SEMINAR REPORT

ON

"HIDE MESSAGE IN IMAGE "

BY

AJAY BELDAR

PRN No: 72145446B

ADITYA CHAVAN

PRN No: 72145490K

OM KARKIKAR

PRN No: 72145644J

SIDDHARTH SOLANKURE

PRN No: 72145892M

Under the guidance of

Prof. G. J. Navale

DEPARTMENT OF COMPUTER ENGINEERING

ALL INDIA SHRI SHIVAJI MEOMRIAL SOCIETY'S

INSTITUTE OF INFORMATION TECHNOLOGY

PUNE 411041

2022-2023

AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY
(I.O.I.T.)

ADDING VALUE TO ENGINEERING

An Autonomous Institute Affiliated to Savitribai Phule Pune University
Approved by AICTE, New Delhi and Recognised by Govt. of Maharashtra
Accredited by NAAC with "A+" Grade | NBA - 5 UG Programmes

Department of Computer Engineering

CERTIFICATE

This is to certify that

AJAY BELDAR PRN No.: 72145446B
ADITYA CHAVAN PRN No.: 72145490K
OM KARKIKAR PRN No.: 72145644J
SIDDHARTH SOLANKURE PRN No.: 72145892M

from **Third Year I^{st} Shift Computer Engineering** has successfully complited his/her seminar work titled

"HIDE MESSAGES IN IMAGE"

at All India Shri Shivaji Memorial Society's Institute of Information Technology, Pune in the partial fulfillment of the Bachelor's Degree in Computer Engineering

Prof. G. J. Navale
Internal Guide

Prof. G. J. Navale
Seminar Coordinator

Seal/Stamp of the college

Dr. S. N. Zaware
Head of the Department
Computer Engineering

Place: PUNE
Date:

ABSTRACT

It is a truth universally acknowledged that “a picture is worth a thousand words”. The emerge of digital media has taken this saying to a complete new level. By using steganography, one can hide not only 1000, but thousands of words even in an averagesized image. This article presents various types of techniques used by modern digital steganography, as well as the implementation of the least significant bit (LSB) method. The main objective is to develop an application that uses LSB insertion in order to encode data into a cover image. Both a serial and parallel version are proposed and an analysis of the performances is made using images ranging from 1.9 to 131 megapixels. The basic Steganography project is to encrypt and decrypt message. This project will help the user to improve secure communication and to store confidential information. In this Project, user can hide information in image using password and share data with end user without knowing any third party or intruder

ACKNOWLEDGEMENT

We take this opportunity to present my votes of thanks to all those guideposts who really acted as lightening pillars to enlighten my way throughout this project that has led to successfuland satisfactory completion of this study. We are really grateful to **prof. G. J. Navale** for providing me with an opportunity to undertake this project and providing me with all the facilities. We are highly thankful to sir for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation, and pains-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated. Lastly, We am thankful to all those, particularly the various friends, who have been instrumental in creating proper, healthy and conductive environment and including new and fresh innovative ideas for me during the project, without their help, it would have been extremely difficult for me to prepare the project in a time bound framework.

Prof. G. J. Navale
AISSMS IOIT, Pune.

INDEX

INTRODUCTION

Throughout time, confidentiality has always been important. Whether it is carved in stone, written on paper or sent over the Internet, correspondence between two persons is exposed to tampering or eavesdropping. Therefore, it is necessary to provide a mechanism that protects written information. One of the most common methods of securing transmitted data is cryptography. The purpose of cryptography is to disguise plaintext in such a way that it becomes unreadable. The resulted text, called cyphertext can then be safely transmitted to the destination. Knowing the algorithm used for encryption, only the receiver will be able to obtain the original message. Access to powerful computers does not only mean better encryption. It can equally be a tool used for breaking a cipher or decrypting a message. No matter how powerful the encryption algorithm is, encrypted data will always arouse suspicion. This is where steganography can help. Steganography can be defined as "the art and science of communicating in a way which hides the existence of communication". Although cryptography and steganography are often confounded, they are essentially different. Whilst the first one "scrambles a message so it cannot be understood", the other one "hides the message so it cannot be seen".

by using the two techniques together one can create a solid and powerful encryption system, better than each of the two components. With digital data as a mean of communication, messages can be hidden in the ones and zeros of text files, pictures, audio or video files. However, in order for the message to remain secret, only certain bits can be used. Hence, before the beginning of the information-hiding process, the stenographic system must identify the redundant bits - those bits that can be modified without altering the integrity of the file. After this step, the least significant bits from the cover medium can be replaced with data that has to be hidden.

