Binary image steganography based on permutation

Juvet Karnel Sadié¹,²,³ · Stéphane Gael R. Ekodeck¹,²,³ · René Ndoundam¹,²,³

Abstract
Gray and color images contain much space for hiding information, and modification of pixel values within a certain range does not cause the image to appear abnormal. However, pixels in binary images possess only two states: black and white. As a result, distortions of binary images can be easily detected. For this reason, hiding information in binary images is more challenging. In this paper, we propose a steganographic scheme based on permutations, which improves the capacity of embedding information in a series of $p$ host binary images. Consider a steganographic scheme $T$ that can hide $p \times Q(m, n)$ bits of data in $p$ binary images, where $Q(m, n)$ represents the embedding capacity of $T$ for an $m \times n$ binary image block. Our scheme takes advantage of permutation to improve this capacity for $p$ binary images so that, instead of $p \times Q(m, n)$ bits, it can hide $p \times \left\lfloor \log_2(p) \right\rfloor + p \times Q(m, n)$ bits. The results obtained by experiments show that our model performs a better hiding process in terms of hiding capacity.

Keywords Steganography · Cover medium · Binary image · Permutation · Embedding capacity · Brute-force attack

1 Introduction

With the development of information technology, a large amount of multimedia data is exchanged on the internet and social media platforms. These data are frequently copied, edited, and distributed by other users and used for purposes other than those intended by the original creator, causing security issues. Information security protects sensitive information from unauthorized activities, including inspection, modification, recording, disruption and destruction. The goal is to ensure the safety and privacy of critical data. To achieve this purpose, information security methods use encryption and data hiding. Cryptography is a process that converts information into an unreadable format called a cipher text file by using encryption techniques [1]. Data hiding methods can be classified into two domains: steganography and watermarking. Digital watermarking is a method of embedding data into digital multimedia content. These data are used to verify the credibility of the content or to confirm the identity of the digital content’s owner. It can be used for multiple purposes, such as copyright protection and source tracking. The word steganography is of Greek origin and means covered writing. It is the process of hiding one message within another (a cover medium) such as a web page, image, or text so that the presence of the hidden message is indiscernible. When a message is hidden in a cover medium, the resulting medium is called a stego-object. The key concept behind steganography is that the message to be transmitted should not be detectable to the naked eye. From its definition, steganography is used to ensure data confidentiality, like encryption. However, the main difference between these methods is that with encryption, anyone can see that the two parties are communicating in secret. Steganography hides the existence of a secret message, and in the best case, no one can detect the presence of the message. When they are combined, steganography and encryption can provide better security.

Steganography requires three features: security, capacity, and robustness [2]. Capacity refers to the amount of information that can be hidden in the medium, whereas security is important when a secret communication is to be kept secret and...
undetectable by eavesdroppers. Last, robustness can be characterized as the amount of modification the stego-medium can withstand before an adversary can destroy the hidden information. Information security is summarized in Fig. 1. Text, image, audio, and video files are used in steganography as cover media. In particular, images are the most popular cover medium [3]. Gray and color images contain much space for hiding information, and modification of pixel values within a certain range does not cause the image to appear abnormal. With the more widespread application of digital technology, much important information, such as personal records, medical records, certificates, handwritten signatures, design drawings, and collections of books, has been scanned into digital documents and stored as binary images. This means that hiding data in binary images has great application potential [4]. However, pixels in binary images possess only two states: black(1) and white(0). As a result, distortions on binary images can be easily detected even by human eyes [4, 5]. For this reason, hiding information in binary images is more challenging.

In this paper, we propose a steganographic scheme based on permutations, which improves the capacity of embedding information in a series of $p$ host binary images. Consider a steganographic scheme $T$ that can hide $p \times Q(m, n)$ bits of data in $p$ binary images, where $Q(m, n)$ represents the embedding capacity of $T$ for an $m \times n$ binary image block. Our scheme takes advantage of permutation to improve this capacity for $p$ binary images so that, instead of $p \times Q(m, n)$ bits, it can hide $p \times \left\lfloor \log_2(p) \right\rfloor + p \times Q(m, n)$ bits. As a data sample for experimentation, we used images of 512 $\times$ 512 bits, as used in Table 3 of [4]. The results obtained show that our model performs a better hiding process in terms of hiding capacity.

