I.J. Modern Education and Computer Science, **2021, 5, 42-57**
Published Online October 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2021.05.05

A Block Permutational Steganographic Algorithm for Scanned Documents and other Images

Saitulaa Naranong
Graduate School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand 10240
E-mail: saitulaa.naranong@gmail.com

Surapong Auwatanamongkol
Graduate School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand 10240
E-mail: surapong@as.nida.ac.th

Received: 28 May 2021; Revised: 15 June 2021; Accepted: 28 June 2021; Published: 08 October 2021

Abstract: Steganography studies the embedding of messages into cover mediums, while obscuring the fact that any message exists. A supplement to encryption, steganographic methods help to avoid attention from adversaries, who may take additional measures if made aware of such messages. Common forms of image steganography, such as Least Significant Bit steganography, alter the first-order statistics of a cover image, allowing for easier detection by methods such as the Wavelet Motion Analyzer. We study steganographic methods based on permutation of pixels in grayscale images, which do not share this disadvantage. A generalization of pixel-swapping methods, our algorithm identifies invariant sets of pixels and intensities, called Permissible Sets, within an image block, and allow their full permutation in the encoding or decoding of messages. This increase in the number of permissible permutations serves to reduce the detectability of our method, while increasing the bit-per-pixel embedding rate. Through direct implementation and comparison, we find our method to be an improvement over previous swap-based steganography for the Microsoft Research Cambridge dataset of general images, and a large improvement for the higher-resolution NoisyOffice dataset of scanned images.

Index Terms: Image, permutation, scanned document, steganography, swap, wavelet motion analyzer

1. Introduction

Digital communications, and the security of the information transmitted through those communications, have become a cornerstone of the modern web [1]. Encryption of data has been, by and large, the main defense to ensure the authenticity of transmitted or received data, and that secret data is not readable by adversaries. Despite this, increasingly sophisticated and oblique attack vectors have been developed to bypass cryptographic solutions. In addition, data breaches are a matter of concern, making it preferable that even stored data is not recognized as such [2, 3]. As such, it has become increasingly worthwhile to combine encryption with data hiding, further enhancing security via data confidentiality [3].

Data hiding techniques may be divided into watermarking and steganography. The former field aims to keep the watermark resistant to attacks, but does not necessarily aim to keep the embedded information hidden [4]: the primary concern of steganography. Steganography is the embedding of messages in some medium, in some way that makes it non-obvious that the message exists. A supplement to encryption, steganography prevents an adversary from determining that there is any message to decode at all: a fact that, if known, could itself have security implications. Data known to be encrypted can be stored *en masse* for later decryption, in the event some weakness to the protocol is later discovered. More immediately, the sender or receiver may be targeted via indirect methods such as malware, deception, various social attacks, or through breaching the security of a data storage endpoint (particularly if the data is stored unencrypted). While these indirect methods do not break the encryption itself, they may nonetheless obtain its contents through some other weakness in the user's system or data storage. It is advantageous, therefore, to obfuscate the fact that any message exists at all to be targeted.

Though classical forms of steganography related to physical messages, steganography in the era of electronic communications relates to information hidden within multimedia files such as text, audio, images, or video [5]. Multiple

means of steganography are possible, including small variations in font [6–9] or spacing [7, 10, 11] within a textual document. Also studied is the hiding of messages within a cover text [12, 13] or audio [14–16]. The effectiveness of any such steganographic method is measured though both detectability (through steganalysis tools), as well as the embedding payload size allowed through the method [17]. There is generally a trade-off between these two factors: as detectability decreases, so does payload size, and vice versa [18]. Other concerns include transparency and robustness. Transparency refers to the lack of noticeable changes between cover and stego-file, while robustness is the resistance of the message against compression, scaling, and noise addition [1].

Image steganography, in particular, begins with a cover image and encodes a hidden message to produce a stego-image; the receiver of the image then decodes the message from the stego-image [1]. This type of steganography sees frequent use; the large number of transmitted bits, and difficulty in distinguishing a subtly-modified image, makes this medium advantageous for the task [19–22]. Common forms of image steganography, such as Least Significant Bit (LSB) steganography, involve the hiding of data within the least significant bits of pixels. While such methods have significant advantages—such as their usability in most images while being indistinguishable to the naked eye—they are known to alter the statistics of the cover image, allowing for easier detection [23–27]. Approaches relating to modified LSB, along with Multi-Level Steganography (MLS), have been explored. Ref. [17] modifies MLS based on dynamic encryption keys with an unlimited number of levels. Ref. [28] explores the use of a secret key to determine encoding/decoding positions in LSB steganography, while [29] studies an improvement of LSB via the use of a byte replacement table based on a secret key. We, however, take a different approach.

Swap-based methods have been previously studied, and have shown some promise in resisting the analyses that detect LSB steganography [30]. They—and, more generally, permutational methods—have the advantage of, unlike LSB, not modifying an image's first-order statistics, as pixels are only permuted, not modified. This obviates the entire class of detection methods that rely on said statistics.

Ref. [31] examines the characteristics of scanner noise with the aim of embedding hidden information within said noise. Their methodology involved the embedding of a noise signal into partially de-noised scans and, according to their analysis, could be detected with an accuracy of 0.862. One of their findings was that, at least for some models of scanner, the pixel values adhere to a normal distribution. Our permutational algorithm avoids, by construction, one of the stated limitations of theirs': that the color histogram is changed, and risks detection. Another approach of a similar paradigm, in which the cover source was modified based on knowledge of camera noise, was taken in [32].

A slight variation of the pixel-swapping algorithm is described in [33]. Their algorithm resists the Sample Pair Analysis [25], which estimates the embedding rate of a hypothetical message in any given image, but is detected by the Wavelet Motion Analyzer [34]. The approach of [33] is largely superseded by [35], which describes a variant block-based pixel-swapping algorithm based partially on comparisons with the block's mean value. Said algorithm was implemented, and evaluated based on detection by the Wavelet Motion Analyzer, for the purpose of comparison. In addition to the Wavelet Motion Analyzer, we also use additional methods to compare the algorithms: the Sample Pair Analysis [25] and Peak Signal Noise Ratio (PSNR) [3]. For both the NoisyOffice dataset [36–38] of scanned documents, and the MSRCv2 dataset [39] of more general images, Wavelet Motion Analyzer results indicate that the algorithm of [35] is outperformed by our own, while the other measures are either roughly comparable or improved.

This paper is organized as follows. Section 2 details our methodology: Section 2.1 describes the algorithm, with the exception of the permutation-to-bitstring (or vice versa) subprocedures, which are described in Section 2.2. Section 2.3 examines the time complexity of the algorithm and its major subprocedures, while Section 2.4 details the actual computation time under testing. Section 2.5 describes the training and testing methodology. Section 3 describes the results of the testing, comparing our algorithm with that of previous work. Finally, we detail our conclusions in Section 4.

2. Methodology

This section describes first our encoding/decoding algorithm, followed by the experimental method for testing its detectability. We study the embedding of messages within 8-bit, grayscale images. We do so both for general images and a type of grayscale image of interest: scanned documents. Noise in scanned backgrounds is already expected, and provides a certain degree of plausible deniability.

We study steganographic methods based on permutation of pixels in grayscale images. The original studied methods were based on pixel-swapping, but said methods were generalized to allow for permutations within 2×2, regions. Eventual generalizations to $n \times n$ regions are forthcoming.

