A Blind Data Hiding Technique with Error Correction Abilities and a High Embedding Payload

H. Y. Liang[1], C. H. Cheng[*2], C. Y. Yang[3], K. F. Zhang[4]

[1,4] Department of Information and Communication Engineering
Chaoyang University of Technology
Taichung, 4139, Taiwan
[2] Department of Electrical Engineering
National Formosa University
Yunlin, 632, Taiwan
* chcheng@nfu.edu.tw
[3] Department of Computer Science
Taipei Municipal University of Education
Taipei, 10042, Taiwan

ABSTRACT
Least significant bit (LSB) embedding is a simple steganographic technique used in the spatial domain that involves replacing the LSB of each pixel in the cover work with the information bits being hidden to achieve information secrecy. However, considering the current prevalence of information theft and sabotage, the stego image produced using LSB embedding cannot prevent malicious illegal acts. Therefore, this study proposes a blind data hiding technique that combines block-coded modulation (BCM) codes with LSB embedding. The proposed method not only provides error correction capabilities to prevent errors during the extraction process, but also has a superior embedding payload. The peak signal noise ratio (PSNR) exceeds 40 dB by using our proposed method. Regarding the simulation results, the proposed method provides a superior PSNR compared to LSB embedding and a higher embedding payload than that offered by other steganographic techniques with error correction capabilities.

Keywords: Least significant bit embedding, data hiding, block-coded modulation codes, Reed-Muller codes.

1. Introduction

Watermarking [1, 2, 3, 4, 5] and steganography [6, 7, 8, 9] are the two main information hiding techniques that are used to protect intellectual property by embedding copyright information into digital media such as images, music, and videos. To explain watermarking, consider invisible watermarking as an example; invisible watermarking is employed to embed visually imperceptible data or ciphers into cover work. Steganography involves embedding information (cipher) in cover work through undetectable changes. A major characteristic of blind data hiding watermarking techniques is that the extraction processing can extract the hidden information without cover work. The extraction process that does require cover work is called informed data hiding. To evaluate the efficiency of steganographic techniques, Chang [10] recommended two criteria to evaluate steganographic techniques, namely, embedding efficiency and embedding payload. Embedding efficiency describes the visual quality of the stego image, and the embedding payload refers to the size of the embedded cipher. Generally, embedding efficiency uses a peak signal noise ratio (PSNR) to evaluate the visual quality. When the PSNR value exceeds 30 dB, the human eye cannot detect visual differences between embedded and nonembedded images. Thus, the optimal information hiding technique should have both a high embedding payload and high embedding efficiency. Considering the current prevalence of information theft and sabotage, producing stego images within acceptable bounds of the two criteria while effectively preventing malicious illegal acts has become a popular research objective.

Chang [10] proposed a novel steganographic technique with error correction capability. The proposed method used Hamming codes [7, 4] and LSB embedding to encode the hidden information into codewords and embed the codewords into pixels of the cover work, respectively. The simulation results indicated that the proposed method provided superior PSNR compared to LSB embedding. Briefly, the proposed method first divided 7 secret bits into 3 bits and 4 bits, which were then used to locate the selected codeword in each row and column of the 8×16 standard array. Next, each selected codeword was embedded into every 7 pixels of the cover work using LSB embedding until all secret bits were embedded. The extraction process involves extracting the LSB of every 7 pixels of the stego image and locating the codeword in every row and column of the 8×16 standard array. Every 7 secret bits extracted can be identified by converting the row number and column number into 3 bits and 4 bits. The method proposed by Chang requires greater memory because the proposed method employs a lookup table for embedding and extracting secret bits. Moreover, the error correction capability of this method can correct only single-bit errors. Therefore, how to further improve the error correction capability and embedding payload of the method proposed by Chang is a popular research topic.

This study combines block coded modulation (BCM) codes with LSB embedding to improve both the error correction capability and embedding payload of Chang's proposed method. To further improve the PSNR of the proposed method, this study employs modulus functions [11] to increase the differences between the cover work and stego image before embedding BCM codewords into the cover work. The simulation results show that the method proposed in this study possess superior error correction and LSB embedding capabilities than that of the method proposed by Chang when the stego image is corrupted by salt-and-pepper noise. Furthermore, our proposed method also offers a higher embedding payload than that provided by Chang's proposed method.

2. Literature Review

In this section, we first describe the encoding and decoding architecture of BCM codes and Reed-Muller codes. Then, we explicate the embedding and extraction processes of the following three techniques: LSB embedding, Chang's proposed method [10], and the modulus function.

2.1 BCM Codes

BCM [12] is a coding scheme that combines both modulation and channel coding and is based on the Euclidean distance between symbols. For convenience, we describe how to construct a 4-QAM BCM code based on a 4-QAM signal constellation with set partitioning. First, symbols within the signal constellations are recursively divided into numerous subsets, where the Euclidean distance between the neighboring symbols of each subset are gradually increased. A 4-QAM BCM code is composed of two component codes C_1 and C_2, which are binary block codes with length *N*. The minimum Hamming distance is d_i, where $1 \le i \le 2$. The two component codes are used to construct a $2 \times N$ codeword array by inserting the codeword $\mathbf{c}_i$ of the i^{th} component code C_i into the i^{th} row of a $2 \times N$ codeword array. This is represented in the following mathematical expression:

$$\begin{pmatrix} \mathbf{c}_1 \in C_1 \\ \mathbf{c}_2 \in C_2 \end{pmatrix} = \begin{pmatrix} c_{1,0} & c_{1,1} & \cdots & c_{1,N-1} \\ c_{2,0} & c_{2,1} & \cdots & c_{2,N-1} \end{pmatrix}$$

Next, a 4-QAM BCM codeword with length *N* is created by modulating each column of the codeword array into a 4-QAM symbol. Figure 1 shows a 4-QAM signal constellation, where the bit indicators are arranged by set partitioning and the ratio between the squared Euclidean distances of the two subsets is 1:2. A 4-QAM BCM codeword $\mathbf{X} = (X_0, X_1, \cdots, X_{N-1})$ can be expressed using the following mathematical equation:

$$X_k = \left(\frac{1}{\sqrt{2}} j^{c_{1,k} + 2c_{2,k}} \right) e^{j\frac{\pi}{4}},\ 0 \le k \le N-1,$$

where $j = \sqrt{-1}$. This mathematical expression shows that the 4-QAM BCM codeword X is composed of two binary component codewords $\mathbf{c}_i \in C_i,\ i = 1,2$.

