steganography

An-Najah National University

By

Jawad T. Abdulrahman, Jihad M. Abdo and Muhammad H. Bani-Shamseh

A thesis submitted in partial fulfillment of the requirements for the degree of

B.S. Degree

2010

Abstract

Hiding Information inside a digital image

Department of Electrical Engineering

In this research we propose a system of communication between two parties in a secure channel; the message to be transmitted, which is a text, is first compressed using Huffman coding technique, and then a cover, gray scale image *C* is used to hide the message inside, the resulting image is *S* (the stego image). At the receiver, first the compressed message is extracted from the stego image, and then it is de-compressed to recover the original message. Our aim is to hide maximum amount of information, with least error in the stego image (BER) so that no other party knows that there is hidden information.

# Table of contents

Table of contents 1

Table of figures 3

1 Introduction 4

1.1.1 Difference between Steganography and Cryptography………..4

1.1.2 Ancient Steganography………………………………………..5

1.1.3 Digital Steganography…………………………………………5

2 Chapter 2: Information Theory and Source Coding 8

2.1 Introduction 9

2.2 What is the importance of information theory? 10

2.3 Introduction to information theory and the entropy 11

2.3.1 Source coding………………………………………………..11

2.4 Entropy of information 12

2.4.1.1 Properties of entropy……………………………………12

2.4.2 Source coding theorem (Shannon’s first theorem)……………14

2.5 Data compression 16

2.5.1 Statistical encoding………………………………………….17

2.5.1.1 Runlength encoding…………………………………….17

2.5.1.2 Shannon-Fano algorithm……………………………….18

2.5.1.3 Huffman coding………………………………………..22

3 chapter 3: Steganography definition and history 26

3.1 Definition of Steganography 27

3.2 Steganography history 28

3.3 Alternatives 29

4 chapter 4: System model 30

4.1 Text model 31

4.2 Image model 32

4.3 Mathematical modeling of the system 33

4.4 Information channels 36

5 Chapter 5: System Performance 38

5.1 Numerical Results 40

5.2 Software Description 45

Bibliography 5

# Table of figures

Figure 1: Plot of binary entropy function versus probability 13

Figure 2: Complete block diagram of the system 30

Figure 3: ***t*** vector generation 31

Figure 4: vector ***t*** sub-divided into *n* sub-blocks each contains *r* bits. 33

Figure 5: ***t’*** vector after adding the control bits to ***t*** 35

Figure 6: the binary symmetric channel 36

Figure 7: Text length vs. entropy 40

Figure 8: The relation between Header length and error 42

Figure 9: Text size vs. Error 43

Figure 10: Header Length vs. Error for different Text sizes 44

Figure 11: Software Main screen 45

# Introduction

During the last few years, communications technology has noticeably improved, in terms of modulation-demodulation, equipments, used frequencies, and the most important evolution was the transformation from analog to digital communication.

Digital communication has added new features that were not available before, most importantly, the ability to store data securely, for very long period of time, which was quite expensive and space consuming when analog communication was used.

In digital communications, data is converted into a sequence of bits (ones and zeros) using special techniques depend on the type of data, for example, the ASCII format is internationally used to store text files, where each character is represented by eight bits.

Digital images are composed of pixels, the pixel represents the intensity level at a specific location, each pixel is represented by 24 bits in RGB format, or 8 bits in gray scale images. In the same way, video files consist of sequence of images. Ultimately, any type of data can be represented by a sequence of 0’s and 1’s which can be stored on memory.

### Difference between Steganography and Cryptography

Steganography is the art of hiding information without the ability for an unauthorized person to detect the presence of information, while in cryptography it is obvious to the public that there is hidden information but nobody can understand this message apart from the authorized person.

The advantage of steganography over cryptography is that hidden messages do not attract attention in steganography, while encrypted data will arouse suspicion and as a result, someone will always try to break this encrypted message.

### Ancient Steganography

The word steganography is of Greek origin, from the two words *steganos* meaning “covered or protected”, and *graphein* meaning “to write” *,* so the word means “concealed writing”.

The first recorded uses of steganography can be traced back to 440 BC when [Herodotus](http://en.wikipedia.org/wiki/Herodotus) mentions two examples of steganography in [The Histories of Herodotus](http://en.wikipedia.org/wiki/The_Histories_of_Herodotus). [Demaratus](http://en.wikipedia.org/wiki/Demaratus) sent a warning about a forthcoming attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface. [Wax tablets](http://en.wikipedia.org/wiki/Wax_tablet) were in common use then as reusable writing surfaces, sometimes used for [shorthand](http://en.wikipedia.org/wiki/Stenography). Another ancient example is that of [Histiaeus](http://en.wikipedia.org/wiki/Histiaeus), who shaved the head of his most trusted slave and tattooed a message on it. After his hair had grown the message was hidden. The purpose was to instigate a revolt against the [Persians](http://en.wikipedia.org/wiki/Persian_Empire).

### Digital Steganography

In digital steganography, the message is converted into binary message, and is hidden inside a cover object, there are many types of digital steganography; audio signal can be hidden inside an image, or inside a video, text files can be hidden inside digital images, text files inside an audio file, and many others.

