International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 24, No. 3 (2010) 433–456
© World Scientific Publishing Company
DOI: 10.1142/S0218001410007968

World Scientific
www.worldscientific.com

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

REVERSIBLE JPEG-BASED HIDING METHOD WITH HIGH HIDING-RATIO

LEE SHU-TENG CHEN*, SIAN-JHENG LIN and JA-CHEN LIN
Department of Computer Science and Information Engineering
National Chiao Tung University, 1001 Ta Hsueh Rd.
Hsinchu, Taiwan, 300, R.O.C.
**stlee@cs.nctu.edu.tw*

The stego-images generated by many existing hiding techniques are not economic in size, and hence need compression. Unfortunately, compression usually destroys the secret content hidden inside. To solve this dilemma, some hiding methods based on compression code (rather than the image itself) are reported. This paper proposes a high-capacity and high-hiding-ratio *"*reversible*"* steganography method based on JPEG-compression code. In the proposed method, the JPEG compression code of an image is used as the cover media. An 8×8 hiding-capacity table is firstly evaluated, which is then utilized to modify the quantization table attached to the given JPEG code. The two quantization tables (modified and original) together can map the DCT coefficients of each block to some larger DCT coefficients, with secret data hidden inside these larger DCT coefficients. In the decoding process, after lossless extraction of the hidden secret data, the proposed method can also recover the original JPEG-compression code. Experimental results show that our method outperforms other JPEG-based hiding methods (reversible or not) regarding both hiding-ratio and stego-image's quality.

Keywords: Reversible data hiding; JPEG; stego code; hiding ratio.

1. Introduction

Transmission of digital media to the Internet is very popular nowadays; however, since the Internet is a public channel, problems such as data security or copyright also arise. Data hiding is a technique to conceal secret data in digital media. It embeds the secret data into the cover image, and the generated stego-image looks like the cover image so that the unauthorized users will not be aware of the existence of the secret data. Data hiding schemes can be divided into two categories: spatial domain [1,5,9,20,23,24,26,27] and frequency domain.[3,4,6,10,12,13,17,21,22,25] Spatial domain schemes embed the secret data into the pixel values of the cover image to obtain stego-images. However, due to the limited bandwidth of networks, digital media (including stego-images) are usually compressed before transmission. Although the smaller size offered by compression can save transmission time, compression using

*Author for correspondence

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

higher compression rate (lossy compression) often destroys or damages the hidden content of a stego media in which the secret data are embedded. Therefore, some hiding methods [3,4,6,10,12,13,17,21,22,25] that belong to the frequency domain had been proposed to hide the secret directly in the Joint Photographic Experts Group (JPEG) compression code.

In Jsteg,[22] the secret data were embedded in the Least Significant Bit (LSB) of the quantized Discrete Cosine Transform (DCT) coefficients whose values are not in $\{1, 0, -1\}$. Because the quantized-coefficients are almost all zero after the DCT transformation and quantization steps of JPEG, the hiding capacity in Jsteg is quite limited. As a result, Chang *et al.*[3] modified the JPEG default quantization table and successfully embedded the secret data into the 26 quantized-coefficients located in the middle-frequency part. Since the quantizers in the middle-frequency part of their quantization table were set to one, the quantized-coefficients in the middle-frequency part were usually nonzero. In order to find a balance between the hiding capacity and the stego-image quality, Tseng and Chang[21] further proposed an adaptive data hiding method by using their capacity table derived from the JPEG default quantization table and the Human Visual System (HVS) to determine the embedded length of secret. Iwata *et al.* also modified in Ref. 10 the boundaries between zero and nonzero quantized-coefficients to hide the secret.

Among the reported hiding techniques, some are reversible.[1,4,6] A reversible hiding technique can loss-freely recover not only the hidden data but also the compression-code of the cover media (or original cover media if no compression is used). In the spatial domain, Barton[1] losslessly compressed the bits to be overlaid, and the compressed data were embedded together with the secret data into the cover image. In the frequency domain, the authentication method of Fridrich *et al.*[6] obtained a secret watermark by hashing the quantized-coefficients. The secret watermark was then embedded in the least significant bits of a further compressed result of the quantized DCT coefficients. Chang *et al.*[4] also used the successive zero-quantized-coefficients in the middle frequency part to embed the secret. In addition, the quantized table was modified to improve the quality of their stego-images. The methods[3,10,12,13,17,21,22,25] also successfully embedded the secret into JPEG compression codes, but these methods are not reversible.

The current paper intends to propose a high hiding-ratio and reversible method based on JPEG compression code. Hence, the method can recover the original JPEG compression code after extracting the secret which can be big in size. The remainder of this paper is organized as follows. Section 2 reviews the related works. Section 3 presents the proposed method. Section 4 provides the experimental results, including comparisons with other methods. Section 5 gives the summary.

Symbols:

Q: The original block-independent 8×8 quantization table used by JPEG
$\tilde{Q}$: Our block-independent 8×8 modified quantization table [see Eq. (14)]

F: An original 8×8 quantized-block of JPEG
$\tilde{F}$: Our 8×8 stego quantized-block after hiding secret in F
HC: Our block-independent 8×8 hiding-capacity table [see Eq. (11)]
level: A user-specified integer between 1 and 3 to control the hiding-capacity table
num: A user-specified integer between 1 and 32 to control how many coefficients are used per block for hiding (apply the same *num* value to all blocks).

2. A Review of JPEG Hiding Methods

In this section, a basic but introductory technique called Jsteg[22] is reviewed. After that, a sequence of Chang's graceful methods[3,4,21] are reviewed, respectively, to let the readers have the detailed background knowledge about this topic.

Jsteg[22] is a very simple yet introductory hiding tool that embeds the secret into JPEG code. Its preprocessing before embedding is just like ordinary JPEG-compression: (1) the given image is divided into nonoverlapping blocks of 8×8 pixels each. Then for each block, apply DCT to transform the 8×8 pixels into the 8×8 DCT coefficients. (2) Next, the 8×8 DCT coefficients of each block are quantized using the JPEG default quantization table shown in Fig. 1(a). After the preprocessing, the secret data are then embedded in the LSB of each quantized-coefficient whose values are not 1, 0, or −1. Finally, each embedded quantized-block is compressed further by the JPEG entropy coding. The JPEG stego-code is thus generated when all blocks are done.

In Chang *et al.*'s hiding method,[3] each 8×8 block is also transformed into 8×8 DCT coefficients using DCT. However, the 8×8 DCT coefficients are then quantized using a modified quantization table shown in Fig. 1(b). Now, the secret data are sequentially embedded into the two-least-signification bits of each of the 26 quantized-coefficients whose corresponding quantization-resolutions in the modified quantization table [Fig. 1(b)] are one. Finally, the JPEG entropy coding is applied to encode the whole 8×8 embedded quantized-block. By processing all blocks in order, the modified quantization table [Fig. 1(b)] is put in the header of the JPEG file, and the JPEG stego-code is generated.

In Tseng and Chang's hiding method,[21] suppose that a given image is already divided into nonoverlapping blocks of 8×8 pixels each and compressed using JPEG. After the JPEG code for this image has been generated, use the JPEG entropy-decoding process to reconstruct back all quantized-blocks $\{F\}$. For each 8×8 quantized-block F, let $F(i, j)$ denote its element at ith row and jth column. Then, for each quantized-block F, calculate its intensity measure

$$G_F = \sqrt{\left(\sum_{i=0}^{7}\sum_{j=0}^{7} F^2(i,\, j)\right) - F^2(0,0)}. \tag{1}$$

Now, if $G_F < T$ for a predefined threshold value T, then the current block F is called a uniform block. Otherwise, F is a nonuniform block. Then, for each quantized-block

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)

16	11	10	16	1	1	1	1
12	12	14	1	1	1	1	55
14	13	1	1	1	1	69	56
14	1	1	1	1	87	80	62
1	1	1	1	68	109	103	77
1	1	1	64	81	104	113	92
1	1	78	87	103	121	120	101
1	92	95	98	112	100	103	99

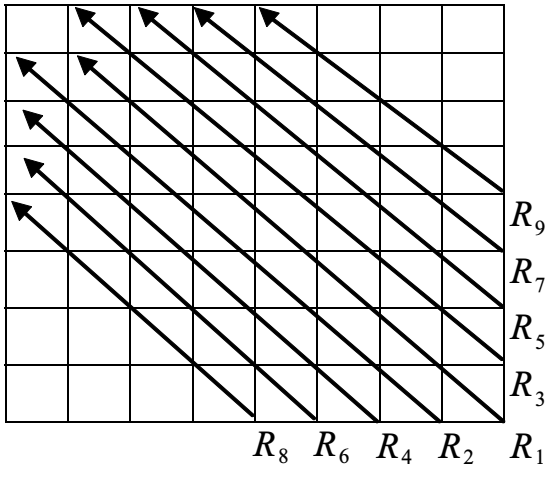
(b)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	1

(c)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	17	28	40	48	56
14	17	22	20	36	61	56	43
18	22	26	39	48	76	72	54
24	25	39	45	57	73	79	64
49	64	55	61	72	85	84	71
72	92	95	69	78	70	72	70

(d)



(e)

Fig. 1. (a) The quantization table used in Jsteg.[22] (b) The modified quantization table used in Chang *et al.*'s method.[3] Note that the 26 quantized-coefficients, which use the shaded area here in quantization process, are the 26 coefficients to hide data. (c) The modified quantization table MQ used in Tseng and Chang's method.[21] (d) The modified quantization table MQ used in Chang *et al.*'s reversible method.[4] (e) The nine sets $\{R_i : 1 \leq i \leq 9\}$ of coefficients are used to embed the secret in Ref. 4.

