Text Hiding In Image Border

Asst. Lect. Razi J. Al-Azawi
Laser and Electronic Optics Engineering Department
University of Technology, Baghdad, Iraq

Abstract

This paper deals with secret-key steganography approach of data hiding, it takes advantage of images with dark background or boundaries.

The main goal is to hide a text in an image without drawing suspicion about the hidden information.

The developed algorithm compares between the ASCII code of the text character and the value of the image pixel, if they are equal, the pixel position (else Character ASCII code) will be stored in the image border after encrypting it.

This developed algorithm has been applied on (gray scale, 256-color, and true color) images, good results were obtained with true color and gray scale images .The 256-color images may draw a suspicion about the hidden information. Binary and multispectral images cannot be used in this algorithm.

الخلاصـــــة

إن الفكرة الأساسية في عملية إخفاء البيانات هي إخفاء وجود البيانات أصلا والمراقب العادي للاتصال لا يرى إلا الغطاء الذي تخفى فيه البيانات ، على انه هنالك علم ***Steganalysis*** *وهو يعني بالبحث عن البيانات المخفاة واستخلاصها أو تدميرها إن تعذر.*

يعنى هذا البحث بفرع إخفاء البيانات ذو المفتاح السري والهدف الرئيسي في هذا البحث هو إخفاء نص مكتوب في صورة بدون إثارة شكوك حول المعلومات المخفاة.

تقارن الخوارزمية المقترحة لهذا البحث بين القيمة الرقمية للحرف من النص وبين قيمة الشدة اللونية لعنصر الصورة وان تساوتا يسجل الموقع (وفي حالة عدم المساواة تسجل قيمة الحرف في جدول ***ASCII****) في إطار الصورة بعد تفجيره بشكل ما.*

طبقت هذه الخوارزمية على عدة أنواع من الصور (التدرج الرمادي ، ٢٥٦ لون ، اللون الحقيقي) وأعطت نتائج جيدة للصور حقيقية الألوان وصور التدرج الرمادي، في حين ان صور ٢٥٦-لون تثير الشك حول البيانات المخفاة فيها. ولم تستخدم الصور ثنائية الألوان والصور متعددة الأطياف لعدم صلاحيتها لهذه الخوارزمية.طبقت بعض العمليات الأمنية على البيانات المخفاة لتحسين أداء الخوارزمية ولزيادة أمنية البيانات.

1. Introduction

That information hiding can be classified into, Stenography and digital watermarking. Each of them has its related classifications, can see **Fig.(1)**.

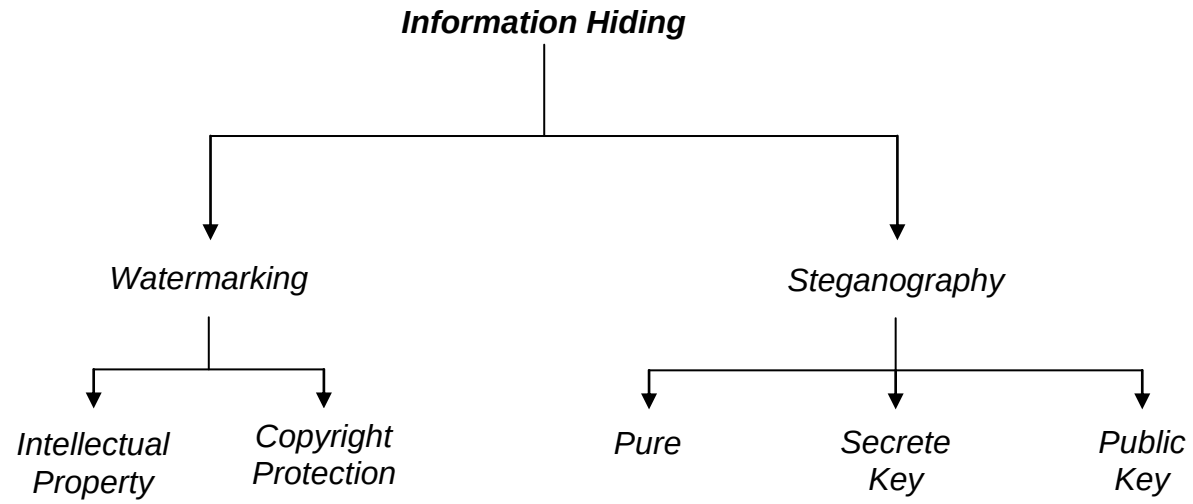

Figure (1) Information hiding

1-1 Steganography

Most of the existing steganography algorithms are performed in pixel domain as it provides more embedding space (capacity), reliability, and controllability in encoding/decoding of the hidden message, and where BMP files deals with the image at the pixel domain [1]. The most of the digital data is represented in compressed form for reduction of storage space and transmission cost, but there are risks of losing embedding data [2].

