A Steganographic Protocol Based on Linear Error-Block Codes

Rabiî Dariti and El Mamoun Souidi
Laboratory of Mathematics, Computer Science and Applications, Faculty of Science, Mohammed V University-Agdal , BP 1014 Rabat, Morocco

Keywords: Steganography, Linear Error-Block Codes, Heterogeneity, Bit Planes.

Abstract: We present a steganographic protocol based on linear error-block codes. Recent works have showed that these codes allow to increase the number of information carrier bits within a given cover by exploiting multiple bit planes (not only LSB plane) from pixels which would not have a perceptible influence on the cover. We employ a parameter, called heterogeneity, to assess the ability of pixels to be modified without perturbing the cover. The quality of the modified cover is handled by tuning a vector of heterogeneity thresholds which determines the number of bit planes that we are allowed to use for each pixel in the cover.

1 INTRODUCTION

The objective of code based steganography is to create a cover y, with minimum modifications regarding the original cover x, such that the syndrome of y is the secret message m. In other words, the vector y must be the closest vector to x whose syndrome is m. Concretely, let $\mathcal{C}$ be a linear error correcting code of parity-check matrix H. The retrieval map requires simply computing the syndrome of the modified cover $R(y) := S(y) = Hy^T$. The embedding algorithm consists of three steps. First compute $u := S(x) - m$, then find e_u, a word of smallest weight among all words of syndrome u. In other words, e_u must be the leader of the coset of syndrome u. This is exactly the syndrome decoding problem. Finally, modify the cover by computing $E(m, x) := x - e_u$. It is easy to verify that we have

$$\begin{aligned} R\big(E(m,x)\big) = S(x - e_u) &= S(x) - S(e_u) \\ &= S(x) - u \\ &= m. \end{aligned} \tag{1}$$

The idea of using error correcting codes in steganography was firstly brought up by (Crandall, 1998). His main objective was to use an error correcting code to randomize the bits used to hide the secret message. His method was called matrix embedding. In 2001, Westfeld implemented an algorithm called the F5 algorithm, which was the first steganographic scheme based on matrix embedding (Westfeld, 2001). He used the Hamming code of length 7, which allows embedding 3 message bits within each 7 cover bits by altering at most one of these bits.

To ensure the imperceptibility of embedding, the majority of steganographic protocols modify only least significant bits. The main condition to be able to use further bits remains the imperceptibility of their alteration. Consider an 8-bit gray-scale image. The number 8 is called the bit-depth; it determines the number of bits within which a pixel is encoded. A set of bits corresponding to a given bit position between 1 and 8 in all pixels is called a *bit signification plane* or, simply, a *bit plane*. The first bit plane, corresponding to the set of bits in position 1, consists of the least significant bits (LSB) of all the pixels. Obviously, the first bit plane is the most suitable to be modified. There are many embedding methods that use bit signification planes in different ways (Zhang, 2007; Zhang, 2007; Liao, 2008; Dariti, 2011). In our method, we use multiple bit planes if their modification does not cause a noticeable change to the cover. We assess the ability of pixels to support alteration by calculating a parameter called heterogeneity, which measures the resemblance between a pixel and the pixels adjacent to it. The more a pixel is different, the higher is its heterogeneity value and the more it is suitable to carry information by modifying its value. We benefit from pixels which have significantly high heterogeneity values to exploit further bit planes.

We define a list of heterogeneity thresholds in order to determine the highest bit plane we can use and the number of pixels for which a given bit plane is to be used. The added value of linear error-block codes comes from their block structure. When blocks are of different sizes, a random single bit error is more probably to occur in a block of big size than in a block of small size. Therefore, we construct block vectors by putting sensitive bits in small size blocks and less sensitive bits in big size blocks. Bits belonging to similar bit planes are put in blocks of the same size. The decoding procedure of linear error-block codes ensures that the probability of modifying small size blocks is less than the probability of modifying bigger ones. Thus the modified cover is expected to have good quality parameters. Moreover, these parameters can be handled by adjusting the heterogeneity thresholds.

