Binary Image Steganography based on Permutation

Juvet Karnel Sadié
Université de Yaounde I
Stéphane Gael Raymond Ekodeck
Universite de Yaounde 1 Faculte des Sciences
Rene Ndoundam (ndoundam@yahoo.com)
Universite de Yaounde 1 Faculte des Sciences https://orcid.org/0000-0003-1105-762X

Research Article

Keywords: Steganography , cover medium, binary image, stego-object, permutation, embedding capacity, brute-force attack

Posted Date: May 27th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-427914/v1

Binary Image Steganography based on Permutation

Juvet Karnel Sadié[1,2,3], Stéphane Gael R. Ekodeck[1,2,3], René Ndoundam[1,2,3]*
[1] *Team GRIMCAPE*
[2] *Sorbonne University, IRD, UMMISCO, F-93143, Bondy, France*
[3] *Department of Computer Science, University of Yaoundé I, P.o. Box 812 Yaounde, Cameroon*
**corresponding author: ndoundam@yahoo.com*

ABSTRACT
We propose a steganographic scheme based on permutations, which improves the capacity of embedding information in a series of p host binary images. Given a host binary image block of size m x n bits and any embedding technique T, where T can hide Q(m, n) bits of data in the image; given p images, T can hide p x Q(m, n) bits of data in these images. Our scheme improves the capacity of embedding information in p images such that, instead of p x Q(m, n) bits, it can hide p x $\log_2(p)$ + p x Q(m, n)bits. The results which have been obtained by experiments, show that our model performs a better hiding process in terms of hiding capacity.

KEYWORD

Steganography , cover medium, binary image, stego-object, permutation, embedding capacity, brute-force attack.

I. INTRODUCTION

The word steganography is of Greek origin and means *covered writing*. It is the hiding of a message within another (cover medium) such as web pages, images or text, so that the presence of the hidden message is indiscernible. Especially, images are the most popular cover media [1]. When a message is hidden in the 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 with bare eyes.

From the definition, Steganography is used to ensure data confidentiality, like encryption. However, the main difference between them is that with encryption, anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can detect the presence of the message. When combined, steganography and encryption can provide more security.

Steganography dates back to ancient Greece, where common practices consisted of etching messages in wooden tablets and covering them with wax. With the modernization of cover media, many other techniques have since been proposed, like Least Significant Bit Replacement (LSB) [2], F5 [3] and many others [4,5], for instance.

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.

More precisely, given a host binary image block of size m x n bits and an embedding technique T, T can allow to hide Q(m, n) bits of data in the image. Given p images, T can hide p x Q(m, n) bits of data in these images. Our scheme improves this capacity. This means that, instead of px Q(m, n) bits, it can hide $p \times \log_2(p) + p \times Q(m, n)$bits. The sender and the receiver share a permutation protocol. To illustrate this, we use the scheme proposed by H.-K. Pan et al. [6] in which given a binary image block of m x n size bits, $E(\log_2(mn+1))$ bits can be hidden in that block, where E(x) denotes the whole lower part of x.

The rest of the paper is organized as follows: in section 2, we present some preliminaries that lead us to the design of our scheme. Then, we present the basic idea in section 3. Section 4 concerns the presentation of our scheme. Experimental results and discussion are stated in section 5. The security of the scheme is shown in section 6 and finally section 7 is devoted to the conclusion.

II. PRELIMINARIES

Issues that most of the existing hiding techniques are trying to solve in steganography, no matter what kind of cover medium is used, are related to the embedding capacity and the security. Thus, in order to design a scheme that takes into consideration these two parameters, we have selected a particular medium to hide our secret data: binary images; and focused our work on how to take advantage of permutations to generate innocent-looking images and by the same way increase their embedding capacity without neglecting their security.

In this light, we found much interest in the work of H.-K. Pan et al [6] who defined a secure data hiding scheme for two-color images. We equally probed into permutation algorithms in order to employ the power of permutations. In our literature therefore, our interest fell on the unranking and ranking permutation algorithms of W. Myrvold and F. Ruskey [7], detailed in the section below.

1. Permutation Generation Methods

Permutation is one of the most important combinatorial object in computing, and can be applied in various applications, for example, the 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 D.H. Lehmer [8]. Several authors [7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] have developed many methods to generate all the possible permutations of n elements.

Also, several works [23, 24, 25, 26, 27, 28] in steganography taking advantage of permutations have been done. H. Hioki [25] in 2013, proposed a permutation steganography, which is an effective method for hiding messages provided where the contents of cover objects are not affected by the rearrangement of their elements. Liu and Lee [28] proposed a

reversible image steganography method based on pixel value ordering (PVO) to improve the steganographic capacity.

In their paper, W. Myrvold and F. Ruskey [7] proposed a ranking function 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 to which, given an integer r between 0 and n! - 1, the value of the function is the permutation of rank r. Their algorithms is presented below [7].

a. 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, … n-1.

