Steganography via Codes for Memory with Defective Cells

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

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

ABSTRACT

Steganography is the art of covert (undetectable) communication in which secret data is embedded in innocuous looking messages by slightly modifying them. The detectability of secret data is influenced, besides other factors, by the placement of embedding changes within the message and by the embedding efficiency—the number of bits embedded per one embedding change. In this paper, we first show that codes for memory with defective cells enable steganographic communication without sharing the placement of embedding changes. This allows construction of a new class of steganographic schemes with improved security. We then describe an efficient coding method for memory with defective cells based on the LT process that is suitable for steganographic applications. In the second part of the paper, we explore a different approach based on random linear codes on small blocks with the goal to decrease the number of embedding changes. The embedding efficiency of this approach is compared to theoretically achievable bounds.

Keywords: steganography, memory with defective cells, embedding efficiency, LT codes, random linear codes.

1. INTRODUCTION

Steganography was originally formalized by Simmons [1] as the prisoners problem. Alice and Bob are prisoners in separate cells who want to develop an escape plan. They are allowed to exchange messages, but their communication is monitored by a warden looking for suspicious activity. Alice and Bob resort to steganography and hide the details of the escape plot in innocuous looking messages. The prisoners' goal is to hide the secret data so that the warden cannot tell whether the exchanged messages are genuine or contain hidden data. In the simplest case (treated in this paper), the warden is passive in that he just observes the traffic and does not interfere with the communication.

For concreteness, we may visualize that the messages exchanged by the prisoners are digital images. Most steganographic schemes for digital images hide secret data by slightly perturbing the values representing the numerical values of individual pixel colors. For example, the secret data may be communicated in the least significant bits (LSB) of colors of selected pixels.

The main requirement of any steganographic technique is *undetectability*—the warden should not be able to distinguish between *cover objects* (original unmodified messages) and *stego objects* (cover embedded with data) with success better than random guessing given a complete knowledge of the steganographic algorithm and the source of cover objects (so called Kerckhoffs' principle). The detectability of data hidden in a stego object is mainly influenced by four factors—the choice of the cover object, the selection rule used to identify individual elements of the cover that could be modified during embedding, the type of embedding operation that modifies the cover elements, and the number of embedding changes.

The placement of embedding changes in the cover is called the selection channel. It is in the interest of both communicating parties to reveal as little as possible about the selection channel as this knowledge can help the warden [2]. Trying to to minimize the detectability of the hidden data, the sender may construct the selection channel using information that is in principle unavailable to the warden. For example, the sender may utilize a high-resolution (or unquantized) version of the cover [3]. Alternatively, the sender may determine the best selection channel by iteratively running known steganalysis algorithms on the stego object. However, an obvious problem with these approaches is that now not only the warden but also the recipient does not have access to the information that determines the selection channel and is thus unable to read the secret data.

The problem of such non-shared selection channels in steganography is equivalent to writing in memory with defective cells [4]. Again, using a grayscale digital image as an example, the set of LSBs of all pixels in the image is the array of binary cells. The LSBs of pixels carrying the secret data (the selection channel) correspond to functioning cells, while the LSBs of unused pixels are cells that are permanently stuck at either 0 or 1. The sender (the writing device) knows the locations and status of the stuck cells. The task is to write into the memory so that the reading device (the recipient), that does not have any information about the stuck cells (the selection channel), can correctly read the secret data.

The possibility to use non-shared selection channels is very empowering as it allows the sender to incorporate arbitrary side information into the embedding process and even use an element of true randomness. It is important, however, that the coding method allows the use of a shared secret between both prisoners (the stego key), otherwise the warden can always try to blindly extract secret data from every image the prisoners exchange. In Section 2, we start with a straightforward approach to memory with defective cells based on variable rate random linear codes. An efficient implementation suitable for steganographic applications based on irregular low density parity check codes (LDPC), the LT codes, is described in Section 3.

The second attribute of steganographic schemes studied in this paper is the embedding efficiency, which is the number of bits embedded per one embedding change. Assuming two steganographic methods share the same source of cover objects, the same selection channel and embedding operation, the one that introduces fewer embedding changes will be less detectable as it decreases the chance that any statistics used by the warden will be sufficiently disturbed to mount a successful steganalysis attack.

It has been established by Crandall [5] and Bierbrauer [6] and independently by Galland et al. [7] that the concept of embedding efficiency is closely related to the covering radius of codes. In particular, a linear code can be used to construct an embedding scheme whose embedding capacity is the code redundancy, while the covering radius corresponds to the maximal number of embedding changes necessary for embedding. With the goal to improve the embedding efficiency, in Section 4, we describe and analyze a simple approach to memory with defective cells using random linear codes on small blocks. Its embedding efficiency is studied in Section 5, where it is compared to theoretically achievable bounds. Results of experiments are interpreted in Section 6. Finally, the paper is concluded in Section 7.

2. STEGANOGRAPHY USING CODES FOR MEMORY WITH DEFECTIVE CELLS

The defective memory is a special case of the Gel'fand-Pinsker channel with informed sender [8]. The Shannon capacity of defective memory with $n-k$ stuck cells is asymptotically k/n per cell and per channel use, a fact that is also easily established using random binning (see, for example [9]). A generalized version of this channel that allows for randomly flipped cells in addition to stuck cells was studied by Heegard et al. [10,11] who proposed partitioned linear block codes, later recognized as instances of nested linear codes [9], and proved that these codes achieve Shannon capacity. 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 can be used to construct a partitioned linear code achieving the channel capacity [9]. This approach, however, would be inefficient for binary cells. By grouping bits into q-ary symbols, the number of stuck symbols could drastically increase when the number of stuck bits is large, which is often the case in steganographic applications.

There are several differences between coding for defective memory and coding for steganography. First, in steganography the number of stuck cells can be quite large (e.g., 90% or more). Second, the number of stuck cells varies significantly with the stego method and for different instances of the cover object. Thus, imposing bounds on the rate $r = k/n$ would result in a decreased embedding capacity. Third, fortunately, steganographic applications are often run off line and do not require real time performance. While it is quite acceptable to spend 2 seconds to embed a $10,000$-bit payload, it is not acceptable to spend this time writing data into memory.