This paper is organized as follows. Section 2 recalls some preliminaries about linear error-block codes and describes their convenience in steganography. Section 3 presents the method used to assess the quality of the modified cover. Section 4 shows how can linear error-block codes ensure a big quantity of payload at the same time as keeping a good quality of the cover. Experimental results are given in Section 5. Finally, Section 6 gives concluding remarks.

2 LINEAR ERROR-BLOCK CODES IN STEGANOGRAPHY

Linear error-block codes are a generalization of linear error correcting codes, introduced by (Feng, 2006). The Hamming metric is replaced by a more general distance called the π-metric, which is defined as follows. For a given non null integer n, $\pi = [n_1][n_2] \dots [n_s]$ is said to be a *partition* of n if $n = n_1 + n_2 + \dots + n_s$ where $n_1 \geq n_2 \geq \dots \geq n_s \geq 1$ and s is a non null integer. In the case

$$\pi = \underbrace{[m_1] \dots [m_1]}_{l_1\ terms} \underbrace{[m_2] \dots [m_2]}_{l_2\ terms} \dots \underbrace{[m_r] \dots [m_r]}_{l_r\ terms}$$

we write $\pi = [m_1]^{l_1}[m_2]^{l_2} \dots [m_r]^{l_r}$ where $m_1 > m_2 > \dots > m_r \geq 1$ and r and l_j for $j = 1,2,\dots,r$ are non null integers. The integers n_i, $i = 1,2,\dots,s$ and m_j, $j = 1,2,\dots,r$, are called *summands* of π. Let q be a prime power and $\mathbb{F}_q$ be the finite field with q elements. Given a partition $\pi = [n_1][n_2] \dots [n_s]$ of n, the vector space $\mathbb{F}_q^n$ over $\mathbb{F}_q$ can be viewed as the product $\mathbb{F}_q^n = \mathbb{F}_q^{n_1} \times \mathbb{F}_q^{n_2} \times \dots \times \mathbb{F}_q^{n_s}$ noted V_π. Consequently, each vector $v \in V_\pi$ can be uniquely written as $v = v_1 v_2 \dots v_s$, where $v_i \in \mathbb{F}_q^{n_i}$. For each pair u and v in V_π, the π-weight $w_\pi(u)$ of u and the π-distance $d_\pi(u,v)$ between u and v are defined by $w_\pi(u) = \#\{i |\ 1 \leq i \leq s, u_i \neq 0 \in \mathbb{F}_q^{n_i}\}$ and $d_\pi(u,v) = w_\pi(u - v) = \#\{i |\ 1 \leq i \leq s, u_i \neq v_i\}$.

An $\mathbb{F}_q$-linear subspace C of V_π is called an $[n,k]_q$ *linear error-block code* of type π over $\mathbb{F}_q$, where $k = \dim_{\mathbb{F}_q} C$. Notice that a classical linear error correcting code is a linear error-block code of type $\pi = [1]^n$.

The idea of using linear error-block codes in steganography was introduced by (Dariti, 2011). As vectors in V_π are concatenations of s blocks, a one-block error vector is a vector in V_π which has a single non-zero block. This does not necessarily mean that only a single one of its bits is non-zero, but that all of its non-zero bits are located within the same block. In steganographic terms, assume that to embed a k-bit message we have to modify r bits of the n-bit cover. These r bits are located in r' blocks where $r' \leq r$. Thus we have to provoke r' errors, whilst with a classical code we provoke r errors. Therefore, the correction capacity of linear error-block codes does not need to be as big as in the classical case. A drawback of this method is the following. Recall that a coset leader is a vector of minimum π-weight within its coset. Let us consider a code where the two vectors $e_1 = 11000$ and $e_2 = 00010$ belong to the same coset. If the code is of type $\pi = [3][1]^2$ the vector e_1 can be considered as coset leader. If we use it to embed some message then we need to modify 2 bits in the cover, whereas with a classical code (type is $\pi = [1]^5$) we need to modify only one bit since the vector e_2 will be the coset leader. Actually, in this method the cover bits are reorganized into blocks of specific sizes in order to give preferential treatment to each block of bits. We explain in the following two facts justifying that it might be more suitable to flip a whole block of relatively big size rather than a single bit from another block of smaller size.