2.1. Algorithm

Both encoder and decoder may be conceptualized as having in common a deterministic image reader subroutine. Said subroutine divides an image into a grid of $n \times n$ blocks (some rows/columns may be left over if n does not divide the image

size), then loops through the blocks using the order given by a base 2 van der Corput sequence [40]. (For example, denote by B the total number of blocks in the image. The fifth member of the van der Corput sequence is $\frac{5}{8}$, so the fifth block traversed would be the block whose index, in lexicographic order, is $\left\lfloor \frac{5}{8} \mathrm{B} \right\rfloor$. Any repeating indices would be skipped in this traversal.) Within each block, the permissible sets of intensities are determined, for use within the encoder/decoder. Because the permissible sets are, by design, an invariant, both encoder and decoder read the same ones. After the permissible sets have been read, the encoder permutes said sets based on the encoded data. The decoder, on the other hand, deduces which permutation was used in order to decode the encoded message.

A permissible set is understood to include the number of occurrences of each intensity (in short, the permissible set of intensities is a multiset), along with the set of pixels corresponding to those intensities. A permutation of a permissible set is a permutation of intensities over the same underlying pixels. An intensity value may be shared by more than one pixel, allowing for different permutations to yield the same intensities over the permissible set pixels. Such permutations are called *indistinguishable*.

Permissible sets may have repeated intensities; this allows for a greater number of allowed permutations than a pure swap-based method, increasing the bit-per-pixel rate of the encoding. The restriction is that the intensities of any two permissible sets must be disjoint; that is, all instances of an intensity value must belong to a single permissible set, to avoid ambiguity.

Other restrictions are also possible: the one we use is to designate an intensity limit L and require that any two intensities within the permissible set may be swapped by a sequence of transpositions, each of intensity difference at most L. More precisely, we require that:

Definition 1 (Sequence-Level Intensity Limit). For each two intensities x,y in a permissible set, there must exist a sequence $x_1,\ldots,x_j$ of intensities, also in the permissible set, such that $x_1 = x$, $x_j = y$, and $|x_{i+1} - x_i| \leq L$.

The permissible sets within a single pixel block are found via a greedy algorithm: at each step, find a maximum-sized multiset of intensities satisfying the above constraints; the set of pixels of any such intensity forms a permissible set. (The Greedy subprocedure is described as Algorithm 1, and the overall procedure summarized in Fig. 1.) When there is more than one maximum-sized permissible set candidate, ties are broken by taking the one containing the minimum intensity. Finally, a canonical ordering of permissible sets is required, in order to sort them: that ordering is by comparing minimum pixel coordinate contained within the set (the coordinates themselves are ordered lexicographically).

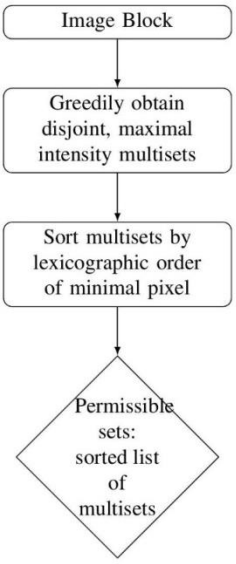


Fig.1. Process of Permissible Set Search within an Image Block

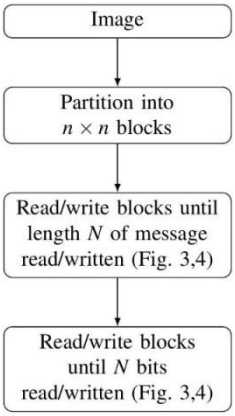


Fig. 2. Process of Encoding/Decoding Data within an Image

Algorithm 1: Permissible Set Search: Greedy Subprocedure

Data:

- A multiset $\mathscr{I}$ of intensity values
- An intensity difference limit L

Result: a maximal permissible subset $X \subset \mathscr{I}$ that satisfies Definition 1

Function ComputePermissibleSetGreedy($\mathscr{I}, L$):

/* Sort the elements of the underlying set in ascending order of intensities: */

$$\{x_1, \ldots, x_m\} = \text{intensities of } \mathscr{I} \text{ (in ascending order)}$$

/* Define $\mathscr{O} : \{1,..,m\} \times \{1,..,m\} \to \mathbb{Z}$, the multiset count between two indicies $p, q \in \{1,..,m\}$, inclusive: */

$$\mathscr{O}(p,q) = \sum_{i=p}^{q} \text{count of } x_i \text{ in } \mathscr{I}$$

/* Define $\mathscr{M} : \{1,..,m\} \to \{1,..,m\}$ to be the function that, given a starting index p, returns the next index q such that $|x_{q+1} - x_q| > L$, or m if there is no such index. ($\mathscr{M}(p)$ is the last index for which the elements between indicies p and $\mathscr{M}(p)$ satisfy Definition 1.) */

$$\mathscr{M}(p) = \min\left(\{m\} \cup \{q : q \in \{p,..,m-1\}, |x_{q+1} - x_q| > L\}\right)$$

/* $p \in \{1,..,m\}$ maximizes the cardinality $\mathscr{O}(p, \mathscr{M}(p))$ between p and its maximum-allowed end-index $\mathscr{M}(p)$ (break ties by choosing the minimum such p) */

$$p = \min\left(\arg\max_{w \in \{1,..,m\}} \mathscr{O}(w, \mathscr{M}(w))\right)$$

/* the sub-multiset of $\mathscr{I}$ whose values are any of $x_p, \ldots, x_{\mathscr{M}(p)}$ */

return $\left\{x : x \in \mathscr{I}, x = x_p \vee \ldots \vee x = x_{\mathscr{M}(p)}\right\}_{\text{multi}}$

In general, for any multiset $\left\{x_1^{k_1}, \ldots, x_j^{k_j}\right\}_{multi}$, the number of distinct permutations is given by [41, 42]:

$$\varphi\left(\left\{x_1^{k_1}, \ldots, x_j^{k_j}\right\}_{multi}\right) = \frac{(k_1 + \cdots + k_j)!}{k_1! \cdot \ldots \cdot k_j!} \tag{1}$$

Denote by $S_1, \ldots, S_k$ the permissible sets in sorted order, and denote the number of distinct permutations for each S_i as $|S_i| = \varphi(S_i)$. Because intensities are (by construction) disjoint between sets, the total number of possible permutations is their product, $N_{perms} = |S_1| \cdot \ldots \cdot |S_k|$. Denote by n_{block} the index of the block the reader is currently looping through, and define $b = \lfloor \log_2 N_{perms} \rfloor$ to be the number of digits that may be encoded in this block.

The encoder receives, as input, a b-digit bitstring, which it interprets as an unsigned integer x_{orig}. We offset by the block index to obtain:

$$x_{offset} = (x_{orig} + n_{block}) \bmod N_{perms}$$

x_{offset} is then represented as a multi-radix number, with radixes $|S_1|, \ldots, |S_k|$:

$$d^1_{|S_1|} \ldots d^k_{|S_k|} = x_{offset}$$

where $0 \leq d^i_{|S_i|} < |S_i|$ is the ith digit. The encoding then proceeds by permuting each S_i, according to Section 2.2: the $d^i_{|S_i|}$th indexed permutation is chosen for S_i, relative to a known base map.

