Handling Uneven Embedding Capacity in Binary Images: A Revisit

[a]Min Wu, [b]Jessica Fridrich, [b]Miroslav Goljan, and [a]Hongmei Gou

[a] ECE Department, University of Maryland, College Park, USA
[b] ECE Department, SUNY Binghamton, Binghamton, USA

ABSTRACT

Hiding data in binary images can facilitate the authentication and annotation of important document images in digital domain. A representative approach is to first identify pixels whose binary color can be flipped without introducing noticeable artifacts, and then embed one bit in each non-overlapping block by adjusting the flippable pixel values to obtain the desired block parity. The distribution of these flippable pixels is highly uneven across the image, which is handled by random shuffling in the literature. In this paper, we revisit the problem of data embedding for binary images and investigate the incorporation of a most recent steganography framework known as *the wet paper coding* to improve the embedding capacity. The wet paper codes naturally handle the uneven embedding capacity through randomized projections. In contrast to the previous approach, where only a small portion of the flippable pixels are actually utilized in the embedding, the wet paper codes allow for a high utilization of pixels that have high flippability score for embedding, thus giving a significantly improved embedding capacity than the previous approach. The performance of the proposed technique is demonstrated on several representative images. We also analyze the perceptual impact and capacity-robustness relation of the new approach.

Keywords: Data hiding, binary image, wet-paper codes, uneven embedding capacity.

1. INTRODUCTION

Binary images, such as signatures, drawings, and scanned documents, are increasingly common in our everyday life. Having the capability of hiding data in binary images can facilitate the authentication, annotation, and tracking of these documents in the digital domain. However, data hiding in binary images is much more difficult than in images with a wide range of colors or brightness levels. In a conventional color picture, minor tuning the color of a small pixel is usually not perceivable by eyes, and coded messages can be conveyed through minor color changes on the pixels. On the other hand, black and white are the only two colors in a binary image and they are drastically different to our eyes.

To hide data in a binary image, our prior work [1, 2] first identifies those places in a binary image where a white pixel can be flipped to black or vice versa without introducing noticeable artifacts. Most of these so-called *flippable pixels* are on the edge of characters or along the border of a stroke. The distribution of these flippable pixels is highly uneven over the image – some places have a lot, and other places, such as the background regions, have none. Studies have shown that when maintaining a simple embedding module to embed one bit in one image block (such as through quantization based embedding), this uneven distribution makes it difficult to hide a large amount of data [3]. This is because the decoder would have to know precisely how many bits are hidden in each part of an image, but there is no room for conveying such overhead information. The problem can be solved by shuffling, a technique that randomly permutes the pixels so that the formerly identified flippable pixels occur more evenly across the image. The equalized distribution of flippable pixels can help carry coded messages more easily (for example, hide one bit in one block of shuffled pixels). The effectiveness of shuffling with respect to various parameters was investigated both analytically

The authors can be reached by email at {minwu, hmgou}@eng.umd.edu for Wu and Gou, and {fridrich, mgoljan}@binghamton.edu for Fridrich and Goljan.

and experimentally [3]. Using shuffling brings additional benefit of enhancing security, making the forgery of watermark difficult.

The main limitation of the shuffling technique for equalizing the uneven embedding capacity is that the number of bits that can be embedded is considerably smaller than the total number of flippable pixels. This limitation is in part due to retaining a relatively simple embedding and detection technique for hiding each bit. In this paper, we ask the following question: *is there a better way to handle the uneven capacity and hide more bits*?

The recently introduced approach to steganography called "writing on wet paper" [4−6] shines new light onto this problem. In writing on wet paper, the encoder first identifies the set of k changeable (flippable) pixels and modifies them to embed a secret message. Although the decoder has no knowledge about the location of flippable pixels, the encoder can communicate on average k bits by applying a "wet paper code" (WPC), also known as code for memory with defective cells [7]. The basic idea is to establish a set of linear equations $\boldsymbol{Dx} = \boldsymbol{m}$, where $\boldsymbol{D}$ is a pseudo-random binary matrix shared by the sender and recipient, $\boldsymbol{m}$ is the vector of the $\boldsymbol{q}$-bit data to be embedded, and $\boldsymbol{x}$ is the binary vector of all pixels. Given $\boldsymbol{D}$ and $\boldsymbol{m}$, the sender solves this set of linear equations and determines the values at which the embedding mechanism should flip the flippable pixels. By jointly considering the embedding of multiple bits as opposed to sticking with the simple technique to hide one-bit data at a time, this new paradigm eliminates the need to separately handle the uneven embedding capacity. The average number of embeddable bits can approach the number of flippable pixels, suggesting a potentially significant gain in the embedding payload.

In Section 2, we review previous approaches to embedding in binary images. Then, in Section 3 we explain wet paper codes, and in Section 4 we incorporate them to the data hiding system for binary images and study the achievable embedding rate as well as the visual impact of embedding. We compare the embedding rate, security, and computational complexity with the existing approach that employs a shuffling-based equalization technique. We also address the implication of wet paper codes on the robustness against minor processing and distortions. The paper is concluded in Section 5.

2. DATA EMBEDDING IN BINARY IMAGES

Hiding data in binary images has received a considerable amount of attention since the 1990s. The bi-level constraint also limits the extension of many approaches proposed for grayscale or color images to binary images. Hiding a large amount of data and detecting without the original binary image is particularly difficult for an additive approach [8]. Several methods for hiding data in specific types of binary images have been proposed by taking advantage of the unique characteristics of these images. For example, Matsui et al. embedded information in dithered images by manipulating the dithering patterns [9]; Maxemchuk et al. changed line spacing and character spacing to embed information in textual images for bulk electronic publications [10]. These approaches cannot be easily extended to other binary images and the amount of data that can be hidden is limited.