2.1. Syndrome coding

Without loss of generality, we will assume that the cover objects are grayscale digital images. Let us assume that the cover image $\mathbf{x}$ consists of n pixels x_i, $x_i \in \{0, 1, \ldots, 255\}$, $i = 1, 2, \ldots, n$. The sender selects k changeable pixels x_j, $j \in J \subset \{1, 2, \ldots, n\}$, $|J| = k$, which is the selection channel. The changeable pixels may be used and modified independently from each other by the sender to communicate a secret message to the recipient, while the remaining pixels corresponding to stuck cells are not modified during embedding.

It is further assumed that the sender and the recipient agree on a mapping

$$b : \{0, 1, \ldots, 255\} \quad \rightarrow \quad \{0, 1\},$$

for example, $b(x) =$ the LSB of x. During embedding, the sender either leaves the changeable pixels x_j, $j \in J$, unmodified or replaces x_j with y_j to modify its bit from $b(x_j)$ to $b(y_j)$. The vector of cover image bits $\mathbf{b_x} = (b(x_1), \ldots, b(x_n))^t$ changes to $\mathbf{b_y} = (b(y_1), \ldots, b(y_n))^t$, where $\mathbf{x}^t$ denotes the transpose of $\mathbf{x}$. To communicate m bits $\mathbf{m} \in \mathbb{F}_2^m$, the sender modifies some changeable pixels x_j, $j \in J$, so that

$$\mathbf{D}\mathbf{b_y} = \mathbf{m}, \tag{1}$$

where $\mathbf{D}$ is an $m \times n$ binary matrix shared by the sender and the recipient. Equation (1) can be further rewritten to

$$\mathbf{D}\mathbf{v} = \mathbf{m} - \mathbf{D}\mathbf{b_x} \tag{2}$$

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

$$\mathbf{H}\mathbf{v} = \mathbf{s}, \tag{3}$$

where $\mathbf{H}$ is an $m \times k$ matrix consisting of those columns of $\mathbf{D}$ corresponding to indices J, $\mathbf{v}$ is an unknown $k \times 1$ vector, and $\mathbf{s} = \mathbf{m} - \mathbf{D}\mathbf{b_x}$ is the $m \times 1$ right hand side.

The task of finding the vector $\mathbf{v}$ amounts to solving a system of m linear equations with k unknowns in $\mathbb{F}_2$. Minimizing the number of embedding changes means finding a solution $\mathbf{v}$ with the minimal weight. Both tasks could be efficiently solved by considering $\mathbf{H}$ as a parity check matrix of some $[k, k - m]$ linear code and imposing structure on $\mathbf{H}$. Thus, solving (3) with minimal number of embedding changes is equivalent to finding the coset leader $\mathbf{v}$ for syndrome $\mathbf{s}$. However, the matrix $\mathbf{H}$ is obtained from $\mathbf{D}$ as a column submatrix defined by the selection channel. Because the selection channel can be arbitrary, e.g., even random or dependent on the cover, it is difficult to impose structure on $\mathbf{D}$ that would be inherited by $\mathbf{H}$ and that would help us incorporate the apparatus of structured codes. Moreover, we need a whole class of good codes for various values of n, k, and m.

What we need is a binary matrix $\mathbf{D}$ whose k-column submatrices are regular with high probability and have the smallest possible covering radius. A natural step here is to look at random linear codes because they are optimal in both respects with increasing code length and fixed relative message length $\alpha = m/k$.

We first address the task of finding a computationally efficient method for solving (3), postponing the issue of minimizing the number of embedding changes to Section 4. We quote below the result obtained for variable rate random linear codes that appeared in [12]. Assuming that the sender always tries to embed as many bits as possible by adding rows to $\mathbf{D}$ while (3) still has a solution, for random binary matrices whose elements are i.i.d. realizations of a random variable uniformly distributed in $\{0, 1\}$, the expected value of the maximum message length $m_{\max}$ that can be communicated in this manner is

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

as $k \to \infty$, $k < n$. Thus, variable-rate random linear codes asymptotically achieve the maximal embedding capacity.

Without any structure in $\mathbf{H}$, the sender might attempt to solve (3) simply using Gaussian elimination. Assuming that the maximal length message ($m = k$) is sent, the complexity of Gaussian elimination is $O(k^3)$, which would lead to impractical performance for large payloads. In [12], 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. In this case, the complexity of embedding is proportional to $n/n_B \times (kn_B/n)^3 = nr^3n_B^2$, where $r = k/n$. Assuming fixed r, the complexity is linear in the number of cover object elements n, albeit with a large constant.

By imposing a special *stochastic* structure on the columns of $\mathbf{D}$, 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.

3. MEMORY WITH DEFECTIVE CELLS USING LT PROCESS

3.1. LT codes

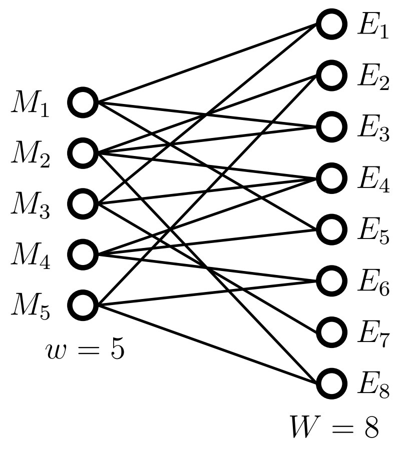
LT codes are universal erasure codes with low encoding and decoding complexity that asymptotically approach the Shannon capacity of the erasure channel. For simplicity, we only use binary symbols noting that the codes can work without any modification with l-bit symbols. The best way to describe the 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