The rest of the paper is organized as follows: in Sect. 2, we present some preliminaries that lead us to the design of our scheme. Then, we present the basic idea in Sect. 3. Section 4 presents our scheme. The experimental results, comparisons, and discussion are given in Sect. 5. The security of the scheme is shown in Sect. 6 and finally, Sect. 7 is the conclusion.

2 Preliminaries

This section presents an overview on permutation generation methods and on binary image steganography.

2.1 Permutation generation methods

Permutation is one of the most important combinatorial objects in computing and can be applied in various applications, for example, scheduling problems. Permutation generation can form the basis of a backtracking program to solve any problem involving reordering a set of items. It is well-known that, for $n$ distinct items, the total number of permutations is $n!$. Permutation generation has a long history. Surveys in the field have been published in 1960 by Lehmer [6]. Among many publications on permutations generation methods, Myrvold and Ruskey [7] proposed a ranking function (Algorithm 2) for the permutations on $n$ symbols which assigns a unique integer in the range $[0, n! - 1]$ to each of the $n!$ permutations. Also, they proposed an unranking function (Algorithm 1) for which, given an integer $r$ between 0 and $n! - 1$, the value of the function is the permutation of rank $r$.

2.1.1 Unranking function

First of all, recall that a permutation of order $n$ is an arrangement of $n$ symbols. An array $\pi[0 \ldots n - 1]$ is initialized to the identity permutation $\pi[i] = i$, for $i = 0, 1, \ldots, n - 1$.

**Algorithm 1 :** Unranking function

```plaintext
Procedure unrank($n, r, \pi$)[7]
begin
if $n > 0$ then
    swap($\pi[n - 1], \pi[r \mod n]$);
    unrank($n - 1, [r/n], \pi$);
end;
end;
```

Note: $swap(a, b)$ exchanges the values of variables $a$ and $b$.

2.1.2 Ranking function

To rank, first compute $\pi^{-1}$. This can be done by iterating $\pi^{-1}[\pi[i]] = i$, for $i = 0, 1, \ldots, n - 1$.

In the algorithm below, both $\pi$ and $\pi^{-1}$ are modified.

**Algorithm 2 :** Ranking function

```plaintext
function rank($n, \pi, \pi^{-1}$):integer[7]
begin
if $n = 1$ then return(0) end;
    $s := \pi[n - 1]$;
    swap($\pi[n - 1], \pi[\pi^{-1}[n - 1]]$);
    swap($\pi^{-1}[s], \pi^{-1}[n - 1]$);
    return($s + n.rank(n - 1, \pi, \pi^{-1})$);
end;
```

2.2 Overview on binary image steganography

Despite the difficulty related to their structures, binary images have been the subject of several works in steganography. The common objective pursued is to increase the loading capacity while limiting the distortion of the binary image. Jung et al. [8] proposed a data hiding method for binary images that relies on block masking to distribute keys to two parts.
and then authenticate the correct authorized part. The proposed method divides the cover image into small subblocks and designs key pairs that determine both where a bit is to be embedded and whether it is possible to embed it there. The key pairs are also required to extract the secret data from the stego image. Ding and Wang [4] constructed coding tables for data hiding and extraction based on HVS in suitable blocks of binary images based on the condition that the content of the image is not obviously changed. Dahiwal and Chhajed [9] proposed a binary image steganographic scheme that aims to minimize the embedding distortion on the texture. The complement of a binary image is taken first, and then the image needs to be rotated. After rotation, the mirroring process is used. The flipping distortion score of pixels is calculated by measuring the flipping distortion of corresponding pixels. Jung [10] proposed a method of hiding data in binary images by checking the bit position and parity bit to hide a secret bit in binary images. The cover image is divided into \( M \times N \) subblocks, and the parity bit of subblock pixels are referenced to embed a secret bit for the suitable pixel. By finding the suitable pixel position at which to insert a secret bit for each subblock, the image quality of the stego image can be improved, while maintaining low computational complexity. Wu and Hwang [11] proposed a data-hiding method based on combination theory. In their scheme, a secret position matrix is designed to improve the hiding capacity which is capable of preventing the least distortion based on the combination theory.