The goal of Steganography is to avoid drawing suspicion to the transmission of a secret message [3].

The public goal is to embed a secrete message in a digital cover by using a private technique for each of them .there are three classified of steganography types as (1) Secret Key Steganography which is cannot any one know the hidden message unless he has the key. The stegobject contains the cover, hidden message and the secret key (2) Public Key Steganography. The public key is stored in public database, whereas the public key is used in the embedding process, the secret key is used to reconstruct the secret message [5], The Stegobject contains the cover, hidden message, private key, and the public key. The private key dose not needs to be agreed upon by the source and Destination prior to imprisonment in this technique (3) Pure Steganography stegobject contains the cover and the hidden message only. In this type the algorithms hide information in a digital cover without using any types of key [6].

Steganography allows for authentication, copyright protection, and embedding a message in an image or another types of cover.There are new kinds of covers such as (1) Hiding in Network (2)Hiding in Image (3) Hiding in Disk Space (4) Hiding in Text (5) Hiding in Audio

The most Simplified of several kinds of image steganography are : (1)Compression Techniques (2) Noise insertion (3) LSB insertion (4) Border Modification .

Steganalysis involves two aspects, detection and distortion of embedded message, which is attacks against hidden data [4], or discovering and rendering useless such covert message [7].

2. Hide and Extract Algorithm

The algorithm can be divided into two main procedures Hide and Extract. The Hide procedure can be divided into three algorithms (Search, Compress, and Hide) and the Extract procedure can be divided too into three algorithms (File Extract, Decompress, Text Extract), Search algorithm has two inner operations (Image Stretch, and Range Compression) as shown in **Fig.(2)**.

Developed programs are built using Visual Basic programming language. Hide procedure is used for comparison the image with the text and recording results, transforming them, and hiding them in the image (cover). Extract procedure is used for extracting the hidden information from image border by extracting the data from stegobject, re-transforming them, and extracting the text from the image according to the re-transformed data. Developed algorithm takes advantage of images with dark background or boundaries, these images allow us to hide our secret message in the border of the image without drawing any suspicion about them. In this approach, secret information is a text (English/Arabic), the covers are gray scale or colored images. Binary and multispectral images will not be used in this approach because their data types are not appropriate for this approach. This algorithm was included with some of security operations in order to avoid steganalysis attacks.

2-1 Search Algorithm

Two operations of security (Image Stretching and Range compression) are applied to the data in order to increase their security, and improve algorithm performance.

Stretches the image and research's image pixels for each character of the text. If there is a pixel value that equal to the character ASCII code, pixel position will be separated into three bytes and stored in a file as previously explained. Else, instead of the three separated bytes, three bytes will be stored in the same file. First one is the complement of the character ASCII code, (i.e. 255-character ASCII code), will be stored in the same file of the positions. Two operations of security (Image Stretching and Range compression) are applied to the data in order to increase their security, and improve algorithm performance.