F, by using a user-defined capacity factor α and the modified quantization table MQ in Fig. 1(c); people can evaluate the capacity table $C = C_F$ corresponding to that block F, which in turn enables people to hide secret data in the elements of block F. (So the capacity table C_F is block-dependent.) To determine the table C_F for a quantized-block F, if (i, j) is neither (0,0) nor (7,7), C_F (i, j) is defined as

$$C_F(i, j) = \min\{\lfloor \log_2(c \times \alpha \times MQ(i, j))\rfloor, \lfloor \log_2|F(i, j)|\rfloor\} \tag{2}$$

where $0 \le i, j < 8$ and c depends on block type (c is 1 if this block F is a uniform block; c is greater than 1 otherwise). The $C_F(0,0)$ is specially defined in Ref. 21 as

$$C_F(0,0) = \min\left\{\left\lfloor \log_2 \frac{c \times \alpha \times MQ(0,0) \times 2}{\log_2(100 - QF)}\right\rfloor, \lfloor \log_2|F(i, j)|\rfloor\right\} \tag{3}$$

where QF (quality factor) is a JPEG-parameter specified by users to control image quality. For the first 63 quantized-coefficients of the current 8×8 quantized-block F, i.e. for each $(i, j) \neq (7, 7)$, after determining the capacity value $C_F(i, j)$ of the quantized-coefficient $F(i, j)$, a sector of secret data with length $C_F(i, j)$ is embedded into the quantized-coefficient $F(i, j)$. (No embedding if $|F(i, j)| \le 1$. These coefficients $F(i, j)$ are bypassed.) As for the final element $(i, j) = (7, 7)$ of the block, a flag bit is embedded in $F(7, 7)$ (0 means "uniform block", and 1 means "nonuniform block"). Finally, apply JPEG entropy-encoding to the current (embedded) quantized-block.

In Chang *et al.*'s reversible hiding method,[4] DCT is applied to transform each 8×8 block of image into the 8×8 DCT coefficients. Next, for each block, the 8×8 DCT coefficients are quantized using a modified quantization table shown in Fig. 1(d). After the 8×8 quantized DCT coefficients have been obtained, the secret data are embedded into the successive zero sequence of each set R_i $(1 \le i \le 9)$ shown in Fig. 1(e). (Notably, among the 8×8 quantized DCT coefficients of a typical image block, many quantized DCT coefficients in the lower right portion of Fig. 1(e) are zeros. That is why people can try to find successive zero sequence of each set R_i.) Now, to embed secret data in the successive zero sequence of each set R_i, the value of $count_i$ that denotes the length of the successive zero sequence from the lower right to the upper left in each set R_i has to be calculated first. In general, if the value of $count_i$ is larger than 1, one secret bit is embedded in the set R_i; otherwise, no embedding in R_i. Now, for each $count_i$ whose value is greater than 1, find the component r_{i,k_i} that is located in the upper left of the successive zero sequence of the set R_i. Then, the to-be-hidden secret bit s_i is embedded in lower right neighbor component r_{i,k_i-1} of r_{i,k_i}, by the embedding formula

$$r_{i,k_i-1} = \begin{cases} 0, & \text{if } s_i \text{ is } 0, \\ 1 \text{ or } -1, & \text{if } s_i \text{ is } 1, \end{cases} \tag{4}$$

where 1 or −1 is randomly selected. Finally, the JPEG entropy coding is applied to encode the current (embedded) quantized-block. After sequentially processing all quantized-blocks, the JPEG stego-code is generated.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

3. The Proposed Method

Section 3.1 gives the basic definitions; Sec. 3.2 is for hiding; Sec. 3.3 is for secret extraction and JPEG-code reversibility.

3.1. *Idea (mapping quantized-coefficients to stego quantized-coefficients)*

Assume that the JPEG (compression) code of an image is given, and we plan to embed secret data in it. Recall that in the quantization process of JPEG, each DCT coefficient $D(i, j)$ is divided (i.e. quantized) by its corresponding quantizer $Q(i, j)$ which is grabbed from a quantization table and then rounded to the nearest integer $F(i, j)$. This integer $F(i, j)$ is called the quantized-coefficient for DCT. Hence,

$$F(i, j) = \text{Round}(D(i, j)/Q(i, j)) \tag{5}$$

where $0 \leq i, j < 8$. In order to hide data in each quantized-coefficient $F(i, j)$, we introduce below a mapping that maps $F(i, j)$ to a stego quantized-coefficient $\tilde{F}(i, j)$.

As shown in Fig. 2, consider a line segment $\overline{\text{ab}}$ in real axis containing the integer point $F(i, j)$ and its two nearby half-unit-away noninteger points $a = F(i, j) - 0.5$ and $b = F(i, j) + 0.5$. By changing the scaling factor from the value $Q(i, j)$ to another value called $\tilde{Q}(i, j)$ (the value of $\tilde{Q}(i, j)$ is evaluated by the Eqs. (11) and (14) mentioned later), i.e. by mapping each real number t to an integer

$$h(t) = \lfloor t \times Q(i, j)/\tilde{Q}(i, j) \rfloor, \tag{6}$$

we get a non-decreasing mapping h that maps [a,b) into [A,B). Here, the two "noninteger" end points $a = F(i, j) - 0.5$ and $b = F(i, j) + 0.5$ are required to be mapped, respectively, to the "integers" $A = A(i, j) = h(a) = h(F(i, j) - 0.5)$ and $B = B(i, j) = h(b) = h(F(i, j) + 0.5)$.

3.1.1. *About $\tilde{Q}(i, j)$ appearing in the mapping h*

For the purpose of perfect recovery of the JPEG quantized-coefficient $F(i, j)$, we wish that "all" integers in the range $[A(i, j), B(i, j))$ can later be traced back to the "unique" integer $F(i, j)$ by the formula $\text{Round}(P \times \tilde{Q}(i, j)/Q(i, j))$. In other words, for any integer P in the range $A(i, j) \leq P < B(i, j)$, we require that the identity $\text{Round}(P \times \tilde{Q}(i, j)/Q(i, j)) = F(i, j)$ always holds.

From the $\tilde{Q}(i, j)$ chosen by our formula (14), we can use Eqs. (7) and (8) below to prove that $\text{Round}(P \times \tilde{Q}(i, j)/Q(i, j)) = F(i, j)$ really holds for each integer P in

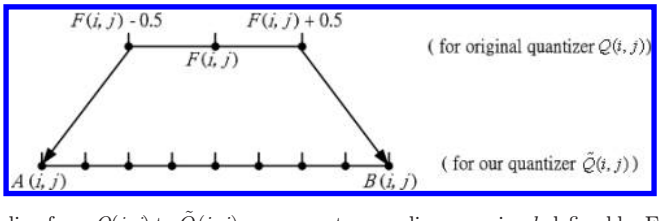


Fig. 2. By rescaling from $Q(i, j)$ to $\tilde{Q}(i, j)$, we can get a rescaling mapping h defined by Eq. (6) that maps, respectively, the two noninteger points $F(i, j) - 0.5$ and $F(i, j) + 0.5$ to two "integer" points $A(i, j)$ and $B(i, j)$.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

the range $A(i, j) \le P < B(i, j)$. (The proof is that: $P \times \tilde{Q}(i, j)/Q(i, j) \ge A(i, j) \times \tilde{Q}(i, j)/Q(i,j) \ge \{(F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j)\} \times \tilde{Q}(i, j)/Q(i, j) = F(i, j) - 0.5$ by Eq. (7). Similarly, Eq. (8) implies $P \times \tilde{Q}(i, j)/Q(i, j) < B(i, j) \times \tilde{Q}(i, j)/Q(i, j) \le \{(F(i, j) + 0.5) \times Q(i, j)/\tilde{Q}(i, j)\} \times \tilde{Q}(i, j)/Q(i, j) = F(i, j) + 0.5$. So, $F(i, j)-0.5 \le P \times \tilde{Q}(i, j)/Q(i, j) < (F(i, j) + 0.5)$, which ensures Round$(P \times \tilde{Q}(i, j)/Q(i, j))= F(i, j)$ because Eq. (5) means that $F(i, j)$ is an integer.)

3.1.2. *Candidates and final winner for the stego quantized-coefficient* $\tilde{F}(i, j)$

Since we wish that all integer points in $[A(i, j), B(i, j))$ can later be traced back to the unique integer $F(i, j)$, we may say that all integer points in the half-open interval $[A(i, j), B(i, j))$ are candidates for the stego quantization-coefficient value $\tilde{F}(i, j)$. To determine which of these candidate points should be picked as the final value of $\tilde{F}(i, j)$, we just inspect the secret data to be hidden. More specifically, each integer $\{A, A + 1, A + 2, \ldots, B-2, B-1\}$ in the half-open interval $[A, B) = [A(i, j), B(i, j))$ corresponds to a hidden value; for example, $\tilde{F}(i, j) = A$ means that the hidden secret is 000, $\tilde{F}(i, j) = A + 1$ means that the hidden secret is 001, $\tilde{F}(i, j) = A + 2$ means that the hidden secret is 010, $\tilde{F}(i, j) = A + 3$ means that the hidden secret is 011, etc.

