, while the data hiding module allows some administrative or personal note to be hidden into the content for better management. Another applications of joint method is the data management in hospital, where officers with different access rights can manage patient's file, which consists of personal information and health / medical history [2].

Joint method can be achieved in various ways [4], including hide-then-encrypt, encrypt-then-hide, as well as redesigned data hiding method that purposely distort image quality (e.g., [2]) and encryption method that operates with restricted states (e.g., [5]). Joint method is further improved to achieve additional features. One of the attractive features is *separability* between decryption and data extraction [6], [7]. In particular, given a separable method, the hidden data can be extracted directly from the encrypted image or the decrypted image. On the other hand, researchers also invented *commutative* methods, where the output is exactly the same for both the hide-then-encrypt and encrypt-then-hide routes [8], [9]. One of the straightforward ways to achieve both the separable and commutative properties is to partition the image into two non-overlapping parts, where each part plays a different role. For example in [8], the sign of the coefficients are randomized to perceptually encrypt the image, while the magnitudes are modulated to hide data. However, the partitioning process becomes more challenging for the raw image because the perceptual semantic of the image is derived from the assemble of pixels.

Despite achieving both the separable and commutative properties, output of the joint methods are often not compression-friendly, i.e., they suffer from significant file size increment when their outputs are compressed using common compression standard such as JPEG [10], which is usually lossy. It is because the redundancy exploited by compression is significantly reduced by encryption, where the output appears random. Although it is possible to start with a compressed content where the partitioning process is more straightforward, there are applications where no loss of information is permitted.

Recently, some compression-friendly encryption methods for raw image are proposed, including [1], [11]–[14]. However, these methods lack the data hiding capability, which has been proven to be practical in content management including authentication, fingerprinting, and watermarking [15]. Therefore, in this work, a histogram based reversible data hiding method is proposed for incorporation into [1] to achieve a commutative and separable joint method. Kurihara et al.'s method is considered because it is compression-friendly to both lossy and lossless standards, and its security robustness is further confirmed in [16], [17]. Instead of manipulating the image entirely, the histogram of each non-overlapping image block is analyzed for data hiding. Two ordering functions are put forward so that the image blocks (in fact, intensity values) can be processed in the exact same order for data hiding and extraction, irregardless whether the image is encrypted.

II. Review of Compression-friendly Encryption Method [1]

This section brief reviews the compression-friendly image encryption method proposed by Kurihara et al. [1]. Suppose $A = \{A_r, A_g, A_b\}$ is a color image of dimension $nM \times nN \times 3$, where A_r, A_g and A_b denote the *red*, *green*, and *blue* channels, respectively. The image A is then divided into non-overlapping blocks $A^{i,j} = \{A_r^{i,j}, A_g^{i,j}, A_b^{i,j}\}$ each of dimension $n \times n \times 3$ for $1 \leq i \leq M$ and $1 \leq j \leq N$.

To generate an encrypted image B, Kurihara et al. proposed the following four encryption modules:

Op1 Block shuffling: The blocks $A^{i,j}$ are shuffled so that the (i', j')-th block (i.e., all channels considered together) takes the position of the (i, j)-th block for $i \neq i'$ and $j \neq j'$, i.e., $B^{i,j} \leftarrow A^{i',j'}$.

Op2 Block-scrambling and block rotation: Each block $A^{i,j}$ is rotated by $0, 90, 180$ or 270^o. The resulting block is flipped in the horizontal, vertical, or both directions.

Op3 Negative-positive transformation: The intensity values in $A^{i,j}$ are negated. For example, $10_{10} = 00001010_2$ is negated to $245_{10} = 11110101_2$ in case there are 8 bits per channel.

Op4 Color component shuffling: The color components within a block $A^{i,j}$ are shuffled. For example, it is possible to have $B_r^{i,j} \leftarrow I_g^{i,j}, B_g^{i,j} \leftarrow I_b^{i,j}$, etc., where $I^{i,j}$ is an intermediate block attained after applying Op1, Op2, and / or Op3.

The user supplied secret key will determine the behavior of the aforementioned encryption modules. It is reported in [1] that the compressibility of B, with respect to JPEG, is high when the block size n coincides with those in the standard, i.e., optimal results are achieved when $n = 8$ or 16.