Pan et al. [12] defined a secure data hiding scheme for two-color images. Given a cover binary image \( F \), they partitioned it into blocks of fixed size \( m \times n \). Their method can hide as many as \( r \) bits of the guest message in each block by modifying at most 2 bits in it, where \( r \) is defined as \( r \leq \lceil \log_2 (mn + 1) \rceil \). Because it seems to be a well-known reference in the field, it has an interesting embedding capacity, and several works refer to it, we use their scheme to illustrate the performance of our method. We present their embedding algorithm (Algorithm 3) and their extraction algorithm (Algorithm 4) below.

### 2.2.1 Embedding algorithm

The basic idea of this scheme is to use a different binary operator to protect the secret key from being compromised and to use a weight matrix to increase the data rate while maintaining the high quality of the host image. To secure the information, they used a secret key with two components:

- \( K \): a randomly selected binary matrix of size \( m \times n \).
- \( W \): a weight matrix which is an integer matrix of size \( m \times n \). \( W \) satisfies the condition that \( \{(W)_{i,j} : i = 1 \ldots m, j = 1 \ldots n\} = \{1, 2, \ldots, 2^r - 1\} \).

Data hiding is achieved by modifying some bits of the cover image \( F \). Algorithm 3 and Algorithm 4 present their embedding and extraction algorithms.
Algorithm 3: Embedding Algorithm

**Input:** The key $K$ and $W$, the secret message to hide $b_1b_2\cdots b_r$
the host binary image $F_i$

**Output:** the stego binary image $F'_i$

**Begin**

1. Compute $F_i \oplus K$, where $\oplus$ is the bitwise exclusive-OR of two equal-size binary matrices.
2. Compute $SUM((F_i \oplus K) \otimes W)$, where $\otimes$ is the pair-wise multiplication of two equal-size matrices and $SUM$ is the sum of all elements in a matrix.
3. From the matrix $F_i \oplus K$, compute for each $w=1\cdots2^r-1$ the following set:
   
   $S_w = (j,k)((W)_{j,k} = w) \land (F_i \oplus K)_{j,k} = 0$)
   
   $\lor (j,k)((W)_{j,k} = 2^r - w) \land ((F_i \oplus K)_{j,k} = 1)$.

   Intuitively, $S_w$ is the set containing every matrix index $(j,k)$ such that, if we complement $[F_i]_{j,k}$, we can increase the sum in step 2 by $w$.

   There are actually two possibilities to achieve this:
   
   i) if $(W)_{j,k} = w$ and $[F_i \oplus K]_{j,k} = 0$, then complementing $[F_i]_{j,k}$ will increase the weight by $w$.
   
   ii) if $(W)_{j,k} = 2^r - w$, and $[F_i \oplus K]_{j,k} = 1$, then complementing $[F_i]_{j,k}$ will decrease the weight by $2^r - w$ or, equivalently, increase the sum by $w$ (under mod $2^r$).

   Also, define $S_w = S'_w$ for any $w \equiv w' \pmod{2^r}$.

3. Define a weight difference $d \equiv (b_1b_2\cdots b_r) - SUM((F_i \oplus K) \otimes W) \pmod{2^r}$.

   If $d = 0$, there is no need to change $F_i$. Otherwise, we run the following steps to transform $F_i$ to $F'_i$:

   a) randomly pick an $h \in \{1, 2, \ldots, 2^r - 1\}$
   
   b) such that $S_{hd} \neq \emptyset$ and $S_{(h-1)d} \neq \emptyset$.
   
   c) randomly pick a $(j,k) \in S_{hd}$ and complement the bit $[F_i]_{j,k}$.
   
   d) randomly pick a $(j,k) \in S_{(h-1)d}$ and complement the bit $[F_i]_{j,k}$.

**end;**

2.2.2 Extraction algorithm

To retrieve the secret message, the process is as follows.

