Comparison Steganography in spatial domain of Image

M.Sc. Amer J. Sadiq
Science College For Girls
Abstract

Most of steganography applications are like games played between information hiders and the attackers that they want to know what the hiders hide in the message and the ways that they hides, so for that reason I developed a new method for hide information in way that confuse the attacker to decode or predict how hiding information in the media and what its mean in presentation. I hide the information directly in the image but in different way of traditional information hiding, I divide the image into 2 parts or more (blocks) and hide the information in these blocks in different representation so the intruders (attackers) cannot predict or detect the hidden data. I apply the information hiding measures (robustness, perceptibility, capacity, peak signal to noise ratio-PSNR) and I discuss these terms later in this paper.

الملخص

أكثر تطبيقات اخفاء البيانات هي مشابهة الى لعبة الاحتفاء بين الأشخاص الذين يقومون بأخفاء البيانات والمهاجمين الذينحاولون معرفة هذه البيانات المخفية والطريقة التي قاموا بها بالاختفاء. ولذا السبب قمت باستحداث طريقة لخدمة البيانات بطريقة تشوش المهاجم من معرفة تمثيل البيانات أو النتبة بكيفية خزن البيانات أو كيفية صياغة البيانات المخزنة. في الطريقة المقترحة خزن البيانات مباشرة بالصورة بطريقة تختلف عن الطرق التقليدية.

في الطريقة المقترحة قسمت الصورة إلى جزئين وختن البيانات فيها بطريقة مختلفة عن الطرق الأخرى بحيث يصعب على المهاجم من التكنولوجيا أو اكتشاف البيانات أو الصيغة المستخدمة لتمثيل البيانات المخفية، وطبقت بها كل معايير إخفاء البيانات مثل المدركية الحسية، الاستيعاب، مقاومة ضد التغيير ونسبة الضوضاء إلى الإشارة وسيتم شرحها لاحقا بالبحث.

1- Introduction

Steganography comes from the Greek words steganos, roughly translating to “covered writing”. Steganographic techniques allow one party to communicate information to another without a third party even knowing that the communication is occurring. The ways to deliver these “secret messages” vary greatly.[1]

“The goal of steganography is to hide messages inside other harmless messages in a way that does not allow any enemy to even detect that there is a second secret message present.”

Examples of steganography date as far back as 440 B.C., where Histius was said to shave the heads of slaves and tattoo messages on them. Once the hair had grown back, the message was effectively hidden until the receiver shaved the heads once again.
Another technique was to conceal messages within a wax tablet, by removing the wax and placing the message on the wood underneath. [1]

Being able to communicate messages between two or more parties without raising suspicion is a powerful technique. This is an important and distinguishing feature from the closely related field of cryptography, where plaintext messages are transformed into unintelligible blocks of ciphertext. While cryptography focuses on privacy, steganography is focused on secrecy, and has no interest in the message itself. [4]

The ability to communicate messages without raising awareness can be a powerful technique, and even more so with the concept of plausible deniability. When applied to cryptography and steganography, this term refers to the ability to plausibly deny that a recovered message was ever encrypted or hidden. In cryptography, this concept is closely related to the onetimepad cipher, where a message could possibly transformed into any other message a theoretically unbreakable cipher. In steganography, it is closely related with the random sampling of bits throughout patterns in the file. Being able to plausibly deny the results of some analysis can be used to ones advantage. [4]

Steganography in digital technology is often associated with files which require a human perspective to verify the integrity and quality. This includes the realm of media files (video, audio, images). Image files lend themselves to exploitation particularly well, which we will explore throughout this article. Focus will primarily be on bitmap formatted images. [2]

2-Digital Steganography

A typical digital steganographic encoder is shown on Figure 1. The message is the data that the sender wishes to remain confidential and can be text, images, audio, video, or any other data that can be represented by a stream of bits. The cover or host is the medium in which the message is embedded and serves to hide the presence of the message. This is also referred to as the message wrapper. The message embedding technique is strongly dependent on the structure of the cover, and in this paper covers are restricted to being digital images. It is not required that the cover and the message have homogeneous structure. For example, it is possible to embed a recording of Shakespeare’s lines (an audio stream message) inside a digital portrait of the famous playwright (an image cover).