Applications of authentication, annotation, and steganography generally require high-rate embedding with blind detection. A common framework for high-rate embedding in binary images is to first identify *flippable* pixels whose change would not incur substantial visual distortion, and to enforce properties of a group of pixels via local manipulation of a small number of flippable pixels. For example, Matsui et al. embed data in fax images by manipulating the run-length features [9]. Mei et al. first scan the image in a pre-determined order to extract objects (such as a connected letter or stroke) and then the data is embedded in the outer boundary of an object at a rate of one bit per a five-pixel long boundary if the center pixel of such a boundary pattern is deemed flippable [11]. Koch and Zhao proposed a data hiding algorithm through changing some pixels around contours to enforce the ratio of black vs. white pixels in a block to be larger or smaller than 1 [12]. Enforcing block-based feature to hide data is also used in the work by Wu et al. [1,2], where the uneven distribution of flippable pixels is handled by applying shuffling to equalize the distribution prior to embedding. In the rest of this section, we shall review this data hiding technique in more detail. As we shall see later in the paper, the shuffling approach can be viewed as a special case of the new paradigm of wet paper coding. The latter naturally handles the uneven embedding capacity through random projections and allows for more efficient use of flippable pixels to hide more data.

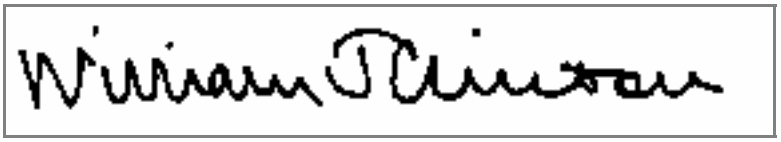
The shuffling-based data hiding technique for binary image first assigns a flippability score to each pixel, which quantifies how unnoticeable the flipping of the pixel is to a human observer. The pixels in the image are then randomly permuted so that the pixels with high flippability distribute more evenly across the image. The shuffled image is then partitioned into blocks and the number of black pixels in each block is counted. One bit is embedded in each block by manipulating the flippable pixels in that block to enforce an odd-even relationship of the black-pixel count. Finally the image is reversely permutated to produce the marked/stego image.

2.1 Flippability score for pixels

When identifying flippable pixels, we take the human perceptual factor into account by studying each pixel and its immediate neighbors to establish a score of how unnoticeable a change on that pixel will be. The score is between 0 and 1, with 0 indicating the pixels that should not be flipped. Flipping pixels with higher scores generally introduces fewer artifacts than flipping those with lower scores.

Our approach of determining the flippability scores is to observe two metrics, smoothness and connectivity. The smoothness is measured by the transitions in a local window, and the connectivity is measured by the number of the black and white clusters. We order all 3×3 patterns in terms of how unnoticeable the change of the center pixel will be. We then examine a larger neighborhood to refine the score, for example, to avoid introducing noise on such special patterns as sharp corners. By changing the parameters in this procedure, we can easily adjust the intrusiveness of different kinds of artifacts and tailor to different types of binary images. More details about calculating the flippability scores can be found in [2]. An example is shown in Figure 1, where most of the pixels with high flippability scores, indicated by black dots in Figure 1(b), are located along the boundary of a stroke.

(a)



(b)

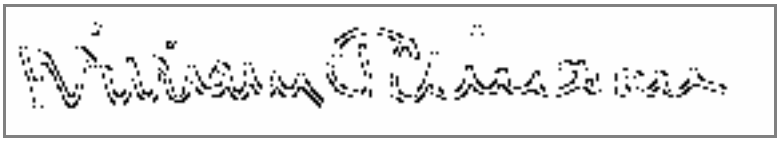


Figure 1: (a) A binary signature image and (b) its pixels with high flippability scores shown in black.

2.2 Embedding and detection mechanism

To embed data in a binary image, we first divide it into blocks and then hide one bit in each block by manipulating pixels with the highest flippability scores in that block to enforce an odd-even parity of the black-pixel count. Specifically, the total number of black pixels in a block is used as the feature for data hiding. To embed a "0" in a block, the total number of black pixels in that block is enforced to an even number. If the original number of black pixels is odd, we flip the pixel with the highest flippability score in the block. Similarly, to embed a "1", the number of black pixels is enforced to an odd number.

Detection can be done by simply checking the enforced relationship in the marked image without the need of the original image. If the total number of black pixels in a block is even, a "0" will be extracted; similarly, a "1" is extracted from a block with an odd number of black pixels.

2.3 Handling uneven embedding capacity via shuffling

As described earlier, we embed multiple bits by dividing an image into blocks and hiding one bit in each block via the enforcement of odd-even relationship. However, the distribution of flippable pixels may vary dramatically from block to block. For example, no data can be embedded in the uniformly white or black regions, while regions with text and drawings may have quite a few flippable pixels, especially on a rugged boundary. This uneven embedding capacity can be seen in Figure 1(b), where most flippable pixels are on the rugged boundaries.

Using variable embedding rate [3] from block to block to accommodate the unevenness is not always feasible for the following reasons. First, a detector has to know exactly how many bits are embedded in each block. Any mistake in estimating the number of embedded bits is likely to cause errors in decoding the hidden data for the current block, and the error can propagate to the following blocks. Second, the overhead for conveying this side information via embedding is significant and could be even larger than the actual number of bits that can be hidden. Using coding to handle the potential mis-synchronization error also adds a considerable amount of overhead. We therefore adopt constant embedding rate to embed the same number of bits in each region and use shuffling to equalize the uneven embedding capacity from region to region.

Figure 2 shows the original histogram of the flippable pixels per 16×16-pixel block in dashed-dotted line. The distribution before shuffling extends from zero to 40 flippable pixels per block, and about 20% of the blocks do not have any flippable pixels. On the other hand, the distribution of flippables after a random permutation of all pixels becomes more even. We perform 1000 shuffles, average the histogram result, and visualize as the circles in Figure 2. It can be seen that almost all blocks have at least one flippable pixel after shuffling.

