Steganography on a Web Environment

Orooba Ismail Ibrahim
Ali Majeed Hasan
ABSTRACT

Steganography is the art of hiding information through digital content in the internet. Messages can be hidden inside all sorts of cover information: text, images, audio, video and more, making web pages a primordial means of action. Most currently steganographic techniques hide information inside images, as this relatively easy to implement but all resources are exploitable. These base images texts or audio are called covers and the result of joint of the cover and the message to hide is called stego. The most important property of a cover source is the amount of data that can be stored inside it, without being perceptible. When a cover is distorted, the cover source will be suspicious and may check more meticulously. A cover with a secret message inside can easily by spread over the world wide web or in newsgroups.

Extensive use of internet increased the interest in developing algorithms and techniques of hiding information. Theses can be categorized in three groups: i) spatial domain techniques like LSB embedding, LSB matching, pixel value differencing, and stochastic modulation; ii) transform domain techniques like utguess, F5, pathchwork, and BST-based Robust Image hiding method; iii) other techniques which are not in spatial or transform domains techniques such as spread spectrum, statistical and distortion.

We propose a new method to embed the text in Image, the method depend on the ASCII code value of character and compare it with the value of the palette of picture, if the ASCII Code value and value of palette is equal, we hide the place of value palette in another place in the picture, in this method we hide place not character. Then load the cover picture in web site on Internet in special website or through e-mail among group of pictures and we can change method of load between time and another.

The main goal of steganography was fulfilled since the security text convey save by Internet.

Keywords: Image, web site, steganography, text in Image steganography, Information Hiding
الخلاصة

علم الاخفاء: هو علم اخفاء المعلومات خلال محطات رقمية في رسائل الإنترنت ممكن أن تكون مخفية داخل كل أنواع المعلومات التي تستخدم للأخفاء سواءً كانت نص أو صورة أو صوت أو صور أو صورة ناطقة أو غيرها جعل الشبكة العنكبوتية وسيلة فعالة للأخفاء. اغلب وسائل الاخفاء الحالية تخفى المعلومة داخل الصورة نسبياً سهلة التشفير ولكن كل المصادر أصبحت مستغلة. الأساسي النصوص الصورة إذا الحقت بدي الغطاء والنتيجة التي تلحق الغطاء بالرسالة المخفية تدعي stego.

أكثر خاصية مهمة للغطاء هي كمية البيانات التي تخزن داخلها. بدون ان تلاحظ عندما يكون مصدر الغطاء مشوه سوف يثير الشكوك ويفحص بطريقة أكثر دقة.

الغطاء مع الرسالة السرية في داخله يمكن نشره بسهولة عبر الشبكة العنكبوتية، الاستخدام الواسع للإنترنت ادى إلى زيادة الاهتمام في تطوير خوارزميات وسائل اخفاء المعلومات يمكن تقسيمها إلى ثلاث مجموعات:

1- أساليب المجال الفارغ مثل الاخفاء في البت الاقل اهمية , مطابقة البت الاقل اهمية stego
2- أساليب تحويل المجال مثل
3- الأساليب الأخرى التي هي ليست في البت الاقل اهمية أو أساليب التحويل

نحن نقترح أسلوب جديد في اختفاء نص في صورة مستخدمان طريقة الاخفاء في البت الاقل اهمية
لكن بطريقة مختلفة تعتمد على مقارنة قيمة الاسكي للرمز ومقارنتها مع قيمة البليت لنقطة في الصورة فاذا تساوت يتم اختفاء مكان النقطة التي ساوت قيمة الاسكي للرمز في مكان اخر من الصورة ومن ثم نشر الصورة عبر الإنترنت بالاتفاق مع الطرف الثاني اما في موقع معين منطق عليه أو عبر الايميل على ان تكون ضمن مجموعة من الصور ويمكن تغيير طريقة النشر عبر الإنترنت بين فترة وآخرة

الهدف الرئيسي هو انجاز توصيل النص السري بطريقة امنة عبر الإنترنت
1 Introduction

Steganography is the practice of hiding information in computer pictures or music and relies on the fact that digital image and MP3 music files are made up of thousands of pieces of binary code, which tell a computer to color a pixel or produce a sound. Because so many small pieces of digital information are involved, a handful can easily be altered to convey secret messages, without changing the overall effect to the naked eye or ear [1].