Algorithm 4: Extraction Algorithm

**Input:** The key $K$ and $W$ and the stego binary image $F'_i$

**Output:** the secret message $b_1b_2\cdots b_r$

**Begin**

compute $SUM((F'_i \oplus K) \otimes W) \pmod{2^r}$ to find the hidden bit stream $b_1b_2\cdots b_r$

**end;**

2.2.3 Security

In [12], Pan et al. identified some possible attacks on their scheme and their cost. From their discussion, a brute-force attack is quite impossible, since there are $2^{mn}$ and $C_{2^r-1}^{mn} \times (2^r - 1)! \times (2^r - 1)^{mn - (2^r - 1)}$ combinations for $K$ and $W$.

Next, considering the chosen-plaintext attack, which uses a differential technique to reduce the search range of $W$, that attack has a very high cost, as long as the block size $(m \times n)$ is reasonably large and the secrecy of $K$ and $W$ is maintained.
3 Basic idea

Before the presentation of our scheme, we first present in this section, the basic idea.
Let us consider three binary images of size $m \times n$ each, denoted by $im_1$, $im_2$ and $im_3$, in which we want to hide the secret message $m$. Additionally, we denote by $T$ the scheme proposed by Pan et al. [12], where given a binary image of size $m \times n$, $T$ can hide up to $\left\lfloor \log_2 (mn+1) \right\rfloor$ bits in that image. For the three images, $T$ can hide up to $3 \times \left\lfloor \log_2 (mn+1) \right\rfloor$ bits.

We propose a scheme that improves on this embedding capacity. The scheme is based on the sending order of the images between the sender (A) and the receiver (B). Without loss of generality, let us consider the function $f$ and its inverse $g$.

$$
\begin{align*}
\text{f}(3, 1, 2) &= 0 \\
\text{f}(2, 3, 1) &= 1 \\
\text{f}(2, 1, 3) &= 2 \\
\text{f}(3, 2, 1) &= 3 \\
\text{f}(1, 3, 2) &= 4 \\
\text{f}(1, 2, 3) &= 5
\end{align*}
$$

and

$$
\begin{align*}
\text{g}(0) &= (3, 1, 2) \\
\text{g}(1) &= (2, 3, 1) \\
\text{g}(2) &= (2, 1, 3) \\
\text{g}(3) &= (3, 2, 1) \\
\text{g}(4) &= (1, 3, 2) \\
\text{g}(5) &= (1, 2, 3)
\end{align*}
$$

3.1 Embedding process

In this subsection, we present point by point the method to embed the secret $m$ in the three binary images $im_1$, $im_2$ and $im_3$, using $T$, the scheme proposed by Pan et al. [12].

- use the scheme of Pan et al. [12] to hide $w_1$ in $im_\alpha$, $w_2$ in $im_\beta$ and $w_3$ in $im_\gamma$.
- send the images in the order $im_\alpha$, $im_\beta$ and $im_\gamma$.

3.2 Retrieval process

This subsection presents how to extract the secret in a series of three binary images received in the order $im_\alpha$, $im_\beta$ and $im_\gamma$, $\alpha \in \{1, 2, 3\}$, $\beta \in \{1, 2, 3\}$, $\gamma \in \{1, 2, 3\}$:

- compute the number $i$ such that $i = f(\alpha, \beta, \gamma)$
- use the scheme of Pan et al. [12] to extract $w_1$ in $im_\alpha$, $w_2$ in $im_\beta$ and $w_3$ in $im_\gamma$
- compute the string $W = w_1 + w_2 + w_3$, where $+$ denotes the concatenation operation.
- compute $z = (W)_{10}$, the decimal representation of $W$.
- compute $x = z + i \times 2^{3 \times \left\lfloor \log_2 (mn+1) \right\rfloor}$
- retrieve the secret $m = (x_2)$, the binary representation of $x$.