The hiding process utilities the sensitivity of human systems, for example, each pixel in gray scale images is represented by 8 bits, which means that there are 2^8=256 different color levels, the human visual system normally cannot distinguish between two subsequent colors, this “defect” can be exploited to hide data in the least significant bit of each pixel. Likely, in audio files, part of the less important data can be replaced by data to be hidden without the ability to sense the noise generated by the hiding process.

In our research we are interested in hiding a text file inside a gray scale image, gray scale images are matrices of pixel, where each pixel represents the intensity value at a specific location[[1]](#footnote-1), each pixel is represented by eight bits, which means that we have 256 different levels; from 00000000 to 11111111.

The first step is to compress the text file, compression is needed in order to reduce the amount of data to be hidden inside the image, which increases the maximum capacity of the image, and reduces the error. Huffman coding is applied to the text file[[2]](#footnote-2), where each character is assigned a code word; these code words are known between the transmitter and the receiver to enable them from compressing, and decompressing the text file. The length of the code word depends on the nature of the language itself; for example, in English language, the character ‘e’ is used more than, say, the character ‘j’.

After compressing the text file, and now we have a sequence of bits represent it, we call it the vector *t* .

The next step is to extract the least significant bit from each pixel and arranging them into a new vector, called the *q* vector. Now we have to hide the *t* vector inside the *q* vector.

The procedure is to divide both the *t* and *q* vectors into packets of data, we believe that this procedure will reduce the error in the image file as we will see later; each packet of data in the *t* vector replaces a packet in the *q* vector where least error occurs.

Dividing the *t* vector into smaller packets requires a header at the beginning of each packet so that the receiver will be able to rearrange them in the correct order, at this point we have to make a tradeoff between the number of packets and the length of the header; increasing the number of packets, reduces the error, and on the other hand, increases the number of bits in the header needed.

Header length ranges from 0 to 9 is used in the software, which enables dividing the *t* vector from 1 packet to 2^9=512 packets, the header which results in the least error is used.

In chapter 2 we discuss data compression, lossless and loosy compression, Huffman coding and Shannon-Fano algorithm, and Entropy of information. In chapter 3 we give an overview of the history of steganography, and types of digital steganography. In chapter 4 we discuss the system model. And in chapter 5 we discuss the system performance.

# Chapter 2: Information Theory and Source Coding

## Introduction

In this chapter we explain information theory and some coding techniques that we will use in text compression.

In order to choose a good code for data compression, the code must achieve maximum efficiency and minimum number of bits used to represent a certain character. There are two types of codes: fixed length (e.g. ASCII), and variable length codes. In fixed length codes the number of bits used to represent each symbol is fixed, and the receiver groups a number of bits and extract the equivalent character, while in variable length codes, the number of bits used to represent each symbol differs from one character to another, and it depends on the probability of this character, those with lower probabilities are assigned higher number of bits, while those with higher probability are assigned lower number of bits.

Using fixed length codes is much easier than using variable length codes; because the code length is constant, and because when using variable length codes it is important to ensure that the code words are prefix free[[3]](#footnote-3) , however, by using variable length codes the average code length is reduced, and this increases the efficiency of the code.

## What is the importance of information theory?

Information theory is the study of how the laws of probability and mathematics in general put some limits on the design of information transmission systems. It provides guidance for those who are searching for new and more advanced communication systems, information theory attempts to give fundamental limits on transmission, compression and extraction from environment of information; it shows how devices are designed to approach these limits.

In 1948, Claude Shannon showed that nearly error-free communication is possible over a noisy channel, provided an appropriate preprocessor called the encoder and postprocessor called the decoder are located at each end of the communication link. Shannon did not tell us how to design the best encoder and decoder and how complex they must be.

Before an event occurs there is an amount of uncertainty, when an event occurs there is an amount of surprise, and after the occurrence of an event there is an amount of information achieved.

These sentences can give a glimpse about information theory logic; as the probability of an event increases, the amount of surprise when this event occurs decreases, and consequently, the amount of information obtained after the occurrence of this event decrease.

Communications is one of the major issues in information theory applications because it gives guidance for the development of modern communication methods and it shows us how much room for improvements remain.

## Introduction to information theory and the entropy

Entropy, mutual information and discrimination are the three functions used to measure information; the significance of an alphabet's entropy tells us how we can represent it with a sequence of bits.

Entropy is the simplest of these functions, it can be used to measure the prior uncertainty in the outcome of a random experiment or equivalently, to measure the information obtained when the outcome observed.Here, we will consider the original picture as the transmitter, and the doped picture (after hiding the text inside it) as the receiver, and we want to study the effect of this process on the quality of the picture.

### Source coding

Source coding includes different techniques to compress data, our main concern in using a specific technique is to reduce the number of bits transmitted without losing any data (remember that our message is a text, any distortion in data would be obvious in the decompression), this would increase the efficiency and reliability of the communication system, data compression at the transmitter is an important stage as it helps reducing the consumption of expensive resources.

There are many source coding techniques such as: Shannon – Fano coding, Huffman coding, and Lempel –ziv, each technique gives different code words for the characters, but the main aim is the same: to reduce the amount of data for transmission.

