International Journal of Security and Its Applications
Vol.7, No.6 (2013), pp.209-228
http://dx.doi.org/10.14257/ijsia.2013.7.6.22

# Grayscale Image Tamper Detection and Recovery Based on Vector Quantization

Jun-Chou Chuang[1], Yu-Chen Hu[2], Chun-Chi Lo[3] and Wu-Lin Chen[2]

[1]*Department of Computer Science and Communication Engineering, Providence University, 200 Chung Chi Rd., Taichung 43301, Taiwan*
[2]*Department of Computer Science and Information Management, Providence University, 200 Chung Chi Rd., Taichung 43301, Taiwan*
[3]*Department of Computer Science and Information Engineering, Providence University, 200 Chung Chi Rd., Taichung 43301, Taiwan*
*lzchung@pu.edu.tw, ychu@pu.edu.tw, wlchen@pu.edu.tw, cclo@pu.edu.tw*

***Abstract***

*This paper proposes a tamper detection and image recovery method for digital images. The proposed method not only locates the alterations but also recovers the tamper regions. In this method, the vector quantization scheme is employed to generate the authentication data. The image to be protected is first compressed by the vector quantization scheme to generate the indices for image blocks. Multi-copies of the indices are embedded into the selected blocks by the pseudo random number generator.*

*To detect the tamper regions, the authenticaiton procedure employes the forward detection strategy and the backward detection mechanism to find out the image modifications. The tamper regions can then be recovered by using the authentication data that was embeded previously if needed. From the experimental results, it is shown that the tamper regions can be clearly detected and well recovered.*

***Keywords:*** *Image authentication, tamper detection, image recovery, digital watermarking, vector quantization*

## 1. Introduction

With the rapid growing of computer hardware and software, people can download all kind of digital media from the Internet easily and quickly. Du to the ease of digital duplication and modification, data security [1] becomes a very important research topic. Traditional data encryption schemes are very suitable for text data or password encryption, but they are not flexibility for the protection of multimedia contents.

There are a lot of media protection schemes such as digital right management (DRM), digital watermarking, image tamper detection, and so on. The goal of DRM is to management the access right of the protected media. The major components of DRM are the end users, the content producers, and the access right controls. People who get the access rights would be able to use the protected media. The access rights include printing, copying, modifying, moving, regions, times, and so on. Sometimes, DRM can be combined with a special hardware device produced by the hardware manufacturers or the copyright holders in order to prevent the media from being illegal distributed and copied. The DVD protection, E-Book protection, music protection, game protection, and video protection are typical applications of DRM.

Among the digital media, the digital images are popularly used in most multimedia applications. To protect the digital images, two main approaches had been introduced. They are the digital watermarking approach and the image authentication approach. Both approaches embed the watermarks into the protected image. The digital watermarking

ISSN: 1738-9976 IJSIA

approach is often called the robust image watermarking because it can resist various image attacks such as blurring, sharpening, zooming, rotation, cutting, and compression. The ownership of the protected image can be identified by extracting the embedded watermarks. Anyone who does not have the ownership cannot remove the embedded watermarks easily.

**Table 1. The Comparison of Digital Watermarking and Image Authentication**

| Approach | Alias name | Embedding payload | Error resistant | Image compression | Application |
|---|---|---|---|---|---|
| Digital watermarking | Robust watermarking | Small | High | Allow | ownership protection |
| Image authentication | Fragie watermarking | Large | Low | Low | Integrity checking |

The image authentication approach is also called the fragile watermarking. It works on the protection of the integrity of an image. Because the embedded watermarks are very sensitive to the changes, any change in the image can be identified and treated as the illegal modifications. Some image authentication methods only work on the detection of tamper regions. Some other image authentication methods additionally provide the image recovery function. In Table 1, we listed the comparison of digital watermarking and image authentication.

Generally, the image authentication methods can be classified into hard authentication [2-9] and soft authentication [10-19]. The main difference between them is that soft authentication allows moderate image modifications; however hard authentication does not allow image modifications. Some image authentication methods work on the raw data of the image. In addition, some other authentication methods work on the compressed images. That is because the digital images are typically stored in the compressed form so that a great deal of storage cost can be saved.

In this paper, we will propose an image tamper detection and recovery method. The proposed method works on the protection of the grayscale image in raw format. The image to be protected is encoded by the vector quantization (VQ) [20] scheme to generate the indices for image blocks. Multi-copies of VQ indexes are embedded into the selected image blocks. When the given image is to be authenticated, the forward and the backward detection mechanism are used to detect the tamper regions. Besides, the image recovery procedure is executed if we need to recover the tamper regions. The rest of this paper is organized as the follows. We will give a review of the vector quantization scheme in Section 2. The proposed scheme, including the embedding procedure, tamper detection procedure, and tamper recovery procedure will be presented in Section 3. The experimental results will be described in Section 4. Finally, conclusions will be given in Section 5.

## 2. The Vector Quantization Scheme

The vector quantization scheme [20-28] is a lossy compression scheme for the grayscale images. The VQ technique includes three main procedures: codebook generation, image encoding, and image decoding. The flowchart of the image encoding and image decoding procedures of VQ is depicted in Figure 1. The goal of the codebook generation procedure is to generate a set of representative codewords. The set of codewords is often called the codebook *CB*. The codebook $CB=\{cw_i \mid i=1, 2,\ldots, N\}$ will be used in the image encoding and image decoding procedures. Let $N$ denotes the size of the codebook and $cw_i$ denotes the $i$-th codeword in the codebook. From the literature, the

LBG algorithm [20] that was proposed by the Linde *et al.*, is the most commonly used algorithm for codebook generation.

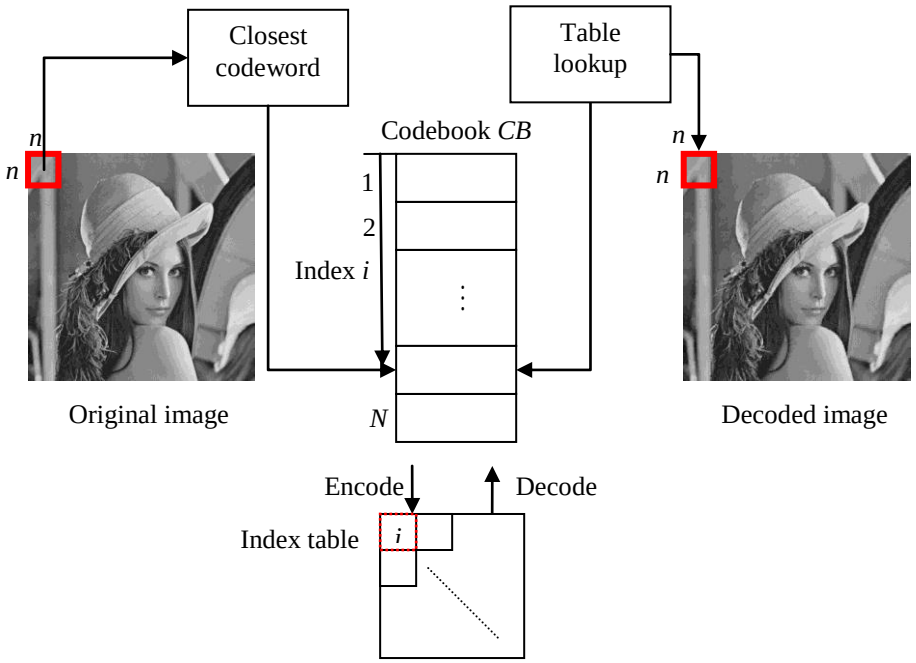



**Figure 1. The Flowchart of VQ Encoding and Decoding**

In the image encoding procedure, the given grayscale image is segmented into non-overlapping image blocks of $n \times n$ pixels. Each $n \times n$ image block can be viewed as a $k$ -dimensional image vector where $k = n \times n$. Each image block is then sequentially encoded in the order of left-to-right and top-to-bottom. Basically the image encoding procedure maps a $k$-dimensional Euclidean space of input image vector $x$ ( $x \in R^k$ ) to a finite subset $CB$, where $R^k \rightarrow CB$. For each image vector $x$, we search the closest codeword by calculating the Euclidean distance of $x$ and each codeword $cw_i$ in the codebook by the following equation:

$$d(x, cw_i) = \sum_{j=1}^{k} (x_j - cw_{ij})^2 , \quad (1)$$

where $x_j$ denotes the $j$-th element of image block $x$, and $c_{ij}$ denotes the $j$-th element of codeword $cw_i$. The codeword that has the minimal squared Euclidean distance with $x$ is selected as the closest codeword. Next, we record this index of the closest codeword. After each block is sequentially processed, the set of indices are collected to form an index table. Each index stored in the index table is of $\log_2 N$ bits.

In the image decoding procedure, each index of $\log_2 N$ bits is sequentially extracted from the index table to reconstruct each image block by performing a simple table lookup operation. The codeword in the codebook for each index is selected to rebuild each image block. After each compressed image block is sequentially reconstructed, the compressed image of VQ is then rebuilt.

## 3. The Proposed Method

The proposed method consists of the watermark generation procedure, the watermark embedding procedure, the tamper detection procedure, and the image recovery procedure. First of all, we present how to generate the watermarks. In the watermark embedding procedure, multi-copies of watermark are embedded into the selected image blocks of the image to be protected. In the tamper detection procedure, the forward detection and the backward detection mechanisms will be described. Finally, the image recovery procedure will be introduced.

### 3.1. The Watermark Generation Procedure

Suppose the image $I$ to be protected is of $W \times H$ pixels. Let $eb$ denotes the number of modified bits of each pixel for data embedding. The last $eb$-bits of each image pixel in $I$ are replaced by 0 to generate the modified image $MI$. The modified image $MI$ is then divided into non-overlapping image blocks of $n \times n$ pixels. Let *vectornum* denotes the total number of image blocks. The following equation can be used to compute the value of v*ectornum*.

$$vectornum = \frac{W \times H}{n \times n}. \quad (2)$$

Then the VQ scheme is used to generate the watermark of each image block $x$. Suppose the codebook $CB$ of $N$ codewords is previously generated. The closest codeword for $x$ is searched and index is recorded. The index of $\log_2 N$ bits is taken as the watermark for $x$. The block diagram of watermark generation is listed in Figure 2.

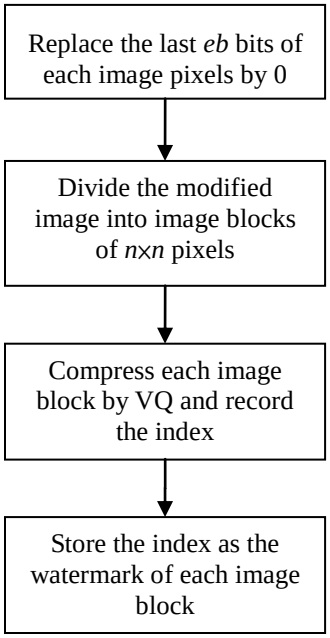



**Figure 2. The Flowchart of the Watermark Generation Procedure**

### 3.2. The Watermark Embedding Procedure

In the watermark embedding procedure, multiple copies of the watermark of each image block are stored into the last $eb$-bits of the pixels in the selected blocks. The flowchart of the watermark embedding procedure is depicted in Figure 3. Let *sno* denotes

the number of copies of the watermark to be embedded. The value of *sno* can be calculated by the following equation.

$$sno = \left\lfloor \frac{W \times H \times eb}{vectornum \times \log_2 N} \right\rfloor. \quad (3)$$

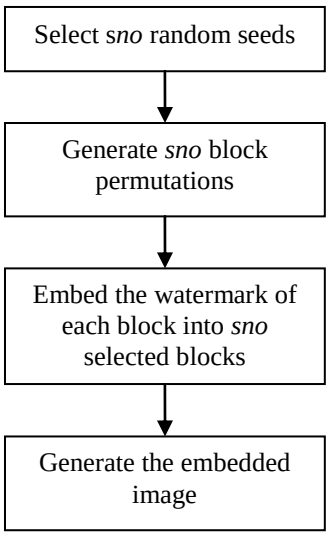



**Figure 3. The Flowchart of the Watermarks Embedding Procedure**

The *sno* copies of watermark are to be embedded into *sno* image blocks. To determine the image blocks to embed these *sno* copies of watermark, *sno* random seeds that were previously chosen are used. The random sequence induces by each random seed is employed to determine one permutation of the image blocks. The size of each block permutation is of size *vectornum*. Let $BP_i$ denotes the *i*-th block permutation generated by using the random sequence induced by *i*-th random seed. The block numbers record in the $BP_i$ will be used to embed the *i*-th copy of the watermark of each image block. Here, $BP_{i,j}$ denotes the the *i*-th block permutation of the *j*-th block. The *i*-th copy of the watermark of *j*-th image block is to be embedded into the image block numbered $BP_{i,j}$. By sequentially embedding each copy of the watermark of each image block into the selected block based on these block permutations, a total of *sno* copies of watermark of each image block are embedded into *I* to generated the embedded image *EI*.

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

(a) Block numbers

| | | | |
|---|---|---|---|
| 4 | 1 | 11 | 10 |
| 13 | 7 | 16 | 5 |
| 2 | 14 | 15 | 9 |
| 8 | 3 | 6 | 12 |

(b) First permutation

| | | | |
|---|---|---|---|
| 15 | 5 | 13 | 14 |
| 4 | 3 | 9 | 8 |
| 1 | 2 | 12 | 16 |
| 7 | 10 | 6 | 11 |

(c) Second permutation

**Figure 4. Example of the Block Permutation**

An example of the permutation of 16 blocks is shown in Figure 4. In this example, the value of *sno* is set to 2. The original block numbers are shown in Figure 4(a). The two block permutations generated by using two random seeds valued 111 and 133 are shown in Figures. 4(a) and 4(b), respectively. When the watermark of the first image block is to be embedded, two copies of the watermark will be embedded into the 4-th block and 15-th block, respectively. Similarly, two copies of the watermark of the second image block will be embedded into the first image block and the 5-th image block, respectively.