III. Proposed Data Hiding Method

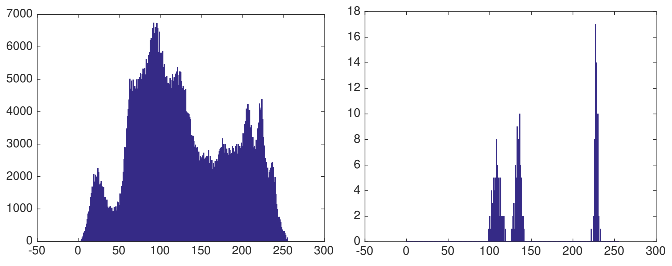
The proposed data hiding method is first described, followed by the discussions on achieving the reversible, commutative and separable properties when it is incorporated with Kurihara et al.'s compression-friendly encryption method [1]. Note that this work consider all encryption modules in [1], except for Op4, i.e., negative-positive transformation.

A. Basic Mechanism for Data Hiding

In this work, the conventional histogram shifting method [18] is modified to hide data into an image, one block at a time. Here, the same notation is adopted where $A^{i,j} = \{A_r^{i,j}, A_g^{i,j}, A_b^{i,j}\}$ denotes a block of $n \times n \times 3$ intensity values.

Let h_k denote the frequency of occurance for intensity value k. For each block $A^{i,j}$, a common histogram $H^{i,j} = \{h_0^{i,j}, h_1^{i,j}, \dots h_{255}^{i,j}\}$ is constructed for all color channels. The superscript 'i, j' will be dropped when there is no confusion. Unlike the traditional histogram shifting methods where most histogram bins are occupied (e.g., see Fig. 1(a)), there are many empty bins in $H^{i,j}$ due to the limited number of intensity values (e.g., 192 in total when $n = 8$). More importantly, the intensity values tend to form clusters, where an example is shown in Fig. 1(b) for $A^{5,6}$ of Lenna. Formally, a cluster is a sequence of consecutive nonempty histogram bins $\{h_k\}_{k=s}^{e}$ for some starting and ending points s and e, respectively. Therefore, the histograms shown in Fig. 1(a) and (b) have 1 and 3 clusters, respectively.

Next, the cluster consisting of the most elements is identified and referred to as $C^{i,j}$. Among all bins $\{h_s, h_{s+1}, \cdots, h_e\}$ in



(a) Histogram of entire image (b) Histogram of $A^{5,6}$

Fig. 1. Illustration of cluster in histogram using Lenna.

$C^{i,j}$ for $s, e \in [0, 255]$ and $s \leq e$, the bin with the highest frequency h_p is identified, which will be utilized for data hiding. Instead of storing the peak value p directly, the cluster $C^{i,j}$ is identified, and the distance $\delta^{i,j} = h_p - h_s$ is computed and stored as side information for data extraction. All bins $h_k \in H^{i,j}$ are then right-shifted for all $k > p$, so that h_{p+1} is empty. Data hiding can be carried out by using the bins h_p and h_{p+1}, where they represent '0' and '1', respectively.

B. Ordering Functions

In Kurihara et al.'s method [1], it is possible that the RGB channels within $A^{i,j}$ are permuted, i.e., Op4. Note that the permutation is restricted within a channel, instead of individual intensity values. To impose the notion of *order* on all RGB channels so that the receiver knows which intensity value to process first, given $A^{i,j}$, the channel (i.e., array of $n \times n$ values) with the lowest values is processed first, followed by the one with the second lowest value, and so forth. As an example, the clusters in Fig. 1(b), in the order from left to right, correspond to the blue, green, and red channels, respectively. Hence the blue channel is considered first despite of the convention of storing the red channel intensity values first. In the event where intensity values from two or more channels contribute to the same cluster, as long as the respective lowest values in each channel do not coincide, the aforementioned order can be imposed. Note that the data hiding process does not affect the relative order among the lowest value of each channel. With this construct, the order of processing is defined irregardless whether the channels are permuted. This also implies that the red channel may not be processed first, but instead the order of processing depends on the lowest intensity value.

Second, it is possible that the blocks $A^{i,j}$ are shuffled by Op1 in [1]. To impose the notion of *order* on all $A^{i,j}$, the following statistics are considered to form a 7-tuple $T^{i,j}$:

S1 Number of clusters in $A^{i,j}$

S2 Length of $C^{i,j}$, denoted by $\lambda^{i,j} = h_e - h_s + 1$

S3 The frequency of $h_p^{i,j}$

S4 The smallest intensity value in $A^{i,j}$, i.e., $\max\{A^{i,j}\}$.

S5 The largest intensity value in $A^{i,j}$, i.e., $\min\{A^{i,j}\}$.