In general, pictures feature areas that can better hide distortion than other areas. The human vision system is unable to detect changes in heterogeneous areas of a digital media due to the complexity of such areas. For example, if we modify the gray values of pixels in smooth areas of a gray-scale image, they will be more easily noticed by the human eye. Furthermore, the pixels in edged areas may tolerate larger changes of pixel values without causing noticeable modifications. So we can keep the changes in the modified image unnoticeable by embedding more data in edged, or heterogeneous, areas than in smooth, or homogeneous, areas. Thus we can use large blocks for the most heterogeneous pixels, and smaller blocks for less heterogeneous pixels. The second fact consists of using further bit planes to the least significant bit plane. It is clear that modifying some bits from a given bit plane is more suitable than modifying fewer bits from a higher bit plane. This gives another solution by putting least significant plane bits in large blocks and higher plane bits in smaller and smaller blocks. The method we present in this paper combines these two solutions. Firstly, we select heterogeneous pixels and use their first bit plane, then we tighten our heterogeneity selection to get only the most heterogeneous ones among them and, if we get enough pixels, we use their second bit plane. We continue this process to get further bits from higher bit planes. Thus, we get more information carrier bits. Meanwhile, our selection criterion ensures a minimum perceptibility of modifications, and the decoding algorithm of linear error-block codes ensures that modifications are most likely to occur over suitable bits, as we are going to see thoroughly within the next sections.

3 COVER QUALITY

An important step in any steganographic scheme is selecting the bits from the cover which are going to carry the message. These bits are called *the selection channel*. Basically, a steganographic scheme aims to minimize the modifications made on the selection channel. Furthermore, a well-chosen selection channel reduces the possibility of detecting the modifications. Hence, our aim is to hide as many bits as possible by making the minimum modifications in the cover.

Choosing the selection channel in image covers is an approximate process. Although there are many parameters that evaluate the quality of a modified image, no parameter can precisely compare the similarity between two images. The *Mean-Squared Error* (MSE) and the *Peak Signal to Noise Ratio* (PSNR) are widely used in the literature. The mean-squared error between two images, presented by their pixel value matrices $I_1(m,n)$ and $I_2(m,n)$, is defined by

$$MSE = \frac{\sum_{m\in M, n\in N}\,[I_1(m,n) - I_2(m,n)]^2}{M \times N} \qquad (2)$$

where M and N are the numbers of rows and columns respectively in the input images. The mean-squared error depends strongly on the image intensity scaling. An MSE of 100.0 for an 8-bit image (with pixel values in the range $0-255$) looks dreadful, but for a 10-bit image (pixel values in $0-1023$) it is barely noticeable.

The Peak Signal-to-Noise Ratio avoids this problem by scaling the MSE according to the image range:

$$PSNR = 10\log_{10}\left(\frac{D^2}{MSE}\right) \qquad (3)$$

where D is the maximal value possible for a pixel of the image, i.e. $D = 2^d - 1$ where d is the bit-depth of the image. PSNR is measured in decibels (dB). It is a good measure for comparing restoration results for the same image, but comparing two completely different images by PSNR is meaningless. Values over $36\,dB$ in PSNR are acceptable in terms of degradation, which means no significant degradation is observed by human eye (Tan, 2003). Nevertheless, both MSE and PSNR rely on pixel intensities instead of image structure.

p_1	p_2	p_3
p_4	p	p_5
p_6	p_7	p_8

Figure 1: Adjacent pixels to a pixel p.

In order to predict the influence of modifying a given pixel on the cover image, the method we propose consists first of assessing the heterogeneity of all cover pixels. Heterogeneity is a statistical value that presents the resemblance between a pixel and the pixels adjacent to it (see Figure 1). Modifying pixels of large heterogeneity values should produce images with unnoticeable changes (small MSE and big PSNR values, relatively). We set heterogeneity $h(p)$ of a pixel p to be the sum of the absolute values of the differences between the value of the pixel and the values of pixels adjacent to it, divided by the number of the adjacent pixels. For pixels which are not located within the edge of the image we have the following formula.

$$h(p) = \frac{1}{8}\sum_{i=1}^{8} |p - p_i| \qquad (4)$$

Heterogeneity of pixels within the edges is assessed by an adapted formula using only the existing adjacent pixels. For smooth pixels we expect $h(p)$ to be a small value and it will have a larger value for more complex pixels.

