BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

DIGITAL STEGANALYSIS
DIGITÁLNÍ STEGOANALÝZA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR ATTILA LAKATOS
AUTOR PRÁCE

SUPERVISOR Ing. JOSEF STRNADEL, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Computer Systems (DCSY) Academic year 2018/2019

Bachelor's Thesis Specification

21575

Student:	**Lakatos Attila**
Programme:	Information Technology
Title:	**Digital Steganalysis**
Category:	Security

Assignment:

1. Classify methods from the areas of digital steganography and steganalysis, summarize i) key concepts, principles and properties regarding the methods, and ii) the state of the art in the areas.
2. Choose i) a type of hidden information (text, image, etc.), its properties and steganographical methods and ii) appropriate steganalytic means and methods.
3. Use the choosen means to implement several steganalytic methods.
4. Demonstrate and evaluate the functionality and properties of the implemented methods in terms of detecting hidden information by steganalytic means.
5. Summarize and discuss results you achieved, suggest a continuation and extension of the proposed solution.

Recommended literature:

- Dle pokynů vedoucího.

Requirements for the first semester:

- Items 1 and 2.

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor:	**Strnadel Josef, Ing., Ph.D.**
Head of Department:	Sekanina Lukáš, prof. Ing., Ph.D.
Beginning of work:	November 1, 2018
Submission deadline:	May 15, 2019
Approval date:	April 9, 2019

Abstract

The aim of this thesis is to introduce the basic theory behind digital steganography and digital steganalysis, present some recent algorithms and developments of the fields. Furthermore, it gives an insight into the ancient and recent history of the above stated scientific disciplines. This thesis elucidates the different approaches towards implementation of stegonography using image files. It also attempts to identify the requirements of a good steganographic and steganalytic algorithm and compares their performance with respect to requirements. In conclusion, the author's reflection towards the results, effectiveness of message hiding and success rate in detection are discussed.

Abstrakt

Cieľom tejto bakalárskej práce je predstaviť základnú teóriu za digitálnou steganografiou a digitálnou stegoanalýzou, prezentovať niektoré aktuálne používané algoritmy a vývoj v týchto oblastí. Okrem toho, poskytuje prehľad o starovekej a nedávnej histórii vyššie uvedených vedeckých disciplín. Táto publikácia ilustruje rôzne prístupy k implementácii steganográfie pomocou obrázkových súborov. Práca sa tiež snaží identifikovať požiadavky dobrého steganografického a stegoanalytického algoritmu a porovnávať ich výkonnosť s požiadavkami. Na záver, diskutuje o úvahe autora k výsledkom, účinnosti ukrývania správ a mieru úspešnosti pri odhaľovaní.

Keywords

digital steganalysis, digital steganography, secret message, security

Kľúčové slová

digitálna stegoanalýza, digitálna steganografia, skrytá správa, bezpečnosť

Reference

LAKATOS, Attila. *Digital Steganalysis.* Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Josef Strnadel, Ph.D.

Rozšírený abstrakt

Utajenie informácií v posledných desaťročiach získalo širšie publikum a to sčasti v dôsledku podozrenia, že technológia môze byť využívaná terroristom na oznamovanie plánov na nadchádzajúce útoky a kvôli týmto predpokladom bola technológia témou rozsiahlej diskusie v rámci IT komunity. Na druhej strane môžu ludia profitovať z ich aplikácie, napríklad môžu byť použité na ochranu autorských práv. Táto práca poskytuje prehľad o digitálnej steganográfie a stegoanalýze, ďalej z hladiska použitého krycieho média sa zameriava hlavne na digitálne obrázky.

Existenciou metód na skrývanie tajných správ v digitálnych obrazoch vznikla zároveň aj stegoanalýza. Pochopenie celkovej štruktúry steganografie je nevyhnutné na odstránenie týchto činností. Práca sa tiež snaží demonštrovať rôzne metódy, ako zistiť existenciu skrytých informácií v nevinne vyzerajúcich nosičoch digitálnych obrazov. Riešením tejto práce tvoria časti ako základná teória za digitálnou steganografiou, stegoanalýzou a implementácia niektorých aktuálne používaných algoritmov z týchto dvoch vedeckých disciplín. Z hľadiska stegoanalýzy táto práca kladie veľký dôraz na cielené techniky, ktoré sú veľmi efektívne pri útoku na štegoobjekty, na ktoré sú zamerané.

Bola zvolená steganografická metóda LSB, ktorá je zároveň najrozšírenejšia a najpoužívanejšia. Na odhalenie skrytých správ zakódované s touto metodou boli naimplementované stegoanalytické algoritmy ako Chi kvadrát test a RS analýza. Tieto dve techniky spolu úzko súvisia, avšak sa odlišujú v podstatných detailoch. U týchto metód je uvedené nielen popis ich fungovania, ale aj výhody a nevýhody, respektíve vhodný spôsob použitia týchto metód. Každá z nich bola vyhodnotená z troch rôznych aspektov, vrátane úspešnosti detekcie štegoobrázka, odhadu veľkosti správy a rýchlosť stegoanalýzy.

Testovanie stegoanalytických metód prebiehalo prostredníctvom dvoch sád PNG obrázkov. Každý obrázok z prvej skupiny bol neskôr prevedený do odtienov sivej. Potom všetky takto vytvorené obrázky boli naplnené informáciou s dĺžkou 30, 70 a 100 percentov ceľkovej kapacity daného obrázka. V druhej skupine sa nachádzajú farebné obrázky naplnené tajnou správou viac ako 12 rôznymi spôsobami.

Na základe experimentálnych výsledkov môžeme konštatovať, že obe stegoanalytické metódy majú svoje výhody a nevýhody, napríklad metoda Chi kvadrát veľmi presne dokáže odhadnúť dĺžku skrytej správy v štegoobrázku ale často označí ich ako podozrivé i vtedy, keď vôbec neobsahujú zakódovanú správu. RS analýza je naopak spoľahlivá pri odhalení podozrivých obrázkov, ale v prípade určenia veľkosti skrytej správy je menej efektivná ako vyššie uvedená metóda. Najväčší rozdiel medzi týmito dvoma metodami je, že metoda Chi kvadrát je nepoužitelná na farebné typy obrázkov.

Cieľom tejto práce bolo oboznámiť čitateľa so základmi digitálnej steganografie a stegoanalýzy. Na začiatku sa LSB zdalo byť nerozlúštiteľný, ale ako sme sa posunuli vpred v pochopení jeho celkovej štruktúry, všetky steganografické algoritmy využívajúce sa vlastnosti LSB boli úspešne steganalizované.

Vzhľadom k tomu, že pokiaľ máme k dispozícii neznámy obrázok, nemusíme vedieť, že ktorá steganografická metáda bola práve použitá. Berúc do úvahy skutočnosť že, že každý rok sa objavia nové a vylepšené metódy na ukrytie skrytej správy, cielená stegoanalýza by mohla byť v budúcnosti menej spoľahlivá. Teda bez pokroku v oblasti stegoanalýzy slepej, sa nové steganografické metódy nebudú dať odhaliť a boj proti nim bude menej efektívny.

Digital Steganalysis

Declaration

Hereby I declare that this bachelor's thesis was prepared as the original author's work under the supervision of Ing. Josef Strnadel, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

. . . .
Attila Lakatos
May 10, 2019

Acknowledgements

I would like to express my sincere gratitude to my supervisor Ing. Josef Strnadel, Ph.D. for guidance and continuous support on my bachelor's thesis.

Contents

Chapter 1

Introduction

The art of information hiding in the last few decades has gained a wider audience due in part to the suspicion that the technology may have been used by terrorists to communicate plans for upcoming attacks. While those claims have never been formally substantiated, the technology has certainly been the topic of widespread discussion among the IT community. On the other side, methods of detecting information and understanding the overall structure of this technology is essential in removing cover from these activities.

The goal of this thesis is to give a brief definition of steganography and steganalysis in general to provide a good understanding of these two terms. Furthermore, demonstrates different methods on how to detect the existence of hidden information in innocent looking carriers of digital images.

This paper is organized as follows: chapter 2 discusses the concepts behind steganography and steganalysis by exploring firstly what it is and how it has been used throughout history and elucidates the exact differences between steganography, cryptography and digital watermarking.

With the respect of the chosen field of steganography - the image steganography is presented, while mentioning its commonly used methods. This chapter also classifies and gives an account of the various approaches that have been proposed for digital image steganalysis specializing in the analysis, identification and interpretation of concealed digital evidence.

Chapter 3 provides technical information about the implementation, including chosen tools. In addition, outlines the structure of the system, showing the various directories and file organisation. Based on this part the strengths and weaknesses of the chosen methods can be analysed which can be seen in chapter 4.

Chapter 5 will encompass the conclusions and the discussion of the results from the evaluation of steganalytic methods.

Chapter 2

Steganography and steganalysis

Steganography is the art and science of invisible communication. The word steganography combines the Greek words stegos, meaning „covered“ and grafia meaning „hiding“ defining it as „covered writing“. The key concept behind steganography is that the message to be transmitted is not detectable to the casual eye. Altough related to cryptography, their aim differs in significant ways. Section 2.3 provides a detailed description about cryptography. In an ideal situation, communications between senders and receivers can not be accessed by third parties [6]. In contrast, steganalysis stands for discovering the existence of hidden information. In general, it is enough to detect if information is hidden in a physical or digital content.

2.1 History

Throughout history, people have always aspired to more privacy and security for their communications. Steganographic techniques have been used for ages and they date back to ancient Greece. The aim of steganographic communication back than and now has not changed, its primary goal lies in hiding information in innocently looking objects and sending it to the expected receiver. However, encoding a hidden message requires awareness of the information hiding procedure. The essential difference between historical steganographic methods from the newer ones is, in fact, only the carrier for secret data. Neither internet access nor digital objects were availabe at that time. Back then, physical covers, such as animal skins and cave paints were the most commonly used resources for concealing information.