From the mapping $h(t) = \lfloor t \times Q(i, j)/\tilde{Q}(i, j) \rfloor$ defined in Eq. (6), where $0 \le i, j < 8$, the points $a = F(i, j)-0.5$ and $b = F(i, j) + 0.5$ are mapped, respectively, to

$$A(i, j) = \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor, \tag{7}$$

$$B(i, j) = \lfloor (F(i, j) + 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor. \tag{8}$$

Notably, according to JPEG standard, the values of $Q(i, j)$ and $\tilde{Q}(i, j)$ must be integers between 1 and 255. Also, if we intend to use the final value of the stego quantization-coefficient $\tilde{F}(i, j)$ to indicate one of the 2^n possible readings of an n-bit hidden secret, then $\tilde{F}(i, j)$ must have at least 2^n possible candidate values. Hence, the half-open interval $[A(i, j), B(i, j))$ in Fig. 2 must contain at least 2^n integers.

In the decoding part, after the secret data have been extracted, in order to recover the original JPEG code, we need to recover the original quantized-coefficient $F(i, j)$ from the stego quantized-coefficient $\tilde{F}(i, j)$. This can be done by applying the $\tilde{Q}(i, j)$-to-$Q(i, j)$ inverse-scaling to reverse $\tilde{F}(i, j)$ back to a real value $\tilde{F}(i, j) \times \tilde{Q}(i, j)/Q(i, j)$. This real number is near $F(i, j)$. In fact, according to the design shown earlier in Fig. 2, the real number will be in the interval $[F(i, j) - 0.5, F(i, j) + 0.5)$, and it can be rounded to the nearest integer as the original $F(i, j)$. In short, $F(i, j)$ can be recovered by

$$F(i, j) = \text{Round}(\tilde{F}(i, j) \times \tilde{Q}(i, j)/Q(i, j)) \tag{9}$$

for each $0 \le i, j < 8$. Finally, to extract the secret data hidden in position (i, j), the value $F(i, j) - 0.5$ can be used to find the value $A(i, j)$ by Eq. (7). Then the decimal value $\tilde{X}(i, j)$ equivalent to the binary hidden secret bits is extracted by

$$\tilde{X}(i, j) = \tilde{F}(i, j) - A(i, j). \tag{10}$$

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

3.2. *Our embedding process*

***Main algorithm for embedding*:**

Input: The data to be hidden; the JPEG compression code of a cover image; two user-defined integer parameters $level \in \{1, 2, 3\}$ and $num \in \{1, 2, \ldots, 32\}$.

Pre-processing: Apply JPEG entropy decoding to the given JPEG code and get all quantized-blocks $\{F\}$ (of size 8×8 each), and the 8×8 quantization table Q.

Main phases:

Phase 1: Plug the *level* value in Eq. (11) to construct an 8×8 "hiding-capacity table" called HC from elements of table Q. (HC is calculated only once; i.e. just like *level* and *num*, HC is also universal for all blocks of the given image.) Now, use Q and HC to create the 8×8 modified quantization table $\tilde{Q}$ by Eq. (14).

Phase 2: For each quantized-block F, sequentially grab and embed the not-hidden-yet data bits in the first *num* quantized-coefficients of F. This transforms F to $\tilde{F}$. Then, encode each stego-block $\tilde{F}$ by JPEG entropy coding.

Output: Output the JPEG stego-code, i.e. the header $\tilde{Q}$ and the sequence of the stego-blocks $\tilde{F}$. (Also include parameter *num* and original table Q for secret-extraction purpose.) – END –

Below we discuss the details about the Tables-Construction phase (Phase 1) and Data-Embedding phase (Phase 2).

***Details about Phase 1.* (*Construction of* 8×8 *hiding-capacity table HC and* 8×8 *modified quantization table* $\tilde{Q}$)**

For $0 \le i, j < 8$, let $HC(i, j)$, $\tilde{Q}(i, j)$, and $Q(i, j)$ denote, respectively, the element at ith row and jth column of the 8×8 hiding-capacity table HC, our modified quantization table $\tilde{Q}$, and the original JPEG quantization table Q. All these three tables are universal, i.e. the same tables are used for all blocks of the image. Notably, the block-independency of HC is quite different from the block-dependency of the hiding table $C = C_F$ used in Ref. 21, as mentioned in the fourth paragraph in Sec. 2.

Let $HC(i, j)$ be the number of secret bits to be embedded in JPEG quantized-coefficient $F(i, j)$ to get our stego quantized-coefficient $\tilde{F}(i, j)$. The value of each integer $HC(i, j)$ is randomly chosen from the range

$$\min\{level, \ \lfloor \log_2 Q(i, j) \rfloor\} \le HC(i, j) \le \lfloor \log_2 Q(i, j) \rfloor \tag{11}$$

Below we explain why $HC(i, j)$ should depend on the original JPEG quantization table Q and the user-defined integer parameter *level*.

As proved later in the ***Range Property***, Eqs. (7) and (8) imply that $\lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor \le B(i, j) - A(i, j) \le \lceil Q(i, j)/\tilde{Q}(i, j) \rceil$. Therefore, in the half-open interval $[A(i, j), B(i, j))$, there are at most $\lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor$ integer points. As mentioned in

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Sec. 3.1, we will only use integer points as candidates for $\tilde{F}(i, j)$. To embed $HC(i, j)$ binary-bits in $F(i, j)$ to get $\tilde{F}(i, j)$, there are $2^{HC(i,j)}$ possible combinations of the input secret bits, i.e. $00\cdots000$; $00\cdots001$; $00\cdots010$; etc. The embedding of each combination corresponds to one of the integer points located in the half-open interval $[A(i, j), B(i, j))$. Therefore,

$$2^{HC(i,j)} \leq \lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor \tag{12}$$

is required for each $0 \leq i, j < 8$. Now, because both the values of $\tilde{Q}(i, j)$ and $Q(i, j)$ are integers between 1 and 255, we have $\lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor \leq \lfloor Q(i, j) \rfloor \leq Q(i, j)$, and therefore

$$2^{HC(i,j)} \leq Q(i, j). \tag{13}$$

This explains why $Q(i, j)$ appears in Eq. (11) to evaluate $HC(i, j)$. As for the integer parameter *level*, it also appears in Eq. (11) because if we directly let $HC(i, j) = \lfloor \log_2 Q(i, j) \rfloor$; then by Eq. (14), the corresponding $\tilde{Q}(i, j)$ in the modified quantization table $\tilde{Q}$ will be $\lfloor Q(i, j)/2^{HC(i,j)} \rfloor = \lfloor Q(i, j)/2^{\lfloor \log_2 Q(i,j) \rfloor} \rfloor = 1$. Then the first *num* elements in the modified quantization table $\tilde{Q}$ are all 1, and attackers may feel suspicious. On the other hand, letting $HC(i, j) \leq \lfloor \log_2 Q(i, j) \rfloor$ will give us more choices; and, by giving $HC(i, j)$ a lower bound involving *level*, we can control the hiding rate, as will be shown later in Tables 1–3.

Finally, to determine the 8×8 modified quantization table $\tilde{Q}$, Eq. (12) implies $2^{HC(i,j)} \leq \lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor \leq Q(i, j)/\tilde{Q}(i, j)$, and because both $Q(i, j)$ and $\tilde{Q}(i, j)$ are positive integers, we have $\tilde{Q}(i, j) \leq Q(i, j)/2^{HC(i,j)}$. Therefore, the $\tilde{Q}(i, j)$ of our modified quantization table $\tilde{Q}$ is simply defined as

$$\tilde{Q}(i, j) = \lfloor Q(i, j)/2^{HC(i,j)} \rfloor, \tag{14}$$

where $0 \leq i, j < 8$. Equation (14) also implies that

$$HC(i, j) = \lfloor \log_2(Q(i, j)/\tilde{Q}(i, j)) \rfloor. \tag{15}$$

Range Property: $\lfloor Q(i, j)/\tilde{Q}(i, j) \rfloor \leq B(i, j) - A(i, j) \leq \lceil Q(i, j)/\tilde{Q}(i, j) \rceil$

Proof. (i) Proof of the upper bound:

$$\begin{aligned}
B(i, j) - A(i, j) &= \lfloor (F(i, j) + 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&\quad - \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&= \lfloor (F(i, j) - 0.5 + 1) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&\quad - \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&= \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) + Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&\quad - \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&\leq \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor + \lceil Q(i, j)/\tilde{Q}(i, j) \rceil \\
&\quad - \lfloor (F(i, j) - 0.5) \times Q(i, j)/\tilde{Q}(i, j) \rfloor \\
&= \lceil Q(i, j)/\tilde{Q}(i, j) \rceil.
\end{aligned}$$

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

(ii) Proof of the lower bound:

$$\begin{aligned} B(i,j) - A(i,j) &= \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &\quad - \lfloor (F(i,j)-0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &= \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &\quad - \lfloor (F(i,j)+0.5-1)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &= \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &\quad - \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j) - Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &\geq \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &\quad - \lfloor (F(i,j)+0.5)\times Q(i,j)/\tilde{Q}(i,j)\rfloor + \lfloor Q(i,j)/\tilde{Q}(i,j)\rfloor \\ &= \lfloor Q(i,j)/\tilde{Q}(i,j)\rfloor. \end{aligned}$$

Details about Phase 2. (Hiding secret data in quantized-blocks $\{F\}$ to get $\{\tilde{F}\}$)

As stated earlier, the HC and $\tilde{Q}$ motioned below are block-independent and will be used universally for all 8×8 blocks of the given image.