S6 Frequency of $\max\{A^{i,j}\}$

S7 Frequency of $\min\{A^{i,j}\}$

In order to ensure S1 and S2 remain invariant before and after data hiding, for each $A^{i,j}$, the first encountered intensity value

p is reserved as a dummy, which will be utilized to ensure that the bins h_p and h_{p+1} are both nonempty. For example, after right-shifting $H^{i,j}$ to make room for data hiding (i.e., $h_p \in C^{i,j}$ is now empty), in the event the data segment to be hidden is a run of 1's, the dummy value is set to p. On the other hand, the dummy value is set to $p+1$ in case the data segment is a run of 0's. Otherwise, the dummy value is set to p by default. This is to ensure that the number of clusters remain unchanged, and at the same time the longest cluster remains to be the longest one even after data hiding. On the other hand, for S3, $\delta^{i,j}$ can be derived from the side information, which can further utilized to derive the peak value p. Although other statistics may change, the relative order between two blocks remains intact due to histogram shifting. Using the aforementioned 7 statistical information, the blocks $A^{i,j}$ are ordered for data hiding and extraction purposes.

C. *Further Control Mechanisms*

Similar to the conventional histogram shifting based methods, not all blocks are suitable for data hiding. It is particularly tricky when both the first and last bins, i.e., h_0 and h_{255}, are nonempty. However, it is empirically found that, the blocks with nonempty h_0 and h_{255} bins are, in general, having high number of clusters. In fact, having both nonempty h_0 and h_{255} suggest a high chance of having impulsive noise in $A^{i,j}$ due to the small number of intensity values (i.e., 192 when $n = 8$). Therefore, the proposed data hiding process is restricted to blocks with less than τ number of clusters. Since the number of cluster remains invariant before and after data hiding, the qualified blocks $A^{i,j}$ remain qualified and hence they can be identified accordingly for processing. In a way, smaller τ corresponds to smoother blocks, and vice versa. This implies that smooth blocks are considered for data hiding in the proposed method.

In addition, it is possible that more that one block to assume the exact same statistical information, i.e., $A^{i,j} = A^{i',j'}$ for $i \neq i'$ and $j \neq j'$. For example, the sky areas in the standard test image Airplane. When such blocks are encountered, the same data segment is embedded to ensure that the same statistical information (defined in Section III-B) is maintained.

Last but not least, the offset information $\delta^{i,j}$ needs to be encoded for proper decoding purpose. In the worse case scenario, the offset information (from the left end of $C^{i,j}$, i.e., h_s, to h_e) requires 8 bits per qualified block for a 24-bit color image. However, statistics of the offsets reveal that this overhead can be reduced to $4 \sim 5$ bits per block by exploiting entropy coding. The distribution of offset values for the test image Lenna is shown in Fig. 2, which requires 4.4847 bits per offset, on average, when using Huffman coding.

D. *Reversibility, Commutative and Separability*

Since the values in $A^{i,j}$ are simply right-shifted, reversibility is straightforward. Note that the value p can be derived from $C^{i,j}$ (hence h_s) and $\delta^{i,j}$. Basically, the values $p+1$ is reduced by unity, and the histogram bins h_k are left-shifted for $k > p+1$.

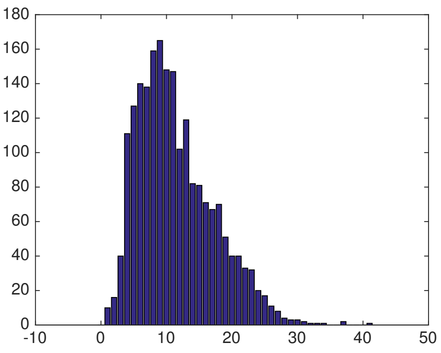


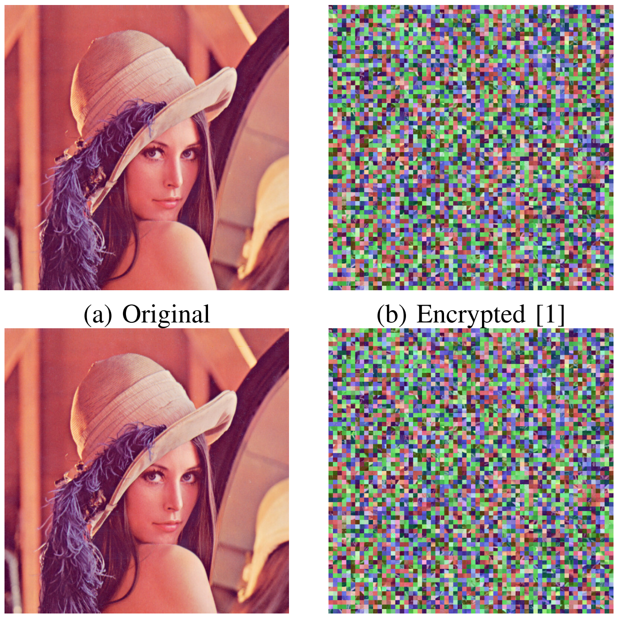
Fig. 2. Distribution of offset values for the test image - Lenna.

