Improving Embedding Efficiency of Covering Codes for Applications in Steganography

Weiming Zhang, Shuozhong Wang, and Xinpeng Zhang

Abstract— **In image steganography, each pixel can carry a ternary message by choosing adding/subtracting one to/from the gray value. Although ternary covering functions can provide embedding efficiency higher than binary ones, it is necessary to convert the binary message into a ternary format. We propose a novel method that improves the embedding efficiency of binary covering functions by fully exploiting the information contained in the choice of addition or subtraction in the embedding. The improved scheme can perform equally well with, or even outperform, ternary covering functions without ternary conversion of the message.**

Index Terms— **Covert communication, steganography, embedding efficiency, covering codes, matrix encoding.**

I. Introduction

THE purpose of steganography is to send secret messages by embedding data into some innocuous cover-objects such as digital images. To reduce possibility of being detected by any third party, it is desirable to increase the embedding efficiency, which is the average number of message bits carried by one embedding change in the cover data. This may be accomplished by using an encoding technique proposed by Crandall [1] who called it matrix encoding. As a typical application of linear covering codes, matrix encoding was used in the well-known steganographic algorithm F5 [2]. The definition of covering codes can be found in [3]. Relations between covering codes and steganography were studied in [4] and [5], and some covering codes used in steganography with good performance reported in [5]. Matrix encoding was also used in large payload applications [6]. In [7], BCH codes were applied to achieve a tradeoff between embedding complexity and efficiency. Another example is the binary image steganography scheme CPT [8] that uses the Varshamov-Tenengolts codes. These encoding methods can improve the embedding efficiency by hiding messages in a block-by-block manner.

All the above-mentioned methods are applications of binary covering codes, which can be used in LSB steganography. For LSB embedding, the stego-coding methods are used in the LSB plane of an image, and adding 1 to a pixel is equivalent to subtracting 1 from the pixel for carrying the secret bit. In fact, by choosing adding/subtracting 1 (± 1 for short), every pixel can carry not just one bit but $\log_2 3$ bits of information, that is, a ternary digit, with the pixel gray value modulo 3. This implies that using ternary covering codes in this "± 1 steganography" can obtain better embedding efficiency than binary covering codes. Willems, et al., [9] provided an upper bound on the performance of " ± 1 steganography" and proposed to use the ternary Hamming and Golay codes. Zhang, et al., [10] brought up a method to improve embedding efficiency by exploiting the modification direction (EMD), which is a generalization of ternary Hamming codes. In all these methods, however, the message to be hidden is usually binary, and must be converted into ternary or m-ary digits before embedding with a ternary covering code or the EMD method.

The problem to be considered here is to make full use of the available information capacity of "± 1 steganography" by directly embedding a binary message without converting into ternary. Mielikainen [11] presented a method in which the choice of whether adding or subtracting one to/from a pixel value depends both on the original gray values and a pair of two consecutive secret bits. In this letter, we propose a novel embedding method that exploits the capacity of "± 1 steganography" more efficiently by extending the block of binary covering codes. This method can significantly improve the embedding efficiency of binary covering codes, and perform equally well with, or even outperform, ternary covering codes without binary-ternary conversion of the message.

II. Proposed Method

Matrix encoding embeds data with the parity check matrix of a linear covering code. Let C be an $[n, n-k]$ binary code with the covering radius R and a parity check matrix $\mathbf{H}$. Then $\mathbf{H}$ implies a covering function [5] $COV(R, n, k)$, which can embed k bits $(m_1, \ldots, m_k) \in F_2^k$ in the LSBs of n pixel gray values $(x_1, \ldots, x_n)$ by at most R changes in the following manner.

$$(m_1, \ldots, m_k)^T = \mathbf{H} \cdot (b(x_1), \ldots, b(x_n))^T \tag{1}$$

where $b(x_i)$ denotes the LSB of x_i.

Note that the covering radius R is the largest number of possible changes and the purpose of covering function $COV(R, n, k)$ is to minimize the average number of embedding changes R_a. In other words, the goal is to maximize the embedding efficiency k/R_a depending on the embedding rate k/n. The distribution of the number of embedding changes is equivalent to the distribution of the coset leaders' weight of the covering code. For perfect codes such as Hamming and Golay codes, there are exactly $\binom{n}{i}$ coset leaders with a weight

Manuscript received March 23, 2007. The associate editor coordinating the review of this letter and approving it for publication was Prof. Deepa Kundur. This work was supported in part by the National Natural Science Foundation of China under Grants 60473022, 60372090, and 60502039, in part by the Shanghai Rising-Star Program under Grants 06QA14022, and in part by the Key Project of Shanghai Municipality for Basic Research under Grant 04JC14037.