Input: (i) The JPEG compression code of a cover image. (ii) The values of two integer parameters *level* ($1 \leq level \leq 3$) and *num* ($1 \leq num \leq 32$).

Step 1. Decode the JPEG compression code using JPEG entropy decoding to get (i) an 8×8 quantization table Q, and (ii) "all" quantized-blocks $\{F\}$ formed of 64 quantized-coefficients each.

Step 2. Use Eqs. (11) and (14) to compute the first *n*um elements of the 8×8 hiding-capacity table HC and the 8×8 modified quantization table $\tilde{Q}$. The values of other 64–*num* entries of HC are set to zeros, and the values of other 64–*num* entries of $\tilde{Q}$ are copied from the corresponding 64–*num* entries of Q.

Step 3. Repeat Steps 3a–3b below in a block-by-block manner, until all quantized-blocks $\{F\}$ are inspected.

Sub-step 3a. For the first *num* quantized-coefficients $\{F(i,j)\}$ of an 8×8 quantized-block F currently being inspected, use Eq. (7) to get $A(i,j)$ from $F(i,j)$. Then sequentially grab next $HC(i,j)$ not-yet-embedded bits from the secret data, and let $X(i,j)$ be its decimal-value equivalent. Compute the desired stego quantized-coefficient $\tilde{F}(i,j)$ by

$$\tilde{F}(i,j) = A(i,j) + X(i,j). \tag{16}$$

Then overwrite $F(i,j)$ by $\tilde{F}(i,j)$ because $F(i,j)$ is no longer needed.

Sub-step 3b. Apply JPEG entropy coding to the embedded quantized-block $\tilde{F}$.

Step 4. Use the modified quantization table $\tilde{Q}$ as the JPEG file header. This completes the generation of the JPEG stego-code. – END –

After the JPEG stego-code has been generated using the preceding procedure, the original quantization table Q and user-defined parameter *num* should also be transmitted together with the produced JPEG stego-code to the receiver. By doing this, the receiver can know both Q and its modified table $\tilde{Q}$, which are essential for decoding.

3.3. *Decoding* (*Extraction of hidden data and lossless recovery of JPEG compression code*)

When a user receives our JPEG stego-code (consisting of a modified quantization table $\tilde{Q}$ and the entropy-coding results of a sequence of stego quantized-blocks $\{\tilde{F}\}$), if he also knows the original JPEG quantization table Q and the value of integer parameter *num*, then he can extract the secret data, followed by reconstructing the original JPEG-compression code. Details are as follows:

***Secret Extraction and JPEG-Compression-Code Reversibility Algorithm*:**

Input: (i) Our JPEG stego-code (consisting of a header, which is an 8×8 modified quantization table $\tilde{Q}$, and the entropy-coding-result of a sequence of stego quantized-blocks $\{\tilde{F}\}$). (ii) Original 8×8 JPEG quantization table Q. (iii) The value of the integer parameter *num* ($1 \leq num \leq 32$).

Step 1. Get the 8×8 modified quantization table $\tilde{Q}$. Then apply JPEG entropy decoding to JPEG stego-code, and thus get all stego quantized-blocks $\{\tilde{F}\}$.

Step 2. Recover the first *num* entries of the 8×8 hiding-capacity table *HC* by Eq. (15).

Step 3. Repeat Steps 3a–3b below in a block-by-block manner until all stego quantized-blocks $\{\tilde{F}\}$ are inspected.

Step 3a. For the first *num* stego quantized-coefficients $\{\tilde{F}(i, j)\}$ of the 8×8 stego quantized-blocks $\tilde{F}$ currently being inspected, inverse transform each $\tilde{F}(i, j)$ back to $F(i, j)$ by Eq. (9). Then transform $F(i, j) - 0.5$ to $A(i, j)$ by Eq. (7). Extract the decimal equivalent $\tilde{X}(i, j)$ of the secret bits by Eq. (10), i.e. $\tilde{X}(i, j) = \tilde{F}(i, j) - A(i, j)$. Then overwrite $\tilde{F}(i, j)$ by $F(i, j)$. So we obtain not only the secret data $\tilde{X}(i, j)$ hidden in $\tilde{F}(i, j)$, but also the original JPEG quantized-coefficient $F(i, j)$ at current block position.

Step 3b. Other than the original JPEG quantized-coefficients (at current block position) which can be used to reconstruct JPEG image immediately, if JPEG code is also wanted for recycling-use or economic-storage reason, then encode the current (secret-already-extracted) quantized-block F using JPEG entropy coding.

Step 4. Put the original quantization table Q in the header of the JPEG file. This completes the reconstruction of the original JPEG compression code, without any loss.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

4. Experimental Results, Comparisons, and Discussions

4.1. *Experimental results and comparisons*

Firstly, six 512×512 gray-level images, Lena, Jet, Boat, Peppers, Baboon, and Zelda, are tested in our experiments. As a preprocessing, they are all compressed using the JPEG source code provided by the fourth public release of the Independent JPEG Group's free JPEG software. The quality factor QF (between 0 and 100) in that JPEG software is used to adjust the quality of the decompressed image. As known by every user, the higher the QF, the larger the created JPEG file (and the better quality of the decompressed image). The to-be-hidden secret data are a random bit stream. The peak-signal-to-noise ratio

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{\text{MSE}} \tag{17}$$

is used to measure the stego-image quality where MSE denotes the mean square error between the pixel values of the original image (without compression) and the decoded stego-image.

Figures 3(a) and 3(d) show, respectively, the decompressed JPEG images of Lena and Jet when no hiding method is used. Figures 3(b) and 3(e) show, respectively, our decompressed JPEG stego-images after the 248 K bits of the secret data are embedded

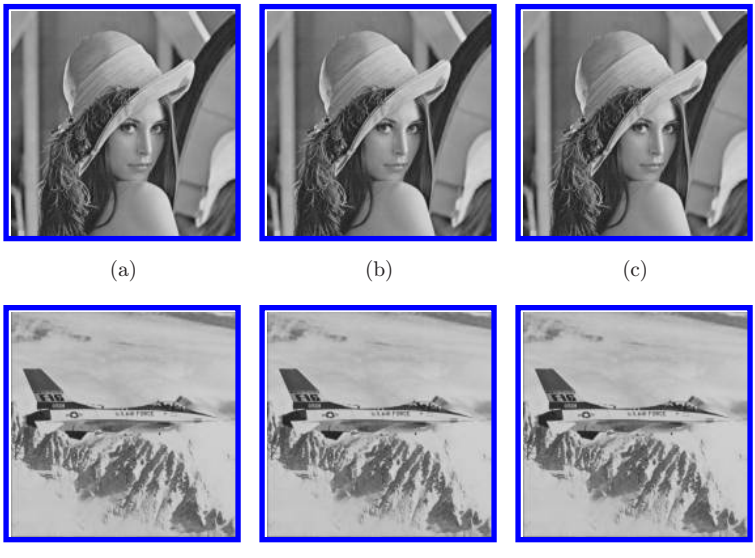


Fig. 3. Images decompressed from JPEG codes. (a) The 41.75 dB JPEG image Lena decompressed from the JPEG code without any hidden secret. (b) The 39.12 dB Lena decompressed from our JPEG stego code after hiding a secret of size 253,952 bits. (c) The decrypted–decompressed Lena (identical to (a)) derived from our JPEG stego code. (d)–(f) Same as (a)–(c) except that the image Lena is replaced by Jet, and the PSNRs of (d) to (f) are 41.15 dB, 38.75 dB, and 41.15 dB, respectively.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

into the JPEG codes of Lena (or 248 K bits into Jet) when our parameters are $level = 2$ and $num = 32$. Figures 3(c) and 3(f) show our "decrypted−decompressed" images from Figs. 3(b) and 3(e), respectively. The quality factor $QF = 85$ is adopted for all figures in Fig. 3.