Normally, the secret information is stored in the least significant parts of an image or tune. In a holiday photo, for example, dozens of pixels in the background could be changed to convey an airline schedule. To a casual observer, or even an FBI investigator, the picture would appear completely innocent, as the vast majority of the pixels are not changed. Anyone who knew where to look, however, would be able to access the information hidden in the altered pixels, which can then be pieced together and read as normal. It is a relatively simple practice and can be done with software available in high street shops or downloaded from the Internet [1].

The obvious reason for using steganography as opposed to cryptography is that anything encrypted would immediately draw attention that information is being deliberately concealed, whereas a message sent using steganography, by all appearances, is just another Gif image or MP3 file [3].

2 Uses of Steganography

Steganography can be used anytime you want to hide data. There are many reasons to hide data but they all boil down to the desire to prevent unauthorized persons from becoming aware of existence of message. In the business world, steganography can be used to hide a secret chemical formula or plans for a new invention. Steganography can also be used for corporate espionage by sending out trade secrets without anyone at the company being any the wiser. Steganography can also be used in the non-commercial sector to hide information that someone wants to keep private. Spies have used it since the time to pass messages undetected [1].

The healthcare industry and especially medical imaging systems may benefit from information hiding techniques. The use standards such as DICOM digital imaging and communication in medicine (which separates image data from the caption, such as the name of the picture is lost, thus, embedding the name of the patient in the image could be a useful safety measure [5].

3 Digital Image

A digital image is a two dimensional function, \( f \), that takes an input two spatial coordinates \( x \) and \( y \) and returns a value \( f(x, y) \). The value \( f(x, y) \) is a gray level of the image at that point. The gray level is also called the intensity.
Digital images are a discretized partition of the spatial images into small cells which are referred to as pixels -picture elements.

4 The new Proposed System
The new proposed system take as input the embedded-object which in our case file of text and the cover object which is either a 256 color-image or gray scale image (size 640 x 480 )
We use in our case an 8-bit image and since 8-bit image and since 8-bit values can only have a maximum of 256 colors the image must be chosen much more carefully .

5 Background
Steganographic software is new and very effective .Such software enables information to be hidden in image, sound and apparently “blank” media .

In the computer, an image is an array of numbers that represent light intensities at various points in the image .A common size is 640 by 480 and 256 colors )or 8 bits per pixel .(Such an image could contain about 300 kilobits of data [2]
There are usually two types of files used when embedding data into an image .The innocent looking image which will hold the hidden information a “container.” A “message” is the information to be hidden .A message may be plain-text, cipher text, other images or any thing that can be embedded in the least significant bits )LSB (of an image.
For example :Suppose we have a 24-bit image 1024 x 768 )this is a common resolution for satellite images, electronic astral photography and other high resolution graphics .(This may produce a file over 2 megabytes in size) 1024x768x24/8 =2,359,296 bytes .(All color variations are derived from three primary colors, Red, Green and Blue .Each primary color is represented by 1 byte) 8 bits .(24-bit images use 3 bytes per pixel .If information is stored in the least significant bit )LSB (of each byte, 3 bits can be a stored in each pixel .The “container” image will look identical to the human eye, even if viewing the picture side by side with the original .Unfortunately, 24-bit images are uncommon )with exception of the formats mentioned earlier (and quite large .They would draw attention to themselves when being transmitted across a network .Compression would be beneficial if not necessary to transmit such a file .But file compression may interfere with the storage of information .

6 The New System Operation
The embedding process of new system has many operations:
1- Select cover image, by taking the image form group of picture stored previously.
2- Input the text to be hidden that can be done by open new file and entered directly or by select a previous stored file.
3- Open –cover (bmp file ) and split the body to blocks , we split the body in order to obtain small number of position to be easy in hide.

Split the body of the image file operation

This operation will split the body image in equal blocks to use these blocks in hide text.