## Entropy of information

The entropy is a function that measures the average amount of information associated with a random variable; imagine a source sending symbols every unit of time, the entropy of this source is defined as the average amount of information per symbol generated by this source.

When a source sends symbols, and these symbols are not equally likely (i.e some symbols are transmitted more than others), then, the amount of information gained by the receiver when receiving one of these symbols depends on the probability of receiving it; if a message appears less frequently than another, then the amount of information gained when receiving this message is more than the amount of information gained when receiving another less frequent message. The following relation describes the relation between the probability of receiving a message, and the amount of information gained after receiving this message:

The amount of information:

$$I=log\_{2}\left(\frac{1}{P\_{k}}\right)$$

Where *Pk*is the probability of receiving a message *k*.

Let *S* be a source of information that transmits *n* messages, $S=\left\{m\_{1},m\_{2},m\_{3},…..,m\_{n}\right\}$, and let *Pi* be the probability distribution for this source, where *i=*1,2,…..,*n*. Then, the entropy of this source (the average amount of information generated by this source per symbol) is[[4]](#footnote-4):

$$H\left(S\right)= \sum\_{S}^{}P\left(m\_{i}\right) log\left(\frac{1}{P(m\_{i})}\right) $$

#### Properties of entropy

1. The entropy is zero if the probability of a message is zero or one (i.e. there is no information obtained if the event is either certain or impossible).
2. The maximum value of the entropy is:

$H\_{max}=log\_{2}\left(M\right)$

Where M is the number of messages, this maximum value of the entropy is only possible if all the messages are equally likely.

A special type of information source is the binary source, this source as its name indicates generates only two symbols $\left\{0,1\right\} $, if we let *P*(1)=*p*, then, *P*(0)= 1-*p*, then the average amount of information obtained per symbol is:

$$H=p log\_{2}\left(\frac{1}{p}\right)+\left(1-p\right) log\_{2}\left(1-p\right)$$

A plot of *H* as a function of *p* is shown in figure 2.1.



Figure : Plot of binary entropy function versus probability

In Figure 1 we can see that if the probability of one of the two symbols is higher than the other, then the average amount of information obtained is decreased, note that the maximum amount of the entropy function is when the two symbols are equally likely, in this case the average amount of information carried by each symbol is 1 bit, and the entropy is zero if the output of the source is either certain (*p*=1), or impossible (*p*=0).

### Source coding theorem (Shannon’s first theorem)

If messages with different probabilities are assigned the same code word length, then the actual information rate is less than the maximum achievable rate, to solve this problem, source encoders are designed to use the statistical properties of the symbols, for example, messages with higher probabilities are assigned lower number of bits, while those with lower probabilities are assigned higher number of bits.

To find the average code word length of the messages is:

$$\overbar{N}=\sum\_{i=0}^{L-1}P\_{k}n\_{k}$$

Where:

*Pk* is the probability of the kth message, and *nk* is the number of bit assigned to it, and L is the number of messages.

Shannon’s first theorem describes the relation between the average code word length, and the entropy of the messages, it states the following:

“Given a discrete memory less source of entropy H, the average code word length $\overbar{N}$ for any distortionless source encoding is bounded as:

$$\overbar{N}\geq H$$

The entropy H set a limit on the average code word length, this limit is that the average number of bits per message cannot be made less than the entropy H, and they may be equal only if the messages are equally likely, in this case we achieve a 100% efficiency and each bit in the code word carries 1 bit of information.

The efficiency of the source encoder is described as:

$$ɳ=\frac{N\_{min}}{\overbar{N}}$$

Where *N*min  is the minimum value of the average code word length that can be achieved, and as we know, this value is the entropy H, so the above equation becomes:

$$ɳ=\frac{H}{\overbar{N}}$$

Our aim from source coding is to make the average code word length nearly equal to the entropy, so that maximum efficiency can be achieved.

## Data compression

Data compression is a very important part in today’s modern world of digital communication, without data compression it would be very difficult to exchange information between two points for the following reasons:

1. Huge amount of data need to be stored, so we need an enormous storage space, as an example, suppose that we have a picture with 256X256 pixels, with RGB format, so each pixel has 24 bits, the size of this picture would be: 256X256X24= 1572864 bits=1.5 Mbytes !

This is a very large amount of data for one picture to store.

1. This huge data increases the complexity of transmission; we need a high capacity channel to handle this high bit rate, and according to Shannon’s theorem on channel capacity, it is possible to transmit information with small probability of error if the information rate in less than the channel capacity C, where:

$$C=B log⁡(1+\frac{S}{N})$$

Where, B is the channel bandwidth, and it varies significantly from one channel to another, the free space has a bandwidth different from that of a twisted pair, coaxial cable, or fiber optics. And $\frac{S}{N}$ is the signal to noise ratio.

There are basically two types of compression: lossless, where the decompressed message is exactly the same as the original message, and it is used for text compression. The other type is lossy compression, where some of the data is lost, and it is best used with images, audio, and video compression.

But, on the other hand, there are some drawbacks of data compression; due to compression, some of data might be lost, the amount of lost data depends on the compression technique, and the data to be compressed, for example, if the lost data belongs to audio, video, or image, then we may overcome these losses because we exploit the sensitivity of human response to the colors and the audio, but on the other hand, text compression must be lossless, because if any data is lost, it affects directly the recovered message.

