Efficient Wet Paper Codes

[a]Jessica Fridrich, [a]Miroslav Goljan, and [b]David Soukal

[a]Dept. of Electrical and Computer Engineering, [b]Dept. of Computer Science
SUNY Binghamton, Binghamton, NY 13902-6000, USA
{fridrich,mgoljan,dsoukal1}@binghamton.edu

Abstract. Wet paper codes were proposed as a tool for constructing steganographic schemes with an arbitrary selection channel that is not shared between the sender and the recipient. In this paper, we describe new approaches to wet paper codes that enjoy low computational complexity and improved embedding efficiency (number of message bits per embedding change). Some applications of wet paper codes to steganography and data embedding in binary images are discussed.

1 Introduction

The placement of embedding changes in the cover object is called the selection channel [1]. This channel is often constructed from a secret shared between the sender and the recipient (e.g., pseudo-random straddling [2]) and may also depend on the cover object itself (adaptive embedding [3]). In general, it is in the interest of both communicating parties not to reveal any information or as little as possible about the embedding changes as this knowledge can help an attacker [4]. Since the sender's main objective is to minimize the detectability of the hidden data, he may construct the selection channel using the knowledge of the cover and any other available side information, such as a high-resolution (or unquantized) version of the cover [5]. Another possibility is to determine the best selection channel by iteratively running known steganalysis algorithms on the stego object. An obvious problem here is that the recipient may not be able to determine the same selection channel and read the message because he does not have access to the cover object or any side information.

The non-shared selection channel in steganography has been called "writing on wet paper" [5–7]. To explain the metaphor, imagine that the cover object X is an image that was exposed to rain and the sender can only slightly modify the dry spots of X (the selection channel) but not the wet spots. During transmission, the stego image Y dries out and thus the recipient does not know which pixels the sender used (the recipient has no information about the dry pixels). Codes for writing on wet paper that are suitable for steganographic applications (in the sense explained below) are called wet paper codes (WPCs).

The problem of non-shared selection channels in steganography is equivalent to "writing in memory with defective cells" introduced by Tsybakov et al. [8]. A memory contains n cells out of which $n–k$ cells are permanently stuck at either 0 or 1. The

writing device knows the locations and status of the stuck cells. The task is to write as many bits as possible into the memory (up to k) so that the reading device, that does not have any information about the stuck cells, can correctly read the data. Clearly, writing on wet paper is formally equivalent to writing in memories with stuck cell (stuck cells = wet pixels).

The defective memory is a special case of the Gelfand-Pinsker channel with informed sender [9]. The Shannon capacity of defective memory with $n–k$ stuck cells is asymptotically k/n per cell, a fact that is also easily established using random binning [10]. A generalized version of this channel that allows for randomly flipped cells in addition to stuck cells was studied by Heegard et al. [11,12] who proposed partitioned linear block codes, later recognized as instances of nested linear codes [10], and proved that these codes achieve Shannon capacity. However, in passive warden steganography, which is the subject of this paper, we will only need codes for the noise-free case.

For memory cells drawn from an alphabet of q symbols, maximum distance separable (MDS) codes, such as Reed-Solomon codes, can be used to construct a partitioned linear code achieving the channel capacity [10]. Each coset of a $[n, n–k, k+1]$ linear MDS code contains all symbol patterns of any $n–k$ stuck cells. Since this code contains q^k cosets, they can be indexed with all possible messages consisting of k symbols. One can then communicate k message symbols by first selecting an appropriate coset and finding in this coset a word with the same pattern of stuck $n–k$ cells as in the memory. Since this word is compatible with the memory defects, it can be written to the memory. The k message symbols are extracted from the index of the coset to which the word belongs. This approach, however, would be inefficient for our application. By grouping bits into q-ary symbols, the number of stuck *symbols* could drastically increase when the number of stuck *bits* is not small, which is often the case in steganographic applications.

There are three main differences in requirements between coding for defective memory and coding for wet paper steganography. First, the number of wet pixels can be quite large (e.g., 90% or more). Second, the number of wet pixels varies significantly with the stego method and for different instances of the cover object. This makes it difficult to assume an upper bound on the rate $r = k/n$ without sacrificing embedding capacity. Third, fortunately, steganographic applications are often run off line and do not require real time performance. It is quite acceptable to spend 2 seconds to embed a 10000-bit payload, but it is not acceptable to spend this time writing data into memory.

With these differences in mind, in [5,6] the authors proposed variable-rate random linear codes and showed that these codes asymptotically (and quickly) reach the channel capacity. They also described a practical implementation using Gaussian elimination on disjoint pseudo-random subsets of fixed size. We briefly summarize this approach to WPCs in Section 2. In the first method of this paper in Section 3, we follow the same approach but propose a different realization by imposing certain stochastic structure on the columns of the parity check matrix to be able to utilize the apparatus of LT codes [13]. This approach offers greatly simplified implementation, lower computational complexity, and improved embedding efficiency. In Section 4, we apply the method of Section 2 to very small blocks with a goal to further improve

the embedding efficiency for short messages in a manner somewhat similar to matrix embedding [15]. A few applications of WPCs in steganography and fragile watermarking are discussed in Section 5. The paper is summarized in Section 6.

2 Random Linear Codes for Writing on Wet Paper

Let us assume that the cover object X consists of n elements $\{x_i\}_{i=1}^{n}$, $x_i \in J$, where J is the range of discrete values for x_i. For example, for an 8-bit grayscale image represented in the spatial domain, $J = \{0, 1, \ldots, 255\}$ and n is the number of pixels in X. The sender selects k *changeable elements* x_j, $j \in C \subset \{1, 2, \ldots, n\}$, $|C|=k$, which is the selection channel. The changeable elements may be used and modified independently from each other by the sender to communicate a secret message to the recipient, while the remaining elements are not modified during embedding.