Conversely, the decoder reverses these steps. The permutation of each S_i is read, its index determined (see, again, Section 2.2), and set as the digit $d^i_{|S_i|}$. From there, construct the multi-radix number

$$x_{offset} = d^1_{|S_1|} \ldots d^k_{|S_k|}$$

with radixes $|S_1|, \ldots, |S_k|$. Finally, remove the offset:

$$x_{orig} = (x_{offset} - n_{block}) \bmod N_{perms}$$

and interpret x_{orig} as a b-digit bitstring, to yield the bitstring stored in this block.

See Fig. 3 for an overview and example of this encoding process, and Fig. 4, similarly, for the decoding process.

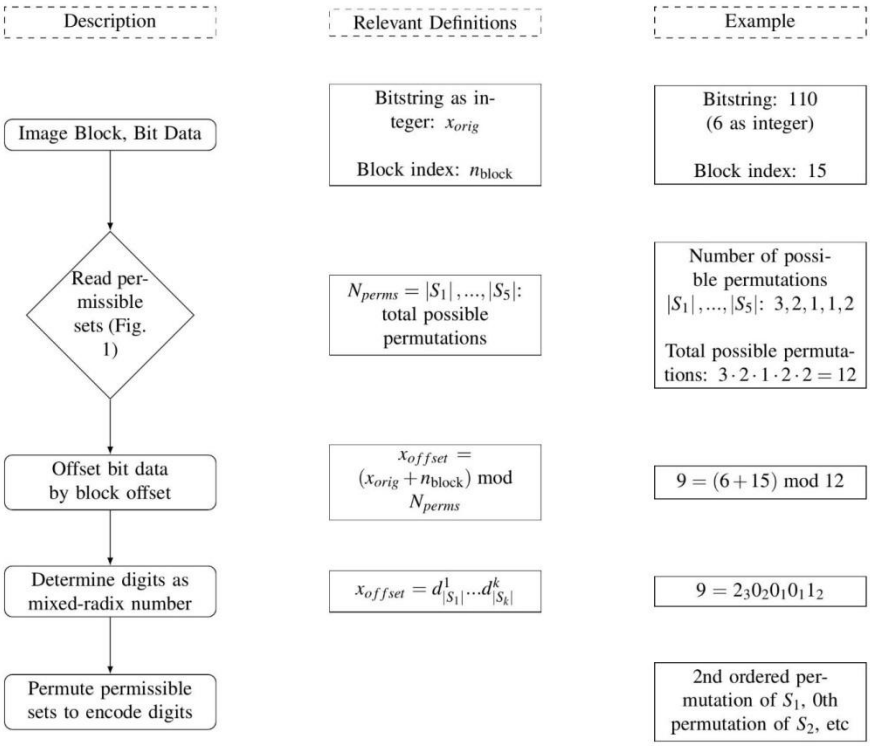


Fig. 3. Process of Encoding Data into Image Block

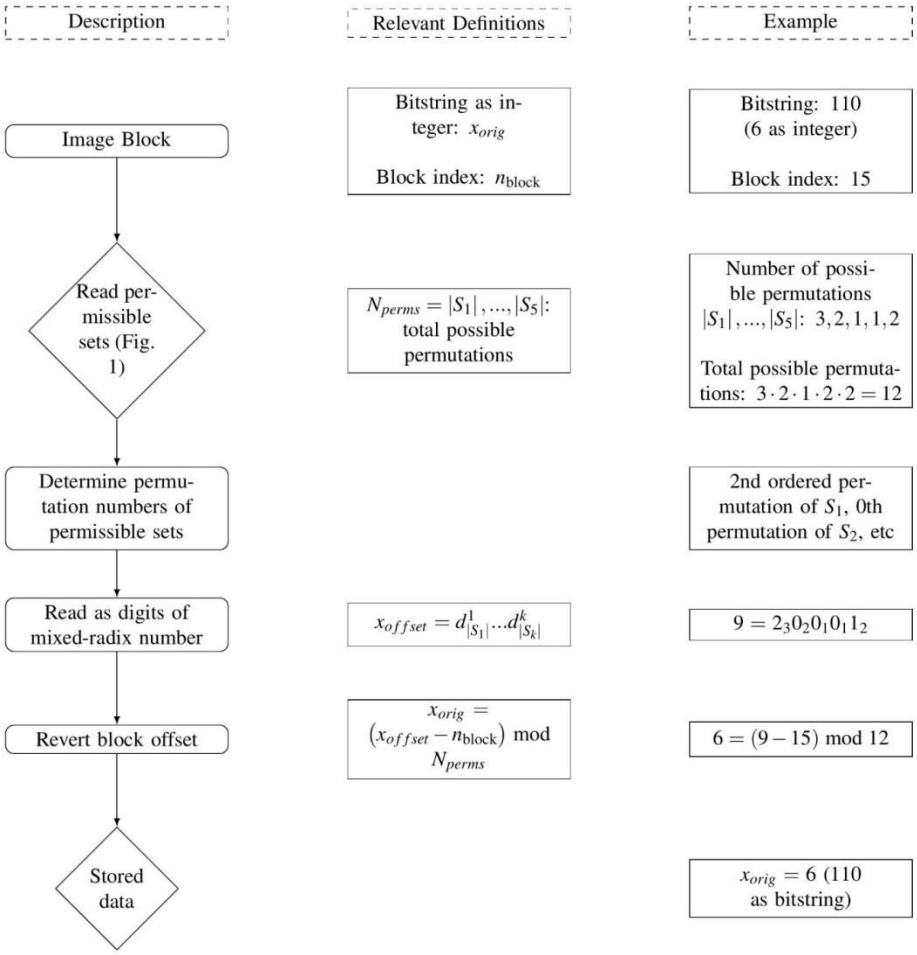


Fig. 4. Process of Decoding Data from Image Block

The only remaining detail is that the encoder must pre-encode the message length, as a fixed-length integer, prior to the actual message (at the beginning of the image block loop). Conversely, the decoder reads said length, prior to decoding the message. In this manner, the stopping point of the message is known by both.

2.2. Permutation Subprocedures

In the sequel, we consider the pixels of an image to be lexicographically ordered by coordinate: $(x_1,y_1) \leq (x_2,y_2) \Longleftrightarrow (x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 \leq y_2)$. A bitstring can be encoded into a permissible set using the following high-level algorithm: begin with a known **base map** of pixel to intensity; the simplest such map is the one that arises from sorting the intensities of a permissible set from lowest to highest among the [lexicographically-ordered] pixels. From there, various possible permutations are possible. One can describe a **permutation-lexicographic ordering** over all such permutations: $\sigma \leq \tau$ if, on the minimum pixel p where $\sigma(p) \neq \tau(p)$, one has $\sigma(p) \leq \tau(p)$.

Given any permissible set S, one may thereby create a permutation-lexicographically sorted list R of all such permutations—or, more accurately, of all distinct permutation results; as permissible sets may contain multiple instances of the same intensity, different permutations may result in indistinguishable mappings of pixel-to-intensity.

The list R can then be used to encode a number i (where $0 \leq i < |S|$): starting from the known base map, enact the permutation result corresponding to index i. Conversely, the decoder determines the index of the permutation result used, thereby extracting the number i.

In practice, the list R of sorted permutation results is purely conceptual: it is far more efficient for an encoder or decoder to compute the index matching a permutation result indirectly.