Table 1 (for image Lena) and Table 2 (for image Jet) give a comparison of our reversible JPEG-based method with nonreversible JPEG-based methods[3,21,22] when the input is a JPEG-QF85 code (created when a gray-value image is compressed using JPEG with JPEG's quality factor $QF = 85$). For simplicity, in the 8×8 hiding capacity table HC, set each entry value $HC(i, j)$ to the left-hand-side value of Eq. (11). The comparisons in Table 1 include hiding capacity (size of hidden secret), hiding ratio (size of hidden secret over size of JPEG stego-code), and the PSNR value of the stego-image. It is observed that our method provides higher hiding capacity, better hiding ratio, while maintaining acceptable quality of stego-images. For example, Table 1 shows that we get PSNR = 39.12 when hiding capacity is 253,952 bits with hiding ratio 35.23%; while Ref. 3 gets PSNR = 29.64 (<39.12) when hiding capacity is 212,992 bits (<253,952 bits) with hiding ratio being 30.0% (<35.23%). Similarly, we can get PSNR = 41.11 when hiding capacity is 122,880 bits and hiding ratio is 23.16%; while Ref. 22 gets PSNR = 40.70 (<41.11) when hiding capacity was 31,933 bits (<122,880 bits) with hiding ratio being 8.85% (<23.16%). Similarly, Ref. 21 gets PSNR = 38.10 (<41.11) when hiding capacity is 54,652 bits (<122,880 bits) with hiding ratio being 14.0% (<23.16%). Most of all, after the secret data have

Table 1. A comparison of our reversible method with some *nonreversible* JPEG-based methods[3,21,22] when the gray-value image Lena is compressed using JPEG with the quality factor $QF = 85$. All numerical data (except ours) are quoted from Ref. 21 directly. Notably, our "decrypted−decompressed" JPEG image is 100% identical to the decompressed JPEG image because of our reversibility.

	User-Defined Parameter (*num*)	User-Defined Parameter (*level*)	Capacity Factor (α)	Hiding Capacity (bits)	Hiding Ratio (%)	Stego PSNR (dB)
JPEG						41.75
Jsteg[22]				31,933	8.85%	40.70
Chang *et al.*[3]				212,992	30.0%	29.64
Tseng and Chang[21]			0.18	33,790	8.68%	40.21
			0.36	51,161	13.2%	38.91
			0.5	54,652	14.0%	38.10
			0.75	62,612	16.1%	36.87
			1.0	63,083	16.2%	36.56
Ours	16	1		65,536	14.78%	40.99
	24	1		98,304	19.74%	40.10
	32	1		131,072	23.33%	38.22
	16	2		122,880	23.16%	41.11
	24	2		188,416	30.01%	40.53
	32	2		253,952	35.23%	39.12
	16	3		126,976	23.60%	41.13
	24	3		208,896	31.76%	40.49

Table 2. Similar to Table 1, except that the image Lena is replaced by the image Jet.

	User-Defined Parameter (*num*)	User-Defined Parameter (*level*)	Capacity Factor (α)	Hiding Capacity (bits)	Hiding Ratio (%)	Stego PSNR (dB)
JPEG						41.15
Jsteg[22]				32,868	9.11%	40.07
Chang *et al.*[3]				212,992	29.9%	28.16
Tseng and Chang[21]			0.18	35,678	9.26%	39.38
			0.36	53,532	13.9%	38.12
			0.5	58,254	15.1%	37.32
			0.75	65,457	17.0%	36.29
			1.0	66,363	17.2%	35.77
Ours	16	1		65,536	14.70%	40.47
	24	1		98,304	19.58%	39.67
	32	1		131,072	23.24%	37.91
	16	2		122,880	23.03%	40.58
	24	2		188,416	29.74%	40.06
	32	2		253,952	35.01%	38.75
	16	3		126,976	23.44%	40.60
	24	3		208,896	31.43%	40.03

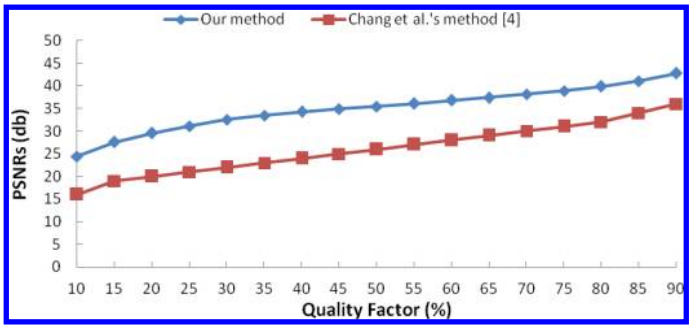
been extracted, our method can reconstruct the original JPEG-QF85 code, but Refs. 3, 21 and 22 cannot.

Having compared our method with nonreversible methods,[3,21,22] we compare below with other reversible methods. Both Refs. 4 and 6 are reversible. However, the original purpose in Ref. 6 was to authenticate the JPEG compression code (the hidden secret is a much smaller authentication watermark [about 4 K bits], rather than our larger-size secret [can be as large as 248 K bits]), and hence not in the same application group as ours or Refs. 3, 4, 21 and 22. Therefore, we omit the comparison with Ref. 6.

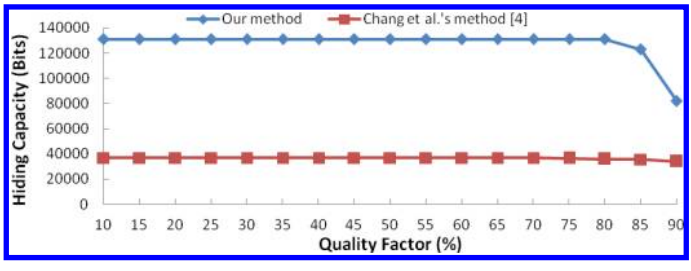
Figure 4 compares two reversible methods: ours and in Ref. 4. Figures 4(a)–4(c), respectively, compare stego-image quality, hiding capacities (size of the secret hidden in the JPEG stego-code), and hiding ratios (size of the secret over size of the JPEG stego-code) for the gray-image Lena. Figure 4(d) compares the average size of the stego-codes created by Ref. 4 and by ours (the average is calculated among six images Lena, Jet, Boat, Peppers, Baboon and Zelda). The two parameters used in our method are $level = 2$ and $num = 16$. As shown in Figs. 4(a)–4(c), our stego-image quality, hiding capacity, and hiding ratio are all better than those in Ref. 4. Although we win in Figs. 4(a)–4(c), our code size would be less attractive than that in Ref. 4. This is shown in Fig. 4(d). The explanation is as follows. In general, larger *level* value is suitable only if we have a very big-size secret (such as those in Fig. 4(b)) because $level = 2$ gives us higher hiding-capacity and higher hiding-ratio at the price of longer output code. (This trade-off is still worthy and more efficient because hiding ratio is high. It is just like paying more money (1.2 to 1.6 times more in Fig. 4(d)) to get much more goods (could be three times more, see Fig. 4(b)).

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

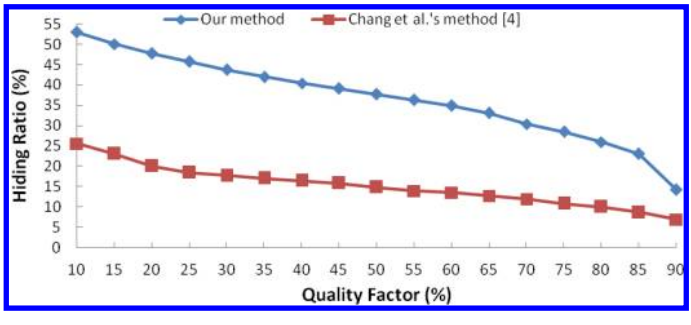
Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.



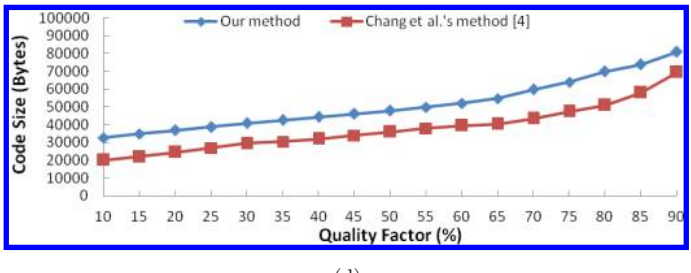
(a)



(b)



(c)



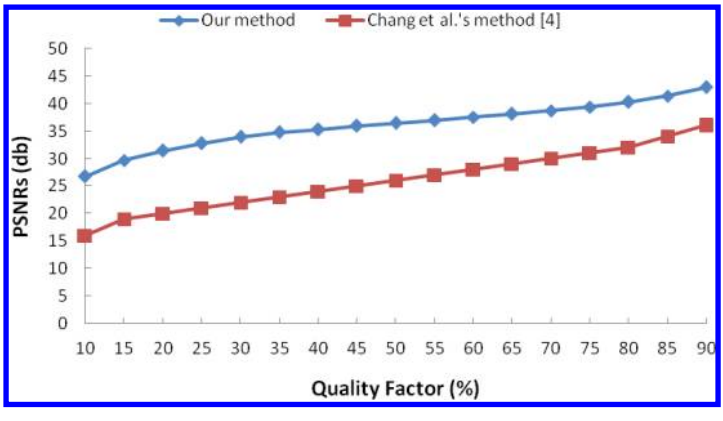
(d)

Fig. 4. Comparing two reversible methods: Ref. 4 and ours (when our parameters are $level = 2$ and $num = 16$). The comparisons are (a) stego-images' quality, (b) hiding capacities, (c) hiding ratio, and (d) average size of the obtained JPEG stego-codes. (a)−(c) are for the gray-value image Lena; while the average in (d) is taken among six gray-value images {Lena, Jet, Boat, Peppers, Baboon, and Zelda}.