Procedure unrank (n, r, π)[7]

begin

if n > 0 then

swap(π[n-1], π[r mod n]);

unrank(n-1, E(r /n) , π);

end;

end;

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

b. Ranking function

To rank, first compute π^{-1}. This can be done by iterating

$\pi^{-1}[\pi[i]] = i$, for i=0, 1,…, n-1.

In the algorithm below, both π and π^{-1} are modified.

function rank (n, π, π^{-1}):integer [7]

begin

if n = 1 then return(0) end;

s:=π [n-1] ;

swap(π[n-1], π[π^{-1} [n-1]]) ;

swap(π^{-1} [s], π^{-1} [n-1]) ;

return (s+n x rank(n-1, π, π^{-1})) ;

end;

2. Two-Color Image Data Hiding

As previously mentioned, H.-K. Pan et al. in [6], presented a secure data hiding scheme for two-color images. In their method, given a cover binary image F, they have partitioned it into blocks of fixed size m x n (assuming for simplicity that F's size is a multiple of m x n). Their method is able to 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 $r \leq E(\log_2(mn+1))$. To secure the information, they used a secret key with two components:

- K: a randomly selected binary matrix of size m x n.
- W: a weight matrix which is an integer matrix of size m x n. W satisfies the condition that $[W]_{\{i,j\}}$:i = 1 … m, j=1 … n = {1, 2, …, 2^{r-1} }.

Data hiding is achieved by modifying some bits of F. Below, we have exhibited how they have done it, by presenting their embedding and extraction algorithms.

a) Embedding Algorithm [6]

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

Output: the stego binary image F'_i

- Compute $F_i \oplus K$, where $\oplus$ is the bitwise exclusive-OR of two equal-size binary matrices.
- 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.
- From the matrix $F_i \oplus K$, compute for each w = 1 … 2^{r-1} the following set:

$S_w = \{(j, k) | (([W]_{\{j,k\}} = w) \wedge ([F_i \oplus K]_{\{j,k\}} = 0)) \vee$

$(j, k) | (([W]_{\{j,k\}} = 2\text{^}r\text{-}w) \wedge ([F_i \oplus K]_{\{j,k\}} = 1))\}$

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 $[W]_{\{j, k\}} = 2^r\text{-}w$, and $[F \oplus K]_{\{j, k\}} = 1$, 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 $w \equiv w' \pmod{2^r}$.

- Define a weight difference

  $d \equiv (b_1 b_2 \ldots b_r) - SUM((F_i \oplus K) \otimes W) \pmod{2^r}$.

  If d = 0, there is no need to change F_i. Otherwise, we run

the following steps to transform F^i to F'^i :

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

To summarize, the above steps ensure the following invariant: $SUM((F_i \oplus K) \otimes W) \equiv (b_1 b_2 \ldots b_r) \pmod{2^r}$

b) Extraction Algorithm [6]

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

Output: the secret message $(b_1 b_2 \ldots b_r$

- compute $SUM((F'_i \oplus K) \otimes W) \mod 2^r$ to find the hidden bit stream $b_1 b_2 \ldots b_r$

c) Security

In [6], H.-K. Pan et al. identified some possible attacks on their scheme and their costs. From their discussion, a brute-force attack is quite impossible, since there are $2^{\{mn\}}$ and $C_{\{2\text{^}r\text{-}1\}}\text{^}\{mn\}*(2^r\text{-}1)!*(2^r\text{-}1)^{mn\text{-}(2\text{^}r\text{-}1)\}}$ 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 x n) is reasonably large and the secrecy of K and W is maintained.

III. 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 x n each, denoted by im_1, im_2 and im_3 respectively. Also, we denote by T the scheme proposed by H.-K. Pan [6], where given a binary image of size m x n, T can hide up to $E(\log_2(mn+1))$ bits in that image. For the three images T would hide up to 3 x $E(\log_2(mn+1))$. We propose a scheme which improves that embedding capacity. The scheme is based on the sending order of the images between the sender (A) and the receiver (B).

Without lost of generality, let us consider the following functions f and its inverse g.

$f(3,1,2) = 0$

$f(2,3,1) = 1$

$f(2,1,3) = 2$

$f(3,2,1) = 3$

$f(1,3,2) = 4$

$f(1,2,3) = 5$

and

$g(0) = (3,1,2)$

$g(1) = (2,3,1)$

$g(2) = (2,1,3)$

$g(3) = (3,2,1)$

$g(4) = (1,3,2)$

$g(5) = (1,2,3)$

Let's denote by x the decimal value of bits to be hidden.