### Statistical encoding

Statistical encoding, or entropy encoding, is a type of lossless compression technique that exploits the probability of occurrence of the set of symbols at the transmitter to assign different code words for these symbols, as we know, if the messages at the transmitter are not equally likely, the entropy H is reduced, and this in turn reduces the information rate.

This problem is solved by coding the messages with different number of bits; messages with higher probability are coded with lower number of bits, and those with lower probability are coded with higher number of bits. In this method the average code word length is nearly equal to the entropy, this increases the efficiency of the code in the sense that each bit in the code word nearly carries one bit of information.

There are many types of statistical encoding: runlength encoding, Lempel – ziv coding, Huffman coding and Shannon-Fano coding, these coding techniques are examples of lossless data compression which is best used for text compression where data losses is not allowed.

Now we are going to discuss these compression techniques and compare between them. However, we will put our interest on Huffman coding, as it is the method we are going to use in our project.

#### Runlength encoding

This technique is used for data with high redundancy, like the data generated by scanning the documents, fax machines, typewriters, etc. these sources normally produce data with long strings of 1’s and 0’s, consider the following string of data:

11110000011111100000111111000

Using runlenght encoding, this string can be encoded as follows:

1,4;0,5;1,6;0,5;1,6;0,3

Each section consists of two numbers separated by a comma; the first represents the repeated digit, and the second is the number of times that this digit is repeated, and the semicolon is to separate between two successive sections.

#### Shannon-Fano algorithm

This coding technique depends upon the probabilities of the characters at the transmitter; symbols with higher probabilities are assigned shorter codes, while those with lower probability are assigned longer codes. to explain this coding technique, let’s assume the following example.

Example: consider a source contains eight messages with the following probabilities:

 Table 2-1 probability distribution of eight messages

|  |  |
| --- | --- |
| Message  | Probability |
| M1 | 0.1875 |
| M2 | 0.125 |
| M3 | 0.0625 |
| M4 | 0.1875 |
| M5 | 0.125 |
| M6 | 0.0625 |
| M7 | 0.0625 |
| M8 | 0.1875 |

We want to find the code word for each message, and calculate the efficiency of the code.

The first step is to arrange the messages in descending order, and then we divide them into two equally sections according to their probabilities, messages in the upper portion are assigned code ‘0’, and those in the lower portion are assigned code ‘1’. Then, each section is further subdivided into two subsections and codes are assigned in the same manner until we have a code for each message.

Table 2-2 Shannon-Fano algorithm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| M1 | 0.1875 | 0 | 0 |  |  |  |
| M4 | 0.1875 | 0 | 1 | 0 |  |  |
| M8 | 0.1875 | 0 | 1 | 1 |  |  |
| M2 | 0.125 | 1 | 0 | 0 |  |  |
| M5 | 0.125 | 1 | 0 | 1 |  |  |
| M3 | 0.0625 | 1 | 1 | 0 |  |  |
| M6 | 0.0625 | 1 | 1 | 1 | 0 |  |
| M7 | 0.0625 | 1 | 1 | 1 | 1 |  |

From Table 2-2, code words are:

Table 2-3 code words for Shannon-Fano alogrithm

|  |  |
| --- | --- |
| Message | Code word |
| M1 | 00 |
| M2 | 100 |
| M3 | 110 |
| M4 | 010 |
| M5 | 101 |
| M6 | 1110 |
| M7 | 1111 |
| M8 | 011 |

To find the entropy of the source:

$$H\left(S\right)= \sum\_{S}^{}P\left(m\_{i}\right) log\left(\frac{1}{P(m\_{i})}\right)$$

Putting the values from Table 2-3 into the equations, the entropy of the source *H*=2.858 bit/message.

To find the efficiency of the code, we need first to calculate the average code word length:

$$\overbar{N}=\sum\_{i=0}^{L-1}P\_{k}n\_{k}$$

This gives the result $\overbar{N}=2.9374 bits\message$

Note that the average code length is higher than the entropy which agrees with Shannon’s first theorem. Hence, the minimum code word average length is H, and it is possible if and only if all the messages are equally likely.

Now, to calculate the efficiency of the code,

$$ɳ=\frac{H}{\overbar{N}}$$

$$ɳ=\frac{2.858}{2.9374}=97.3\%$$

The redundancy in the code is a measure of the redundancy in the bits in the encoded message sequence.

$$ɣ=1-ɳ$$

$$ɣ=1-0.973=2.7\%$$

Note that using Shannon-Fano coding technique, the average code length is approximately equals to the entropy of the source, this means that each bit in the message carries 0.973 bit of information, if we ignored the fact that the message are not equiprobable, then we would need 3 bits to encode these messages, and in this case, each bit in the code carries 0.952 bit of information.

#### Huffman coding

Huffman coding is a statistical coding technique, it aims to make the average code word length nearly equals to the entropy, to explain this coding technique, let us consider the following example.

Table 2-4 shows four messages emitted by a source and their probability distributions, we want to encode these messages using Huffman coding, and Shannon-Fano algorithm, and compare between the two techniques.