The authors are with the School of Communication and Information Engineering, Shanghai University, 200076, China.

W. Zhang (corresponding author) is also with the Department of Information Research, Information Engineering Institute, Zhengzhou 450002, China (e-mail: nlxd_990@yahoo.com.cn).

Digital Object Identifier 10.1109/LCOMM.2007.070438.

i, $0 \le i \le R$, so that the average number of changes can be calculated as $R_a = \frac{1}{2^k}\sum_{i=0}^{R} i\binom{n}{i}$.

Now taking Hamming and Golay codes as examples, we show how the performance of binary covering functions can be improved by extending their blocks.

A. *"Hamming+1" Scheme*

The parity check matrix of a Hamming code yields a covering function $COV(1, 2^k - 1, k)$, $k \ge 1$, i.e., embed k bits $(m_1, \ldots, m_k)$ into the LSBs of $2^k - 1$ pixel gray values $(x_1, \ldots, x_{2^k-1})$ using at most one change. This covering function is defined by $(m_1, \ldots, m_k)^T = \mathbf{H} \cdot (b(x_1), \cdots, b(x_{2^k-1}))^T$, where $\mathbf{H}$ is the parity check matrix of $[2^k - 1, 2^k - 1 - k]$ Hamming code.

We propose the following "Hamming+1" scheme by appending one pixel after the block of Hamming covering function. It embeds $k+1$ bits $(m_1, \ldots, m_k, m_{k+1})$ into 2^k pixel gray values $(x_1, \ldots, x_{2^k-1}, x_{2^k})$ using at most one change:

$$(m_1, \ldots, m_k)^T = \mathbf{H} \cdot (b(x_1), \cdots, b(x_{2^k-1}))^T \tag{2}$$

$$m_{k+1} = \left(\left\lfloor \frac{x_1}{2} \right\rfloor + \cdots + \left\lfloor \frac{x_{2^k-1}}{2} \right\rfloor + x_{2^k}\right) \bmod 2 \tag{3}$$

In this way, the first k bits are embedded into LSBs of the first $2^k - 1$ pixel values using the $COV(1, 2^k - 1, k)$ Hamming covering function, and the $(k+1)$-th bit is a function of all 2^k pixels including the appended one. Note that, by adding or subtracting one to/from a pixel value x, its LSB $b(x)$ always becomes the same binary value $b(x) \oplus 1$, but $\lfloor x/2 \rfloor \bmod 2$, which is the second least significant bit of x, can either be 0 or 1. Therefore, when (2) does not hold, one pixel value, say, x_i, $1 \le i \le 2^k - 1$, has to be changed. By choosing $x_i + 1$ or $x_i - 1$, both (2) and (3) can hold simultaneously without changing x_{2^k}. On the other hand, when (2) holds but (3) does not, the first $2^k - 1$ pixels need not to be changed, and we can modify x_{2^k} by randomly increasing or decreasing one to satisfy (3). This means that we can embed $k+1$ bits of message in 2^k pixels with at most one change. However, above embedding process may fail when the pixel value x_i is saturated, e.g., $x_i = 0$ or 255 in an 8-bit gray scale image. For instance, consider $x_i = 255$ but unfortunately $x_i + 1$ should be chosen to make (3) hold. In this case, only $x_i - 1$ is permitted, and therefore we have to select $x_i - 1$ to satisfy (2) and change x_{2^k} to satisfy (3) simultaneously. Nonetheless, the effect of this additional change on the overall performance can be neglected since these situations rarely occur.

Because with $COV(1, 2^k - 1, k)$ Hamming covering function, i pixels need to be modified with probability $\binom{2^k-1}{i}/2^k$, $i = 0$ or 1, the average number of changes of the above "Hamming+1" scheme can be calculated as

$$\begin{aligned} R_a &= P\{(2) \text{ does not hold}\} \times 1 \\ &\quad + P\{(2) \text{ holds}\} \times P\{(3) \text{ does not hold}\} \times 1 \\ &= \frac{\binom{2^k-1}{1}}{2^k} + \frac{\binom{2^k-1}{0}}{2^k} \times \frac{1}{2} = \frac{2^{k+1}-1}{2^{k+1}} \ . \end{aligned} \tag{4}$$

This means that, with the "Hamming+1" scheme, we can embed one more bit than $COV(1, 2^k - 1, k)$ at the cost of $1/2^{k+1}$ more changes. The embedding efficiency is equal to $(k+1)2^{k+1}/(2^{k+1} - 1)$, and the embedding rate is $(k+1)/2^k$. Fig. 1 shows that the performance of Hamming covering functions is improved significantly by using the "Hamming +1" scheme. Note that when taking $k = 1$, the "Hamming+1" scheme becomes the method as described in [11].