1) **Embedding process**

- compute the number i by $i = x / 2^{\{3x\ E(\log2(mn+1))\}}$;
- compute the number z such that $z = x \mod (2^{\{3xE\ (\log2(mn+1))}), 0 \leq z < 2^{\{3x\ E(\log2(mn+1))\}}$;
- compute $W=(z)_2$, the binary representation of z.
- divide W into three blocks w_1, w_2 and w_3 of $E(\log_2(mn+1))$ bits each ;
- compute S=g(i) ;
- if $S = (\alpha, \beta, \gamma)$, $\alpha \in \{1,2,3\}$, $\beta \in \{1,2,3\}$, $\gamma \in \{1,2,3\}$

- use the scheme of H.-K. Pan [6] to hide w_1 in im_α, w_2 in im_β and w_3 in im_γ
- send the images in the order im_α, im_β and im_γ

2) Retrieval process

- If the arrival order of the images is im_α, im_β and im_γ, $\alpha \in \{1,2,3\}$, $\beta \in \{1,2,3\}$, $\gamma \in \{1,2,3\}$
- compute the number i such that $i=f(\alpha,\beta,\gamma)$
- use the scheme of H.-K. Pan et al. [6] to extract w_1 in im_α, w_2 in im_β and w_3 in im_γ
- compute $W=w_1||w_2||w_3$, where || denotes the concatenation operation.
- compute $z=(W)_{10}$, the decimal representation of W.
- compute $x = z + i \times 2^{\{3xE(\log2(mn+1))\}}$

Taking into account the sending order of the images, we can hide the secret with decimal value between 0 and $6 \times 2^{\{3x\ E(\log2(mn+1))\}}-1$. This means that, the maximum decimal value that can be hidden for three images is $6 \times 2^{\{3x\ E(\log2(mn+1))\}}$, which in binary is represented on $E(\log_2(6 \times 2^{\{3x\ E(\log2(mn+1))\}}) = E(\log_2(6)) + E(\log_2(2^{\{3\ x\ E(\log2(mn+1))\}}) = 2 + 3 \times E(\log_2(mn+1))$ bits.

For p images we can hide up to $\log_2(p!) + p \times E(\log_2(mn+1))$ bits.

Given p binary images, table 1 presents a comparison in terms of embedding capacity between the scheme of H.-K. Pan [6] 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 is denoted by T.

IV. 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 of size m x n, T can hide Q(m, n) data bits. Initially, the two communicating parties must share a set of N binary images of size m x n each, denoted $i_1, i_2, \ldots, i_N$.

Secondly, the set is divided into s blocks of p binary images. In other terms, the N images are divided as :

$(i_1, i_2, \ldots, i_p), (i_{\{p+1\}}, i_{\{p+2\}}, \ldots, i_{\{2p\}}), , (i_{\{(s-1)p+1\}}, i_{\{(s-1)p+2\}}, \ldots, i_{\{sp\}})$, 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.

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

1- Embedding Algorithm

Here, we present our embedding scheme.

Algorithm: embedding

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 embed;

l: the number of blocks of the secret message;

π : the initial permutation;

Q(m,n): the number of bits to hide in a binary image of size m x n bits;

K: the key related to T;

Output:

stego binary images of m x n bits each;

begin

1. Divide M into l blocks of b bits; $b = p \times Q(m,n) + E(\log_2(p!))$;

2. for each block M_i, $1 \leq i \leq l$;

a. Compute $val_i = (M_i)_{\{10\}}$, the decimal representation of M_i;

b. Compute the number $Nperm_i$ by $Nperm_i = E(val_i / 2^{\alpha})$, where $\alpha = p \times Q(m, n)$;

c. Compute $\pi' = unrank(p, Nperm_i, \pi)$, the permutation corresponding to the number $Nperm_i$; π' can be considered as $\pi'(1), \pi'(2), \ldots, \pi'(p)$;

d. Organize the p images of block N_i relatively to the permutation π';

e. Compute the number r_i such that $r_i = val_i \mod (2^{\alpha})$, $0 \leq r_i < 2^{\alpha}$;

f. Compute β_i by $\beta_i = (r_i)_2$, the binary representation of r_i on α bits;

g. Divide β_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 the order $i_{\{\pi'(1)\}}, i_{\{\pi'(2)\}}, \ldots, i_{\{\pi'(p)\}}$.

end;

2- Retrieval Algorithm

Here, we present the process of extracting the secret embbeded in the binary images.

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;

l: the number of blocks of the secret message;

π : the initial permutation;

Q(m,n): the number of hidden bits in a binary image of size m x n bits;

K: the key related to T;

β: 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 vect and compute $\beta \Leftarrow \beta \| vect$;

b. Compute r, the decimal representation of β;

c. Build the number $Nperm_i$ by $Nperm_i = rank(p, \pi', \pi'^{\{-1\}})$;

d. Compute val_i by $val_i = Nperm_i \times 2^{\alpha} + r$;

e. Compute M_i, the binary representation of val_i;

f. Compute $M \Leftarrow M \| M_i$, where $\|$ denote the concatenation;

end ;

3- Remark

For a block of p images, we have p! possible permutations, and by this fact the maximum value of val_i is $2^{\alpha \times (p!).}$ This means that, we can hide up to $\log_2(val_i)$ bits per images blocks, which is $E(\log_2(p!))+ p \times Q(m, n)$.