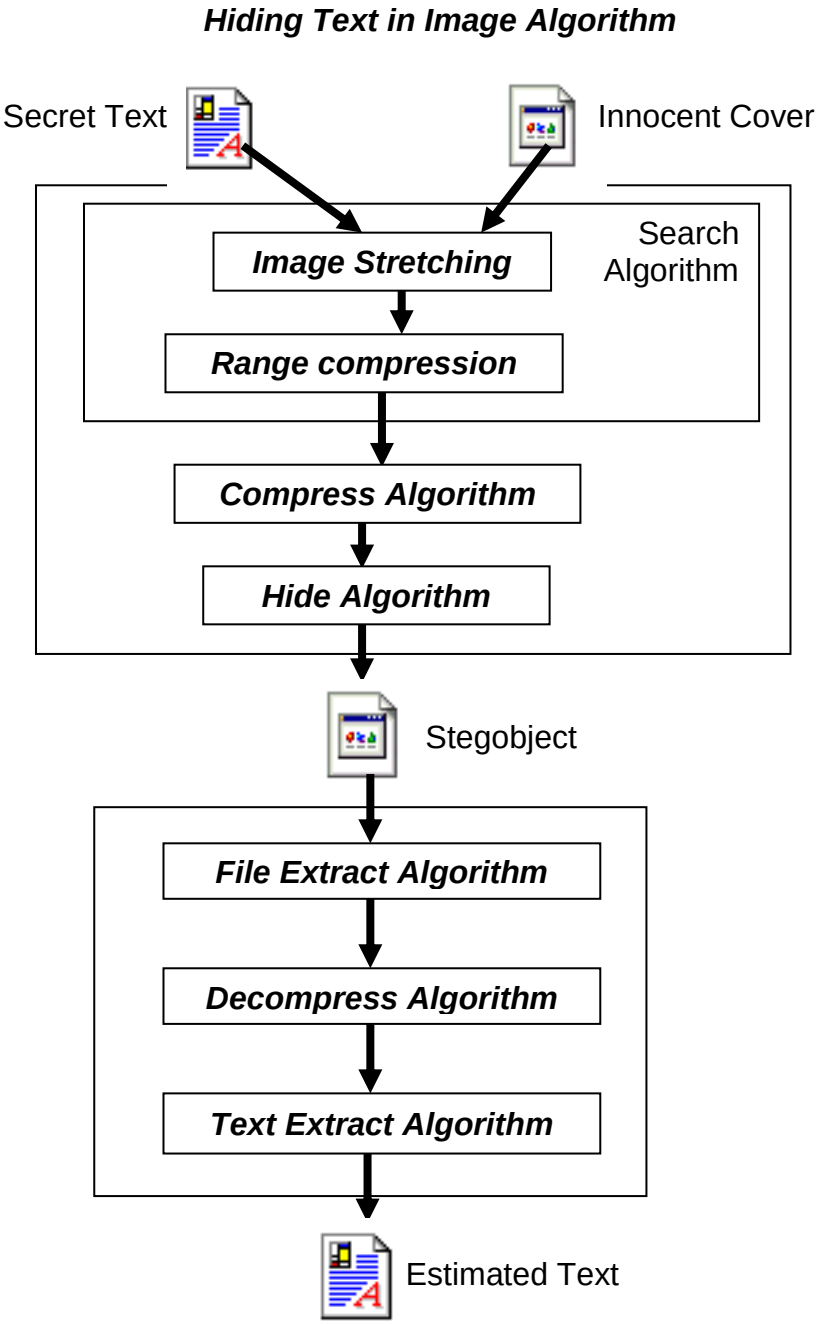

Figure (2) Block diagram of the developed algorithm

2-1-1 Stretch Algorithm

Image stretching is one of image enhancement operations. Image pixel values can be expanded, compressed, or moved to any other range (within 0-255 range) in order to explain some unseen details of the image.

Stretch equation subtracts the minimum number from each number, divides subtraction result by numbers range (maximum number-minimum number), and multiplies division results by the stretched rang. This operation can convert numbers from original to another range, which starts with (0). If beginning value of the stretched numbers does not desired to

be (0), the beginning number can be determined by adding starting value to multiplying result. The general form of stretching equation will be [4]:

$$\left[A_{st} = \frac{A_{old} - min}{max - min} * Z_{st} + start \right] \quad \text{..... (1)}$$

where:

A_{st} : Result number from stretch equation applying.
A_{old}: Original pixel value, which is desired to be stretched.
min: Smallest value among numbers, which are desired to be stretched
max: Largest value among them
Z_{st}: Result stretched range, into which pixel values may be stretched
Start: Starting number of Z_{st}