2.2 Reed-Muller codes

This study employs Reed-Muller codes as the component codes for constructing a BCM code.

Let *m* and *r* be the two positive integers and $0 \le r \le m$. A *r*th-order binary Reed-Muller code with a code length of $N = 2^m$, which is a

$$\left[N = 2^m, k = \sum_{i=0}^{r} \binom{m}{i}, d = 2^{m-r} \right]$$

linear block code, can be expressed as RM(*r*,*m*). Assuming $1 \le i \le m$, let $\mathbf{x}_i$ be a vector of length 2^m that exists in the binary Galois field with the following characteristics:

$$\mathbf{x}_i = (\underbrace{0,\cdots,0}_{2^{i-1}},\underbrace{1,\cdots,1}_{2^{i-1}},\underbrace{0,\cdots,0}_{2^{i-1}},\cdots,\underbrace{1,\cdots,1}_{2^{i-1}})$$

where $(\underbrace{0,\cdots,0}_{2^{i-1}})$ and $(\underbrace{1,\cdots,1}_{2^{i-1}})$ are 2^{i-1}-tuples with all zeroes and all ones as the content, respectively [12]. For example, if *m* = 3, the three vectors existing in the binary Galois field of length 8 can be expressed as

$$\mathbf{x}_1 = (0,1,0,1,0,1,0,1),$$
$$\mathbf{x}_2 = (0,0,1,1,0,0,1,1),$$
$$\mathbf{x}_3 = (0,0,0,0,1,1,1,1).$$

2.2.1 Encoding Reed-Muller codes

For an *r*th-order binary Reed-Muller code with a code length of 2^m, the generator matrix G of RM(*r*,*m*) can be expressed as follows:

$$\mathbf{G} = \begin{pmatrix} \mathbf{1} \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \\ \mathbf{x}_1\mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m\mathbf{x}_{m-1} \\ \vdots \\ \mathbf{x}_1\mathbf{x}_2\cdots\mathbf{x}_r \end{pmatrix},$$

where 1 is a vector of length 2^m and the content is all ones. In other words, an *r*th-order binary Reed-Muller codeword is obtained through a linear combination of the vectors

$$\{1, \mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{x}_1\mathbf{x}_2, ..., \mathbf{x}_m\mathbf{x}_{m-1}, ..., \mathbf{x}_1\mathbf{x}_2\cdots\mathbf{x}_r\}.$$

For example, *m* = 3 and *r* = 2, the generator matrix of RM (2,3) is a matrix sized $k \times n = 7 \times 8$, as shown below.

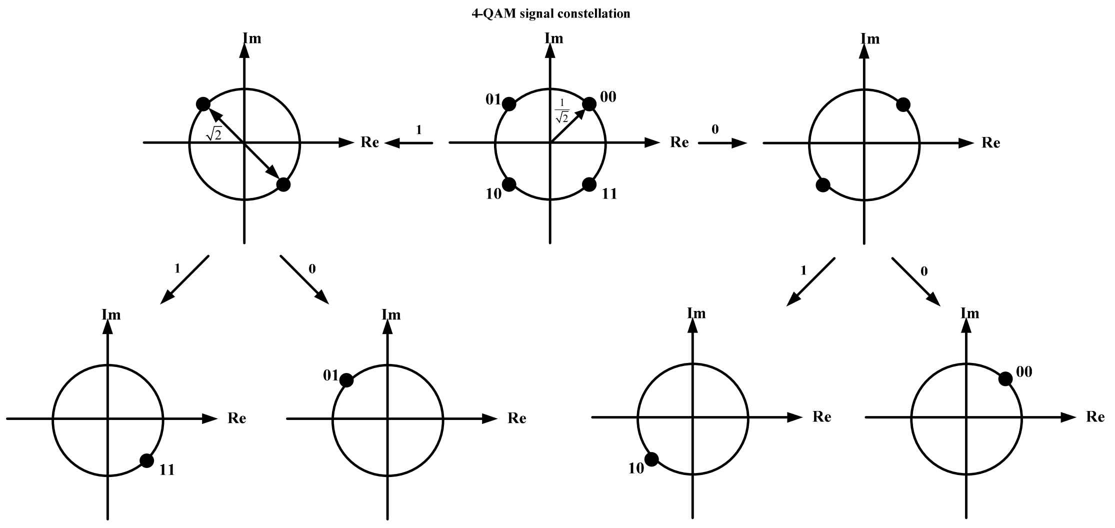

Figure 1. The set-partitioned 4-QAM signal constellation.

$$\mathbf{G} = \begin{pmatrix} 1 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_1\mathbf{x}_2 \\ \mathbf{x}_1\mathbf{x}_3 \\ \mathbf{x}_2\mathbf{x}_3 \end{pmatrix} = \begin{pmatrix} 1,1,1,1,1,1,1,1 \\ 0,1,0,1,0,1,0,1 \\ 0,0,1,1,0,0,1,1 \\ 0,0,0,0,1,1,1,1 \\ 0,0,0,1,0,0,0,1 \\ 0,0,0,0,0,1,0,1 \\ 0,0,0,0,0,0,1,1 \end{pmatrix}$$