Table ‑4: Probability distributions of four massages

|  |  |
| --- | --- |
| Message | Probability |
| M1 | 0.4 |
| M2 | 0.3 |
| M3 | 0.15 |
| M4 | 0.15 |

1. Huffman coding:

We arrange the messages in descending order according to their probabilities, then, we add the probabilities of the two lowest messages and give one of them code ‘0’ and the other code ‘1’, and rearrange the new probabilities, and complete in the same process until we end up with two probabilities their sum is 1.

 0

M1 0.4 0.4 0.6

 0

M2 0.3 0.3

 0.4

 0 1

M3 0.15 0.3

 1

M4 0.15

 1

To obtain the code words, we to trace the path of a message through the arrows to the end, and get a sequence of bits, to obtain the code word for that message, we read the sequence for the LSB to the MSB. Table 2-5 shows the code words for the messages.

Table 2-5 Code words for Huffman coding

|  |  |
| --- | --- |
| Message | Code word |
| M1 | 1 |
| M2 | 01 |
| M3 | 000 |
| M4 | 001 |

The entropy of the source H=1.871 bits/message.

The average code word length $\overbar{N}$=1.9 bits\message

And the efficiency of the code ɳ=98.47%.

1. Using Shannon-Fano algorithm

Table 2-6 Shannon-Fano coding

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M1 | 0.4 | 0 |  |  |
| M2 | 0.3 | 1 | 0 |  |
| M3 | 0.15 | 1 | 1 | 0 |
| M4 | 0.15 | 1 | 1 | 1 |

The entropy H=1.871 bits/message.

The average code word length $\overbar{N}$=1.9 bits\message.

And the efficiency ɳ=98.47%

Note that the efficiency for Huffman code and Shannon-Fano algorithm is the same, and they have the same code word length. However, the code words are different.

Using Huffman or Shannon-Fano coding algorithm gives uniquely decodable- prefix free codes, this means that no code word is a prefix of any other code word. For example, if message *m*1 has code word 001, and message *m*2 has code word 010, then this code is prefix free.

*m1* : 0 0 1 0 code word

 Prefix of *m1*

But, if the code word for *m1* is 001, and that for *m2* is 0010, then this code is not prefix free, as the prefix of the code word of *m2* is a code word for *m*1.

In Huffman coding, the combined probabilities of two messages can be located as high or as low as possible, in either case, the average code word length is the same, however, if we place them as high as possible, the code words of the individual messages will have less variance from the average code word length.

The variance:

$$ơ=\sum\_{}^{}P\_{i} \left(N\_{i}-\overbar{N}\right)^{2}$$

# chapter 3: Steganography definition and history

## Definition of Steganography

Hiding information has become an important research area in recent time. There are many techniques for hiding information; one of these techniques is known as steganography . The word steganography comes from the Greek name “stegano” which means hidden or secret, and “graphy” which means writing or drawing and literally the word means hidden writing. Unlike data encryption, in steganography the information (secret massage) is hidden inside a cover object and no one must detect the message because no one knows that these information is hidden except the person to whom we send the massage, so the goal is to prevent the detection of a secret message by hiding the messages inside other harmless messages (cover massage) in a way that does not allow any enemy to even detect that there is a second secret message present. While in cryptography, information is encrypted, we know that there is a hidden message, but we cannot understand it.

There are many forms of steganography as we will see in the next section; such as hiding a text inside an image, hiding a voice inside an image, and so many other forms.

 When hiding a text inside an image, the least significant bits (LSB) method is usually used. In this method, one can take the binary representation of the hidden data and overwrite the LSB of each byte within the cover image. We will use an image to hide the resultant compressed text bits in the LSB of that image. The image is divided into small elements called pixels, each pixel represents the intensity of a certain location. In gray scale images each pixel is represented by 8 binary bits (one byte), that allows 255 different intensity levels their binary representation ranges from 00000000 to 11111111, the message bits is embedded in the most right bit which is called the LSB and this process has the least effect on the image, which is almost undetectable by the human visual system.

## Steganography history

Since man first started communicating over written messages, the need for security has been in a high demand. In the past, messages could easily be detected by a third party other than the one the message in sent to. This all changed during the time of the Greeks, around 500 B.C., when Demaratus first used the technique of Steganography.

Demaratus was a Greek citizen who lived in Persia because he was banned from Greece. While in Persia he witnessed Xerxes, the leader of the Persians, build one of the greatest naval fleets the world has ever known. Xerxes was going to use this fleet to attack Greece in a surprise attack. Demaratus still felt a love for his homeland and so decided he should warn Greece about the secret attack. He knew it would be hard to send the message over to Greece without it being intercepted. This is when he came up with the idea of using a wax tablet to hide his message. Demaratus knew that blank wax tablets could be sent to Greece without anyone knowing. To hide his message, he scraped all the wax away from the tablet leaving only the wood from underneath. He then scraped his message into the wood and when he finished, recovered the wood with the wax. The wax covered his message and it looked like it was just a blank wax tablet. Demaratus’s message was hidden and so he sent this to Greece. The hidden message was never discovered by the Persians and successfully made it to Greece. Because of this message, Greece was able to defeat the invading Persian force.