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

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

$$W > \beta w = w + O(\sqrt{w}\ln^2(w/\delta)). \tag{5}$$

The encoding bits can also be obtained from message bits using matrix multiplication with the bi-adjacency binary matrix $\mathbf{A}$ (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):



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

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

> *Find an encoding bit that has a single 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 $\mathbf{Ax} = \mathbf{y}$ with a random matrix $\mathbf{A}$ 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 $\mathbf{H}$ was obtained from $\mathbf{D}$ by removing columns, $\mathbf{H}$ inherits the distribution of Hamming weights of columns from $\mathbf{D}$ but not the distribution of its rows. However, as explained in detail below, the LT process can be used to quickly bring $\mathbf{H}$ 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 $\mathbf{A}$. For an n-dimensional binary vector $\mathbf{r}$, let $\mathrm{w}_j(\mathbf{r})$ denote the Hamming weight of $(r_j, \ldots, r_n)$ (e.g., $\mathrm{w}_1(\mathbf{r}) \equiv \mathrm{w}(\mathbf{r})$ is the usual Hamming weight of $\mathbf{r}$). We first find a row $\mathbf{r}$ in $\mathbf{A}$ with $\mathrm{w}_1(\mathbf{r}) = 1$ (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 $\mathbf{A}$ to a lower

k	Gauss	LT	β	P
1,000	0.023	0.008	1.098	43%
10,000	17.4	0.177	1.062	75%
30,000	302	0.705	1.047	82%
100,000	9320	3.10	1.033	90%

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.

triangular form by permuting its rows and columns. Continuing the process, we search for another row $\mathbf{r}$ with $\mathrm{w}_2(\mathbf{r}) = 1$ (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 $\mathbf{r}$ with $\mathrm{w}_3(\mathbf{r}) = 1$, etc. Assuming the process finishes successfully, at the end the permuted matrix $\mathbf{A}$ will be lower diagonal with ones on its main diagonal.

Returning to the problem of solving the system $\mathbf{Hv} = \mathbf{s}$ with m equations for k unknowns, $m < k$. By applying the above process of row and column permutations to $\mathbf{H}^t$, we bring $\mathbf{H}$ to the form $[\mathbf{U}, \mathbf{H}']$, where $\mathbf{U}$ is a square $m \times m$ upper triangular matrix with ones on its main diagonal and $\mathbf{H}'$ is a binary $m \times (k - m)$ matrix. We can work directly with $\mathbf{H}$ if we replace in the algorithm above the word 'row' with 'column' and vice versa. We now require, however, that the Hamming weights of columns of $\mathbf{H}$ follow the RSD and the message length m satisfies (from (5))

$$k > \beta m = m + O(\sqrt{m} \ln^2(m/\delta)). \tag{6}$$

This means that there is a small capacity loss 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 $\mathbf{D}$ 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 1s in $\mathbf{D}$ (and thus in $\mathbf{H}$) is $O(\ln(k/\delta)/k)$. Therefore, the complexity of embedding 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 $\mathbf{D}\mathbf{b}_{\mathbf{x}}$, 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 $\mathbf{D}\mathbf{b}_{\mathbf{y}}$.

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 [12]. 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.

4. CODING ON SMALL BLOCKS

In the previous section, we showed that LT codes can be used to construct computationally efficient data hiding algorithms. We now address the issue of the number of embedding changes because the embedding efficiency is an important attribute of steganographic schemes.

Ideally, the sender should choose a solution $\mathbf{v}$ to (3) with the smallest Hamming weight (the coset leader). On the other hand, we would like to do so using a computationally efficient process similar to the LT process. In fact, there are several degrees of freedom in the matrix LT process that can be used to decrease the Hamming weight of the solution $\mathbf{v}$. However, our attempts to adjust the LT process were only moderately successful when compared to theoretically achievable bounds (Section 4.2 in [13]). It is possible, though, that other stochastic properties may be imposed on $\mathbf{D}$ that will allow finding solutions of (3) with small weight efficiently.

Another possibility would be to use existing decoding algorithms for LDPCs based on belief propagation. However, the decoding algorithms are poor quantizers when the syndrome is selected randomly (the iterative algorithms do not converge).

Therefore, we turned our attention to random codes of small length where fast exhaustive searches are computationally feasible. In this case, however, we need to study how much is lost on optimality of coding as random codes are only asymptotically optimal.

4.1. Meet-in-the-middle algorithm

We remind that the cover image has n pixels and k changeable pixels and that we wish to communicate m message bits. The sender and receiver agree on a small integer p (e.g., $p < 20$) and using the stego key divide the cover image into $n_B = m/p$ disjoint pseudo-random blocks of cardinality $n/n_B = pn/m$ (for simplicity we assume the quantities above are all integers). Each block will contain on average $k/n \times pn/m = pk/m = p/\alpha$ changeable pixels, where $\alpha = m/k$, $0 \leq \alpha \leq 1$, is the relative message length. The sender will use a pseudo-random binary $p \times pn/m$ matrix $\mathbf{D}$ for embedding up to p bits. The matrix $\mathbf{D}$ can be different or the same in each block and may also depend on a secret stego key. Note that since duplicates and zero columns in $\mathbf{D}$ do not help, as long as* $n/n_B = pn/m < 2^p$, we can generate $\mathbf{D}$ so that its columns are non-zero and mutually different.

As described in Section 2, in each block the sender forms a binary sub-matrix $\mathbf{H}$ of $\mathbf{D}$ and calculates the syndrome $\mathbf{s}$. The matrix $\mathbf{H}$ will have exactly p rows and, on average, p/α columns. Let $C_1 \subset \mathbb{F}_2^p$ be the set of all columns of $\mathbf{H}$, and $C_{i+1} = C_1 + C_i - (C_1 \cup \cdots \cup C_i) - \{\mathbf{0}\}$, for $i = 1, \ldots, p-1$. Note that $C_i = \emptyset$ for $i > R$, where R is the covering radius of $\mathbf{H}$. Also note that C_i is the set of syndromes that can be obtained by adding i columns of $\mathbf{H}$ but no less than i (equivalently, C_i is the set of all coset leaders of weight i).