Let φ, the formula for the number of unique permutations of a generic multiset, be as in Equation (1). Denote by P_S the set of all possible permutation results on S. For the decoder, we define the permutation-result-to-index function $\phi : P_S \rightarrow$

$\{0, \ldots, |S| - 1\}$, where ϕ (σ) is the index of the permutation result σ in R. It can be computed as follows: let $\sigma_1,\ldots,\sigma_n = \sigma$ be the intensity values of σ, where σ_i is the intensity corresponding to the ith pixel (in lexicographic order). Then,

$$\begin{aligned}\phi(\sigma) &= \phi(\sigma_1, \ldots, \sigma_n) \\ &= \sum_{k=0}^{n-1} \text{number of permutations } \tau \text{ with prefix } \sigma_1 \ldots \sigma_k \text{ such that } \tau_{k+1} < \sigma_{k+1} \\ &= \sum_{k=0}^{n-1} \left[\sum_{I \in \{\sigma_{k+1}, \ldots \sigma_n\}, I < \sigma_{k+1}} \varphi(\{\sigma_{k+1}, \ldots, \sigma_n\}_{multi} - \{I\}_{multi}) \right]\end{aligned}$$

In pseudocode, ϕ is written as Algorithm 2.

In reverse, an encoder uses as its main subprocedure the index-to-permutation assignment function $\rho : \{0, \ldots, |S| - 1\} \to P_S$ which, given a target index integer, computes the permutation in the list R corresponding to said index. It avoids a full linear search by treating the possible permutation results as a multi-level tree, the level i nodes corresponding to the possible intensities of the ith pixel of a permutation result (with previous pixels fixed via the ancestor nodes). φ, from Equation (1), computes the total number of permutations down any particular ith-level node, without requiring actual traversal. Thus, by looping through the ith level nodes in order of increasing intensity, and stopping before the cumulative number of permutations has exceeded the target index, the tree traversal corresponding to said index is recursively computed. The pseudocode description of this subprocedure is Algorithm 3.

Algorithm 2: ϕ, the function that computes the index in R of a given permutation result

```
   Data: intensities σ_1...σ_n of a Permissible Set of pixels, in lexicographic order
   Result: index: integer decoded from said intensities
1  Function φ(intensities σ_1...σ_n):
2      index = 0
3      for 0 ≤ k < n do
4          // D can be any ordered iterable type
5          Initialize D as σ_{k+2}, ..., σ_n. Sort and remove duplicates of D.
6          for I ∈ D do
7              if I ≥ σ_{k+1} then
8                  Break
9              end
10             // φ as defined in Equation (1)
11             index = index + φ({σ_{k+1}, ..., σ_n}_multi − {I}_multi)
12         end
13     end
14     return index
```

Algorithm 3 is presented in a simpler form; various optimizations are possible. Notably, $\text{rem}_{uniq}^{sorted}$ can be precomputed once, and passed on to all recursive calls, so that a sort need not be performed each recursive call. The loop would then skip any instance where τ is no longer present in rem_{all}. A similar optimization would result if the multiset rem_{all} were implemented as a sorted, mutable array of intensities and counts, in which case line 8 would be made redundant.

Algorithm 3: ρ, the function that computes a permutation result of a given index in R

Data:

1. targetIndex: a number to encode as a permutation
2. $\mathbf{rem}_{all}$: size S multiset of all intensities within a Permissible Set of pixels

Result: a length S string of intensities, encoding the integer targetIndex

Function $\rho(int\ targetIndex,\ \mathbf{rem}_{all})$**:**
  return $\rho_{recursive}(targetIndex, \varepsilon, 0, \mathbf{rem}_{all})$

Function $\rho_{recursive}(int\ targetIndex,\ prefix,\ \mathbf{currentIndex},\ \mathbf{rem}_{all})$**:**
  if $\mathbf{rem}_{all} = \emptyset$ **then**
    return prefix
  end

  // $\mathbf{rem}_{uniq}^{sorted}$ can be any iterable type
  $\mathbf{rem}_{uniq}^{sorted} = \mathbf{rem}_{all}$, in sorted order, with duplicates removed

  for $\tau \in \mathbf{rem}_{uniq}^{sorted}$ **do**
    // Remove τ from $\mathbf{rem}_{all}$ to form $\mathbf{rem}_{\tau}^{minus}$
    $\mathbf{rem}_{\tau}^{minus} = \mathbf{rem}_{all} - \{\tau\}_{\text{multi}}$

    // index assuming we skipped to the next token
    $\mathbf{index}_{\tau}^{skipping} = \mathbf{currentIndex} + \varphi\left(\mathbf{rem}_{\tau}^{minus}\right)$

    // Does skipped index exceed target (i.e. does the target actually lie within current token?)
    if $\mathbf{index}_{\tau}^{skipping} > targetIndex$ **then**
      // Recurse, appending current token to **prefix**
      return $\rho_{recursive}(targetIndex,\ \mathbf{prefix} * \tau, \mathbf{currentIndex}, \mathbf{rem}_{\tau}^{minus})$
    end

    // Otherwise we're skipping to next token
    $\mathbf{currentIndex} = \mathbf{index}_{\tau}^{skipping}$
  end

2.3 *Time Complexity*

We estimate the time complexity of the various algorithms. Let B denote the block size in pixels (for example, $B = 4$ for a 2×2 block).

The formula ϕ, given by Equation (1), plays a part in several algorithms. If we define $n = k_1 + ... + k_j$, the time complexity is $O(n)$, which in practice is $O(B)$. In the numerator, the number of terms being added is at most n, attained when $k_1 = ... = k_j = 1$. Since $k_1 + ... k_j = n$, the number of terms multiplied to attain the factorial is at most n as well. For the denominator, since again, $k_1 + ... k_j = n$ the number of terms being multiplied in all the $k_i!$ terms is at most n, as is the product itself.

For Algorithm 1, note that $O(m) = O(B)$. The costliest operation is the computation of p in line 9, which has an outer loop (with outer variable w) of size m. The contents of each loop include the computation of $M(p)$, costing $O(m)$, along with a summation of $O(m)$ terms, each sum term of which costs $O(1)$ when the multiset is implemented as a hash set. Overall, the total cost is $O(m^2) = O(B^2)$. Note that Algorithm 1 is only a greedy subprocedure of the permissible set search. In the worst case, a block may consist of permissible set singletons, forcing $O(B)$ calls to this subprocedure, and making the total time complexity of the permissible set search $O(B^3)$.

For Algorithm 2, note that $O(n) = O(B)$. Within an outer loop of size n, the costliest operation is the inner loop of size $O(n)$. Within this inner loop is an application of ϕ costing $O(n)$. Overall, the total time complexity is $O(n^3) = O(B^3)$.

For Algorithm 3, note that the original size of $\mathbf{rem}_{all}$ is $O(B)$, with at least one element removed per recursive call: this can be treated as an outer loop of size $O(B)$. The actual loop, thought of as an inner loop, is over $\mathbf{rem}^{sorted}{}_{uniq}$, which is of size $O(B)$. The costliest operation in this actual loop is an $O(B)$ call to ϕ. Overall, the total time complexity is $O(B^3)$.