By calculating the heterogeneity values, we construct a matrix, corresponding to the cover image, the entries of which indicate the location of the selection channel. Moreover we assign to each pixel the number of bit planes we are allowed to exploit. This is done by determining a list of thresholds that present the heterogeneity value required for modifying a given bit plane. A simple formula to calculate the thresholds is given by

$$t_j = \frac{\max_{p\in\mathcal{P}}(h(p)) - \min_{p\in\mathcal{P}}(h(p))}{\#(\mathcal{P})} \times j, \text{for } j = 1,2,\dots,r \quad (5)$$

where $\mathcal{P}$ is the list of all cover pixels and r is the greatest bit plane we decided to use. The values t_j can be adjusted to allow us to have more or less pixels in a given level. For example, we consider the 8-bit 256×256 gray-scale image shown in Figure 2 and we set the list of heterogeneity thresholds to $(30{,}50{,}80)$. This produces three sets of pixels that we call *pixel levels*, which we illustrate in Figure 3. First pixel level contains the pixels whose heterogeneity value is greater than 30. Only the first bit plane of these pixels is exploited. In the second pixel level we exploit the first and the second bit planes. The heterogeneity values of these pixels must be greater than 50. The third pixel level is the most resistant to alteration; the heterogeneity value of its pixels must be greater than 80, so we can exploit all of their first 3 bit planes. The pixels of heterogeneity value less than 30 are not used.

Figure 2: The cover image.

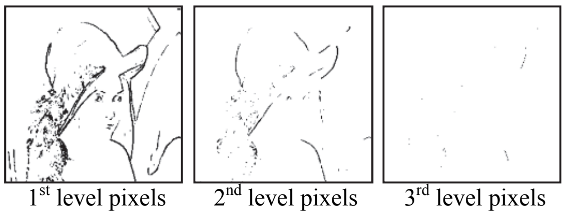

Figure 3: Heterogeneity levels.

In this way we construct an overall matrix, called *pixel levels matrix*, whose entries are in $\{0,1,2,\dots,r\}$. They are represented in Figure 4 by gray values starting from 0 (white) for pixels which are not used, to 3 (black) for 3rd level pixels. The sender and the receiver should agree on this matrix in addition to the code's parity-check matrix before communicating.

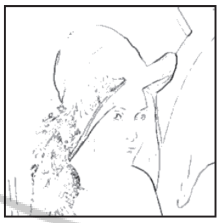

Figure 4: Pixel levels matrix for thresholds (30,50,80).

4 HANDLING COVER BITS WITH LINEAR ERROR-BLOCK CODES

The pixel levels matrix defines, for each pixel level $i = 1,2,\dots,r$, a sequence of N_i bits. Generally, in any significant image most areas are homogeneous, thus the number of pixels within high heterogeneity thresholds goes fewer and fewer. This is suitable for our method since they will be put in small size blocks. However, the number of bit planes we are going to use should provide enough pixels within high thresholds. If this was not the case, we should simply decrease heterogeneity thresholds, although this would also decrease the quality parameters.

Our embedding process starts by splitting the secret message into parts of $n - k$ bits. To embed a given part, we construct a carrier vector of n bits using different pixel levels in the following way. First we consider a partition $\pi = [m_1]^{l_1}[m_2]^{l_2}\dots[m_r]^{l_r}$ of the integer n where $m_1 > m_2 > \cdots > m_r \geq 1$ and r is the number of bit planes to be used. From each pixel level $\{1,\dots,r\}$ we take an $l_i m_i$-bit vector x_i and construct a carrier vector in the form $x = (x_1, x_2, \dots, x_r)$ where $x_i \in \mathbb{F}_2^{l_i m_i}$. The maximum number of carrier vectors we can construct is $\min\left\{\left\lfloor \frac{N_i}{l_i m_i} \right\rfloor, i = 1,2,\dots,r\right\}$. This number is handled (increased or decreased) by our choices of partition and/or heterogeneity thresholds.

