Implementation of Various Techniques in Steganography and Steganalysis

Chirag M. Patel¹, Amit Agrawal², Reena M. Patel³

¹Electronics and Communication Department, Silver oak Collage Of engi & Research
²Electronics and Communication Department, Silver oak Collage Of engi & Research
³Electronics and Communication Department, Silver oak Collage Of engi. & Research

Abstract — There could be some important data that need to be secured (hide) during transmission. Therefore, how to hide the secret messages during transmission becomes an important research issue. Steganography provides a kind of data hiding method that conceals the existence of the secret messages in the media. Today Steganography are mostly used for hiding a data. An image is selected as the media to hide the secret message (text) in it. Steganography techniques have led to an increased interest in Steganalysis techniques. Steganalysis techniques are nothing but investigation of hidden information.

In this work, my goal is to compare a number of steganalysis techniques. Here I go to discuss & implement some of many well known embedding techniques/methods which are used in steganography like spatial domain techniques & transform domain Techniques. I have implemented hide & seek algorithm based on LSB in spatial domain and JSteg. OurGuess based on DCT in transform domain. Here I also discuss about their capacity, delectability (perceptibility), robustness to attacks and their histogram based on results of implementation. I have also implemented some of basic steganalytical schemes associated with “targeted” steganalysis, including visual and statistical attacks.

Keywords—[Steganography; Steganalysis; hide & Seek; Jsteg; Outguess]

I. INTRODUCTION

Steganography is the art and science of secret communication, aiming to hide the existence of the Secret data (message) from a third party. The word steganography comes from the Greek word “steganos” meaning covered (or secret) and “graphy” meaning writing (or drawing). So, steganography literally means covered writing [1]. Steganography simply takes one piece of information and hides it within another. The art and science of writing hidden messages in such a way that no one, apart from the intended recipient, knows of the existence of the message. 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.

Steganography includes a vast array of techniques for hiding messages in a variety of media. Among these methods are invisible inks, microdots, digital signatures, covert channels and spread-spectrum communications. Today, thanks to modern technology, steganography is used on text, images, sound, signals, and more. The advantage of steganography is that it can be used to secretly transmit messages without the fact of the transmission being discovered. Often, using encryption might identify the sender or receiver as somebody with something to hide.

Steganalysis is Detection of steganography by a third party. Steganalysis will analyze whether a given content, contains any secret message hidden into it, or we can say that The Investigation of Hidden Information [2].

Steganalysis is an extremely difficult science, as it relies on insecure steganography. If steganography is to be successful, it should leave no indication that a secret message exists. Thus, if the model has been created successfully, it should be a difficult task for any third party to spot that alteration has occurred.

In terms of development, Steganography is comprised of two algorithms, one for embedding and one for extracting. The embedding process is concerned with hiding a secret message within a cover work, and is the most carefully constructed process of the two. A great deal of attention is paid to ensuring that the secret message goes unnoticed if a third party were to intercept the cover work. The extracting process is traditionally a much simpler process as it is simply an inverse of the embedding process, where the secret message is revealed at the end. The entire process of steganography for images can be presented graphically as two inputs are required for the embedding process.

The next step is to pass the inputs through the Stego-system Encoder, which will be carefully engineered to embed the message within an exact copy of the cover work, such that minimum distortion is made; the lower the distortion, the better the chances of undetectability. The stego-system encoder will usually require a key to operate, and this key would also be used at the extraction phase. This is a security measure designed to protect the secret message. Without a key, it would be possible for someone to correctly extract the message if they managed to get hold of the embedding or extracting algorithms. However, by using a key, it is possible to randomize the way the stegosystem encoder operates, and the same key will need to be used when extracting the message so that the stegosystem decoder knows which process to use. This means that if the algorithm falls into enemy hands, this extremely unlikely that they will be able to extract the message successfully. The resulting output from the stego-system encoder is the stegogramme, which is designed to be as close to the cover work as possible, except it will contain the secret message. This stegogramme is then sent over some communications channel along with the key that was used to embed the message. Both the stegogramme and the
key are then fed into the stego-system decoder where an estimate of the secret message is extracted. Stego-system is shown in fig 1.

II. COMMON ELEMENT OF STEGANOGRAPHY

- Cover medium - innocuous carrier, a file which has hidden information inside of it.
- Embedded message - plaintext, cipher text, images, etc.
- Stego-key - password
- Stego-medium - secret message, the medium in which the information is hidden.
- Cover medium + embedded message + stego-key = stego-medium

![Stego-system](image1)

Figure 1. (a) Stego Image Generated at Sender Side

![Stego-system](image2)

(b) Message extraction at receiver side

III. STEGANOGRAPHY IN VARIOUS MEDIA

The Steganography is possible in various media as shown in fig 2. In my paper I used Image for implementation of steganography.