That was the first known case of the use of Steganography and since that time the complexity of Steganography has exploded.

## Alternatives

In chapter one we discussed Data compression and some important concepts about information theory which is an important part in steganography.

Steganography works by replacing bits of useless or unimportant data in regular computer files (such as graphics, sound tracks, text files, HTML, or even floppy disks) with bits carrying information.

The information to be hidden in the cover data is known as the “embedded” data, the “stego” data is the data containing both the cover object and the embedded information inside. Logically, the process of hiding data inside the cover object is called embedding.

In our project, we hide a text inside an image which is used as the cover object. However, this is not the only form of steganography; other forms uses a sound track as the cover object, and hide a text or image inside it, other forms of steganography even hide an image inside another image.

All forms of steganography share the same idea, which is exploiting the sensitivity of human sense to detect the variation in the cover object, by removing the redundant data, and replacing it by useful information.

# chapter 4: System model

In this chapter we introduce the behavior of our system as a transmitter, receiver, and channel. We represent the operation of hiding a text inside an image as a communication channel (Binary Symmetric Channel) in which the cover image C before the hiding process is transmitted through a channel, the stego image S (the image after hiding the text inside it) is received at the receiver side, and the distortion happened in the image due to the hiding process is a noise added by the channel.

As we have discussed in chapter 3, we exploit the low sensitivity of the human visual system to hide a text inside a gray scale image. More precisely, we use the Least Significant Bit (LSB) of the pixels to hide the text, and this process might result in flipping some of these bits, and in turn causes degradation in the image, our aim is to maintain this distortion as low as possible.

Figure 2 shows the block diagram for our system.

 Channel

Data compression

Stegano-graphy

Stega analysis

De-compression

Message Message

*anЄ{0,1}*

Figure : Complete block diagram of the system

## Text model

The first step is to convert the text into a sequence of binary bits {0’s and 1’s}, as we have explained in chapter two, the ASCII system is not an appropriate choice for representing the English characters because it assigns constant code lengths for all characters (one byte), regardless of their probabilities distribution, while in fact, English characters are not equiprobable; for example, the letter “e” has appears more frequently than, say the letter “q”. Therefore, characters with higher probabilities are assigned code words with different length than those with lower probabilities.

Therefore, by applying Huffman coding algorithm on the characters, and by taking into consideration the probability distribution of the characters, which are in our case the English characters, and some special additional characters, which depends on the language itself, we convert the message to be hidden inside the cover image into a vector ***t***.

Figure : ***t*** vector generation

My e-mail password is: str123mnb

10011010111001001001110100101101000101001

 Message

 ***t***

The ***t*** vector is a sequence of o’s and 1’s, we denote *p*(0)=*p*, which is the probability of zero, and *p*(1)=1-*p*, which is the probability of 1. As we have seen in chapter two, the entropy of t is maximized when *p*=1-*p*=0.5, we use the entropy as a measure of the statistical distribution of 0’s and 1’s in both the text and the image.

## Image model

A digital image is the most common type of carrier used for steganography. A digital image is produced using a camera, scanner or other device. The digital representation is an approximation of the original image. The system used for producing the image focuses a two dimensional pattern of varying light intensity and color onto a sensor.

An image can be considered as a 2-dimentional matrix, in which each element has an intensity value *f(x,y)*, where *x,y* are spatial(space) coordinates, each one of these elements is called image element, or pixel, where each pixel has an intensity value.

In order to convert a picture into a digital image, the coordinates as well as the amplitude needs to be digitized, digitizing the coordinate’s values is called sampling, and digitizing the amplitude values is called quantization. After that, when *x,y,* and the amplitude value *f* has finite, discrete quantities, the image is called digital image.

There are many types of digital images, the simplest type is the monochrome (binary images), where each pixel is represented by only one bit (0 or 1), either white or black intensity, which would be a bad choice for hiding information inside as any change in the intensity value would result in a huge degradation in the image. Another type is the RGB format, where each pixel is represented by three intensity values: Red, Green, and Blue, which means that each pixel is represented by three bytes.

In our project, we use the gray scale images, where each pixel is represented by one byte, the intensity values of pixels range from 0 to 255, and we use the LSB of pixels to hide the ***t*** vector.

## Mathematical modeling of the system

The next step after making the ***t*** vector is to make the ***q*** vector, which represents a sequence of bits taken from the LSB of the intensity values of a sequence of pixels.

There are many ways to make the ***q*** vector; the simplest one is by taking the LSB of the pixels sequentially, row by row, starting from the first pixel until reaching the last one. Another way is the odd-even algorithm, in which the LSB of even pixels are taken, and then the LSB of odd pixels are added to from the ***q*** vector.

However, in our system we use a different technique; we divide the ***t*** vector into a number of sub-blocks and then each sub-block is hidden inside a sequence of LSB’s of the image, by dividing the ***t*** vector into smaller sub-blocks it becomes easier to find a sequence of LSB’s in the image that has approximately the same sequence of bits as this sub-block.

Suppose that the vector ***t*** has a length of *T* bits, and suppose that it is sub-divided into *n* sub-blocks with *r* bits inside each as shown in Figure 4.