Assuming the input message is $\mathbf{u} = (1,0,1,0,0,0,0)$, the output codeword c is expressed as follows:

$$\mathbf{c} = \mathbf{uG} = (1,0,1,0,0,0,0) \begin{pmatrix} 1,1,1,1,1,1,1,1 \\ 0,1,0,1,0,1,0,1 \\ 0,0,1,1,0,0,1,1 \\ 0,0,0,0,1,1,1,1 \\ 0,0,0,1,0,0,0,1 \\ 0,0,0,0,0,1,0,1 \\ 0,0,0,0,0,0,1,1 \end{pmatrix}$$

= (1,1,0,0,1,1,0,0)

2.2.2 Decoding Reed-Muller codes

The Reed decoding algorithm is a type of decoding algorithm for RM(*r*,*m*) [12]. Using RM(2,3) as an example, assuming the input message is $\mathbf{u} = (u_0, u_1, u_2, u_3, u_{12}, u_{13}, u_{23})$ and the received codeword is $\mathbf{r} = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8)$, the decoded value u'_{12} of input message bit u_{12} is determined by the results of the two decision equations below.

$$u_{12}^{(1)} = r_5 + r_6 + r_7 + r_8$$
$$u_{12}^{(2)} = r_1 + r_2 + r_3 + r_4$$

If both $u_{12}^{(1)}$ and $u_{12}^{(2)}$ are 1, then the input message bit u_{12} is decoded as u'_{12}=1. Conversely, if both $u_{12}^{(1)}$ and $u_{12}^{(2)}$ are zeroes, then the decoded input message bit u'_{12} is zero. Calculation of the above equations is known as binary addition; that is, 1+1=0, 1+0=1, 0+1=1, and 0+0=0. The corresponding positions in the received codeword of two decision equations are based on the non-zero positions in the two vectors $\{\mathbf{x}_3, 1+\mathbf{x}_3\}$. The decision equations for decoding input messages u_{13} and u_{23} are determined by summing the non-zero positions from $\{\mathbf{x}_2, 1+\mathbf{x}_2\}$ and $\{\mathbf{x}_1, 1+\mathbf{x}_1\}$ of the received codeword for decision equations, as shown below.

$$u_{13}^{(1)} = r_3 + r_4 + r_7 + r_8$$
$$u_{13}^{(2)} = r_1 + r_2 + r_5 + r_6$$
$$u_{23}^{(1)} = r_2 + r_4 + r_6 + r_8$$
$$u_{23}^{(2)} = r_1 + r_3 + r_5 + r_7$$

The first stage correction involves adjusting the received codeword according to the decoded $(u'_{12}, u'_{13}, u'_{23})$. After this adjustment, the received codeword of the first stage correction is defined as $\mathbf{r}' = \mathbf{r} - u'_{12}\,\mathbf{x}_1\mathbf{x}_2 - u'_{13}\,\mathbf{x}_1\mathbf{x}_3 - u'_{23}\,\mathbf{x}_2\mathbf{x}_3$. The input m-essage (u_1, u_2, u_3) is then decoded from the con-tent of $\mathbf{r}'$, where the input message bit u_1 may be decoded to determine the decoded value u'_1 using the results of the following four decision equations.

$$u_1^{(1)} = r'_7 + r'_8$$
$$u_1^{(2)} = r'_5 + r'_6$$
$$u_1^{(3)} = r'_3 + r'_4$$
$$u_1^{(4)} = r'_1 + r'_2$$

If most of the results of these four decision equations are zeroes, the input message u_1 is decoded as u'_1= 0. Conversely, if most of the resul-ts of these four decision equations is 1, then the input message u_1 is decoded as u'_1= 1. The four decision equations were decoded by summing the corresponding non-zero positions of the codewor-ds received from the four vectors

$$\{\mathbf{x}_2\mathbf{x}_3, (1+\mathbf{x}_2)\mathbf{x}_3, \mathbf{x}_2(1+\mathbf{x}_3), (1+\mathbf{x}_2)(1+\mathbf{x}_3)\}.$$

Therefore, the decision equations for decoding the input messages u_2 and u_3 are based respectively on the sum of the corresponding non-zero positio-ns from the received codewords in

$$\{\mathbf{x}_1\mathbf{x}_3, (1+\mathbf{x}_1)\mathbf{x}_3, \mathbf{x}_1(1+\mathbf{x}_3), (1+\mathbf{x}_1)(1+\mathbf{x}_3)\}$$

and

$$\{\mathbf{x}_1\mathbf{x}_2,(1+\mathbf{x}_1)\mathbf{x}_2,\mathbf{x}_1(1+\mathbf{x}_2),(1+\mathbf{x}_1)(1+\mathbf{x}_2)\}.$$

The decoded (u_1',u_2',u_3') are then used in the second stage correction. The received codeword of the second stage correction is defined as

$$\mathbf{r}'' = \mathbf{r}' - u'_1\,\mathbf{x}_1 - u'_2\,\mathbf{x}_2 - u'_3\,\mathbf{x}_3\ ,$$

which is used to determine the decoded value of the input message bit u_0. If the majority of the content value for $\mathbf{r}''$ is 0, then the input message bit u_0 is decoded as u_0' = 0. Conversely, u_0 is decoded as u_0' = 1 if the majority of the content value for $\mathbf{r}''$ is 1. Finally, the input message decoded using the Reed decoding algorithm is

$$\mathbf{u}' = (u_0',u_1',u_2',u_3',u_{12}',u_{13}',u_{23}')\ .$$

2.3 Chang's proposed method [10]

For a cover work sized $\mathrm{M}\times W$, the method proposed by Chang employs the Hamming code [7,4] to hide information. The amount of data that can be hidden is $\lfloor(\mathrm{M}\times W)/7\rfloor$, where $\lfloor y\rfloor$ denotes the smallest positive integer that is less than *y*. Chang's proposed method hides information using the Hamming code [7,4], with a standard array of 8×16. Seven bits are read from the data to be hidden and then divided into 3 bits and 4 bits. The 3 bits are used to express which of the 8 coset leaders should be selected, and the 4 bits denote which of the 16 codewords should be chosen. Chang's proposed method adds the selected coset leader and codeword using the vector from additions as the information to embed into the cover work. The method to embed information involves obtaining 7 pixels from the cover work and then replacing the LSB content of the 7 pixels with the selected processed information in sequence.