An early version of steganography was employed by the Greek ruler Histaeus when he had to send a secret message to Miletus[1]. He carried out to shave the head of a slave in order to tattoo the message on the slaves scalp. When the slave's hair had grown long enough not be recognizable he was dispatched to Miletus. After the serf reached his destination, his head was shaved and the message had been decoded. Another time-honored example of concealing information in physical objects dates as early as first century AD when Demeratus, a Greek wanted to notify his allies about an upcoming invade on Greece. At that time, wax tablets[2] were used as portable writing surfaces, which in this case was also the cover object. He melt the wax, so he was able to directly access the wood in order to engrave the message into it. Next, he covered the tablet with wax again as if it was just

[1]Miletus: It was an ancient Greek city on the western coast of Anatolia

[2]wax tablet: Is a tablet made of wood and covered with a layers of wax

an ordinary one. It took some time, until it was delivered and successfully decoded by a certain woman, called Gorgo.[16]

Another simple form of encrytion where the plaintext[3] is mixed with a large amount of nonencrypted text is called *null cipher*. Basically, encrypted message is included in the text and also can be seen by human eyes. However, if no key or actual encryption is involved, it requires intelligence and extensive human effort to decipher it. Throughout history, it has not been used quiet regulary because it is considered as time-consuming as complicated to produce seemingly innocent covertexts and simultaneously not being suspicous. The following text demonstrates how the above stated method has been used since ancient times. Stringing together the initial of every third reveals „VUTFIT“ as the hidden message.

***V**aluable informations shouln't get **U**ncovered in message encryption **T**echnologies but some cryptanalysts[4] **F**ind a loophole in **I**mperfect algorithms entirely allowing **T**hem to decode messages.*

2.2 Present state

The fast growth in communication technologies has lead steganography and steganalysis to receive a lot of attention in the past few decades. Some are interested in concealing informations, such as protecting their personal data or by hiding the very existence of a communication. On the contrary, others are keen on detecting the existence of communications or in rare situations the content of the whole message. As a matter of fact, there are many legal users for steganography and steganalysis. As an illustration, steganography can be used to create robust watermarking to track and authenticate documents, to ensure private conversations or to manage electronic money [16].

2.3 General concepts

This section deals with the concepts and definitions used in the field of steganography. Firstly, starting by defining the meaning of basic terms and then going over basic steganography and steganalysis features.

Cover object: cover refers to the object used as the carrier to hide messages into it. Throughout the years, many kinds of cover objects have been introduced, for instance image, audio, video as well as executable files.

Embedded data: is the message which needs to be protected from third parties as part of cover objects.

Stego object: it refers to the final object created by a certain steganographic method, which is carrying the hidden message. In an ideal situation it can not be decoded by third parties.

Stego key: it is used to control the embedding process, such as selection of pixels carrying the message [12].

Figure 2.1 demonstrates how the embedded data is being concealed into a cover object resulting a so-called stego object, meanwhile not looking suspicious to human eye.

[3]plaintext: also known as cleartext is an unecrypted infromation

[4]cryptanalyst: The reverse engineering employed to decode the message

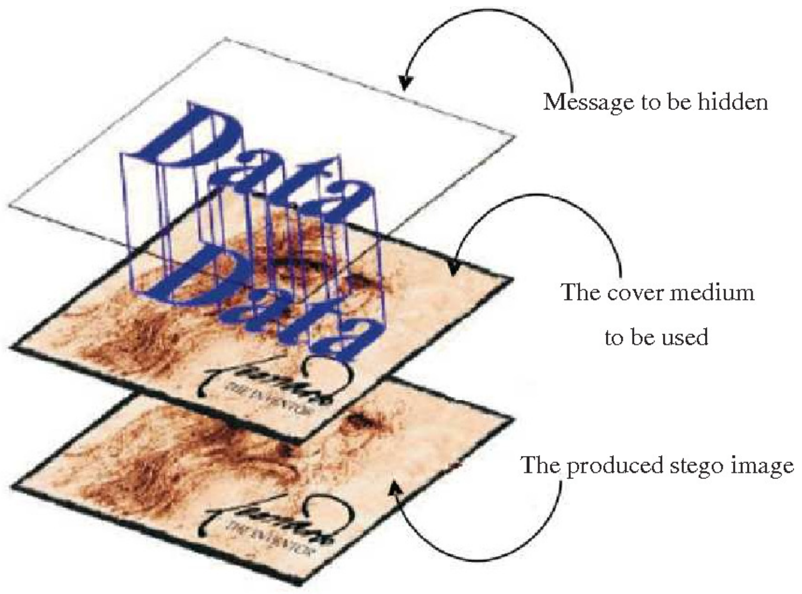


Figure 2.1: Data hiding process [16]

When discussing different information hiding techniques, there are three basic criteria that need to be taken into consideration, such as capacity, security and robustness. These three aspects should be considered as main goals, meanwhile it is indeed needed to find balance between them. Therefore, in the following, each aspect is deeply discussed [12].

- **Capacity** refers to the amount of information that can be hidden in the cover object. In other words, it represents the maximum number of characters that can be encoded into the cover medium.
- **Security** is also defined as undetectability, meaning not to get noticed by third parties. A steganographic method is considered to be insecure, when an eavesdropper is able to distinguish between cover objects and stego objects.
- **Robustness** is the amount of modification that a stego object can withstand without starting to lose its content. In other terms, it is the difficulty of successfully removing hidden information from a stego object without distorting its main content.

In general, steganography aims to hide as much data as it can into cover object, simultaneously striving for not to be differentiable from cover objects. The more data is embedded in a cover, the more likely is going to be recognizable by adversaries. A special case of information hiding is *digital watermarking.* It is a special type of embedding information for instance, mark, label or tag into a multimedia object in order that mark can be later extracted and used for a variety of purposes including to verify the credibility of the content, to recognize the identity of the digital medium's owner and most importantly, to ensure copyright protection [14].

Both steganography and digital watermarking employ steganographic techniques to embed data secretly into cover objects, but there is a significant difference between them. In digital watermarking visual appearance of embedded messages are not considered as a security threat, because the existence of hidden data is not a secret. The level of security rather lies in the difficulty of removing a logo from digital object. In contrary, steganography aims for imperceptibility to human eye.

Another form of information hiding that manipulates messages in order to cipher it is *cryptography*[5]. In general, the key difference between steganography and cryptography is that the first one hides the message so it can not be seen, in contrary, the second one scrambles the message to not be readable by third parties through obtaining a camouflaged form of message. It also provides several encoding schemes for achieving the security. However, both techniques can be combined together resulting more advanced security solutions. Thus, information hiding methods should be robust against potential distortions. Consequently, if steganography fails and the message was detected, the encrypted message is also a troublesome part for cryptoanalists [6].

Figure 2.2 summarizes the main features of each previously mentioned discipline.

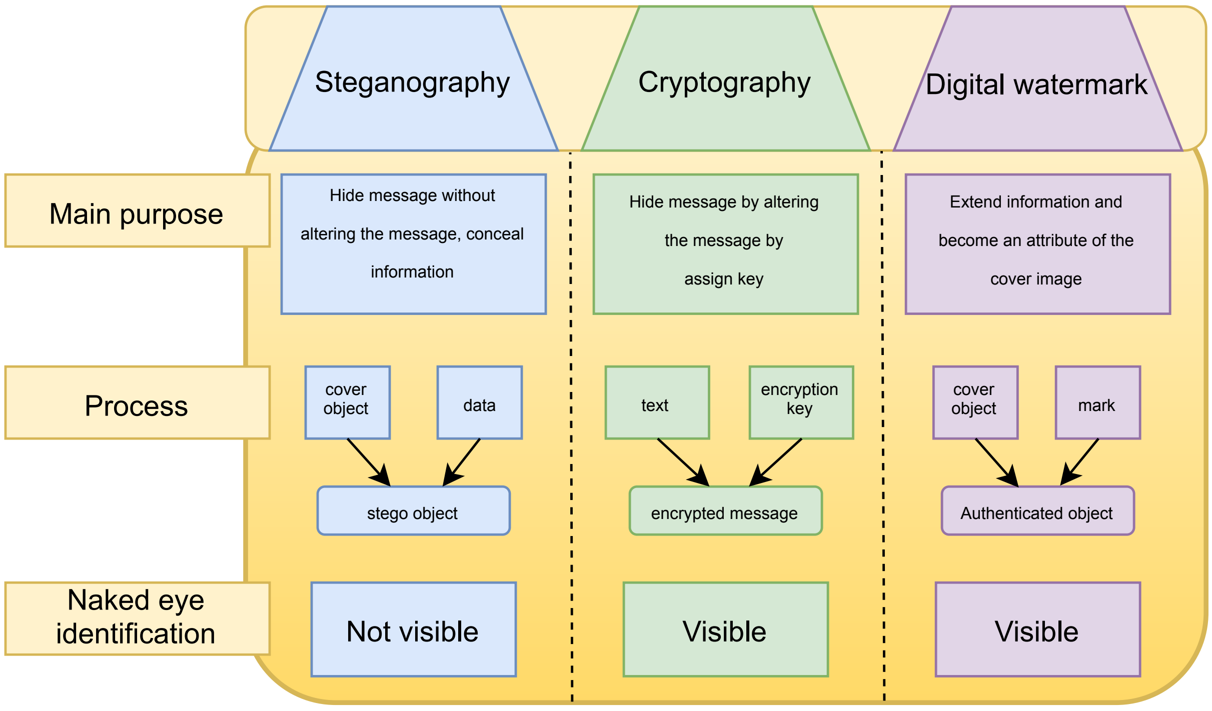


Figure 2.2: Differences between steganography, cryptography and water marking

[5]cryptography: the term has Greek influence which means covered writing

2.4 Digital steganography

Over the last few decades, digital communication has become a major part of people's life. Unfortunately, each step in the advancement of technology, like communication over the internet might have some consequences. The rapid development of communication devices and usage of public domain channels[6] require a massive information security system that is capable to deny adversaries from accessing confidential data. With the increasing communication traffic demand, information security has become a very important discipline.