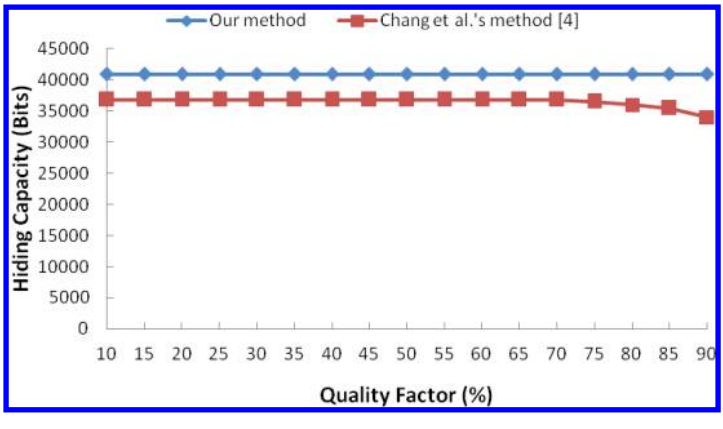
If a user wants to reduce the output code size, he may reduce the not-small gains in hiding capacity and hiding ratio by using lower *level* value (i.e. let $level = 1$) and lower *num* value. One such example is given in Fig. 5 where the two parameters used by our method are $level = 1$ and $num = 10$. It is observed that our three curves in Figs. 5(a)−5(c) are still a little bit more competitive than those in Ref. 4 while our code-size curve in Fig. 5(d) gets a tie with those in Ref. 4. In all experiments above, when we count our stego-code size, we already include the space for the parameter *num* and two quantization tables Q and $\tilde{Q}$.

4.2. *Parameters setting and security analysis*

From the discussion in the final two paragraphs of Sec. 4.1, we have the following rules to determine the two parameter values. As a key rule, $level = 1$ is for secrets of moderate size, $level = 2$ or 3 is for secrets of very large size. After the *level* value has been



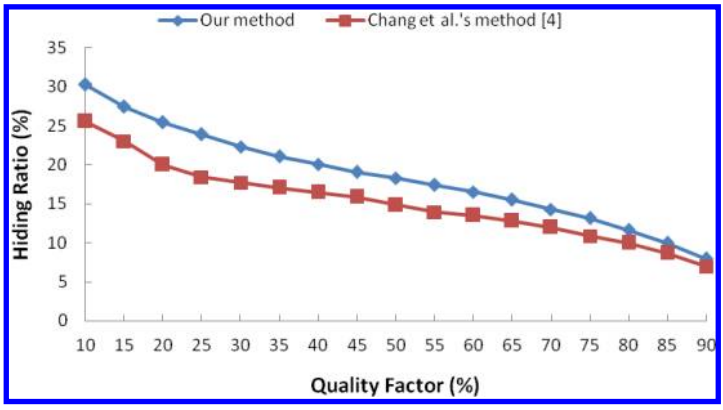
(a)



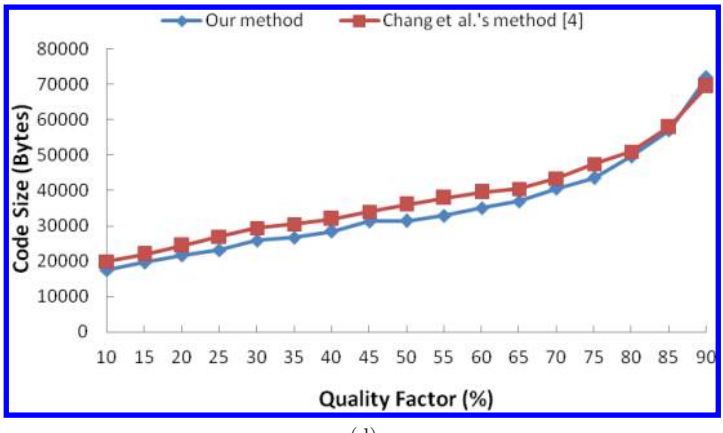
(b)

Fig. 5. Same as Fig. 4 (but the two parameters are now $level = 1$ and $num = 10$).

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.



(c)



(d)

Fig. 5. (*Continued*)

determined, then adjust value of *num*. For a fixed *level* value, larger *num* value means more hiding capacity and also means more efficient hiding ratio, but the code size will increase and the PSNR value of the stego-image will slightly go down. Therefore, when the *num* value is adjusted, start from a moderate num value such as $num = 16$. If the secret is too large to be hidden in, then increase *num* value, else decrease *num* slightly to reduce the size of the output JPEG stego-code as long as the secret can still be hidden in after the cut of *num* value.

In the proposed method, under the same *num* value, using a higher *level* to force the use of higher $HC(i, j)$ values [see Eq. (11)] can yield higher hiding-capacity (than using lower *level* and lower $HC(i, j)$ values) because $HC(i, j)$ means the hiding capacity (number of hidden bits can be embedded) in a quantized-coefficient. Higher $HC(i, j)$ values naturally increase the hiding capacity of the whole image. Also note that a little improvement (but not as obvious as the benefits in hiding capacity) of the stego-image quality might be obtained by using higher *level* value, especially when we compare the PSNR value of $level = 2$ with that of using $level = 1$.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

We may conduct another experiment to observe this phenomenon. Let the gray-value image Lena be compressed by JPEG using $QF = 75$. Then, the obtained JPEG code is utilized to hide the secret using various settings of *level* and *num* values. For simplicity, the value of each entry $HC(i, j)$ in the 8×8 hiding capacity table HC is set to be the left-hand side value of Eq. (11). From this assignment rule, using higher *level* value really gives higher hiding capacity, as shown in Table 3, from which we also see that the capacity and PSNR benefits are less obvious when *level* value jumps from 3 to 4. The reason is that the first 32 values of $\lfloor \log_2 Q(i, j) \rfloor$ are all in the range $\{1, 2, 3, 4\}$ when $QF = 75$ or 85, and according to the assignment rule, the chosen values of $HC(i, j)$ are almost identical no matter $level = 3$ or 4. (They all get $HC(i, j) = \lfloor \log_2 Q(i, j) \rfloor$ if $\lfloor \log_2 Q(i, j) \rfloor$ is 1 or 2 or 3. The only difference occurs when $\lfloor \log_2 Q(i, j) \rfloor = 4$. In that case, $level = 4$ implies $HC(i, j) = \lfloor \log_2 Q(i, j) \rfloor = 4$ while $level = 3$ allows $HC(i, j)$ can be 3 or 4 by Eq. (11). In summary, $HC(i, j)$ values are almost identical between the two cases $level = 3$ and $level = 4$.) But $HC(i, j)$ values can be very different between the two cases $level = 1$ and $level = 2$, as can be seen from Eq. (11). Since using $level = 4$ does not get much more hiding capacity than using $level = 3$ does, using *level* 4 is not recommended for security concern. As a summary, we recommend *level* values 2 and 3. (Skip the use of $level = 1$ if very large hiding capacity is required.) Also, for security concern again, if *level* value 3 is used, we recommend that the *num* value had better be at most 24 because $num = 32$ will abnormally enlarge the size of JPEG stego-code. (The stego-code will be 98,933 bytes for Lena if (*level, num*) = (3, 32), and hence does not stay in ordinary plain JPEG code's length range 8,119−94,581 bytes.)

Because our method is a steganography scheme, and, as suggested by one of the reviewers, for a steganography scheme, the security of secret data should be tested. Therefore, we run Provos's Stegdetect steganography test program[17] and Guillermito's

Table 3. The hiding capacity, hiding ratio, and stego-image quality of the proposed method with different *level* and *num* values. The gray-value image is the Lena compressed by JPEG with the quality factor $QF = 75$.

(*level*, *num*)	Hiding Capacity (bits)	Hiding Ratio (%)	Stego PSNR (dB)
(1, 16)	65,536	18.91%	38.56
(2, 16)	131,072	28.56%	38.87
(3, 16)	163,840	32.38%	38.94
(4, 16)	167,936	32.81%	38.96
(1, 24)	98,304	24.15%	37.09
(2, 24)	196,608	35.76%	37.78
(3, 24)	262,144	41.07%	37.91
(4, 24)	278,528	41.96%	37.96
(1, 32)	131,072	27.60%	34.48
(2, 32)	262,144	40.59%	35.59
(3, 32)	360,448	46.74%	35.95
(4, 32)	405,504	47.40%	35.81

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

Chi-Square test program[8] to test the security of our generated JPEG stego-codes. The Stegdetect attack is mentioned in Ref. 18 and used in Refs. 7, 11, 14 and 19. Our method can resist these attacks. (Both Chi-square and Stegdetect are tools that can be used by unauthorized users to detect whether or not a stego-image or a JPEG code contains some secret.)

For the Chi-square attack, Guillermito's Chi-square steganography test program[8] is used to perform statistical analysis. Figure 6(a) shows the 512×512 gray-level image Lena without hiding any secret. Figure 6(b) shows the stego-image obtained by embedding some secret into Fig. 6(a) using 1-LSB substitution method. Figure 6(c) shows our decompressed JPEG stego-image Lena (the secret is still in it and not yet extracted), when two parameters are $level = 2$ and $num = 32$. Figures 6(d)−6(f) show, respectively, the results when performing the Chi-square steganography test program in Figs. 6(a)−6(c). The cross curve checks the probability that pairs of values follow a random distribution, and the circular one represents the average value of all LSBs in one block of pixels. According to the examples in Sec. 3.5 in Ref. 8, each circular curve in Figs. 6(d)−6(f) will not cause unauthorized users' attention because each circular curve in Figs. 6(d)−6(f) is around $0.5 = (0 + 1)/2$.