According to Stirling's formula, $\log_2(p!) \approx p(\log_2(p)-1,44)$.

For instance, in the case where Q(m, n) is equal to $E(\log_2(mn+1))$ as proposed in [6], the maximum capacity of hiding of p images is about $p \times E((\log_2(p)-1, 44)) + p \times E(\log_2(mn+1))$.

V. Experimental results and discussion

In this section, we present experimental results and discussion about our scheme. In the Experimental results subsection, we first present an example which describes step by step our embedding algorithm. We end the subsection by proposing tables and figure which generalize our results.

1- Experimental results

Consider the following twenty binary images, figure i_1, figure i_2, figure i_3, figure i_4, … figure i_{19} and figure i_{20} in which we want to embed the secret message M = 1001001111111111111111111001111010010110100111110111111110111101111010010111 1101101101111111101110011111010010110110110111111101110111001111010111;

a. Embedding process

In this example we have:

- N=20, the number of binary images;
- p=5, the number of images per block;
- s=4, the number of images block;
- M, the secret message;
- $b= E(\log_2(5!)) + 5 \times 6 = 36$.
- l=4, the number of blocks of the secret message

- Divide M into 4 blocks of 36 bits;

M_1 = 100100111111111111111111001111010010;

M_2 = 110100111110111111110111101111010010;

M_3 = 111110110110111111110111001111010010;

M_4 = 110110110111111101110111001111010111;

- for the block M_1 = 100100111111111111111111001111010010;

- $val_1 = (M_1)_{10}$ = 39 728 444 370;
- Compute the number $Nperm_1 = E(39\ 728\ 444\ 370 / 2^{\{5x6\}}) = 36$;
- Compute π' = unrank(5, 36, π) = 3 1 5 4 2, the permutation corresponding to the number 36;
- the block N_1 is reorganized as follows: i_3, i_1, i_5, i_4, i_2;
- Compute the number $r_1 = E(39\ 728\ 444\ 370 \mod 2^{\{30\}}) = 1\ 073\ 738\ 706$;
- Compute $\beta_1 = (1\ 073\ 738\ 706)_2$ = 111111 111111 111111 001111 010010;
- Divide β_1 into 5 blocks of 6 bits each: $(\beta_1)_1$ = 111111, $(\beta_1)_2$ = 111111, $(\beta_1)_3$ = 111111 , $(\beta_1)_4$ =001111, $(\beta_1)_5$ =010010;
- use the technique T with the key K to embed $(\beta_1)_1$ = 111111 in i_3, $(\beta_1)_2$ = 111111 in i_1 , $(\beta_1)_3$ = 111111 in i_5 , $(\beta_1)_4$ = 001111 in i_4 and $(\beta_1)_5$ =010010 in i_2;
- Send the binary images in the order i_3, i_1, i_5, i_4, i_2.

- for the block M_2 = 110100111110111111110111101111010010;
- $val_2 = (M_2)_{10}$ = 56 891 505 618;
- Compute the number $Nperm_2 = E(56\ 891\ 505\ 618 / 2^{\{5*6\}}) = 52$;
- Compute π' = unrank(5, 52, π) = 7 6 9 10 8, the permutation corresponding to the number 52;
- The block N_2 is reorganized as follows: $i_7, i_6, i_9, i_{\{10\}}, i_8$;
- Compute the number $r_2 = E(56\ 891\ 505\ 618 \mod 2^{\{30\}}) = 1\ 056\ 930\ 770$;
- Compute $\beta_2 = (1\ 056\ 930\ 770)_2$ = 111110 111111 110111 101111 010010;
- Divide β_2 into 5 blocks of 6 bits each: $(\beta_2)_1$ = 111110, $(\beta_2)_2$ = 111111, $(\beta_2)_3$ = 110111 ,$(\beta_2)_4$ =101111, $(\beta_2)_5$ =010010;
- use the technique T with the key K to embed $(\beta_2)_1$ = 111110 in i_7, $(\beta_2)_2$ = 111111 in i_6, $(\beta_2)_3$ = 110111 in i_9, $(\beta_2)_4$ = 101111 in $i_{\{10\}}$, $(\beta_2)_5$ =010010 in i_8;
- Send the binary images in the order $i_7, i_6, i_9, i_{\{10\}}, i_8$