The following subsections introduce most frequently used steganographic carriers of cover objects and clarify the exact pros and cons of using them. Besides that, each subsection intends to identify the requirements of a good steganographic cover object, but first of all, the reader has to be aware of how the whole process looks like. Figure 2.3 demonstrates a single communication that has been successfully delivered to the receiver.

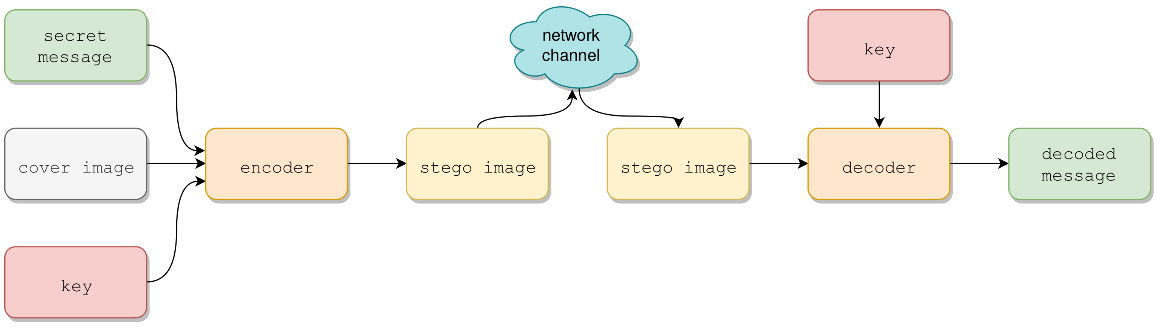


Figure 2.3: Steganography process between sender and receiver

To make a well thought out decision on which type of cover object should the thesis deal with, there had to be a research for all types of well-known cover objects. Present day steganographic methods are way harder to notice than their ancient forms, but the main principle has not changed at all. The only difference between ancient and contemporary steganographic methods is the carrier of cover object.

Text steganography

Throughout history, a number of different cover objects have been introduced. Nowadays, text steganography becomes more and more forgotten due to lack of large scale redundancy in it. After taking into consideration previously mentioned property, it seems to be the most difficult kind of steganography.

During the process of text hiding in text files, it is particularly important to preserve its overall structure. Otherwise, with a simple miscalculation the final stego object might get damaged, resulting useless text files [3].

[6]public domain channel: for example Internet

Audio steganography

Audio steganography in a nutshell, is the technique to hide information within audio files by unfairly taking advantage of human auditory system. HAS[7] is the sensory system responsible for the sense of hearing over a large range of frequencies. However, it contains holes large enough to conceal secret information while not changing the quality of sound. It may produce an audio stego object which does not raise suspicion to human ear [2].

The key difference between using audio and text steganography lies in their capacity, in other words, the number of confidental characters that can be stored in a cover object without raising attention.

Video steganography

Using video as a cover is a relatively new approach towards steganography. In general, it is a combination of audio and image files. It is essential that only certain frames should be manipulated not to cause an unpredictable behavior of steganographic method, making final stego object indistinguishable from the original one after re-extraction. On the other hand, video files are much larger than stand-alone image or audio covers and therefore have much larger capacity [2].

Video manipulation is also harder for a normal person to recognize that it has been slightly modificated. The main reason that lead the author to this statement was caused by the fact that typically every single video frame is visible only for a couple of seconds. Secondly, it is well known that only a few percent of videos contain sharp images.

Network steganography

Thanks to the huge amount of data and vast number of various internet protocols more and more possibilities related to data hiding have been born in the last few decades. Internet-protocol steganography utilizes communication protocol's control elements as a cover for hidden data which differs from the usual methods used so far. Consequently, the protocol header serves as a carrier that conceals secret characters, simultaneously not making possible for third parties to distinguish if the header was produced by default or it has been slightly modificated by a steganographic encoding algorithm [4].

[7]HAS: human auditory system

2.5 Digital image steganography

The huge advancement in the industry of photo cameras and consequently the continuous appearance of various image files attracted stegonography users to embed data into digital image objects. Due to lack of high sensitivity toward HVS[8] and the transfer of huge amount of images through internet made image steganography the most commonly used amongst cover objects. As a result to these contributions, in the past few years, more and more steganographic techniques have been proposed. Each technique modifies the cover image in its unique way, simultaneously making it not to be recognizable to the naked human eye. However, only a few of them has the ability to apply bit alterations on colored images, others only on gray-scale cover objects.

The most significant fact to be able to detect suspicious pixels in image cover objects lies in understanding the basic logic behind the methods of this field. In this section, the thesis will discuss the most common approaches toward contemporary digital image steganography, by classifying them into 2 major groups: *spatial domain embedding* and *transform domain embedding* [17].

2.5.1 Spatial domain techniques

The Spatial domain technique category includes algorithms that are based on hiding a secret message in the least significant bit layers of images. A huge amount of steganography users have been attracted by using LSB[9] techniques due to its simplicity. In addition, in terms of capacity, its payload is huge. However, the lack of complexity has some consequences, for instance, it is weak in resisting attacks, like compression and transforms.

Least significant bit(LSB)

As it was mentioned earlier in subsection 2.5.1, least significant bit is the most common and simplest approach for concealing information in digital images. Basically, the algorithm is defined as the following: each bit of the secret message is distributed, which makes possible to hide it evenly within the image. As a result, the picture does not raise suspicion to the human eye, however, statistical properties of the stego object change significantly. These arguments will be underpinned and demonstrated in section 2.6. In the last few years, various approaches have been proposed using LSB, some of them relies on changing the last bit of certain pixels by incrementing or decrementing its value, others seek for areas where the probability of getting caught by steganalysers[10] is much more smaller, like near edges [12]. Figure 2.4 demonstrates how a simple least significant bit encoding process works by using 2 bits from each character.

[8]HVS: stands for human visibility system
[9]LSB: Least significant bit
[10]steganalyser: steganographic analyzer

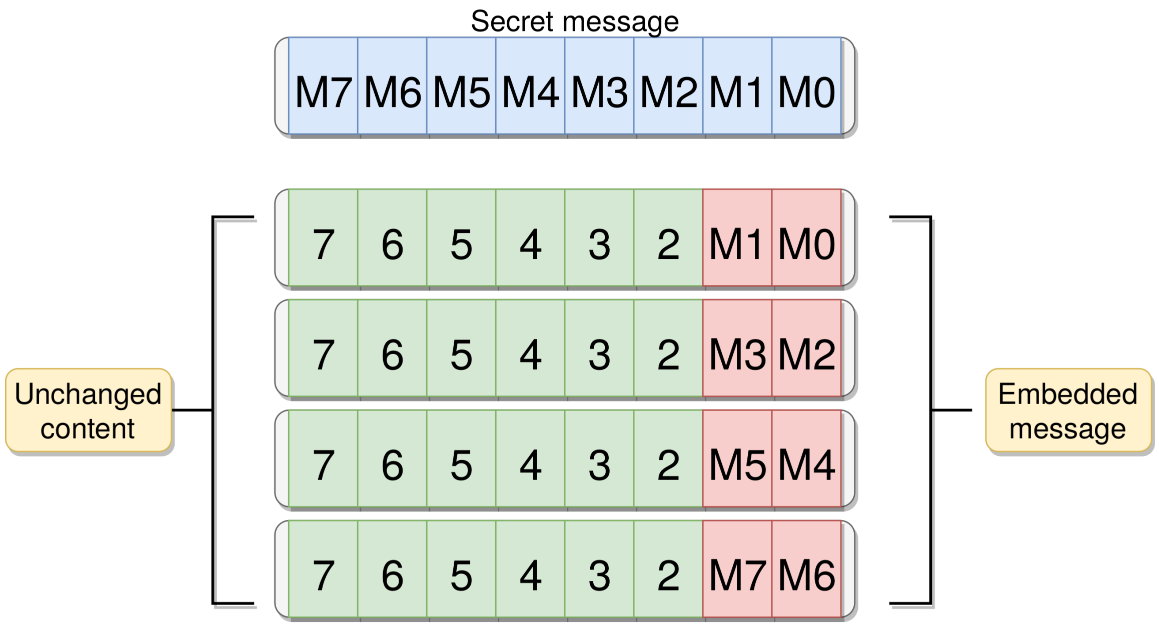


Figure 2.4: LSB encoding process

Randomised LSB

To improve the security of an LSB embedding algorithm, it is highly recommended to include non-sequential approach. The core of the embedding process is basically the same as it was in subsection 2.5.1. However, at the very beginning, it scatters the order of the pixels by applying a so-called *Pseudo Random Number Generator*(PRNG) on the image. In addition, PRNG takes a number k that is called seed and produces a shuffled version of the original image. To summarize, this method is considered to be more secure due to it does require the right key to be able to successfully decode the hidden data [16].

2.5.2 Transform domain techniques

As it was mentioned earlier in subsection 2.5.1, using LSB embedding techniques is the easiest way to hide the existence of a communication, but they are highly vulnerable to be discovered by third parties. To avoid and counteract this, a whole new category of various steganographic methods has been developed during the last few decades, resulting a more comfortable and less suspicious outcome - *transform domain.* This division ensures more robustness against resisting attacks, like compression and transforms than spatial domain techniques by manipulating the image indirectly using various approaches. The basic logic behind this category is to make a use of redundancies in Discrete cosine transform domain, which is used in JPEG compression. Due to this fact, the thesis will now focus on JPEG formats [16].

To get a clear overview of these techniques, it is essential to mention how JPEG lossy compression works. First of all, it converts the spatial domain into the frequency domain, which presents the data as high and low frequencies. Due to lack of high sensitivity toward

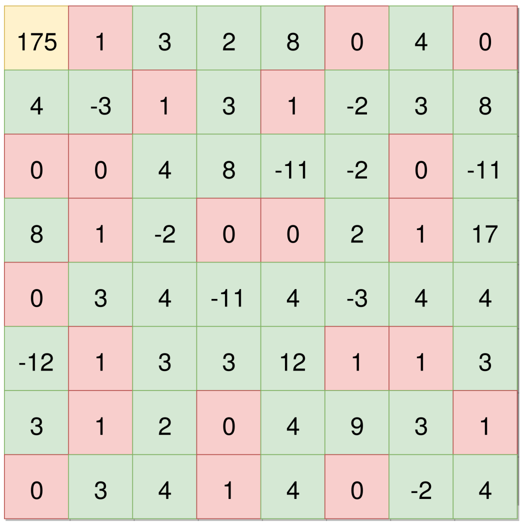
HVS[11], high frequency information can be abandoned, meaning it results an unnoticable image distortion that is unrecognizable by human eye [16].