PROBLEM STATEMENT

Making of a project for user to improve secure communication and data storing using steganography concepts. Using steganography, information can be hidden in carriers such as images, audio files, text files, videos and data transmissions. .

OBJECTIVES

Data hiding has formidable technical challenges. Perceptual or statistical holes to be filled with data in host signals are likely to be removed by means of lossy signal compression. Important factor to achieve successful data hiding technique is to find holes which are not convenient to be exploited by compression algorithms. The main challenge is filling data in this kind of holes in a way that is not easy for compression algorithms to exploit it. An enhanced challenge is filling the holes in a manner that remains invariant against signal transformation in big scale. Following counted features and restrictions are the criteria which a data embedding algorithm must meet :

• Quality of host signal should not be degraded objectionably and the perceptibility of embedded data must be kept minimal.

• The data must be embedded into whole body of the target media rather than wrapper or header. Therefore it would be kept intact in different formats.

• The data must be secure against intentional and intelligent removal attempts such as filtering, encoding, cropping, channel noise, lossy compressing, resampling, scanning and printing, digital to analog (D/A) conversion, analog to digital (A/D) conversion, and etc.

• Since data hiding goal is to keep the embedded data into host signal, embedded data asymmetrical encoding is desirable feature but not essential.

• To guaranty data integrity error correction coding is necessary. Degradation of embedded data at signal modification time is unavoidable.

PROJECT BUILDING

Images are the most popular files used for data hiding. For almost all image file formats there is at least one steganographic algorithm. All these algorithms can be grouped in three categories: spatial domain, frequency domain, and adaptive methods.

1) Image Spatial Domain Steganography: Spatial domain methods are based on encoding at the level of the LSBs. Least significant bit insertion is a common, simple approach to embedd information into a cover image. Depending on the size of the message and that of the image, one can use from the 1st to the 4th LSB. Using four bits however, will most likely produce visible artefacts. In its simplest form, LSB makes use of BMP images, since they use lossy compression.

2) Pixels' values are stored explicitly, easing the process of data embedding. Palette based images such as GIF files are also a very popular format, which supports up to 8 bits per pixel, thus allowing a single image to reference a palette of up to 256 distinct colors. One disadvantage of this method is that an altered pixel value could point to a totally different color than the initial one. If for BMP images, modification of the LSB means a new but very similar color to the original one, in this case the new value is a new palette index. A solution is the sorting of the lookup table so that similar colors have adjacent positions.

3) Image Frequency Domain Steganography: The most common file format used for this type of steganography is JPEG. It is a very popular file format, mainly because of its small sizes. JPEG is a frequently used method of lossy compression for digital photography. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. The first step of JPEG compression is converting the RGB representation to YUV. Y corresponds to luminance or brightness, while U and V stand for chrominance and color.

Taking advantage of the increased sensitivity of the human eye to changes in brightness, the color data is downsampled in order to reduce the total size of the file. The second step is the actual transformation of the image; usually a Discrete Cosine Transform (DCT) is used, but there is also possible to use Discrete Fourier Transform (DFT) . The next step is quantization.

Here, another property of the human eye is exploited, namely that the human eye is good at spotting small differences in brightness over a relatively large area, but not that good so as to distinguish between different strengths in the high frequency brightness . Consequently, magnitudes of the high-frequency components can be stored with a lower accuracy than the low-frequency components. And finally, the resulting data for all 8x8 blocks is compressed with a lossless variant of Huffman encoding. With a lossy compression scheme, JPEG files were at first considered useless for steganography.