The image with the secretly embedded message produced by the encoder is the stego-image. The stegoimage should resemble the cover image under casual inspection and analysis. In addition, the encoder usually employs a stego-key which ensures that only recipients who know the corresponding decoding key will be able to extract the message from a stego-image. [3]
Recovering the message from a stego-image requires the stego-image itself and a corresponding decoding key if a stego-key was used during the encoding process. The original cover image may or may not be required; in most applications it is desirable that the cover image not be needed to extract the message. Steganography is not the same as cryptography. In cryptography, the structure of a message is changed to render it meaningless and unintelligible unless the decryption key is available. Cryptography makes no attempt to disguise or hide the encoded message. Steganography does not alter the structure of the secret message, but hides it inside a cover. It is possible to combine the techniques by encrypting a message using cryptography and then hiding the encrypted message using steganography. The resulting stego-image can be transmitted without revealing that secret information is being exchanged. Furthermore, even if an attacker were to defeat the steganographic technique and detect the message from the stego-image, he would still require the cryptographic decoding key to decipher the encrypted message.[3]

3-Characterizing Data Hiding Techniques
Steganographic techniques embed a message inside a cover; various features characterize the strengths and weaknesses of the methods. The relative importance of each feature depends on the application [4].

**Hiding Capacity**: Hiding capacity is the size of information that can be hidden relative to the size of the cover. A larger hiding capacity allows the use of a smaller cover for a message of fixed size, and thus decreases the bandwidth required to transmit the stego-image.

**Perceptual Transparency**: The act of hiding the message in the cover necessitates some noise modulation or distortion of the cover image. It is important that the embedding occur without significant degradation or loss of perceptual quality of the cover. In a secret communications application, if an attacker notices some distortion that arouses suspicion of the presence of hidden data in a stego-image, the steganographic encoding has failed even if the attacker is unable to extract the message. Preserving
perceptual transparency in an embedded watermark for copyright protection is also of paramount importance because the integrity of the original work must be maintained [6].

For applications where the perceptual transparency of embedded data is not critical, allowing more distortion in the stego-image can increase hiding capacity, robustness, or both.

**Robustness**: Robustness refers to the ability of embedded data to remain intact if the stego-image undergoes transformations, such as linear and non-linear filtering, addition of random noise, sharpening or blurring, scaling and rotations, cropping or decimation, lossy compression, and conversion from digital to analog form and then reconversion back to digital form (such as in the case when a hard copy of a stego-image is printed and then a digital image is formed by subsequently scanning the hardcopy.) Robustness is critical in copyright protection watermarks because pirates will attempt to filter and destroy any watermarks embedded in images. Anti-watermarking software is already available on the Internet and have been shown effective in removing some watermarks. These techniques can also be used to destroy the message in a stego-image.

**Tamper Resistance**: Beyond robustness to destruction, tamper-resistance refers to the difficulty for an attacker to alter or forge a message once it has been embedded in a stego-image, such as a pirate replacing a copyright mark with one claiming legal ownership. Applications that demand high robustness usually also demand a strong degree of tamper resistance. In a copyright protection application, achieving good tamper resistance can be difficult because a copyright is effective for many years and a watermark must remain resistant to tampering even when a pirate attempts to modify it using computing technology decades in the future.

**Other Characteristics**: Computational complexity of encoding and decoding is another consideration and individual applications may have additional requirements. For example, for a copyright protection application, a watermark should be resistant to collusion attacks where many pirates work together to identify and destroy the mark.[3]

**PSNR**: Developers and implementers of lossy image compression methods need a standard metric to measure the quality of reconstructed images compared with the original ones. The better a reconstructed image resembles the original one, the bigger should be the value produced by this metric. Such a metric should also produce a dimensionless number, and that number should not be very sensitive to small variations in the reconstructed image.

A common measure used for this purpose is the *peak signal to noise ratio* (PSNR). It is familiar to workers in the field, it is also simple to calculate, but it has only a limited, approximate relationship with the perceived errors noticed by the human visual system. This is why higher PSNR values imply closer resemblance between the reconstructed and the original images, but they do not provide a guarantee that viewers will like the reconstructed image.