Recently, a huge number of different algorithms have been proposed, but the thesis will only deal with recently featured ones. There are several widely used transform domain techniques that could be conceivably used, but in order to focus on a certain category, this section is based on using *Discrete Cosin Transform*.

JSteg

The *JSteg algorithm* has been introduced by Derek Upham and it is in a nutshell a combination of two previously mentioned techniques, which include embedding data in LSB that is at the very beginning converted from spatial domain into frequency domain representing the data as high and low frequencies. Actually, it only differs from LSB because it embeds the message in only certain coefficients of the image, rather than using its pixel values. As a result of that, it barely modificates statistical properties of the cover image. In addition, there is no key required that allows anyone who knows the decoding process possible to extract its content.

First of all, the image is split into 8x8 pixel areas, also-called blocks which are transformed into multidimensional arrays of 64 coefficients via Discrete cosine transform [16]. In order to represent these values as whole numbers, each value is rounded by a quantizer. At this state, the quality of the image might get reduced in smaller quantities because larger coefficients lose precision while smaller ones might get rounded to zero. An example of how an 8x8 DCT block might look like is shown in figure 2.5.



175	1	3	2	8	0	4	0
4	-3	1	3	1	-2	3	8
0	0	4	8	-11	-2	0	-11
8	1	-2	0	0	2	1	17
0	3	4	-11	4	-3	4	4
-12	1	3	3	12	1	1	3
3	1	2	0	4	9	3	1
0	3	4	1	4	0	-2	4

Figure 2.5: 8x8 table containing DCT coefficients

It should also be noted that as the figure shows, there can be distinguished 3 types of coefficients in each 8x8 block. The value at the top left corner, also refered to as DC coefficient represents the mean value of all the other coefficients. Changing its value would

[11]HVS: human visibility system

highly influence the final output , for this reason JSteg does not embed data over DC coefficients. The remaining 63 coefficients, also-called AC values are used for embedding information bits. However, in order to make the communication more secure, it is neccessary to leave out coefficients with values of zero or one due to the algorithms feature that does not permit embedding on any AC coefficient that is equal to zero or one.

OutGuess 0.1
Neither LSB from spatial techniques, nor JSteg algorithm from transform domain techniques was considered very secure due to their common disadvantage - embedding the secret message sequentially resulting perceptible changes in histogram [18]. In order to counteract this, a new approach has been proposed by Neils Provos that is basically a combination of some of the previously mentioned methods, including Randomized LSB and JSteg [18].

Outguess 0.1 is an embedding algorithm that improves the encoding process by using a pseudo-random number generator. Its primary goal is to overcome steganalysis attacks that look after modifications in DCT histograms. If it is compared to JSteg, the basic logic has not changed at all - the key difference between them lies in using DCT coefficients in a seemingly random order without raising suspicion to χ^2 steganalytic method[12]. Moreover, to retrieve the secrete message by extracting the data bits from DCT coefficients, it is unlikely to happen without being aware of the right key that was used in the encoding process.

2.6 Digital image steganalysis

In section 2.5, various approaches toward digital image steganography have been introduced that gave a brief overview about recently used techniques which embed secret data in digital images. If it is proved that it was used for malicious purposes, then it is imperative that people intend to seek for more accurate and reliable steganalytical schemes capable of breaking steganography.

An important fact that should be noted that a steganalysis attack is considered to be successfull if it is able to detect the existence of a hidden communication with high confidence. Unfortunately, to uncover the actual content and track down its real meaning is unlikely to happen due to the fact that sometimes cryptographical algorithms could have been involved.

The ability to detect secret messages is related to the message length, in other words, a small message embedded within a theoretically larger cover object might not result as much percent of manipulations as if its content was longer [8]. Algorithms for digital image steganalysis can be typically divided into 2 main categories: *specific* and *targeted.*

2.6.1 Targeted steganalysis

The thesis first looks at techniques that are designed for a special purpose to counteract a certain steganographic algorithm. The results from targeted steganalysis approaches are far more accurate, however, they can not be extended to work against multiple embedding algortihms. As opposed to the previously mentioned two main categories of steganalysis,

[12] χ^2: Chi square test, deeply discussed in 2.6.1.3

where the embedding methods were categorized according to the approach that was taken in the embedding process, here it is classified into another three groups that include *visual*, *structural* and *statistical attacks*. This section will focus on describing how steganalysis can be used to counteract embedding methods mentioned in section 2.5.

2.6.1.1 Visual attacks

It is basically stripping away the significant parts of the digital image in order for the naked human eye to analyze whether or not there are any suspicious structures. This can be done by the observation of a single bit plane, or directly on the image. In fact, the very first rule of steganography should involve that any kind of modification made to the cover image should not result quality degradation, otherwise it may lose its functionality [18]. That is the reason why steganalysis by visual attack was mainly used in its early stage.

An example of a successfull visual attack on a greyscale BMP digital image is demonstrated in figure 2.6.

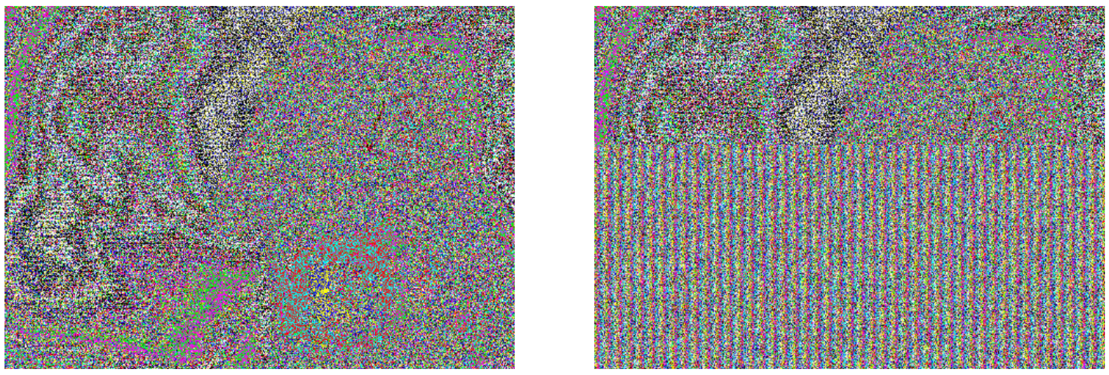


Figure 2.6: The observer's attention is called out by the distortion of the image quality [18]

It is clear that the natural image on the left side might not contain any suspicious structure that would call out the observer's attention. However, after viewing the stego object that is suspected to have been made with sequential LSB embedding, it is obvious that the bottom two-thirds of the image would require further investigation. In addition, if we assume that it contains embedded data it is also possible to extrapolate an estimate length of the altered content, as well as to calculate approximately its overal length by dividing the number of modified pixels by the total number of pixels to get a rough estimation.

What we witness here is the fact that in this example the data was encoded directly after it was converted from ASCII[13] to binary form. If we take into consideration that the embedded text consists of lowercase English letters, it can be deduced that ASCII codes corresponding to these characters fall in the range of 65-90. Thus, the frequency of zeros might dramatically exceed the frequency of ones. If the original cover containing seemingly even distribution of LSB values is modified in the previously mentioned way, it results perceptible changes which might be suspicous to the naked human eye. [8].

As it was mentioned in subsection 2.5.1, a much more effective way to hide secret data is to embed into seemingly random locations of LSB values. Whilst the information is split

[13]ASCII: American Standard Code for Information Interchange

up over the entire image in a theoretically random order, therefore it is most likely to be imperceptible to human visible systems.

2.6.1.2 Structural attacks

Visual attacks on digital images are not used nowadays due to their inefficiency and the fact that a wild range of different stegogrammes are being transmitted over the internet day by day, thus it has lead steganalysis users to somehow categorize each of them. Structural attacks have been created to take advantage of the properties that are mostlikely to exist for a certain steganographic algorithm. For instance, a certain type of LSB embedding operates only on images with constant size - 320x480 pixels [8]. It means that files with this particular image size are automatically considered as stegoggrames and are marked as suspicious.

Structural attacks are not meant to detect that a certain object contains embedded data or not, instead they strive to detect whether they might contain any of the well-known side effects of various steganographic methods. Images that carry suspicious flags are immediately sent for further investigation, nonetheless, not every one of them might have been altered. Computer generated images might seem fake to structural attacks, because they are not affected by the natural elements, which a normal photograph might posses such as shadowing, light or sampling [8]. An incorrect choice can lead to an increase in false-negatives, which is not favorable to steganalysers. With this in mind, structural steganalysis is usually followed by a more thorough investigation.

2.6.1.3 Statistical attacks

As the title implies, this subsection focuses on statistical properties of digital images to determine whether or not they contain hidden data. In general, structural attacks strive to find out if a phenomenon occurs randomly within a digital medium. If we assume that it does not happen to occur in a seemingly random way, then it is marked as a suspicious object and it is usually followed by a more thorough investigation. As it is known, there is usually more image data than message data, however, it is enough to invoke a statistical attack. A huge advantage of statistical attacks against any other types of steganalytic methods lies in they do not require a deep knowledge about how the cover medium actually looks like, instead they just make a hypothesis that is either accepted or ignored [8].

In the last few decades, a large number of various structural attacks have been proposed. The thesis will focus on describing the most effective techniques, as well as what they are capable of and more importantly on which cover mediums can be applied to.