2-1-2 Range compression

Pixel position is variant in a large range, which may exceeds the range of pixel value (0-255), i.e. it cannot be stored in a byte, it must be transformed into the suitable range. The Suggested transforming method is to separate each two digits of the position value in a byte, this separation can be done by the following set of equations (2)-(3), which are implemented for this purpose. For images that dose not exceed (999999) bytes in size, pixel position can be divided into three bytes. After separation, these bytes will be stored in a file as a reverse manner as follows:

$$b = \frac{a}{10000} \quad \text{..... (2)}$$

Separating the first two digits by decimal point:

$$b_1 = int(b) \quad \text{..... (3)}$$

Cutting the two separated in equation (2) digits using integer operator:

$$b_{11} = (b - b_1) * 100 \quad \text{..... (4)}$$

Separating the second two digits from resultant number in equation (2):

$$b_2 = int(b_{11}) \quad \text{..... (5)}$$

Cutting the two digits that separated in equation (4)

$$b_3 = (b_{11} - b_2) * 100 \quad \text{..... (6)}$$

Taking the reminder two digits from resultant number in equation (5):

where:

a: the original pixel position from the image to be separated

b_1, b_2, b_3: the resultant numbers of separation

b_{11}: temporary variable for equations requirement

For more security, the resultant numbers (b1, b2, and b3) will be stored as (b3, b2, and b1). For example, if there is an arbitrary position in the image file like 12869, which will be displayed at the coordinates (22, 21) of (584,364) image, see **Fig.(3)**, it will be separated into 1, 28, and 69, by applying equations (2)-(6) as shown below. These numbers will be stored as 69, 28, and 1:

a= 12869 pixel position in the image file

b= 1.2869 separating (1) from the other digits, Equation (2)

b_1= int (1.2869)=1 cutting (1) from the other digits, Equation (3)

*b_{11}= (1.2869-1)*100=28.69 separating (28) from the remainder digits, Equation, (4)*

b_2= int (28.69) =28 cutting (28) from the remainder digits, Equation (5)

*b3= (28.69-28)*100=69 separating and cutting (69) from the remainder digits,Equation (6)*

Numbers like (69, 28, and 1) may give different imagination about the original number (12869).

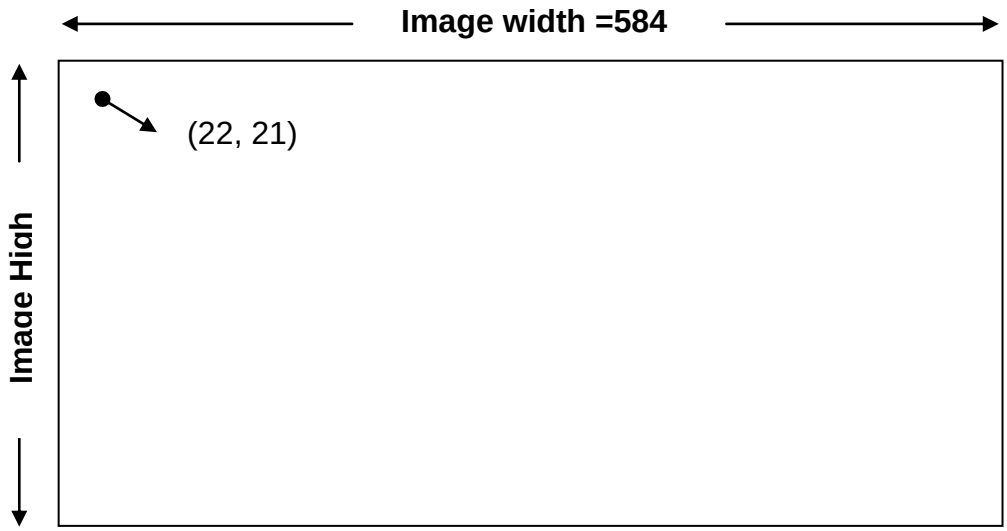

Figure (3) Image pixel with 12869position and (22,21) display coordinates

2-1-3 Algorithm Steps

Stretches the image and seeks image pixels for each character of the text. If there is a pixel value that equal to the character ASCII code, pixel position will be separated into three bytes and stored in a file as previously explained. Else, instead of the three separated bytes, three bytes will be stored in the same file. First one is the complement of the character ASCII code, (i.e. 255- character ASCII code), second one is a random number RND of (0-99) range, which is used for disguise only, and the third is (0) value as a flag for knowing that these three values are not separated position. Algorithm steps can be explained as:

- Start
- Open (image, text, position) files
- Stretch the image file

Loop

- Read text Character
- Search image file for pixel value =character ASCII code
- If found Then
- Split pixel position into three bytes, two digits for each
- Else
- Split Character ASCII code into three bytes as
- (255 – character ASCII code)
- RND of (0-99) range(0)
- End If
- Print results in Position file
- If not end of text file
- Go to loop
- End if
- End.