- for the block M_3 = 111110110110111111110111001111010010;
- $val_3 = (M_3)_{10}$ = 67 494 704 082;
- Compute the number $Nperm_3 = E(67\ 494\ 704\ 082 / 2^{\{30\}}) = 62$;
- Compute π' = unrank(5, 62, π) = 15 12 14 11 13, the permutation corresponding to the number 62;
- The block N_3 is reorganized as follows: $i_{15}, i_{12}, i_{14}, i_{11}, i_{13}$;
- Compute the number $r_3 = E(67\ 494\ 704\ 082 \mod 2^{30}) = 922\ 710\ 994$;
- Compute $\beta_3 = (922\ 710\ 994)_2$ = 110110 111111 110111 001111 010010;
- Divide β_3 into 5 blocks of 6 bits each: $(\beta_3)_1$ = 110110, $(\beta_3)_2$ = 111111, $(\beta_3)_3$ = 110111 ,$(\beta_3)_4$ = 001111, $(\beta_3)_5$ =010010;

- use the technique T with the key K to embed $(\beta_3)_1 = 110110$ in i_{15}, $(\beta_3)_2 = 111111$ in i_{12}, $(\beta_3)_3 = 110111$ in i_{14}, $(\beta_3)_4 = 001111$ in i_{11}, $(\beta_3)_5 = 010010$ in i_{13};
- Send the binary images in the order i_{15}, i_{12}, i_{14}, i_{11}, i_{13}.

  - for the block M_4 = 110110110111111101110110011111010111;
- $val_4 = (M_4)_{10} = 58\ 921\ 022\ 423$;
- Compute the number $Nperm_4 = E(58\ 921\ 022\ 423 / 2^{30}) = 54$;
- Compute π' = unrank(5, 54, π) = 17 16 19 18 20, the permutation corresponding to the number 54;
- the block N_4 is reorganized as follows: i_{17}, i_{16}, i_{19}, i_{18}, i_{20};
- Compute the number $r_4 = E(58\ 921\ 022\ 423 \mod (2^{30})) = 938\ 963\ 927$;
- Compute $\beta_4 = (938\ 963\ 927)_2 = 110111\ 111101\ 110111\ 001111\ 010111$;
- Divide β_4 into 5 blocks of 6 bits each: $(\beta_4)_1 = 110111$, $(\beta_4)_2 = 111101$, $(\beta_4)_3 = 110111$, $(\beta_4)_4 = 001111$, $(\beta_4)_5 = 010111$;
- use the technique T with the key K to embed $(\beta_4)_1 = 110111$ in i_{17}, $(\beta_4)_2 = 111101$ in i_{16}, $(\beta_4)_3 = 110111$ in i_{19}, $(\beta_4)_4 = 001111$ in i_{18}, $(\beta_4)_5 = 010111$ in i_{20};
- Send the binary images in the order i_{17}, i_{16}, i_{19}, i_{18}, i_{20}.

At the arrival, the receiver will use our retrieval algorithm to obtain the secret data. We see with this simple example that we have hidden 144 bits instead of only 120 bits.

The table 2 presents the embedding capacity of our scheme for p binary images in comparison with the scheme of H.-K. Pan et al. [6] and the table 3 presents the embedding capacity of our scheme for p binary images in comparison with any other embedding scheme T.

More generally, the figure 21 presents a comparison in terms of embedding capacity for N images between our proposition and traditional techniques.

Note that, since our method can allow for p images, using the technique proposed in [6], to embed about p x E((log2(p)-1,44)) + p x E(log2(mn+1)) bits, if p = 3 x n x m, it can hide 3.n.m x E(log2((3.n.m)-1,44)) + 3.n.m x E(log2(mn+1)) which is approximatively equal to 3.n.m x E(log2(mn+1)) + 3.n.m x E(log2(mn+1)). We see by this that the hiding capacity has doubled.

2- Discussion

In order to extract the secret message, the images must arrive in the order they were sent. This requires a fairly secure transmission channel and a reliable transfer protocol.

Initially, the two communicating parties must share a large set of N binary images of size m x n each, denoted $i_1, i_2, \ldots, i_N$, but the number of images effectively used depend on the size of the secret message.

In order to increase the security of our scheme, the value of p must be choosen so that the scheme will be resistant to brute-force attack. Therefore, its value must be greater than 100, (p>100).

VI. Security and complexity of our scheme

1- Security of our scheme

Our scheme inherits the strengths and the weaknesses of the technique T used. In particular, if the technique T used is the scheme of H.-K. Pan [6], when the opponent captures a copy of the images in our scheme, a brute-force attack of our scheme requires a brute-force attack of H.-K. Pan et al.'s scheme [6]. The brute-force attack of that scheme requires $2^{\{mn\}}$ and $C_{\{2\^r-1\}}\^\{mn\}*(2^r-1)!*(2^r-1)^{mn-(2\^r-1)\}}$ combinations for the key K and the weight matrix W, as mentioned in section II.2.3 relative to the security of the scheme proposed by H.-K. Pan et al. [6].