Data is extracted from the receiver by reading the LSB content of the 7 pixels in the sequence that data are embedded into the cover work. The 7 bits obtained using this method are multiplied by the parity check matrix H of the Hamming code [7,4] to obtain the 3 bits syndrome. The coset leader corresponding to the syndrome is added to the extracted 7 bits to obtain the codeword, and the 4 bits corresponding to the codeword are revealed using the standard array, thereby retrieving the embedded 7 bits of hidden data. Because Chang's proposed method requires the standard array to be built beforehand, extra memory is required in both the transmitter and the receiver to store the 128 vectors required for embedding and decoding. If the code length *N* for the Hamming code is extended, the vectors requiring storage increase to the *n*th power of 2, that is, 2^n, which increases the overall costs. Thus, we propose a steganographic technique based on BCM architecture and select Reed-Muller codes as component codes. This is primarily because the Reed decoding algorithm can be decoded through easy calculations and does not require substantial memory to build the standard array, thereby effectively improving the high storage requirements of Chang's proposed method.

2.4 Modulus function

Thien and Lin [11] proposed a steganographic technique to improve the visual quality of the stego image using modulus functions. The expression of modulus function is defined as *c* = *a* mod *b*, where *a* is the dividend, *b* is the divisor, and *c* is the reminder. For example, 3 mod 2 equals 1, that is, the division of 3 by 2 leaves a remainder of 1. Let *x* be a pixel used to hide data in the cover work, *z* be the hidden information, and *y* equal $\log_2 z$. First, the embedding process of modulus function is employed to compute the following equation:

$$w=z-(x \bmod y).$$

The result *w* is applied to determine the minimum variance *e* using the following equations:

$$e=\begin{cases} w,\ \text{if}\ \left(-\left\lfloor\dfrac{y-1}{2}\right\rfloor\right)\le w\le\left\lceil\dfrac{y-1}{2}\right\rceil \\ w+y,\ \text{if}\ (-y+1)\le w\le\left(-\left\lfloor\dfrac{y-1}{2}\right\rfloor\right) \\ w-y,\ \text{if}\ \left(\left\lceil\dfrac{y-1}{2}\right\rceil\right)\le w\le(y-1) \end{cases}$$

where $\lfloor s \rfloor$ represents the smallest integer closest to *s*, and $\lceil v \rceil$ represents the largest integer closest to *v*. Finally, the modified pixel x', which replaces x in the cover work, is obtained by calculating $x' = x + e$. The modulus function extraction process involves computing $z = x' \bmod y$ to extract the information hidden in the stego image. Although the modulus function offers superior visual quality, it has no error correction capabilities to prevent errors in the extraction process. Therefore, this study uses BCM codes to improve the error correction capabilities of the modulus function.

3 The proposed method

This study proposes a steganographic technique that offers both error correction capabilities and superior visual quality. The proposed method, which can be used in blind data hiding watermarking systems, integrates BCM codes to improve the error correction capabilities of LSB embedding. This study also employs a modulus function to improve the PSNR of the proposed method because using more LSBs to embed hidden information reduces the PSNR. Because every bit of a binary sequence converted by a pixel can be considered a subset of the signal constellation, this study employs various order Reed-Muller codes as the component codes of BCM codes to protect the LSB of each pixel when errors occur during the extraction process. Because the impact on visual quality is lower when information in hidden the first LSB compared to the other LSB, the block codes used to hide information in the first LSB must have equal or superior error correction capabilities to maintain both a high error correction capability and embedding payload. Therefore, the block code encoding length of the first LSB must be less than that of the second LSB. For example, if the block code chosen for the first LSB is a RM(2,4) with a code length of 16 and the ability to correct 1-bit errors, then the block code chosen for the second LSB must use a block code longer than 16 bits and correct 1 or more bits, such as RM(2,5) with a code length of 32 and the ability to correct 3-bit errors. The data encoding and extraction processes of the method proposed in this study are explained below.

In this study, a grayscale image sized at $\mathrm{M} \times W$ was used as the cover work, and Reed-Muller codes were used as the component codes of BCM codes.

3.1 Data embedding procedures

Assume *n* is the number of LSBs used for hiding information. Let the block code used for hiding information in the *i*th LSB be $RM(r_i, m_i)$, where $RM(r_i, m_i)$ is a

$$\left[n_i = 2^{m_i},\ k_i = \sum_{i'=0}^{r_i} \binom{m_i}{i'},\ d_i = 2^{m_i - r_i} \right]$$