Let us index the elements of π and write it in the form $\pi = [n_1][n_2]\dots[n_s]$ where $n_1 \geq n_2 \geq \dots \geq n_s \geq 1$. For one bit modification in the n-bit carrier vector, the probability that this bit is located in the

i^{th} block is $\frac{n_i}{n}$. Consequently, the modification of the first plane bits is more probable than the modification of the second plane bits, and the modification of the second plane bits is more probable than the modification of the third plane bits, and so on. This ensures a good quality of the modified image beforehand.

Let us see now what happens in the classical case where $\pi = [1]^n$. In this case only one heterogeneity threshold is used and it determines the lowest heterogeneity value a modifiable pixel must have. Thus, the whole message is to be embedded within least significant bits (first bit plane). This ensures a very good quality of the modified cover on one hand, but on the other hand the number of information carrier bits will be small. We can provide more bits by decreasing the heterogeneity threshold, but their number remains limited in comparison with the general case. Therefore, the classical code based steganography would work only for small size messages.

The choice of partition can be improved by the following. On one hand, in order to have a minimum influence on the cover quality, we take $m_r = 1$ so that a minimum number of r^{th} plane bits will be used. On the other hand, changing an i^{th} plane bit of a pixel gives the same MSE and PSNR values as changing four $(i-1)^{th}$ plane bits. This leads us to set the summands of the partition in such a way that each summand is four times its successor. Consequently, the partition should have the form $\pi = [4^{r-1}]^{l_{r-1}}[4^{r-2}]^{l_{r-2}} \dots [4]^{l_1}[1]^{l_0}$. This requires the code length to be related to the number of used bit planes r by $n = \sum_{i=1}^{r} 4^{i-1} = \frac{4^{r+1}-1}{3}$. However, the selection channel can provide better choices of partition by analyzing the image structural properties. The number of bit planes to use depends essentially on the density of pixels with high heterogeneity value in the cover. If the cover bits were supposed to have equal influence on image quality, for example all the pixels have close heterogeneity values and/or only the LSB plane can be used, then π turns to the classical type $\pi = [1]^n$.

Now we recapitulate the steps of our protocol. Let m be an $(n-k)$-bit message.

Secret sharing:

(a) Compute pixels heterogeneity, choose the number of bit planes to use r, set heterogeneity thresholds and compute the pixel levels matrix,

(b) Choose an $(n-k) \times n$ parity-check matrix H and a partition $\pi = [n_1][n_2]\dots[n_s]$ with r distinct summands.

Embedding:

(a) Using the pixel levels matrix construct an n-bit carrier vector of the form $x = (x_1^1 \dots x_{n_1}^1, x_1^2 \dots x_{n_2}^2, \dots, x_1^s, \dots, x_{n_s}^s)$,

(b) Compute $u = Hx^T - m$,

(c) Find e the leader of the coset with syndrome u,

(d) Compute $y = x - e$,

(e) Update the image by replacing each bit x_j^i by the bit y_j^i.

Retrieval:

(a) Using the pixel levels matrix construct $y = \left(y_1^1 \dots y_{n_1}^1, y_1^2 \dots y_{n_2}^2, \dots, y_1^s, \dots, y_{n_s}^s\right)$,

(b) Compute $m = Hy^T$.

5 RESULTS

We tested our protocol by embedding the 8-bit 64×64 gray-scale image shown in Figure 5, which contains 32768 information bits, within the 8-bit 256×256 gray-scale image shown in Figure 2, whose each bit plane contains 65536 bits.

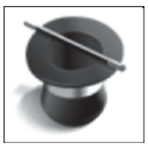

Figure 5: The message image.

In the following we present the results of applying several partitions with the [7,3] linear error-block code given by the parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

This code embeds a 4-bit message in a carrier of 7 bits. Thus, our message requires 8192 carrier vectors. The maximum number and the average number of altered bits cannot be described by the code parameters as can be done in the classical case by the covering radius value ρ and the ratio $\frac{\rho}{n}$ respectively. These parameters do not give a precise assessment in our case since they do not depend on the partition summands. Even if we consider the block-covering radius ρ_π (Dariti, 2013) it would give misleading values, since the alteration of a

Table 1: Image quality using a [7,3] linear error-block code.