Chi square test

Chi square test, also known as χ^2 test is basically a statistical hypothesis analysis that compares statistical properties of a digital image medium with theoretically expected values such that either accepts the hypothetical statement, meaning the suspected image is a stegoggrame or it declines the hypothesis by not finding any signs of data embedding.

Chi square test relies on the fact that LSB embedding induces categories of two sample values which only differ from each other in the LSBs and are likely to be transformed into each other by the data embedding process - *pairs of values(PoV)*. For example, a pixel with value of 64 in the cover medium will either change to 65(if the current message bit is a 1) or stay 64. Similarly, a pixel with a value of 65 will either change to 64 or stay 65. With this in mind, (64, 65) are pairs of values. To summarize, pairs of values in a grayscale image must correspond to the following form: $\{2m, 2m+1 \mid 0 \leq m \leq 127\}$.

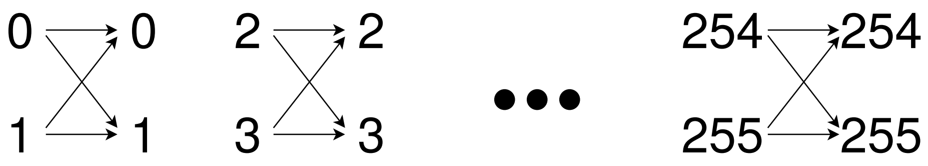


Figure 2.7: Pairs of values in a greyscale image

The inventors of the Chi square test, Andreas Westfeld and Andreas Pfitzmann claim that in an ideal situation, it is uncommon for the frequency of pixel value $2m$ to be equal to the frequency of pixel value $2m + 1$ in a simple image without embedded data. On the other hand, after a secret message is embedded into a cover image the frequencies of 2m and 2m+1 will nearly become equal [18]. Taking advantage of this, the Chi square test was designed to detect these almost equal POVs in images.

The algorithm is defined as the following:

1. Let x and y be two arrays such that

$$\left.\begin{array}{l} x_m = frequency(2m) \\ y_m = frequency(2m+1) \end{array}\right\} \; 0 \leq m \leq 127.$$

   As default, each field is set to zero. What we witness here is that *frequency(i)* counts the occurence of the given pixel specified by i within the image and stores it in the above stated arrays.

2. The theoretically expected frequency for each gray value is determined by:

$$e_m = \frac{x_m + y_m}{2}.$$

3. At this stage it is required to calculate the exact number of categories. In a grayscale image it is achieved by dividing the total number of grayscale values by 2 resulting 128 different categories. In addition, the minimum frequency condition is set to 4 that means whenever the sum of $2m$ and $2m + 1$ is less than 5, both values are set to zero and the total number of categories is decremented by 1.

4. The difference between observed and expected values is measured by the following χ^2 function with $v - 1$ degrees of freedom(DOF), while v represents the total number of categories:

$$\chi^2_{v-1} = \sum_{i=1}^{v} \frac{(x_m - e_m)^2}{e_m}.$$

   The result of this function determines whether the investigated image is a stegogramme or not:

   - **cover image** - x_m and e_m differ in a significant way,

    - **stego image** - x_m and e_m are approximately the same, therefore, χ^2_{v-1} is relatively low.

5. The last step of the algorithm is to calculate the p-value, in other words, the probability of embedding by integrating to find the area under the probability density function with χ^2_{v-1} as its upper limit [9]:

$$p(v) = 1 - \frac{1}{2^{\frac{v-1}{2}}\Gamma(\frac{v-1}{2})} \int\limits_0^{\chi^2_{v-1}} e^{-\frac{u}{2}} u^{\frac{v-1}{2}-1} du.$$

   The larger the χ^2_{v-1} value gets, the more the density function, $1-p$ converges to 1. Thus, the probability of embedding becomes less possible. However, if the density function is relatively small, the probability of embedding becomes larger.

RS analysis

In section 2.6.1.3, we described how the Chi square test can be applied to detect hidden data produced by sequential embedding algorithms. However, the success of the steganalysis attack relied on the very fact that the message was encoded in the same order that it was re-scanned by χ^2. Unfortunately, several steganographic methods randomise the embedding process, making it impossible for the χ^2 test to detect possible stegogrammes.

RS analysis is a more accurate and reliable method that was developed by Jessica Fridrich, for detecting hidden messages embedded in LSBs which have been encoded non-sequentially. In addition, by inspecting the *lossless capacity*, the estimated message length can also be determined [7]. In this scientific field, an algorithm is said to be lossless if after decoding each data bit from a stegoggrame, it is possible to recover the exact original form of the image. Furthermore, it can be employed in 24bit color and 8bit grayscale images as well. To be able to understand how RS analysis works, it is required to make a brief overview of lossless data embedding.

Prerequisites

The whole process starts with the assumption that a cover image has $M \times N$ pixels. In an 8bit grayscale image each pixel might have 256 different values, in other words, $P = \{0, 1, \ldots, 255\}$. In RS analysis, according to a *predefined constant* n - the total number of pixels in a group, the image is divided into disjoint groups each having n neighboring pixels. As an example, it is recommended to set it to either 2×2 or 1×4 pixels in a row, the thesis will further progress with the first one. Next, it assigns a real number to each pixel group $G = (x_0, x_1, \ldots, x_{n-1})$ that is calculated by the discrimination function $f(x_0, x_1, \ldots, x_{n-1})$. The discrimination function describes the regularity of each pixel group. From now on, the following discrimination function is going to be used:

$$f(x_0, \ldots, x_3) = \sum_{i=1}^{2} |x_0 - x_i| + |x_3 - x_i|.$$

The return value of this function is directly proportional to the frequency of each pixel group from vector G. In an ideal situation, LSB embedding increases the regularity or in other words, noisiness in the image. Consequently, three so-called *invertible function operations* are used on pixel x:

- Flipping function $F_1(x) : 0 \Leftrightarrow 1, 2 \Leftrightarrow 3, 4 \Leftrightarrow 5, 6 \Leftrightarrow 7, \dots, 254 \Leftrightarrow 255$,
- Shifter LSB flipping $F_{-1}(x) : -1 \Leftrightarrow 0, 1 \Leftrightarrow 2, 3 \Leftrightarrow 4, 5 \Leftrightarrow 6, \dots, 255 \Leftrightarrow 256$,
- Identity permutation $F_0(x) : F(x) = x$.

Therefore, operation $F_1(x)$ and $F_{-1}(x)$ are used on values from group G based on mask M. M is an *n-tuple* with only 3 possible values $\{-1, 0, 1\}$. According to the choosen discrimination function, operation and mask, group of pixels is distributed into the following categories:

- Regular groups $G \in R \Leftrightarrow f(F(G)) > f(G)$,
- Singular groups $G \in S \Leftrightarrow f(F(G)) < f(G)$,
- Unusable groups $G \in U \Leftrightarrow f(F(G)) = f(G)$.

The main goal of the flipping function F is to simulate the act of invertible noise adding. For example, after using the flipping function it will automatically increase the discrimination function return values. With this in mind, the number of regular groups will be increased and the number of singular groups will be decreased [13].

Steganalytic method

If we define S_M as the total number of singular groups for mask M(in percent) and, similarly, R_M as the total number of regular groups for mask M(in percent) then

$$S_M \approx S_{-M} \wedge R_M \approx R_{-M}. \tag{2.1}$$

Equation 2.1 can be justified by the fact, application of the flipping operation F_1 to an image after its colors have been shifted by one results the same as applying F_{-1} on the original image.

$$F_{-1}(x) = F_1(x + 1) - 1 \qquad\qquad \forall x \in P$$

However, equation 2.1 becomes untrue if, for instance, the image has been altered by a steganographic method using LSB randomization. RS analysis defines p as the length of a random bit stream represented in percent that has been embedded into the image, so it flips $\frac{p}{2}\%$ bits. It results a huge difference between S_M and R_M but as value p increases, the gap between them reduces until it reaches 50% and they become equal. In contrast to the previous relationship, S_{-M} and R_{-M} are directly proportional to the embedded message length p. In other words, the greater the value of p gets, the bigger the gap between S_{-M} and R_{-M} grows. In figure 2.8, x-axis represents the percentage with flipped LSBs and y-axis represents the number of singular/regular groups with mask M/-M.

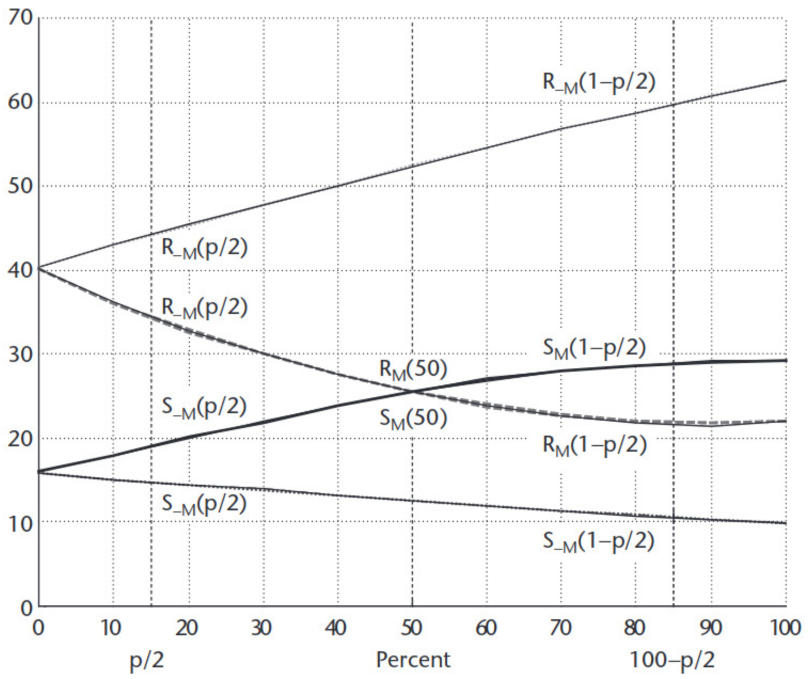


Figure 2.8: RS diagram [7]

Authors of RS analysis state that according to various experiments, it is possible to approximately represent R_M and S_M with second degree polynomals, while R_{-M} and S_{-M} can be modelled with straight lines. To be able to determine the exact form of these functions, it is indispensable to calculate their parameters:

- According to statistics, an avarage LSB embedding will flip only 50% of the embedded area(p/2). Thus, starting points of the relative number of regular and singular groups are located in $R_M(p/2), R_{-M}(p/2), S_M(p/2)$ and $S_{-M}(p/2)$.
- However, if we flip each LSB within the image then the relative number of singular and regular groups will correspond to $R_M(1-p/2), R_{-M}(1-p/2), S_M(1-p/2)$ and $S_{-M}(1-p/2)$.
- To calculate their middle points, it is neccessary to randomize the first half of LSBs and then determine the relative number of each group. To get a relatively precise appropriate number this process needs to be repeated until the number of samples is enough to get a rough estimation of the middle points $R_M(1/2)$ and $S_M(1/2)$. However this operation is time-consuming - the authors of RS analysis propose another solution based on two assumptions:
  - If we assume that I_1 represents a point where curves S_M and S_{-M} intersect each other and, similarly, I_2 represents a point where curves R_M and R_{-M} intersect each other then I_1 and I_2 have the same x value - equation 2.1.
  - The curves S_M and R_M intersect each other when p reaches 50percent. In other words, an image that has been fully embedded in LSBs has zero lossless capacity.