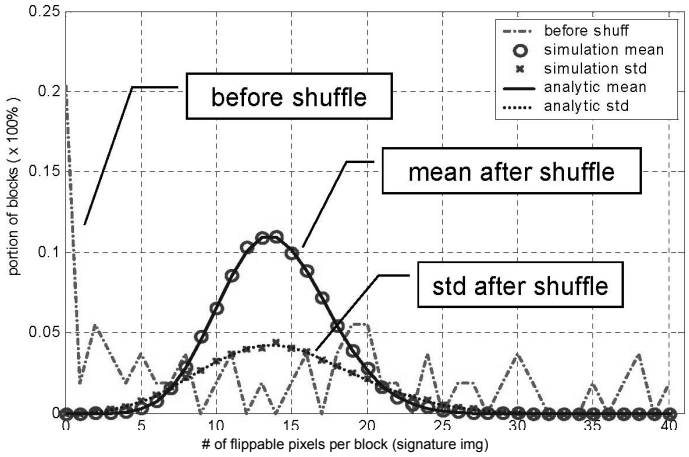


Figure 2: Analysis and simulation of the effect of shuffling for the binary image in Figure 1(a).

To provide a clearer idea on the effect of shuffling, we analyze the expected histogram of flippable pixel count per block. Let μ_s be the number of blocks each having exactly s flippable pixels (s = 0, …, B), B the block size, n the total number of pixels, q the number of blocks, and k the number of flippable pixels. It can then be shown [3] that the mean and variance of each normalized bin of the histogram is

$$E\left[\frac{\mu_s}{q}\right]=\frac{\binom{B}{s}\binom{n-B}{k-s}}{\binom{n}{k}},\qquad Var\left[\frac{\mu_s}{q}\right]=\frac{1}{q}\frac{\binom{B}{s}\binom{n-B}{k-s}}{\binom{n}{k}}-\left(1-\frac{1}{q}\right)\frac{\binom{B}{s}\binom{B}{s}\binom{n-2B}{k-2s}}{\binom{n}{k}}-\left[\frac{\binom{B}{s}\binom{n-B}{k-s}}{\binom{n}{k}}\right]^2. \tag{1}$$

We can see that if the expected histogram describes the probability distribution of a random variable, the mean value of this random variable equals to k/q. For the signature image of Figure 1(a), we have B = 256, n = 48×288, q = 54, and k = 753 (for flippability equal or above 0.1). The analytic results are shown in Figure 2, along with the simulation results from 1000 random shuffles. The simulation results conform to the analysis and the percentage of blocks with no or few flippables is extremely low.

Overall, through shuffling, we dynamically assign the flippable pixels in complex regions and along rugged boundaries to carry more data than less active regions. This is done without the need of specifying much side information that is image dependent. Shuffling also enhances security since the receiver side needs the shuffling table or a key for

generating the table to correctly extract the hidden data. A disadvantage of shuffling, however, is that the number of bits hidden is considerably smaller than the total number of flippable pixels and the utilization of flippable pixels is still quite low. This is because to ensure a sufficiently low left tail in the histogram (thus every block has at least one flippable for hiding one bit without severe distortion), the average number of flippables per block (k/q) must be much larger than 1.

3. WET PAPER CODES FOR DATA EMBEDDING

The wet paper codes were proposed as a solution for a specific scenario that frequently occurs in steganography called "writing on wet paper". To explain this metaphor, imagine that the cover object X is an image that was exposed to rain and the encoder can only slightly modify the dry spots of X but not the wet spots. During transmission, the stego image Y dries out and thus the decoder does not know which pixels the sender used. The rain can be random, pseudo-random, completely determined by the encoder, or an arbitrary mixture of all. The task is to design a method that enables both parties to exchange secret messages. The problem of embedding in binary images fits this "writing on wet paper" paradigm if we realize that the dry pixels correspond to flippable pixels. Since the act of embedding itself may modify the flippability scores of neighboring pixels, the decoder will not be able to correctly identify the pixels that were used for embedding by the encoder.

3.1 Wet paper as memory with defective cells

The writing on wet paper is known among information theorists as "writing to a computer memory with defective cells"[7]. The dry pixels correspond to correctly functioning cells in a memory, while the wet pixels correspond to defective cells that are already "stuck" at either 0 or 1 and cannot be modified by the encoder (or the writing device). The encoder knows the positions and state of the defective cells, while the decoder (the reading device) has no information about the stuck cells.

Computer memory with defective cells was studied in the past [7, 13–16] and it is known that the Shannon capacity for this channel is k, provided there are $n–k$ stuck cells in an n-memory cell. For non-binary cells drawn from a set of q symbols, examples of codes that reach this capacity are Reed-Solomon codes, or in general any MDS (Maximum Distance Separable) codes [14]. When applied to binary memories by grouping bits into q-ary symbols, however, this approach is especially inefficient when the number of stuck bits is not small, which is, unfortunately, often the case in steganographic applications. Moreover, one would also need to assume a certain upper bound on the number of defective cells (wet pixels), which will further limit the capacity of this approach and its usefulness. Fridrich et al. [4, 5] proposed variable-rate random linear codes and showed that these codes asymptotically (and quickly) achieve the Shannon capacity and described an efficient practical implementations using a fast software implementation of structured Gaussian elimination or using LT codes [6]. Another advantage of this approach was that the encoder could also impose a preference on which dry pixels should be used first if a message shorter than the maximum capacity was to be embedded. In this paper, we briefly describe this code and refer the reader to [4–6] for more details.