Finally, we address the time complexity for the steganographic algorithm as a whole. Let M denote the message size. Although pathological cases are possible (for example, the algorithm would be unusable on an image consisting of all the

same pixel intensity), we expect the number of blocks needed to encode/decode the message to be $O(M)$. Given this, the traversal of the blocks also has amortized $O(M)$ time complexity: the elements of a van der Corput sequence can be computed in amortized $O(1)$ time, and an optimized hash set to ensure block indices are not repeated can also be populated and checked in amortized $O(1)$ time, per element. All other operations on the block level are no worse than the calls to the above subprocedures, which cost $O(B^3)$. Overall, the algorithm's time complexity amortizes to $O(MB^3)$, both for encoder and decoder.

2.4 Computation Time

Table 1 lists the average computation times, both of our algorithm and of [35], for the MSRCv2 [39] and NoisyOffice [36–38] datasets. Tests were run on an Intel 7th Generation Core i7 notebook computer. The proposed algorithm is approximately 2-3 times slower for the measured datasets, which is a trade-off for its reduced detectability. Both algorithms, however, have an acceptable runtime for typical usage, particularly if only a few images need to be processed.

Table 1. Average computation time per dataset image (in seconds) for the proposed algorithm, as well as [35].

Dataset	(Median Width, Median Height)	Average Time ([35])	Average Time (proposed algorithm)
MSRCv2	(213,320)	0.16	0.38
NoisyOffice	(1300,1535)	2.37	8.59

2.5. Training and Testing

The steganography methods are trained and tested on two datasets: the NoisyOffice [36–38] dataset of scanned documents, and the MSRCv2 [39] dataset of more general images. The two datasets are evaluated separately. The NoisyOffice dataset contains "RealNoisyOffice" and "SimulatedNoisyOffice" subsets; the former is further partitioned into "single resolution" and "double resolution" images. All tests in this paper are run on the "single resolution" images of the "RealNoisyOffice" subset.

For each tested bit-per-pixel rate, the two datasets are augmented with a duplicated copy of the images. A message of random bits, simulating encrypted data, is embedded in each duplicate image using the steganography method being tested.

For each image–with or without an encoded message–the Wavelet Motion Analyzer [34] is used to create a feature. As in [35], a Fischer linear classifier is used to discriminate between features resulting from images containing, or not containing, a message.

10-fold cross-validation is used to train and evaluate said classifier. The Area Under the Receiver Operating Curve (AROC) is used to evaluate the detectability of the steganography method–the average AROC across the ten folds is taken.

In addition to the Wavelet Motion Analyzer, the algorithms are also compared using Sample Pair Analysis [25] and Peak Signal Noise Ratio (PSNR) [3].

3. Results and Discussion

We measure the detection rates for our algorithm. Results are shown in Table 2 and Table 3, for the NoisyOffice dataset of scanned images [36–38] and the MSRCv2 dataset of more generic images [39], respectively. The main measurement of detectability is the Area under the Receiver Operating Curve (AROC). The lower the area, the less detectable the corresponding steganographic method. Our method has shown low detectability, for given bit-per-pixel encoding rates, as well as allowing for higher encoding rates at the cost of detectability.

We compare our own algorithm with a prior permutational algorithm, of which the best performing is the pixel swap-based algorithm of [35]. We implement the algorithm of [35] in order to directly compare performance using the same sample images. Certain parameters of [35] were open to specification. To ensure that their algorithm effectiveness was being favorably estimated, we tested using a range of open parameters for each bit-per-pixel rate, taking the best-performing result for each such rate. This comprehensive approach provides an ideal-performance measure, one perhaps even better than would ordinarily be realistic (as parameters would ordinarily be fixed, rather than the best-performing ones chosen). This approach is not used for our own algorithm, which is left with no unfixed parameters.

Fig. 5 and Fig. 6 show graphs of AROC results against bit-per-pixel rates, for the respective datasets. Tested in each graph is the algorithm of [35] (under the aforementioned optimized parameters) along with our own method, under a 2×2 block size and varying choices of the Max Intensity Difference parameter. (The number of data points varies per steganographic method, as some do not support higher bit-per-pixel rates.)

The results corresponding to [35] are shown in Table 4 and Table 5, for the NoisyOffice and MSRCv2 datasets, respectively. These may be compared to our own results in Table 2 and Table 3. For both datasets, our method is shown to

be less detectable for each tested bit-per-pixel rate, under the tested Max Intensity Difference parameters. For the NoisyOffice dataset in particular, our proposed algorithm has proved to be far less detectable, by as much as 49%, as measured by AROC. For the MSRCv2 dataset, our algorithm is still less detectable, but the improvement is not to as great a degree.

Both algorithms, we found, are uniformly more detectable under the NoisyOffice dataset. The reasons for this–and, in particular, for the high detectability of [35] with this dataset–are unclear, and to be followed up upon. Possibilities range from NoisyOffice's higher resolution, to unexpected sources of noise within the MSRCv2 dataset. An answer to this question may help to characterize the images best suited for comparable steganographic methods.

In addition to the Wavelet Motion Analyzer method, two additional steganalysis systems were tested: Peak Signal Noise Ratio (PSNR) and Sample Pair Analysis. PSNR measures the change made to a stego-image [3] and ranges between 0 and 100; higher values are preferred. Sample Pair Analysis outputs a value estimating the length of a hidden message, as a proportion of the image; values closer to zero are preferred. As with the Wavelet Motion Analyzer test, we took the best result of [35] for a range of unspecified parameters, in order to favorably estimate their algorithm effectiveness. The proposed algorithm has a better PSNR for the same bit-per-pixel embedding rates (the precise difference depending on the Max Pixel Intensity Difference parameter). The Sample Pair Analysis results were roughly comparable, both algorithms' estimated hidden message lengths being close to zero. See Table 4 and Table 5, for results corresponding to [35]; our own results are found in Table 2 and Table 3.

As our algorithm improves upon [35], it appears to provide the best detectability results within the class of permutation-based steganographic methods. The trade-off is a longer operation time per message length, which may make the algorithm less usable for some applications. This is not a concern for a reasonable number of images. (Additionally, overly long messages, requiring a large number of cover images, may draw an adversary's attention regardless).