According to the aboved specified assumptions and rules, the authors of the RS analysis extrapolated the following equation:

$$2(d_1 + d_0)x^2 + (d_{-0} - d_{-1} - d_1 - 3d_0)x + d_0 - d_{-0} = 0, \tag{2.2}$$

$$\begin{aligned} d_0 &= R_M(p/2) - S_M(p/2), \\ d_1 &= R_M(1 - p/2) - S_M(1 - p/2), \\ d_{-0} &= R_{-M}(p/2) - S_{-M}(p/2), \\ d_{-1} &= R_{-M}(1 - p/2) - S_{-M}(1 - p/2). \end{aligned}$$

Finally, a rough estimation of the length of the message can be derived from 2.3, where x represents the root of the quadratic equation 2.2 [7]:

$$p(x) = \frac{x}{x - \frac{1}{2}}. \tag{2.3}$$

2.6.2 Blind steganalysis

Up to this point we have only dealt with steganalytic methods that have been designed for a special purpose to counteract a certain steganographic algorithm. Blind steganalysis, also-called universal steganalysis strives to detect suspicious contents in digital cover mediums without being aware of the details of employed steganographic algorithms. It is considered to be more practicable due to the very fact, that it can be extended to work against multiple embedding algorithms.

Actually, it is fairly similar to pattern recognition, which strives to classify images into two major groups - *cover* and *stego* as accurately as it is possible. In general, universal steganalysis consists of two particular steps including **feature analysis** and **classification**. Features are considered to be sensitive to data hiding algorithms, which are used to take down the camouflage of a potential stego object. In an ideal situation, features are different for cover and stego images. Figure 2.9 demonstrates how pattern recognition system looks like in universal steganalysis [20].

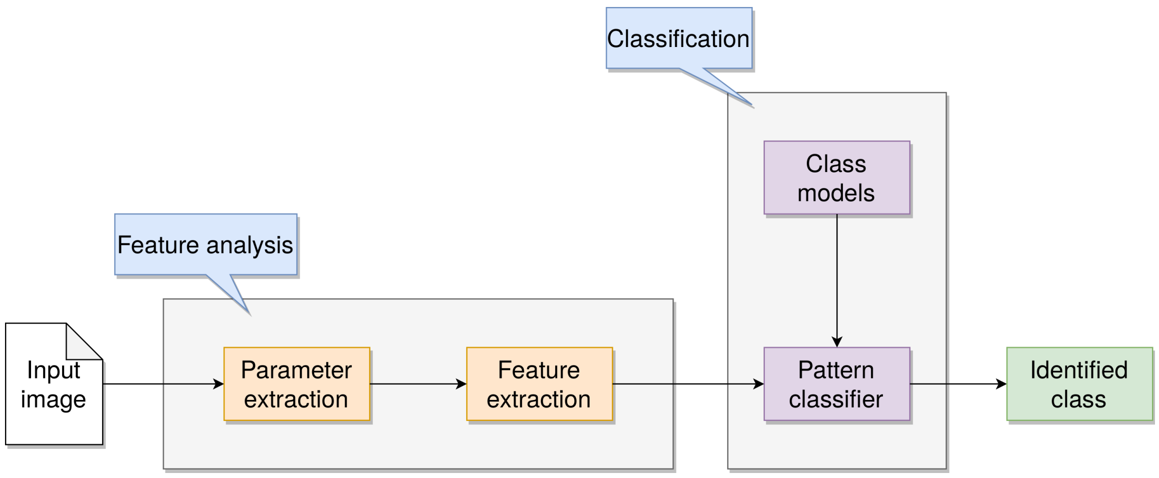


Figure 2.9: General structure of blind steganalysis [20]

The whole process starts with feature analysis, which involves *parameter extraction* and *feature extraction.* The first one ensures that information connected with pattern recognition is collected from the digital medium by constructing a multi-dimensional feature vector. In order to ensure high performance and computational efficiency, it is indeed needed to reduce the dimensionality of the feature vector. The primary task of *feature exctraction* is to provide this functionality.

In the next step, the classifier is able to determine whether it is a stego object or cover image. Pattern recognition system can be divided into two seperate but deeply contiguous parts including training stage and testing stage.

Let's assume that we have N samples of cover and stego images. As a matter of fact, each sample consists of a tuple - feature vector y and associated label ω.

If y_i is original $\implies \omega_i = 1$, otherwise, $\omega_i = -1$. The primary task of the classifier is to learn how to map $y_i \to \omega_i$. This produces a *trained machine.* Once the training stage is finished, it produces a so-called decision function f. In the testing stage, this function is used to determine whether an image does contain embedded data or not.

Chapter 3

Implementation

In this chapter of the thesis, we are going to describe in detail what kind of tools have been used to implement steganalytic algorithms described in section 2.6, furthermore, how certain modules build on each other and last but not least chosen libraries that meet our needs.

3.1 Technologies

First of all, as for the main programming language, *Python3.6*[1] has been selected due to its efficient high-level data structures and relatively simple but also effective approach to object-oriented programming. Combining its elegant syntax and the fact that it does many tasks at runtime, it is an ideal language for rapid application development.

In order to be able to manipulate with pixels, it is unavoidable not to import a library containing basic image processing functionalities, that provides extensive file format support. For this purpose, *OpenCV*[2] and *Pillow*[3] libraries have been taken into consideration. *Pillow* provides less functionality than the first one, nevertheless it ensures pixel manipulation in a safe level that fits our needs.

3.2 Pre-development

At the pre-development stage, a lot of emphasis was placed on designing the final application from the perspectives of reusability and expandability. For instance, to support further image file formats or add new methods. With this in mind, a class named *Common* has been created that ensures methods related to image processing and pixel manipulation. Both *RSAnalysis* and *ChiSquare* implement a separate algorithm for hidden content detection within images described in 2.6.1.3, furthermore, each of them is derived from the previously mentioned class.

Figure 3.1 clearly maps out the structure of the system by modeling its classes, attributes and operations to provide a general overview of the schematics of the application.

[1]Python3.6: `https://devdocs.io/python~3.6/`
[2]OpenCV: `https://pypi.org/project/opencv-python/`
[3]Pillow: `https://pillow.readthedocs.io/en/stable/`

