A Lossless Data Hiding Scheme for VQ Indexes Based on Joint Neighboring Coding

Andrew Rudder and The Duc Kieu*
Department of Computing and Information Technology,
Faculty of Science and Technology,
The University of the West Indies, St. Augustine, Trinidad and Tobago
[e-mail: andrew.rudder@gmail.com; ktduc0323@yahoo.com.au]
*Corresponding author: The Duc Kieu

Received January 11, 2015; revised May 8, 2015; accepted June 23, 2015; published August 31, 2015

Abstract

Designing a new reversible data hiding technique with a high embedding rate and a low compression rate for vector quantization (VQ) compressed images is encouraged. This paper proposes a novel lossless data hiding scheme for VQ-compressed images based on the joint neighboring coding technique. The proposed method uses the difference values between a current VQ index and its left and upper neighboring VQ indexes to embed n secret bits into one VQ index, where n = 1, 2, 3, or 4. The experimental results show that the proposed scheme achieves the embedding rates of 1, 2, 3, and 4 bits per index (bpi) with the corresponding average compression rates of 0.420, 0.483, 0.545, and 0.608 bit per pixel (bpp) for a 256 sized codebook. These results confirm that our scheme performs better than other selected reversible data hiding schemes.

Keywords: Data hiding, reversible data embedding, watermarking, steganography, vector quantization

http://dx.doi.org/10.3837/tiis.2015.08.013 **ISSN : 1976-7277**

1. Introduction

The ability to transmit information securely has always been of great importance. As new ways and formats are developed for sending, storing, and representing information, novel methods must also be developed to ensure the security of sensitive information. In our highly connected world and with the prevalent use of public network channels for sending sensitive information, developing new methods to protect secret information from unauthorized entities is of paramount importance to both individuals and organizations.
Cryptography is one of the methods used for securing sensitive information [1-3]. It has proven to be a good method for securing information, however it has its own weakness. That is, the encrypted information is fully exposed during transmission. As a result, attackers can procure the encrypted data and decrypt it once the encryption method is known. In the case where the encryption method is not known, an attacker can still store the encrypted data and attempt to decrypt it at his leisure.

The information hiding (also called data hiding or data embedding) field seeks to protect sensitive information by hiding the fact that a message even exists [4]. This is done by hiding the sensitive data or secret message within another medium called a cover object that does not raise suspicion when transmitted [5]. Two areas of study have emerged within the information hiding field, namely digital watermarking and steganography [6]. In watermarking, the information embedded into a cover object (e.g., image, audio, video, or text) is pertinent to the cover object and is used to verify its validity. In steganography, however, the main purpose of the cover object is to hide the fact that covert communication is occurring.

All techniques developed within the steganographic and watermarking fields can be categorized into reversible (also called lossless, invertible, or distortion-free) and irreversible (also called lossy) techniques. Reversible techniques [7-9] are those that guarantee the perfect reconstruction of a cover object after a secret message has been extracted. Irreversible techniques cannot guarantee this [10,11]. Whether a reversible or irreversible technique is used in a particular situation depends on the application. If the cover object must be fully recovered such as for military or healthcare purposes, a reversible technique must be chosen. However, for situations where the full recovery of the cover object is not required, an irreversible technique may be applied.

Information hiding techniques can be applied to three domains, namely the spatial [12-14], frequency (or transformed) [15,16], and compressed [17-22] domains. In the spatial domain, information is hidden by modifying the amplitudes of the pixel values whereas in the frequency domain it is achieved by modifying the transformed coefficients. Many image compression algorithms have been used for storing and transmitting images across networks. Methods within the compressed domain such as vector quantization (VQ) [23] seek to embed secret information during the encoding of the VQ index table. Each domain has its own strengths and weaknesses with regard to embedding (or hiding) capacity, storage space, processing time, and other features. This article proposes a lossless data hiding system in the VQ-compressed domain that is suitable to low bandwidth communication channels.

In 2009, Chang et al. [24] proposed a reversible information hiding scheme for VQ indices using joint neighboring coding (JNC). Their method hides one secret bit into a VQ index in raster scan order. The average embedding rate of this method is 0.984 bpi with a compression rate of 0.510 bpp for a 256 sized codebook. The main weakness of their method is that it

embeds less than 1 bpi and their compression rate is greater than 0.5 bpp. Later in 2009, Wang and Lu [25] proposed a path optional lossless data hiding method also based on JNC to improve the embedding capacity of Chang et al.'s scheme [24]. Wang and Lu's method uses two paths to allow a user to select the path 1 or 2 to conceal two or three secret bits into one VQ index, respectively. For a codebook of size 256, this scheme is able to achieve the average embedding rate of 1.953 bpi with the compression rate of 0.573 bpp for the first path and the average embedding rate of 2.884 bpi with the compression rate of 0.641 bpp for the second path. While there is an improvement in the embedding rate, the scheme however has a high compression rate. In 2013, Lee et al. [26] presented a lossless data hiding scheme that achieves the embedding rate of 2.952 bpi with the compression rate of 0.546 bpp for the 256 sized codebook. This scheme improves the embedding capacity and compression rate of the previous schemes [24, 25]. Their scheme, however, degrades quickly as the codebook size is increased. The two schemes [24, 25] were also improved by Kieu and Ramroach [27]. In this scheme, the first column and the first row of the VQ index table are used as the seed area. That is, there is no secret data embedding in this region. Consequently, the embedding rate and compression rate of this method can be further improved by eliminating the seed area.

To surmount the weaknesses of Chang et al.'s [24], Wang and Lu's [25], and Lee et al.'s [26] methods, we propose a novel lossless data hiding scheme for VQ-compressed images based on JNC. The proposed method conceals n secret bits, where n = 1, 2, 3, or 4, into each VQ index. The scheme achieves the various embedding rates of 1, 2, 3, and 4 bpi with compression rates of 0.420, 0.483, 0.545, and 0.608 bpp, respectively, based on the codebook sized 256. These results demonstrate that the proposed approach is better than previous methods [24-26].

The rest of this paper unfolds in the following order. An overview of vector quantization, the methods proposed by Chang et al. [24], Wang and Lu [25], and Lee et al. [26] are covered in Section 2. The proposed method is then presented in Section 3. The experimental results are presented in Section 4, followed by a conclusion in Section 5.

2. Related Works

In this section, we present a review of the vector quantization (VQ) and Lee et al.'s method [26].

2.1 Vector Quantization

Vector quantization (VQ) is a lossy data compression method commonly used in image compression that is based on the block coding principle [23]. There are three steps associated with this method: codebook generation, VQ encoding, and VQ decoding. In the first step, the codebook CB containing N k-dimensional codewords CW_i's is generated by using the LBG clustering algorithm [28], where i = 0, 1, …, N - 1 and CW_i = (cw_{i1}, cw_{i2}, …, cw_{ik}). The greyscale image I sized H×W to be compressed is then partitioned into non-overlapping image block B's sized hs×ws, where hs×ws = k (e.g., H = W = 512, N = 256, hs = ws = 4, k = 16). In the second step, the VQ encoder takes each image block B in raster scan order and compresses it by examining the codebook CB and selecting the codeword CW_i with the minimum Euclidean distance from B. The index value of the selected codeword CW_i is then inserted into the VQ index table T sized (H/hs)×(W/ws) in raster scan order. The final step is the reconstruction of the original image I from the received VQ index table T and the codebook

CB. Each block *B* of the original image is reconstructed by taking the respective VQ index in *T* and looking up the codeword CW_i associated with it.

2.2 Lee et al.'s Method

Lee et al. [26] proposed a lossless data hiding method for VQ indices based on neighboring correlation. In general, the scheme embeds *r* or *v* secret bits from a secret message *S* into one VQ index of a VQ index table *T* sized (*H*/*hs*)×(*W*/*ws*) of a VQ-compressed image *I* sized *H*×*W*, where *hs*×*ws* represents the image block size used during the VQ encoding process. In this scheme, the codewords in the codebook are sorted according to the mean values of the codewords before the VQ encoding is carried out. The VQ index table *T* is composed of (*H*/*hs*)×(*W*/*ws*) VQ indices $T_{i,j}$'s, where $i = 0, 1, ..., (H/hs) - 1$ and $j = 0, 1, ..., (W/ws) - 1$. Lee et al.'s method uses a predefined parameter *z* that is half the length of the coarse sub-codebook. The parameter *r* is the number of secret bits to be hidden in the case of a fine hit and *v* is the number of secret bits to be concealed in the case of a coarse hit. The values of *r* and *v* are calculated by $r = \lceil \log_2 N \rceil - 2$, $v = \lceil \log_2 N \rceil - 3 - \lceil \log_2 (2z) \rceil$, where *N* is the codebook size. Let *y* be the current VQ index to encode (i.e., $y = T_{i,j}$), *yl* be the VQ index to the left (i.e., $yl = T_{i,j-1}$) and *yt* be the VQ index above (i.e., $yt = T_{i-1,j}$). Additionally, let *fsb*0 and *fsb*1 be the fine sub-codebooks associated with *yl* and *yt*, respectively, and let *csb*0 and *csb*1 be the coarse sub-codebooks associated with *yl* and *yt*, respectively. These sub-codebooks are defined by $fsb0 = \{yl\}$, $fsb1 = \{yt\}$, $csb0 = \{yl - z, ..., yl - 1, yl + 1, ..., yl + z\}$, and $csb1 = \{yt - z, ..., yt - 1, yt + 1, ..., yt + z\}$, respectively. If *y* belongs to *fsb*0 or *fsb*1, a fine hit state is obtained. If *y* belongs to *csb*0 or *csb*1, a coarse hit state is achieved, otherwise a miss state is obtained.