π	Heterogeneity thresholds	Number of blocks provided by each bit plane	MSE	PSNR (dB)
$[1]^7$	1.8	8286	0.1798	55.5816
$[4][1]^3$	$(4,7)$	$(9399, 8387)$	0.3711	52.4349
$[4][3]$	$(4,7)$	$(9399, 8387)$	0.6487	50.0104
$[5][2]$	$(4,7)$	$(7519, 12580)$	Not enough 1st plane blocks	
$[5][2]$	$(3.5,11.3)$	$(8207, 8327)$	0.2951	53.4308
$[4][2][1]$	$(4.7, 11, 20)$	$(8521, 8521, 8514)$	0.7438	49.4159

given block can be caused by the alteration of any number of its bits, which will give the same value for different number of altered bits within the same block. Consequently, the reliability of our protocol is only assessed by the influence of alteration on the image, presented by the MSE and PSNR values. In Table 1 we also give the number of blocks provided by each bit plane to construct carrier vectors. The classical protocol corresponds to the classical type $[1]^7$ given in the first row of the table. Note that it uses only least significant bits. Therefore, to get enough information carrier bits, we had to set the heterogeneity threshold to a very low value.

By comparing MSE and PSNR values, it is clear that partition $[4][1]^3$ is better than $[4][3]$. Both of them use 2 bit planes and require the same number of bits, but the first partition has a better structure as was described in Paragraph 5, Section 4. Partition $[5][2]$ with heterogeneity thresholds $(4,7)$ does not provide enough bits to embed the message. We adjusted these to $(3.5,11.3)$ which provided higher embedding capacity, though at the cost of some cover quality degradation. Notice that partition $[4][2][1]$ as it uses 3 bit planes, provides a large embedding capacity, which allowed us to set high heterogeneity thresholds, and obtain close MSE and PSNR values to the other partitions.

6 CONCLUSIONS

Comparing to classical code based steganography protocols, our scheme based on linear error-block codes increases the number of exploitable bits in a given cover. Specifically, multiple bit planes in an image are exploited whilst maintaining good MSE and PSNR values. A major factor to get the maximum benefit from this scheme is the choice of the cover, as heterogeneous pixels within an image promote using multiple bit planes. Heterogeneity thresholds determine the number of bit planes to use in each pixel in order to keep good image quality parameters.

In this paper we compared the results of embedding using different types of a given code, including the classical LSB embedding which corresponds to the classical type. Forthcoming works involve finding optimal codes to use with this method. Also, by statistical studies, the list of heterogeneity thresholds is likely to be optimized.

REFERENCES

Crandall, R., 1998. Some notes on steganography. *Posted on the steganography mailing list*.

Westfeld, A., 2001. F5 - a steganographic algorithm. In *Ira S. Moskowitz, editor, Information Hiding, Lecture Notes in Computer Science, vol. 2137, pp. 289-302.* Springer Berlin Heidelberg.

Zhang W., Zhang, X., Wang, S., 2007. A double layered "plus-minus one" data embedding scheme. *IEEE Signal Processing Letters, vol. 14, no. 11, pp. 848-851.*

Zhang, X., Zhang, W., Wang, S., 2007. Efficient double-layered steganographic embedding. *Electronics Letters, vol. 43, no. 8, pp. 482-483.*

Liao, X., Wen, Q. Y., 2008. Embedding in two least significant bits with wet paper coding. In *Computer Science and Software Engineering, vol. 3, pp. 555-558.* IEEE Computer Society.

Dariti, R., Souidi, E. M., 2011. Improving Code-Based Steganography with Linear Error-Block Codes. In *Digital Information Processing and Communications, CCIS, vol. 189, pp. 310-321*, Springer Berlin Heidelberg.

Feng, K., Xu, L., Hickernell, F. J., 2006. Linear error-block codes. *Finite Fields and Their Applications, vol. 12, no. 4, pp. 638-652.*

Tan, J., Luo, X., Cheng, Q., 2003. A lossless data embedding scheme for medical in application of e-diagnosis. In *Proceedings of the 25th Annual International Conference of the IEEE EMBS Cancun.*

Dariti, R., Souidi, E. M., 2013. Packing and Covering Radii of Linear Error-Block Codes. *International Journal of Mathematical, Computational Science and Engineering, vol. 7, no. 4, pp. 14-18.* World Academy of Science, Engineering and Technology.