Let us assume that the cover binary image $\boldsymbol{x}$ consists of n elements $\{x_i\}_{i=1}^{n}$, $x_i \in \{0,1\}$. The encoder specifies a set of k flippable pixels $\{x_j\}$ as described in Section 2, where $j \in C \subset \{0, 1, \ldots, n–1\}$ and $|C| = k$. During embedding, the encoder either leaves the flippable pixels unmodified ($y_j = x_j$), or replaces x_j with $y_j = 1 – x_j$. The embedded image $\boldsymbol{y}$ also consists of n pixels $\{y_i\}_{i=1}^{n}$.

3.2 Basic embedding and extraction

Consider the case of embedding q bits $\boldsymbol{m} = \{m_1, \ldots, m_q\}^{\mathsf{T}}$ in X. For simplicity, we first assume that q is known to the message decoder, and we will later show how to relax this assumption. A secret watermarking key is used to generate a pseudo-random binary matrix $\boldsymbol{D}$ of dimensions $q \times n$. During embedding, the flippable pixels x_j, $j \in C$, are modified so that the embedded image $\boldsymbol{y} = \{y_i\}_{i=1}^{n}$ satisfies

$$\boldsymbol{D}\boldsymbol{y} = \boldsymbol{m}. \tag{2}$$

Thus, the sender needs to solve a system of linear equations in GF(2) (in binary arithmetic). The maximal length of the message that can be communicated in this way is related to the expected rank of the matrix $\boldsymbol{D}$ and is discussed below. Note that the selection channel of Anderson and Petitcolas [17] is a special case of (2) with $\boldsymbol{D} = [1, \ldots, 1]$.

The message $\boldsymbol{m}$ is extracted simply by performing a matrix multiplication $\boldsymbol{m} = \boldsymbol{D}\boldsymbol{y}$ using the shared matrix $\boldsymbol{D}$. The bigger computational complexity is obviously on the encoder's side who needs to solve (2).

The assumption that the decoder knows q can be relaxed as follows. The matrix $\boldsymbol{D}$ can be generated in a row-by-row manner rather than as a two-dimensional array of $q \times n$ bits. In this way, the encoder can reserve the first $\lceil \log_2 n \rceil$ bits of the message $\boldsymbol{m}$ for a header conveying the number of rows in $\boldsymbol{D}$ – the message length q. The symbol $\lceil x \rceil$ represents the smallest integer larger than or equal to x. To extract the message, the decoder first generates the first $\lceil \log_2 n \rceil$ rows of $\boldsymbol{D}$, multiplies them by the received vector $\boldsymbol{y}$, and reads the header (including the message length q). Then, the rest of $\boldsymbol{D}$ is generated to extract the complete message $\boldsymbol{m} = \boldsymbol{D}\boldsymbol{y}$.

3.3 Solvability of the embedding equation array

We now briefly summarize the arguments in [5] to explain the issue of solvability of (2) and determine the average number of bits that can be embedded. Obviously, for small q, (2) will have a solution with a very high probability and this probability decreases with increasing q. We rewrite (2) to

$$\boldsymbol{D}\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{D}\boldsymbol{x} \tag{3}$$

using the variable $\boldsymbol{v} = \boldsymbol{y} - \boldsymbol{x}$ with non-zero elements corresponding to the pixels the encoder must change to satisfy (2). In the system (3), 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, we 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 $\boldsymbol{v}$, (3) now becomes

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{D}\boldsymbol{x}, \tag{4}$$

where $\boldsymbol{H}$ is a binary $q \times k$ matrix consisting of those columns of $\boldsymbol{D}$ corresponding to indices C, and $\boldsymbol{v}$ is an unknown $k \times 1$ binary vector. This system has a solution for an arbitrary message $\boldsymbol{m}$ as long as $rank(\boldsymbol{H}) = q$. The probability $P_{q,k}(s)$ that the rank of a random $q \times k$ binary matrix† is s for $s \le \min(q, k)$, is in [18], Lemma 4

$$P_{q,k}(s) = 2^{s(q+k-s)-qk} \prod_{i=0}^{s-1} \frac{(1-2^{i-q})(1-2^{i-k})}{(1-2^{i-s})}. \tag{5}$$

It can be shown that for a fixed k, $P_{q,k}(q)$ very quickly approaches 1 as q decreases, $q<k$. This suggests that, on average, the number of bits that one can embed is close to k. Assume that the sender always tries to embed as many bits as possible by adding rows to $\boldsymbol{D}$ while maintaining that (4) still has a solution. Let $q_{\max}$ be the expected maximum number of bits that can be embedded in this manner given k flippable pixels. It can be shown that [5]

$$q_{\max}(k) = k + O(2^{-k/4}). \tag{6}$$

3.4 Practical realization of wet paper codes

The practical implementation of WPC has been discussed in detail in [4–6]. Here we summarize the main ideas. First, it can be shown that (6) also holds when 1's and 0's in the random binary matrix $\boldsymbol{H}$ do not occur with the same probability. In fact, the density δ of 1's in $\boldsymbol{H}$ can be as small as $\log_2(k)/k$ [4,5,19]. This means that the complexity of message extraction is $nq\log_2(k)/k$. The embedding is more complex because it requires solving a system of q linear equations (4). If solved by Gaussian elimination, the complexity of embedding would be $O(q^3)$ plus the cost of calculating the right hand side of (4), which is $(n-k)q\log_2(k)/k$. To reduce the complexity, we proposed [4,5] to divide the image into R disjoint subsets and then perform the embedding on each subset separately. Similar to shuffling, a random subset partitioning ensures that each subset will have approximately the same number of flippables. The

† The matrix elements of a random binary matrix here are i.i.d. realizations of a random variable with equal probability to be 0 and 1.

complexity of this structured Gaussian elimination is $O(R(q/R)^3)=O(q^3/R^2)$. For a properly chosen R, this approach can decrease the computational complexity quite significantly.