There are three main cases in their scheme. The first case occurs if *y* belongs to either *fsb*0 (i.e., $y = yl$) or *fsb*0 (i.e., $y = yt$) and this constitutes a fine hit. In this case, a 2-bit indicator code 00 or 01 is appended to the output code stream *CS*, followed by the next *r* secret bits $s_1 s_2 ... s_r$ from the secret stream *S* (i.e., $CS = CS||00||s_1 s_2 ... s_r$ or $CS = CS||01||s_1 s_2 ... s_r$). The second case is where *y* either exists in *csb*0 or *csb1* and this constitutes a coarse hit. In this case, the $\log_2(2z)$-bit binary representation of the *y*'s position, denoted as *csbpos*, in the corresponding coarse sub-codebook is recorded. If *y* exists in *csb*0, the bit stream $100||csbpos||s_1 s_2 ... s_v$ is appended to the output code stream *CS*. If *y* exists in *csb*1, the bit stream $101||csbpos||s_1 s_2 ... s_v$ is added to the output code stream *CS*. In the final case where *y* is not equal to *yl* or *yt* and does not belong to *csb*0 or *csb*1 (i.e., miss state), the bit stream $11||y_2$ is appended to the output code stream *CS*, where y_2 is the $\lceil \log_2 N \rceil$-bit binary representation of *y*.

The above embedding procedure is continued until all remaining VQ indices of *T* are processed. The other details of this method can be found in [26]. Lee et al.'s approach performs very well for smooth images (e.g., Tiffany image) but it has a poor performance for complex images (e.g., Baboon image). In addition, the performance of this scheme degrades quickly when the codebook size *N* is greater than 128 (e.g., $N = 256$, 512, and 1024).

3. The Proposed Method

3.1 The Encoding and Embedding Phase

The large compression rates of Chang et al.'s [24] and Wang Lu et al.'s [25] schemes may arouse attackers' suspicions. In this section, we present the proposed lossless data hiding scheme for VQ indexes based on joint neighboring coding [24]. The proposed scheme can embed *n* secret bits per VQ index in raster scan order, where $n = 1, 2, 3$ or 4. We limit the value of *n* to be less than or equal to 4 because for values of *n* greater than 4, the smallest average

compression rate attainable is greater than 0.608 bpp for the codebook size N = 256. To increase the security of secret data distribution, it is supposed that the secret data has been encrypted by using a well-known cryptosystem such as AES [2] before secret data hiding takes place. Therefore, even though adversaries can somehow extract the encrypted secret data from the output code stream, they still cannot comprehend the real information without also having a secure decryption key.

Firstly, a greyscale cover image I sized $H \times W$ is compressed by a VQ encoder that uses an N-sized codebook CB containing N k-dimensional codewords (e.g., $H = W = 512$, $N = 256$, and $k = 16$). The proposed method uses the codebook that is sorted as mentioned in Chang et al.'s method [24]. The result of which is a VQ index table T sized $(H/hs) \times (W/ws)$, where $hs \times ws$ is the image block size used by the VQ encoder (e.g., $hs = ws = 4$). The VQ index table $T = \{T_{i,j}\}$, where $0 \le i \le H/hs - 1$, $0 \le j \le W/ws - 1$, and $0 \le T_{i,j} \le N - 1$, is scanned in raster scan order for encoding and embedding. Secret data hiding occurs during the encoding of each VQ index. This process conceals n secret bits from the secret message S into each VQ index in T. The first VQ index y located at the first row and column of T (i.e., $y = T_{0,0}$) is converted to its $\lceil \log_2 N \rceil$-bit binary representation y_2. The first n secret bits $s_1 s_2 ... s_n$ are then read from S and y is encoded by $y_2 || s_1 s_2 ... s_n$, where $||$ denotes the concatenation operation. The encoded bit stream $y_2 || s_1 s_2 ... s_n$ is then appended to the output code stream CS (i.e., $CS = CS || y_2 || s_1 s_2 ... s_n$).

For all remaining VQ indices, the encoding and embedding process is performed as follows. The next VQ index y is read from T (i.e., $y = T_{i,j} \ne T_{0,0}$). In general, the encoding is performed on each VQ index by using two difference values dl and du, where dl is the difference between the VQ index to the left of y and y (i.e., $dl = l - y$, where $y = T_{i,j}$, $l = T_{i,j-1}$), and du is the difference between the VQ index directly above y and y (i.e., $du = u - y$, where $y = T_{i,j}$, $u = T_{i-1,j}$,). The VQ indexes in the first row of T (i.e., $y = T_{0,j}$ where $1 \le j \le W/ws - 1$), however, have no upper neighboring VQ indexes to calculate du and so dl is used in this case. That is, set $l = T_{0,j-1}$ and $u = l$, the difference value du is then computed by $du = u - y$. Similarly, the VQ indexes in the first column of T (i.e., $y = T_{i,0}$ where $1 \le i \le H/hs - 1$) have no left neighboring VQ indexes to compute dl and so du is used in this case. That is, set $u = T_{i-1,0}$ and $l = u$, the difference value dl is then computed by $dl = l - y$. The calculations of the difference values dl and du are shown in **Fig. 1**.

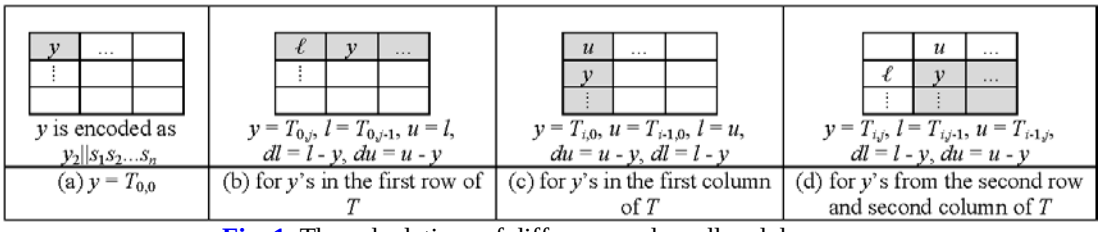

Fig. 1. The calculations of difference values dl and du

The next n secret bits $s_1 s_2 ... s_n$ are then read from S and the values of dl and du are examined. If either dl or du is equal to 0 (i.e., case 1: the VQ index to the left or above has the same value as the current VQ index y), y is encoded by $s_1 s_2 ... s_n || 000$ or $s_1 s_2 ... s_n || 001$, respectively. The encoded bit stream $s_1 s_2 ... s_n || 000$ or $s_1 s_2 ... s_n || 001$ is then appended to the code stream CS (i.e., $CS = CS || s_1 s_2 ... s_n || 000$ or $CS = CS || s_1 s_2 ... s_n || 001$).

If du and dl are not equal to 0, only the value of dl is used to encode y. The absolute value of dl, denoted as $|dl|$, ranges from 0 to $N - 1$ inclusive. To represent dl requires m bits, where $1 \le m$

$\leq \lceil \log_2 N \rceil$, the notation $\lceil \rceil$ denotes the ceiling function, and N is the codebook size (e.g., N = 256). As with previous schemes, a particular value of m is chosen as a threshold value.

In the case where $|dl|$ cannot be represented in m bits (i.e., case 2: $|dl| > 2^m - 1$), y is encoded by $s_1 s_2 ... s_n || 01 || y_2$, where y_2 is the $\lceil \log_2 N \rceil$-bit binary representation of y. The encoded bit stream $s_1 s_2 ... s_n || 01 || y_2$ is then appended to the code stream CS (i.e., $CS = CS || s_1 s_2 ... s_n || 01 || y_2$).