Furthermore, as far as the spy does not have the initial permutation, he would have to use the brute-force attack. The brute-force attack requires for p images p! combinations. According to Stirling's formula, this means it requires $(p/e)^p \times \sqrt{2\pi p}$ combinations. Finally, he would have to break p x s times, the scheme proposed by H.-K. Pan et al. [6] which requires by brute-force attack an exponential combination.

2- Complexity of our scheme

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

a) The technique T is the scheme of H.-K. Pan et al. [6]

In this case, the complexity of our scheme is the maximum of the complexity of the scheme of H.-K. Pan et al. [6] and the complexity of the others elementary instructions of our scheme. The complexity of the scheme of H.-K. Pan et al. [6] is O(m x n) for a binary image block of size m x n. The others elementary instructions of our scheme are in l x O(p). We can obviously conclude that in this case, the time complexity is O(m x n)

b) The technique T is any other scheme

Here, the time complexity is given by max(l x O(p), C), where C represents the complexity of the technique T used.

VII. Conclusion

In this paper, we have proposed a steganographic scheme based on permutations which improves the capacity of embedding information in a series of p host binary images. To illustrate its performance, we have used the method proposed by H.-K. 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.

VIII. Abbreviations

E(x) denotes the whole lower part of x

IX. Declarations

1- Availability of data and materials

Not applicable

2- Competing interests

The authors declare that they have no competing interests.

3- Funding

Not applicable

4- Authors' contributions

The idea behind this paper was proposed by René Ndoundam. He supervised the work from the beginning until the end. Juvet Karnel Sadié and Stéphane Gael R. Ekodeck Devoloped the idea. Juvet Karnel Sadié is responsible for this complete write-up. He also performed the experimentations. All authors read and approved the final manuscript.

5- Acknowledgements

This work was supported by UMMISCO, LIRIMA and the University of Yaounde 1. The authors are grateful for this support.

6- Authors' information

Affiliation

University of Yaounde I, Faculty of Sciences, Department of Computer Science, P.o.Box 812 Yaounde, Cameroon.

Juvet Karnel Sadié

University of Yaounde I, Faculty of Sciences, Department of Computer Science, P.o.Box 812 Yaounde, Cameroon.

Stéphane Gael R. Ekodeck

University of Yaounde I, Faculty of Sciences, Department of Computer Science, P.o. Box 812 Yaounde, Cameroon.

René Ndoundam

X. Bibliography

1- S. Sun, "A Novel edge based image steganography with 2^k correction and Huffman encoding", *Information Processing Letters*, Vol. 116, Issue 2, pp. 93-99, 2016.

2- C.K. Chan , L.M. Chen, "Hiding Data in Images by Simple LSB Substitution", *Pattern Recognition*, Vol. 37, No. 3, pp. 469-474, 2004.

3- A. Westfeld, "F5 - A Steganographic Algorithm High Capacity Despite Better Steganalysis", *I.S. Moskowitz (Ed.): IH 2001, Lecture Notes in Computer Sciences*, Vol. 2137, pp. 289-302, 2001.

4- Saiful Islam, Mangat R Modi, Phalguni Gupta, "Edge-based image steganography", *EURASIP Journal on Information Security*, 2014:8.

5- Mamta Jain, Saroj Kumar Lenka, "Diagonal queue medical image steganography with Rabincryptosystem", *Brain Informatics, Springer*, (2016) 3:39–51.

6- H.-K. Pan, Y.-Y. Chen and Y.-C. Tseng, "A secure Data Hiding Scheme for Two-Color Image", *Computers and Communications, Proceedings of the 5th IEEE Symposium*}, pp. 750-755, 2000.

7- W. Myrvold , F. Ruskey, "Ranking and Unranking Permutations in Linear Time", *Information Processing Letters*,Vol. 79, Issue 6, pp. 281-284, 2001.

8- D.H. Lehmer ,"Teaching Combinatorial Tricks to a Computer", *Proceedings of Symposium in Applied Mathematics, Combinatorial Analysis, American Mathematical Society*, Vol. 10, pp. 179-193, 1960.

9- A. Nijenhuis, H.S. Wilf, "Combinatorial Algorithms: For Computers and Calculators", 2nd Edition, *Academic Press*, New York, 1978.

10- C.D. Savage, "Generating Permutations with k-differences", *SIAM Journal Discrete Mathematics*, Vol.3, No. 4, pp. 561-573, 1990.

11- C.T. Djamégni and M. Tchuente, "A Cost-Optimal Pipeline Algorithm for Permutation Generation in Lexicographic Order", *Journal of Parallel Distribed Computing*, Vol.44, No. 2, pp. 153-159, 1997.

12- D.E. Knuth, "The Art of Computer Programming, Volume 4, Fascicle 2: Generating all Tuples and Permutations", *Addison-Wesley Professional,* 2005.