In the watermark embedding procedure, *sno* copies of the watermark of each image block are stored into *sno* selected image blocks, respectively. In other words, each image block is used to store *sno* copies of watermarks. A total of ($sno \times \log_2 N$) bits are embedded into the last *eb* bits of the pixels in each image block of $n \times n$ pixels. If the total number of bits to be embedded are less than $eb \times (n \times n)$, partial data in the last eb bits of the pixels are not used for data embedding.

| | | | |
|---|---|---|---|
| 119 | 116 | 117 | 117 |
| 120 | 119 | 119 | 123 |
| 118 | 115 | 117 | 120 |
| 118 | 120 | 118 | 115 |

(a) Original image block

| | | | |
|---|---|---|---|
| $(01110111)_2$ | $(01110100)_2$ | $(01110101)_2$ | $(01110101)_2$ |
| $(01111000)_2$ | $(01110111)_2$ | $(01110111)_2$ | $(01111011)_2$ |
| $(01110110)_2$ | $(01110011)_2$ | $(01110101)_2$ | $(01111000)_2$ |
| $(01110110)_2$ | $(01111000)_2$ | $(01110110)_2$ | $(01110011)_2$ |

(b) Binary representation of the first block

| | | | |
|---|---|---|---|
| 118 | 117 | 117 | 117 |
| 121 | 119 | 118 | 123 |
| 119 | 115 | 116 | 120 |
| 118 | 120 | 118 | 114 |

(c) Embedded block

| | | | |
|---|---|---|---|
| $(01110110)_2$ | $(01110101)_2$ | $(01110101)_2$ | $(01110101)_2$ |
| $(01111001)_2$ | $(01110111)_2$ | $(01110110)_2$ | $(01111011)_2$ |
| $(01110111)_2$ | $(01110011)_2$ | $(01110100)_2$ | $(01111000)_2$ |
| $(01110110)_2$ | $(01111000)_2$ | $(01110110)_2$ | $(01110010)_2$ |

(d) Binary representation of the embedded block

**Figure 5. An example of the Watermark Embedding Procedure**

An example of watermark embedding procedure when *sno* is set to 2 is depicted in Figure 5. Figure 5(a) shows the first image block of 4×4 pixels. According to the two block permutations listed in Figure 4, we find that the watermark of the second block is to be embedded into the first block from the first block permutation. Similarly, the watermark of the 9-th block is to be embedded into the first block as shown in the second block permutation. Suppose these two 8-bit watermarks of the second block and the 9-th block are 125 and 200, respectively. The binary representation of 125 and 200 are $(01111101)_2$ and $(11001000)_2$, respectively. A total of 16 bits are to be embedded into the 4×4 image block. In this example, the number of embedded bits of each pixel (*eb*) equals 1. The first watermark 125 is embedded into the first eight pixels. The other watermark 200 is embedded into the last eight pixels. The embedded result of the first block is listed in Figure 5(c). To illustrate the embedded process, the binary representation of the embedded block is listed in Figure 5(d).

### 3.3. The Tamper Detection Procedure

Let *T* denotes the image of $W \times H$ pixels to be authenticated. The system parameters including *sno*, *eb*, *n*, and *N* should be known before the tamper detection procedure is executed. In addition, the same codebook *CB* of *N* codewords is stored and used here. These *sno* random seeds used to determine the block permutations should be also stored.

To extract *sno* copies of the watermark of each image block, these *sno* block permutations that are induced by the random sequences should be generated first. Based on these *sno* block permutations, we can determine the *sno* selected blocks that were used to embed the watermark so that we can extract the embedded data. Here, $w_{i,j}$ denotes the extracted the *j*-th copy of the watermark for image block *i*. By sequentially extracting *sno* copies of the watermark for each image block, the set of the embedded data is now available.

The image *T* to be authenticated is taken to generate another copy of the watermark for each image block. The last *eb*-bits of each image pixel in *T* are replaced by 0 to generate the modified image *TI*. The modified image *TI* is then divided into non-overlapping $n \times n$ image blocks. Then the VQ scheme is used to generate the watermark of each image block by using the codebook *CB*. The closest codeword in the codebook for each image

block is searched and the index of the closest codeword is taken as the watermark. Here, $w_{i,vq}$ denotes the watermark for the *i*-th image block generated by VQ.

To check whether each image block is tampered or not, we apply the proposed forward detection and the backward detection mechanisms sequentially. The forward detection mechanism will be used to generate the forward detected image *FDI*. Besides, the backward detection mechanism will be used to generate the backward detected image *BDI*. The tamper detected image is then generated by *FDI* and *BDI*.

The forward detection mechanism consists of three processes: initial block detection, the majority voting, and the isolated block removal. These three processes are sequentially executed to generate *FDI*. The details of the forward detection mechanism are described in the following section.

### 3.3.1. The Forward Detection Mechanism

(a) Initial Block Detection

For the image block numbered *i*, we need to check whether these (*sno*+1) copies of the watermark including $w_{i,vq}$ and $w_{i,j}$ where $1 \leq j \leq sno$ are identical. If they are identical, this image block is classified as a clear block. Otherwise, it is classified as a tampered block. By sequentially processing each image block using the above-mentioned rule, the first-stage *FDI* can be generated.

(b) Majority Voting

The image type of each block in the first-stage *FDI* can be either a clear block or a tampered block. Note that a clear block may be classified as a tampered block because some copies of the watermark that were embedded in the other blocks are tampered. The majority voting process can be used to solve the mentioned problem when the number of modified watermark is less than or equal to *sno*/2. In the major voting process, each tampered block numbered *i* is changed to a clear block if the total number of the watermark $w_{i,j}$ that is equal to $w_{i,vq}$ is greater than or equal to *sno*/2. We assume some minor copies of the watermark are damaged because the blocks they were embedded are tampered. By sequentially processing each tampered block using the above-mentioned rule, the second-stage *FDI* is be generated.

(c) Isolated Block Removal

The goal of the isolated block removal is to further refine the second-stage *FDI*. I may happen that one image block is not tampered but it is classified as a tampered block because more than *sno*/2 copies of the watermark that were embedded in the other blocks are tampered. This situation may be happen rarely and detected result of the image block has high possibility to be an isolated tamper block. Since the tamper regions are typically much larger than one single image block, these isolated tampered blocks should be removed. To remove the isolated tamper blocks, we need to check whether each tampered block is an isolated block. Then, each isolated tampered block is changed to a clear block. After performing the isolated block removal process, the forward detected image *FDI* is now available.

An example of the forward tamper detection for the first image block is described in the following. Four tampered examples are shown in Figure 6. Based on the block permutations shown in Figure 4, two copies of the watermark of the first block are embedded into the 4-th and 15-th image bocks, respectively. In the first tampered example as shown in Figure 6(a), four image blocks numbered 3, 4, 7, 8 are tampered. Suppose $w_{1,vq}$ equals to $w_{1,1}$ and $w_{1,2}$ is damaged. When the initial block detection process is executed, image block is classified as a tampered image block. But, it is then changed to a clear image block after the majority voting process is executed.

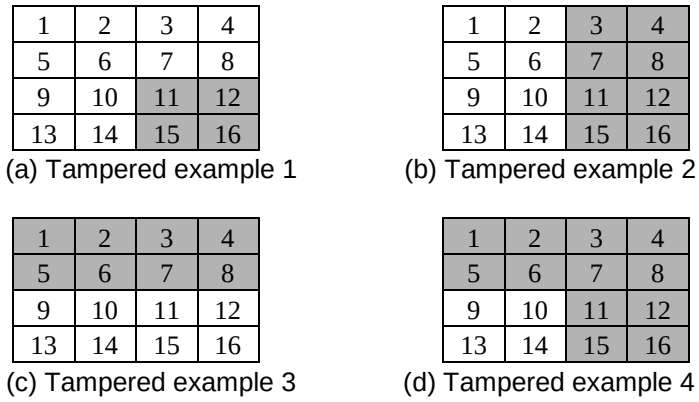



**Figure 6. Tampered Examples**

In the second tampered example as shown in Figure 6(b), suppose $w_{1,2}$ and $w_{1,1}$ are damaged. When the initial block detection process is executed, image block is classified as a tampered image block. In addition, it is still classified as a tampered image block after the majority voting process is executed. Suppose this block is an isolated tampered block, then it is changed to the clear block after the isolated block removal process is executed.