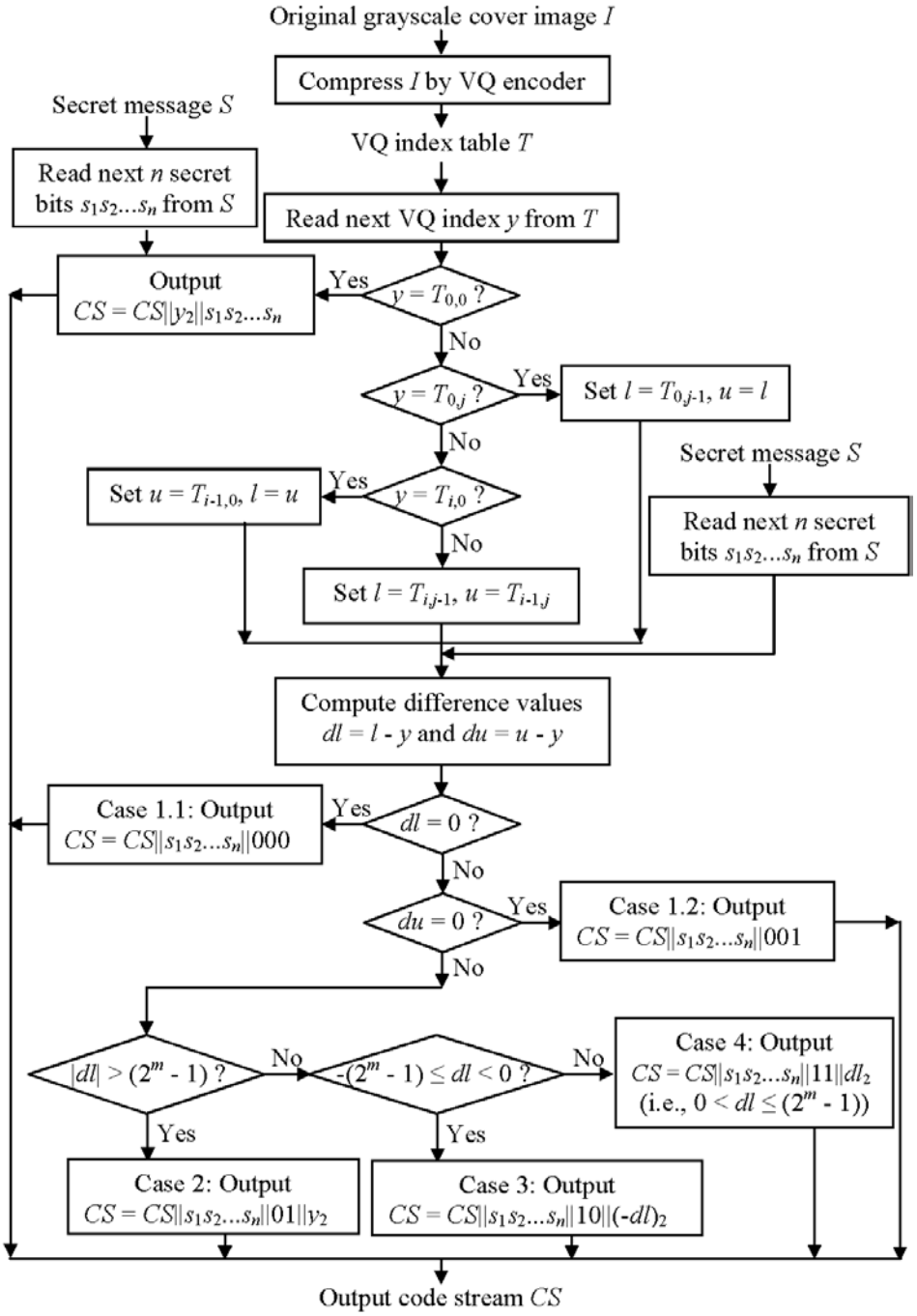

Fig. 2. The flow chart of the proposed encoding and embedding scheme.

If dl is greater than or equal to $-(2^m - 1)$ and less than 0 (i.e., case 3: $-(2^m - 1) \le dl < 0$), y is encoded by $s_1s_2...s_n||10||(-dl)_2$, where $(-dl)_2$ is the m-bit binary representation of the absolute value of dl. The encoded bit stream $s_1s_2...s_n||10||(-dl)_2$ is then added to the code stream CS (i.e., $CS = CS||s_1s_2...s_n||10||(-dl)_2$).

If dl is greater than 0 and less than or equal to $2^m - 1$ (i.e., case 4: $0 < dl \le 2^m - 1$), y is encoded by $s_1s_2...s_n||11||dl_2$, where dl_2 is the m-bit binary representation of dl. The encoded bit stream $s_1s_2...s_n||11||dl_2$ is then sent to the code stream CS (i.e., $CS = CS||s_1s_2...s_n||11||dl_2$).

The above encoding and embedding process is repeated for the next VQ index in raster scan order until all VQ indexes in the VQ index table T are processed. The proposed encoding and embedding method is summarized in **Table 1**. The flowchart of the proposed encoding and embedding scheme is shown in **Fig. 2**. The proposed encoding and embedding algorithm is described next.

Table 1. Summary of the proposed encoding and embedding scheme

Secret bits	Case 1		Case 2	Case 3	Case 4
	Case 1.1: $dl = 0$	Case 1.2: $du = 0$	$\|dl\| > 2^m - 1$	$-(2^m - 1) \le dl < 0$	$0 < dl \le 2^m - 1$
$s_1s_2...s_n$	$s_1s_2...s_n\|\|000$	$s_1s_2...s_n\|\|001$	$s_1s_2...s_n\|\|01\|\|y_2$	$s_1s_2...s_n\|\|10\|\|(-dl)_2$	$s_1s_2...s_n\|\|11\|\|dl_2$

The encoding and embedding algorithm

Input: A grayscale cover image I sized $H \times W$, a codebook CB sized N, a secret message S, the preset values of m and n, where $1 \le m \le \lceil \log_2 N \rceil$ and n = 1, 2, 3, or 4

Output: The binary code stream CS

Step 1: Compress I by using a VQ encoder to obtain the VQ index table T sized $(H/hs) \times (W/ws)$, where $hs \times ws$ is the size of an image block used by the VQ encoder.

Step 2: Read the next VQ index y from the VQ index table T in raster scan order.

Step 3: If y is the top-left element of T (i.e., $y = T_{0,0}$), then

Step 3.1: Read the next n secret bits $s_1s_2...s_n$ from the secret message S.

Step 3.2: Append $y_2||s_1s_2...s_n$ to CS (i.e., $CS = CS||y_2||s_1s_2...s_n$), where y_2 is the $\lceil \log_2 N \rceil$-bit binary representation of y and the notation || denotes the concatenation operation.

Step 4: If y is not the top-left element of T (i.e., $y \ne T_{0,0}$), then

Step 4.1: If y is located in the first row of T except $T_{0,0}$ (i.e., $y = T_{0,j}$ where $1 \le j < W/ws$), then

Set l to be the left neighboring VQ index of y (i.e., $l = T_{0,j-1}$) and $u = l$.

Step 4.2: If y is positioned in the first column of T except $T_{0,0}$ (i.e., $y = T_{i,0}$ where $1 \le i < H/hs$), then

Set u to be the upper neighboring VQ index of y (i.e., $u = T_{i-1,0}$) and $l = u$.

Step 4.3: If y is located from the second row and second column of T (i.e., $y = T_{i,j}$ where $1 \le i < H/hs$ and $1 \le j < W/ws$), then

Set l to be the left neighboring VQ index of y (i.e., $l = T_{i,j-1}$).

Set u to be the upper neighboring VQ index of y (i.e., $u = T_{i-1,j}$).

Step 5: Read the next n secret bits $s_1s_2...s_n$ from the secret message S.

Step 6: Compute the difference values $dl = l - y$ and $du = u - y$.

Step 7: If $dl = 0$ or $du = 0$ (i.e., case 1), then

Step 7.1: If dl equals 0 (i.e., case 1.1), then encode y by $s_1s_2...s_n||000$.

Append the bit stream $s_1s_2...s_n||000$ to CS (i.e., $CS = CS||s_1s_2...s_n||000$).

Step 7.2: If du is equal to 0 (i.e., case 1.2), then encode y by $s_1s_2...s_n||001$.

Append the bit stream $s_1s_2...s_n||001$ to CS (i.e., $CS = CS||s_1s_2...s_n||001$).

Step 8: Else if $|dl| > 2^m - 1$ (i.e., case 2), then encode y by $s_1s_2...s_n||01||y_2$.

Append the bit stream $s_1s_2...s_n||01||y_2$ to CS (i.e., $CS = CS||s_1s_2...s_n||01||y_2$).
Step 9: Else if $-(2^m - 1) \le dl < 0$ (i.e., case 3), then encode y by $s_1s_2...s_n||10||(-dl)_2$,
where $(-dl)_2$ is the m-bit binary representation of the absolute value of dl.
Append the bit stream $s_1s_2...s_n||10||(-dl)_2$ to CS (i.e., $CS = CS||s_1s_2...s_n||10||(-dl)_2$).
Step 10: Else (i.e., case 4: $0 < dl \le 2^m - 1$) encode y by $s_1s_2...s_n||11||dl_2$,
where dl_2 is the m-bit binary representation of dl.
Append the bit stream $s_1s_2...s_n||11||dl_2$ to CS (i.e., $CS = CS||s_1s_2...s_n||11||dl_2$).
Step 11: Repeat steps 2 to 10 until all VQ indexes of the VQ index table T are processed.
Step 12: Output the binary code stream CS.

3.2 The Decoding and Extracting Phase

The decoding and extracting process is the inverse process of the encoding and embedding process. At the receiving side, with the received code stream CS and N-sized codebook CB, the decoder can extract the embedded secret message and restore the original VQ indices. The flowchart of the proposed decoding and extracting scheme is shown in **Fig. 3**. The summary of the proposed decoding and extracting algorithm is given below.

The decoding and extracting algorithm
Input: The binary code stream CS, the codebook CB sized N,
the preset values of m and n, where $1 \le m \le \lceil \log_2 N \rceil$ and n = 1, 2, 3, or 4
Output: The extracted secret message S and the reconstructed cover image I' sized $H \times W$
Step 1: Let the extracted secret message S and the recovered VQ index table T be empty.
Step 2: If the currently decoded VQ index y is the top-left element of T (i.e., y is at the position $T_{0,0}$), then
Step 2.1: Read the next $\lceil \log_2 N \rceil$ bits from CS and convert them into the decimal value de.
Step 2.2: Recover the original VQ index by $y = de$.
Step 2.3: Read the next n bits $c_1c_2...c_n$ from the code stream CS.
Step 2.4: Extract the n secret bits by $s_1s_2...s_n = c_1c_2...c_n$.
Step 2.5: Update the extracted secret message by $S = S||s_1s_2...s_n$.
Step 3: If the currently decoded VQ index y is not the top-left element of T (i.e., y is not at the location $T_{0,0}$), then
Step 3.1: If the currently decoded VQ index y is located in the first row of T except the position $T_{0,0}$ (i.e., y is at the position $T_{0,j}$ where $1 \le j < W/ws$), then
Set l to be the left neighboring VQ index of y (i.e., $l = T_{i,j-1}$) and $u = l$.
Step 3.2: If the currently decoded VQ index y is positioned in the first column of T except the position $T_{0,0}$ (i.e., y is at the position $T_{i,0}$ where $1 \le i < H/hs$), then
Set u to be the upper neighboring VQ index of y (i.e., $u = T_{i-1,j}$) and $l = u$.
Step 3.3: If the currently decoded VQ index y is located from the second row and second column of T (i.e., y is located at $T_{i,j}$ where $1 \le i < H/hs$ and $1 \le j < W/ws$), then
Set l to be the left neighboring VQ index of y (i.e., $l = T_{i,j-1}$).
Set u to be the upper neighboring VQ index of y (i.e., $u = T_{i-1,j}$).
Step 4: Read the next n bits $c_1c_2...c_n$ from the code stream CS.
Step 5: Extract the n secret bits by $s_1s_2...s_n = c_1c_2...c_n$.
Update the extracted secret message by $S = S||s_1s_2...s_n$.
Step 6: Read the next two bits c_1c_2 from the code stream CS.
Step 7: If $c_1c_2 = 00$ (i.e., case 1: $dl = 0$ or $du = 0$), then
Step 7.1: Read the next bit c_3 from the code stream CS.
Step 7.2: If $c_3 = 0$ (i.e., case 1.1: $dl = 0$), then recover the original VQ index by $y = l$.