Hiding information into the redundant bits would have been of no use, since these are left out in the compression process. However, properties of this algorithm have been exploited in order to develop steganographic techniques. One of the most important aspects of JPEG compression is the fact that it is divided into lossy and lossless stages. The DCT and quantization phases are lossy, but the final Huffman stage is not. Consequently, LSB insertion could be done exactly before the Huffman encoding is applied. Encoding data at this stage is not only safe – as there will be no loss –, but it is also difficult to detect since it is not in the visual domain.

4) Adaptive Steganography: The previous two sections described algorithms that work either with the image spatial domain or the frequency domain. There are however stegano graphic techniques that make use of both. These methods take statistical global features of an image before attempting to interact with its LSB or DCT coefficients [6]. Then, based on these statistics, the most suitable places for data hiding will be identified. In this way, the stego-message will be less likely to be discovered. One of the simplest and most used methods of steganalysis is noise detection – a good algorithm will avoid smooth areas of uniform colour. A complex colour scheme is also important, as variations will not be easy to detect

ALGORTIHM DESCRIPTION

Hiding information inside images is one of the most popularsteganographic techniques used nowadays. The method we choose for data encryption is the Least Significant Bit (LSB) approach using BMP images.

A. Bitmap Images

An important property of this type of images is the fact that all pixels are explicitly written in the file, a number for each component of each pixel: red, green, blue. A popular type of bitmap files is that in which each pixel is represented using24 bits - one byte for each color component. This gives 256 possibilities for each color plane, respectively 16, 777, 216 for one pixel. Altering the least significant bits will result in a color slightly different from the original one. What is most important to steganography is the fact that the human eye is unable to detect such differences.

B. Data Hiding Using LSB Insertion

Least significant bit insertion is one of the most commonly used methods of data hiding, due to its simplicity. The amountof data that can be hidden into an image depends on the sizeof the image and the number of least significant bits used for encryption. Having a 200x200 image and using only the least significant bit from each color component one gets 120, 000 bits that can be used in the process of data hiding. If we want to hide characters, the amount of data that can be encryptedis 15, 000 characters. This would represent a text that is twiceas long compared to the Declaration of Independence [1]. However, if we need more space we can always use a bigger image or more bits from each pixel, taking into consideration that the more bits we use, the lower the quality of the final image will be.

The process of data hiding is quite simple. Using one bit from each color component of a pixel, we get three bits per pixel; this means we need three pixels to hide a letter, i.e., one byte. Figure 1 contains an example of how the letter A (10000011) can be hidden into an image, using onlythe LSB for each pixel's color component. The

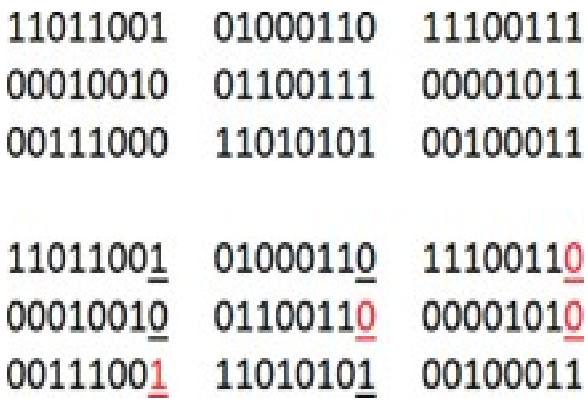

Figure 1: Example of LSB Insertion.

first three linesare the original values for the three pixels needed. The first column is red, the second blue and the third green. The lastbits from the next three lines hide the letter A – the underlined zeros and ones – the red color marks the bits that have changedin order to hide the data. As shown in Figure 1, only some of the bits' values change.On average, the LSB requires only half the bits in an image to change [2]. Moreover, even using both the least and the secondleast significant bit will not affect the image noticeably

Figure 2: Trurh is Iternal.

Ideally, the MS should be served by the closest node of the network. That would invoke the highest probability that an MS is served by the cell with the strongest signal and the cell with the highest RSS value was used, once the positioning request is received, the serving location mobile center (SLMC), based on the Cell-ID of the highest reported RSS value for each site, a specific positioning model is applied. The

model inputs (MIs) are RSS values of a particular set of BSs The RI list is then, in SLMC, matched with the MIs, which are specific for the area of a particular model [**?**]. The RI values that do not have corresponding MIs are discarded, whereas the missing MIs are entered with a threshold value of $-110dBm$. This threshold value is the minimal value of the RxLev parameter. On the other hand, the maximal value of the inputs was $-47dBm$, which is the maximal value of the RxLev parameter.