2-2 Compress Algorithm

Position file, which result from Search algorithm executing contains numbers of (0-99) range. Hidden data of this range cause large variation in border colors. **Figure (4-a)** explains image with hidden data of (0-99) range, in which **Fig.(4-b & c)** is an expanded portion of the border with perceptible color differences, they may draw a suspicion about the hidden information. This range must be reduced in order to reduce color variation. Numbers can be transformed by replacing them with distances of central number as shown in **Table (1)**. Each number (X) of (0-99) range will be replaced with two numbers. First one is (X-50), second one is RND, which is used to explain whether X is larger or smaller than (50), i.e. if X is larger than 50 RND will be of (26- 50) range, else it will be of (0-25) range. Distances of (50) will transform (0-99) range into (0-50) range, i.e. (99-50) will transform (99) into (49) and all the resultant numbers will be less or equal to (50). Numbers of (0-50) range are still cause noticeable color variation. For more transforming, distances between (0-50) range numbers

and (25) will transform (0-50) range into (0-25) range. Each number (Y) of (0-50) range will be replaced with two numbers. First one is the distance of (25), and the second one is RND to explain if Y is larger or smaller than (25), i.e. if Y is larger than 25, RND will be of (13-25) range, else it will be of (0-12) range.

Resultant numbers from search algorithm will be replaced with four numbers of (0-25) ranges. For resultant numbers from previous example (section 3-2-1-2) 69, 28, and 1, **Table (1-a)** explains original numbers, their distances of (50), and RNDs as indicators. **Table (1-b)** explains resultant numbers from table (1-a), their distances of (25), and RNDs as indicators. **Table (1-c)** explains original numbers and the final results from this transformation, which will be stored in compress file. Border with hidden data of (0-25) range will appear as black border.

Figure (5-a) explained stegobject with hidden data of (0-25) range. **Figures (5-b & c)** are expanded portions of the border without perceptible color differences, it may prevent drawing a suspicion about the hidden information, especially if they will be hidden in an image with dark background. Following steps can perform this transformation:

- Start
- Open (Position, Compress) files
- Let st (1 … 4) be a matrix of four elements

Loop

- Read x (from position file)
- Let the first element in St be the absolute value of the distance of (50) (St(1) = x – 50)
- If x is grater than (50) Then
- Let the third element in St = RND of (26-50) range as indicator
- Else
- Let the third element in St = RND of (0-25) range as indicator
- End If
- For each of the first and third element in st
- Let the element be absolute value of the distance of (25)
- If the element is grater than 25 Then
- The successor element in st = RND of (13-25) range
- Else
- The successor element in st = RND of (0-12) range : End If
- Store st elements in Compress file
- If not end of Position file
- Go to loop : End If
- Store RND of (26-30) range in Compress file as indicator of the ending of the hidden data
- End.

Table (1) Range compression

(a) Distances between the numbers and (50)
(b) Distances between (a) numbers and (25)
(c) The final number of a & b, for each original number, four numbers will be stored

(a)	**Distances of 50**	
Numbers	**Distance of 50**	**RND (indicator)**
69	19	38
28	22	7
1	49	19
(b)	**Distances of 25**	
Numbers	**Distance**	**RND**
19	6	6
38	13	15
22	3	9
7	18	8
49	24	19
19	6	2

(c)	**Results Numbers**			
Numbers	**Stored as**			
69	6	6	13	15
28	3	9	18	8
1	24	19	6	2