![Steganography in various media](image3)

Figure 2. Steganography in various media

- Least Significant Bit Insertion
- Masking and Filtering
- Algorithms and Transformations

The most common and popular method of modern day steganography is to make use of the LSB of a picture’s pixel information. Thus the overall image distortion is kept to a minimum while the message is spaced out over the pixels in the images. This technique works best when the image file is larger than the messages file. Many stego tools make use of least significant bit (LSB). For example, 11111111 is an 8-bit binary number. The rightmost bit is called the LSB because changing it has the least effect on the value of the number. The idea is that, the LSB of every byte can be replaced with message bit makes a little change to the overall file. The binary data of the secret message is broken up and then inserted into the LSB of each pixel in the image file. The difference between say 11111111 and 11111110 is likely to be undetectable by the human eye. Therefore, the least significant bit can be used. LSB insertion works well with gray-scale images. It is possible to hide data in the least and second least significant bits and the human eye would still not be able to discern it. LSB method is used in Hide & Seek technique.

3.1.1 Hide & Seek technique

The simplest form of image steganography is the method known as Hide & Seek which replaces the LSBs of pixel values (also referred to as the spatial domain) with the bits from the message bit stream. This algorithm is perform mainly by two ways, first one is Sequentially & other one is randomly. Both the above type of techniques implemented for 8 bit & 24 bit.

```plaintext
for i = 1, ..., l(m)
do
    p ← LSB(ci)
    if p ≠ mi then
        ci ← mi
    end if
end for
```

Fig.3. Hide & seek encoding process

The encoding process (as shown in Fig 3) shows that the entire algorithm can be implemented by writing just a few lines of code [3]. The algorithm works by taking the first pixel of the image ci and obtaining its LSB value (as per line 2 of the Algorithm). This is typically achieved by calculating the modulus 2 of the pixel value. This will return 0 if the number is even and 1 if the number is odd, which effectively tells us the LSB value. We then compare this value with the message bit mi that we are trying to embed. If they are already the same, then we do nothing, but if they are different then we replace ci with mi. This process continues whilst there are still values in m that need to be encoded.

The decoding phase is even simpler. As the encoder replaced the LSBs of the pixel values in c in sequence, we already know the order that should be used to retrieve the data. Therefore all we need to do is calculate the modulus 2 of all the pixel values in the stegogramme s, and we are able to reconstruct m as m'. Fig 4 shows the algorithm of the hide & seek decoding process.

```plaintext
for i = 1, ..., l(s) do
    m'i ← LSB(si)
end for
```

Fig.4. Hide & seek decoding process

Note that this time we run the loop for l(s) instead of l(m). This is because the decoding process is completely separate from the encoding process and therefore has no means of knowing the l(m). If a key were used, it would probably reveal this information, but instead we simply retrieve the LSB value of every pixel. When we convert this to ASCII, the message will be readable up to the point that the message was encoded, and will then appear as garbage when we are reading the LSBs of the image data.

For the randomized approach the image data c is usually shuffled using a Pseudo Random Number Generator (PRNG). This generator will take the image data c and produce a shuffled version C according to a seed k that is specified by the encoder. Only the drawback of this algorithm is that it is easily detected by Visual Attacks.
As shown in fig 5, it is clear that original image contains organized LSB Plane where in the case of stego image it contains unorganized LSB Plane.

3.1.2 Transform Domain Techniques

This technique hides data in mathematical functions that are often used in algorithms. These techniques try to encode message bits in the transform domain coefficients of the image. Data embedding performed in the transform domain is widely used for robust data hiding. Similar techniques can also realize large-capacity embedding for steganography. Candidate transforms include discrete cosine transform (DCT), discrete wavelet transform (DWT), and discrete Fourier transform (DFT). By being embedded in the transform domain, the hidden data resides in more robust areas, spread across the entire image, and provides better resistance against signal processing. For example, we can perform a block DCT and, depending on payload and robustness requirements, choose one or more components in each block to form a new data group which is pseudo randomly scrambled and undergoes into a second layer transformation. Modification is then carried out on the double transform domain coefficients using various schemes. These techniques have high embedding and extraction complexity. Because of the robustness properties of transform domain embedding, these techniques are generally more applicable to aspect of data hiding.

**DCT based algorithm**

The classic and still most popular domain for image processing is that of the Discrete Cosine Transform. The DCT allows an image to be broken up into different frequency bands, making it much easier to embed watermarking information into the middle frequency bands of an image.

As shown in fig 6, the basic steps of DCT base algorithm are as follow:
- Each color plane is partitioned into 8x8 blocks
- Apply DCT to each block
- Values are quantized by dividing with preset quantization values (in a table)
- Values are then rounded to nearest integer
[1] JSteg: JSteg algorithm only differs from the Hide & Seek algorithm because it embeds the message data within the LSBs of the DCT coefficients of cover image, rather than its pixel values.