In the third tampered example as shown in Figure 6(c), suppose $w_{1,vq}$ and $w_{1,1}$ are damaged. When the initial block detection process is executed, image block is classified as a tampered block. In addition, it is classified as a tampered block after the majority voting process is executed. In the last tampered example as shown in Figure 6(d), suppose $w_{1,vq}$, $w_{1,1}$, and $w_{1,2}$ are damaged. When the initial block detection process is executed, this image block is classified as a tampered block. It is also classified as a tampered block after the majority voting process is executed. The detected results of different stages of the forward detection mechanism for the first image block are listed in Table 2.

**Table 2. Results of the Forward Detection Mechanism for the First Image Block**

| Process \ Example | Example 1 | Example 2 | Example 3 | Example 4 |
|---|---|---|---|---|
| Initial block detection | tampered block | tampered block | tampered block | tampered block |
| Majority voting | clear block | tampered block | tampered block | tampered block |
| Isolated block removal | N/A | clear block | N/A | N/A |

### 3.3.2 The Backward Detection Mechanism

The backward detection mechanism is to find out the tamper blocks based on the damaged watermark found in the forward detection process. Recall that the damaged watermark may be found either in the majority voting process or the isolated block removal process. The detailed descriptions of how to determine the tampered blocks using the damaged watermark to generate the backward detected image *BDI* are described in the following.

(a) Backward Detection from Majority Voting

In the majority voting process, the tampered block numbered $i$ is changed to a clear block if the total number of the watermark $w_{i,j}$ that is equal to $w_{i,vq}$ is greater than or equal to $sno/2$. Each watermark $w_{i,m}$ where $1 \leq m \leq sno$ that is not equal to $w_{i,vq}$ is treated as a

damaged watermark. Based on the *m*-th block permutation, we know that this watermark is embedded into the selected block numbered $BP_{m,i}$. Then, the $BP_{m,i}$-th image block is classified as a tampered block in *BDI*. By sequentially processing each damaged watermark, the first-stage *BDI* is generated.

Continue the tampered example as shown in Figure 6(a), the first image block is classified as a tampered image block after the initial block detection process is executed. But, it is then classified as a clear image block after the major voting process is performed. In this example, the watermark $w_{1,2}$ is treated as a damaged watermark. Based on the first block permutation $BP_2$, we find that the watermark $w_{1,2}$ is embedded in the image block numbered $BP_{2,1}$. From Fig. 4(c), we find that $BP_{2,1}$ is 15. Therefore, the 15-th image block is classified as a tampered image block in the backward detection process.

(b)Backward Detection from Isolated Block Removal

In the isolated block removal process, the tampered block numbered *i* is changed to a clear block if it is a isolated tampered block. Each watermark $w_{i,m}$ where $1 \leq m \leq sno$ that is not equal to $w_{i,vq}$ is treated as a damaged watermark. This watermark was embedded into the selected block numbered $BP_{m,i}$ according the *m*-th block permutation. The $BP_{m,i}$-th image block is then classified as a tampered block. Each damaged is sequentially process to find out the tampered blocks. Then, by combining these tampered blocks into the first-stage *BDI*, the resultant *BDI* is generated.

Continue the tampered example as shown in Figure 6(b), the first image block is classified as a tampered image block after the initial block detection process and the major voting process are sequentially executed. In this example, the $w_{1,1}$ and $w_{1,2}$ are treated as the damaged watermarks. Based on the two block permutations as shown in Figures. 4(b) and 4(c), we find that they are embedded in the image blocks numbered $BP_{1,1}$ and $BP_{2,1}$, respectively. Then, the 4-th and 15-th image blocks are classified as the tampered image blocks in the backward detection process.

### 3.3.3. The Generation of Tamper Detected Image

Recall that *FDI* and *BDI* are generated by using the forward detection mechanism and backward detection mechanism, respectively. To generate the tamper detected image (*TDI*), each image block *x* will be classified as either a clear block or a tampered block. Here, $type_{FDI}$ and $type_{BDI}$ denote the block types of *x* in *FDI* and *BDI*, respectively. Each image block *x* is classified as a clear block if both $type_{FDI}$ and $type_{BDI}$ are clear. Otherwise, *x* is classified as a tampered block.

The reason why we combine *FDI* and *BDI* to generate the tamper detected image is that some tampered blocks may be not correctly detected by using either the forward detection mechanism or the backward detection mechanism. It may happen that one image block is modified but the modified pixels in the block are quite similar to its original contents. Suppose the same closest codeword in the codebook of the original image block and the modified block is found in VQ. In other words, the watermark generated by VQ kept unchanged even if the block is modified. Only the last *eb*-bits of the pixels in the block are affected. This situation may not be correctly by using the forward detection mechanism. But it can be found by using the backward detection mechanism.

### 3.4. The Image Recovery Procedure

When a tampered block is to be recovered, these *sno* copies of the watermark that were embedded into other image blocks are checked. Suppose the *i*-th image block is tampered, the watermark $w_{i,vq}$ is assumed to be damaged. Recall that the *m*-th copy of the watermark of the image block was embedded into the image block numbered $BP_{m,i}$. All we have to do

is to find out an un-damaged watermark that will be used to recover the image block. If the watermark was stored in a clear block, we assume that it is un-damaged. Otherwise, we assume that it is damaged.

To find out a clear block, these *sno* block numbered $BP_{m,i}$ where $1 \leq m \leq sno$ are sequentially checked. Upon finding a clear block, the un-damaged watermark is used to recover the tampered block. Recall that the watermark of each image block is the index of the closest codeword in the codebook. By recovering the image block using the corresponding codeword in the codebook, the tampered block is rebuilt. By successively recovering each tampered block by the same way, the tampered image is now reconstructed.

## 4. Experimental Results

The simulations are executed on Microsoft windows XP with an Intel Core Duo 2.2GHz CPU and 512Mbytes RAM. The codes are implemented by using Bloodshed Dev C++. The eight grayscale images of 512×512 pixels "Airplane", "Boat", "Goldhill", "Toys", "Girl", "Lenna", "Peppers", and "Sailboat" were used in our experiments. The first four testing images as shown in Figure 7 are used as the training images for codebook design by the LBG algorithm. The last four testing images are shown in Figure 8. Each testing image is partitioned into a set of non-overlapping image block of 4×4 pixels. The terminated threshold used in the LBG algorithm is set to 0.001. Image qualities of the VQ-compressed image with different codebook sizes are listed in Table 3. It is shown that the image quality of the VQ-compressed image increase as the increases of the codebook size. Average image qualities of 26.179 dB, 30.679 dB, and 33.461 dB are obtained in VQ when the codebooks sizes are 16, 256, and 4096, respectively.