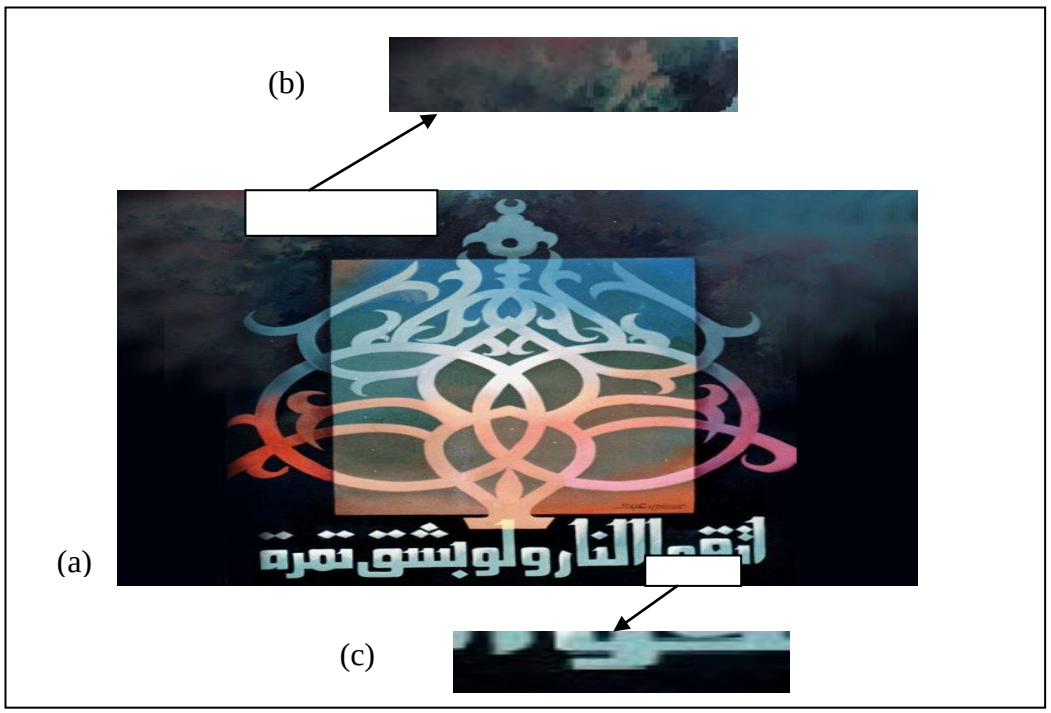

Figure (4) (a) Image with hidden data of (0-99) range (b) & (c) Expanded portion from the top & bottom of image border, color differences in the image border can be easily

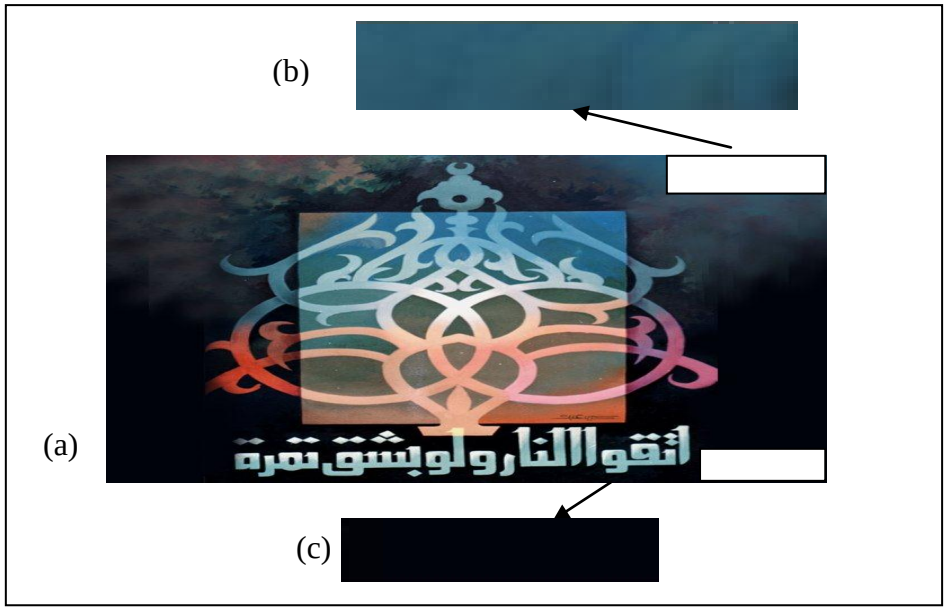

Figure (5) (a) Image with hidden data of (0-25) range (b) & (c) Expanded portion from the top & bottom of image border with unnoticeable color differences in the image border