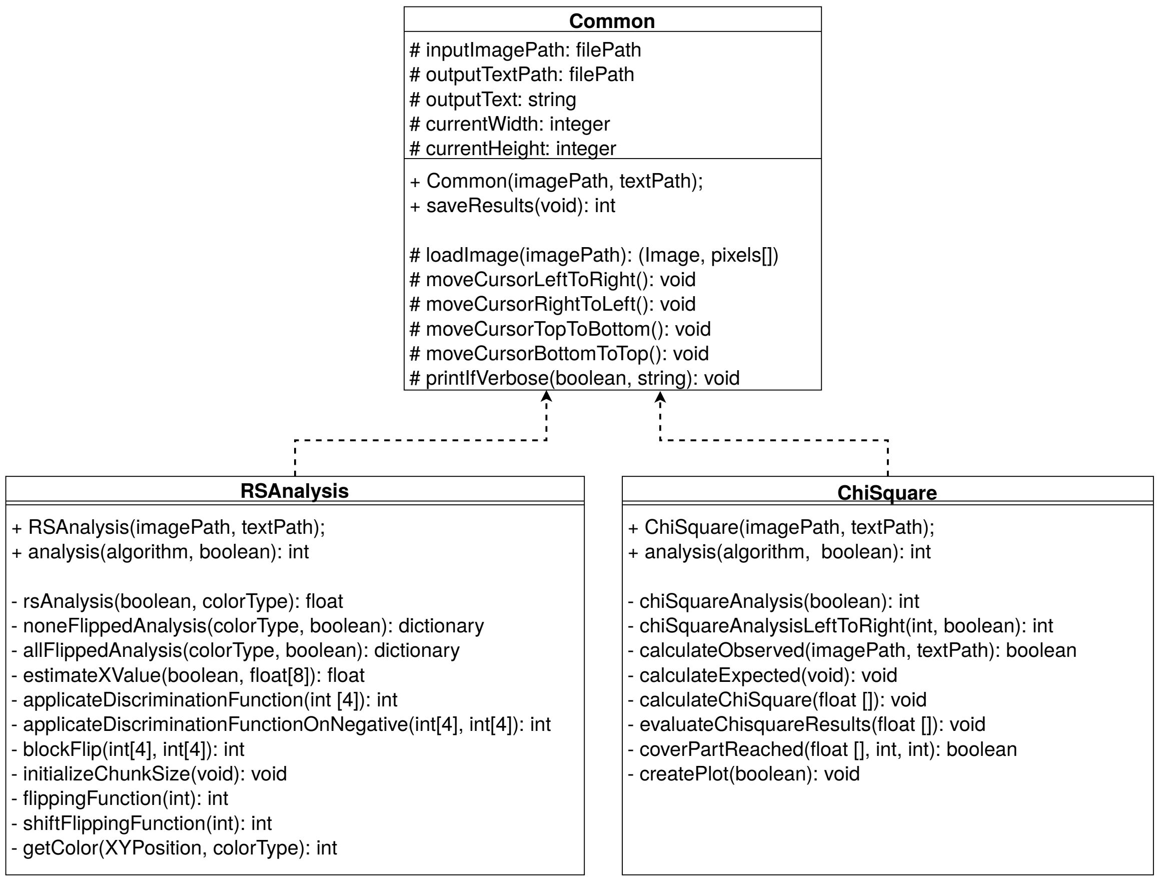

Common
# inputImagePath: filePath
# outputTextPath: filePath
# outputText: string
# currentWidth: integer
# currentHeight: integer
+ Common(imagePath, textPath);
+ saveResults(void): int
# loadImage(imagePath): (Image, pixels[])
# moveCursorLeftToRight(): void
# moveCursorRightToLeft(): void
# moveCursorTopToBottom(): void
# moveCursorBottomToTop(): void
# printIfVerbose(boolean, string): void
RSAnalysis
+ RSAnalysis(imagePath, textPath);
+ analysis(algorithm, boolean): int
- rsAnalysis(boolean, colorType): float
- noneFlippedAnalysis(colorType, boolean): dictionary
- allFlippedAnalysis(colorType, boolean): dictionary
- estimateXValue(boolean, float[8]): float
- applicateDiscriminationFunction(int [4]): int
- applicateDiscriminationFunctionOnNegative(int[4], int[4]): int
- blockFlip(int[4], int[4]): int
- initializeChunkSize(void): void
- flippingFunction(int): int
- shiftFlippingFunction(int): int
- getColor(XYPosition, colorType): int
ChiSquare
+ ChiSquare(imagePath, textPath);
+ analysis(algorithm, boolean): int
- chiSquareAnalysis(boolean): int
- chiSquareAnalysisLeftToRight(int, boolean): int
- calculateObserved(imagePath, textPath): boolean
- calculateExpected(void): void
- calculateChiSquare(float []): void
- evaluateChisquareResults(float []): void
- coverPartReached(float [], int, int): boolean
- createPlot(boolean): void

Figure 3.1: Class diagram

The code is written with the effort of a minimum number of duplicate parts. The possible extension of the application would therefore consist of adding a new class for another type of steganalytic method that is derived from class, Common.

In our case, the implemented Chi square test starts scanning the image at the upper left corner and goes over it horizontally. A suggestion on extending Chi square test would consist of changing *moveCursorleftToRight()* to *moveCursorRightToLeft()*. In other words, it would start scanning the image at the bottom right corner, so it could detect an embedded message which was encoded in this way.

3.3 Development

Since the final program is a command-line application, it accepts various inputs as parameters, as well as options, refered to as flags. Input arguments and associated options must go through a deep parsing process that is achieved by *argparse* module. Basically, it helps to write a user-friendly command-line interface, moreover, automatically generates help and usage messages and issues errors when users give the program invalid arguments.

Steganalysis.py file includes the above-mentioned process, furthermore creates an instance of either RSAnalysis or ChiSquare class depending on which algorithm has been selected.

Object initialization takes two arguments, including a valid path to an image file and a text file where results are going to be saved. Furthermore, it is immediately followed by *analysis(verbose)* method either executing Chi square test or RS analysis. Its return value indicates whether the image is a stego or a cover. If *verbose* is set to true, it reports additional details as to what the program is actually doing.

```
suspectedImageChi = ChiSquare(image.png, results1.txt)
returnCodeChi = suspectedImageChi.analysis(verbose)

suspectedImageRS = RSAnalysis(image.png, results2.txt)
returnCodeRS = suspectedImageRS.analysis(verbose)
```

3.3.1 ChiSquare.py

As we know from section 2.6.1.3 that Chi square test is efficient against sequential embedding algorithms, it can be achieved by different approaches in terms of order of pixel accessing. In other words, starting at the upper left corner and going over the image horizontally, perhaps vertically or doing exactly the opposite.

Besides the fact that *calculateObserved(chunkSize, route)* method uses the first approach, it also creates a histogram by calculating the frequency of each pixel. *Chunksize* indicates how many pixels are going to be processed in a single step. Currently it is set to the total number of pixels within the image, however, we will modify it by the end of this section. *Route* tells the order in which pixels are going to be processed. Next, each value within the histogram has to meet certain criteria, otherwise the value is erased from the buffer. According to the frequency of each pixel and basic theory behind *pairs of values* mentioned in 2.6.1.3, estimated values are calculated via *calculateExpected()* method. At this point, an appropriate Chi square test function is required. Fortunately, Python provides a module, named `scipy.stats` that contains a large number of probability distributions as well as a growing library of statistical functions. Amongst them there is the *chisquare(observed, expected)* method that fits our needs. Its return value determines the following:

- return value > 0.01 - accept the hypothetical statement(stego image),
- return value $<= 0.01$ - decline the hypothetical statement(cover image).

This is ensured by the *calculateChiSquare(probabilities)* method which performs Chi square test and saves its return value into the *probabilities* array.

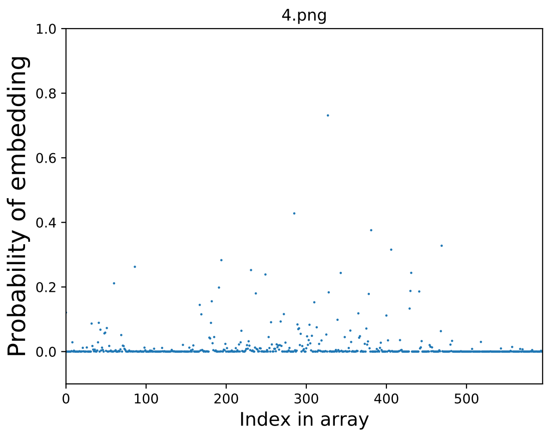
At this point, the user should already be aware of the fact, whether the digital medium does contain embedded data or not. Unfortunately, experiments showed that this method is only effective against images, which have been fully embedded in LSBs. The possibility of that happening is relatively small.

However, there is a way to make this process more reliable and effective. Instead of processing the entire pixel buffer in a single step, we rather divide it into equal parts(chunk), each having 1024 pixels. After that, a Chi square test is performed on each chunk of pixels, based on the above mentioned method. The *analysis(verbose)* method inside ChiSquare class is defined as the following:

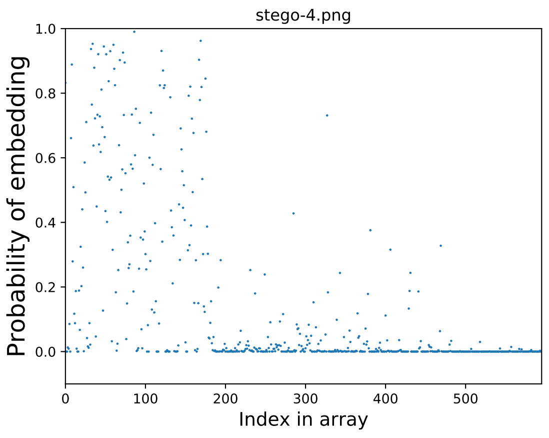
```
while not isOver:
    isOver = not self.calculateObserved(chunkSize, "LeftToRight")
    self.calculateExpected()
    self.calculateChiSquare(self.probabilities)

rc = self.evaluateChisquareResults(self.probabilities)
return rc
```

Note that *calculateObserved()* returns a boolean value indicating whether we have processed every single pixel within the image or not. For further analysis, each return value is stored in an array of floats, used to determine whether there is a significant difference between the expected frequencies and the observed frequencies - ensured by the *evaluateChisquareResults(probabilities)* method. If the consecutive values within the array are relatively small, we can mark the object as a cover image, otherwise it is considered as a stego image. With this in mind, the approximate length of the message can be determined.

In addition, before performing chisquare test on the given digital medium, it is also possible to request a plot of values resulted from each Chi square analysis. In general, it provides further information where the message should approximately be hidden.



(a) Cover image analysis



(b) Stego image analysis

Figure 3.2: Chi square analysis outcome

Figure 3.2a demonstrates on an example how the outcome of a Chi square test might look like on a digital image, which has not been altered yet. Almost every single value, obtained from Chi square analysis is approaching zero. In contradistinction to this, figure 3.2b illustrates the exact opposite (it has been altered up to 30% of its capacity). It is obvious that the first third of the plot raises suspicion to the detector. Based on the size of the area we can roughly estimate the length of the message.

3.3.2 RSanalysis.py

As it is known from subsection 2.6.1.3, Chi square analysis has its upper limits, not only is ineffective against non-sequential embedding techniques, but also powerless if the pixels consist of multiple channels, like in case of an RGB color picture. From this point of view, Regular-singular analysis is much more practicable. Of course, it can be applied to single channel grayscale images too. In this case, the entire analysis will be focusing on the one and only channel of each pixel, meanwhile in case of RGB images the whole process is repeated three times, for each color. The algorithm is defined as the following:

```python
group1 = self.noneFlippedAnalysis(color, verbose)
group2 = self.allFlippedAnalysis(color, verbose)

x = self.estimateXValue(
    verbose,
    group1['positiveRegular'], group1['negativeRegular'],
    group2['positiveRegular'], group2['negativeRegular'],
    group1['positiveSingular'], group1['negativeSingular'],
    group2['positiveSingular'], group2['negativeSingular'],
)

# Calculate the probability of embedding
try:
    p = abs(x / (x - 0.5))
except ZeroDivisionError:
    p = 0

return p * 100
```

Similarly to the previous algorithm, it is indeed needed to divide the pixel buffer into equally large groups, however, in this case we will process only 2x2 blocks of pixels at a certain time. According to the authors of RS analysis, using 4x1 blocks of pixels is also a solution, nevertheless experimental results were far more unaccurate than results got from the previous version. A more detailed description in respect of the experimental results can be found in chapter 4. As the name of the method implies, the first step also includes calculating the number of regular, singular and unusable groups.