It is further assumed that the sender and the recipient agree on a public symbol function S, which is a mapping $S: J \to \mathbb{F}$, where $\mathbb{F}$ is a finite field of q symbols. Although we do not consider it in this paper, S could in principle depend on the element position in X and a secret stego key shared by the sender and the recipient. For simplicity, the reader can assume that $\mathbb{F}$ is the Galois Field GF(2) and $S(x)$ the LSB of x (Least Significant Bit).

During embedding, the sender either leaves the changeable elements x_j, $j \in C$, unmodified or replaces x_j with some element y_j to modify its symbol from $S(x_j)$ to $S(y_j)$. The vector of cover object symbols $\boldsymbol{b_x} = (S(x_1), \ldots, S(x_n))^{\mathsf{T}}$ changes to $\boldsymbol{b_y} = (S(y_1), \ldots, S(y_n))^{\mathsf{T}}$, where "T" denotes transposition. To communicate m symbols $\boldsymbol{s} = (s_1, \ldots, s_m)^{\mathsf{T}}$, $s_i \in \mathbb{F}$, the sender modifies the changeable elements x_j, $j \in C$, so that

$$\boldsymbol{D}\boldsymbol{b_y} = \boldsymbol{s}, \tag{1}$$

where $\boldsymbol{D}$ is an $m \times n$ matrix with elements from $\mathbb{F}$ shared by the sender and the recipient. Thus, similar to the coset coding approach by Heegard [11], the recipient reads the message as the syndrome of the received symbol vector $\boldsymbol{b_y}$ with the parity check matrix $\boldsymbol{D}$. Heegard chose $\boldsymbol{D}$ to guarantee that (1) has a solution for any pattern of $n–k$ stuck cells. In [5,6], the authors showed that the high volatility of k among steganographic schemes and over different covers can be well handled by randomizing Heegard's approach and choosing $\boldsymbol{D}$ as a pseudo-random $m \times n$ matrix $\boldsymbol{D}$ generated from a stego key.

To study the solvability of (1) for pseudo-random matrices $\boldsymbol{D}$, (1) is rewritten to

$$\boldsymbol{D}\boldsymbol{v} = \boldsymbol{s} - \boldsymbol{D}\boldsymbol{b_x} \tag{2}$$

using the variable $\boldsymbol{v} = \boldsymbol{b_y} – \boldsymbol{b_x}$ with non-zero elements corresponding to the symbols the sender must change to satisfy (1). In (2), there are k unknowns v_j, $j \in C$, while the remaining $n – k$ values v_i, $i \notin C$, are zeros. Thus, on the left hand side, the sender can remove from $\boldsymbol{D}$ all $n – k$ columns i, $i \notin C$, and also remove from $\boldsymbol{v}$ all $n – k$ elements v_i with $i \notin C$. Keeping the same symbol for $\boldsymbol{v}$, (2) now becomes

$$\boldsymbol{Hv} = \boldsymbol{z}, \tag{3}$$

where $\boldsymbol{H}$ is an $m \times k$ matrix consisting of those columns of $\boldsymbol{D}$ corresponding to indices C, $\boldsymbol{v}$ is an unknown $k \times 1$ vector, and $\boldsymbol{z} = \boldsymbol{s} - \boldsymbol{D}\boldsymbol{b}_x$ is the $k \times 1$ right hand side. Thus, the sender needs to solve a system of m linear equations with k unknowns in $\mathbb{F}$. The probability that (1) will have a solution for an arbitrary message $\boldsymbol{s}$ is equal to the probability that $\mathrm{rank}(\boldsymbol{D})=m$. The rank of random rectangular matrices over finite fields was studied in [14]. In particular, the probability $P(\mathrm{rank}(\boldsymbol{D})=m) = 1-O(q^{m-k})$ with decreasing m, $m < k$, k fixed.

Let us assume that the sender always tries to embed as many symbols as possible by adding rows to $\boldsymbol{D}$ while (3) still has a solution. It can be shown [6] that for random binary matrices whose elements are iid realizations of a random variable that is uniformly distributed in $\{0,1\}$, the average maximum message length $m_{\max}$ that can be communicated in this manner is

$$m_{\max} = k + O(2^{-k/4}) \tag{4}$$

as k goes to infinity, $k < n$. A similar result can be established in the same manner for a finite field $\mathbb{F}$ with q symbols. Thus, this variable-rate random linear code asymptotically (and quickly) reaches the Shannon capacity of our channel.

The main complexity of this communication is on the sender's side, who needs to solve m linear equations for k unknowns in $\mathbb{F}$. Assuming that the maximal length message $m = k$ is sent, the complexity of Gaussian elimination for (3) is $O(k^3)$, which would lead to impractical performance for large payloads, such as $k > 10^5$. In [5], the authors proposed to divide the cover object into n/n_B disjoint random subsets (determined from the shared stego key) of a fixed, predetermined size n_B and then perform the embedding for each subset separately. The complexity of embedding is proportional to $n/n_B(kn_B/n)^3 = nr^3n_B^2$, where $r = k/n$ is the rate, and is thus linear in the number of cover object elements, albeit with a large constant.

By imposing a special stochastic structure on the columns of $\boldsymbol{D}$, we show in the next section that it is possible to use the LT process to solve (3) in a much more efficient manner with a simpler implementation that fits well the requirements for steganographic applications formulated in the introduction.