Another approach to solving the system (4) more efficiently is to impose a structure on the columns of matrix $\boldsymbol{D}$ to avoid having to solve the equations altogether. In another paper in this volume [6], the apparatus of LT codes [20] is used to generate the columns of $\boldsymbol{D}$ so that their Hamming weight follows the "robust soliton distribution". A modified LT process can then be used to quickly bring $\boldsymbol{H}$ into an upper diagonal form just by permuting its rows and columns. The advantage of this approach is a significantly decreased computational complexity when compared to the structured Gaussian elimination and a greatly simplified implementation. Also, this approach leads to higher embedding efficiency in terms of the number of embedded bits per embedding change [21].

4. DATA EMBEDDING IN BINARY IMAGES USING WET PAPER CODES

Using WPC embedding to replace the shuffling and block-based embedding module in our prior work, we arrive at an improved data hiding system for binary images. The block diagrams in Figure 3 show the embedding and extraction process of the proposed approach. In the following subsections, we evaluate the capacity, perceptual quality, computational complexity, and robustness of the proposed approach.

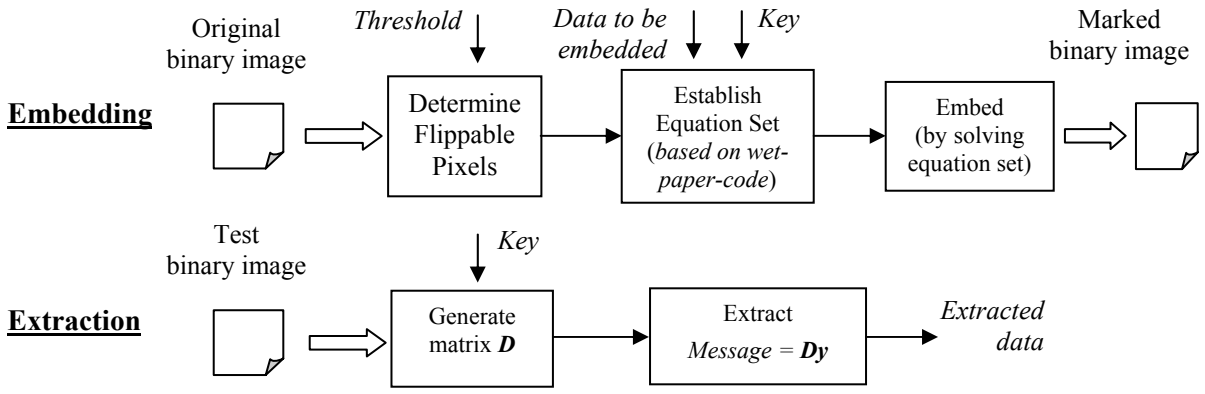
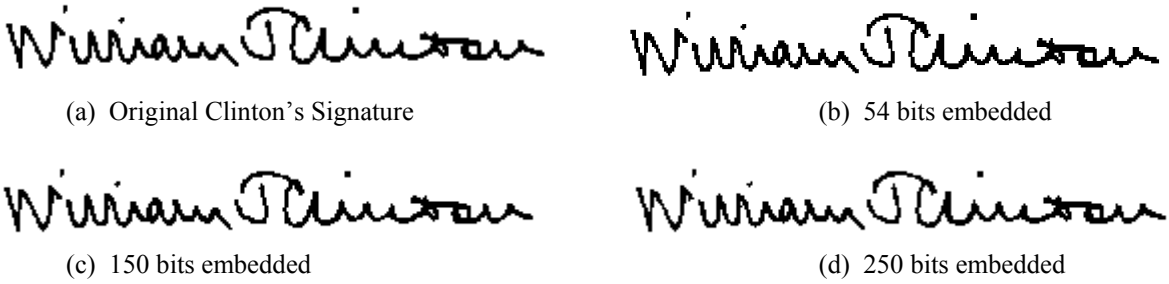


Figure 3: Block diagram of the proposed binary image data hiding system employing the Wet-Paper Codes.

4.1 Capacity and perceptual quality

Figure 4(a) is the original 48×288 binary image "Clinton's signature" in which 250 pixels are with flippability score 0.625, 32 pixels with score 0.375, 3 pixels with 0.25, 86 with score 0.125, 382 with score 0.1, 662 with 0.05, 71 with 0.01, and the remaining 12338 pixels have score 0. Figures 4(b)–(d) show the same image after embedding $q = 54$, 150, and 250 bits, respectively, using wet paper codes with $k = 371$ flippable pixels selected as all pixels with flippability score above 0.1. The pixels that were actually flipped were all pixels with the highest flippability score, with the exception of the highest payload (250 bits) where the WPC algorithm flipped one pixel with the score of 0.375. Visual comparison of the images above indicates that the flippability rule allows the wet paper code to encode up to 250 bits into this test image without introducing visually detectable artifacts. This is an almost five-fold improvement over the shuffling approach that embeds one bit per 16×16 block for a total of 54 bits.

As the second test, we used a cartoon image "Hunter" shown in Figure 5(a) consisting of 120×150 pixels. This image has $k = 523$ flippable pixels with flippability score strictly above 0.1 and 359 pixels with the highest flippability score. The WPC approach was used to embed 359 bits, as shown in Figure 5(b). The pixels flipped during embedding were all with the highest flippability score with the exception of one with the score of 0.375. As a comparison, the embedding capacity for the shuffling approach is 70 bits, which again suggests a five-fold improvement in embedding capacity by the WPC based embedding.