linear block code and $i = 1, 2, \cdots, n$. The linear block code selected as the component code must satisfy the conditions of $n_{j+1} > n_j$ and $d_{j+1} \geq d_j$, $j = 1, 2, \cdots, 6$. First, a grayscale image sized at $\mathrm{M} \times W$ was adopted as the cover work, and the block divisions required for hiding information in the *i*th LSB were calculated. That is, the $\mathrm{M} \times W$ pixels $p_f, 1 \leq f \leq M \times W$ of the grayscale image were divided into $\left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$ blocks $\mathbf{b}_{i'}^{(i)}$, $1 \leq i' \leq \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$ and $i = 1, 2, \cdots, n$, where each block contains 2^{m_i} pixels. The proposed method is used to determine the size of the information hidden as *S* and sized at $\sum_{i=1}^{n} \left\lfloor \frac{\mathrm{M} \times W}{2^{m_i}} \right\rfloor \times (k_i)$ bits, where *n* is the number of LSBs used to hide the information. Then, the k_i bits obtained from the data being hidden and marked as $\mathbf{u}_i = \left(u_{i,1}, u_{i,2}, \cdots, u_{i,k_i}\right)$ are converted into Reed-Muller codewords $\mathbf{c}_i$ after the Reed-Muller encoding process is performed. Specifically, the codeword $\mathbf{c}_i$ is generated from $\mathbf{c}_i = \mathbf{u}_i \mathbf{G}_i$, where $\mathbf{G}_i$ is the generator matrix for $RM(r_i, m_i)$. To perform embedding, each pixel value p_f is first converted into an 8-bit binary sequence $\mathbf{l}_f = (l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8)$ representing the grayscale pixel value, where the grayscale pixel value p_f ranges between 0 and 255. Next, the proposed method sequentially embeds the content of codeword $\mathbf{c}_i$ with the length of 2^{m_i} into the *i*th LSB of the divided blocks. This method can be expressed as the *i*th LSB ($\mathbf{I}_{s_i}^{(v_i)} \in \mathbf{b}_{v_i}^{(i)}$)= $c_{i,s_i}^{(v_i)}$. $\mathbf{I}_{s_i}^{(v_i)}$ is the binary sequence converted from the s_i *th* pixel in block v_i, $\mathbf{b}_{v_i}^{(i)}$ is the v_ith block divided during the *i*th LSB embedding, and $c_{i,s_i}^{(v_i)}$ is the s_i *th* element from the codeword that is hidden in the v_ith block in the *i*th LSB, where $1 \le s_i \le 2^{m_i}$ and $1 \le v_i \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$. Each modified pixel p'_t of the v_1th block after the *n*-bit LSB embedding is used as the input value of the modulus function, where $t = 1,2,\cdots 2^{m_1}$ and *n* is the number of LSBs used for hiding information. We first employed modulus operation to calculate the following equation:

$$w_t = p'_t - (p_t \bmod 2^i),$$

where p_t is the original *t*th pixel of the v_1th block in the cover work and $t = 1,2,\cdots,2^{m_1}$. Then, the resulting value w_t was used to determine the minimum variance e_t through the following equations:

$$e_t = \begin{cases} w_t, \text{ if } \left(-\left\lfloor \frac{2^i-1}{2} \right\rfloor\right) \le w_t \le \left\lceil \frac{2^i-1}{2} \right\rceil \\ w_t + 2^i, \text{ if } \left(-2^i+1\right) \le w_t \le \left(-\left\lfloor \frac{2^i-1}{2} \right\rfloor\right), \\ w_t - 2^i, \text{ if } \left(\left\lceil \frac{2^i-1}{2} \right\rceil\right) \le w_t \le (2^i-1) \end{cases} \quad (1)$$

where $\lfloor s \rfloor$ represents the smallest integer closest to *s*, $\lceil v \rceil$ represents the largest integer closest to *v*, *i* = 1,2,..., n, and $t = 1,2,\cdots,2^{m_1}$. After calculating the modulus function, the resulting pixel p_t^* is obtained using the following decision equations.

$$p_t^* = \begin{cases} (p_t^* + e_t), \text{ if } 0 \le p_t^* + e_t \le 255 \\ (p_t^* + e_t - 2^i), \text{ if } p_t^* + e_t > 255, \\ (p_t^* + e_t + 2^i), \text{ if } p_t^* + e_t < 0 \end{cases} \quad (2)$$

where $t = 1,2,\cdots,2^{m_1}$. The embedding process is explained below.

Procedure: Data embedding

Input: Hidden data *S*, *n* LSBs embedded into a pixel, and a grayscale image $M \times W$ in size, where the grayscale image is used as the cover work. $RM(r_i, m_i)$ is used as the block codes for hiding information in the *i*th LSB, where $n_{j+1} > n_j$ and $d_{j+1} \ge d_j$, $j = 1,2,\cdots,6$ and *i*=1,2,...,*n*.

Output: A stego image **I'** $M \times W$ in size.

Step 1: Define the sequence of the $M \times W$ pixels in the grayscale image *I*, where $p_f, 1 \le f \le M \times W$, and divide the blocks according to the following procedure:
Blocks required to hide information in the *i*th LSB: Divide the $M \times W$ pixels $p_f, 1 \le f \le M \times W$ in the grayscale image into $\left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$ blocks $\mathbf{b}_{v_i}^{(i)}$, $1 \le v_i \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$, where each block contains 2^{m_i} pixels.

Step 2: Read the next k_{v_i} data bits from the payload *S* and mark as

$$\mathbf{s}_{v_i} = \left(u_{1,1}, u_{1,2}, \cdots, u_{1,k_{v_i}}\right).$$

Step 3: Generate the codeword $\mathbf{c}_{v_i}$ from

$$\mathbf{c}_{v_i} = \mathbf{s}_{v_i} \mathbf{G}_i,$$

where $\mathbf{G}_i$ is the generator matrices of $RM(r_i, m_i)$.

Step 4: Replace the *i*th LSBs content of the binary sequence that corresponds with the 2^{m_i} pixels in block $\mathbf{b}_{v_i}^{(i)}$ with content $c_{i,s_i}^{(v_i)}$ from the codeword $\mathbf{c}_{v_i}$. This can also be expressed as the *i*th LSB ($\mathbf{I}_{s_i}^{(v_i)} \in \mathbf{b}_{v_i}^{(i)}$)

$= c_{i,s_i}^{(v_i)}$, where $1 \le s_i \le 2^{m_i}$ and $1 \le v_i \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$.