Let $\mathbf{s} = \mathbf{h}_{j_1} + \cdots + \mathbf{h}_{j_r}$ be the minimal number of columns of $\mathbf{H}$ adding up to $\mathbf{s}$, $r \leq R$. Then, $\mathbf{s} + \mathbf{h}_{j_1} + \cdots + \mathbf{h}_{j_{\lfloor r/2 \rfloor}} = \mathbf{h}_{j_{\lfloor r/2 \rfloor + 1}} + \cdots + \mathbf{h}_{j_r}$, which implies $(\mathbf{s} + C_{j_{\lfloor r/2 \rfloor}}) \cap C_{j_r - j_{\lfloor r/2 \rfloor}} \neq \emptyset$

*This will be satisfied for embedding in typical digital media files because we use $p \approx 20$ (see below).

Algorithm 1 Meet-in-the-middle algorithm for finding coset leaders

```
1. If $\mathbf{s} \in C_1$, $v_{j_1} = 1$ and $v_j = 0$ otherwise. Stop.
   Else $l = 1, r = 1$

2. If $(\mathbf{s} + C_l) \cap C_r \neq \emptyset$ then there is a solution $\mathbf{v}$ of weight $l + r$ determined by any
   vector from the intersection. Stop.
   Else $\{$if $l = r$, $r = r + 1$ else $l = l + 1\}$

3. Go to 2
```

and $\mathbf{v}$ with zeros everywhere except for indices $j_1, \ldots, j_r$ solves (3). This leads to the Algorithm 1 for finding the coset leaders.

After the solution $\mathbf{v}$ is found, the sender modifies the pixels in the block accordingly—the non-zero elements of $\mathbf{v}$ determine pixels x_i within the block where embedding changes must take place to change their bit values $b(x_i)$. We remind that the modified block of pixels in the stego image is denoted $\mathbf{y}$.

The extraction algorithm is very simple. The recipient knows n from the stego image and knows p as this is a publicly shared parameter. Since the message length m is used in dividing the image into blocks, it needs to be communicated in the stego image as well. This can be arranged in many different ways, for example, by isolating from the image a small subset (using the stego key) and embedding $\lceil \log_2 m \rceil$ bits in it using simple Gaussian elimination. Knowing m, the recipient uses the secret stego key and partitions the rest of the stego image into the same disjoint blocks as the sender and extracts p message bits $\mathbf{m}$ from each block of pixels $\mathbf{y}$ as $\mathbf{m} = \mathbf{D}\mathbf{y}$.

Before we analyze the proposed algorithm and describe other implementation issues, in the next section we briefly review known bounds on embedding efficiency.

4.2. Bounds on embedding efficiency

The embedding efficiency of a steganographic method is defined as the number of random message bits embedded per one embedding change. Let $R \leq k/2$ be the covering radius of a linear $[k, k-m]$ code C with $m \times k$ parity check matrix $\mathbf{H}$. This means that every syndrome can be generated by adding at most R columns of $\mathbf{H}$. Because there are $\binom{k}{i}$ different sums of i columns of $\mathbf{H}$, we have the sphere-covering bound

$$2^m \leq \sum_{i=0}^{R} \binom{k}{i} = V(k, R) \leq 2^{kH(R/k)}, \tag{7}$$

where $V(k, R)$ is the volume of a ball of radius R in $\mathbb{F}_2^k$ and $H(x) = -x \log_2 x - (1 - x) \log_2(1 - x)$ is the binary entropy function. The second inequality is a frequently used bound in coding (e.g., Lemma 2.4.3 in [14]).

We define the lower embedding efficiency $\underline{e}$ as the ratio $\underline{e} = m/R$, which is the number of embedded bits per embedding change in the worst case when we have to make all R changes.

For practical purposes, steganographers are more interested not in the worst case but the expected number of embedding changes for messages uniformly distributed in $\mathbb{F}_2^m$, which is the average weight of all coset leaders. It is easy to see that this quantity is equal to the "average covering radius" R_a defined as

$$R_a = \frac{1}{2^k} \sum_{\mathbf{x} \in \mathbb{F}_2^k} d(\mathbf{x}, C), \tag{8}$$

where C is the set of all codewords and $d(\mathbf{x}, C) = \min_{\mathbf{y} \in C} d(\mathbf{x}, \mathbf{y})$ is the distance between $\mathbf{x}$ and the code C. Thus, we define the embedding efficiency as $e = m/R_a$.

Obviously, $\underline{e} \le e$, which is why $\underline{e}$ is called the lower embedding efficiency. The inequality (7) enables us to derive an upper bound on $\underline{e}$ and eventually on e.

From (7),

$$\begin{aligned} \alpha = m/k &\le H(R/k) \\ H^{-1}(\alpha) &\le R/k \\ \underline{e} = \frac{m}{R} &\le \frac{\alpha}{H^{-1}(\alpha)}, \end{aligned} \tag{9}$$

where $H^{-1}(x)$ is the inverse of $H(x)$ on $[0, 1/2]$. Thus, we have obtained an upper bound on the lower embedding efficiency for a fixed relative message length α. It is possible to show that $\frac{\alpha}{H^{-1}(\alpha)}$ is also an asymptotic upper bound on e for linear codes $[k, k - \alpha k]$ as $k \to \infty$. The proof of this statement can be found in [15].

Furthermore, it is known that the upper bound is asymptotically achievable using almost all random linear codes $[k, k - \alpha k]$ with $k \to \infty$ (see Theorem 12.3.5. in [14], page 325).