**Input:** Message, Cover image  
**Output:** Stego Image

Input $d$ to DCT domain $p$ for $i = 1, ... , l(m)$

   while $p = DC$ or $p = 0$ or $p = \pm 1$
   
   $p = next DCT coefficient from d$

   end while

LSB(pi) ← mi

end for

convert each 8x8 block back to spatial domain

Fig 7. JSteg encoding process

The JSteg encoding algorithm does not embed message data over any of the DC coefficients for every block. In addition to this, the algorithm also does not permit embedding on any AC coefficient equal to 0, 1 or -1 as shown in fig 7. The decoding process do the reverse procedure of encoding one which is shown in fig 8.

**Input:** Stego image  
**Output:** Message

Input $s$ to DCT domain $p$ for $i = 1, ... , l(s)$

   while $p = DC$ or $p = 0$ or $p = \pm 1$
   
   $p = next DCT coefficient from d$

   end while

LSB(pi) ← mi

end for

convert each 8x8 block back to spatial domain

Fig 8. JSteg decoding process

[2] Outguess: The first version of OutGuess, designed by Neils Provos [4], improved the JSteg algorithm by scattering the embedding locations over the entire image according to a PRNG on image $c$ derived using seed $k$. Outguess algorithm only differs from the JSteg algorithm by single approach which is it embeds the message data in randomise DCT coefficients.

**Input:** Message, Cover image  
**Output:** Stego Image

convert image $c$ to DCT domain $d$ in 8x8 blocks

for $i = 1, ... , l(m)$

   $p \leftarrow$ DCT($d$)

   while $p = DC$ or $p = 0$ or $p = \pm 1$

   $p \leftarrow$ next DCT coefficient from $d$

   end while

   LSB(pi) ← mi

end for

generate randomised block $C$ for data $d$

for $i = 1, ... , l(s)$

   $p \leftarrow$ DCT($C_i$)

   while $p = DC$ or $p = 0$ or $p = \pm 1$

   $p \leftarrow$ next DCT coefficient from $C$

   end while

   $mi \leftarrow$ LSB(pi)

end for

convert each 8x8 block back to spatial domain

Fig 9. OutGuess encoding process

The encoding process of the OutGuess algorithm [4] as shown in fig 9 is essentially a combination of both the randomised LSB algorithm and the JSteg algorithm. As Algorithm shows, the first step is to convert the image to the DCT domain, in exactly the same way as we saw for JSteg. Then, the coefficients are shuffled into a seemingly random order using a PRNG according to a seed. The message data is then embedding using the same technique as for JSteg before finally inversing the shuffle such that the coefficients are back in the correct positions. The image is then converted back to the spatial domain and thus the stegogramme $s$ is produced.

The decoding process for OutGuess is as we might expect. Firstly, the stegogramme $s$ is converted to the DCT domain, before being shuffled using the same $k$ that was used in the encoding process. We then retrieve the message data by extracting the LSBs from all the coefficients whose values are neither a DC value, nor a 0 or a 1. Drawback of these both the algorithm is they are easily detected by Histogram Attacks as shown in fig 10.

@IJAERD-2016, All rights Reserved
**Input:** Stego image  
**Output:** Message  
convert image s to DCT domain d in 8x8 blocks  
get randomised block S for data d  
for i = 1, ..., l(s) do  
    p ← DCT(Si)  
    while p = DC or p = 0 or p = ±1 do  
        p = next DCT coefficient from S  
    end while  
    mi ← LSB(pi)  
end for  

Fig 10. OutGuess decoding process  

**Histogram Attacks**  
As Andreas Westfeld discussed in [3], the JSteg algorithm introduces Pairs of Values (PoVs) as a result of sequential bit-flipping. It is possible to illustrate these PoVs by extracting all of an image’s DCT AC coefficients and tallying their frequencies of occurrence. If we split the values into bins we can narrow the results to a focused subsection and display the results by centering them across a specified range X. The result is referred to as a histogram.  

![Histogram of Original Image](image1)  
![Histogram of Stego Image](image2)  
![Histogram of Difference](image3)  

Fig. 10. Histogram results for JSteg Algorithm
IV. RESULT & CONCLUSION

It can be clearly observed that LSB is not very robust technique compared to other methods. The LSB plane of the stegogramme clearly shows the traces of embedding on the top-most region of the image. This proves that the attack function is capable of displaying the correct information for both a clean and distorted image but at the same time data hiding capacity is much more in LSB. JSteg is transform domain technique similar to LSB (which is spatial domain technique). OutGuess is modified version of JSteg. Visual Attack is not applicable on DCT base algorithm. After quantization, JSteg & OutGuess replaces the least significant bits (LSB) of the frequency coefficients by the secret message. The embedding mechanism skips all coefficients with the values 0 or ±1.

However, the statistical attack [5] on JSteg & OutGuess reliably discovers the existence of embedded messages, because JSteg & OutGuess replaces bits and, thus, it introduces a dependency between the value’s frequencies of occurrence that only differs in this bit position.

ACKNOWLEDGMENT

I would like to express my gratitude to all those who gave me the possibility to complete this paper. I would like to thank Electronic & Communication Department of silver-oak collage, Ahmedabad for giving me support and data for the paper. I am bound to my parents for their stimulating support and encouragement which helped me in all the time of writing this paper.

REFERENCES