3 Realization of Wet Paper Codes Using the LT Process

3.1 LT Codes

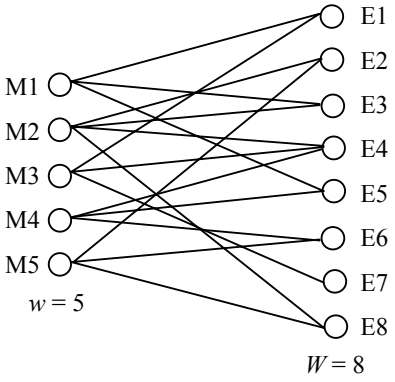
In this section, we briefly review LT codes and their properties relevant for our application, referring the reader to [13] for more details. LT codes are universal erasure codes with low encoding and decoding complexity that asymptotically approach the Shannon capacity of the erasure channel. For simplicity, we only use binary symbols noting that the codes can work without any modification with l-bit symbols. The best way to describe the encoding process is using a bipartite graph (see an example in

Fig. 1) with w message bits on the left and W encoding bits on the right. Each encoding bit is obtained as an XOR of approximately $O(\ln(w/\delta))$ randomly selected message bits that are connected to it in the graph. The graph is generated randomly so that the degrees of encoding nodes follow so-called robust soliton distribution (RSD). The probability that an encoding node has degree i, is $(\rho_i + \tau_i)/\beta$, where

$$\rho_i = \begin{cases} \frac{1}{w} & i = 1 \\ \frac{1}{i(i-1)} & i = 2,...,w \end{cases}, \tau_i = \begin{cases} R/(iw) & i = 1,...,w/R-1 \\ R\ln(R/\delta)/w & i = w/R \\ 0 & i = w/R+1,...,w \end{cases}, \beta = \sum_{i=1}^{w} \rho_i + \tau_i , \quad (5)$$

and $R = c\ln(w/\delta)\sqrt{w}$ for some suitably chosen constants δ and c. It is possible to uniquely determine all w message bits with probability better than $1-\delta$ from an arbitrary set of W encoding bits as long as

$$W > \beta w = w + O(\sqrt{w}\ln^2(w/\delta)). \quad (6)$$



$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 1. Left: Bipartite graph with 5 message symbols and 8 encoding symbols. Right: Its bi-adjacency matrix.

The encoding bits can also be obtained from message bits using matrix multiplication in GF(2) with the bi-adjacency binary matrix $\boldsymbol{A}$ (Fig. 1). The decoding can be obviously done by solving a system of W linear equations with w unknowns – the message bits. The RSD allows solving the linear system by repeating the following simple operation (the LT process):

> *Find an encoding bit that has only one edge* (*encoding bit* E7 *in Fig. 1*). *Its associated message bit* (M3) *must be equal to this encoding bit. As the message bit is now known, we can XOR it with all encoding bits that are connected to it* (E1 *and* E4) *and remove it and all its edges from the graph. In doing so, new encoding*

nodes of degree one (E1) *may be created. This process is repeated till all message bits are recovered.*

The decoding process fails if, at some point, there are no encoding bits of degree 1, while there are still some undetermined message bits. The RSD was derived so that the probability of failure of the LT process to recover all message bits is smaller than δ. The decoding requires on average $O(w \ln(w/\delta))$ operations.

3.2 Matrix LT Process

We can consider the LT process as a method for a fast solution of an *over-determined* system of equations $\boldsymbol{Ax} = \boldsymbol{y}$ with a random matrix $\boldsymbol{A}$ for which the Hamming weights of its rows follow the RSD. However, we cannot use it directly to solve (3) because (3) is *under-determined* and we are seeking one solution, possibly out of many solutions. In addition, because $\boldsymbol{H}$ was obtained from $\boldsymbol{D}$ by removing columns, $\boldsymbol{H}$ inherits the distribution of Hamming weights of columns from $\boldsymbol{D}$ but not the distribution of its rows. However, as explained in detail below, the LT process can be used to quickly bring $\boldsymbol{H}$ to the *upper* triangular form simply by permuting its rows and columns. Once in this form, (3) is solved using a back substitution.

The LT process on the bipartite graph induces the following row/column swapping process on its bi-adjacency matrix $\boldsymbol{A}$. For an n-dimensional binary vector $\boldsymbol{r}$, let $\mathrm{w}_j(\boldsymbol{r})$ denote the Hamming weight of $(r_j, \ldots, r_n)$ (e.g., $\mathrm{w}_1(\boldsymbol{r}) \equiv \mathrm{w}(\boldsymbol{r})$ is the usual Hamming weight of $\boldsymbol{r}$). We first find a row $\boldsymbol{r}$ in $\boldsymbol{A}$ with $\mathrm{w}_1(\boldsymbol{r}) = 1$ (say, the 1 is in the j_1-th column) and exchange it with the first row. Then, we exchange the 1st and the j_1-th unknowns (swapping the 1st and j_1-th columns). At this point in the LT process, the value of the unknown No. 1 is determined from the first equation. In the matrix process, however, we do not evaluate the unknowns because we are only interested in bringing $\boldsymbol{A}$ to a *lower* triangular form by permuting its rows and columns. Continuing the process, we search for another row $\boldsymbol{r}$ with $\mathrm{w}_2(\boldsymbol{r}) = 1$ (say, the 1 is in the j_2-th column). If the LT process proceeds successfully, we must be able to do so. We swap this row with the second row and swap the 2nd and j_2-th columns. We continue in this way, now looking for a row $\boldsymbol{r}$ with $\mathrm{w}_3(\boldsymbol{r}) = 1$, etc. At the end of this process, the permuted matrix $\boldsymbol{A}$ will be lower diagonal with ones on its main diagonal.