3.3 Remarks

Taking into account the sending order of the three images, the decimal value of the secret that we can hide is between 0 and $3! \times 2^{3 \times \left\lfloor \log_2 (mn+1) \right\rfloor} - 1$. This means that the maximum value that can be hidden for three images is $6 \times 2^{3 \times \left\lfloor \log_2 (mn+1) \right\rfloor}$, which in binary is represented on

$$
\begin{align*}
\log_2 (6 \times 2^{3 \times \left\lfloor \log_2 (mn+1) \right\rfloor}) &= \left\lfloor \log_2 (6) \right\rfloor + \\
\log_2 (2^{3 \times \left\lfloor \log_2 (mn+1) \right\rfloor}) &= 2 + 3 \times \left\lfloor \log_2 (mn+1) \right\rfloor
\end{align*}
$$

In other words, for three images, we can hide $2 + 3 \times \left\lfloor \log_2 (mn+1) \right\rfloor$ bits instead of $3 \times \left\lfloor \log_2 (mn+1) \right\rfloor$ bits. For $p$ images we can hide up to $\left\lfloor \log_2 (p^5) \right\rfloor + p \times \left\lfloor \log_2 (mn+1) \right\rfloor$ bits.

Given $p$ binary images, Table 1 presents a comparison in terms of embedding capacity between the scheme of Pan et al. [12] and our scheme.

The following section formalizes this basic idea by presenting our scheme in the general case, where the number of images used is $N$ and the embedding scheme used is any embedding technique denoted by $T$.

4 Scheme design

In this section, we present the details of our scheme.
Let’s denote by $T$ an embedding technique for which given a binary image block $I$ of size $m \times n$, $T$ can hide $Q(m, n)$ data bits in $I$.
Initially, the two communicating parties must share a set of $N$ binary images of size $m \times n$ each, denoted by $i_1, i_2, \ldots, i_N$. Secondly, the set is divided into $s$ blocks of $p$ binary images. In other terms, the $N$ images are divided as:
(i_1, i_2, \ldots, i_p), \quad (i_{p+1}, i_{p+2}, \ldots, i_{2p}), \ldots, (i_{(s-1)p+1}, i_{(s-1)p+2}, \ldots, i_{sp}), \text{ with } N = sp.

This means:
The first block is: \( N_1 = (i_1, i_2, \ldots, i_p) \).
The second block is: \( N_2 = (i_{p+1}, i_{p+2}, \ldots, i_{2p}) \).
More generally, for the \( k \)th block, \( 1 \leq k \leq s, N_k = (i_{(k-1)p+1}, i_{(k-1)p+2}, \ldots, i_{kp}) \).
The value of \( p \) is also shared between the sender and the receiver.

**Algorithm 5:** Embedding algorithm

**Algorithm:** embedding

**Input:**
- \( N \): the set of binary images;
- \( p \): the size of images block;
- \( s \): the number of images block;
- \( M \): the secret message to embed;
- \( l \): the number of blocks of the secret message;
- \( \pi \): the initial permutation;
- \( Q(m, n) \): the number of bits to hide in a binary image of size \( m \times n \) bits;
- \( K \): the key related to \( T \);

**Output:**
- stego binary images of \( m \times n \) bits each;

**begin:**
1. Divide \( M \) into \( l \) blocks of \( b \) bits; \( b = p \times Q(m, n) + \log_2(p!) \);
2. Divide the \( N \) images into \( l \) blocks of \( p \) images.
3. for each block \( M_i, 1 \leq i \leq l \):
   a. Compute \( \text{val}_i = (M_i)_10 \), the decimal representation of \( M_i \);
   b. Compute the number \( N\text{perm}_i \) by \( N\text{perm}_i = \lfloor \text{val}_i / 2^\alpha \rfloor \), where \( \alpha = p \times Q(m, n) \);
   c. Compute \( \pi' = \text{unrank}(p, N\text{perm}_i, \pi) \), the permutation corresponding to the number \( N\text{perm}_i \); \( \pi' \) can be considered as \( \pi'(1), \pi'(2), \ldots, \pi'(p) \);
   d. Organize the \( p \) images of block \( N_i \) relatively to the permutation \( \pi' \);
   e. Compute the number \( r_i \) such that \( r_i = \text{val}_i \mod (2^\alpha), 0 \leq r_i < 2^\alpha \);
   f. Compute \( \beta_i \) by \( \beta_i = (r_i)_2 \), the binary representation of \( r_i \) on \( \alpha \) bits;
   g. Divide \( \beta_i \) into \( p \) blocks \( (\beta_i)_1, (\beta_i)_2, \ldots, (\beta_i)_p \) of \( Q(m, n) \) bits each;
   h. For each block \( (\beta_i)_k, 1 \leq k \leq p \), use the technique \( T \) with the key \( K \) to embed \( (\beta_i)_k \) in the binary image \( I_{\pi'(k)} \);
   i. Send the images in this order: \( I_{\pi'(1)}, I_{\pi'(2)}, \ldots, I_{\pi'(p)} \);

**end:**

The embedding process begins by dividing the secret message \( M \) into \( l \) blocks of \( b \) bits, such that \( M = M_1 || M_2 || \cdots || M_l \), with \( b = p \times Q(m,n) + p \times \log_2(p!) \).