**Table 3. Image Qualities of the VQ-compressed Images With Different Codebook Sizes**

| Sizes / Images | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|---|---|---|
| Airplane | 25.983 | 27.800 | 29.315 | 30.648 | 31.450 | 32.307 | 33.061 | 33.831 | 34.683 |
| Boat | 25.371 | 26.779 | 27.832 | 29.084 | 29.941 | 30.722 | 31.508 | 32.238 | 33.118 |
| Goldhill | 26.277 | 27.692 | 28.892 | 29.780 | 30.587 | 31.379 | 32.295 | 33.090 | 33.998 |
| Toys | 25.772 | 27.449 | 29.014 | 30.428 | 31.604 | 32.471 | 33.481 | 34.296 | 35.244 |
| Girl | 27.414 | 28.412 | 29.413 | 30.429 | 31.245 | 31.960 | 32.661 | 33.283 | 33.726 |
| Lenna | 26.785 | 27.943 | 29.121 | 30.195 | 30.972 | 31.723 | 32.427 | 32.945 | 33.387 |
| Pepper | 26.783 | 28.245 | 29.476 | 30.289 | 31.145 | 31.783 | 32.390 | 32.893 | 33.252 |
| Sailboat | 25.048 | 26.173 | 27.016 | 27.827 | 28.487 | 28.986 | 29.550 | 29.898 | 30.281 |

(a) Airplane


(b) Boat


(c) Goldhill

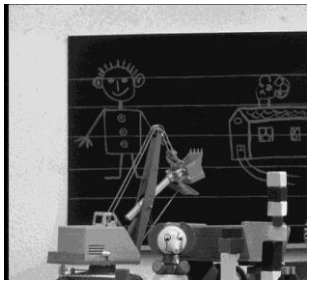
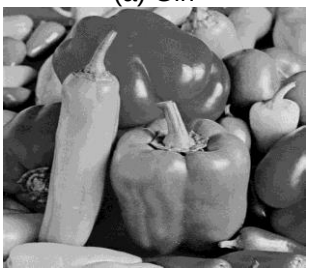

(d) Toys

**Figure 7. Training Set of Images for Codebook Design**

In the experiments, the codebook size $N$ is set to 256 in the proposed scheme. The index of the closest codeword in the VQ codebook for each 4×4 image block is stored in 8 bits. In other words, 8-bit watermark for each image block is needed. In the simulations, four *sno* values including 2, 4, 6, and 8 are set. The corresponding *eb* values are 1, 2, 3, and 4 when *sno* values are set to 2, 4, 6, and 8, respectively.


(a) Girl


(b) Lenna


(c) Pepper

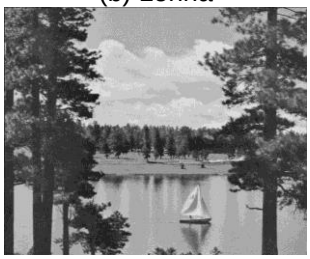

(d) Sailboat

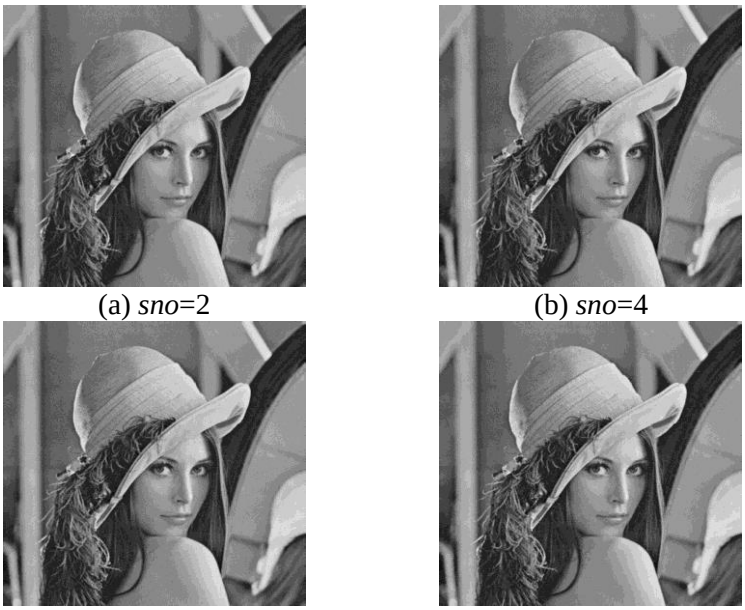
**Figure 8. Testing Images of 512×512 Pixels**

**Table 4. Image qualities of the Embedded Images of the Proposed Scheme with Different sno Values**

| Images \ *sno* | *sno*=2 | *sno*=4 | *sno*=6 | *sno*=8 |
|---|---|---|---|---|
| Airplane | 51.136 | 43.947 | 37.682 | 31.433 |
| Boat | 51.160 | 44.048 | 37.765 | 31.522 |
| Goldhill | 51.146 | 44.073 | 37.780 | 31.659 |
| Toys | 51.142 | 44.053 | 37.806 | 31.609 |
| Girl | 51.124 | 44.125 | 37.841 | 31.700 |
| Lenna | 51.140 | 44.126 | 37.866 | 31.706 |
| Pepper | 51.131 | 44.129 | 37.884 | 31.759 |
| Sailboat | 51.161 | 44.140 | 37.836 | 31.668 |

In Table 4, image qualities of the embedded images of the proposed scheme are listed. It is shown that the image quality of the embedded image decrease when more copies of the watermark are embedded. Average image qualities of 51.143 dB, 44.080 dB, 37.808 dB, and 31.632 dB of the embedded images are obtained in the proposed scheme when *sno* values are set to 2, 4, 6, and 8, respectively.

To understand the visual qualities of the embedded images of the proposed scheme are listed in Figure 9. In this example, the testing image "Lenna" is used. The visual qualities of the embedded images in Figures. 9(a) to 9(c) are quite good. It is hard to distinguish the different between each embedded image and the original image as shown in Figure 8(b). The visual quality of the embedded image in Figure 9(d) is poor. That may due to the last 4 bits of each pixel are used embedded the watermark data in it.



(a) *sno*=2 (b) *sno*=4

(c) *sno*=6 (d) *sno*=8

**Figure 9. Embedded Images of the Testing Image "Lenna" by using the Proposed Scheme with Different sno Values**

We added a flower object in the embedded image "Lenna" at the shoulder, as shown in Figure 10. In this tamper test, some modified pixels in the embedded object are quite similar to the original pixels in the embedded image. By doing so, some image blocks in the embedded image may encode by the same codeword in the codebook as that of the original blocks. In other words, the indices of some image block may keep unchanged even when the object had been added. Results of the difference between the tampered image and the embedded image of the proposed scheme with different *sno* values are listed in Table 5.

(a) *sno*=2 (b) *sno*=4

(c) *sno*=6

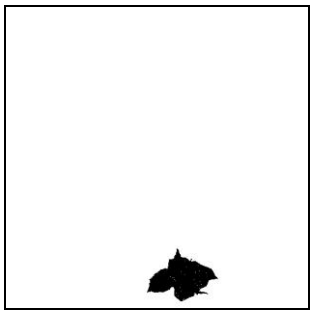
(d) *sno*=8

**Figure 10. The Tampered Image using the Embedded Images as shown in Figure 9**

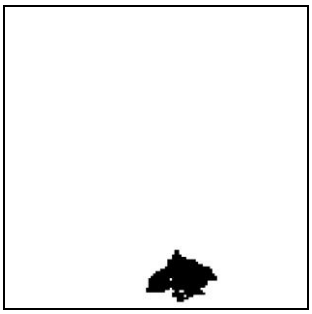
**Table 5. Total Number of the Differences between each Embedded Image and its Tampered Image in Terms of the Image Pixel, VQ index, and Image Block**

| Factors / *sno* | Pixel difference | Index difference | Block difference |
|---|---|---|---|
| *sno* = 2 | 5186 | 345 | 371 |
| *sno* = 4 | 5184 | 345 | 371 |
| *sno* = 6 | 5185 | 349 | 371 |
| *sno* = 8 | 5178 | 347 | 371 |