Returning to the WPC of Section 2, we need to solve the system $\boldsymbol{Hv} = \boldsymbol{z}$ with m equations for k unknowns, $m < k$. By applying the above process of row and column permutations to $\boldsymbol{H}^{\mathrm{T}}$, we bring $\boldsymbol{H}$ to the form $[\boldsymbol{U}, \boldsymbol{H}']$, where $\boldsymbol{U}$ is a square $m \times m$ upper triangular matrix with ones on its main diagonal and $\boldsymbol{H}'$ is a binary $m \times (k-m)$ matrix. We can work directly with $\boldsymbol{H}$ if we replace in the algorithm above the word 'row' with 'column' and vice versa[1]. In order for this to work, however, the Hamming weights of *columns* of $\boldsymbol{H}$ must follow the RSD and the message length m must satisfy (from (6))

$$k > \beta m = m + O(\sqrt{m} \ln^2(m/\delta)) . \tag{7}$$

[1] To distinguish this process, which pertains to a binary matrix, from the original LT process designed for bi-partite graphs, we call it the "matrix LT process".

This means that there is a small capacity loss of $O(\sqrt{m}\ln^2(m/\delta))$ in exchange for solving (3) quickly using the matrix LT process. This loss depends on the public parameters c and δ. Since the bounds in Luby's analysis are not tight, we experimented with a larger range for δ, ignoring its probabilistic interpretation. We discovered that it was advantageous to set δ to a much larger number (e.g., $\delta = 5$) and, if necessary, repeat the encoding process with a slightly larger matrix $\boldsymbol{D}$ till a successful pass through the LT process is obtained. For $c = 0.1$, the capacity loss was about 10% ($\beta = 1.1$) of k for k=1500 with probability of successful encoding about 50%. This probability increases and capacity loss decreases with increasing k (see Table 1).

To assess the encoding and decoding complexity, let us assume that the maximal length message is sent, $m \approx k/\beta$. The density of 1's in $\boldsymbol{D}$ (and thus in $\boldsymbol{H}$) is $O(\ln(k/\delta)/k)$. Therefore, the encoding complexity of the WPC implemented using the LT process is $O(n\ln(k/\delta) + k\ln(k/\delta)) = O(n\ln(k/\delta))$. The first term arises from evaluating the product $\boldsymbol{Db}_x$, while the second term is the complexity of the LT process. This is a significant savings compared to solving (3) using Gaussian elimination. The decoding complexity is $O(n\ln(k/\delta))$, which corresponds to evaluating the product $\boldsymbol{Db}_y$.

Table 1. Running time (in seconds) for solving $k\times k$ and $k\times\beta k$ linear systems using Gaussian elimination and matrix LT process ($c = 0.1$, $\delta = 5$); P is the probability of a successful pass.

k	Gauss	LT	β	P
1000	0.023	0.008	1.098	43%
10000	17.4	0.177	1.062	75%
30000	302	0.705	1.047	82%
100000	9320	3.10	1.033	90%

The performance comparison between solving (3) using Gaussian elimination and the matrix LT process is shown in Table 1. The steeply increasing complexity of Gaussian elimination necessitates dividing the cover object into subsets as in [5]. The LT process, however, enables solving (3) for the whole object at once, which greatly simplifies implementation and decreases computational complexity at the same time. In addition, as will be seen in Section 4, the matrix LT process can modified to improve the embedding efficiency.

3.3 Communicating the Message Length

Note that for the matrix LT process, the Hamming weights of columns of $\boldsymbol{H}$ (and thus $\boldsymbol{D}$) must follow the RSD that depends on m, which is unavailable to the decoder. Below, we show a simple solution to this problem, although other alternatives exist.

Let us assume that the parameter m can be encoded using h bits (in practice, h~20 should be sufficient). Using the stego key, the sender divides the cover X into two pseudo-random disjoint subsets X_h and $X–X_h$ and communicates h bits using elements from X_h and the main message using elements from $X–X_h$. We must make sure that X_h

will contain at least h changeable elements, which can be arranged for by requesting that $|X_h|$ be a few percent larger than h/r_{min}, where r_{min} is the minimal value of the rate $r=k/n$ that can be typically encountered (this depends on the specifics of the steganographic scheme and properties of the covers). Then, using the stego key the sender generates a pseudo-random $h\times|X_h|$ binary matrix $\boldsymbol{D}_h$ with density of 1's equal to ½. The sender embeds h bits in X_h by solving the WPC equations (3) with matrix $\boldsymbol{D}_h$ using a simple Gaussian elimination, which will be fast because $\boldsymbol{D}_h$ has a small number of rows. The message bits are hidden in $X–X_h$ using the matrix LT process with matrix $\boldsymbol{D}$ generated from the stego key using the parameter m.

The decoder first uses his stego key (and the knowledge of h and r_{min}) to determine the subset X_h and the matrix $\boldsymbol{D}_h$. Then, the decoder extracts m (h bits) as the syndrome (1) with matrix $\boldsymbol{D}_h$ and the symbol vector obtained from X_h. Knowing m, the decoder now generates $\boldsymbol{D}$ and extracts the message bits as a syndrome (1) with matrix $\boldsymbol{D}$ and the symbol vector obtained from $X–X_h$.

4 Embedding Efficiency

The number of embedding changes in the cover object influences the detectability of hidden data in a major manner. The smaller the number of changes, the smaller the chance that any statistics used by an attacker will be disrupted enough to mount a successful attack. Thus, schemes with a higher embedding efficiency (number of random message bits embedded per embedding change) are less likely to be successfully attacked than schemes with a lower embedding efficiency.

The first general methodology to improving embedding efficiency of data hiding schemes was described by Crandall [15] who proposed an approach using covering codes (Matrix Embedding). This idea was later made popular by Westfeld in his F5 algorithm [2]. A formal equivalence between embedding schemes and covering codes is due to Galand and Kabatiansky [16]. From their work, we know that the number of messages that can be communicated by making at most l changes in a binary vector of length k is bounded from above by $2^{k\mathrm{h}(l/k)}$, where $\mathrm{h}(x) = -x\log_2(x) - (1-x)\log_2(1-x)$, assuming $k\to\infty$ and $l/k = const.<1/2$. Additionally, they pointed out that embedding schemes based on random linear coverings asymptotically achieve this bound, which means that the bound is tight.