Step 5: Compute the resulting value w_t according to $w_t = p'_t - (p_t \bmod 2^i)$, where p_t is the original *t*th pixel of the v_1th block in the cover work and $t = 1,2,\cdots 2^{m_1}$. Use the resulting value w_t to determine the minimum variance e_t according to (1). Finally, the resulting pixel p_t^* $t = 1,2,\cdots 2^{m_1}$ can be obtained using (2).

Step 6: Replace the *t*th pixel p_t of the v_1th block with the resulting pixel p_t^*, where $t = 1,2,\cdots,2^{m_1}$ and $1 \le v_1 \le \left\lfloor \frac{M \times W}{2^{m_1}} \right\rfloor$.

Step 7: Repeat Steps 2 to 6 until all payload *S* is embedded.

3.2 Data extraction procedures

Assuming the receiver is aware that the stego image is a grayscale image and the parameters of the Reed-Muller codes are r_i and m_i, when the receiver receives the stego image, all the stego image pixels use modulo 2^i operation. Additionally, the stego image after modulus operation is divided into $\left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$ blocks for extracting information from the *i*th LSB. Specifically, each block used to extract information from the *i*th LSB contains 2^{m_i} pixels

$$p_{t',received}^{(v_{t'})}, \quad t' = 1,2,\cdots,2^{m_i}, \text{ and } 1 \le v_{t'} \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor.$$

Pixels from each block containing *i*th LSB hidden information are converted into a binary sequence

$\mathbf{I}_{t'}^{(v_{t'})} = \left(I_1^{(v_{t'})}, I_2^{(v_{t'})}, I_3^{(v_{t'})}, I_4^{(v_{t'})}, I_5^{(v_{t'})}, I_6^{(v_{t'})}, I_7^{(v_{t'})}, I_8^{(v_{t'})}\right)$, where $t' = 1,2,\cdots,2^{m_i}$ and $1 \le v_{t'} \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$. The *i*th LSB f-rom the 2^{m_i} pixels in the v_ith steganographic blocks is extracted, where $1 \le v_i \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$. The extra-cted LSBs are combined into the binary sequences and expressed as

$$\mathbf{d}_i^{(v_i)} = \left(i\text{th LSB}(\mathbf{I}_1^{(v_i)}), i\text{th LSB}(\mathbf{I}_2^{(v_i)}), \cdots, i\text{th LSB}(\mathbf{I}_{2^{m_i}}^{(v_i)})\right).$$

Finally, the binary sequences $\mathbf{d}_i^{(v_i)}$ from each block are decoded with a r_ith order Reed decoding algorithm until all the payload *S* is extracted. The data extraction process can be consolidated into the following procedures:

Procedure: Data extraction

Input: A stego image I' $M \times W$ in size with n LSBs embedded in each pixel, where the codewords are generated by $RM(r_i, m_i)$ and *i* = 1,2,..., *n*.

Output: Original payload *S*

Step 1: First, all $M \times N$ pixels

$$p_{f,received}^{'}, \quad 1 \le f \le M \times W,$$

in the stego image **I'** employ modulo 2^n operation. After modulus operation, divide the $M \times W$ pixels $p_{f,received}^{*}$, $1 \le f \le M \times W$, into the $\left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$ information hiding blocks of the *i*th LSB, where each of the *i*th LSB blocks contain 2^{m_i} pixels

$$p_{t',received}^{(v_{t'})}, \quad t' = 1,2,\cdots,2^{m_i}$$

and

$$1 \le v_{t'} \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor.$$

Step 2: Read each pixel in the subsequent *i*th LSB information hiding blocks and convert them into the binary sequences

$$\mathbf{I}_{t'}^{(v_{t'})} = \left(I_1^{(v_{t'})}, I_2^{(v_{t'})}, I_3^{(v_{t'})}, I_4^{(v_{t'})}, I_5^{(v_{t'})}, I_6^{(v_{t'})}, I_7^{(v_{t'})}, I_8^{(v_{t'})}\right),$$

where $t' = 1, 2, \cdots, 2^{m_i}$ and $1 \le v_{t'} \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor$.

Step 3: Extract the *i*th LSB from 2^{m_i} binary sequences $\mathbf{I}_{t'}^{(v_{t'})}$,

$$t' = 1, 2, \cdots, 2^{m_i} \text{ and } 1 \le v_{t'} \le \left\lfloor \frac{M \times W}{2^{m_i}} \right\rfloor.$$

Assemble the extracted LSBs into binary sequences expressed as

$$\mathbf{d}_i^{(v_i)} = \left(i\text{th LSB}(\mathbf{I}_1^{(v_i)}), \cdots, i\text{th LSB}(\mathbf{I}_{2^{m_i}}^{(v_i)})\right).$$

Step 4: Decode the extracted codewords $\mathbf{d}_i^{(v_i)}$ into $\mathbf{s}_i$, using the r_i th order Reed decoding algorithm.

Step 5: Repeat Steps 2 to 4 until all payload *S* is extracted.

4. Experimental results

In this study, we conducted simulations using 8 grayscale images with 256×256 pixels, as shown in Fig. 2. We assessed the degree of visual differences between the cover work and the stego image with the embedded information using the PSNRs. PSNR is defined as

$$\text{PSNR=10}\log_{10}\left(\frac{255^2}{\text{MSE}}\right),$$

where $\text{MSE} = \frac{1}{M \times W} \sum_{i=0}^{M-1} \sum_{j=0}^{W-1} \left[I_{i,j} - I'_{i,j} \right]^2$, $\mathbf{I}$ is the original grayscale image, and $\mathbf{I}'$ is the stego image. Generally, when the PSNR exceeds 30 dB, visually distinguishing the differences between the stego image and the cover work is extremely difficult. To compare the payload size that can be embedded using various information hiding techniques, we assessed various embedding payloads *P* (calculated using the unit of bits-per-pixel, bpp), which can be defined as

$$P = \frac{\|S\|}{M \times W},$$

where $\|S\|$ refers to the size of the payload *S*. The larger the P value, the more data can be hidden in the stego image. Generally, PSNR declines with increases in the embedding payload.