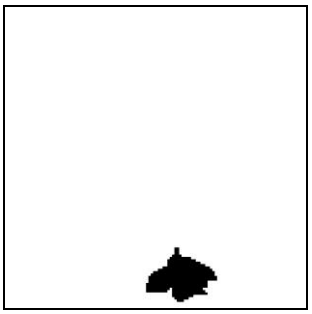
Three tpypes of difference are listed. Pixel difference records the total number of pixel differs in the embedded image and the tampered image. Index difference records the total numbers of index for each VQ-compressed block of 4×4 pixels in the embedded image and the tampered image. Finally, block difference records the total number of the blocks differs in the embedded image and the tampered image. If the number of different pixels is greater than or equal to 1, this block is classified as a different block. From Table 5, it is shown that 5186, 5184, 5185, and 5178 different pixels are found when *sno* values are set to 2, 4, 6, and 8, respectively. In addition, 345, 345, 349, and 347 different indices are found when *sno* values are set to 2, 4, 6, and 8, respectively. That indicates that the total number of the embedded blocks having the same index with the tampered blocks is not the same for different *sno* values.



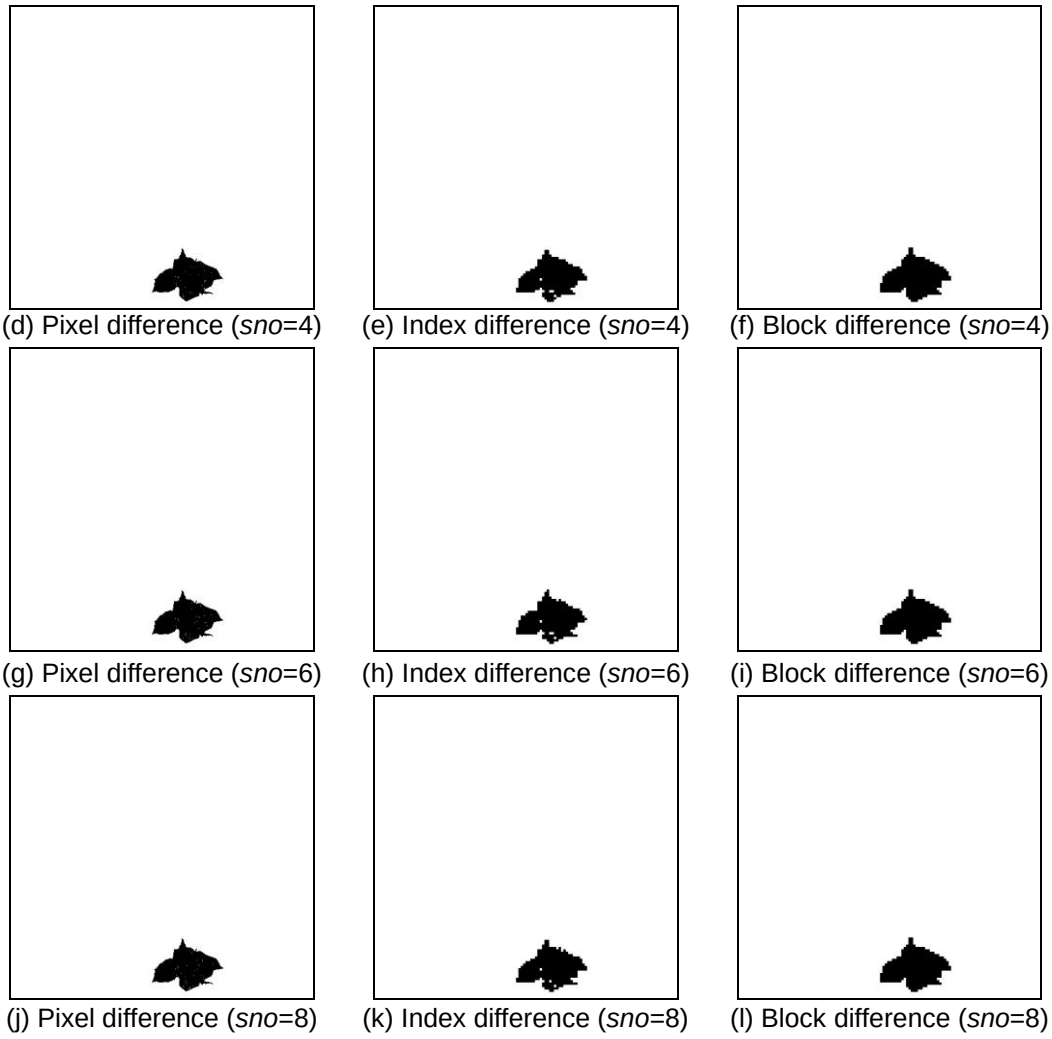
(a) Pixel difference (*sno*=2)



(b) Index difference (*sno*=2)



(c) Block difference (*sno*=2)



(d) Pixel difference (*sno*=4) (e) Index difference (*sno*=4) (f) Block difference (*sno*=4)

(g) Pixel difference (*sno*=6) (h) Index difference (*sno*=6) (i) Block difference (*sno*=6)

(j) Pixel difference (*sno*=8) (k) Index difference (*sno*=8) (l) Block difference (*sno*=8)

**Figure 11. Difference Images between each Embedded Image and its Tampered Image with Different sno Values**

To understand the three different types of comparison, results of the difference images between each embedded image and its tampered image with different *sno* values are shown in Figure 11. From the results as shown in Figures. 11(b), 11(e), 11(h), 11(k), it is shown that some white spots are found in the tamper objected. Each white spot represents one image block that having the same index in its corresponding embedded image and the tamper image. In addition, it is shown that the four block difference images as shown in Figures 11(c), 11(f), 11(i), 11(l) are the same.

In the tamper detection procedure, the forward detection mechanism and the backward detection mechanism are sequentially executed to find out the detected image. The total numbers of tampered blocks found at different detecting steps are shown in Table 6. It is shown that the number of the tampered blocks found in the initial block detection process increases as the increment of the *sno* value. By applying the majority voting process, the false detection problem significantly improved. After the forward detection mechanism is executed, the numbers of the tampered blocks equal 348, 349, 344, and 345 when the *sno* values are set to 2, 4, 6, and 8, respectively. In addition, 362, 368, 367, and 371 tamper blocks are found by performing the backward detection mechanism when the *sno* values are set to 2, 4, 6, and 8, respectively. By combing the tampered blocks found in either the forwards detection mechanism or the backward detection mechanism, the two-way detection results are obtained.