Denoting the pixels of the original image by $P_i$ and the pixels of the reconstructed image by $Q_i$ (where $1 \leq i \leq n$), we first define the *mean square error* (MSE) between the two images as
\[
\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (P_i - Q_i)^2 \quad \text{(1)}
\]

It is the average of the square of the errors (pixel differences) of the two images. The root mean square error (RMSE) is defined as the square root of the MSE, and the PSNR is defined as

\[
\text{PSNR} = 20 \log_{10} \frac{\text{max}[P_i]}{\text{RMSE}} \quad \text{(2)}
\]

The absolute value is normally not needed, since pixel values are rarely negative. For a bi-level image, the numerator is 1. For a grayscale image with eight bits per pixel, the numerator is 255. For color images, only the luminance component is used.

Greater resemblance between the images implies smaller RMSE and, as a larger PSNR. The PSNR is dimensionless, since the units of both numerator and denominator are pixel values. However, because of the use of the logarithm, we say that the PSNR is expressed in decibels (dB). The use of the logarithm also implies less sensitivity to changes in the RMSE. For example, dividing the RMSE by 10 multiplies the PSNR by 2. Notice that the PSNR has no absolute meaning. It is meaningless to say that a PSNR of, say, 25 is good. PSNR values are used only to compare the performance of different lossy compression methods or the effects of different parametric values on the performance of an algorithm. The MPEG committee, for example, uses an informal threshold of PSNR = 0.5 dB to decide whether to incorporate a coding optimization, since they believe that an improvement of that magnitude would be visible to the eye.

Typical PSNR values range between 20 and 40. Assuming pixel values in the range \([0, 255]\), an RMSE of 25.5 results in a PSNR of 20, and an RMSE of 2.55 results in a PSNR of 40. An RMSE of zero (i.e., identical images) results in an infinite (or, more precisely, undefined) PSNR. An RMSE of 255 results in a PSNR of zero, and RMSE values greater than 255 yield negative PSNRs. [5]

### 4-The proposed system method

In this section, we explain how proposed method hide information into target content and retrieve information from the target content without damaging it. With our method we use a key; this key is a password and we take every character in the password and multiply it with series of numbers in order to get the status "ab" not equal to "ba" and then extract sub key from that key; this sub key used to determine which bit will choose for hide information.

So the extractor use the same key and sub key that used by the embedder to determine the bit used for hide. For the applications for digital watermarking and steganography, this extraction keys must be shared among embedder and extractor in order to extract a proper hidden bit codes from the target content. Considering the difficulties for secret
key transportation, this method should be applied in situations where the embedder and the extractor are same person or use certification authorities to assure the integrity of the key.

5-The hiding process

In this section I describe how is the hiding process done, after reading the image I divide it into 2 parts or more but I use 2 parts to explain the hiding process. Also I convert the text that I want to hide inside the cover image to what equivalent in binary as array of binary. After dividing the image into 2 parts I used some geometric functions that extracted from sub key taken from the password to determine the hiding function; the hiding function like circles, spirals, and any other geometric shapes.

After that I take the binary array of the text in order to hide in the image, and I apply the geometric function on the both side of the image to generate locations in both parts of the image and these locations are symmetric; for example when hiding function generate location (90, 50) the proposed method take location (90, 50) in part one and part two, after that I take the value of that location.

The covered image I used is a colored image so every pixel (location) composed of 3 byte each byte contain one color; each byte contain one of the colors (Red, Green, Blue). I take every color and convert it what equivalent in binary and I take the last significant bit for symmetric locations and colors in two parts of the image; and put these binary bit values of two parts in two variables c1 and c2.