Algorithm : split the body of the image file

Input : The body of the image file
Output : The file that contains the block of the body

Step1 : Divide the width of the image by 20 will =32 block.
Divide the height of the image by 20 will =24 block.

Step2 : For K=1 to 24 do
    For j=1 to 20 do
        Part-po=0;
        Call procedure dis-lin,
        End for j
    End for k

Step3 : For j=1 to 32 do
    Save each block into the output file.
    End for

End algorithm

4- Find the position where to hide the bytes of the text in the blocks of image(cover ) (and store the block numbers and the number of position in file to use later.

This operation is done by read a block from block image file and test if block is suitable to hide a byte from text file or not . The testing is done by compare the value of pallate with ASCII of character . Then we find the position that we hide the character and save the block number and position in file.

If the value of character not found we convert it to captial if it is small and to small if it is captial.

Algorithm : Search

Input : a (The file that contains the sperated block of the body image)
b (The file that contains the text to hide)
Output : The file that contains the position

Step1 : find the number of character in the text file and put in count.
Step2 : Initialize the value of j=0
Step3 : While j<=count do
    Read a block from the block image file
    Test if the block is suitable to hide a byte from text file or
    Not (call procedure block-test).
    If the block is suitable then
        Read a character from the text file.

Find the position in the block where the byte.
Will be hide( call find pos procedure).
Save the position where the text byte to hide and
The block number in a file.
Increment the value of j by 1.
End if
Increment the value of block-no by 1.
End while

Procedure block – test
Begin
Search in the block neglate the first row and last row in each Block.
For  I=20 to 380
  If the(Sorc[h] >=60 and  Sorc[h] <=130 (then block return
  the test result as true
  else
  return it as false
Endif
Endfor
End

Procedure find pos
Begin
Step1 :For I=20 to 380 do
  If Sorc[h] = ASCII value of the character you want to
  Hide then pos=I; (100 %much)
End if
Step2 :If pos-=1
Begin
  C=0;
  If (((c<20 (and )ch<97 )))
    ch=ch+32; (convert the character from capital to small)
  If (((c<20 (and )ch>=97 ()
    ch=ch-32; (convert the charcter from Small to capital)
End if
  For )I=20; I<380; h (++
    If sorc]I=[ch then pos=I;
Endif
End
5- Substitute the character in the position get from the search program, in fact the substitution it merely locate the position that hide in it.
This operation is done by read character from text file and locate the position were to hide the character in the block.Replace the character in the specific position and hide the position of the replaced character in the first row and last row of the block.

Algorithm : Substitute
Input : a (The file that contains the text).
        b (The file that the block of the image).
        c (The file that contains the positions).
Output : The file that contains the block of the image.

Step 1: For I = 1 to the no-of char-text
    Read the block-no and position-number.
    Find the specified block-number.
    Call the procedure sub-pro.
Endfor

Step 2: Write the resulted file in the output file.

6- Hide position will hide the block numbers where the character was hidden.

Algorithm : Hide-pos
Input : a (The file that contains the block of the body image)
        b (The file that contains the block numbers where the bytes of the character were hidden)
Output : a (The file of blocks of the body image)
        b (The file that contains the key)

Step 1: Open the file of the block-number.
Step 2: Read the contains of the file in an array.
Step 3: Find the first block no position in the block file.
Step 4: Save the second block no in the current block in the first row (the second half of the first row) and in the last row (the second half of the last row).
Step 5: Steps 3 and 4 will be performed on all of the chain.
End.

7- Combine the blocks into a one file and then combine the header and the body file into a file to perform the stego-object bmp file.

Algorithm : Extracting text
Input : Bmp file that contain the text, the key
Output : The text file
Step 1: Open the bmp file by reading the header and getting the body of the file.
Step 2: Split the file into blocks.
Step 3: By using the key get the first block position that contains the first byte of the text.
Step 4: Read the second byte of the key (which is the length of the text that is hidden in the image).
Step 5: For \( I = 1 \) to the length of the message do
Step 6: Find the position of the embedded character from the first row and the last row of the block.
   Get the row position from the first row.
   Convert the binary value to a decimal value.
   Get the column position from the last row.
   Convert the binary value to a decimal value get the position by applying the following equation:
   \[ \text{Pos} = (r + 1 \times (20) - 20 - c) \]
   Get the character value from the specific position.
Step 7: From the current block find the position of the next block.
   The number of the block is hidden in the first row (the second half) and the second half of the last row.
Step 8: End for
End algorithm Extracting text.