Achieving a high embedding payload without being visually distinguishable by the human eye and simultaneously preventing errors is currently a popular topic of steganography research. The experimental results indicate that the method proposed in this study hides and protects data in the first and second LSB of each pixel in the cover work. To further analyze the error correction capabilities of LSB embedding, the method proposed by Chang, and the method proposed in this paper, we compared situations where stego images are disrupted by salt-and-pepper noise [13]. The value of each noisy pixel in the salt-and-pepper noise ranged between 0 and 255 and the image size was 256×256. The position where the stego image was disrupted by noise was randomly distributed. We used the error correction rate, which is expressed below, to evaluate the error correction capabilities of each technique.

$$C = \frac{b_c}{b_e}.$$

In this equation, b_c is the number of correct bits and b_e is the number of error bits after data extraction.

Figure 3 shows the block codes employed for the first LSB and second LSB embedding using RM(2,4) and RM(3,5). The error correction capability of both RM(2,4) and RM(3,5) can correct single-bit errors. For embedding data in the first LSB, the 256×256 original image was divided into 4,096 blocks with lengths of 8. For embedding data in the second LSB, the 256×256 image was divided into 2,048 blocks with lengths of 16. In Fig. 1, the payload size of each stego image is 98,304 bit and $P = \frac{\|S\|}{M \times W} = \frac{98,304}{256 \times 256} = 1.5$. The method proposed in this study guarantees PSNR values above 40 dB. Figure 4 shows the baboon image converted into a stego image using LSB embedding, Chang's proposed method, and the method proposed in this study. The experimental parameters are identical to those employed in Fig. 3; that is, the block codes employed for the first LSB and second LSB embedding are RM(2,4) and RM(3,5), respectively. The steganographic data extraction was disrupted by the same salt-and-pepper noise.

Assuming that half the 256×256 pixels in the stego image are disrupted by salt-and-pepper noise, as shown in Fig. 4(a), regardless of whether the PSNR of Chang's proposed method is superior to that of the method proposed in this study and of 2-bits LSB embedding, the embedding payload of the method proposed in this study is higher than that of Chang's proposed method. In Fig. 4(b), the error correction rate of the method proposed in this study is superior to that of the two other techniques. Table 1 shows a comparison of the PSNR, embedding payload, and error correction rates of the three steganographic techniques applied to 8 images. However, no error correction rate is displayed for LSB embedding in Table 1 because

LSB embedding does not possess error correction capabilities. The results in Table 1 show that the method proposed in this study provides a PSNR and embedding payload nearly equivalent to that of 2-bits LSB embedding while offering almost dou-ble the error correction rates provided by the other two techniques.

Figure 2. Eight grayscale images 256×256 pixels in size.

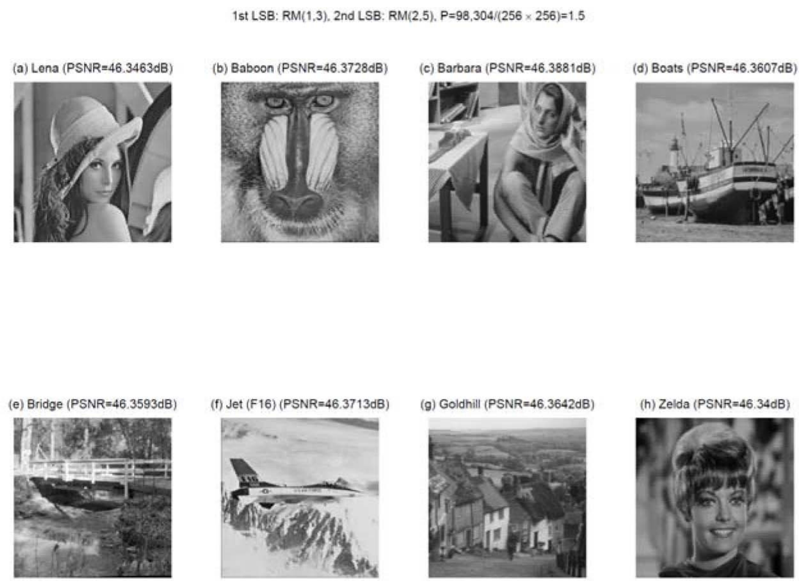
1st LSB: RM(1,3), 2nd LSB: RM(2,5), P=98,304/(256 × 256)=1.5
(a) Lena (PSNR=46.3463dB)
(b) Baboon (PSNR=46.3728dB)
(c) Barbara (PSNR=46.3881dB)
(d) Boats (PSNR=46.3607dB)
(e) Bridge (PSNR=46.3593dB)
(f) Jet (F16) (PSNR=46.3713dB)
(g) Goldhill (PSNR=46.3642dB)
(h) Zelda (PSNR=46.34dB)

Figure 3. The visual quality and embedding payload of the stego images created using the method proposed in this study.