Step 7.3: Else (i.e., $c_3 = 1$, case 1.2: $du = 0$), then restore the original VQ index by $y = u$.
Step 8: Else if $c_1c_2 = 01$ (i.e., case 2: $|dl| > 2^m - 1$), then
Read the next $\lceil \log_2 N \rceil$ bits from *CS* and convert them into the decimal value *de*.
Reconstruct the original VQ index by $y = de$.
Step 9: Else (i.e., case 3: $-(2^m - 1) \le dl < 0$ or case 4: $0 < dl \le 2^m - 1$)
Read the next *m* bits from *CS* and convert them into the decimal value *dl*.
If $c_1c_2 = 10$ (i.e., case 3: $-(2^m - 1) \le dl < 0$), then set $dl = -dl$.
Recover the original VQ index by $y = l - dl$.
Step 10: Insert the restored VQ index *y* to the VQ index table *T* in raster scan order.
Step 11: Repeat steps 3 to 10 until all bits of the code stream *CS* are processed.
Step 12: Restore the cover image *I'* sized $H \times W$ from the constructed VQ index table *T* sized $(H/hs) \times (W/ws)$ by using the VQ decoder.

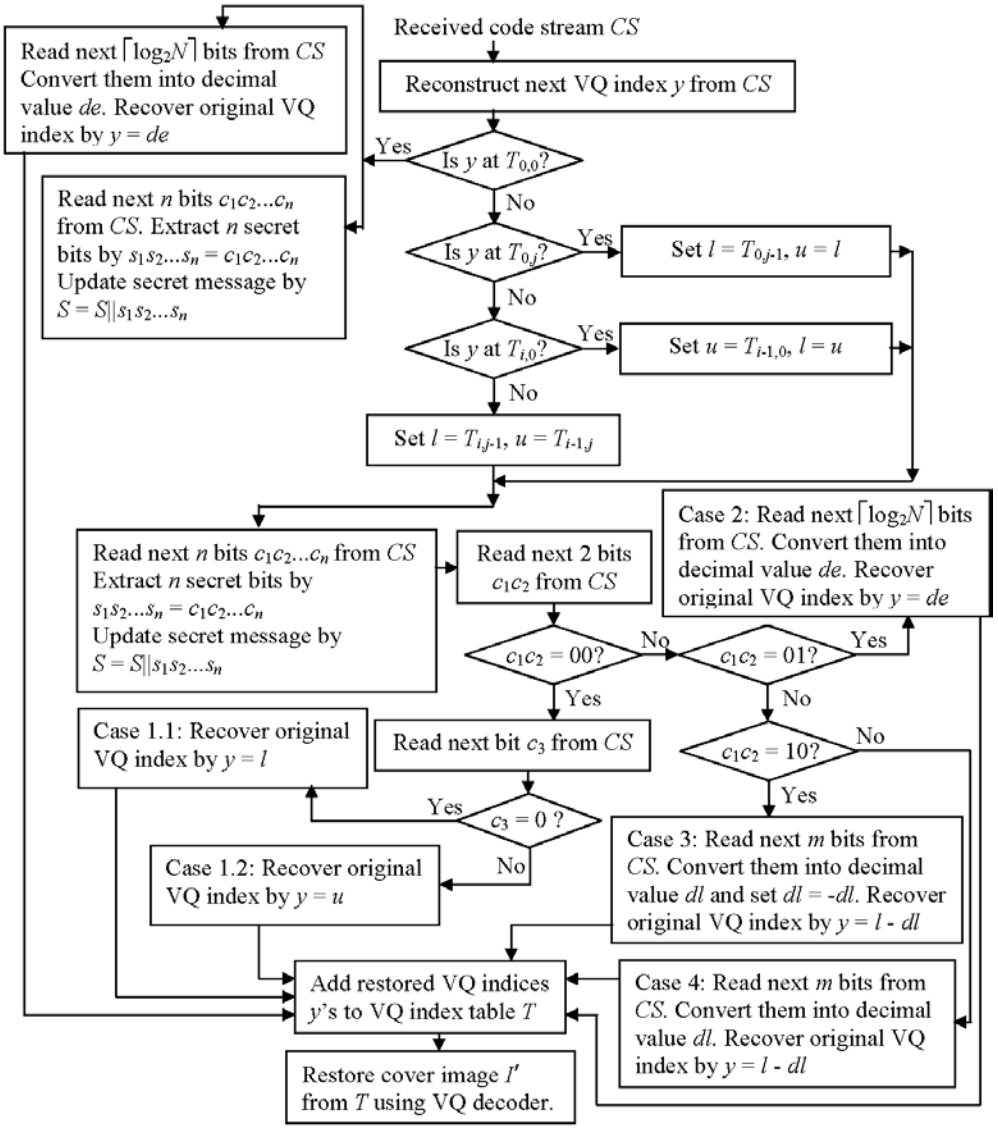

Fig. 3. The flowchart of the proposed decoding and extracting scheme.

4. Experimental Results and Discussion

The proposed scheme was implemented by using Microsoft Visual C++ 2010 software running on the Intel Core i7, 2.2 GHz CPU, and 6 GB RAM hardware platform. The binary secret message S was randomly generated by using the `rand()` function. Seven grayscale cover images all sized 512×512 were used to test the previous and proposed schemes and are shown in **Fig. 4**.

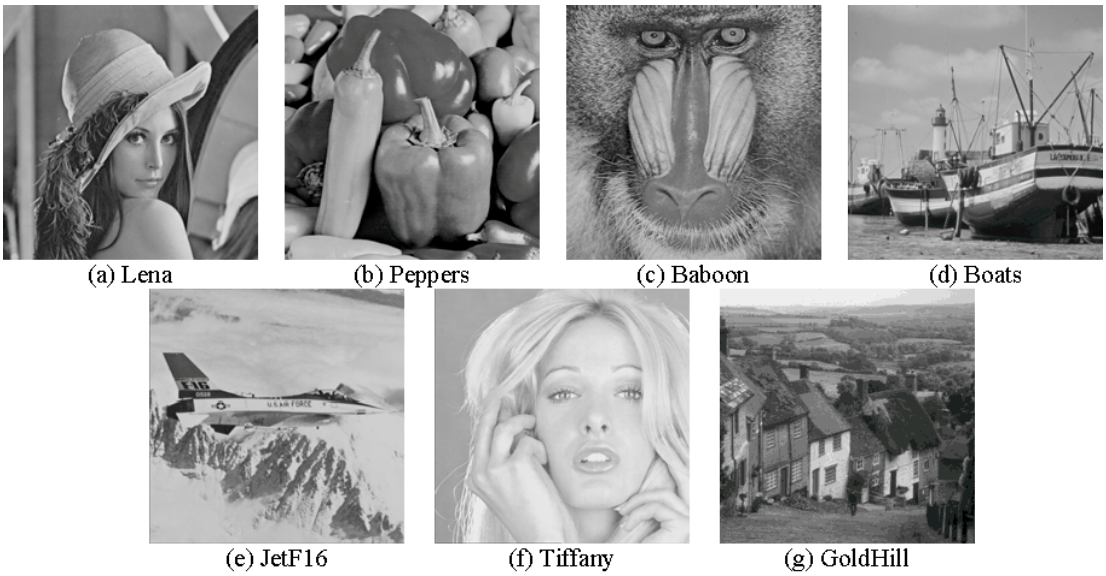

Fig. 4. Grayscale cover images used in performance tests

Sorted codebooks of sizes N = 128, 256, 512, and 1024 consisting of 16-dimensional codewords were used to generate the VQ index tables for each test image (i.e., $k = hs \times ws = 16$). The performances of the proposed method for the values of n = 1, 2, 3, and 4 (denoted as P1, P2, P3, and P4, respectively) are compared to Chang et al.'s scheme [24], Wang and Lu's scheme [25], and Lee et al.'s scheme [26].

In order to evaluate the performance of the proposed scheme, four criteria, compression rate measured in bit per pixel (bpp), embedding rate measured in bits per index (bpi), embedding efficiency, and visual quality of reconstructed images were used. The compression rate is defined by $CR = ||CS|| / H \times W$ (bpp), where $||CS||$ is the length of the output code stream CS and $H \times W$ represents the number of pixels in the original cover image. The embedding rate is defined by $ER = ||S|| / NI$ (bpi), where $||S||$ is the total number of secret bits that can be embedded into a VQ index table and NI is the number of indices in the VQ index table (i.e., $NI = (H/hs) \times (W/ws)$).