 *r*

 *t1  t2 …………………………………….. tn*

***t***

 *T*

Figure : vector ***t*** sub-divided into *n* sub-blocks each contains *r* bits.

The number of sub-blocks *n* can be found from the following relation:

$$n=\left⌈\frac{T}{r}\right⌉$$

After sub-dividing the vector ***t***, the next step is to add a header at the beginning of each sub-block so that the receiver will be able to reconstruct the original ***t*** vector, we call these bits used as a header *control* bits.

The number of control bits needed to represent the number of sub-blocks is *k*:

$$k=log\_{2}n=log\_{2}\left⌈\frac{T}{r}\right⌉$$

After adding k bits at the beginning of each of the *n* sub-blocks, the new vector is called ***t’***, the length of this new vector is *T’*, where:

$$T^{'}=T+nk=T+nlog\_{2}n$$

The last equation shows us that increasing the number of sub-blocks in ***t*** (i.e. reducing the number of bits per sub-block *r*) increases the length of the expanded vector ***t’***, that in turn increases the number of controlling bits required, which is a disadvantage because that makes most of the contents of ***t’*** are controlling bits, which does not provide any information. On the other hand, increasing *n* increases the probability of correlation between sub-blocks and LSB’s of the intensity values of the pixels in the image, which in turn reduces the BER.

This last statement shows a contradiction; do we have to further sub-divide the vector ***t*** and in turn reduce the BER? Or do we have to reduce the number of sub-blocks and reduce the number of controlling bits which do not contribute in carrying information? This tradeoff between the BER and information capacity needs to be taken into consideration when making the vector ***t’***.

Now, and after forming the new vector ***t’***, the next step is to start the embedding process. The first row of the image is excluded; it is only used to store information about the number of sub-blocks *n*, and the number of bits *T* in the vector ***t***, so that the receiver will exactly extract the sub-blocks and re-arrange them into their correct order.

Then, starting from the second row, we take the LSB of each pixel and make the ***q*** vector, which has the same size as ***t’*** vector (*T’* bits), which is sub-divided into the same number of sub-blocks (*n*) and has the same number of bits as ***t ’*** (*r*+*k*).

After that, each sub-block in ***t’*** is compared with those in ***q***, and the bits in this sub-block is over written to that in ***q*** which is most correlated with it, due to this process some of the bits in ***q*** are changed, the new vector is called ***q’*** our aim is to maintain these changes as low as possible.

The BER is defined as the number of bits that have been changed due to the embedding process, divided by the total number of bits in ***q’***, and it can be measured by XORing the ***q*** and ***q’*** vectors, and counting the number of one’s in the resulting vector, as the logical function XOR is 1 whenever the two bits compared are different, then we divide the number of one’s by the length of ***q***.

As an example, the ***t*** vector in ***Figure 5*** is made after replacing each character by its code word, and then it is sub-divided into 6 sub-blocks, each sub-block contains 10 bits, ***t’*** vector is the same as ***t***, but with a header added at the beginning of each sub-block so that the receiver side can re-group the sub-blocks into their correct decision.

10100110101010101001010110100101001010001110101010 1001011010

0001010011010 0011010101001 0100101101001 0110100101000 1001110101010 1011001011010

***t***

***t’***

Header

Figure : ***t’*** vector after adding the control bits to ***t***

Note that the length of ***t*** is 60 bits, while the length of ***t’*** is 60+3\*6=78 bits.

## Information channels

Figure 6 shows a Binary Symmetric Channel (BSC), where *pc*(0) and *pc(1)* are the probabilities of sending 0 and 1 respectively, while *ps*(0) and *ps*(1) are the probabilities of receiving 0 and 1 respectively.

 0 p 0

Figure : the binary symmetric channel

 *C* 1-*p S*

 1 p 1

The probability *p* is the probability of receiving a 0 if a 0 is sent, or the probability of receiving a 1 if a 1 is sent (no error), while the probability 1-*p* is the probability of error (probability of flipping the LSB of a certain pixel). Accordingly, the probability *p* is the probability of correct decision, while the probability *p*=1-*p* is the probability of error. Where:

$$p=p({s\_{j}=1}/{c\_{i}=1)=p({s\_{j}=0}/{c\_{i}=0)}}$$

$$\overbar{p}=p({s\_{j}=1}/{c\_{i}=0)=p({s\_{j}=0}/{c\_{i}=1)}}$$

Define $p\_{ij}=p({s\_{j}}/{c\_{i}})$ which is the probability of receiving symbol *sj* given that the symbol *ci* was sent. A relation between the two input symbols, and the two output symbols can be derived easily; for example, there are two ways in which a 1 might be received: if a 0 is sent, a 1 is received with probability *p01* , if a 1 is sent, a 1 is received with probability *p11* , therefore we write:

$$p\left(s\_{j}=0\right)=p\left(c\_{i}=0\right)p\_{00}+p(c\_{i}=1)p\_{10}$$

$$p\left(s\_{j}=1\right)=p\left(c\_{i}=0\right)p\_{01}+p(c\_{i}=1)p\_{11}$$

By substituting: *p00*=*p11*=*p*, and *p01*=*p10*=*p*  in the above equations, we get the following:

$$p\left(s\_{j}=0\right)=p\left(c\_{i}=0\right)p+p(c\_{i}=1)\overbar{p}$$

$$p\left(s\_{j}=1\right)=p\left(c\_{i}=0\right)\overbar{p}+p(c\_{i}=1)p$$

From the above equations we see that if *p*=1 , which is the optimum case (BER=0), in this case there would be no distortion in the image.