**Table 6. Total Numbers of the Tampered Blocks at Different Detection Stage of the Proposed Scheme**

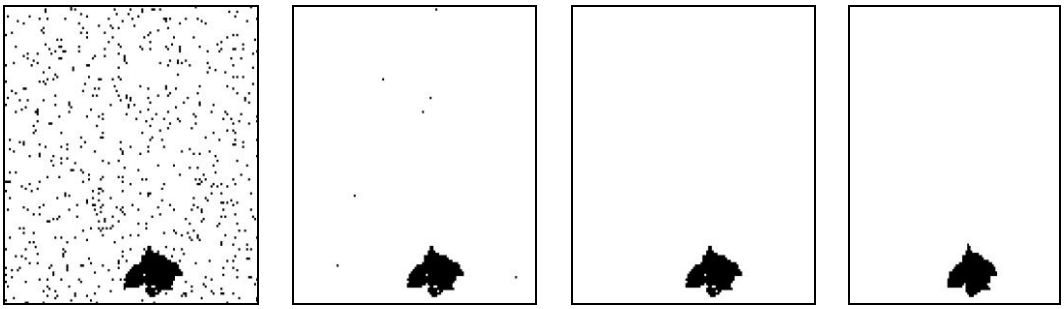
| stage / *eb* | Initial block detection | Majority voting | Isolate block Removal | Backward detection | Two-way detection |
|---|---|---|---|---|---|
| *sno* = 2 | 1017 | 355 | 348 | 362 | 366 |
| *sno* = 4 | 1663 | 349 | 349 | 368 | 368 |
| *sno* = 6 | 2255 | 344 | 344 | 367 | 367 |
| *sno* = 8 | 2838 | 345 | 345 | 371 | 371 |

The detected images by the proposed tamper detection procedure are shown in Figure 12. From the results, it is shown more tampered blocks are found in the initial block detection process when a large *sno* value is used. By applying the majority voting process, great deals of the false detected blocks are removed. But, some isolated tampered blocks can be found in the detected images after the major voting process is executed. In addition, some white spots can be found in the detected images as shown in Figures 12(c), 12(f), 12(j), and 12(n) after the forward detection mechanism is executed. This problem can be resolved by executing the backward detection mechanism as shown in Figures 12(d), 12(g), 12(k), and 12(o).

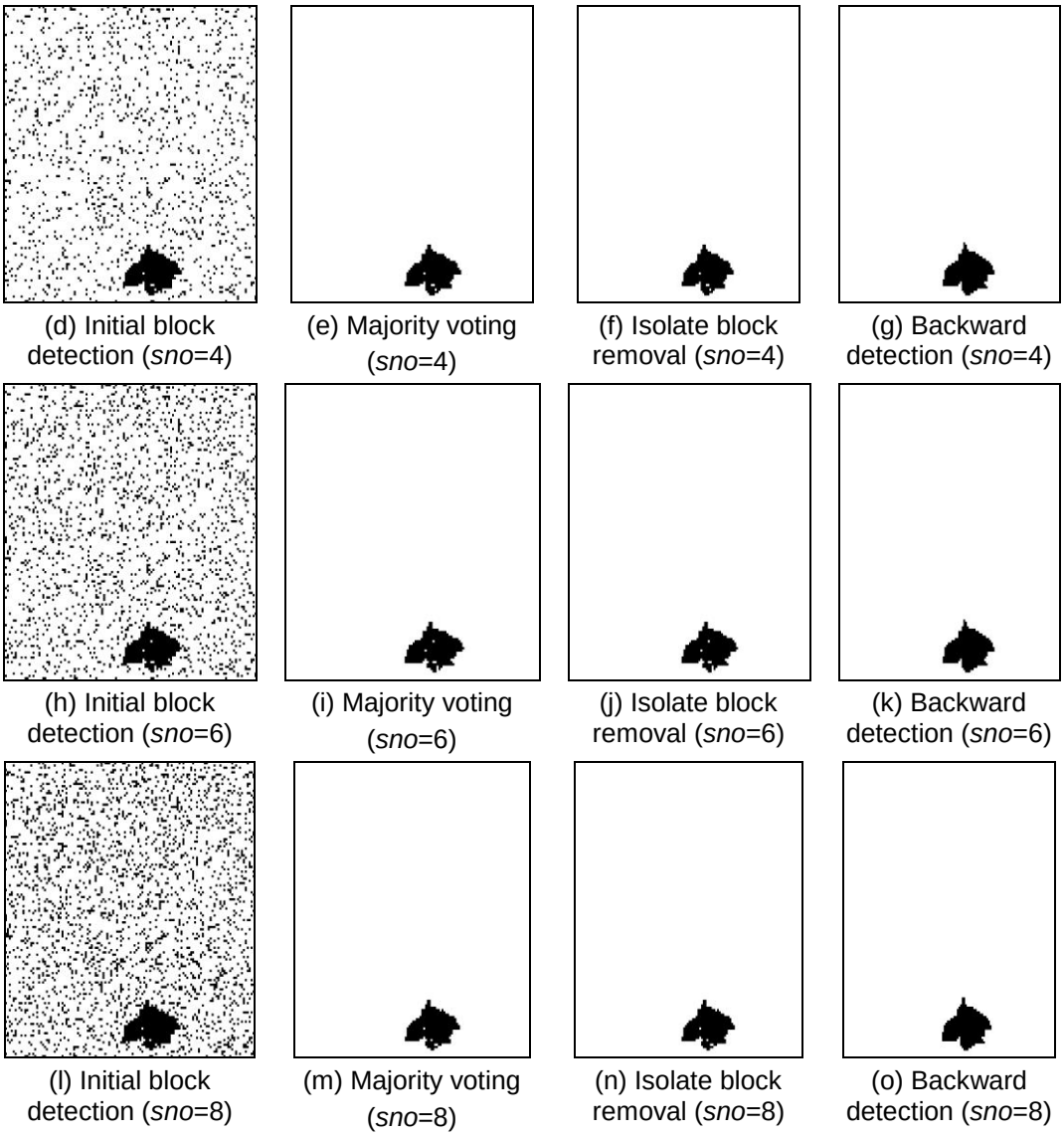
To understand the performance of the proposed image recovery procedure, the recovered images of the proposed scheme with different *sno* values are shown in Figure 13. From the results, it is shown that the tampered objected are well recovered by using the proposed image recovery procedure. It is hard to distinguish the difference between each recovered image and its corresponding embedded image as shown in Fig. 9 from the Human Visual system.

## 5. Conclusions

In this paper, we proposed a novel image authentication with tamper recovery scheme by using the vector indexing. The proposed method not only locate the alterations but also recover the tampering regions. The watermarks were generated from the host image by the VQ encoding and multi-copies of watermarks were embedded into the last *eb* bits of image pixels of the selected blocks. The embedding sequence of image blocksb for watermarks embedding is by the pseudo random number generator with some known seeds. To verify and recover the tampered image, the forward detection mechanism and the backward detecton mechanism are proposed which can effective improve the detection accuracy. From to the experimental results show that, the tampered places were detected and well recovered.



(a) Initial block detection (*sno*=2) (b) Majority voting (*sno*=2) (c) Isolate block removal (*sno*=2) (d) Backward detection (*sno*=2)



(d) Initial block detection (*sno*=4) (e) Majority voting (*sno*=4) (f) Isolate block removal (*sno*=4) (g) Backward detection (*sno*=4)

(h) Initial block detection (*sno*=6) (i) Majority voting (*sno*=6) (j) Isolate block removal (*sno*=6) (k) Backward detection (*sno*=6)

(l) Initial block detection (*sno*=8) (m) Majority voting (*sno*=8) (n) Isolate block removal (*sno*=8) (o) Backward detection (*sno*=8)

**Figure 12. Detected Images of the Proposed Tamper Detection Procedure**

(a) *sno*=2

(b) *sno*=4

(c) *sno*=6

(d) *sno*=8

**Figure 13. Recovered Images of the Proposed Scheme**

## Acknowledgements

This research was supported by Providence University, Taichung, Taiwan under contract the National Science Council, Taipei, R.O.C. under contract NSC 101-2221-E-126-014.