The embedding efficiency (EE) is the number of secret bits embedded when one bit of the output code stream CS is transmitted. The embedding efficiency is defined by $EE = ||S|| / ||CS|| = ER / (CR \times hs \times ws)$, where $hs \times ws$ is the size of an image block used by the VQ encoder.

4.1 Experiments on Selecting Appropriate Parameters *m* and *z*

The compression rate performance of Chang et al.'s scheme [24], Wang and Lu's scheme [25], and the proposed scheme is affected by the preset value of the parameter m that is used to represent the difference value d in the binary representation. The performances with regard to

the embedding rate and compression rate of Lee et al.'s scheme [26] depend on the predetermined value of the parameter z that is used for the coarse sub-codebooks. The impact of the preset values of the parameter m on the proposed scheme are presented in **Tables 2-5**. and that of the parameter z on Lee et al.'s method are shown in **Tables 6-9** in the appdenix.

Table 2. The effect of different values of m on compression rate of the proposed scheme with n = 1, 2, 3, and 4 for the codebook size N = 128

m	1	2	3	4	5	6	7
a) n = 1							
Lena	0.395	0.376	0.362	0.356	0.365	0.383	0.408
Peppers	0.388	0.373	0.356	0.351	0.359	0.377	0.401
Baboon	0.524	0.501	0.478	0.461	0.465	0.494	0.540
Boats	0.367	0.352	0.343	0.338	0.345	0.360	0.379
JetF16	0.358	0.343	0.337	0.334	0.340	0.352	0.370
Tiffany	0.292	0.281	0.282	0.288	0.297	0.307	0.318
GoldHill	0.429	0.410	0.387	0.378	0.390	0.416	0.448
Average *CR*	0.393	0.377	0.364	**0.358**	0.366	0.384	0.409
b) n = 2							
Lena	0.458	0.439	0.424	0.418	0.427	0.446	0.470
Peppers	0.450	0.435	0.418	0.413	0.422	0.440	0.463
Baboon	0.586	0.563	0.541	0.524	0.527	0.557	0.603
Boats	0.430	0.414	0.406	0.400	0.407	0.422	0.442
JetF16	0.421	0.405	0.399	0.397	0.403	0.415	0.433
Tiffany	0.354	0.344	0.344	0.351	0.359	0.370	0.381
GoldHill	0.492	0.473	0.449	0.440	0.453	0.479	0.511
Average *CR*	0.456	0.439	0.426	**0.420**	0.428	0.447	0.472
c) n =3							
Lena	0.520	0.501	0.487	0.481	0.490	0.508	0.533
Peppers	0.513	0.498	0.481	0.476	0.484	0.502	0.526
Baboon	0.649	0.626	0.603	0.586	0.590	0.619	0.665
Boats	0.492	0.477	0.468	0.463	0.470	0.485	0.504
JetF16	0.483	0.468	0.462	0.459	0.465	0.477	0.495
Tiffany	0.417	0.406	0.407	0.413	0.422	0.432	0.443
GoldHill	0.554	0.535	0.512	0.503	0.515	0.541	0.573
Average *CR*	0.518	0.502	0.489	**0.483**	0.491	0.509	0.534
d) n = 4							
Lena	0.583	0.564	0.549	0.543	0.552	0.571	0.595
Peppers	0.575	0.560	0.543	0.538	0.547	0.565	0.588
Baboon	0.711	0.688	0.666	0.649	0.652	0.682	0.728
Boats	0.555	0.539	0.531	0.525	0.532	0.547	0.567
JetF16	0.546	0.530	0.524	0.522	0.528	0.540	0.558
Tiffany	0.479	0.469	0.469	0.476	0.484	0.495	0.506
GoldHill	0.617	0.598	0.574	**0.565**	0.578	0.604	0.636
Average *CR*	0.581	0.564	0.551	0.545	0.553	0.572	0.597

Table 3. The effect of different values of m on compression rate of the proposed scheme with n = 1, 2, 3, and 4 for the codebook size N = 256

m	1	2	3	4	5	6	7	8
a) n = 1								
Lena	0.501	0.476	0.444	0.426	0.428	0.446	0.476	0.512
Peppers	0.488	0.463	0.434	0.414	0.416	0.436	0.464	0.498

Baboon	0.623	0.603	0.572	0.544	0.530	0.541	0.581	0.636
Boats	0.440	0.421	0.395	0.386	0.389	0.403	0.426	0.453
JetF16	0.443	0.407	0.392	0.387	0.392	0.407	0.430	0.459
Tiffany	0.388	0.354	0.333	0.333	0.346	0.366	0.388	0.411
GoldHill	0.537	0.513	0.478	0.453	0.448	0.471	0.507	0.550
Average *CR*	0.489	0.462	0.435	**0.420**	0.421	0.439	0.467	0.503
b) $n = 2$								
Lena	0.563	0.539	0.506	0.488	0.491	0.509	0.539	0.575
Peppers	0.551	0.525	0.496	0.477	0.479	0.498	0.527	0.561
Baboon	0.685	0.666	0.634	0.607	0.592	0.603	0.644	0.698
Boats	0.502	0.484	0.457	0.449	0.451	0.466	0.488	0.516
JetF16	0.506	0.469	0.454	0.450	0.454	0.470	0.493	0.521
Tiffany	0.450	0.417	0.396	0.396	0.409	0.429	0.451	0.474
GoldHill	0.599	0.575	0.541	0.515	0.511	0.533	0.570	0.612
Average *CR*	0.551	0.525	0.498	**0.483**	0.484	0.501	0.530	0.565
c) $n = 3$								
Lena	0.626	0.601	0.569	0.551	0.553	0.571	0.601	0.637
Peppers	0.613	0.588	0.559	0.539	0.541	0.561	0.589	0.623
Baboon	0.748	0.728	0.697	0.669	0.655	0.666	0.706	0.761
Boats	0.565	0.546	0.520	0.511	0.514	0.528	0.551	0.578
JetF16	0.568	0.532	0.517	0.512	0.517	0.532	0.555	0.584
Tiffany	0.513	0.479	0.458	0.458	0.471	0.491	0.513	0.536
GoldHill	0.662	0.638	0.603	0.578	0.573	0.596	0.632	0.675
Average *CR*	0.614	0.587	0.560	**0.545**	0.546	0.564	0.592	0.628
d) $n = 4$								
Lena	0.688	0.664	0.631	0.613	0.616	0.634	0.664	0.700
Peppers	0.676	0.650	0.621	0.602	0.604	0.623	0.652	0.686
Baboon	0.810	0.791	0.759	0.732	0.717	0.728	0.769	0.823
Boats	0.627	0.609	0.582	0.574	0.576	0.591	0.613	0.641
JetF16	0.631	0.594	0.579	0.575	0.579	0.595	0.618	0.646
Tiffany	0.575	0.542	0.521	0.521	0.534	0.554	0.576	0.599
GoldHill	0.724	0.700	0.666	0.640	0.636	0.658	0.695	0.737
Average *CR*	0.676	0.650	0.623	**0.608**	0.609	0.626	0.655	0.690

Table 4. The effect of different values of *m* on compression rate of the proposed scheme with $n = 1, 2, 3$, and 4 for the codebook size $N = 512$

m	1	2	3	4	5	6	7	8
a) $n = 1$								
Lena	0.601	0.582	0.544	0.512	0.497	0.506	0.531	0.569
Peppers	0.589	0.569	0.538	0.500	0.484	0.491	0.517	0.554
Baboon	0.706	0.692	0.670	0.639	0.608	0.591	0.607	0.656
Boats	0.526	0.505	0.475	0.453	0.446	0.452	0.473	0.502
JetF16	0.567	0.535	0.487	0.468	0.470	0.484	0.510	0.545
Tiffany	0.468	0.442	0.416	0.398	0.397	0.410	0.435	0.465
GoldHill	0.623	0.602	0.572	0.536	0.511	0.515	0.544	0.587
Average *CR*	0.583	0.561	0.529	0.501	**0.488**	0.493	0.517	0.554
b) $n = 2$								
Lena	0.664	0.644	0.606	0.574	0.559	0.568	0.593	0.631
Peppers	0.652	0.631	0.600	0.562	0.546	0.554	0.580	0.617
Baboon	0.768	0.754	0.732	0.701	0.670	0.654	0.670	0.718
Boats	0.588	0.567	0.537	0.515	0.508	0.515	0.535	0.565

JetF16	0.629	0.597	0.550	0.531	0.533	0.547	0.572	0.608
Tiffany	0.530	0.505	0.478	0.460	0.459	0.472	0.498	0.528
GoldHill	0.686	0.665	0.635	0.599	0.573	0.577	0.606	0.649
Average *CR*	0.645	0.623	0.591	0.563	**0.550**	0.555	0.579	0.617
c) $n = 3$								
Lena	0.726	0.707	0.669	0.637	0.622	0.631	0.656	0.694
Peppers	0.714	0.694	0.663	0.625	0.609	0.616	0.642	0.679
Baboon	0.831	0.817	0.795	0.764	0.733	0.716	0.732	0.781
Boats	0.651	0.630	0.600	0.578	0.571	0.577	0.598	0.627
JetF16	0.692	0.660	0.612	0.593	0.595	0.609	0.635	0.670
Tiffany	0.593	0.567	0.541	0.523	0.522	0.535	0.560	0.590
GoldHill	0.748	0.727	0.697	0.661	0.636	0.640	0.669	0.712
Average *CR*	0.708	0.686	0.654	0.626	**0.613**	0.618	0.642	0.679
d) $n = 4$								
Lena	0.789	0.769	0.731	0.699	0.684	0.693	0.718	0.756
Peppers	0.777	0.756	0.725	0.687	0.671	0.679	0.705	0.742
Baboon	0.893	0.879	0.857	0.826	0.795	0.779	0.795	0.843
Boats	0.713	0.692	0.662	0.640	0.633	0.640	0.660	0.690
JetF16	0.754	0.722	0.675	0.656	0.658	0.672	0.697	0.733
Tiffany	0.655	0.630	0.603	0.585	0.584	0.597	0.623	0.653
GoldHill	0.811	0.790	0.760	0.724	0.698	0.702	0.731	0.774
Average *CR*	0.770	0.748	0.716	0.688	**0.675**	0.680	0.704	0.742