### 4.1 Embedding algorithm

We have a secret message \( M \) to embed in \( N \) binary images, using a steganographic scheme that can hide \( Q(m,n) \) bits in a binary image of size \( m \times n \). The scheme \( T \) uses a key \( K \).

Algorithm 5 presents the process to hide the secret message \( M \) into the \( N \) binary images, using the technique \( T \).
4.2 Retrieval algorithm

Algorithm 6 presents how to extract the secret message embedded in a series of $N$ binary images, using a steganographic scheme $T$ for which $T$ can hide $Q(m, n)$ in a binary image of size $m \times n$.

**Algorithm 6**: Retrieval algorithm

**Algorithm**: Retrieval

**Input**:
- $N$: the set of binary images;
- $p$: the size of images block;
- $s$: the number of images block;
- $M$: the secret binary message to retrieve;
- $\pi$: the initial permutation;
- $Q(m, n)$: the number of hidden bits in a binary image of size $m \times n$ bits;
- $K$: the key related to $T$;
- $\emptyset$: an empty string;

**Output**:
- $M$: the secret binary message embedded;

begin:

1. for each images block $N_i$, $1 \leq i \leq s$,
   a. Use the technique $T$ with its related key $K$ to retrieve in the image $i_{\pi'}(k)$ ($1 \leq k \leq p$), the $Q(m, n)$ embedded bits noted $\text{vect}$ and compute $\beta \leftarrow \beta + \text{vect}$;
   b. Compute $r$, the decimal representation of $\beta$;
   c. Build the number $N_{\text{perm}i}$ by $N_{\text{perm}i} = \text{rank}(p, \pi', \pi'^{-1})$;
   d. Compute $\text{val}_i$ by $\text{val}_i = N_{\text{perm}i} \cdot 2^r + r$;
   e. Compute $M_i$, the binary representation of $\text{val}_i$;
   f. Compute $M \leftarrow M + M_i$, where $+$ denote the concatenation;

end;

5 Experimental results, comparisons, and discussions

In this section, we present experimental results and discussions of our scheme. In the experimental results subsection, we first give an example that describes step by step of the proposed embedding algorithm. We end the subsection by proposing a comparison in terms of embedding capacity for $p$ images between our proposition and other techniques. The section ends with a discussion subsection.

5.1 Experimental results and comparisons

Consider the following five binary images, in which we want to embed the secret message $M = 10010011111111111111111001111010010$ (Figs. 4, 5, 6, 7, 8).
5.1.1 Embedding process

In this example we have:

- \( p = 5 \), the number of images per block;
- \( s = 1 \), the number of images block;
- \( M \), the secret message;
- \( Q = 6 \), the number of bits to hide per image;
- \( b = \lceil \log_2(5!) \rceil + 5 \times 6 = 36 \).
- \( l = 1 \), the number of blocks of the secret message

1. Compute the number \( \text{val} = (M)_10 = 39\,728\,444\,370 \);
2. Compute the number \( N_{\text{perm}} = \lceil 39\,728\,444\,370/2^{5\times6} \rceil = 36 \);
3. Compute \( \pi' = \text{unrank}(5, 36, \pi) = 3 \, 1 \, 5 \, 4 \, 2 \), the permutation corresponding to the number 36;
4. the block \( N \) is reorganized as follows: \( i_3, i_1, i_5, i_4, i_2 \);
5. Compute the number \( r = 39\,728\,444\,370 \mod(2^{30}) = 1\,073\,738\,706 \);
6. Compute \( \beta = (1073738706)_2 = 111111\,111111\,111111\,001111\,010010 \);
7. Divide \( \beta \) into 5 blocks of 6 bits each: \( (\beta_1) = 111111 \), \( (\beta_2) = 111111 \), \( (\beta_3) = 111111 \), \( (\beta_4) = 001111 \), \( (\beta_5) = 010010 \);
8. use the technique \( T \) with the key \( K \) to embed \( (\beta_1) = 111111 \) in \( i_3 \), \( (\beta_2) = 111111 \) in \( i_1 \), \( (\beta_3) = 111111 \) in \( i_5 \), \( (\beta_4) = 001111 \) in \( i_4 \) and \( (\beta_5) = 010010 \) in \( i_2 \);
9. Send the binary images in the order \( i_3, i_1, i_5, i_4, i_2 \) as presented below (Figs. 9, 10, 11, 12, 13).