## References

[1] R. L. Riverst, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," ACM Truncation on communications, vol. 21, no. 2, **(1978),** pp. 120-126.
[2] F. Bartolini, A. Tefas, M. Bami and I. Pitas, "Image authentication techniques for surveillance applications," Proceedings of IEEE, vol. 89, no. 10, **(2001)**, pp. 1403-1418.
[3] C. W. Wu, "On the design of content-based multimedia authentication systems," IEEE Transactions on Multimedia, vol. 4, no. 3, **(2002)**, pp. 385-393.
[4] M. U. Celik, G. Sharma and A. Tekalp, "Lossless watermarking for image authentication: a new framework and an implementation," IEEE Transactions on Image Processing, vol. 15, no. 4, **(2006)**, pp. 1042-1049.
[5] C. C. Chang, Y. S. Hu and T. C. Lu, "A watermarking-based image ownership and tampering authentication scheme," Pattern Recognition Letters, vol. 27, no. 5, **(2006)**, pp. 439–446.
[6] T. Y. Lee and S. D. Lin, "Dual watermark for image tamper detection and recovery," Pattern Recognition, vol. 41, no. 11, **(2008)**, pp. 3497-3506.
[7] R. C. W. Phan, "Tampering with a watermarking-based image authentication," Pattern Recognition, vol. 41, no. 11, **(2008)**, pp. 2493-3496.
[8] C. W. Yang and J. J. Shen, "Recover the tampered image based on VQ indexing," Signal Processing, vol. 90, no. 1, **(2010)**, pp. 331–343.
[9] J. C. Chuang and Y. C. Hu, "An adaptive image authentication scheme for vector quantization compressed image," Journal of Visual Communication and Image Representation, vol. 22, no. 5, **(2011)**, pp. 440-449.
[10] D. Kundur and D. Hatzinakos, "Digital watermarking for telltale tamper proofing and authentication," Proceedings of the IEEE, vol. 87, no. 7, **(1999)**, pp. 1167–1180.
[11] C. S. Lu and H. Y. M. Laio, "Multipurpose watermarking for image authentication and protection," IEEE Transactions on Image Processing, vol. 10, no. 10, **(2001)**, pp. 435-439.
[12] C. Y. Lin and S. F. Chang, "A robust image authentication method distinguish JPEG compression from malicious manipulation," IEEE Transactions on Circuits and Systems of Video Technology, vol. 11, no. 2, **(2001)**, pp. 153-168.
[13] W. N. Lie, G. S. Lin and S. L. Chen, "Dual protection of JPEG images based on informed embedding and two-stage watermark extraction techniques," IEEE Transactions on Information Forensics and Security, vol. 1, no. 3, **(2006)**, pp. 330-341.
[14] S. Jun and M.S. Alam, "Fragility and robustness of binary-phase-only-filter-based fragile/semi-fragile digital image watermarking," IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 3, **(2008)**, pp. 595-606.
[15] J. C. Patra, J. E. Phua and C. Bornand, "A novel DCT domain CRT-based watermarking scheme for image authentication surviving JPEG compression," Digital Signal Processing, vol. 20, no. 6, **(2010)**, pp. 1597–1611**.**
[16] X. Qi and X. Xin, "A quantization-based semi-fragile watermarking scheme for image content authentication," Journal of Visual Communication and Image Representation, vol. 22, no. 2, **(2011)**, pp. 187–200.
[17] Y. C. Hu, C. C. Lo, W. L. Chen and C. H. Wen, "Joint image coding and image authentication based on AMBTC," Journal of Electronic Imaging, vol. 22, no. 1, **(2013)**, 013012 (1-12).

[18] Y. C. Hu, W. L. Chen, C. C. Lo and C. M. Wu, "A novel tamper detection scheme for BTC compressed images," Opto-Electronics Review, vol. 21, no. 1, **(2013)**, pp. 137-146.
[19] Y. C. Hu, C. C. Lo, C. M. Wu, W. L. Chen and C. H. Wen, "Probability-based tamper detection scheme for BTC-compressed Images based on quantization levels modification," International Journal of Security and Its Applications, vol. 7, no. 3, **(2013)**, pp. 11-32.
[20] Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector quantizer design," IEEE Transactions on Communications, vol. 28, **(1980)**, pp. 84-95.
[21] C. C. Chang and Y. C. Hu, "A fact codebook search algorithm for vector quantization," IEEE Transactions on Consumer Electronics, vol. 44, no. 4, **(1998)**, pp. 1201–1208.
[22] Y. C. Hu and C. C. Chang, "An effective codebook search algorithm for vector quantization," Imaging Science Journal, vol. 51, no. 4, **(2003)**, pp. 221-234.
[23] P. Y. Tsai, C. C. Chang, M. H. Lin and Y. C. Hu, "An adaptive bit rate encoding mechanism based on irregular sampling and side-match vector quantization', Imaging Science Journal, vol. 52, no. 2, (**2004)**, pp. 73-84.
[24] Y. C. Hu, B. H. Su and C.C. Tsou, "Fast VQ codebook search algorithm for grayscale image coding," Image and Vision Computing, vol. 26, no. 5, **(2008)**, pp. 657-666.
[25] Y. C. Hu, J. C. Chuang, C. C. Lo and C. Y. Lee, "Effect grayscale image compression technique based on VQ," Opto-Electronics Review, vol. 19, no. 1, (**2011)**, pp. 104-113.
[26] Y. C. Hu, W. L. Chen, C. C. Lo and J. C. Chung, "An improved vector quantization scheme for grayscale image compression," Opto-Electronics Review, vol. 20, no. 2, (**2012**), pp. 187-193
[27] Y. C. Hu, C. H. Wen, C. C. Lo and W. L. Chen, "Image vector quantization using geometric transform and lossless index coding," Optical Engineering, vol. 52, no. 3, (**2013**), 037402
[28] Y. C. Hu, W. L. Chen, C. C. Lo, C. M. Wu and C. H. Wen, "Efficient VQ-based image coding scheme using inverse function and lossless index coding," Signal Processing, vol. 93, no. 9, (**2013**), pp. 2432-2439.

## Authors

**Jun-Chou Chuang** received his PhD. degree in computer science and information engineering from the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan in 2004. Currently, Dr. Chuang is an assistant professor in the Department of Computer Science and Communication Engineering, Providence University, Sha-Lu, Taiwan. His research interests include multimedia security, data hiding, digital watermarking and signal processing.

**Yu-Chen Hu** received his PhD. degree in computer science and information engineering from the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan in 1999. Currently, Dr. Hu is a professor in the Department of Computer Science and Information Management, Providence University, Sha-Lu, Taiwan. He is a member of ACM and IEEE. Dr. Hu Servers as the Editor-in-Chief of International Journal of Image Processing since 2009. He joints the editorial boards of several other journals including International journal of Security and Its Applications, International Journal of Signal Processing, Image Processing and Pattern Recognition, International Journal of Digital Contents and Applications and so on. His research interests include image and signal processing, data compression, information hiding, and data engineering.

**Chun-Chi Lo** received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1989, 1991, and 1996, respectively. In 2004, he joined the faculty of the Department of Computer Science and Information Engineering, Providence University, Taiwan. Currently, he is an Assistant Professor. His research interests include wireless sensor networks, mean-field annealing, and combinatorial optimizations.

**Wu-Lin Chen** is currently an associate professor in the department of Computer Science and Information Management at Providence University. He received his M.S. and Ph.D. degrees in the School of Industrial Engineering at Purdue University in 1995, and 1999, respectively. His research interests include operations research, production management, and stochastic models.