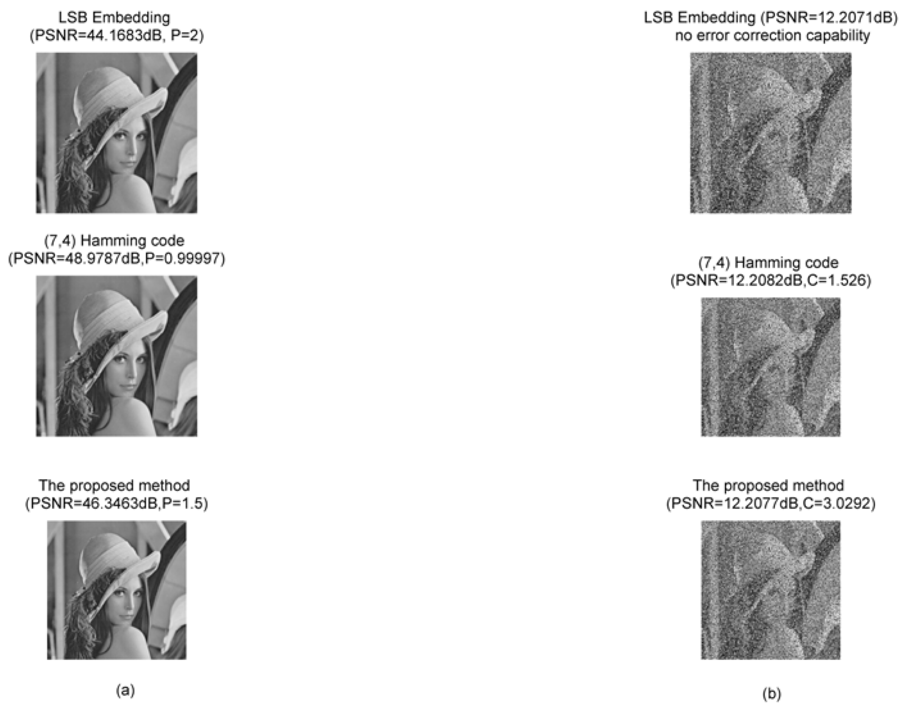

Figure 4. Comparison of the robustness of the three steganographic techniques under salt-and-pepper noise disruption: (a) undisrupted; and (b) after salt-and-pepper noise disruption.

Image Name	PSNR			Embedding payload (P)			Error correction rate (C)		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
Lena	44.1683	48.9787	46.3463	2	0.99997	1.5	-	1.526	3.0292
Baboon	44.1586	50.1165	46.3728	2	0.99997	1.5	-	1.526	3.0292
Barbara	44.1398	48.6595	46.3881	2	0.99997	1.5	-	1.526	3.0292
Boats	44.1584	48.1376	46.3607	2	0.99997	1.5	-	1.526	3.0292
Bridge	44.1371	51.1283	46.3593	2	0.99997	1.5	-	1.526	3.0292
Jet (F16)	44.1502	47.0562	46.3713	2	0.99997	1.5	-	1.526	3.0292
Goldhill	44.1483	50.8054	46.3642	2	0.99997	1.5	-	1.526	3.0292
Zelda	44.1732	49.7253	46.34	2	0.99997	1.5	-	1.526	3.0292

Table 1. Comparison of the three steganographic techniques according to their PSNR, embedding payload, and error correction rate under salt-and-pepper noise disruption: (a) LSB embedding; (b) Chang's proposed method; and (c) the method proposed in this study.

5. Conclusion

This study proposed a blind data hiding method with error correction capabilities and high embedding payloads by combining LSB embedding and BCM to produce stego images. Compared with Chang's proposed method, although the PSNR of the method proposed in this study is less than optimal, this method does not require substantial memory to construct standard arrays for extracting data. Additionally, our proposed method offers significantly superior embedding payloads and error correction capabilities.

Acknowledgements

This work was supported in part by the National Science Council (NSC) of Republic of China under grant NSC 101-2221-E-324-026 and NSC 101-2221-E-150-059 .

References

[1] F. Hartung and M. Kutter, "Multimedia Watermark-ing Techniques," Proceedings of the IEEE, vol. 87, no. 7, pp. 1079-1107, 1999.

[2] G. C. Langelaar, et al., "Watermarking digital image and video data. A state-of-the-art overview," IEEE Signal Processing Magazine, vol. 17, no.5, pp. 20-46, 2000.

[3] S. C Nirmal and B. Parthasarathy, "An Innovative Fusion Technique for Secured Video Watermark-ing," International Journal of Engineering and Innovative Technology, vol. 2, no. 2, pp. 20-22, 2012.

[4] S. Anand, et al., "Digital Watermarking," Internati-onal Journal of Enginee-ring and Innovative Technology, vol. 2, no. 2, pp. 151-155, 2012.

[5] C. C. Ramos, et al., "A Blind Video Watermarking Scheme Robust To Frame Attacks Combined With MPEG2 Compression," Journal of Applied Research and Technology, vol.8, no. 3, pp. 323-339, 2010.

[6] H. C. Wu, et al., "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," IEE Proceedings Vision, Image and Signal Proces-sing, vol. 152, no. 5, pp. 611-615, 2005.

[7] A. D. Ker, "Steganalysis of LSB matching in grayscale images," IEEE Signal Processing Letters, vol. 12, no.6, pp. 441-444, 2005.

[8] P. R. Rudramath and M. R. Madki, "Improved BPCS Steganography Based Novel Approach for Data Embedding," International Journal of Engine-ering and Innovative Technology, vol. 1, no.3, pp. 156-159, 2012.

[9] P. V Bodhak and B. L Gunjal, "Improved Protection In Video Steganography Using DCT & LSB," International Journal of Engineering and Innovativ-e Technology, vol. 1, no. 4, pp. 31-37, 2012.

[10] C. C. Chang,et al., "A High Payload Steganographic Scheme Based on (7, 4) Hamming Code for Digital Images," International Symposium on Electronic Commerce and Security, Guangzhou, China, 2008, pp. 16-21.

[11] C. C. Thien and J. C. Lin, "A Simple and High-hiding Capacity Method for Hiding Digit-by-digit Data in Images Based on Modulus Function," Pattern Recognition, vol. 36, no. 12, pp. 2875-2881, 2003.

[12] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications", Englewood Cliffs, NJ: Prentice-Hall, 2004, pp. 1063-1074.

[13] G. Pok, J. Liu, and A. S. Nair, "Selective removal of impulse noise based on homogeneity level information," IEEE Transactions on Image Processing, vol. 12, no. 1, pp. 85-92, 2003.