However, the cross curve in Fig. 6(e) is close to one. This gives unauthorized users some information: very likely that some secret data were embedded in the corresponding stego-image (Fig. 6(b), the stego-image for 1-LSB method). To the contrary, both cross curves in Figs. 6(d) and 6(f) are close to zero. This gives unauthorized users the probabilities that some secret data in the original image [Fig. 6(a)] or in our stego-image [Fig. 6(c)] are both close to zero (hence, very low).

As for the Stegdetect attack, the 512×512 24-bit color image Lena is compressed by JPEG with $QF = 85$ and then used as the input JPEG code to embed the secret for several JPEG hiding methods (Refs. 13, 17, 22, 25 and ours). Then, Provos's Stegdetect steganography test program [17] is applied to detect whether the generated JPEG steg-codes (for Refs. 13, 17, 22 and 25 and ours) contain the secret or not.

Figure 7(a) is the decompressed color image Lena which does not hide any secret in the input JPEG code called "OriginalLena.jpg". When people use the hiding methods F5,[25] JPHide,[13] Jsteg,[22] and OutGuess[17] to hide data in "OriginalLena.jpg", let the created JPEG stego-codes be "Stego-of-F5.jpg", "Stego-of-JPHIDE.jpg", "Stego-of-JSTEG.jpg", and "Stego-of-OUTGUESSOLD.jpg", respectively. Figures 7(b)−7(e) show the four JPEG stego-images decompressed from these four JPEG stego-codes. Figures 7(f)−7(m) show our JPEG stego-images decompressed from our eight JPEG stego-codes (the secret is still in it and not yet extracted) when two parameters (*level*, *num*) are (1, 16), (1, 24), (1, 32), (2, 16), (2, 24), (2, 32), (3, 16), (3, 24), correspondingly. Here, our method uses only the quantized DCT coefficients of the brightness component Y of the input JPEG code "OriginalLena.jpg" to embed the secret. The color components Cb and Cr are not used in hiding to minimize the color distortion in the embedded image (as suggested by, or used in, Refs. 2, 4 and 10).

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

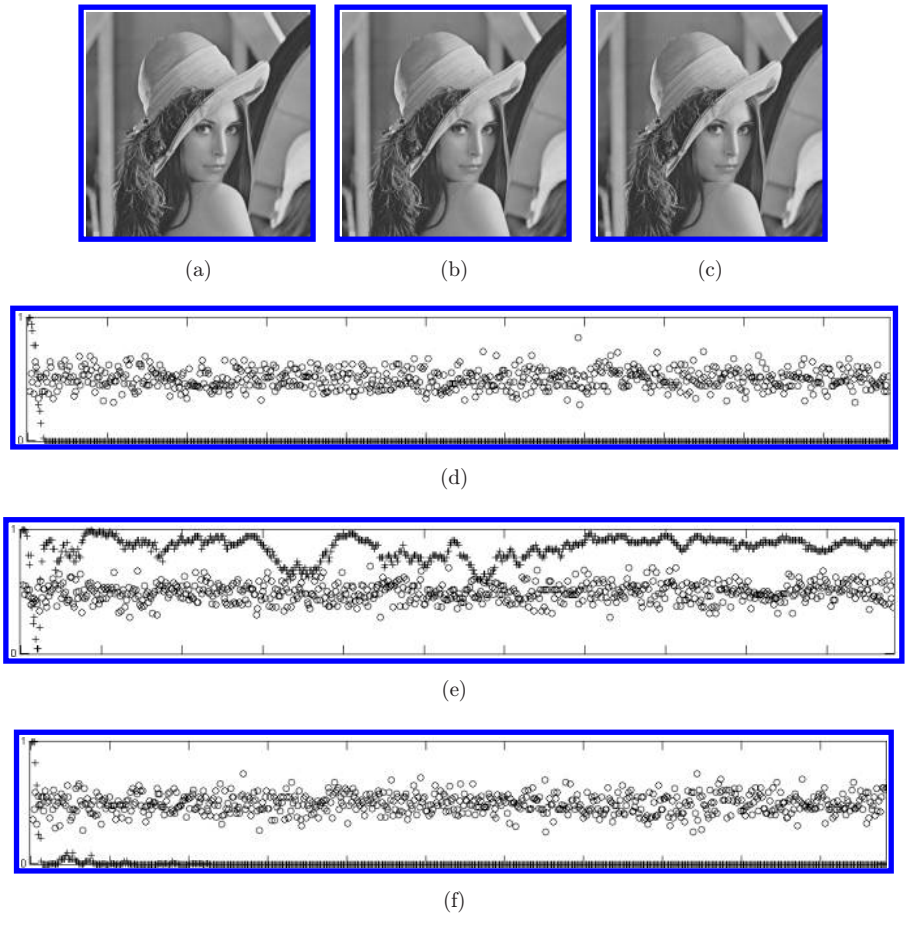


Fig. 6. An example of the statistical attack using Chi-square analysis. (a) Gray-level image Lena without hiding any secret; (b) the stego-image Lena when embedding secret using 1-bit LSB substitution method; (c) image decompressed from our JPEG stego-code Lena (the secret is still inside and not-yet extracted); (d) Chi-square result of the original image [Fig. 6(a)]; (e) Chi-square result of the 1-LSB stego-image [Fig. 6(b)]; (f) Chi-square result of our JPEG stego-image [Fig. 6(c)].

When we use the Stegdetect detection tool (introduced in Ref. 18 and used in Ref. 7, 11, 14 and 19) to analyze whether a secret is hidden in a JPEG code, four stego-codes "Stego-of-F5.jpg", "Stego-of-JPHIDE.jpg", "Stego-of-JSTEG.jpg", and "Stego-of-OUTGUESSOLD.jpg" yield strongest response (the highest score *** in terms of the measure is provided by Ref. 18) indicating these JPEG stego-codes must contain some hidden data. (The same phenomenon exists when the role of the image Lena is replaced by Jet or other images.) However, when the JPEG stego-codes created by ours are tested by Stegdetect tool, all of our JPEG stego-codes can pass the Stegdetect test (all the responses are "negative", rather than * or ** or ***), no matter the image is Lena, Jet, Boat, Peppers, Baboon, or Zelda. In other words, the

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

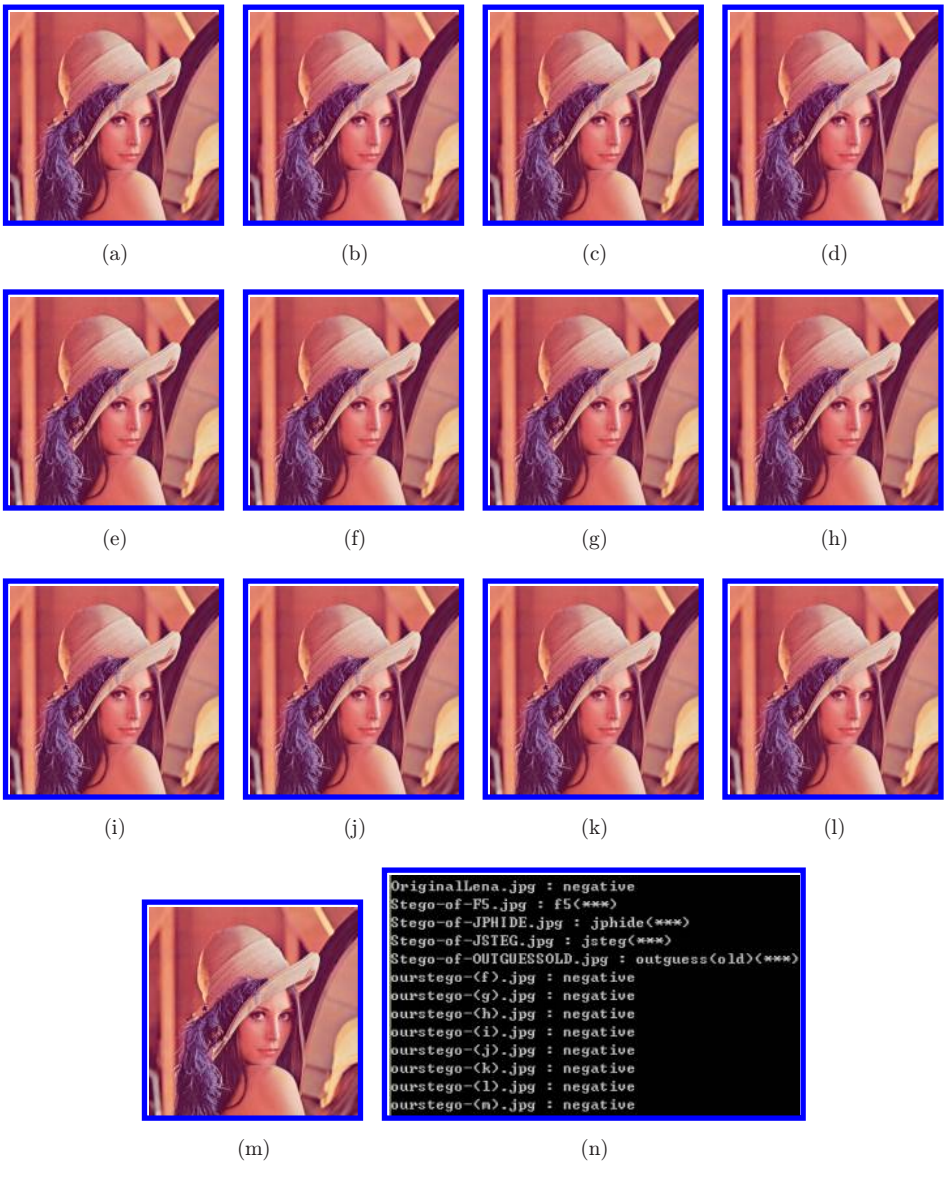


Fig. 7. Results of Stegdetect analysis: (a) the image decompressed from the JPEG-Q85 code without any hidden secret; (b)−(e) the stego-images decompressed from the JPEG stego-codes for F5,[25] JPHide,[13] Jsteg,[22] OutGuess,[17] respectively. (f)−(m) the stego-images decompressed from our eight JPEG stego-codes, respectively; (n) the results of Stegdetect analysis of the thirteen JPEG codes, the decompressed images of which are in (a)−(m).

unauthorized user will let our JPEG stego-codes pass if he uses Stegdetect tool to detect our codes.