13- G.D. Balbine, "Note on Random Permutations, *Mathematics of Computation*, Vol. 21, No. 100, pp. 710-712, 1967.

14- H.F. Trotter, "Perm (Algorithm 115)", *Communications of the ACM*, Vol. 5, No. 8, pp. 434-435, 1962.

15- K.H. Rosen , J.G. Michaels, J.L. Gross, J.W. Grossman, D.R. Shier, "Handbook of Discrete and Combinatorial Mathematics", *CRC Press*, 2000.

16- M.B. Wells, "Generation of Permutations by Transposition", *Mathematics of Computation*, Vol. 15, No. 74, pp. 192-195, 1961.

17- M.K. Shen, "Generation of Permutations in Lexicographical Order", *Communications of the ACM*, Vol. 6, No. 9, pp. 517, 1963.

18- R. Durstenfeld, "Random Permutation", *Communications of the ACM*, Vol. 7, No. 7, pp. 420, 1964.

19- R. Sedgewick, "Permutation Generation Methods", *ACM Computing Surveys*, Vol. 9, No. 2, pp. 137-163, 1977.

20- R.J. Ord-Smith, "Generation of Permutations in Lexicographic Order", *Communications of the ACM*, Vol. 11, No. 2, pp. 117, 1968.

21- S.M. Johnson, "Generation of Permutations by Adjacent Transposition", *Mathematics of Computation*, Vol. 17, No. 83, pp. 282-285, 1963.

22- Md Khairullah, "A novel steganography method using transliteration of Bengali text", *Journal of King Saud University -Computer and Information* Sciences, 2018.

23- A.M. Shihab, R.K. Mohammed, W.M. Abed, "Evaluating the Performance of the Secure Block Permutation Image Steganography Algorithm", *International Journal of Network Security and its Applications*, Vol. 5, No. 5, 2013.

24- H. Al-Bahadili, "A Secure Block Permutation Image Steganography Algorithm", *International Journal on Cryptography and Information Security*, Vol. 3, No. 3, pp. 11-22, 2013.

25- H. Hioki, "Data Embedding Methods Not Based on Content Modification", *Multimedia Information Hiding Technologies and Methodologies for Controlling Data*, Chapter 7, pp. 272-294, 2012.

26- H.-C. Lin and S.-J Lin, "Piecewise Permutation Steganography for 3D Humanoid Mesh Models", 2nd *International Conference on Computer Science and its Applications*, pp. 355-360, 2009.

27- S.G.R. Ekodeck, R. Ndoundam, "PDF steganography based on Chinese Remainder Theorem", *Journal of Information Security and Applications*, Vol. 29, No. 1, pp. 1-15, 2016.

28- Hsing-Han Liu, Chuan-Min Lee, "High-capacity reversible image steganography based on pixel value ordering", *EURASIP Journal on Image and Video Processing*, 2019:54.