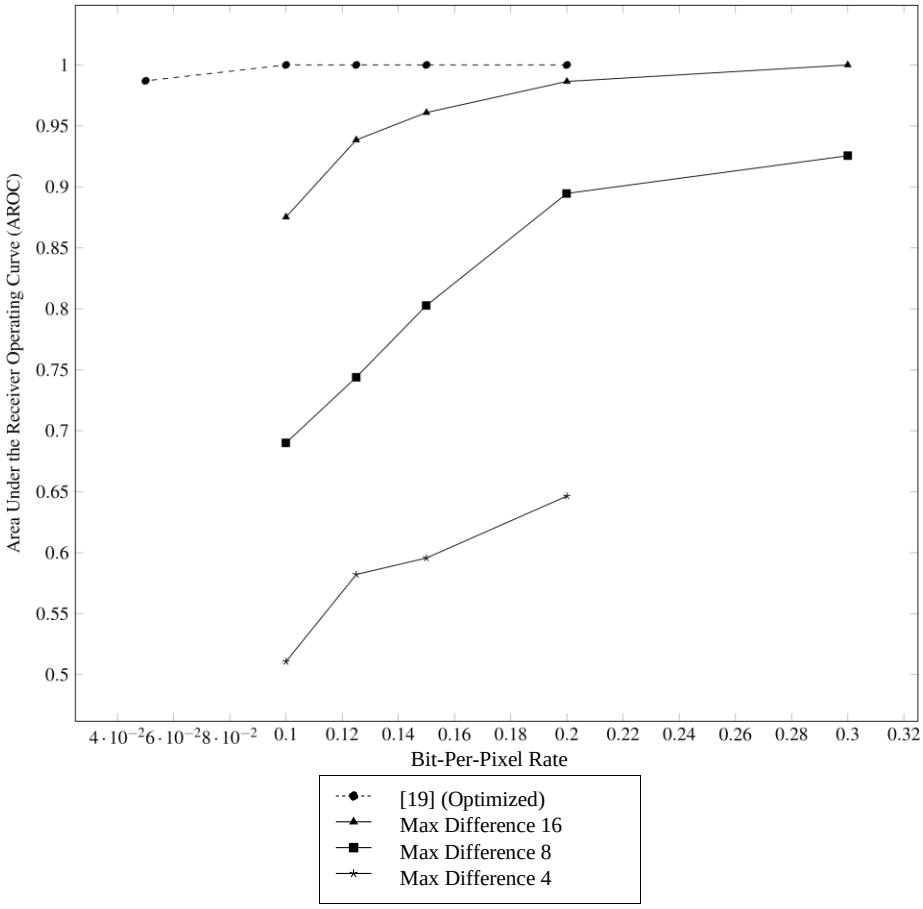


Fig. 5. Area Under the Receiver Operating Curve (AROC) results graphed against bit-per-pixel rate for the NoisyOffice dataset [36-38]. The lower the AROC, the less detectable the steganography method.

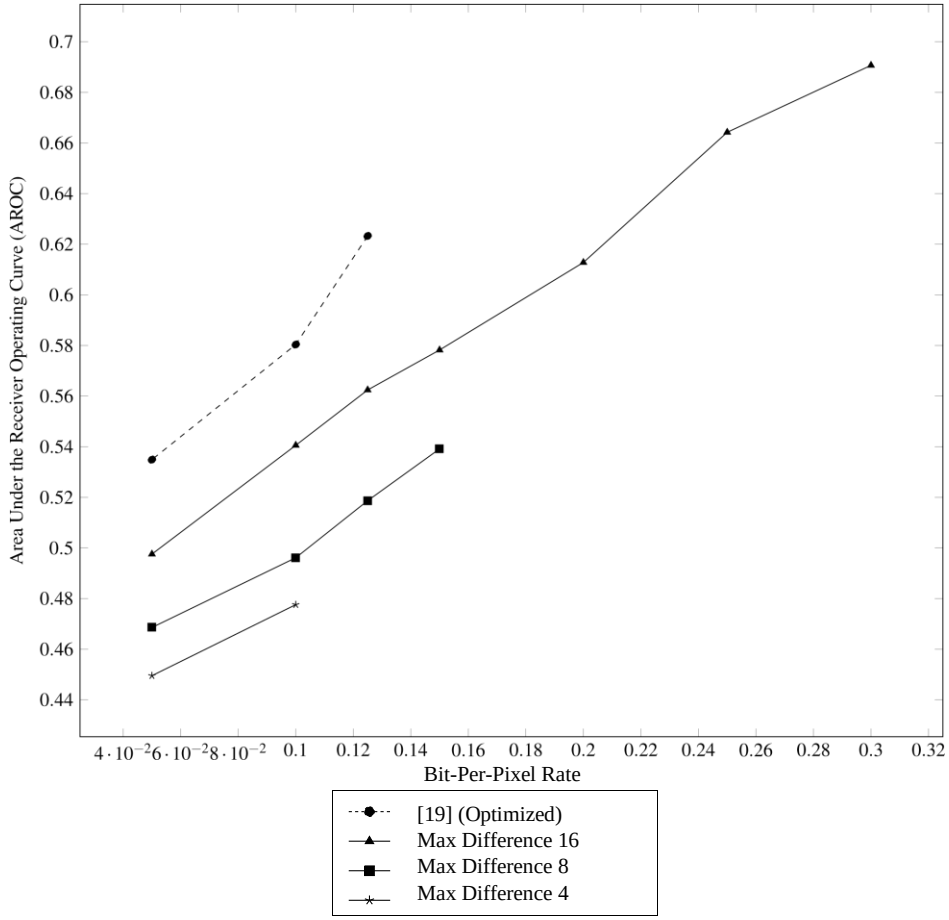
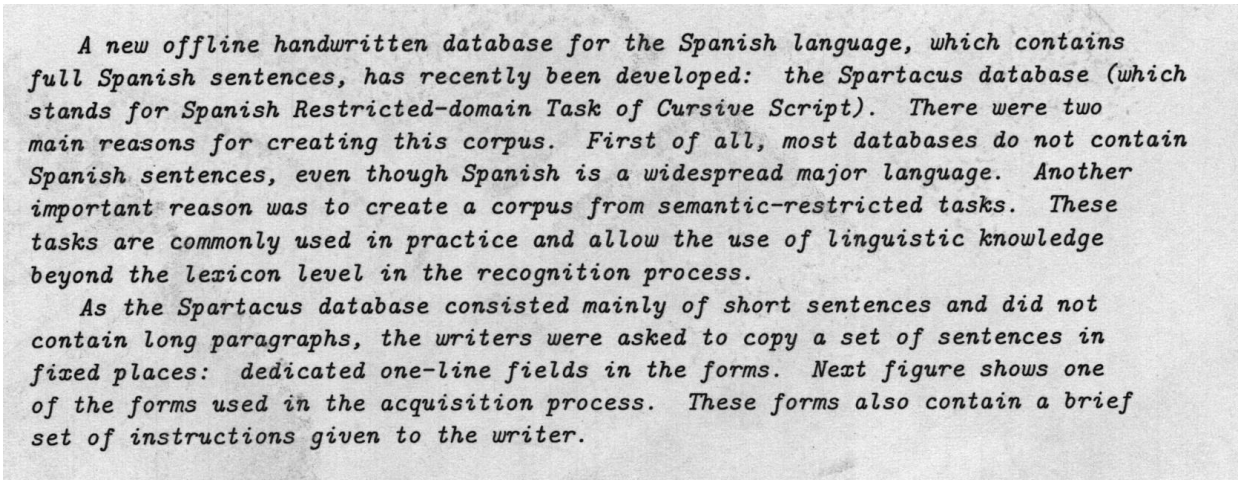


Fig. 6. Area Under the Receiver Operating Curve (AROC) results graphed against bit-per-pixel rate for the MSRCv2 dataset [39]. The lower the AROC, the less detectable the steganography method.


A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two main reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, even though Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used in practice and allow the use of linguistic knowledge beyond the lexicon level in the recognition process.

As the Spartacus database consisted mainly of short sentences and did not contain long paragraphs, the writers were asked to copy a set of sentences in fixed places: dedicated one-line fields in the forms. Next figure shows one of the forms used in the acquisition process. These forms also contain a brief set of instructions given to the writer.