Now I take the first location in binary array of text that we want to hide and check the value, if the value is one the I will check the values of c1 and c2, if the values of c1 and c2 not equal (c1=1 and c2=0,or c1=0 and c2=1) and the value of the binary array of the text is 1 then I leave the c1 and c2 as it without change, but if it's 0 I will change c1 or c2 or both as shown in the below table

<table>
<thead>
<tr>
<th>Binary array of the text</th>
<th>Encode for the value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,1) or (1,0)</td>
</tr>
<tr>
<td>0</td>
<td>(0,0) or (1,1)</td>
</tr>
</tbody>
</table>

If value of binary array of text is 0 and values of c1 and c2 are ((0, 1), (1, 0)) here I must change c1 or c2 from 1 to 0 or from 0 to 1 but the change is not arbitrary; if c1=1 and I must change its value to 0 but I must make check before change, I check the bit in location 2 in least significant bit of the selected color of selected location, if c1=1 and the bit in location 2 =0 so I change the bit in location 1 to 0 other wise I don’t change it, same work for c2.
6-Hiding Algorithm for the proposed system

Step1: read the image.
Step2: convert the text to binary.
Step3: divide the image into 2 parts.
Step4: call geometric function to obtain symmetric locations in 2 parts of the image.
Step5: read the contents of locations (colors: red, green, and blue).
Step6: convert each color to its binary and obtain c1 and c2.
Step7: compare the values of c1 and c2 with the binary bit of text and do the following:
   If binary bit of text =1 and c1=c2 then check the second bit of selected color in each location according to second bit change the first bit of one of the 2 parts.
   If binary bit of text=1 and c1<>c2  then do nothing.
   If binary bit of text=0 and c1<>c2 then check the second bit of selected color in each location according to second bit change the first bit of one of the 2 parts.
   If binary bit of text=0 and c1=c2 then do nothing.
Other case change one of the bit of the 2 parts of the image.
Step8: if reach end of text go to step 10.
Step9: go to step 4.
Step 10: end.

7-The extracting process of the proposed system

In this section I describe the extracting of hiding data in the covered image, first read the covered image and divide it into two parts, now the extractor have a password ; this password it’s the same password used to hide data by the hider. This password contains a sub key used to select a geometric function that used in hide information process.

After selecting the geometric function, applying this function to obtain symmetric locations in both parts of the covered image, read the content of the location (Red, Green and Blue) convert colors to binary take the least significant bit, store these values of first bit in c1 and c2 and compare the value of c1 and c2 if c1=c2 then its mean here store binary bit 0 of hiding text, if c1<>c2 its mean binary bit 1 of hiding text.

8-Extracting Algorithm for the proposed system

Step1: read the covered image.
Step2: divide covered image into 2 parts.
Step3: call the geometric function to obtain symmetric locations in the 2 parts.
Step4: read the content of locations (colors: red, green and blue)
Step5: convert each color to its binary and obtain c1 and c2.
Step6: compare the values of c1 and c2 with the binary bit of text and do the following:
   If c1=c2 decode binary bit 0 of hiding text.
   If c1<>c2 decode binary bit 1 of hiding text.
9-Experimental Results of the proposed method

In this experiment, we use many images which are 256*192 in size and all these images will describe in figures and give the following psnr results as shown in figure

![Figure 2 PSNR of different images by using our proposed system](image)

As you see the results of PSNR when we hiding the same text in this different images, I get very good PSNR results, between 20 to 40 that mean the quality of resulted covered image is very good and I keep PSNR in range of accepted values.

If I compare these results of our proposed system with a traditional bit insertion methods I get a different gap of PSNR values as shown in figure 3; the cause of this different results is I don’t need to change bits to represent hiding data in the image in many locations and if we need to change we change only one bit like traditional so the result of proposed system already less than or equal to traditional bit insertion methods but not greater, in proposed method I get quality of covered image closed to original image, distribute of hiding data in the image in a way even if the attacker detect there is an information or different in the covered image he couldn’t Know the representation of hiding data.
Figure 3 compare between proposed system method and traditional bit insertion methods

Conclusion

I suggest to use my proposed method in hiding information in image, why; because in my proposed method I use 2 bits to represent 1 bit of hiding text, the representations of these 2 bits is different depending in the bits in cover image, so the attacker cannot predict the representation of bits in the cover image even if he know the image is containing information inside it, also I kept the measure of quality of covered image is between the allowed range (20-40) and these result could not be more than the traditional methods and also the proposed method distribute hiding data in image in a way that it difficult to attacker to determine location of data and the representation of data.
References