For the commutative property, it is noteworthy that the histogram of intensity values $H^{i,j}$ within each block $A^{i,j}$ remains unchanged with respect to Op1, Op2, and Op4 in [1], because they only change the position of the intensity values. Since the proposed ordering functions maintain the relative order before and after encryption, the intensity values can be processed in the same order irregardless of the status of the image. Similarly, since the statistics remains intact, the hidden data can be extracted irregardless whether the image is encrypted. Therefore, the proposed data hiding method is also separable when deployed to form a joint method together with Kurihara et al.'s method [1].

IV. Experiment

As a proof of concept, the proposed data hiding method is implemented in Matlab (version R2015a) running on macOS Sierra (Version 10.12.2) with 16GB of main memory. Five standard test color images each of size 512×512, namely, Airplane, Lenna, Mandrill, Peppers and Splash are considered to evaluate the proposed data hiding method. Unless specified otherwise, we set $\tau = 10$. In addition, the Huffman table constructed with Mandrill's statistics is used as the common table because the offsets $\delta^{i,j}$ in Mandrill span a wide range of values. Using Lenna as the representative example, Fig. 3 shows the original image, encrypted image using [1], as well as the corresponding output with data hidden by using the proposed data hiding method. Note that Fig. 3(d) shows the output of a joint method.

First, the raw hiding capacity (bits) for each image is recorded in Table I. Results suggest that the proposed method can accommodate more payload when the image is relatively smooth (e.g., Airplane, Lenna), and vice versa (e.g., Mandrill). It is an expected results because textured image tends to have more clusters, hence less blocks are qualified for data hiding. Table I also records the number of qualified blocks when $\tau = 10$, as well as the actual size of the side information when encoded using Huffman coding with Mandrill's statistics. In the right most column, the effective capacity (bits) is recorded, which is the result after subtracting the size of the side information and the number of qualified blocks. The number of blocks is subtracted because one bit is spent as dummy to



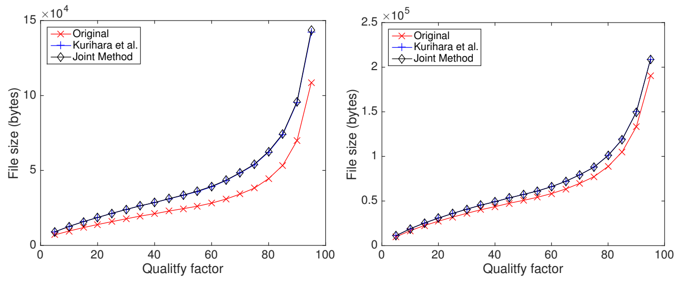
(a) Original (b) Encrypted [1]

(c) With data hidden (d) Joint method

Fig. 3. Output images

TABLE I
RESULTS FOR STANDARD TEST IMAGES

Test Image	Qualified Blocks	Raw capacity (bits)	Offset Overhead	Effective capacity (bits)
Airplane	2706	62331	13860	45765
Lenna	2063	24453	10744	11646
Mandrill	272	2658	1365	1021
Peppers	1420	23070	7642	14008
Splash	2942	47910	15789	29179



(a) Lenna (b) Mandrill

Fig. 4. Graph of file size against quality factor for JPEG

ensure the consistency of S1 and S2 (see Section III-B) before and after data hiding. When compared to the conventional joint methods, our average effective capacity (i.e., 20323.8 bits) is higher than that of Ong et al.'s method [5] (i.e., 9015.0 bits), but inferior to that of Ong et. al's method [2] (i.e., 598982.8 bits). However, it should be noted that both [5] and [2] are not compression friendly.

Second, the performance in terms of compressibility using ISO JPEG compression standard [10] is evaluated. Specifically, the compressibility of the original image, output generated by Kurihara et al.'s method [1] (i.e., encrypted), as well as the output generated by the joint method (i.e., encrypted + data hidden) are considered. Figure 4(a) and (b) show the results for the test images Lenna and Mandrill, respectively. It is observed that, irregardless of the texture of the image, the original image ('$\times$' mark) has the smallest file size after compression. More importantly, the file size between [1] ('$+$' mark) and the proposed joint method ('$\diamond$' mark) is very small for both Lenna and Mandrill. Specifically, the observed average file size increment across QFs is $< 1\%$ for both images. Similar outcome is observed for other test images, and we omit the results here. These results suggest that the compressibility of Kurihara et al.'s method [1] is retained despite data is hidden into it using the proposed reversible data hiding method.