Table 5. The effect of different values of m on compression rate of the proposed scheme with $n = 1, 2, 3$, and 4 for the codebook size $N = 1024$

m	1	2	3	4	5	6	7	8	9
a) $n = 1$									
Lena	0.698	0.682	0.655	0.620	0.585	0.573	0.583	0.612	0.655
Peppers	0.673	0.659	0.632	0.599	0.562	0.550	0.562	0.594	0.635
Baboon	0.780	0.771	0.756	0.730	0.700	0.668	0.656	0.676	0.726
Boats	0.612	0.588	0.551	0.517	0.507	0.507	0.520	0.546	0.681
JetF16	0.638	0.605	0.553	0.520	0.513	0.520	0.539	0.570	0.608
Tiffany	0.553	0.515	0.475	0.445	0.438	0.446	0.469	0.502	0.538
GoldHill	0.707	0.693	0.671	0.638	0.599	0.576	0.583	0.616	0.663
Average *CR*	0.666	0.645	0.613	0.581	0.558	**0.549**	0.559	0.588	0.644
b) $n = 2$									
Lena	0.760	0.744	0.718	0.682	0.647	0.635	0.645	0.675	0.718
Peppers	0.736	0.721	0.695	0.661	0.624	0.613	0.625	0.656	0.697
Baboon	0.843	0.834	0.818	0.793	0.762	0.730	0.718	0.738	0.789
Boats	0.674	0.651	0.613	0.580	0.570	0.569	0.582	0.608	0.643
JetF16	0.701	0.668	0.615	0.582	0.575	0.583	0.601	0.633	0.671
Tiffany	0.615	0.577	0.538	0.508	0.500	0.509	0.532	0.565	0.600
GoldHill	0.769	0.755	0.733	0.700	0.661	**0.638**	0.646	0.679	0.726
Average *CR*	0.728	0.707	0.676	0.644	0.620	**0.611**	0.621	0.651	0.692
c) $n = 3$									
Lena	0.823	0.807	0.780	0.745	0.710	0.698	0.708	0.737	0.780
Peppers	0.798	0.784	0.757	0.724	0.687	0.675	0.687	0.719	0.760
Baboon	0.905	0.896	0.881	0.855	0.825	0.793	0.781	0.801	0.851
Boats	0.737	0.713	0.676	0.642	0.632	0.632	0.645	0.671	0.706
JetF16	0.763	0.730	0.678	0.645	0.638	0.645	0.664	0.695	0.733
Tiffany	0.678	0.640	0.600	0.570	0.563	0.571	0.594	0.627	0.663

GoldHill	0.832	0.818	0.796	0.763	0.724	0.701	0.708	0.741	0.788
Average *CR*	0.791	0.770	0.738	0.706	0.683	**0.674**	0.684	0.713	0.754
d) $n = 4$									
Lena	0.885	0.869	0.843	0.807	0.772	0.760	0.770	0.800	0.843
Peppers	0.861	0.846	0.820	0.786	0.749	0.738	0.750	0.781	0.822
Baboon	0.968	0.959	0.943	0.918	0.887	0.855	0.843	0.863	0.914
Boats	0.799	0.776	0.738	0.705	0.695	0.694	0.707	0.733	0.768
JetF16	0.826	0.793	0.740	0.707	0.700	0.708	0.726	0.758	0.796
Tiffany	0.740	0.702	0.663	0.633	0.625	0.634	0.657	0.690	0.725
GoldHill	0.894	0.880	0.858	0.825	0.786	0.763	0.771	0.804	0.851
Average *CR*	0.853	0.832	0.801	0.769	0.745	**0.736**	0.746	0.776	0.817

It can be seen from **Tables 2-9** that to obtain the optimal (i.e., smallest value) compression rate with the codebooks of sizes N = 128, 256, 512, and 1024, the selected values of the parameter m for the proposed scheme are m = 4, 4, 5 and 6, respectively. For Lee et al.'s scheme [26], the parameter z is chosen so that the scheme achieves its maximum embedding rate. Thus, the suggested values of the parameter z for this method are z = 2, 4, 8 and 16, respectively.

4.2 Comparing the Proposed Scheme with Previous Schemes

In this section, we compare the performance results of the proposed scheme with the schemes proposed by Chang et al. [24], Wang and Lu [25] with paths 1 and 2 as well as Lee et al. [26]. The comparative results among the simulated methods with regard to average embedding rate *ER*, compression rate *CR*, and embedding efficiency *EE* for the codebooks of sizes N = 128, 256, 512 and 1024 are shown in **Table 10**. It is noted that, the codebook sizes N = 128, 256, 512, and 1024, the selected values of the parameter m for Chang et al.'s scheme [24] are m = 1, 4, 6, and 7, respectively. The suitable values of the parameter m for Wang and Lu's scheme [25] with the path 1 are m = 2, 3, 5, and 6, respectively. The proper values of the parameter m for Wang and Lu's scheme [25] with the path 2 are m = 2, 4, 5, and 6, respectively.

Table 10. Results of average *ER* (bpi), *CR* (bpp), and *EE* of simulated methods with their selected values of m and z for codebook sizes N = 128, 256, 512, and 1024 and test images in **Fig. 4**

Methods	Factors	128	256	512	1024
Chang et al. [24]	*ER*	0.984	0.984	0.984	0.984
	CR	0.400	0.510	0.583	0.649
	EE	0.154	0.121	0.106	0.095
		$m = 1$	$m = 4$	$m = 6$	$m = 7$
Wang and Lu [25] path 1	*ER*	1.953	1.953	1.953	1.953
	CR	0.504	0.573	0.638	0.696
	EE	0.242	0.213	0.191	0.175
		$m = 2$	$m = 3$	$m = 5$	$m = 6$
Wang and Lu [25] path 2	*ER*	2.884	2.884	2.884	2.884
	CR	0.572	0.641	0.706	0.765
	EE	0.315	0.281	0.255	0.236
		$m = 2$	$m = 4$	$m = 5$	$m = 6$
Lee et al. [26]	*ER*	3.128	2.952	2.740	2.679
	CR	0.475	0.546	0.612	0.673
	EE	0.412	0.338	0.280	0.249
		$z = 2$	$z = 4$	$z = 8$	$z = 16$

Proposed with n = 1 (P1)	*ER*	1.000	1.000	1.000	1.000
	CR	0.358	0.420	0.488	0.549
	EE	0.175	0.149	0.128	0.114
		m = 4	m = 4	m = 5	m = 6
Proposed with n = 2 (P2)	*ER*	2.000	2.000	2.000	2.000
	CR	0.420	0.483	0.550	0.611
	EE	0.298	0.259	0.227	0.205
		m = 4	m = 4	m = 5	m = 6
Proposed with n = 3 (P3)	*ER*	3.000	3.000	3.000	3.000
	CR	0.483	0.545	0.613	0.674
	EE	0.388	0.344	0.306	0.278
		m = 4	m = 4	m = 5	m = 6
Proposed with n = 4 (P4)	*ER*	4.000	4.000	4.000	4.000
	CR	0.545	0.608	0.675	0.736
	EE	0.459	0.411	0.370	0.340
		m = 4	m = 4	m = 5	m = 6

While both Chang et al.'s scheme [24] and the proposed scheme with n = 1 produce embedding rates near 1 bpi, the method P1 achieves a true 1 bpi while Chang et al.'s scheme obtains 0.984 bpi. Chang et al.'s scheme does not achieve 1 bpi as it uses the first row and first column as a seed area and does not embed secret data in this area. Additionally, P1 achieves the embedding rate of 1 bpi with significantly lower compression rates than Chang et al.'s scheme across all codebook sizes tested as can be seen in **Table 10**. The reason for Chang et al.'s high compression rates is due to the fact that this scheme uses m padding bits 0's when the difference value $d = 0$ or $|d| > 2^m - 1$. With respect to the embedding efficiency, **Table 10** demonstrates that the embedding efficiencies of the scheme P1 are higher than those of Chang et al.'s scheme [24] regardless of the codebook sizes. This confirms the superiority of the scheme P1 over Chang et al.'s scheme.

A similar trend can be seen from **Table 10** when comparing Wang and Lu's scheme [25] with the path 1 which produces near 2 bpi to the scheme P2. Wang and Lu's method with the path 1 produces the embedding rate of 1.953 bpi while P2 produces a true 2 bpi. As can be seen from **Table 10**, the method P2 achieves a slightly higher embedding rate with much smaller compression rates regardless of the codebook sizes. The higher compression rates of Wang and Lu's scheme [25] with the path 1 are also due to the use of m padding bits 0's in the case where the difference value $d = 0$. For the embedding efficiency comparison, **Table 10** indicates that the embedding efficiencies obtained by Wang and Lu's method [25] with the path 1 are 0.242, 0.213, 0.191, and 0.175 whereas those attained by the method P2 are 0.298, 0.259, 0.227, and 0.205. This demonstrates that the scheme P2 surpasses Wang and Lu's scheme with the path 1.