Finally, we discuss the size of our product. The size of the JPEG code (without hiding any secret) for the gray-level image Lena with the quality factors from 10 to 95

is between 8,119 and 94,581 bytes. The size of our JPEG stego-codes listed in Table 1 are 55,432; 62,236; 70,237; 66,307; 78,484; 90,114; 67,257; and 82,207 bytes when the two parameters (*level*, *num*) are (1, 16); (1, 24); (1, 32); (2, 16); (2, 24); (2, 32); (3, 16); and (3, 24); respectively. Therefore, people will not feel strange about the sizes of our JPEG stego-codes because they fall in the reasonable range from 8,119 to 94,581 bytes. Similar observation exists when the image Lena is replaced by Jet or others.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

5. Summary

This paper proposes a JPEG-based hiding method with high capacity, high hiding-ratio, and reversibility to recover JPEG-compression code. The JPEG compression code of an image is used as the input cover media. Equation (11) creates an 8×8 hiding-capacity table HC according to the original 8×8 JPEG quantization table Q. Then Eq. (14) uses the table HC to create an 8×8 modified quantization table $\tilde{Q}$. The obtained tables HC and $\tilde{Q}$ are used universally for all blocks of the image. After that, according to the 8×8 tables Q and $\tilde{Q}$, the secret data are embedded in JPEG quantized-blocks $\{F\}$ to get our stego quantized-blocks $\{\tilde{F}\}$, which are then encoded using JPEG entropy coding. The hiding capacity and hiding ratio are adjustable by changing values of two integer parameters *level* and *num*. The proposed method can also resist the Chi-square and Stegdetect attacks so that the unauthorized users will not notice the existence of the hidden secret data.

In decoding, after lossless extraction of the secret data, the original JPEG compression code can also be reconstructed. Experimental results show that our method provides high hiding-capacity and high hiding-ratio, while maintaining good quality of stego-images. As shown in Tables 1−2 and Figs. 4−5, our method can compete with other JPEG-based hiding methods.[3,4,21,22] The proposed method is useful for images stored and transmitted in JPEG format. The reversibility allows the recovered JPEG code to be used again and again, without any degrading even after infinitely many times of rehiding.

Acknowledgments

This work is supported by National Science Council, Taiwan, R.O.C., under Grant NSC 96-2221-E-009-039. The authors also thank the three reviewers for valuable suggestions.

References

1. J. M. Barton, Method and apparatus for embedding authentication information within digital data, U.S. Patent 5646997 (1997).
2. J. J. Chae and B. S. Manjunath, Data hiding in video, *Proc. 6th IEEE Int. Conf. Imag. Process.* **1**(1) (1999) 311−315.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

3. C. C. Chang, T. S. Chen and L. Z. Chung, A steganographic method based upon JPEG and quantization table modification, *Inform. Sci.* **141**(1) (2002) 123−138.
4. C. C. Chang, C. C. Lin, C. S. Tseng and W. L. Tai, Reversible hiding in DCT-based compressed images, *Inform. Sci.* **177**(14) (2007) 2768−2786.
5. C. C. Chang, M. H. Lin and Y. C. Hu, A fast and secure image hiding scheme based on LSB substitution, *Int. J. Patt. Recogn. Artif. Intell.* **16**(4) (2001) 399−416.
6. J. Fridrich, M. Goljan and R. Du, Invertible authentication watermark for JPEG images, in *Proc. ITCC*, Las Vegas, Nevada (2001), pp. 223−227.
7. F. Goel, M. Garuba, C. Liu and T. Nguyen, The security threat posed by steganographic content on the Internet, *Proc. 4th IEEE Int. Conf. Information Technology* (2007), pp. 794−798.
8. Guillermito, Chi-square Steganography Test Program, http://www.guillermito2.net/stegano/tools/index.html.
9. Y. C. Hu and M. H. Lin, Secure image hiding scheme based upon vector quantization, *Int. J. Patt. Recogn. Artif. Intell.* **18**(6) (2004) 1111−1130.
10. M. Iwata, K. Miyake and A. Shiozaki, Digital steganography utilizing features of JPEG images, *IEICE Trans. Fund.* **E87-A**(4) (2004) 929−936.
11. G. C. Kessler, An overview of steganography for the computer forensics examiner, *Forensic Sci. Commun.* **6**(13) (2004) 45−48.
12. H. Kobayashi, Y. Noguchi and H. Kiya, A method of embedding binary data into JPEG bitstreams, *IEICE Trans. Inform. Syst.* **2**(6) (2000) 1469−1476.
13. A. Latham and J. P. Hide, http://linux01.gwdg.de/∼alatham/stego.html.
14. H. Malik, R. Chandramouli and K. P. Subbalakshmi, Steganalysis: Trends and Challenges, in *Multimedia Forensics and Security*, ed. C. T. Li (Idea Group Publishing, 2008), p. 257.
15. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, (Van Nostrand Reinhold, New York, 1993).
16. OpenJPEG, http://www.openjpeg.org.
17. N. Provos, OutGuess and Stegdetect Steganography Test Program, http://www.outguess.org/download.php.
18. N. Provos and P. Honeyman, Detecting steganographic content on the Internet, Center for Information Technology Integration Technical Report (2001) 1−11.
19. N. Provos and P. Honeyman, Hide and seek: An introduction to steganography, *IEEE Security and Privacy* **1**(3) (2003) 32−44.
20. C. C. Thien and J. C. Lin, A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function, *Patt. Recogn.* **36**(13) (2003) 2875−2881.
21. H. W. Tseng and C. C. Chang, High capacity data hiding in JPEG-compressed images, *Informatica* **15**(1) (2004) 127−142.
22. D. Upham, Jsteg, http://packetstormsecurity.org/crypt/stego/DOS/jsteg.zip.
23. R. Z. Wang and Y. D. Tsai, An image-hiding method with high hiding capacity based on best-block matching and k-means clustering, *Patt. Recogn.* **40**(2) (2007) 398−409.
24. S. J. Wang, Steganography of capacity required using modulo operator for embedding secret image, *Appl. Math. Comput.* **164** (2005), pp. 99−116.
25. A. Westfeld, F5, http://wwwrn.inf.tu-dresden.de/%7Ewestfeld/f5.html.
26. D. C. Wu and W. H. Tsai, A steganographic method for images by pixel-value differencing, *Patt. Recogn. Lett.* **24** (2003) 1613−1626.
27. H. C. Wu, N. I. Wu, C. S. Tsai and M. S. Hwang, Image steganographic scheme based on pixel-value differencing and LSB replacement methods, *IEE Proc. Vis. Imag. Sign. Process.* **152**(5) (2005) 611−615.

Lee Shu-Teng Chen received his B.S. in computer and information science from National Chiao Tung University (NCTU), Taiwan, in 1999, and M.S. in computer science and information engineering from National Taiwan University, Taiwan, in 2001.

He is a Ph.D. candidate in the Department of Computer Science and Information Engineering at NCTU. His current research interests include data hiding and image sharing.

Sian-Jheng Lin received his B.S. and M.S. in computer and information science from NCTU, Taiwan, in 2004 and 2006, respectively. He is a Ph.D. candidate in the Department of Computer Science and Information Engineering at NCTU.

His current research interests include image sharing and image processing.

Ja-Chen Lin received his B.S. and M.S. from NCTU, Taiwan. In 1988 he received his Ph.D. in mathematics from Purdue University, West Lafayette, Indiana. He joined the Department of Computer Science and Information Engineering at NCTU, where he became a professor.

His research interests include pattern recognition and image processing. He is a member of the Phi-Tau-Phi Scholastic Honor Society.

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.

This article has been cited by:

1. Che-Wei Lee, Wen-Hsiang Tsai. 2013. A data hiding method based on information sharing via PNG images for applications of color image authentication and metadata embedding. *Signal Processing* . [CrossRef]

Int. J. Patt. Recogn. Artif. Intell. 2010.24:433-456. Downloaded from www.worldscientific.com
by NATIONAL CHIAO TUNG UNIVERSITY on 04/24/14. For personal use only.