Using this result, we can now derive an upper bound on the embedding efficiency, which is defined as the ratio between the payload length m and the number of embedding changes l. Let $\boldsymbol{H}$ be the $m\times k$ binary matrix from (3). Assuming the Hamming weight of $\boldsymbol{v}$ is at most l (i.e., we perform up to l embedding changes out of k possible changes), we have for the payload length m, $m \le k\,\mathrm{h}(l/k)$, or

$$\frac{m}{l} \le \frac{m/k}{\mathrm{h}^{-1}(m/k)}, \tag{8}$$

where h^{-1} is the inverse of h on [0,1/2].

4.1 Block Minimal Method

In general, the problem of finding a solution to (3) with the minimum Hamming weight is an NP complete problem. For a very small m, however, it is possible to find the minimal Hamming weight solution quickly using brute force. This suggests applying the method of Section 2 on small blocks. The sender and receiver agree on a small integer p (e.g., $p<20$) and using the stego key divide the cover object into $n_B = m/p$ disjoint random blocks of cardinality $n/n_B = pn/m$. Each block will contain on average $rpn/m = pk/m$ changeable elements (for simplicity we assume the quantities above are all integers). The sender will use a random binary $p\times pn/m$ matrix $\boldsymbol{D}_B$ for embedding up to p bits in each block as follows.

Let $\boldsymbol{H}_B$ be a submatrix of $\boldsymbol{D}_B$ with columns corresponding to changeable elements and C_1 be the set of *unique* columns of $\boldsymbol{H}_B$. Note that $\boldsymbol{H}_B$ will in general be different for different blocks. For two sets C, C' of binary vectors from $\{0,1\}^p$, we define $C \oplus C' = \{x \in \{0,1\}^p | \, x = c + c', c \in C, c' \in C'\}$, arithmetic in GF(2). Messages $\boldsymbol{s} \in C_1$ can be communicated using one change, messages $\boldsymbol{s} \in C_2 = C_1 \oplus C_1 - C_1$ using two changes, in general, messages $\boldsymbol{s} \in C_i = C_{i-1} \oplus C_1 - (C_1 \cup \ldots \cup C_{i-1})$ using i changes[2]. If $\bigcup_{i=1}^{p} C_i = \{0,1\}^p$, which happens if and only if $\text{rank}(\boldsymbol{H}_B)=p$, the system $\boldsymbol{H}_B\boldsymbol{v} = \boldsymbol{z}$ will have a solution for all $\boldsymbol{z} \in \{0,1\}^p$. The probability of this is $1-O(2^{p(1-k/m)})$, which quickly approaches 1 with decreasing message length m (for p and k fixed) or with increasing p (for m and k fixed) because $m < k$.

The average number of changes, l_p, in each block can be obtained as

$$l_p(r,k,m) = E\{|\, \boldsymbol{v} \,|\} = 2^{-p} \sum_{i=1}^{p} iE\{|\, C_i \,|\}\,, \tag{9}$$

where the expected value is taken over random messages and $p\times pn/m$ matrices $\boldsymbol{D}_B$ from which columns are selected with probability r (to obtain $\boldsymbol{H}_B$). Since the number of columns in $\boldsymbol{H}_B$ is approximately pk/m, l_p is mostly a function of the ratio k/m rather than the specific values of r, k, or m. From Fig. 2, we see that the embedding efficiency p/l_p increases with shorter messages (larger k/m) for a fixed p and it also increases in a curious non-monotonic manner with increasing p for k/m fixed. In general, one should use as large p as possible. In theory, when $p=m$, we would obtain the optimal solution with the smallest Hamming weight. However, a practical limit on the largest usable p is imposed by a rapidly increasing computational complexity. The most expensive operation in determining the set C_i is the term $C_{i-1} \oplus C_1$, which may require up to $p2^{2p-2}$ bit XOR operations. To obtain reasonable running times, we recommend $p \le 18$ (see Table 2 generated on a PC equipped with a 3.5GHz Pentium IV). We also note that the WPC based on random linear codes (Section 2) or LT process (Section3) achieves embedding efficiency of only about 2.

[2] The symbol '–' stands for the set difference.

Table 2. Embedding time (in seconds) for $n=10^6$, $k=5\times10^4$ for Block Minimal method.

k/m	2	3	4	5	6	7	8	9	10
p=16	2.86	1.72	1.24	0.92	0.76	0.64	0.52	0.42	0.43
p=17	8.80	5.38	3.78	3.01	2.50	2.18	1.60	1.53	1.45
p=18	27.92	17.53	12.41	9.67	7.87	7.54	5.87	5.66	4.83

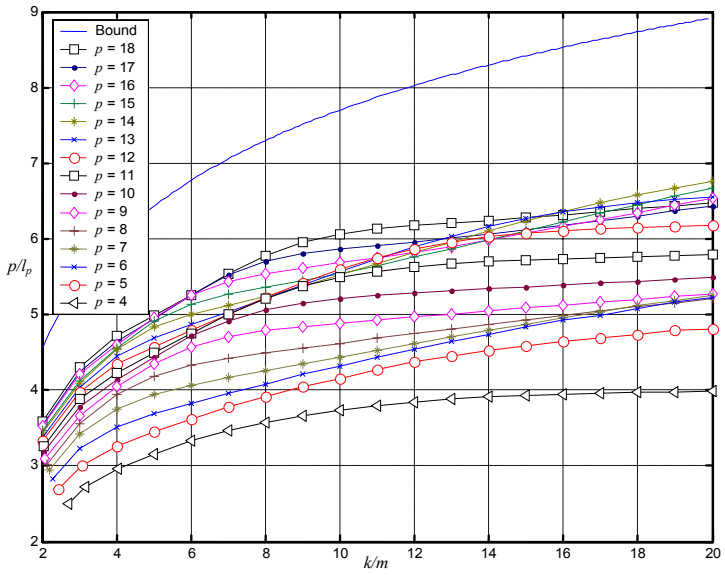


Fig. 2. Embedding efficiency for Block Minimal method as a function of k/m for various p and r=1/20. The upper bound is given by (8).