B. *"Golay+2" Scheme*

The $[23, 12]$ Golay code, whose covering radius is 3, implies a $COV(3, 23, 11)$ Golay covering function, which can also be improved as the "Golay+1" scheme using the method described in the previous subsection. Since the largest change is 3 in the Golay covering function, we can try to append two pixels to the block and get the "Golay+2" scheme, which can embed 13 bits $(m_1, \ldots, m_{13})$ in 25 pixel gray values $(x_1, \ldots, x_{25})$ by making at most 3 changes as follows.

$$(m_1, \ldots, m_{11})^T = \mathbf{H} \cdot (b(x_1), \ldots, b(x_{23}))^T \tag{5}$$

$$(m_{12}, m_{13}) = \left(\left\lfloor \frac{x_1}{2} \right\rfloor + \cdots + \left\lfloor \frac{x_{23}}{2} \right\rfloor + x_{24} + x_{25}\right) \bmod 4 \tag{6}$$

where $\mathbf{H}$ is the parity check matrix of $[23, 12]$ Golay code. In this scheme, the first 11 bits are embedded in the LSBs of the first 23 pixel values using the Golay covering function, and the last two bits are denoted by a function of all 25 pixel values as in (6).

To calculate the average number of changes, consider the following 4 cases:

Case I. In the first 11 pixels, gray values of three pixels, for example x_1, x_2, x_3, need to be changed. By adding or subtracting one to/from these three pixel values respectively, $(\lfloor x_1/2 \rfloor + \lfloor x_2/2 \rfloor + \lfloor x_3/2 \rfloor) \bmod 4$ can take all the values in Z_4. In this case, (6) can always hold without changing x_{24} and x_{25}.

Case II. Two pixel values, say x_1 and x_2, are changed. By ± 1, $(\lfloor x_1/2 \rfloor + \lfloor x_2/2 \rfloor) \bmod 4$ can only take three values in Z_4. Therefore, to make (6) hold, one of x_{24} and x_{25} should be modified with probability $1/4$.

Case III. One pixel value, say, x_1, is changed. By ± 1, $\lfloor x_1/2 \rfloor \bmod 4$ can only take two values of Z_4, Therefore, to make (6) hold, one of x_{24} and x_{25} should be modified with probability $1/2$.

Case IV. No change is made in the first 23 pixels. In this case, both x_{24} and x_{25} must be modified when (6) does not hold which has probability $3/4$.

On the other hand, with $COV(3, 23, 11)$ Golay covering function, a total of i pixels need to be changed with probability $\binom{23}{i}/2^{11}$, $0 \le i \le 3$. Therefore the average number of changes in the "Golay+2" scheme can be calculated as

$$\begin{aligned} R_a &= \left(\frac{\binom{23}{1}}{2^{11}} + \frac{\binom{23}{2}}{2^{11}} \times 2 + \frac{\binom{23}{3}}{2^{11}} \times 3\right) \\ &\quad + \left(\frac{\binom{23}{2}}{2^{11}} \times \frac{1}{4} + \frac{\binom{23}{1}}{2^{11}} \times \frac{1}{2} + \frac{\binom{23}{0}}{2^{11}} \times \frac{3}{4} \times 2\right) \\ &= 2.89 \ . \end{aligned} \tag{7}$$

Hence the "Golay+2" scheme can achieve the embedding efficiency $13/2.89 = 4.50$ with the embedding rate $13/25 =$

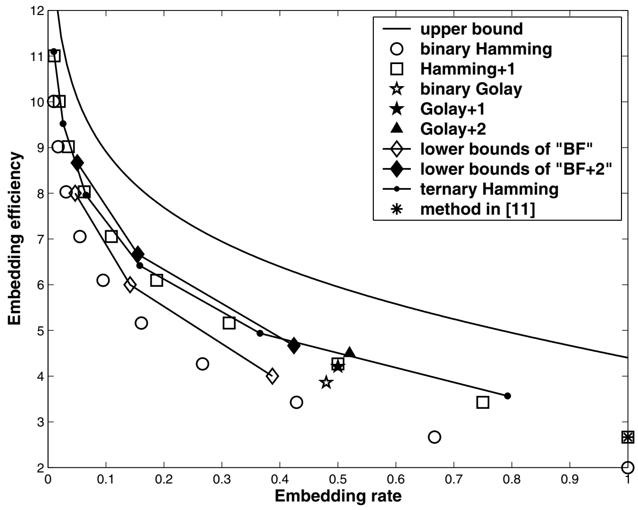

Fig. 1. Performance comparison among improved schemes, some binary covering functions, ternary Hamming covering function and the method in [11].