2-3 Hide Algorithm

After applying Compress algorithm, result numbers will be stored in the image border. For more security, each three numbers will be stored in a reverse manner. In order to give a fancy fashion about these numbers, they will be stored in the top, bottom, left, and right of the image border respectively in each time, see **Fig.(6)**, this can be done according to the following steps

- Start
- Open (Compress, image) files

Loop

- Read three numbers from Compress file
- Hide them in one side of the image border in a reverse manner
- If not end of Compress file Go to loop.
- Hide RND of (26-30) as indicator to the data end
- fill the reminder of the border with RND of hidden data range (0-25)
- End.

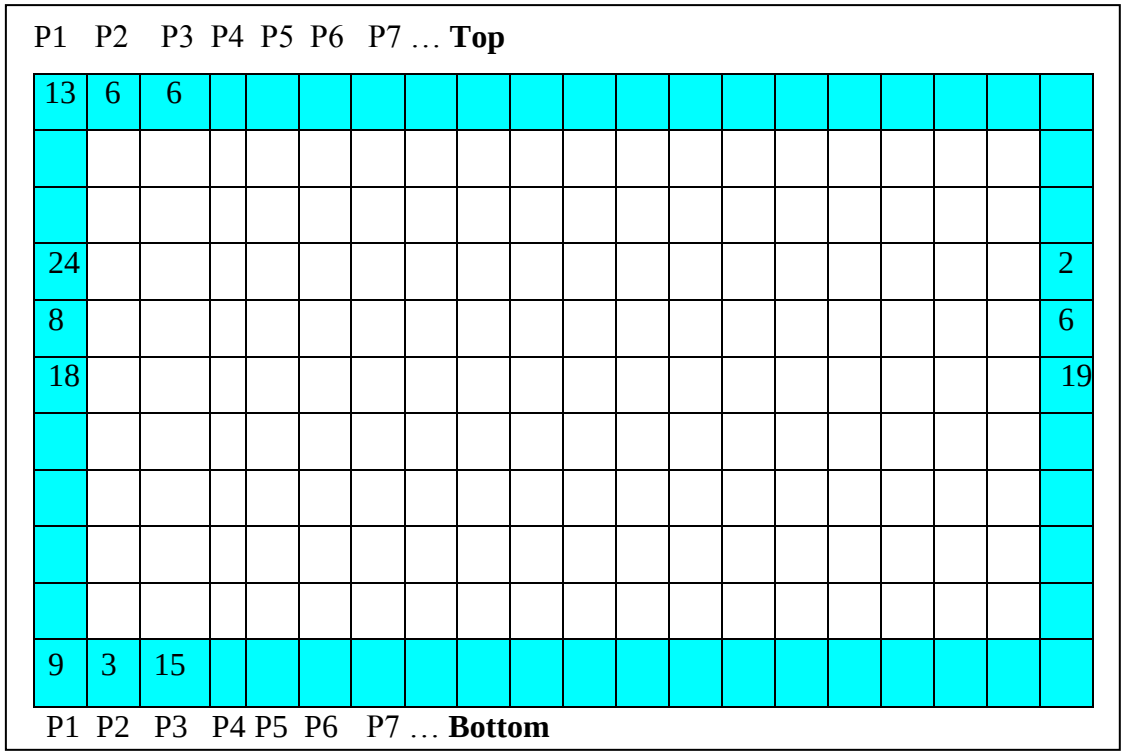

Figure (6) Storing data in the image border in unordered manner

In brief, Hide procedure can be explained as follows, a character ASCII code in exact position will be found-after stretching-in another position (12869 as example). It will be encrypted and stored in the image border as (12) numbers {(13, 6, and 6) in the top, (9,3, and 15) in the bottom, (24, 8, and 18) in the left, and (2,6,and 19) in the right) of the image border. This may give completely different imagination about the original character.

3. Extract Procedure

Extract procedure tends to recover the hidden information from the image. Extract procedure contains three algorithms File Extract, Decompress, and Text Extract, which are the opposite of Hide procedure algorithms Hide, Compress, and Search respectively in order to reveres effects of each of them.