(a) Original Clinton's Signature (b) 54 bits embedded

(c) 150 bits embedded (d) 250 bits embedded

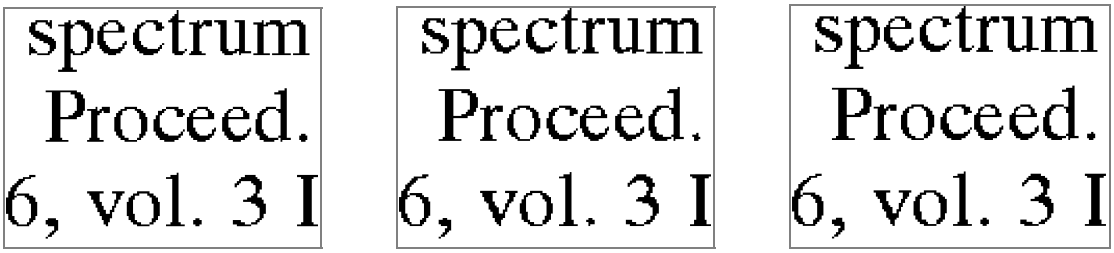
Figure 4: Results on applying WPC embedding to the Clinton's Signature image, which has a total number of 371 pixels having high flippability score.

(a) Original image (b) Embedded with 359 bits

Figure 5: The original and embedded images "Hunter".

As WPC based approach allows a much higher utilization of flippable pixels, the amount of pixels being flipped by the embedding process is larger than in the shuffling based approach. The above results on the signature and cartoon drawings show that flipping more pixels with high a flippability score does not affect the perceptual quality on these hand-written/hand-drawn images. On the other hand, if an image contains objects that a human observer expects to have a defined shape, such as fonts, flipping more pixels may have a larger perceptual impact. One example is the characters in scanned text documents, as shown in Figure 6. To ensure the imperceptibility, the flippability assignment should be adjusted according to the type of the images. For example, we can impose a more stringent constraint on the minimum distance between flippable pixels for text images. The above hand-drawn images employ a trimming step [2] to ensure any 3×3 window has no more than one pixel with a high flippability score; and the trimming window can be enlarged for text document images. Figure 6 is a zoomed-in segment from a 1000×1000 scanned text document to illustrate the impact of the trimming window. A 3×3 trimming window gives a total of 4453 flippables with a score higher than 0.5, a 5×5 trimming gives 3121 high flippables, and a 7×7 trimming gives 2308 high flippables. If we use all available flippables to carry hidden data, on average about half of the flippable pixels would be changed. Comparing Figures 6(b) and (c), we can see that the larger trimming window ensures a larger minimum distance between two

flipped pixels and thus helps maintain good perceptual quality, such as around the dot symbols. The more aggressive trimming reduces the number of high-scoring flippable pixels, showing a tradeoff between the imperceptibility and capacity.



(a) Original (a zoomed-in segment) (b) Results with 3×3 trimming (c) Results with 7×7 trimming

Figure 6: Visual impact of embedding changes for different sizes of the trimming window.

4.2 Computational complexity

The cost of embedding and extraction depends on several factors, such as the total number of pixels n, the number of flippable pixels k, and the message length q. For WPC implementation using Gaussian elimination on disjoint subsets, the number of subsets, R, also influences the complexity. The computational complexity of the shuffling approach is linear with respect to n with a small proportionality constant. The Gaussian elimination has the highest complexity, while the implementation using the LT process has the smallest computational requirements.

Table 1. Computational complexity (for LT process, $\delta \approx 1$).

	Shuffling	Structured Gaussian	LT process
Encoder	$O(n)$	$O((n-q)qR/k \log_2(k/R)+q^3/R^2)$	$O(n \log(q/\delta)/q)$
Decoder	$O(n)$	$O(qnR/k \log_2(k/R))$	$O(n \log(q/\delta)/q)$

4.3 Robustness

Whether it is for binary images or continuous-tone images, high embedding capacity and high perceptual quality are the two primary requirements when designing data embedding systems for authentication, annotation, and steganography applications. In addition, we have seen from the data embedding literature for continuous-tone images that some level of resilience against noise can make a data embedding technique very useful for robust annotation (surviving distortion due to compression) and content-based authentication (which distinguishes malicious content tampering versus content-preserving processing). In this subsection, we examine the robustness of data embedding in binary images that employs the wet paper codes. We consider a simple distortion model where some pixels of a marked image are flipped due to noise. Reliable extraction of hidden data under such distortion is possible if we apply appropriate error correcting coding (ECC) to the data payload before embedding into the host image. We are interested in the raw error probability before error correcting, as it provides a guideline on selecting what ECC to use.

In the following analysis, we model the pseudo-random binary matrix $\boldsymbol{D}$ in the WPC approach as a random binary $q \times n$ matrix, whose elements are i.i.d. realizations of a random variable that is 1 with probability δ and 0 with probability $1-\delta$. Recall that $\delta = (R/k) \log_2(k/R)$ for the Gaussian elimination on R random subsets and $\delta = \log_2(q/\delta)/q$ for the implementation that uses the LT process. Assuming that m pixels were flipped due to noise, we calculate how many message bits will be incorrectly extracted using the WPC with matrix $\boldsymbol{D}$.

We note that the i-th message bit is extracted as XOR of pixels whose indices are determined by 1's in the i-th row of $\boldsymbol{D}$. Thus, it is extracted as incorrect if and only if there are an odd number of noise-flipped pixels among them. The probability the i-th bit will be incorrect is thus

$$P(\delta) = m\delta(1-\delta)^{m-1} + \binom{m}{3}\delta^3(1-\delta)^{m-3} + \ldots = \frac{1-(1-2\delta)^m}{2}. \quad (7)$$

The total number of incorrectly extracted message bits is therefore $qP(\delta) \approx mq\delta$ for small δ.