Fig. 7. Sample cover image Fontfte_Noisep_RE.bmp (modified Jan 2021 to be a grayscale bitmap) from the NoisyOffice Dataset [36–38]

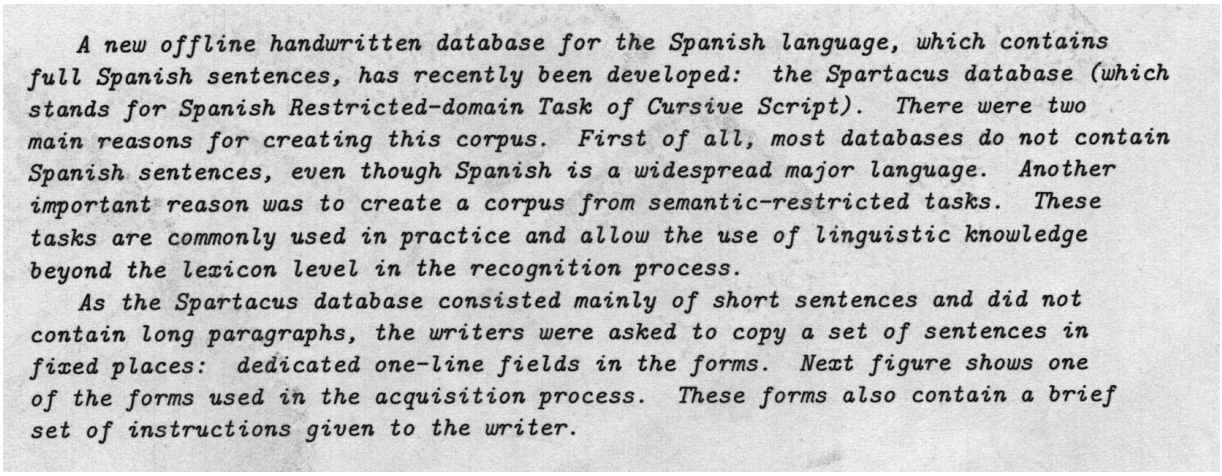


Fig. 8. Sample modified image Fontfte_Noisep_RE.bmp from the NoisyOffice Dataset [36–38] (modified Jan 2021). Parameters: Max Intensity Difference: 8, Bit-per-pixel rate: 0.30

Fig. 9. Sample cover image 9_1_s.bmp (modified Jan 2021 to be a grayscale bitmap) from the MSRCv2 dataset [39]

Fig. 10. Sample modified image 9_1_s.bmp from the MSRCv2 dataset [39] (modified Jan 2021). Parameters: Max Intensity Difference: 8, Bit-per-pixel rate: 0.30

Table 2. Results for the proposed algorithm, tested on the NoisyOffice dataset [36–38]. The second column contains the result of the Fischer Wavelet Motion Analyzer under 10-fold cross-validation, as measured by the Area Under the Receiver Operating Curve (AROC). Peak Signal Noise Ratio (PSNR) and Sample Pair Analysis results have also been included.

Max Pixel Intensity Difference	Embedding (bit/pixel) rate	AROC (Average Fold Result)	PSNR	Sample Pair Analysis
16	0.3	1.00	37.4	0.02
16	0.2	0.99	39.1	0.02
16	0.15	0.96	40.4	0.02
16	0.125	0.94	41.2	0.03
16	0.1	0.88	42.1	0.02
8	0.3	0.93	41.0	0.02
8	0.2	0.89	42.8	0.02
8	0.15	0.80	44.0	0.02
8	0.125	0.74	44.8	0.02
8	0.1	0.69	45.8	0.02
4	0.2	0.65	46.9	0.02
4	0.15	0.60	48.1	0.02
4	0.125	0.58	48.9	0.02
4	0.1	0.51	49.9	0.02

Table 3. Results for the proposed algorithm, tested on the MSRCv2 dataset [31]. The second column contains the result of the Fischer Wavelet Motion Analyzer under 10-fold cross-validation, as measured by the Area Under the Receiver Operating Curve (AROC). Peak Signal Noise Ratio (PSNR) and Sample Pair Analysis results have also been included.

Max Pixel Intensity Difference	Embedding (bit/pixel) rate	AROC (Average Fold Result)	PSNR	Sample Pair Analysis
16	0.3	0.69	38.5	0.01
16	0.25	0.66	39.3	0.01
16	0.2	0.61	40.3	0.01
16	0.15	0.58	41.5	0.01
16	0.125	0.56	42.3	0.01
16	0.1	0.54	43.3	0.01
16	0.05	0.50	46.3	0.01
8	0.15	0.54	45.3	0.01
8	0.125	0.52	46.1	0.01
8	0.1	0.50	47.0	0.01
8	0.05	0.47	50.1	0.01
4	0.1	0.48	51.1	0.01
4	0.05	0.45	54.1	0.01

Table 4. Results for the algorithm of [35], tested on the NoisyOffice dataset [36–38]. The second column contains the result of the Fischer Wavelet Motion Analyzer under 10-fold cross-validation, as measured by the Area Under the Receiver Operating Curve (AROC). Peak Signal Noise Ratio (PSNR) and Sample Pair Analysis results have also been included. For each embedding rate, the highest score of [35] (across all parameters) is chosen.

Embedding (bit/pixel) rate	AROC (Average Fold Result)	PSNR	Sample Pair Analysis
0.2	1.00	36.5	0.02
0.15	1.00	37.7	0.02
0.125	1.00	38.5	0.02
0.1	1.00	39.8	0.02
0.05	0.99	43.1	0.02

Table 5. Results for the algorithm of [35], tested on the MSRCv2 dataset [39]. The second column contains the result of the Fischer Wavelet Motion Analyzer under 10-fold cross-validation, as measured by the Area Under the Receiver Operating Curve (AROC). Peak Signal Noise Ratio (PSNR) and Sample Pair Analysis results have also been included. For each embedding rate, the highest score of [35] (across all parameters) is chosen.

Embedding (bit/pixel) rate	AROC (Average Fold Result)	PSNR	Sample Pair Analysis
0.125	0.62	39.2	0.00
0.1	0.58	40.1	0.00
0.05	0.53	43.4	0.00

4. Conclusions

We study steganographic methods based on permutation of pixels in grayscale images. Our method generalizes from pixel swapping to allow for permutations within a region of an image. Through both direct reimplementation and indirect comparison, our algorithm was found to be less detectable and to have a higher maximum bit-per-pixel rate, than other steganographic methods, noise and swap-based. In particular, reduced detectability was demonstrated against [35] for the NoisyOffice and MSRCv2 datasets, using the Wavelet Motion Analyzer-based detector. Both the reduced detectability and improved bit-per-pixel rate further the goal of safe, and subtle message-passing via steganography. Detectability was particularly reduced for the NoisyOffice dataset, suggesting further follow-up to characterize the kinds of images best suited to the various steganographic algorithms.

References

[1] A. F. Nilizadeh and A. R. N. Nilchi, "Steganography on rgb images based on a "matrix pattern" using random blocks.," *International Journal of Modern Education & Computer Science*, vol. 5, no. 4, 2013.

[2] M. Gokul, R. Umeshbabu, S. K. Vasudevan, and D. Karthik, "Hybrid steganography using visual cryptography and lsb encryption method," *International Journal of Computer Applications*, vol. 59, no. 14, 2012.

[3] R. Sharma, R. Ganotra, S. Dhall, and S. Gupta, "Performance comparison of steganography techniques," *International Journal of Computer Network and Information Security*, vol. 10, no. 9, pp. 37–46, 2018.

[4] O. Hosam, "Attacking image watermarking and steganography-a survey," *International Journal of Information Technology and Computer Science*, vol. 11, no. 3, pp. 23–37, 2019.

[5] P. Selvigrija and E. Ramya, "A survey on steganography using multimedia files," *International Journal of Computer Science Engineering (IJCSE)*, vol. 4, no. 02, Mar. 2015.