V. Conclusions

A data hiding method is designed for incorporation into the compression-friendly encryption method [1]. The encryption modules in [1] are analyzed to identify invariant statistics, which are in turn exploited for data hiding irregardless whether the image is encrypted. The conventional histogram shifting based data hiding method is enhanced, and two ordering functions are defined to facilitate the data hiding and extraction processes. The proposed data hiding method is reversible, and it forms a commutative as well as separable joint method when incorporated with [1]. Results suggest that the encrypted image can be efficiently compressed using JPEG.

As future work, we aim to improve the proposed data hiding method to function when the negative-positive transformation module, i.e., Op3, is also considered to encrypt the image. In addition, the compressibility of the proposed joint method will be verified using more compression standards.

References

[1] Kenta Kurihara, Masanori Kikuchi, Shoko Imaizumi, Sayaka Shiota, and Hitoshi Kiya, "An encryption-then-compression system for jpeg/motion JPEG standard," *IEICE Transactions*, vol. 98-A, no. 11, pp. 2238–2245, 2015.

[2] Simying Ong, KokSheik Wong, and Kiyoshi Tanaka, "A scalable reversible data embedding method with progressive quality degradation functionality," *Sig. Proc.: Image Comm.*, vol. 29, no. 1, pp. 135–149, 2014.

[3] Reza Moradi Rad, KokSheik Wong, and Jing-Ming Guo, "A unified data embedding and scrambling method," *IEEE Trans. Image Processing*, vol. 23, no. 4, pp. 1463–1475, 2014.

[4] SimYing Ong, *Data Insertion and Scrambling for Unified Scalable Information Hiding*, Ph.D. thesis, University of Malaya, 2015.

[5] S. Ong, K. Wong, and K. Tanaka, "Reversible and tunable scrambling-embedding method," in *International Symposium on Intelligent Signal Processing and Communications Systems*, Nov 2013, pp. 608–613.

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

[7] M. Fujiyoshi, "Separable reversible data hiding in encrypted images with histogram permutation," in *IEEE International Conference on Multimedia and Expo Workshops*, July 2013, pp. 1 – 4.

[8] S. Lian, Z. Liu, R. Zhen, and H. Wang, "Commutative encryption and watermarking in video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, pp. 774 – 778, 2007.

[9] M. Cancellaro, F. Battisti, M. Carli, G. Boato, F. G. B. De Natale, and A Neri, "A commutative digital image watermarking and encryption method in the tree structured haar transform domain," *Signal Processing: Image Communication*, vol. 26, no. 1, pp. 1–12, Jan. 2011.

[10] William B.Pennebaker and Joan L.Mitchell, *JPEG: still image data compression standard*, Van Nostrand Reinhold, 1992.

[11] Kenta KURIHARA, Shoko IMAIZUMI, Sayaka SHIOTA, and Hitoshi KIYA, “An encryption-then-compression system for lossless image compression standards,” *IEICE Transactions on Information and Systems*, vol. E100.D, no. 1, pp. 52–56, 2017.

[12] G. Gankhuyag, S. Hong, and Y. Choe, “Compression friendly medical image encryption based order relation,” in *2013 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2013, pp. 568–569.

[13] W. Liu, W. Zeng, L. Dong, and Q. Yao, “Efficient compression of encrypted grayscale images,” *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 1097–1102, April 2010.

[14] Mark Johnson, Prakash Ishwar, Vinod M. Prabhakaran, Daniel Schonberg, and Kannan Ramchandran, “On compressing encrypted data,” *IEEE Trans. Signal Processing*, vol. 52, no. 10, pp. 2992–3006, 2004.

[15] Y. Tew and K. Wong, “An overview of information hiding in H.264/AVC compressed video,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 2, pp. 305 – 319, February 2014.

[16] Tatsuya Chuman, Kenta Kurihara, and Hitoshi Kiya, “On the security of block scrambling-based etc systems against jigsaw puzzle solver attacks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, p. To appear.

[17] T. Chuman, K. Kurihara, and H. Kiya, “Security evaluation for block scrambling-based etc systems against extended jigsaw puzzle solver attacks,” in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, July 2017, pp. 229–234.

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