# Chapter 5: System Performance

In this chapter we discuss the effect of dividing the text binary vector into blocks on the error when hiding it in the image.

As we showed earlier, the text file is compressed using Huffman coding technique in order to minimize the size of the text; each character is assigned a unique code, the length of code words depends on the nature of the language; for example, in English language the character “e” is repeated more than, say, the character “j”, so, the codeword for “e” must be shorter than that for the “j”.

Our objective is to hide maximum data in the image with least error that appears due to the embedding process:

 

After representing the text by the binary vector *t,* it is divided into packets, and a header is added at the beginning of each packet, the header length depends on the number of packets and is given by:

$$k=log\_{2}\left(n\right)$$

Where *n* is the number of packets.

Now we have the new *t* vector, which includes packets of data, and each packet is labeled by a header.

In the same manner, the *q*  vector, which represents the least significant bits of the image pixels is divided into packets of the same length as in the *t*  vector, each packet in the text binary vector is compared with the packets of the image binary vector, and replaces that which results in least error, until all the packets are hidden in the image, after that a new image, called the stego image, which contains the hidden data is saved.

## Numerical Results

As discussed earlier, and from figure 1 we see that the maximum entropy for a binary source is one, which occurs when both symbols (1 and 0) have the same probability. Based on that, and by studying the following figure which relates the text length in Kbits with its entropy we can see that as the length of the text increases, its entropy increases as well.



 Figure : Text length vs. entropy

The reason behind increasing the entropy of the text is that as the text size increases the distribution of the 0’s and 1’s in the binary vector *t* become approximately equal.

The average code word length is 4.5 bits/character, which is much less than the ASCII representation which uses 7 bits for each character, in this way the text size can be reduced significantly which maximizes the amount of data that can be hidden inside the image.

Some bits in the image are reserved to store the size of the text and the length of the header so that the data can be extracted.

Suppose that a 256x256 image will be used, the number of bits in the *q* vector is 256\*256= 65536 bits, and suppose that a header of 6 bits is used which divides the text into 2^6= 64 packets of data.

To calculate the maximum size of text that can be hidden inside the image is 65536-6\*64= 65152 bits, dividing by the average code word length, 14000 characters can be hidden in this image, approximately.

The following figure shows the relation between the header length and the error in the image:



 Figure : The relation between Header length and error

Note that, when the header length is zero (i.e. the text is hidden inside the image as one packet) the error is 0.5, increasing the header length (dividing the text into smaller packets of data) reduces the error down to a critical value, any farther increase in the header will not be efficient. So, an algorithm should be developed to search for this critical value and choose it.

To emphasize the importance of dividing the text into smaller packets of data, lets take a look at figure 9 which shows the relation between the text size and the error, once the text is hidden inside the image as it is, and once a header length of 2 is chosen which divides the data into 4 packets.



 Figure : Text size vs. Error

From figure 9 we can see that the error has been significantly reduced when the data is divided into packets.

Figure 10 demonstrates the relation between the header length and the error that occurs in the image for different text sizes.

 

 Figure : Header Length vs. Error for different Text sizes

From the figure we can see that as the text size increase, the error in the image increases, and increasing the header length will reduce the error up to a critical value where further increase in the header length will not be efficient.

## Software Description

Figure 11 shows the main screen of the software:



Figure : Software Main screen

You can either choose one of the built in images by clicking on one of them, or you can load your image from the PC, after that click on “load text” to choose your text file, and then click on “write data”, this way, the software will do the algorithm and hide the text inside the image, to save the Stego image, click “save image”.

The button “Analysis” will show some important results, like the size of the text file before compression, and after compression, the compression ratio, and the error in the image.

If you have the Stego image, to extract the data hidden inside it, click on “load image”, choose the image, and click “Read data”, then “save file” to save the hidden text file.

# Bibliography

**Abramson, Norman.** *Information theory and coding.*

**Bierbrauer, Juergen.** *Introduction to coding theory.*

**Blahut, Richard E.** *Principles and practice of information theory.*

**Chitode, J. S. 2007.** *Information Coding Techniques.* s.l. : Technical Publications Pune, 2007.

**Glover, Grant.** *Digital Communications.*

**Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins.** *Digital Image Processing Using Matlab.*

1. See Digital Image Processing Using MATLAB, Rafael C. Gonzalez. [↑](#footnote-ref-1)
2. Huffman Coding is to be discussed later in Chapter 2. [↑](#footnote-ref-2)
3. Prefix of the code word means any sequence which is initial part of the code word. A prefix free code is a code in which no code word is a prefix of any other code word. [↑](#footnote-ref-3)
4. See Digital Communications, by Glover, Grant. [↑](#footnote-ref-4)