3-1 File Extract Algorithm

The File Extract algorithm tends to reveres the effect of Hide algorithm. It reads three numbers from each of top, bottom, left, and right of the image border, each three numbers will be stored in the file in a reverse manner as an opposite of Hide algorithm. Results will be stored in Recover file in order to use them in Decompress algorithm. File Extract algorithm can be performed according to the following steps:

- Start
- Open (Recover, image) files

Loop

- Read three numbers (n_1, n_2, and n_3) from the one side of the image border
- Store them in Recover file as (n_3, n_2, and n_1)
- If not end of hidden Data
- Go to Loop
- End if
- End.

3-2 Decompress Algorithm

The Decompress algorithm tends to reverse Compress algorithm effect by reconverting numbers from (0-25) range to (0-99) range and stores results in Decompress file. Four numbers will be taken. As previously explained, the second and fourth numbers are indicators. They explain if the first or third numbers are larger or smaller than (25). If the indicator is less than 12, number will be subtracted from (25). Else the number will be added to (25), i.e. two numbers of (0-50) range will be obtained. The second one is indicator. If it is less than 25 the first will be subtracted from (50). Else it will be added to (50).

Each four numbers that obtained from Recover file (results of File Extract algorithm) will be reconverted to one number of (0-99) ranges. The following steps can do this:

- Start
- Open (Recover, Decompress) files

Loop

- Read four numbers (m_1, m_2, m_3, and m_4) from Recover file
- For the first and the third number do
- If the indicator (successor number) is less than 12
- Subtract the number from (25)
- else
- Add it to (25)
- end if
- If the second of the two result numbers is less than (25)
- Subtract the first one of them from (50)
- else
- add it to (50)
- end if
- if not end of Recover file
- Go to loop
- End If
- End.

3-3 Text Extract Algorithm

Text Extract algorithm is the opposite of Search algorithm, where the pixel positions were divided into three numbers. Each three numbers from Decompress file, which result from Decompress algorithm, will be constructed into one pixel position by equation (7). Equation (7) will reverse the effect of equations (2)-(6):

$$pos = x3*10000 + x2*100 + x1 \quad \text{..... (7)}$$

After getting the positions and before extracting the character from the position (pos) of the image, it must be stretched according to equation (1) to obtain the stretched image where the pixels were searched for character ASCII code. The corresponding characters of resultant numbers in ASCII table will be stored in Estimated Text file, which is the estimation of the original hidden text. The estimated text can be obtained via next steps.

- Start
- Open (Decompress, Image, and Estimated Text) files
- Stretch the image

Loop

- Read three numbers (X1, X2, and X3) from Decompress file
- Construct position by equation (pos = X3 * 10000 + X2 * 100 + X1)
- Read pixel value (Y) from the position pos of the image file
- Get the corresponding character of (Y) in ASCII table
- store results in Estimated Text file
- If not end of Decompress file
- Go to loop
- End If
- End.

4. References

1. S., Areepongsa, Y. F., Syed, N., Haewkamnred, and K. R., Rao, ***"Steganography for Low Bit-Rate Wavelet Based Image Coder"***, University of Texas, 2000.

2. Neil, F. Johnson, and Sushil, Jajodia, ***"Steganalysis of Image Created using Current Steganography Software"***, Information Hiding Second International Workshop, Springers, 1998.

3. Neil, F. Johnson, Zoran Duric, and Sushil, Jajodia, ***"Information Hiding: Steganography and Watermarking-Attacks and Countermeasures"***, Kluwer Academic Publishers, 2001.

4. Sowers, Sabrina, and Youssef, ***"Testing Digital Watermarking Resistance Destruction"***, Information Hiding Second International Workshop, Springers, 1998.

5. Neil, F. Johnson, ***"History and Steganography"***, Internet Survey, 2000.

6. A. S., Rahma, and M. J., Jawad, ***"Using Duplicating Palettes Method in BMP Images for Information Hiding"***, Journal of First Workshop on Information Hiding technologies, 2004.

7. Christian Cachim, ***"An Information Theoretic Model for Steganography"***, Internet Survey, 2001.