The major phases in providing the location estimate are given in figure. 5 where the implimented SVM model is trained and tested by the RI list for the real time data the location coordinates were predicted.

LSB IMAGE STEGANOGRAPHY

LSB Steganography is an image steganography technique in which messages are hidden inside an image by replacing each pixel's least significant bit with the bits of the message to be hidden.

To understand better, let's consider a digital image to be a 2D array of pixels. Each pixel contains values depending on its type and depth. We will consider the most widely used modes — **RGB(3x8-bit pixels, true-color) and RGBA(4x8-bit pixels, true-color with transparency mask).** These values range from 0–255, (8-bit values)

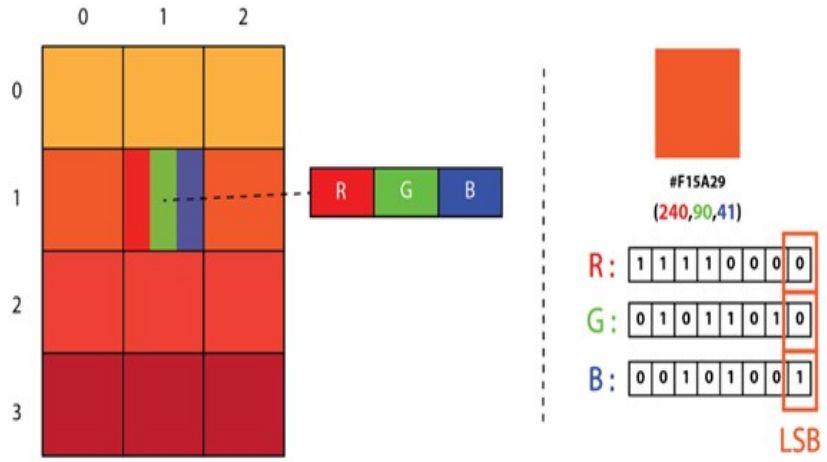

Figure 3: Representation of Image as a 2D Array of RGB Pixels

Representation of Image as a 2D Array of RGB Pixels We can convert the message into decimal values and then into binary, by using the ASCII Table. Then, we iterate over the pixel values one by one, after converting them to binary, we replace each least significant bit with that message bits in a sequence. To decode an encoded

image, we simply reverse the process. Collect and store the last bits of each pixel then split them into groups of 8 and convert it back to ASCII characters to get the hidden message.

PYTHON IMPLEMENTATION

Now, we will try to implement the above concept step-by-step with the help of Python Libraries — PIL and NumPy.

Step 1: Import all the required python libraries

```
import numpy as np
from PIL import Image
```

Step 2: Make the Encoder Function Firstly, we write the code to convert the source image into a NumPy array of pixels and store the size of the image. We check if the mode of the image is RGB or RGBA and consequently set the value of n. We also calculate the total number of pixels.

```
def Encode(src, message, dest):

    img = Image.open(src, 'r')
    width, height = img.size
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4    total_pixels = array.size//n
```

Secondly, we add a delimiter at the end of the secret message, so that when the program decodes, it knows when to stop. We convert this updated message to binary form and calculate the required pixels.

```
message += "$t3g0"
b_message = ''.join([format(ord(i), "08b") for i in message])
req_pixels = len(b_message)
```

Thirdly, we make a check if the total pixels available is sufficient for the secret message or not. If yes, we proceed to iterating the pixels one by one and modifying their least significant bits to the bits of the secret message until the complete message including the delimiter has been hidden.

```
if req_pixels > total_pixels:
    print("ERROR: Need larger file size")

else:
    index=0
    for p in range(total_pixels):
        for q in range(0, 3):
            if index < req_pixels:
                array[p][q] = int(bin(array[p][q])[2:9] +
b_message[index], 2)
                index += 1
```