4.3. Algorithm complexity and implementation issues

In Algorithm 1, in the worst case, we need to calculate all sets $C_1, \ldots, C_{\lceil R/2 \rceil}$. The cardinalities of C_i increase with i, achieve a maximum for $i \approx R_a$, and then quickly fall off to zero for $i > R_a$. With increasing length of the code (or increasing p) and fixed α, $R_a \to R$. This means that the Meet-in-the-middle algorithm avoids computing the largest of the sets C_i. Nevertheless, we will need to keep in memory the sets $C_i, i = 1, \ldots, \lceil R/2 \rceil$ and the indices $j_1, \ldots, j_i$ for each element of C_i. Because on average $|C_1| = p/\alpha$, we have on average $|C_i| \le \binom{p/\alpha}{i}$. Thus, the total memory requirements are bounded by $O\left(R/2 \cdot \binom{p/\alpha}{R/2}\right) \approx O\left(p \cdot 2^{p/\alpha H(R\alpha/2p)}\right) \approx O\left(p 2^{\beta p}\right)$, where $\beta = \frac{H(H^{-1}(\alpha)/2)}{\alpha} < 1$, because $R \approx p/\alpha H^{-1}(\alpha)$ for large p from (9). For example, for $\alpha = 1/2$, $\beta = 0.61$. To obtain a bound on the computational complexity, note that we need to compute $C_1 + C_i$ for $i = 1, \ldots, R/2$. Thus, the computational complexity is also bounded by $O\left(R/2 \cdot p/\alpha \cdot \binom{p/\alpha}{R/2}\right) \approx O\left(p 2^{\beta p}\right)$.

At this point, we note that we studied other fast approaches for finding coset leaders, such as the method based on non-expurgated syndrome trellis proposed by Wadayama [16]. Because the computational complexity of Wadayama's method is $O(p 2^p)$, it is asymptotically slower than the Meet-in-the-middle method.

m/k	0.3	0.4	0.5	0.6	0.7	0.8	0.9
m'/k	0.3	0.4	0.5	0.591	0.660	0.696	0.698

Table 2. Decrease of embedding capacity due to non-solvability of (3) as a function of the relative message length m/k. The values were obtained experimentally for a cover image with $n = 10^6$ pixels, $k = 50,000$ randomly selected changeable pixels, and $p = 18$.

We now comment on the solvability of (3). The equation $\mathbf{Hv} = \mathbf{s}$ will have a solution for all $\mathbf{s} \in \mathbb{F}_2^p$ if and only if rank($\mathbf{H}$) $= p$. The probability of this is $1 - O(2^{p(1-k/m)})$, as this is the probability that a random binary matrix with dimension $p \times p/\alpha$, $\alpha = m/k$, will have full rank (see, for example, [17]). This probability quickly approaches 1 with decreasing message length m or with increasing p (for fixed m and k) because $m < k$.

For k/m close to 1 ($m \sim k$), the probability that rank($\mathbf{H}$) $< p$ may become large enough to encounter a failure to embed all p bits in some blocks. For example, for $p = 18$ and $k/m = 2$, $n = 10^6$, $k = 50,000$, the probability of failure is about 0.0043. The fact that the number of columns in $\mathbf{H}$ varies from block to block also contributes to failures. We note that the probability of failure exponentially quickly decreases to zero with increasing k/m.

To make the method applicable to as wide range of the parameters k, n, and m as possible, 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 $\mathbf{H}$ form a matrix of rank p_i. Furthermore, let $f(q), q = 0, \ldots, p-1, p$, be the probability distribution of p_i over the blocks and random matrices $\mathbf{H}$. The information necessary to communicate p_i is $H(f)$, the entropy of f. Denoting by $E\{f\}$ the expected value of the distribution f, the average number of bits that can be encoded per block is thus $E\{f\} - H(f) \leq p$. Therefore, the pure payload $m' = m(E\{f\} - H(f))/p$ that can be embedded is slightly smaller than m. From Table 2, we show the embedding capacity loss for selected values of m/k. We see that while this loss is negligible for $m/k < 0.6$, it imposes a limit on the maximal relative message length that can be embedded using this method to $\alpha_{\max} = m'/k < 0.698$. In other words, payloads longer than roughly 70% of the maximal embeddable message cannot be embedded using this approach.

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 together with the first bit.

In general, the embedding efficiency improves[‡] with the increasing value of p. However, a practical limit on the largest usable p is imposed by the exponentially increasing complexity and memory requirements. Table 3 shows the embedding time for a one-mega-pixel image (image with $n = 10^6$ pixels), for $k = 50,000$ changeable pixels, for various values of

[†]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.

[‡]Detailed analysis of how the embedding efficiency depends on p is in Section 5.

m/k	0.1	0.2	0.25	0.33	0.5
$p = 17$	0.73	2.29	2.3	2	2.24
$p = 18$	1.17	4.58	4.19	3.58	3.8
$p = 19$	5.28	10.35	7.99	6.59	9.05
$p = 20$	15.74	17.37	12.82	10.19	18.68

Table 3. Embedding time in seconds for messages of different relative length for various values of p. The values were obtained experimentally for a cover image with $n = 10^6$ pixels, $k = 50,000$.

p on a PC equipped with a 3.4 MHz Intel Pentium IV processor. From our simulations, we recommend $p \leq 19$ to keep the embedding time of the order of seconds.

5. EMBEDDING EFFICIENCY

With increasing code length and fixed relative message length, random linear codes asymptotically achieve the theoretical upper bound (9) on embedding efficiency. However, the computational complexity of the proposed coding method imposes a limit on the practically usable code length. In this section, we calculate the embedding efficiency of the proposed method, referring to [18] for more details.

Given two integers p and n, let $\mathcal{H}(p, n)$ be the ensemble of all binary matrices of dimension $p \times n$ with n different non-zero columns. The average number of embedding changes for a given matrix $\mathbf{H} \in \mathcal{H}(p, n)$ is the average covering radius R_a of $\mathbf{H}$ (here calculated in the syndrome space using the sets C_i defined in Section 4.1)

$$R_a = 2^{-p}(|C_1| + 2|C_2| + \cdots + R|C_R|). \tag{10}$$

Let $c_i(p, n)$, $i = 1, \ldots, p$, be the expected value of $|C_i|/2^p$ over matrices $\mathbf{H}$ drawn uniformly from $\mathcal{H}(p, n)$. The expected value of R_a over matrices $\mathbf{H}$ drawn uniformly from $\mathcal{H}(p, n)$ is denoted $r_a(p, n) = \sum_{i=1}^{p} ic_i(p, n)$. An approximate but sufficiently accurate expression for $c_i(p, n)$ has been derived in [18].