For small k/m, the probability that rank($\boldsymbol{H}_B$)<p may become large enough to encounter a failure to embed in certain blocks. For example, for $p = 18$ and $k/m = 2$, Prob(rank($\boldsymbol{H}_B$)<18) ≈ 0.003. We note that this problem is not an issue for k/m>3 as this probability is very small. The encoder needs to communicate the number of bits embedded in each block. Let us assume k, n, and m are fixed. For the i-th block, let p_i be the largest integer for which the first p_i rows of $\boldsymbol{H}_B$ form a matrix of rank p_i. Furthermore, let $f(q)$, $q = p, p-1, \ldots, 0$, be the probability distribution of p_i over the blocks and random matrices $\boldsymbol{H}_B$. The information necessary to communicate p_i is $H(f)$, the entropy[3] of f. The average number of bits that can be encoded per block is thus $E(f)-H(f) \le p$ because $E(f) \le p$. Thus, the pure payload $m' = m(E(f)-H(f))/p$ that can be embedded is slightly smaller than m. From Table 3, we see that this loss is negligible for $k/m \ge 2$. It also limits the Block Minimal method to cases with $k/m > 1.4$ (i.e., the

[3] In practice, the compressed bit-stream will be slightly larger than $H(f)$. Since f is not known to the decoder beforehand, adaptive coders, such as adaptive arithmetic coder, should be used.

method cannot be used when the payload is longer than roughly 70% of the maximal embeddable message).

Table 3. Capacity loss for n=10^6, k=50000, p=18, for Block Minimal method.

k/m	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2	3
k/m'	1.42	1.42	1.44	1.49	1.56	1.63	1.72	1.81	1.90	2.006	3.000

From the practical point of view, the sequence p_i should be compressed and then embedded, for example, one bit per block, as the first bit in each block. The decoder first extracts p bits from each block, decompresses the bit sequence formed by the first bits from each block, reads p_i for all blocks, and then discards p–p_i bits from the end of each block message chunk.

In Fig. 3, we show the embedding efficiency p/l_p as a function of k/m for p=18 and compare the performance to other approaches. The graph takes into account the capacity loss discussed in the paragraph above.

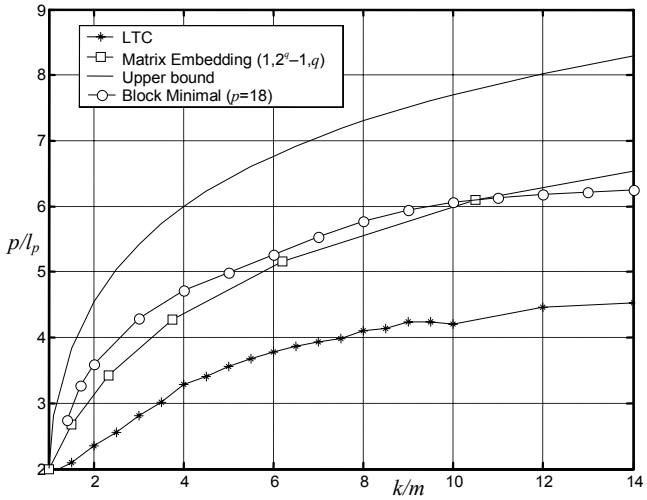


Fig. 3. Embedding efficiency as a function of the ratio k/m for n = 10^6, k = 5×10^4. The curves show the upper bound (8), Matrix Embedding for q = 1, …, 6 (assuming all n elements can be changed or k = n), Block Minimal, and the LTC algorithm (see Section 4.2).

We note that this approach can be considered as some stochastic form of matrix embedding. In matrix embedding, *all* cover elements can be modified, which means the encoder can for example choose $\boldsymbol{D}_B$ to be the $p\times(2^p-1)$ parity check matrix of a binary Hamming code [2] and always have at most one embedding change (C_1=$\{0,1\}^p$) to embed p bits in 2^p–1 pixels – matrix embedding (1, 2^p–1, p). In our application, however, we work with $\boldsymbol{H}_B$, which is obtained as a submatrix of $\boldsymbol{D}_B$ defined by the selection channel. Thus, we cannot guarantee that C_1=$\{0,1\}^p$ and have to allow more than one embedding change.

4.2 Improving Embedding Efficiency with LT Process (LTC algorithm)

In this section, we present a few simple modification of the matrix LT process with the goal to decrease the Hamming weight of the obtained solution *without significantly increasing the computational complexity*. Recalling Section 3.2, if the columns of $\boldsymbol{D}$ follow the RSD, the system $\boldsymbol{Hv} = \boldsymbol{z}$ can be brought to the form $[\boldsymbol{U}, \boldsymbol{H}']\tilde{\boldsymbol{v}} = \tilde{\boldsymbol{z}}$, where $\boldsymbol{U}$ is an $m \times m$ upper triangular matrix, $\boldsymbol{H}'$ is a binary $m \times (k-m)$ matrix, and $\tilde{\boldsymbol{v}}$, $\tilde{\boldsymbol{z}}$ are correspondingly permuted vectors $\boldsymbol{v}$ and $\boldsymbol{z}$. By setting $\tilde{v}_i = 0$ for $i > m$ and solving the rest using back substitution, we have $\mathrm{w}(\tilde{\boldsymbol{v}}) \approx m/2$ (assuming random message bits independent of the cover), which gives an embedding efficiency of roughly 2.