15 / 26

Finally, we have the updated pixels array and we can use this to create and save it as the destination output image.

```
array=array.reshape(height, width, n)
enc_img = Image.fromarray(array.astype('uint8'), img.mode)
enc_img.save(dest)
print("Image Encoded Successfully")
```

With this, our encoder function is done and should look something like this -

```
def Encode(src, message, dest):

    img = Image.open(src, 'r')
    width, height = img.size
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
```

```
    elif img.mode == 'RGBA':
        n = 4    total_pixels = array.size//n

    message += "$t3g0"
    b_message = ''.join([format(ord(i), "08b") for i in message])
    req_pixels = len(b_message)

    if req_pixels > total_pixels:
        print("ERROR: Need larger file size")

    else:
        index=0
        for p in range(total_pixels):
            for q in range(0, 3):
                if index < req_pixels:
                    array[p][q] = int(bin(array[p][q])[2:9] +
b_message[index], 2)
                    index += 1

        array=array.reshape(height, width, n)
        enc_img = Image.fromarray(array.astype('uint8'), img.mode)
        enc_img.save(dest)
        print("Image Encoded Successfully")
```

Step 3: Make the Decoder Function Firstly, we repeat a similar procedure of saving the pixels of the source image as an array, figuring out the mode, and calculating the total pixels.

```
def Decode(src):

    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4

    total_pixels = array.size//n
```

Secondly, we need to extract the least significant bits from each of the pixels starting from the top-left of the image and store it in groups of 8. Next, we convert

these groups into ASCII characters to find the hidden message until we read the delimiter inserted previously completely.

```
hidden_bits = ""
for p in range(total_pixels):
    for q in range(0, 3):
        hidden_bits += (bin(array[p][q])[2:][-1])

hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits),
8)]

message = ""
for i in range(len(hidden_bits)):
    if message[-5:] == "$t3g0":
        break
    else:
        message += chr(int(hidden_bits[i], 2))
```

Finally, we do a check if the delimiter was found or not. If not, that means there was no hidden message in the image.

```
message += "$t3g0"
b_message = ''.join([format(ord(i), "08b") for i in message])
req_pixels = len(b_message)
```

With this, our decoder function is done and should look something like this —

```
def Decode(src):

    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))
```

```
    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4    total_pixels = array.size//n

    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(0, 3):
            hidden_bits += (bin(array[p][q])[2:][-1])

    hidden_bits = [hidden_bits[i:i+8] for i in range(0,
len(hidden_bits), 8)]

    message = ""
    for i in range(len(hidden_bits)):
        if message[-5:] == "$t3g0":
            break
        else:
            message += chr(int(hidden_bits[i], 2))
    if "$t3g0" in message:
        print("Hidden Message:", message[:-5])
    else:
        print("No Hidden Message Found")
```

Step 4: Make the Main Function For the main function, we ask the user which function they would like to perform — Encode or Decode. For Encode, we ask the user the following inputs — source image name with extension, secret message, and destination image name with extension. For Decode, we ask the user for the source image, that has a message hidden.

```
def Stego():
    print("--Welcome to $t3g0--")
    print("1: Encode")
    print("2: Decode")

    func = input()
```

```
    if func == '1':
        print("Enter Source Image Path")
        src = input()
        print("Enter Message to Hide")
        message = input()
        print("Enter Destination Image Path")
        dest = input()
        print("Encoding...")
        Encode(src, message, dest)

    elif func == '2':
        print("Enter Source Image Path")
        src = input()
        print("Decoding...")
        Decode(src)

    else:
        print("ERROR: Invalid option chosen")
```

Step 5: Put all the above functions together and our own LSB Image Steganography program is ready. Check out my GitHub repository for the complete code.