Both Wang and Lu's scheme [25] with the path 2 and Lee et al.'s scheme [26] produce embedding rates around 3 bpi and are comparable with the scheme P3. **Table 10** shows that Wang and Lu's scheme and P3 both have the constant embedding rates of 2.884 bpi and 3 bpi, respectively, regardless of the codebook sizes. Lee et al.'s method, however, has the variable embedding rates of 3.128, 2.952, 2.740, and 2.679 bpi for the codebooks of sizes 128, 256, 512, and 1024, respectively.

From **Table 10** we can see that Wang and Lu's scheme [25] with the path 2 has a lower embedding rate than the scheme P3. This is due to the fact that their scheme requires a large seed area (i.e., first two rows, first two columns, and the rightmost column of the VQ index

table) whereas P3 truly embeds secret data into every VQ index. Wang and Lu's scheme with the path 2 also has much higher compression rates across all codebook sizes. This is because the utilization of the neighboring VQ indices in computing the difference value d by Wang and Lu's scheme leads to a large value of d. In addition, m padding bits 0's are used by this scheme when the difference value $d = 0$. In terms of the embedding efficiency, it can be observed from **Table 10** that the embedding efficiencies achieved by Wang and Lu's method [25] with the path 2 are 0.315, 0.281, 0.255, and 0.236, those gained by Lee et al.'s [26] scheme are 0.412, 0.338, 0.280, and 0.249, and those obtained by P3 are 0.388, 0.344, 0.306, and 0.278. Thus, it can be deduced that the embedding efficiency of the scheme P3 is superior to that of Wang and Lu's method [25] with the path 2. We can see from **Table 10** that, for the 128 sized codebook, Lee et al.'s method has the *EE* value of 0.412 which is better than the P3's result of 0.388 for the same codebook size. This is supported by their slightly higher embedding rate of 3.128 bpi versus 3 bpi for P3 and a better compression rate of 0.475 bpp versus 0.483 bpp for P3. For the three remaining codebooks, however, the scheme P3 has the better *EE* values of 0.344, 0.306, and 0.278 whereas the *EE* values of Lee et al.'s method are 0.338, 0.280, and 0.249. This is due to the fact that Lee et al.'s scheme embeds less secret data when the codebook size increases whereas our scheme remains the constant embedding rate regardless of the codebook size. Additionally, with the exception of the 128 sized codebook, the compression rate of our method stays within 0.001 of theirs. Thus, it can be concluded that the performance of Lee et al.'s method is better than that of the scheme P3 for the 128 sized codebook and the scheme P3 has a better performance than Lee et al.'s scheme for the codebooks sized 256, 512, and 1024.

Table 11. Comparative results of Lee et al.'s scheme [26] and P3 for smooth image Tiffany and complex image Baboon

a) Tiffany image					
Methods	Factors	128	256	512	1024
Lee et al. [26]	*ER*	4.345	4.281	4.186	4.275
	CR	0.444	0.515	0.585	0.642
	EE	0.612	0.520	0.447	0.416
		$z = 2$	$z = 4$	$z = 8$	$z = 16$
Proposed with $n = 3$	*ER*	3	3	3	3
	CR	0.413	0.458	0.522	0.571
	EE	0.454	0.409	0.359	0.328
		$m = 4$	$m = 4$	$m = 5$	$m = 6$
b) Baboon image					
Lee et al. [26]	*ER*	1.447	1.125	0.992	0.942
	CR	0.514	0.584	0.648	0.709
	EE	0.176	0.120	0.096	0.083
		$z = 2$	$z = 4$	$z = 8$	$z = 16$
Proposed with $n = 3$	*ER*	3	3	3	3
	CR	0.586	0.669	0.733	0.793
	EE	0.320	0.280	0.256	0.236
		$m = 4$	$m = 4$	$m = 5$	$m = 6$

Lee et al.'s method [26] has an excellent performance in terms of the embedding rate, compression rate, and embedding efficiency for smooth images such as the Tiffany image, as shown in **Table 11**. For the Tiffany image, their method is capable of attaining the embedding rates of 4.345, 4.281, 4.186, and 4.275 bpi at the compression rates of 0.444, 0.515, 0.585, and 0.642 bpp for the codebooks sized 128, 256, 512, and 1024, respectively. Thus, it is clear that

Lee et al.'s method surpasses the scheme P3 for smooth images. As can be seen from **Table 11**, however, Lee et al.'s method performs poorly when applied to complex images (e.g., Baboon image) and this effect worsens as the codebook size increases. We can clearly see this from the results of the Baboon image. Thus, it can be said that the scheme P3 is superior to Lee et al.'s method for complex images.

None of the previous schemes [24-26] achieves average embedding rates around 4 bpi. From **Table 10**, we can see that, for the 128 sized codebook, the scheme P4 achieves the embedding rate of 4 bpi at the average compression rate of 0.545 bpp that is within an acceptable range. Even though for the codebook of size 256, the scheme P4 has a higher compression rate of 0.608 bpp for the embedding rate of 4 bpi. Thus, we believe that this is still viable. **Table 10** further shows that there is a jump in the compression rate for the 512 and 1024 sized codebooks, making use of P4 at these codebook sizes less alluring.

The visual quality of the reconstructed images was evaluated by using peak signal-to-noise ratio (*PSNR*) which is defined as follows.

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE} \text{ (dB), where } MSE = \frac{1}{H \times W} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} (I(i, j) - I'(i, j))^2 .$$

The *MSE* (mean square error) is the difference between the original cover image I sized $H \times W$ and the reconstructed image I' sized $H \times W$, and $I(i, j)$ and $I'(i, j)$ are the values of the pixels located at the ith row and jth column of I and I', respectively. Because the simulated schemes reversibly conceal secret bits into VQ indices in a VQ index table, the original VQ indices can be completely recovered. Therefore, the *PSNR* values of the simulated methods are the same as those of the VQ compression method. **Fig. 5** shows the *PSNR* values of the simulated schemes for the codebook size $N = 256$.

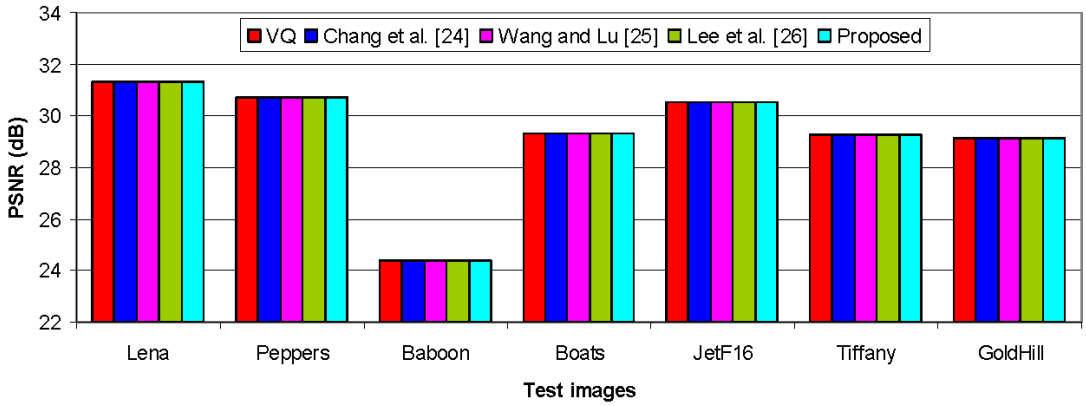

Fig. 5. Visual quality results of restored cover images of simulated schemes.

5. Conclusion

In this paper, we propose a novel lossless data hiding method for VQ indices based on joint neighboring coding. The proposed method embeds n secret bits into one VQ index, where n = 1, 2, 3, 4. The proposed approach can obtain the embedding rates of 1, 2, 3, and 4 bpi. Using the embedding efficiency *EE* as a fair comparison factor among the simulated methods, our method surpasses the three previous works, namely Chang et al.'s [24] and Wang and Lu's [25] schemes for all codebook sizes except Lee et al.'s method [26] for the 128 sized codebook.

Thus, we can conclude that our method is a viable method for use in a data hiding application such as digital libaries or secret communications.

Appendix

Table 6. The effect of different values of z on embedding rate and compression rate of Lee et al.'s scheme [26] for the codebook size $N = 128$

Images	z	1	2	4	8
Lena	*ER*	3.074	3.142	3.086	2.899
	CR	0.483	0.475	0.467	0.456
Peppers	*ER*	3.155	3.220	3.157	2.992
	CR	0.481	0.473	0.467	0.455
Baboon	*ER*	1.364	1.447	1.387	1.131
	CR	0.525	0.514	0.502	0.484
Boats	*ER*	3.423	3.489	3.428	3.276
	CR	0.474	0.467	0.462	0.455
JetF16	*ER*	3.543	3.609	3.538	3.397
	CR	0.472	0.464	0.460	0.453
Tiffany	*ER*	4.340	4.345	4.239	4.090
	CR	0.450	0.444	0.442	0.440
GoldHill	*ER*	2.597	2.644	2.567	2.354
	CR	0.494	0.486	0.477	0.460
Average *ER*		3.071	**3.128**	3.057	2.877
Average *CR*		0.483	**0.475**	0.468	0.458

Table 7. The effect of different values of z on embedding rate and compression rate of Lee et al.'s scheme [26] for the codebook size $N = 256$