There are several avenues to be explored to improve the embedding efficiency. First, the shorter the message, the more choices will the encoder have for selecting the columns in the matrix LT process and thus "steer" the algorithm to decrease $\mathrm{w}(\tilde{\boldsymbol{v}})$. Second, the RSD was derived for the case when $k=\beta m$. It is quite possible that for shorter messages, some other distribution will be more suitable for our goal. Note that this problem does not have any equivalent for the erasure codes because there is no need to collect more than the minimal necessary number of symbols for decoding. We postpone the question of the column weight distribution to our future research and in this section only briefly investigate the first option.

Recall that in the i-th step in the matrix LT process on $\boldsymbol{H}$, we select a column $\boldsymbol{c}$ in $\boldsymbol{H}$ with $\mathrm{w}_i(\boldsymbol{c}) = 1$. The set of columns $\boldsymbol{c}$ with $\mathrm{w}_i(\boldsymbol{c}) = 1$ will be called *ripple* at the i-th step (note the difference in terminology as used in [13]). Note that if $\tilde{z}_i = 0$, for all $i > i_0$, the matrix LT process can be stopped after i_0 steps because we can set $\tilde{v}_i = 0$, $i > i_0$. Also, the smaller the index i_0, the more zeros there will be in $\tilde{\boldsymbol{v}}$. Thus, in our choice of columns from the ripple we should prefer columns $\boldsymbol{c}$, where the 1 is in row j for which $\tilde{z}_j = 1$. This way, all 1's in $\tilde{\boldsymbol{z}}$ will hopefully be "depleted" sooner, producing a smaller i_0 and thus a solution with a smaller Hamming weight. To further utilize the remaining degree of freedom in our choice of columns from the ripple, it is advantageous to prefer denser columns in steps $i<i_0$, while preferring sparser columns in steps $i \geq i_0$. An explanation of this "recipe" is skipped due to lack of space in this paper.

Because the LT process has no effect on $\mathrm{w}(\tilde{\boldsymbol{z}}) = \mathrm{w}(\boldsymbol{z})$, we must have $i_0 \geq \mathrm{w}(\boldsymbol{z})$. In the best case when $i_0 = \mathrm{w}(\boldsymbol{z})$, the expected value of $\mathrm{w}(\tilde{\boldsymbol{v}})$ will be approximately $\mathrm{w}(\boldsymbol{z})/2$ (assuming $\boldsymbol{U}$ is random). Because the expected value of $\mathrm{w}(\boldsymbol{z})$ taken over random messages is $m/2$, we cannot obtain better embedding efficiency than 4.

To further improve the efficiency, we observe that the smaller the $\mathrm{w}(\boldsymbol{z})$, the smaller the $\mathrm{w}(\tilde{\boldsymbol{v}})$. Thus, it might be possible to reduce $\mathrm{w}(\boldsymbol{z})$ before the LT process starts by adding selected columns of $\boldsymbol{H}$ to $\boldsymbol{z}$. Keeping in mind that we need to preserve the low computational complexity of the matrix LT process, we propose the following simple and intuitive preprocessing step. Before starting the matrix LT process, we search for a column $\boldsymbol{c}$ in $\boldsymbol{H}$ such that $\mathrm{w}(\boldsymbol{z}-\boldsymbol{c}) < \mathrm{w}(\boldsymbol{z}) - \log_2(m)$. If such columns exist, we choose the one leading to the smallest Hamming weight $\mathrm{w}(\boldsymbol{z}-\boldsymbol{c})$. We subtract $\boldsymbol{c}$ from $\boldsymbol{z}$, remove it from $\boldsymbol{H}$, assign 1 to the corresponding component of $\boldsymbol{v}$, and search for another column. This is repeated until no such column can be found. The term

$\log_2(m)$ is our ad hoc choice that gave us a good compromise between a small increase in computational complexity and performance improvement. Note that the embedding rate can now grow without a limit for increasing k/m because the probability of finding a well-fitting column increases with increasing k/m.

The embedding efficiency of the LTC algorithm, that includes some other minor improvements not described in this paper, is shown in Fig. 3. Its computational complexity is roughly comparable to the matrix LT process of Section 3.

We note that there are other simple measures that can be adopted to further improve the performance of the matrix LT process. For example, our preliminary experiments indicate that allowing occasional row adding during the matrix LT process has the potential to improve the embedding efficiency as well as significantly increase the probability of a successful pass. This issue is part of our future research.

We close this section with an observation that in steganography there is another possibility to minimize the impact of embedding changes different from increasing the embedding efficiency. Depending on the selection criteria applied by the sender, each changeable element can be assigned a numerical value, or changeability score, that somehow captures how undetectable the modification of that element is. For example, elements in highly textured areas of the cover image may have a higher score than elements in less textured areas. For short messages of length m, one may be better off (depending on the score distribution) narrowing the set of changeable elements from k to those $k' = \beta m$ elements with the highest score instead of maximizing the embedding efficiency with k changeable elements, $k >> m$.

5 Applications in Steganography

Wet paper codes free the sender from having to consider the problem of communicating the selection channel to the recipient and thus they give him complete freedom in choosing the placement of embedding modifications. In adaptive steganography, for example, because the act of embedding itself modifies the cover, special care usually needs to be taken to make sure that the recipient identifies the same selection channel. WPCs not only solve this problem but also allow the sender to use selection channels that are *in principle* unavailable to the recipient and thus any attacker, such as channels determined from a high-resolution (or unquantized) version of the cover (see Perturbed Quantization Steganography [5] for more details).