RESULTS

Figure 4: Normal Image

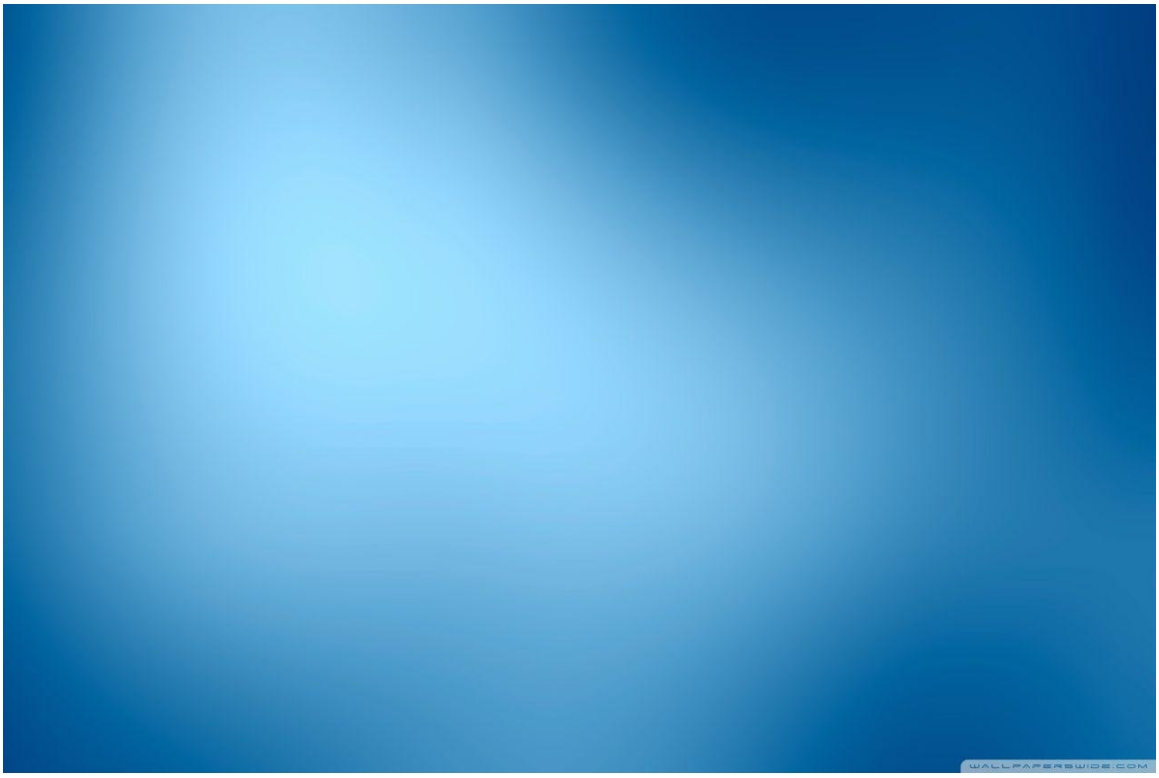

Figure 5: Stego Image

ADVANTAGES OF STEGANOGRAPHY

• The advantage of steganography is that messages do not send consideration to themselves. Clearly detectable encrypted message no matter how tough will stimulate suspision and may in themselves be compromising in countries where encryption is illegitimate.

• In steganography, cryptography secures the contents of a message, steganography can be said to secure both messages and connecting parties.

• This approach featured security, capacity, and robustness, the three needed element of steganography that creates it beneficial in hidden exchange of data through text files and creating secret communication.

• There are some important files carrying confidential data can be in the server in and encrypted form and No intruder can receive some beneficial information from the initial file during transmit.

• With the need of Steganography Corporation government and law enforcement agencies can connect privately.

• The major objective of steganography is to connect privately in a completely imperceptible aspect and to prevent drawing uncertainty to the transmission of a hidden information. It is not to maintain others from understanding the hidden data, but it is to maintain others from thinking that the data even exists. If a steganography approach generates someone to suspect the carrier medium, thus the method has unsuccessful.

• The advantage of steganography is that it can be generally used to secretly send messages without the case of the transmission being found. By using encryption, it can recognize the sender and the receiver.

• Steganography has a double component of protection such as first, the file itself is secret and second, the data in it is encoded

DISADVANTAGE OF STEGANOGRAPHY

• There are large number of information, huge file size, therefore someone can suspect about it.

• If this approach is gone in the wrong hands such as hackers, terrorist, criminals then this can be very much critical.

• Steganography is not without its disadvantages. However, these can be rectified and once it is performed and it can strengthen the element of steganography.

• Most data hiding approach take advantage of human perceptual deficiency, but they have

deficiency of their own. However, these can be independently rectified.