8- The load image on special web site

The new proposed system has a key which is used to extract the embedding image from stego-image. Two types of key were used, the first one is the name of file image that it is known by recipient, the second one is the size of text file and number of position we began hide in it the second key is transferred separately.

The following steps describe the algorithm:

-Open Cover (the bmp-file) Operation
   This operation will open the bmp file and save header in a file and save the palette value of body in another file.
-Split the body of the image file operation
   This operation will split the body image in equal blocks to use these blocks in hide text, we split the body in order to obtain small number of position to be easy in hide.
-Find the position operation this operation is done by read a block from block image file and test if block is suitable to hide a byte from text file or not. The testing is done by compare the value of the palette with ASCII value of character. Then we find the position that we hide the character and save the block number and position in a file.
-Substitute the character operation this operation is done by read character from
text file and locate the position were to hide the character in the block. Replace the character in specific position and hide the position of the replaced character in the first row and last row of the block.

To find the coordinates (i.e., row and column number) of position we apply the following equations:
\[ R = \text{pos div } 20 \] ……………… .1
\[ R : \text{row} \]
Pos : position in array
The equation for the column position is:
\[ C = \text{pos mod } 20 \] ……………… .2
\[ C : \text{column} \]
We substitute the row value that is calculated, from equation 1 into the current block that we search on it. We substitute it into first row (in LSB). While for the column value that is calculated from equation 2, we substitute it into last row of the current search block. We substitute it into LSB, in this operation the substitute of character don’t change any thing in the pixel because the value of pixel in the palette equal to ASCII value of character, only change the places in the picture that we substitute the position of character in it. This change is unnoticeable because the number of position is small and substitute in LSB.

7 Extracting The Embedding Text

Extracting the text is done by using the key stored into first block position that contains the first byte of the text and read the second byte of the key which is the length of the text that is hidden in the image.

Later get the row position from the first row and convert the binary value to a decimal value and in the same way we position by applying the following equation
\[ \text{Pos} = (r+1 \times (20) -20-c) \]
Get the character value from the specific position.
a -The following steps describe the algorithm:
-Open the bmp file by reading the header and getting the body of the file
-Split the file into blocks
- by using the key get the first block that contains the first byte of the text.
-Read the second byte of the key (which is the length of the text that is hidden in the image)
-For I =1 to the length of the message do
-Find the position of the embedded character from the first and the last row of the block.
Get the row position from the first row.
Convert the binary value to a decimal value.
Get the column position from the last row
Convert the binary value to a decimal value, get the position by applying the following equation:
Pos=(r+1 * (20) -20-c)  
Get the character value from specific position.

-From the current block find the position of the next block.
The number of the block is hidden in the first row (the second half) and the second half of the last row.

- The end

8 The Interface of Program for The new proposed system
the Interface of program contains the following options:
1-embedding this option contains four options as a drop list
   - Select cover Image: we select picture save as bmp from group of pictures
   - Select text file: we select text file we saving previously contain the text that we wish to hide it
   - Hide: this option hide the text file in cover Image
   - Show picture: this option show the picture after hide text in it
2-Extracting: this option concludes the text from the Image
   1- Text editor: this option use to input the text file that you want to hide it in image
   2- exit

9 Hide Information in image Detection
A new proposed method to distinguish between pure images and images with hidden text, based on a generated matrix called Smart Matrix.
9.1 The Smart Matrix (SM)

The Smart matrix (SM) is a square matrix generated from original gray-scale levels of the image, whereby its dimension is equal to the dimension of original image. This matrix is produced by taking each pixel in binary form, inverts it, and stores it in new matrix is called inversion matrix, then segment the inversion matrix into a number of pixel frames such that each pixel frame (PF) consists of two neighbor pixels. These frames will overlap when they are viewed in a row vector wise. This means that the second pixel from the first frame represents the first pixel in the second frame in this vector. After completing segmentation process, the second pixel is subtracted from the first pixel in each pixel frame and must set the last column of matrix to zero. The resultant matrix is called (Subtraction Matrix).