For comparison, we also analyze the error probability of the shuffling method. The shuffling-based technique reviewed in Section 2 can be viewed as a special case of the WPC where each column of the $\boldsymbol{D}$ matrix has only one non-zero entry. This is because a block is a basic unit for hiding one bit of data and the blocks are mutually disjoint. As a result, each pixel is involved in the embedding of only one bit by the shuffling approach, while the general WPC allows a pixel to be involved in the embedding of more than one bit. The density δ of the corresponding matrix $\boldsymbol{D}$ in the shuffling approach is B/n, as opposed to $\log_2(k)/k$ for the random $\boldsymbol{D}$ matrix of the general WPC approach. Using analysis similar to (1) from Section 2.3, we can calculate the mean value of the percentage of blocks each having exactly r noise-flipped pixels:

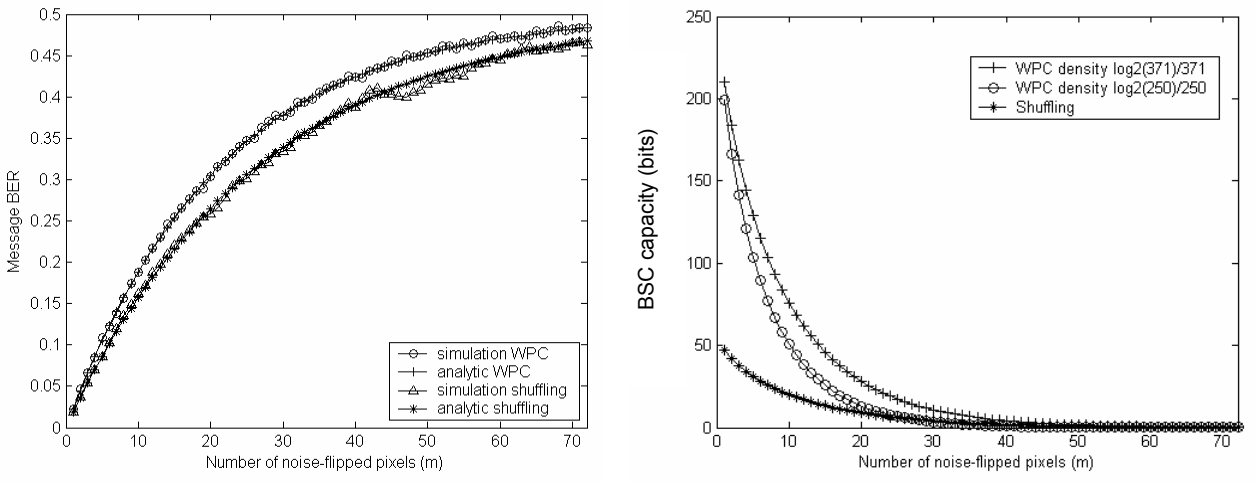
$$E\left[\frac{\mu_r}{q}\right] = \frac{\binom{B}{r}\binom{n-B}{m-r}}{\binom{n}{m}}, \tag{8}$$

where $q = n/B$ is the number of blocks (also the message size), B is the block size, n is the total number of pixels, and m is the number of affected pixels. Since each odd r will bring 1-bit message error, the message BER for the shuffling method is

$$\sum_{r \text{ odd}} E\left[\frac{\mu_r}{q}\right]. \tag{9}$$

Further discussions on the robustness issues of the shuffling approach for binary images can be found in [2].

We use the Clinton's signature as an example to examine the robustness of the WPC and shuffling approaches. As in Section 4.1, for the WPC approach, the total number of flippable pixels $k = 371$, the matrix density $\delta = \log_2(k)/k$, and a total of 250 bits are embedded; for the shuffling approach, the block size $B = 256$, the number of blocks $q = 54$, and the equivalent matrix density is 1/54. The analytic results of m-pixel flipping are shown in Figure 7(a), along with the simulation results from 100 random realizations. We can see that the analysis and simulation agree very well. At the settings given here, the shuffling based approach appears to have smaller bit error rate than the WPC with a pseudo-random random matrix $\boldsymbol{D}$. It is true for both approaches that the lower the density of $\boldsymbol{D}$, the lower the error rate. On the other hand, the low density is a double-edge sword. For the shuffling approach, the low matrix density together with the requirement of having at least one flippable per block limits the embedding capacity. For the WPC approach, when the density of $\boldsymbol{D}$ decreases below $\log_2(k)/k$, the probability of finding a solution to (3) quickly decreases. This reveals a tradeoff between robustness and solvability.



(a) (b)

Figure 7: Robustness comparison of the shuffling and WPC based approach

The robustness analysis and simulation that have been presented so far focus on the bit error rate. Since WPCs allows many more bits to be embedded, it is possible to use a strong ECC code to correct bit errors. As such, it is possible that the actual payload to be reliably conveyed by WPCs becomes higher than the shuffling based approach. An upper bound on the achievable payload can be established through modeling the erroneous channel as a binary symmetric channel (BSC) and examining its Shannon channel capacity. According to information theory [22], a BSC channel with cross-over probability p can reliably convey up to $1–h(p)$ bit per channel use, where $h(p) = –p\log(p) – (1–p)\log(1–p)$. We can obtain the cross-over error probabilities for the shuffling and the WPC approaches based on the BER curves in Figure 7(a), and denote them as p_1 and p_2, respectively. After getting $1–h(p_1)$ and $1–h(p_2)$, we multiply them by the corresponding number of channels available. In our case, the number of channels is the number of (raw) bits embeddable prior to ECC, which is 54 for the shuffling approach and 250 for the general WPC, respectively. As we can see from Figure 7(b), overall the proposed data hiding approach for binary image based on WPC is capable of reliably conveying more bits than the previous shuffling based approach when the amount of noise/distortion is moderate.

5. CONCLUSIONS

In this paper, we revisit the problem of data embedding for binary images and investigate the incorporation of a most recent steganography framework of the wet paper coding to improve the embedding capacity. The wet paper codes naturally handle the uneven embedding capacity through randomized projections. As opposed to the previous shuffling-based approach, where only a small portion of the flippable pixels are actually utilized in the embedding, the wet paper codes allow for a high utilization of pixels that have high flippability score for embedding, thus giving a significantly improved embedding capacity than the previous approach. We demonstrate on several representative images that the proposed technique provides a payload enhancement by about five times prior to error correcting coding. We also analyze and compare the perceptual impact, computational complexity, and the robustness of the new approach with the shuffling approach, and reveal the tradeoff between the capacity, robustness, and computational complexity.