• The major disadvantage of steganography is that, unlike cryptography, it needed a lot of overhead to hide associatively few bits of information. Because the steganographic system is found, it is rendered useless. However, it fares no worse than cryptography and is still the preferred medium

APPLICATIONS

1. Confidential communication and secret data storing

The "secrecy" of the embedded data is essential in this area. Historically, steganography have been addressed in this area. Steganography provides us with: (A) Potential capability to hide the existence of confidential data (B) Hardness of detecting the hidden (i.e., embedded) data (C) Enhancing the secrecy of the encrypted data In practice, when you use some steganography, you must first select a vessel data according to the size of the embedding data. The vessel should be innocuous. Then, you embed the confidential data by using an embedding program (which is one component of the steganography software) together with some key. When extracting, you (or your party) use an extracting program (another component) to restore the embedded data by the same key ("common key" in terms of cryptography). In this case you need a "key negotiation" with your party before you start confidential communication.

2. Protection of data alteration

We take advantage of the fragility of the embedded data in this application area.

3. Access control system for digital contents distribution

In this area embedded data is "hidden", but is "explained" to publicize the content. Today, digital contents are getting more and more commonly distributed over Internet than before. For example, music companies release new albums on their Webpage in a free or charged manner. However, in this case, all the contents are equally distributed to the people who can make access to the page. So, an ordinary Web distribution scheme is not suited for a "case-by-case" and "selective" distribution. Of course it is always possible to attach digital contents to e-mail messages and send them to the customers. But it will takes a lot of cost in time and labor

If you have some valuable content, which you think it is distributable if someone really needs it, and if it is possible to upload that content on Internet in some covert

manner. And if you can issue a special "access key" to extract the content selectively, you will be very happy about it. A steganographic scheme can help realize this type of system.

CONCLUSION

The goal of this project was to implement an application that uses the LSB steganography method in order to hide and recover data. Because communication involves a sender and a receiver, there are two ways in which the application can run: as an encoder or as a decoder. For the encoding part the message is hidden into the least significant bits of a bmp image, thus resulting the stego-image. This image is then given to the decoder to extract the data that has been hidden.

The parallelization of the serial version of the algorithm was done using boss-worker threading model. Both data hiding and data recovery can be done in parallel. Neither process depends on the number of threads, and thus any combination of threads can be used. The only variables that must be constant for both processes are the number of bits used from each color component and the size of the chunks in which the input image is divided in order to parallel encode and decode the data. To test the limits and performance of the algorithms we used three different images of: 1.9 MP, 24.3 MP and 131.7 MP respectively. The messages that were hidden into the input images were either books or other images. 165 The best performance for the data hiding part was obtained for test cases 2 and 3.

In conclusion, the main objectives of this project were accomplished. The result was an application that can be used to encode into and decode data from images. The serial execution time was reduced to a quarter. The main reason speedup did not reach higher values than 4 is the increase in I/O operations. Data processing is not very time consuming, as it is reduced to bit operations, which are very fast.

REFERENCES

[1] B. Granthan, "Bitmap steganography: An intoduction," April 1997.

[2] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," April 2007.

[3] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," January 2004.

[4] T. Morkel, J. H. P. Eloff, and M. S. Olivier, "An overview of image steganography," in Proceedings of the Fifth Annual Information Security South Africa Conference ISSA2005, L. L. Hein S Venter, Jan H P Eloffand M. M. Eloff, Eds., Sandton, South Africa, June/July 2005, published electronically.

[5] J. Krenn, "Steganography and steganalysis," January 2004.

[6] A. Cheddad, "Steganoflage: A new image steganography algorithm," Ph.D. dissertation, University of Ulster, September 2009.

[7] J. P. F. Bradford Nichols, Dick Buttlar, Pthreads Programming: A POSIXStandard for Better Multiprocessing. O'Reilly Media, 1996.

[8] D. R. Butenhof, Programming with POSIX Threads. Addison-WesleyProfessional, 1997.

[9] M. D. Scott J. Norton, Thread Time: The MultiThreaded ProgrammingGuide. Prentice Hall PTR, 1996