The expected number of embedding modifications of the algorithm 1 can be calculated as follows. The number of changeable pixels in each block is a random variable κ that follows hyper-geometric distribution

$$\Pr(\kappa = j) = \frac{\binom{k}{j}\binom{n-k}{n/n_B - j}}{\binom{n}{n/n_B}}, \; j = 0, \ldots, n/n_B. \tag{11}$$

Thus, the average number of embedding changes is

$$\begin{aligned} R_a(p) = E\{r_a(p, \kappa)\} &= E\left\{\sum_{i=1}^{p} ic_i(p, \kappa)\right\} \\ &= \sum_{j=1}^{n/n_B}\sum_{i=1}^{p} ic_i(p, j)\Pr(\kappa = j). \end{aligned} \tag{12}$$

Finally, the expected value of the embedding efficiency is

$$e(p) = p/R_a(p). \tag{13}$$

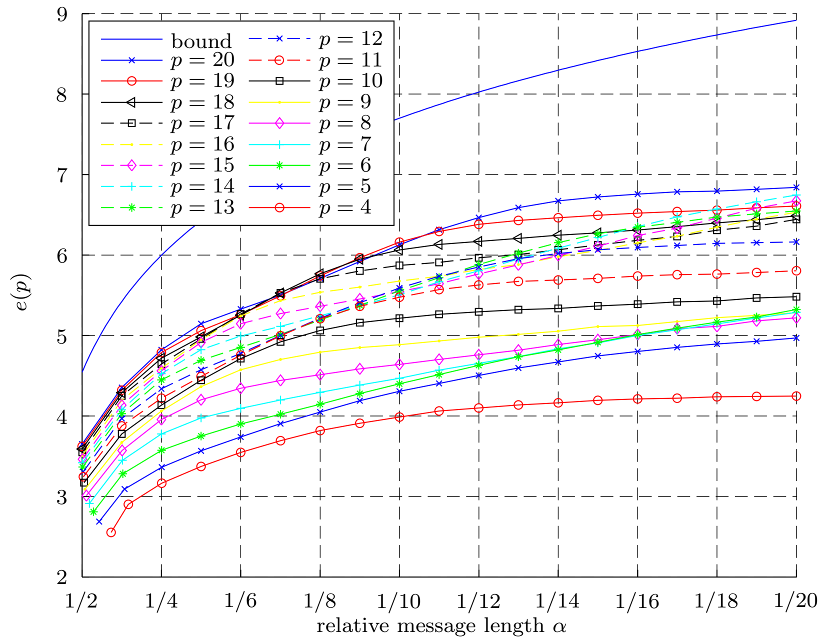


Figure 2. Embedding efficiency $e(p)$ as a function of relative message length α for $p = 4, \ldots, 20$ averaged over 100 embeddings in a one-megapixel image.

6. EXPERIMENTS AND THEIR INTERPRETATION

Figure 2, shows the embedding efficiency as a function of the ratio $\alpha^{-1} = k/m$ for $n = 10^6$, $k = 50,000$, and for $p = 4, \ldots, 20$. It was obtained by averaging over 100 embeddings in a cover object with the same parameters k, n, and m. The solid curve is the asymptotic upper bound (9).

For a fixed p, the efficiency increases with shorter messages (decreasing α). Once the number of changeable pixels in each set exceeds 2^p, the embedding efficiency starts saturating at $p/(1 - 2^{-p})$, which is the value that all curves in Figure 2 reach asymptotically with decreasing α. This is because the p/α columns of $\mathbf{H}$ eventually cover the whole space $\mathbb{F}_2^p$ and thus we embed every non-zero syndrome $\mathbf{s} \neq \mathbf{0}$ using one embedding change (when $\mathbf{s} = \mathbf{0}$ no embedding changes are necessary).

Notice that for fixed α, the embedding efficiency increases in a curious non-monotone manner with increasing p. To see this interesting phenomenon more clearly, we plot e as a function of p for various fixed relative message lengths α. The result is shown in Figure 3. The diagram shows the expected value of embedding efficiency obtained from (13) as a function of $p = 4, \ldots, 80$. Each curve corresponds to a different value of $\alpha = 1/2, 1/3, \ldots, 1/200$.

We see from Figure 3 that with increasing value of p the embedding efficiency increases and reaches the asymptotic value given by the bound (9). However, this increase is not monotone. In fact, it is not always true that increasing p will improve the embedding efficiency. For $p = 19$ and $\alpha = 1/10$ (embedding at 10% of embedding capacity), we obtain an improvement only after we increase p beyond 24. Without this knowledge, we may increase p from 19 to 22 hoping to improve the performance because, in general, increasing p improves embedding efficiency. However, in this case we only increase the embedding time while the embedding efficiency, in fact, decreases!