Public key steganography [1] also benefits from WPCs because they enable message extraction without revealing any information about the selection channel. Thus, the matrix $\boldsymbol{D}$ can be made public to allow everybody to extract from the stego object a message potentially encrypted using an asymmetric cryptography without revealing any information about the placement of the embedding changes. Additionally, because each message bit is extracted as an XOR of many elements (e.g., $\ln(k/\delta)$ elements for the implementation using the matrix LT process), the "power of parity" [1] further helps mask the presence of secret message.

Another interesting application is the possibility to construct steganographic methods that cannot be subjected to brute force stego key searches of the type [17] because the embedding can contain an element of true randomness.

Lastly, we mention data hiding in binary images. In [18], the sender first identifies the set of "flippable" pixels that can be modified for embedding. Because the act of embedding itself modifies the pixel "flippability" status, the set of flippable pixels can not be shared with the recipient. To solve this problem, Wu proposed block embedding combined with random shuffling. The block embedding however, leaves most of the flippable pixels unused, leaving only a fraction of the embedding capacity for the payload. Because this situation exactly corresponds to writing on wet paper, the capacity of this data embedding method can be dramatically improved using WPCs [19]. In this application, the WPCs with improved embedding efficiency (Section 4) are particularly important as they help decrease the visual impact of embedding.

6 Summary

Wet paper codes enable steganography with non-shared (arbitrary) selection channels. In this paper, we describe a new approach (the matrix LT process) to wet paper codes using the apparatus developed for irregular low-density parity check erasure codes called LT codes. The new approach offers greatly simplified implementation and a substantially decreased computational complexity. We also present a few simple modifications of the matrix LT process to improve the embedding efficiency while preserving its low computational complexity. Additionally, we introduce another, different, approach to wet paper codes called Block Minimal embedding that provides significantly improved embedding efficiency and also enjoys low computational complexity suitable for steganographic applications. Finally, we briefly discuss a few applications to steganography and data embedding.

Acknowledgements

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under the research grants number FA8750-04-1-0112 and F30602-02-2-0093. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government. Special thanks belong to Petr Lisoněk for directing our attention to LT codes.

References

1. Anderson, R.J. and Petitcolas, F.A.P.: "On the Limits of Steganography". IEEE Journal of Selected Areas in Communications. Special Issue on Copyright and Privacy Protection, vol. **16(**4) (1998) 474–481

2. Westfeld, A. “High Capacity Despite Better Steganalysis (F5–A Steganographic Algorithm)”. In: Moskowitz, I.S. (ed.): Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, vol. 2137. Springer-Verlag, Berlin Heidelberg New York (2001) 289–302
3. Karahan, M., Topkara, U., Atallah, M., Taskiran, C., Lin, E., and Delp, E.: “A Hierarchical Protocol for Increasing the Stealthiness of Steganographic Methods”. Proc. ACM Multimedia and Security Workshop. Magdeburg Germany (2004) 16–24
4. Westfeld, A. and Böhme, R.: “Exploiting Preserved Statistics for Steganalysis”. In: Fridrich, J. (ed.): Information Hiding. 6th International Workshop. Lecture Notes in Computer Science, vol. 3200. Springer-Verlag, Berlin Heidelberg New York (2004) 67–81
5. Fridrich, J., Goljan, M., and Soukal, D.: “Perturbed Quantization Steganography with Wet Paper Codes”. Proc. ACM Multimedia and Security Workshop. Magdeburg Germany (2004) 4–15
6. Fridrich, J., Goljan, M., Lisoněk, P., and Soukal, D.: “Writing on Wet Paper”, (journal version) to appear in IEEE Trans. on Sig. Proc. Special Issue on Media Security (2005)
7. Fridrich, J., Goljan, M., Lisoněk, P., and Soukal, D.: “Writing on Wet Paper”, Proc. SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII. San Jose (2005) 328–340
8. Kuznetsov, A.V. and Tsybakov, B.S.: “Coding in a Memory with Defective Cells”. Probl. Inform. Transmission, vol. **10**. (1974) 132–138
9. Gelfand, S.I. and Pinsker, M.S.: “Coding for Channel with Random Parameters”. Probl. Pered. Inform. (Probl. Inform. Transm.), vol. **9**(1). (1980) 19–31
10. Zamir, R., Shamai, S., and Erez, U.: “Nested Linear/Lattice Codes for Structured Multiterminal Binning”. IEEE Trans. Inf. Th., vol. **48**(6). (2002) 1250–1276
11. Heegard, C.: “Partitioned Linear Block Codes for Computer Memory with ‘Stuck-at’ Defects”. *IEEE Trans. Inf. Th.* vol. **29**. (1983) 831–842
12. Heegard, C. and El-Gamal, A.: “On the Capacity of Computer Memory with Defects”. IEEE Trans. Inf. Th., vol. **29**. (1983) 731–739
13. Luby, M.: “LT Codes”, Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science. (2002) 271–282
14. Brent, R.P., Gao, S., and Lauder, A.G.B.: “Random Krylov Spaces Over Finite Fields”. SIAM J. Discrete Math. vol. **16**(2). (2003) 276–287
15. Crandall, R.: “Some Notes on Steganography”. Posted on Steganography Mailing List. http://os.inf.tu-dresden.de/~westfeld/crandall.pdf (1998)
16. Galand, F. and Kabatiansky, G.: “Information Hiding by Coverings”. Proc. ITW2003. Paris France (2003) 151–154
17. Fridrich, J., Goljan, M., Soukal, D., and Holotyak, T.: “Forensic Steganalysis: Determining the Stego Key in Spatial Domain Steganography”. Proc. SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII. San Jose (2005) 631–642
18. Wu, M., Tang E., and Liu B., “Data Hiding in Digital Binary Image”. Proc. Conf. on Multimedia & Expo (CD version). New York (2000)
19. Wu, M., Fridrich, J., Goljan, M., and Gou, H.: “Data Hiding in Digital Binary Images: A Revisit”. Proc. SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII. San Jose (2005) 194–205