Images	z	1	2	4	8	16
Lena	*ER*	2.604	2.718	2.841	2.757	2.405
	CR	0.569	0.562	0.548	0.531	0.517
Peppers	*ER*	2.764	2.905	2.987	2.932	2.593
	CR	0.566	0.558	0.546	0.529	0.516
Baboon	*ER*	0.934	1.026	1.125	1.074	0.711
	CR	0.603	0.597	0.584	0.565	0.542
Boats	*ER*	3.419	3.482	3.557	3.476	3.214
	CR	0.552	0.547	0.537	0.525	0.516
JetF16	*ER*	3.361	3.543	3.564	3.423	3.139
	CR	0.553	0.543	0.533	0.524	0.515
Tiffany	*ER*	4.129	4.297	4.281	4.099	3.787
	CR	0.535	0.525	0.515	0.507	0.503
GoldHill	*ER*	2.111	2.221	2.306	2.251	1.889
	CR	0.579	0.572	0.560	0.540	0.521
Average *ER*		2.760	2.885	**2.952**	2.859	2.534
Average *CR*		0.565	0.558	**0.546**	0.532	0.519

Table 8. The effect of different values of z on embedding rate and compression rate of Lee et al.'s scheme [26] for the codebook size $N = 512$

Images	z	1	2	4	8	16	32
Lena	*ER*	2.158	2.303	2.457	2.592	2.393	1.913
	CR	0.647	0.641	0.631	0.611	0.593	0.580
Peppers	*ER*	2.306	2.436	2.601	2.739	2.597	2.145

	CR	0.645	0.640	0.630	0.612	0.593	0.578
Baboon	*ER*	0.664	0.766	0.884	0.992	0.919	0.508
	CR	0.675	0.670	0.663	0.648	0.627	0.601
Boats	*ER*	3.166	3.267	3.414	3.480	3.330	2.974
	CR	0.630	0.625	0.616	0.603	0.590	0.579
JetF16	*ER*	2.633	2.784	2.984	3.012	2.749	2.290
	CR	0.638	0.631	0.618	0.601	0.589	0.579
Tiffany	*ER*	4.039	4.093	4.224	4.186	3.957	3.559
	CR	0.612	0.607	0.596	0.585	0.574	0.567
GoldHill	*ER*	1.815	1.949	2.085	2.177	2.067	1.619
	CR	0.654	0.648	0.639	0.624	0.603	0.581
Average *ER*		2.397	2.514	2.664	**2.740**	2.573	2.144
Average *CR*		0.643	0.637	0.628	**0.612**	0.596	0.581

Table 9. The effect of different values of z on embedding rate and compression rate of Lee et al.'s scheme [26] for the codebook size $N = 1024$

Images	*z*	1	2	4	8	16	32	64
Lena	*ER*	1.691	1.819	2.033	2.226	2.309	2.078	1.528
	CR	0.723	0.719	0.710	0.697	0.677	0.657	0.643
Peppers	*ER*	2.023	2.144	2.333	2.510	2.586	2.384	1.852
	CR	0.717	0.714	0.706	0.694	0.675	0.655	0.641
Baboon	*ER*	0.501	0.579	0.694	0.834	0.942	0.833	0.386
	CR	0.742	0.739	0.734	0.725	0.709	0.688	0.662
Boats	*ER*	2.881	3.035	3.231	3.445	3.441	3.167	2.721
	CR	0.704	0.700	0.692	0.677	0.662	0.652	0.641
JetF16	*ER*	2.521	2.761	3.102	3.269	3.156	2.802	2.277
	CR	0.709	0.702	0.689	0.673	0.660	0.649	0.639
Tiffany	*ER*	3.795	4.076	4.325	4.403	4.275	3.904	3.397
	CR	0.689	0.680	0.668	0.655	0.642	0.633	0.628
GoldHill	*ER*	1.538	1.644	1.780	1.948	2.041	1.906	1.389
	CR	0.725	0.722	0.716	0.705	0.688	0.664	0.642
Average *ER*		2.136	2.294	2.500	2.662	**2.679**	2.439	1.936
Average *CR*		0.716	0.711	0.702	0.689	**0.673**	0.657	0.642

References

[1] R.M. Davis, "The data encryption standard in perspective," *IEEE Communications Magazine*, vol. 16, no. 6, pp. 5-9, 1978. Article (CrossRef Link)

[2] M.A. Wright, "The advanced encryption standard," *Network Security 2001*, vol. 10, pp. 11-13, 2001. Article (CrossRef Link)

[3] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978. Article (CrossRef Link)

[4] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*, 2nd Edition, Morgan Kauffman, 2007. Article (CrossRef Link)

[5] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996. Article (CrossRef Link)

[6] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn, "Information hiding - a survey," in *Proc. of IEEE Special Issue on Protection of Multimedia Content*, vol. 87, no. 7, pp. 1062-1078, 1999. Article (CrossRef Link)

[7] Y. Hu, H. Lee, and J. Li, "DE-based reversible data hiding with improved overflow location map,"

IEEE Transactions on Circuits and Systems for Video Technology, vol. 19, no. 2, pp. 250-260, 2009. Article (CrossRef Link)
[8] F. Peng, X. Li, and B. Yang, "Improved PVO-based reversible data hiding," *Digital Signal Processing*, vol. 25, pp. 255-265, 2014. Article (CrossRef Link)
[9] D. Coltuc, "Low distortion transform for reversible watermarking," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 412-417, 2012. Article (CrossRef Link)
[10] C.H. Yang, C.Y. Weng, S.J. Wang, and H.M. Sun, "Adaptive data hiding in edge areas of images with spatial LSB domain systems," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 488-497, 2008. Article (CrossRef Link)
[11] W. Hong and T.S. Chen, "A novel data embedding method using adaptive pixel pair matching," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 176-184, 2012. Article (CrossRef Link)
[12] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 831-841, 2003. Article (CrossRef Link)
[13] A.M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147-1156, 2004. Article (CrossRef Link)
[14] D.M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 721-730, 2007. Article (CrossRef Link)
[15] X. Zhang, S. Wang, Z. Qian, and G. Feng, "Reversible fragile watermarking for locating tampered blocks in JPEG images," *Signal Processing*, vol. 90, no. 12, pp. 3026-3036, 2010. Article (CrossRef Link)
[16] C.C. Chang, P.Y. Pai, C.M. Yeh, and Y.K Chan, "A high payload frequency-based reversible image hiding method," *Information Sciences*, vol. 180, no. 11, pp. 2286-2298, 2010. Article (CrossRef Link)
[17] J.S. Pan, M.T. Sung, H.C. Huang, and B.Y. Liao, "Robust VQ-based digital watermarking for the memoryless binary symmetric channel," *IEICE Transactions on Fundamentals*, vol. E-87A, no. 7, pp. 1839-1841, 2004. Article (CrossRef Link)
[18] C.H. Yang and Y.C. Lin, "Reversible data hiding of a VQ index table based on referred counts," *Journal of Visual Communication and Image Representation*, vol. 20, no. 6, pp. 399-407, 2009. Article (CrossRef Link)
[19] C.C Chang, T.D. Kieu, and Y.C Chou, Reversible information hiding for VQ indices based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 20, no. 1, pp. 57-64, 2009. Article (CrossRef Link)
[20] C.H. Yang and Y.C. Lin, "Fractal curves to improve the reversible data embedding for VQ-indexes based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 21, no. 4, pp. 334-342, 2010. Article (CrossRef Link)
[21] C.C. Chang, T.S. Nguyen, and C.C Lin, "A reversible data hiding for VQ indices using locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 22, no. 7, pp. 664-672, 2011. Article (CrossRef Link)
[22] T.D. Kieu, A. Rudder, and W. Goodridge, "A reversible steganographic scheme for VQ indices based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 25, no. 6, pp. 1378-1386, 2014. Article (CrossRef Link)
[23] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4-29, 1984. Article (CrossRef Link)
[24] C.C. Chang, T.D. Kieu, and W.C. Wu, "A lossless data embedding technique by joint neighboring coding," *Pattern Recognition*, vol. 42, no. 7, pp. 1597-1603, 2009. Article (CrossRef Link)
[25] J.X. Wang and Z.M. Lu, "A path optional lossless data hiding scheme based on VQ joint neighboring coding," *Information Sciences*, vol. 179, no. 19, pp. 3332-3348, 2009. Article (CrossRef Link)
[26] J.D. Lee, Y.H. Chiou, and J.M. Guo, "Lossless data hiding for VQ indices based on neighboring correlation," *Information Sciences*, vol. 221, pp. 419-438, 2013. Article (CrossRef Link)

[27] T.D. Kieu and S. Ramroach, "A reversible steganographic scheme for VQ indices based on joint neighboring coding," *Expert Systems with Applications*, vol. 42, no. 2, pp. 713-722, 2015. Article (CrossRef Link)
[28] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84-95, 1980. Article (CrossRef Link)

Andrew Rudder received the B.S. degree in Computer Science in 2003 and M.S. degree in Computer Science in 2006 from The University of the West Indies, St. Augustine, Trinidad and Tobago. Currently, he is a Ph.D. candidate in Computer Science at The University of the West Indies, St. Augustine, Trinidad and Tobago. Since 2007, he has been with the Department of Computing and Information Technology, Faculty of Science and Technology, The University of the West Indies, St. Augustine, Trinidad and Tobago, where he is currently an assistant Lecturer. His research interests include information hiding, data compression, and image processing.

The Duc Kieu received the B.S. degree in Mathematics from the University of Pedagogy, Vietnam, in 1995, the B.S. degree in Information Technology from the University of Natural Sciences, Vietnam, in 1999, the M.S. degree in Computer Science from Latrobe University, Australia, in 2005, and the Ph.D. degree in Computer Science from Feng Chia University, Taiwan, in 2009. Since 2010, he has been with the Department of Computing and Information Technology, Faculty of Science and Technology, The University of the West Indies, St. Augustine, Trinidad and Tobago, where he is currently a Lecturer. His research interests include information hiding, data compression, and image processing.