[6] H. I. Alsaadi, M. K. Al-Anni, R. M. Almuttairi, O. Bayat, and O. N. Ucan, "Text steganography in font color of ms excel sheet," in *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*, 2018, pp. 1–7.

[7] R. B. Krishnan, P. K. Thandra, and M. S. Baba, "An overview of text steganography," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, IEEE, 2017, pp. 1–6.

[8] A. A. Ali, and A. H. Seddik., "New text steganography technique by using mixed-case font," *International Journal of Computer Applications*, vol. 62, no. 3, 2013.

[9] W. Bhaya, "Text Steganography based on Font Type in MS-Word Documents," *Journal of Computer Science*, vol. 9, pp. 898–904, June 2013.

[10] S. T. A. Shah and A. Khan, "Text steganography using character spacing after normalization," *International Journal of Scientific and Engineering Research*, vol. Vol 11, p. 949, Feb. 2020. DOI: 10.14299/ijser.2020.02.05.

[11] L. Y. Por and B. Delina, "Information hiding: A new approach in text steganography," in *WSEAS international conference. Proceedings. Mathematics and Computers in Science and Engineering*, World Scientific, Engineering Academy, and Society, vol. 7, 2008.

[12] S. Rico-Larmer, "Cover text steganography: N-gram and entropy-based approach," in proceedings of the of the Conference on Cybersecurity Education, Research and Practice, Oct. 2016.

[13] S. Roy and M. Manasmita, "A novel approach to format based text steganography," in *proceedings of the 2011 international conference on communication, Computing & Security*, 2011, pp. 511–516.

[14] P. Jayaram, H. Ranganatha, and H. Anupama, "Information hiding using audio steganography - A survey," *The International Journal of Multimedia & Its Applications (IJMA)*, vol. 3, pp. 86–96, 2011.

[15] F. Djebbar, B. Ayad, K. A. Meraim, and H. Hamam, "Comparative study of digital audio steganography techniques," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2012, no. 1, pp. 1–16, 2012.

[16] K. Gopalan, "Audio steganography using bit modification," in *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, IEEE, vol. 1, 2003, pp. I–629.

[17] S. Elshare and N. N. El-Emam, "Modified multi-level steganography to enhance data security," *International Journal of Communication Networks and Information Security*, vol. 10, no. 3, p. 509, 2018.

[18] R. J. Mstafa and K. M. Elleithy, "An efficient video steganography algorithm based on bch codes," Asee, 2015.
[19] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal processing*, vol. 90, no. 3, pp. 727–752, 2010.
[20] M. S. Subhedar and V. H. Mankar, "Current status and key issues in image steganography: A survey," *Computer science review*, vol. 13, pp. 95–113, 2014.
[21] B. Li, J. He, J. Huang, and Y. Q. Shi, "A survey on image steganography and steganalysis," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 2, no. 2, pp. 142–172, 2011.
[22] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. Ho, and K.-H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, vol. 65, pp. 46–66, 2018.
[23] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," *Proceedings of the 2001 Workshop on Multimedia and Security: New Challenges*, Oct. 2002. DOI: 10.1145/1232454.1232466.A.
[24] D. Ker, "Improved detection of LSB steganography in grayscale images," in *International workshop on information hiding*, Springer, 2004, pp. 97–115.
[25] S. Dumitrescu, X. Wu, and Z. Wang, "Detection of LSB steganography via sample pair analysis," in *International workshop on information hiding*, Springer, 2002, pp. 355–372.
[26] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color, and gray-scale images," *IEEE multimedia*, vol. 8, no. 4, pp. 22–28, 2001.
[27] D. Neeta, K. Snehal, and D. Jacobs, "Implementation of LSB steganography and its evaluation for various bits," in *2006 1st International Conference on Digital Information Management*, IEEE, 2006, pp. 173–178.
[28] S. M. Karim, M. S. Rahman, and M. I. Hossain, "A new approach for lsb based image steganography using secret key," in *14th international conference on computer and information technology (ICCIT 2011)*, IEEE, 2011, pp. 286–291.
[29] A. Buchaev, A. Mustafaev, V. Galyaev, and A. Bagandov, "Increasing the steganographic resistance of the lsb data hide algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, 2021.
[30] A. Sur, P. Goel, and J. Mukhopadhyay, "A novel steganographic algorithm resisting. targeted steganalytic attacks on LSB matching," in *International Workshop on Digital Watermarking*, Springer, 2008, pp. 199–208.
[31] E. Franz and A. Schneidewind, "Pre-processing for adding noise steganography," in *International Workshop on Information Hiding*, Springer, 2005, pp. 189–203.
[32] P. Bas, "Natural steganography: cover-source switching for better steganography," arXiv:1607.07824v1 [cs.MM] 26 July. 2016.
[33] P. Goel, "Data hiding in digital images: A steganographic paradigm," *Indian Institute of Technology, Kharagpur*, 2008.
[34] M. Goljan, J. Fridrich, and T. Holotyak, "New blind steganalysis and its implications," *Proceedings of the SPIE*, vol. 6072, Feb. 2006. DOI: 10.1117/12.643254.
[35] A. Sur, P. Goel, and J. Mukhopadhyay, "Adaptive pixel swapping based steganography reducing embedding noise," Jun. 2011, pp. 299–304. DOI: 10.1007/978-3-642-21786-9_49.
[36] F. Zamora-Martnez, S. Espaa-Boquera, and M. Castro-Bleda, "Behaviour-based clustering of neural networks applied to document enhancement," in *International Work-Conference on Artificial Neural Networks*, Springer, 2007, pp. 144– 151.
[37] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.
[38] M. Castro-Bleda, S. Espana-Boquera, J. Pastor-Pellicer, and F. Zamora-Mart ́nez, *The NoisyOffice Database: A corpus to train supervised machine learning filters for image processing*, 2019.
[39] A. Criminisi, *Microsoft Research Cambridge MSRC-v2 Image Database (21 object classes)*, 2005.
[40] L. Kuipers and H. Niederreiter, *Uniform distribution of sequences*. Courier Corporation, 2012.
[41] R. A. Brualdi, *Introductory combinatorics*. Pearson Education India, 1977.
[42] R. P. Stanley, *Enumerative combinatorics*. Cambridge University Press, vol.1, 2011.

Authors' Profiles

Saitulaa Naranong is a Ph.D. candidate in Computer Science at the National Institute of Development Administration (NIDA). He received a Master degree in Mathematics from Texas A&M University, and a bachelor degree from Middlebury College, U.S.A., with a major in Mathematics and minor in Computer Science. He is a lecturer in the Department of Mathematics and Statistics, Faculty of Science and Technology at Thammasat University, Thailand. His research interests includes applications of machine learning and computer security.

Surapong Auwatanamongkol received a Bachelor degree in Electrical Engineering from Chulalongkorn University, Thailand, a Master degree in Computer Science from Georgia Institute of Technology, USA, and a doctoral degree of Computer Science from Southern Methodist University, USA. He is an associate professor at the school of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand. His current research interests include Evolutionary Computation, Machine Learning, Image and Data Analytics.

How to cite this paper: Saitulaa Naranong, Surapong Auwatanamongkol, " A Block Permutational Steganographic Algorithm for Scanned Documents and other Images", International Journal of Modern Education and Computer Science(IJMECS), Vol.13, No.5, pp. 42-57, 2021.DOI: 10.5815/ijmecs.2021.05.05