Another segmentation process will be performed on subtraction matrix to produce a large number of submatrices of size (2 x 2) without overlapping in both sides (horizontal, vertical), and then each sub matrix is multiplied by a filter with similar size of submatrix (2 x 2). The element of filter has values [1 0; 0 1].

The following algorithm covers the calculation of smart matrix elements:

1. Take a natural image of any size.
2. Invert the value of each pixel in binary form and store it in new matrix (inversion matrix).
3. Segment the inversion matrix into a large number of pixel frames such that each pixel frame (PF) consists of two neighbor pixels.
4. Take these frames and subtract the second pixel from first pixel in every frame and set the last column of the inversion matrix of step 2 to zero. The resultant matrix is called subtraction matrix.
5. Decompose the subtraction matrix into square segments of dimension (2 x 2) without overlapping.
6. Multiply the decomposed matrix by the smart filter of size (2 x 2), having elements [1 0; 0 1].

This smart matrix can be used to distinguish between pure image and images with hidden information in both subjective test and objective test.

9.1.1 Hidden Information Detection using Subjective Test

This algorithm can be applied to the smart matrix to distinguish between the pure image and image with hidden information. By plotting the smart matrix in
three-dimensional form, the image with hidden information from pure image can be distinguished by looking at the z-coordinates of points above a grid in the x-y of the smart matrix.

From the plot, if the elements of smart matrix approach stability (few numbers of spikes) then the image is pure, but for noisy image the elements of smart matrix approach instability (higher numbers of spikes).

Figure (2) represents implementation of smart matrix algorithm. Figure (2a) shows the result of plotting the smart matrix for pure image, and how the elements of smart matrix approach stability (few numbers of spikes). But figures (2b) show the result of plotting the smart matrix for noisy image, and how the elements of smart matrix approach instability (higher numbers of spikes).

Figure (2) Generation of smart matrix (right) Corresponding to the image (left)

9.1.2 Hidden Information Detection using Objective Test

This algorithm will use the smart matrix to distinguish between pure image and image with hidden information using objective test. The procedure of the proposed algorithm followed in this detection is given below:

Step 1: Generate the smart matrix by the same procedure given in section 9.1

Step 2: Take the standard deviation for smart matrix:
\[ \text{Std} = \sqrt{\frac{\sum_{i=1}^{N-1} \sum_{j=1}^{M-1} (C_{ij} - m)^2}{M \times N}} \]

where

- \text{Std} represents the standard deviation.
- \( C_{ij} \) represents pixels of the smart matrix.
- \( m \) represents the mean of the smart matrix.
- \( M, N \) represent the dimensions of the smart matrix.

Step 3: Compare the standard deviation from the previous step with a threshold value. If the standard deviation is less than the threshold, then the image is pure; if it is more than that, then the image contains hidden information. The choice of the threshold depends upon experimental tests, and it is found that the typical value is between 0.08 and 0.1.

10 Experimental Results
The proposed system has been built using Borland C++ and can run on Pentium I computer and above, the setting of screen must be 800 X 600.

The results of the proposed system have been illustrated in the following

Example 1 The cover

The text results size 95 character with space is:
The cover with text

Example 2  The cover

The text results size 65 character with space is:
“this research to hide text in picture by new method very simple”
The cover with text

Conclusions

The proposed system proved to be a good system used to hide a text in image by compare value of palette with ASCII of character and if equal we hide position un another Place then spread on web.

- In the proposed system don't change any thing in the pixels that hide in it but change in another pixels in which the security of system is increased, in case the third person explore any thing in picture he explore, merrily numbers don't he know it meaning.

- The proposed system proved to be easy to use and efficient in terms security and hide every things about the patients.

- When spread on web we can frequently change the method of spread on.
REFERENCES


64