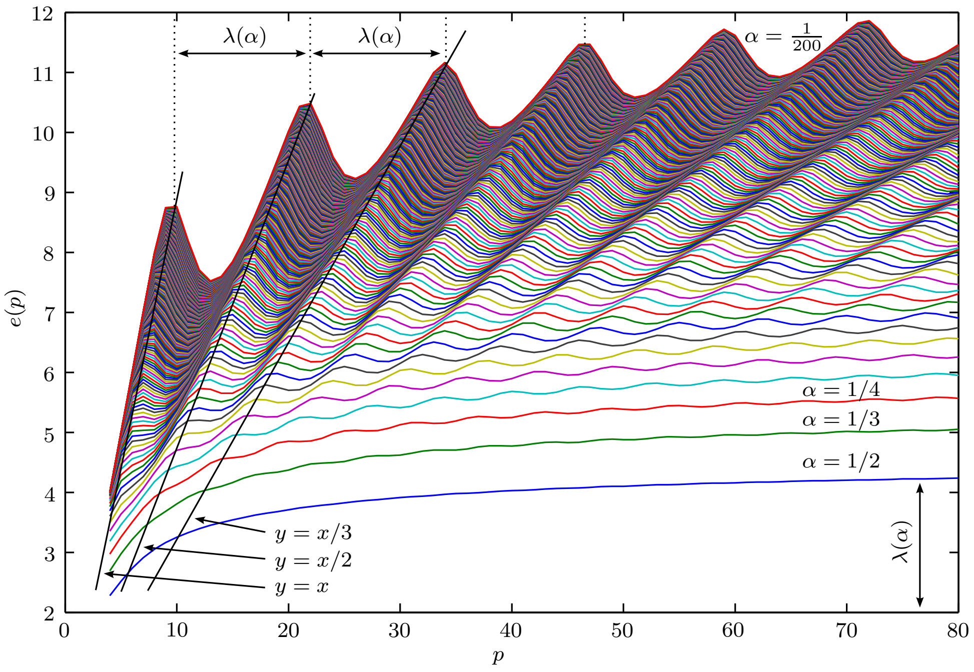


Figure 3. Embedding efficiency $e(p)$ as a function of p for $\alpha = 1/2, 1/3, \ldots, 1/200$.

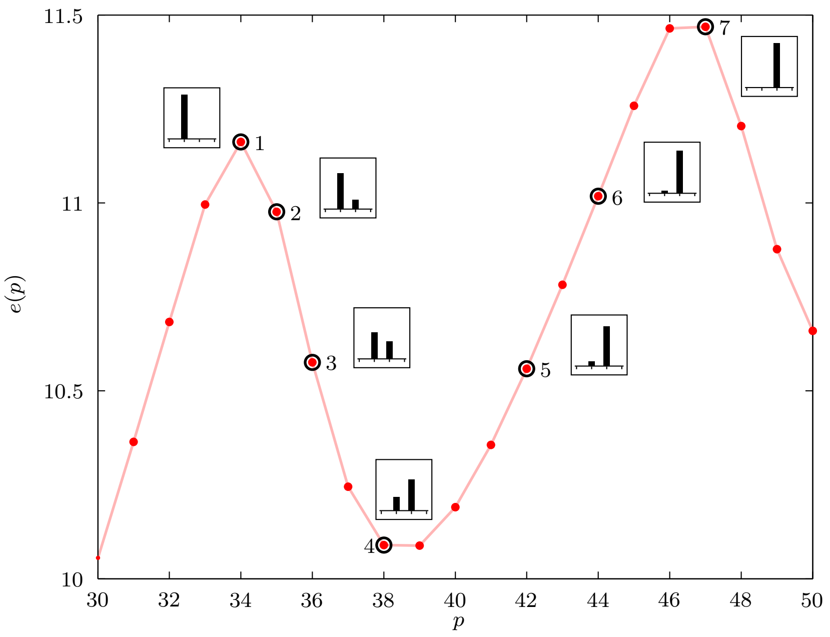


Figure 4. Enlarged portion of $e(p)$ for $p \in [30, 50]$ for $\alpha = 1/200$.

We now provide brief qualitative explanation of the origin of the non-monotone behavior as well as some quantitative description of the diagram. Due to space limitations, details of the exposition are sometimes omitted, only outlining the main ideas.

For $\mathbf{H} \in \mathcal{H}(p, p/\alpha)$ and $i \le p$, $|C_i| \le \binom{p/\alpha}{i} \le 2^{\frac{p}{\alpha}H(i\alpha/p)}$. For any $\epsilon > 0$, let $i(\epsilon) = (1-\epsilon)\frac{p}{\alpha}H^{-1}(\alpha)$. Thus,

$$|C_{i(\epsilon)}| \le 2^{\frac{p}{\alpha}H\left((1-\epsilon)H^{-1}(\alpha)\right)} = 2^p 2^{-\frac{p}{\alpha}\epsilon H'(\xi)},$$

where $\xi \in (H^{-1}(\alpha) - \epsilon, H^{-1}(\alpha))$ from Taylor expansion of $H(x)$ at $H^{-1}(\alpha)$. Thus, with $p \to \infty$ we have $c_{i(\epsilon)}(p, p/\alpha) \le 2^{-\frac{p}{\alpha}\epsilon H'(\alpha)} \to 0$ because $H'(x) > 0$ is decreasing on $[0, 1/2)$. A little more careful argument can be made to show that $E\{c_{i(\epsilon)}(p,\kappa)\} \to 0$, where κ is the random variable (11).

As the upper bound on $c_{i(\epsilon)}$ is exponential in p and because we know that $\sum_{i=1}^{p} c_i = 1$, we can say that with increasing p, the peak of the distribution c_i moves to

$$k \doteq \frac{p}{\alpha}H^{-1}(\alpha), \tag{14}$$

which is the asymptotic covering radius of matrices from $\mathcal{H}(p, p/\alpha)$. This also implies that the embedding efficiency $e = p/R_a(p) \to \frac{\alpha}{H^{-1}(\alpha)} = \lambda$ with $p \to \infty$.

The wave character of $e(p)$ is caused by the corresponding wave character exhibited by R_a. To obtain an insight, inspect Figure 4, which shows a zoomed portion of one "wave" exhibited by $e(p)$ for $\alpha = 1/200$ and the corresponding distribution $c_i(p, p/\alpha)$. Note that the local maxima (the circled points No. 1, 7 in the figure) attained for $p = 34$ and $p = 47$ correspond to cases when $c_3(p, p/\alpha) \approx 1$ and $c_4(p, p/\alpha) \approx 1$, respectively. In these cases, the average covering radius R_a is very close to an integer and the distribution c_i undergoes very small changes. As a result, the growth in R_a is the smallest and thus the ratio p/R_a experiences a peak. With p increasing from $p = 34$ to 47, R_a quickly starts moving from 3 to 4. Because $e = p/R_a$, the decrement in e is largest when the increment in R_a is the largest (point No. 3), which occurs when the peaks switch their places. After they switch places, the changes to the distribution c_i become very subtle again and during this period e experiences growth.