At the arrival, the receiver will use our retrieval algorithm to obtain the secret data.

Remark:

- The Pan et al. [12] algorithm applied to these data allows 6 bits \( \times 5 = 30 \) bits to be hidden.
- For the same data, our scheme allows 36 bits to be hidden.

Table 2 presents a comparison in terms of embedding capacity (EC) for \( p \) images between our proposition and other techniques. The size of each binary image is \( 512 \times 512 \), as used in Table 3 of [4]. From this table, we can observe that for \( p \) images, regardless of the technique \( T \) used, our scheme improves the embedding capacity by \( p \times \lceil \log_2(p) \rceil \) bits.
Table 2 Embedding capacity of our scheme for \( p \) binary images in comparison with other techniques

| Techniques | Embedding capacity (EC) | EC for \( p \) images |
|------------|-------------------------|------------------------|
| Scheme of Pan et al. [12] | 18 | \( 18 \times p \) |
| Scheme of Ding and Wang [4] | 14.555 | \( 14.555 \times p \) |
| Scheme of Wu and Hwang [11] | 10.314 | \( 10.314 \times p \) |
| Ours with \( T = \) scheme of Pan et al. [12] | 18 | \( p \times \lceil \log_2(p) \rceil + 18p \) |
| Ours with \( T = \) scheme of Ding and Wang [4] | 14.555 | \( p \times \lceil \log_2(p) \rceil + 14.555 \times p \) |
| Ours with \( T = \) scheme of and Hwang [11] | 10.314 | \( p \times \lceil \log_2(p) \rceil + 10.314 \times p \) |

Fig. 3 The graphical representation of the extracting process

Fig. 4 Image \( i_1 \)

Fig. 5 Image \( i_2 \)

Fig. 6 Image \( i_3 \)

Generally, Table 3 presents the embedding capacity of our scheme for \( p \) binary images in comparison with any other embedding scheme \( T \).

Note that, since our method can allow for \( p \) images, using the technique proposed in [12], to embed approximately \( p \times \lceil (\log_2(p) - 1, 44) \rceil + p \times \lceil \log_2(mn + 1) \rceil \) bits, if \( p = 3 \times n \times m \), it can hide \( 3.n.m \times \lceil \log_2((3.n.m) - 1, 44) \rceil + 3.n.m \times \lceil \log_2(mn + 1) \rceil \) bits, which is approximately equal
Our scheme inherits the strengths and weaknesses of the technique \( T \) used. Suppose that \( T \) is the scheme of Pan et al. [12]. In this case, when the opponent captures a copy of the images to retrieve the secret message, he needs the key. Without the key, the opponent has to proceed with a brute-force attack. The brute-force attack on our scheme requires a brute-force attack on Pan et al.’s scheme [12].

The brute-force attack on that scheme requires \( 2^{mn} \) and \( C_{2^{r}-1}^{mn} \ast (2^{r} - 1)! \ast (2^{r} - 1)^{mn - (2^{r} - 1)} \) combinations for the key \( K \) and the weight matrix \( W \) (Sect. 2.2.3).

Furthermore, since the spy does not have the initial permutation, he has to use the brute-force attack. The brute-force attack for \( p \) images requires \( p! \) combinations. According to Stirling’s formula, this requires \( (\frac{p}{e})^p \sqrt{2\pi p} \) combinations. Finally, the spy has to break the scheme proposed by Pan et al. [12] \( p \times s \) times, which requires by brute-force attack an exponential number of combination.