XI. Figure

image i_1

image i_2

image i_3

image i_4

image i_5

image i_6

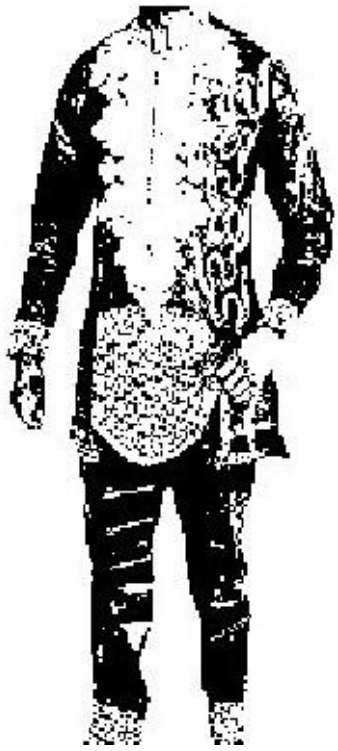


image i_7

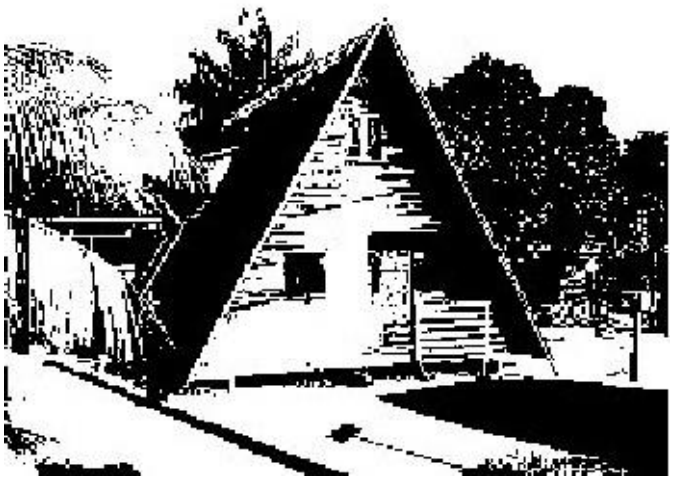


image i_8

image i_9

image i_{10}

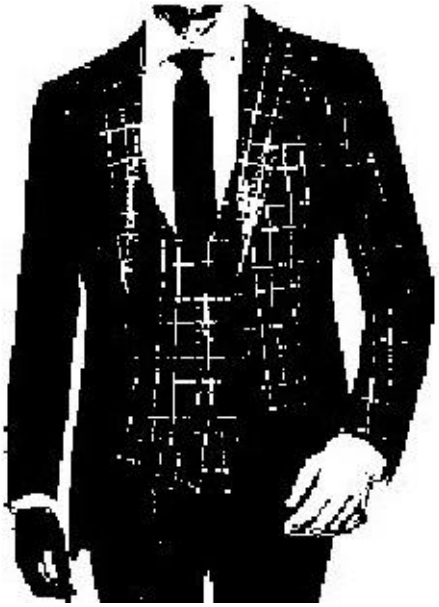


image i_{11}

image i_{12}

image i_{13}

image i_{14}

image i_{15}

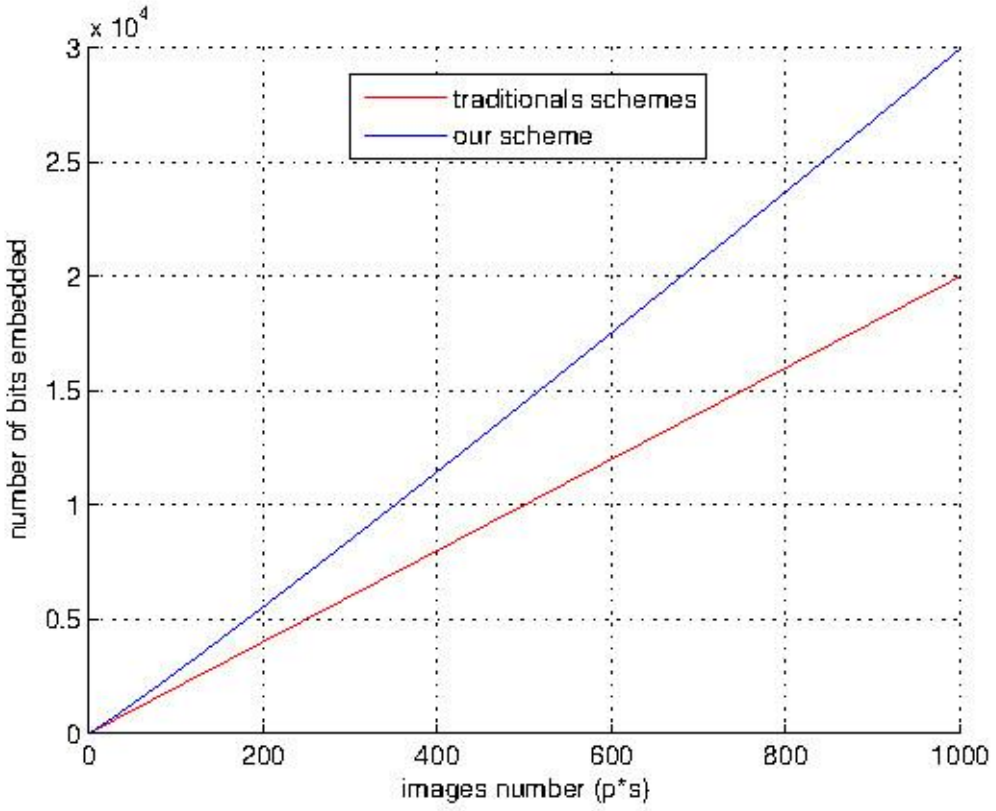
image i_{16}

image i_{17}

image i_{18}

image i_{19}

image i_{20}



A graphical comparison of the embedding capacity of our scheme and the embedding capacity of traditionals schemes.

XII. Tables

Table 1 : Embedding capacity of our scheme for p binary images in comparison with the scheme of H.-K. Pan et al. [6]

	3	P
scheme of H.-K. Pan [6]	3 x E($\log_2$(mn+1))	p xE($\log_2$(mn+1))
our scheme	2 + 3 xE($\log_2$(mn+1))	Log_2(p!)+ p x E($\log_2$(mn+1))

Table 2 : Embedding capacity of our scheme for p binary images in comparison with the scheme of H.-K. Pan et al. [6]

	P
scheme of H.-K. Pan [6]	p x E($\log_2$(mn+1))
our scheme	P xE($\log_2$(p))+ p x E($\log_2$(mn+1))

Table 3 : Embedding capacity of our scheme for p binary images in comparison with any other embedding scheme T

	P
T	p x Q(m,n)
our scheme	P xE($\log_2$(p))+ p x Q(m,n)