Several issues on practical implementation are addressed in detail in the related publications by the coauthors, and thus omitted in the current paper. For example, how to convey such parameters as the message length q or the number of flippable pixels k to the decoder can be arranged for in several different ways as discussed in [4–6, 21]; the detailed procedure in computing the flippability scores can be found in [2].

Looking ahead, we see that the implementation of WPCs using the LT process offers other, potentially significant advantage, for embedding in binary images. Our recent results indicate that it is possible to increase the embedding efficiency (the number of embedded bits per change) by slightly modifying the LT process. In both shuffling and the WPC using Gaussian elimination, the embedding rate is roughly 2 bits per change, which can be further improved to 3 or more, depending on the length of the embedded message. Taking the signature image as an example, the number of embedding changes decreases from roughly 26 to 12 when embedding 54 bits, thus will further reduce the visual impact of embedding. This topic is part of our future directions and is expected to be elaborated upon in an upcoming paper [21].

ACKNOWLEDGEMENTS

The work on this paper was supported in part by the U.S. National Science Foundation (NSF) under grant# CCR-0133704 (CAREER), and Air Force Research Laboratory, Air Force Material Command, USAF, under the research grant# 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 National Science Foundation, Air Force Research Laboratory, or the U.S. Government.

REFERENCES

[1] M. Wu, E. Tang, and B. Liu: "Data Hiding in Binary Image", *Proc. of IEEE International Conference on Multimedia & Expo (ICME'00)*, vol.1, New York City, NY, pp.393–396, 2000.

[2] M. Wu and B. Liu: "Data Hiding in Binary Image for Authentication and Annotation", *IEEE Trans. on Multimedia*, vol. 6, no. 4, pp.528-538, August 2004.

[3] M. Wu and B. Liu: "Data Hiding in Image and Video: Part-I – Fundamental Issues and Solutions", *IEEE Trans. on Image Proc.*, vol. 12, no.6, pp.685–695, 2003.

[4] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed Quantization Steganography with Wet Paper Codes", *Proc. ACM Multimedia and Security Workshop*, Magdeburg, Germany, Sep. 20–21, pp. 4–15, 2004.

[5] J. Fridrich, M. Goljan, Petr Lisoněk, and D. Soukal, "Writing on Wet Paper", submitted to *IEEE Trans. on Sig. Proc*., Special Issue on Media Security, 2005.

[6] J. Fridrich, M. Goljan, Petr Lisoněk, and D. Soukal, "Writing on Wet Paper", *SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia VII*, San Jose, January 17–20, 2005.

[7] B.S. Tsybakov and A.V. Kuznetsov, "Coding in a Memory with Defective Cells", *Probl. Inform. Transmission* **10**, pp. 132–138, 1974.

[8] Y. Liu, J. Mant, E. Wong, and S.H. Low: "Marking and Detection of Text Documents Using Transform-domain Techniques," Proc. of SPIE, vol. 3657, *Electronic Imaging (EI'99) Conference on Security and Watermarking of Multimedia Contents*, San Jose, CA, 1999.

[9] K. Matsui and K. Tanaka: "Video-steganography: How to Secretly Embed a Signature in a Picture," *Proc. of IMA Intellectual Property Project*, vol. 1, no. 1, 1994.

[10] N. F. Maxemchuk and S. Low, "Marking text documents," in *Proc. IEEE ICIP'97*, 1997.

[11] Q. Mei, E. K. Wong, and N. Memon, "Data hiding in binary text documents," in *SPIE Proc Security and Watermarking of Multimedia Contents III*, San Jose, CA., Jan 2001, vol. 2.

[12] E. Koch and J. Zhao: "Embedding Robust Labels Into Images for Copyright Protection", *Proc. of the International Congress on Intellectual Property Rights for Specialized Information*, Knowledge & New Technologies, 1995.

[13] C. Heegard and A. El-Gamal, "On the Capacity of Computer Memory with Defects," *IEEE Trans. Inform. Theory* **29**, pp. 731–739, 1983.

[14] R. Zamir, S. Shamai, U. Erez, "Nested Linear/Lattice Codes for Structured Multiterminal Binning", IEEE *Trans. Inform. Theory* **48**(6), pp. 1250–1276, 2002.

[15] C. Heegard, "Partitioned Linear Block Codes for Computer Memory with 'Stuck-at' Defects," *IEEE Trans. Inform. Theory* **29**, pp. 831–842, 1983.

[16] G. Cohen, "Applications of Coding Theory to Communication Combinatorial Problems, *Discrete Math.* **83** (2–3), pp. 237–248, 1990.

[17] R.J. Anderson and F.A.P. Petitcolas, "On the Limits of Steganography", *IEEE Journal of Selected Areas in Communications*, Special Issue on Copyright and Privacy Protection 16(4), pp. 474−481

[18] R.P. Brent, S. Gao, and A.G.B. Lauder, "Random Krylov Spaces Over Finite Fields", *SIAM J. Discrete Math*. **16**(2), pp. 276–287, 2003.

[19] C. Cooper, "On the Rank of Random Matrices", *Random Structures and Algorithms* **16**(2), pp. 209–232, 2000.

[20] M. Luby, "LT Codes", *Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science*, November 16–19, pp. 271–282, 2002.

[21] J. Fridrich, M. Goljan, and D. Soukal, "Writing on Wet Paper with LT Codes", in preparation for 7th International Workshop on Information Hiding, Barcelona, Spain, June 2005.

[22] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley Series in Telecommunications, John Wiley & Sons, New York, 1991.