6.2 Complexity of our scheme

The time complexity of our scheme depends on the technique \( T \) used. We can then distinguish two cases.

6.2.1 The technique \( T \) is the scheme of Pan et al. [12]

In this case, the complexity of our scheme is the maximum of the complexity of the scheme of Pan et al. [12] and the complexities of the other elementary steps of our scheme. The complexity of the scheme of Pan et al. [12] is \( O(m \times n) \) for a binary image block of size \( m \times n \). The other elementary steps of our scheme have \( l \times O(p) \) complexity. We can obviously concludes that in this case, the time complexity is \( O(m \times n) \).

6.2.2 The technique \( T \) is any other scheme

In this case, the time complexity is \( \max(l \times O(p), C) \), where \( C \) represents the complexity of the technique \( T \) used.

7 Conclusion

In this paper, we have proposed a steganographic scheme based on permutations that improves the capacity of embedding information in a series of \( p \) host binary images. To illustrate its performance, we used the method proposed by Pan et al., and the results obtained, showed the feasibility of the proposed scheme and comparatively to the related studies, showed that it improves the embedding capacity.

Acknowledgements This work was supported by UMMISCO, LIRIMA, and the University of Yaounde I. The authors are grateful for this support.

Author Contributions RN conceived and directed this research. JKS investigated, implemented and wrote the paper. SGRE investigated this research. All authors reviewed and approved the final manuscript.

Funding Not applicable.

Availability of data and materials The authors confirm that the data supporting the findings of this study are available within the article.
Declarations

Conflict of interest Not applicable.

Ethical approval Not applicable.

References

1. Kim, C., Yang, C.-N., Baek, J., Leng, L.: Survey on data hiding based on block truncation coding. Appl. Sci. 11, 9209 (2021). https://doi.org/10.3390/app11199209
2. Por, L.Y., Delina, B.: Information hiding: a new approach of text steganography. In: Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science (ACACOS ’08), Hangzhou, China, pp. 689–695. ISBN: 978-960-6766-49-7 (2008)
3. Sun, S.: A novel edge based image steganography with $2^k$ correction and Huffman encoding. Inf. Process. Lett. 116(2), 93–99 (2016)
4. Ding, W., Wang, Y.: Data hiding in binary image with high payload. Arab. J. Sci. Eng. (2018). https://doi.org/10.1007/s13369-018-3130-5
5. Feng, B., Lu, W., Sun, W.: Secure binary image steganography based on minimizing the distortion on the texture. IEEE Trans. Inf. Forensics Secur. 10(2), 243–255 (2015)
6. Lehmer, D.H.: Teaching combinatorial tricks to a computer. In: Proceedings of Symposium in Applied Mathematics, Combinatorial Analysis, vol. 10, pp. 179–193. American Mathematical Society, Providence (1960)
7. Myrvold, W., Ruskey, F.: Ranking and unranking permutations in linear time. Inf. Process. Lett. 79(6), 281–284 (2001)
8. Jung, K., Yoo, K.: Data hiding method in binary images based on block masking for key authentication. Inf. Sci. 277, 188–196 (2014). https://doi.org/10.1016/j.ins.2014.02.016
9. Priyanka Sambhaji, D., Gyankamal, J.: Minimum distortion secured binary image steganography. Int. J. Sci. Dev. Res. 4(5), 228–231 (2019)
10. Jung, K.: Data hiding of digital images based on bit position and parity bit. Imaging Sci. J. 60(6), 329–337 (2012). https://doi.org/10.1179/174313112X13522089179550
11. Wu, N., Hwang, M.: Development of a data hiding scheme based on combination theory for lowering the visual noise in binary images. Disp. J. 49, 116–123 (2017). https://doi.org/10.1016/j.displa.2017.07.009
12. Pan, H.-K., Chen, Y.-Y., Tseng, Y.-C.: A secure data hiding scheme for two-color image. In: IEEE Transactions on Communications, vol. 50, no. 8, pp. 1227–1231 (2002)
13. Leversha, G., Lovacz, L., Pelikan, J., Vesztergombi, K.: Discrete mathematics, elementary and beyond. Math. Gaz. 88(512), 378–379, 290 (2004)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.