Because the local peaks in e correspond to cases when R_a is an integer, the peaks lie on lines $y = p/k$ for k integer. The lines for $k = 1, 2$, and 3 are shown in the figure. Finally, because for large p, $R_a \approx R \approx p/\alpha H^{-1}(\alpha) = p/\lambda$, and because at the peaks R_a is an integer, the peaks will asymptotically occur at $p_k = k\lambda$, which gives an asymptotic expression for the "wavelength" $p_{k+1} - p_k \to \lambda$. This completes the quantitative explanation of Figure 3.

7. CONCLUSIONS

In this paper, we showed an application of codes for memory with defective cells in steganography with side information available only to the sender. Steganography with side information at the sender's side allows for construction of new schemes with better security because the selection channel—the placement of embedding changes—does not have to be shared with the receiver and can thus be derived from information in principle unavailable to the receiver and thus any attacker (elements of true randomness obtained from an external source, or higher-quality version of the cover media).

Random codes and, specifically, low density parity check codes can be used for coding with memory with defective cells. Asymptotically, random codes are optimal: the actual capacity of the scheme (exponentially) quickly achieves the Shannon capacity. Even though random codes perform well from the theoretical stand point, an efficient implementation must be addressed. In this paper, we showed how to use the process of decoding LT codes (a special class of irregular low density parity check codes) to quickly find a codeword that is compatible with the memory defects.

Even though steganographic schemes based on coding for memory with defective cells generally improve security since the attacker does not know where the embedding changes took place, we can improve security further by trying to reduce the number of changes we make (increase embedding efficiency). This is a problem of finding a codeword with minimal Hamming weight. We showed that by restricting our attention to blocks that are small enough, we can solve this problem by an exhaustive search (the Meet-in-the-middle algorithm). We compared the performance of this algorithm to theoretically achievable bounds.

The experiments with embedding efficiency of the Meet-in-the-middle algorithm revealed somewhat surprising behavior of the actual embedding efficiency. The embedding efficiency increases in a non-monotone matter with increasing block length. This phenomenon and its consequences were quantitatively explained it the last section of this paper.

In our future effort, reflecting on our previous work on application of LT codes to steganography, we plan to investigate low density parity check codes and their iterative decoding algorithms with the intention to obtain good quantizers suitable for steganography with non-shared selection channels with improved embedding efficiency.

8. 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 and Alexander Barg for many useful discussions.

REFERENCES

1. G. Simmons, "The prisoners' problem and the subliminal channel," in *CRYPTO83—Advances in Cryptology*, pp. 51–67, Aug 22–24 1984.

2. A. Westfeld and R. Böhme, "Exploiting preserved statistics for steganalysis," in *Pre-Proceedings, Information Hiding: 6th International Workshop, IH 2004*, J. Fridrich, ed., *Lecture Notes in Computer Science* **3200**, Springer-Verlag, (Toronto, Canada), May 23–25 2004.
3. J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography using wet paper codes," in *MM&Sec '04: Proceedings of the 2004 multimedia and security workshop on Multimedia and security*, J. Dittmann and J. Fridrich, eds., *Proceedings of ACM*, ACM Press, (New York, NY, USA), Dec. 6 2004.
4. A. Kuznetsov and B. Tsybakov, "Coding in a memory with defective cells," **10**, pp. 132–138, 1974.
5. R. Crandall, "Some notes on steganography." Posted on Steganography Mailing List. `http://os.inf.tu-dresden.de/~westfeld/crandall.pdf`, 1998.
6. J. Bierbrauer, "On Crandall's problem." Personal communication, 1998.
7. F. Galand and G. Kabatiansky, "Information hiding by coverings," in *Proc. ITW2003*, pp. 151–154, (Paris, France), 2003.
8. S. I. Gel'fand and M. S. Pinsker, "Coding for channel with random parameters," in *Pered. Inform. (Probl. Inform. Trans.)*, **9(1)**, pp. 19–31, 1980.
9. R. Zamir, S. Shamai, and U. Erez, "Nested linear/lattice codes for structured multiterminal binning," in *IEEE Trans. Inf. Th.*, **48(6)**, pp. 1250–1276, 2002.
10. C. Heegard, "Partitioned linear block codes for computer memory with 'stuck-at' defects," in *IEEE Trans. Inf. Th.*, **29**, pp. 831–842, 1983.
11. C. Heegard and A. El-Gamal, "On the capacity of computer memory with defects," in *IEEE Trans. Inf. Th.*, **29**, pp. 731–739, 1983.
12. J. Fridrich, M. Goljan, P. Lisoněk, and D. Soukal, "Writing on wet paper," in *to appear in IEEE Trans. on Sig. Proc. (Special Issue on Media Security)*, T. Kalker and P. Moulin, eds., 2005.
13. J. Fridrich, M. Goljan, and D. Soukal, "Efficient Wet Paper Codes," in *Proceedings, Information Hiding: 7th International Workshop, IHW 2005*, *Lecture Notes in Computer Science*, Springer-Verlag, (Barcelona, Spain), 2005.
14. G. D. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, *Covering Codes*, vol. 54, Elsevier, North-Holland Mathematical Library, 1997.
15. J. Fridrich and D. Soukal, "Matrix embedding for large payloads," in *submitted to IEEE Transactions on Information Security and Forensics*, 2005.
16. T. Wadayama, "An algorithm for calculating the exact bit error probability of a binary linear code over the binary symmetric channel," pp. 331–337.
17. R. Brent, S. Gao, and A. Lauder, "Random Krylov spaces over finite fields," in *SIAM J. Discrete Math.*, **16(2)**, pp. 276–287, 2003.
18. J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," in *submitted to IEEE Transactions on Information Security and Forensics*, July 2005.