0.52. And the pairs of performance parameters, (embedding rate, embedding efficiency), for binary Golay and "Golay+1" schemes are respectively $(0.48, 3.86)$ and $(0.50, 4.21)$. Obviously, both the embedding rate and the embedding efficiency are improved by extending the block.

C. Improving Other Binary Covering Functions

It is clear that with the above method, we can improve performance of other covering functions such as those in [5] and [7]. Most known good covering functions have the largest changes R such that $R \leq 3$, and only a few with $R = 4$ or 5. A covering function $COV(R, n, k)$ can be improved either by appending one pixel if $R = 1, 2$, or by appending two pixels if $R \geq 3$. For example, the following covering functions

$$COV(3, 31, 12), COV(3, 127, 18), COV(3, 511, 24) \quad (8)$$

proposed by Bierbrauer and Fridrich [5], which we call "BF" schemes for brevity, can be improved by appending two pixels and called correspondingly "BF+2" schemes. For some covering functions $COV(R, n, k)$, it is hard to calculate the average number of changes R_a. In this case, we substitute the largest changes R for R_a to calculate a lower bound of embedding efficiency as k/R, and evaluate the embedding efficiency of the "BF" and "BF+2" schemes with this lower bound. For the "BF" schemes, the performance parameters, (embedding rate, lower bound of embedding efficiency), are $(0.39, 4)$, $(0.14, 6)$ and $(0.05, 8)$, which are increased respectively to $(0.42, 4.67)$, $(0.16, 6.67)$ and $(0.05, 8.67)$ when using "BF+2".

III. Performance Comparison

Performance comparison has been made between these improved schemes with the ternary Hamming covering functions in Fig. 1 where the upper bound is derived from the theoretic result of [9]. Using ternary Hamming codes, one can get higher embedding efficiency than those previous binary covering functions, but has to convert the binary message to ternary digits first. Fig. 1 illustrates that the "Hamming+1" approaches the performance of "ternary Hamming", and the "Golay+2" exceeds the "ternary Hamming". The "BF+2" schemes also provide better performance because the lower bounds of the "BF+2" are even somewhat higher than the "ternary Hamming".

References

[1] R. Crandall, "Some notes on steganography," Posted on steganography mailing list http://os.inf.tu-dresden.de/ westfeld/crandall.pdf (1998).

[2] A. Westfeld, "F5: a steganographic algorithm," in *Proc. 4th Int. Workshop Information Hiding 2001*, Lecture Notes in Computer Science, vol. 2137, pp. 289-302, 2001.

[3] J. Bierbrauer, *Introduction to Coding Theory*, Section 14.2, Chapman and Hall, CRC Press, 2005.

[4] F. Galand and G. Kabatiansky, "Information hiding by coverings," in *Proc. IEEE Information Theory Workshop 2004*, pp. 151-154, 2004.

[5] J. Bierbrauer and J. Fridrich, "Constructing good covering codes for applications in steganography," available: http://www.math.mtu.edu/ jbierbra/ (2006).

[6] J. Fridrich and D. Soukal, "Matrix embedding for large payloads," *IEEE Trans. Inf. Security Forensics*, vol. 1, no. 3, pp. 390-394, Sept. 2006.

[7] D. Schönfeld and A. Winkler, "Embedding with syndrome coding based on BCH codes," in *Proc. 8th ACM Workshop on Multimedia and Security*, pp. 214-223, 2006.

[8] Y.-C. Tseng, Y.-Y. Chen and H.-K. Pan, "A secure data hiding scheme for binary images," *IEEE Trans. Commun.*, vol. 50, no. 8, pp. 1227-1231, 2002.

[9] F. Willems and M. Dijk, "Capacity and codes for embedding information in gray-scale signals," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 1209-1214, Mar. 2005.

[10] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Commun. Lett.*, vol. 10, no. 11, pp. 781-783, Nov. 2006

[11] J. Mielikainen, "LSB matching revisited," *IEEE Signal Processing Lett.*, vol. 13, no. 5, pp. 285-287, May 2006.