It is basically applying flipping function *F1(x)* and shifter LSB flipping *F-1(x)* mentioned in subsection 2.6.1.3 on each block, while measuring the regularity of each pixel group. In order to determine the regularity of a group the following scheme is going to be used:

$$f(x_0, x_1, x_2, x_3) = |x_0 - x_1| + |x_0 - x_2| + |x_1 - x_3| + |x_2 - x_3|.$$

Both *noneFlippedAnalysis(color, verbose)* and *allFlippedAnalysis(color, verbose)* implement previously mentioned functionalities. The only difference which distinguishes them is the mask used inside the flipping functions (see 2.6.1.3). The first argument, *color* indicates which channel is going to be analysed. Their return value gives the total number of regular, singular and unusable groups. These values are then required to calculate the x value.

The number of regular and singular groups at p/2 and 1-p/2, illustrated in figure 2.8 define the straight lines, as well as provide enough constraints to uniquely determine the parabolas and their intersections. To do that, the x value of the intersection point can be estimated from the root of the quadratic equation 2.2 - *estimateXValue(verbose, points[])*. Thus, the probability of embedding and also the estimated length of the message can be derived using equation 2.3.

During the implementation phase, several problems have occurred related to optimization - the previous operations are time-consuming. Let's assume that we intend to analyze a high resolution image that is 4032 pixels wide and 3024 pixels high, in other words it contains more than 12 million pixels in all. In order to make some aspect of it work more efficiently, to spend less time with performing mathematical calculations, only a certain amount of all pixels will be processed. A question which arises, is whether reducing the amount of processed pixels makes final results much less inaccurate.

Surprisingly, after performing some experiments, it has been proven that this assumption is false in the sense of accuracy. As a result of optimization the mathematical calculations have been performing more than 10 times faster, meanwhile the precision almost remained the same, making less than 0.001 differences compared to their original outcome.

3.3.3 Steganography.py

There were several things that needed to be accomplished before being able to try out how effective, perhaps ineffective are both methods mentioned above. As for the first task, a steganographic method has been implemented. Nowadays, a large amount of various steganographic executables can be found over the internet, however, I strived to implement a simple LSB embedding algorithm, described in 2.5.1. It basically consist of sequentially replacing the LSBs of the pixels by the message that needs to be sent. Figure 3.3 demonstrates the order in which pixels are handled.

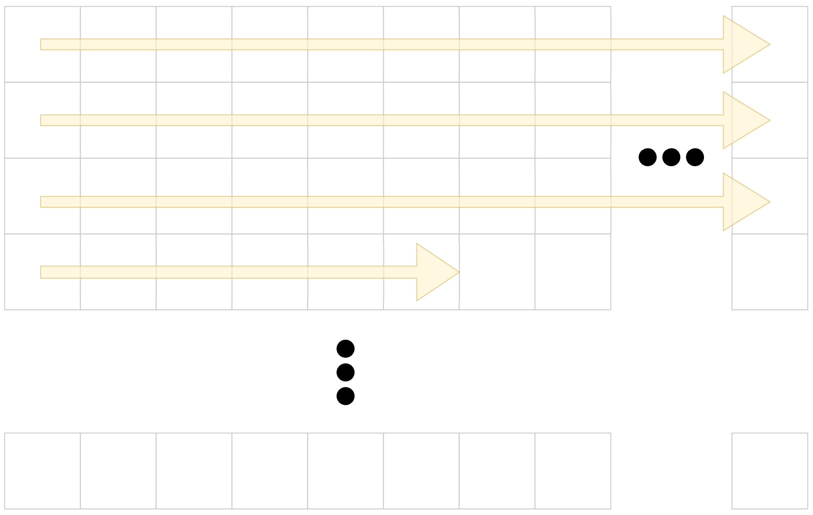


Figure 3.3: LSB embedding route

Chapter 4

Testing

In order to throughly evaluate steganalytic methods by means of their reliability and accuracy, there are certain requirements which need to be done. These prerequisities include choosing an appropriate type of image format that suits our needs and gathering a huge amount of digital images. As the test set grows larger, the manual testing gets progressively more difficult, which clearly shows a need for automated testing.

4.1 Data format and set

PNG(Portable network graphics) is an extensible file format for the lossless and portable storage of raster images. It was developed as an improved replacement for GIF(Graphics interchange format) that allows storage of images with greater color depth and other important information, with minimal cost to developers.

It supports indexed-color(with or without Alpha channel), grayscale and also true-color images. PNG was designed to be portable on the Internet, however, it is useless for professional-quality print graphics, because it does not support non-RGB color spaces.

As for PNG compression, it uses a 2-stage compression process which includes *pre-compression: filtering* and *compression: DEFLATE*. It basically uses a lossless data compression algorithm involving a combination of LZ77 and Huffman coding that suits the needs of a steganography user [19].

The sampling of images which was chosen had to be large enough to avoid random coincidences such as a message within an image is easily detected by a steganalytic method if it has few colors and relatively large smooth regions [5].

The first part of the data set consisted of images 4032 pixels wide and 3024 pixels high acquired by an iPhone 6S mobile phone. The number of test images almost reached 180 samples. Since the final program does not support JPG file formats, each element within the data set had to be converted into PNG format. For this purpose, the *jpg2png*[11] online tool was employed. Inasmuch as the strength of the Chi square analysis relies on detecting embedded data within grayscale images, it clearly showed a need for creating a grayscale version of each image.

Subsequently, we employed a steganographic method mentioned in 2.5.1 on all the elements of the data set. Thus producing an unmodified cover image and 3 additional stego images containing a hidden message with size of 30%, 70% and 100% of its capacity.

The second part of the data set involves digital images 1920 pixels wide and 1080 pixels high. It was provided by Tomáš Kolarčík, whose Bachelor's thesis focuses on digital image steganography. Thanks to him I obtained additional 60 images which have been altered in 12 different ways thus producing more than 1000 samples.

The main difference between the two image sets can be seen on table 4.1. By means of comparison, it is similar to black box and white box testing.

	First data set	Second data set
1	It is a way of testing in which the **internal structure** of the steganographic algorithm **is known**.	It is a way of testing in which the **internal structure** of the steganographic algorithm **is unknown**.
2	Knowledge of implementation is **required**.	Knowledge of implementation is **not required**.
3	It is **most time-consuming**.	It is **less time-consuming**.
4	The entire **algorithm is available** for testing.	Only the **name** of the algorithm is **known**.

Table 4.1: Main difference between data sets.

4.2 Shell script

As the final test set grew larger, manual testing became impracticable. It showed a need for automated testing. For this purpose a simple but also effective shell script has been implemented. It requires 3 arguments, including a valid path to the image set, the name of an appropriate steganalytic method and a path to an either existing or non-existing file, where the final results will be saved.

The script searches the directory tree rooted at the given starting-point, specified by the first argument. Each found image with an extension of png will go through a certain steganalytic method depending on which one was selected by the user. Evaluated results produced by *steganalysis.py* are redirected. Redirection simply means capturing the output and sending it as an input to a file, specified by the user.

The usefulness of these tests can be measured from 2 different aspects. Firstly, one can assess how many stego images are altered. A second approach is to study the differences between measured message sizes and the expected ones.

Basic terms

Researchers have recently used different approaches for evaluation of steganalytic programs. Each of them require a certain amount of knowledge regarding basic concepts. Therefore, we define the meaning of basic terms used in the field of steganalysis during evaluating experimental results [10].

- True positive(TP) - correctly indicates that the stego medium is a stego.
- True negative(TN) - correctly indicates that the cover medium is a cover.
- False positive(FP) - wrongly indicates that the cover medium is a stego.
- False negative(FN) - wrongly indicates that the stego medium is a cover.
- Accuracy value(AV) - it is the degree of closeness of measurements of an observed quantity to that quantity's true value.

$$AV = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

A more comprehensive illustration can be seen in figure 4.1.

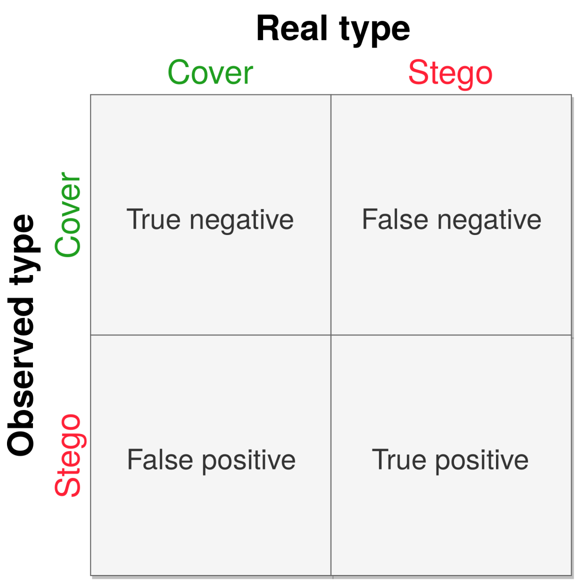


Figure 4.1: Basic terms used during evaluating results

4.3 Experimental results - Chi square analysis

In the following, we present the experimental results of the proposed Chi square analysis outlined in subsection 3.3.1. The set of experiments, obtained from the first data set consists of only grayscale PNG images, deeply discussed in 4.1.

In the first instance, authors of the Chi square algorithm claim that the probability of embedding is near 1 if the entire image has been altered in its LSBs [20]. From a realistic perspective, let's examine the following assumption: the probability of embedding is directly proportional to the embedded message length. Accordingly, the total number of true positives for each set of images should increase with the length of the message. Unfortunately though, these claims have never been formally substantiated. The results obtained from experiments show that the previously mentioned assumption is untrue. As it is seen in figure 4.2, deviation between each group is almost negligible.

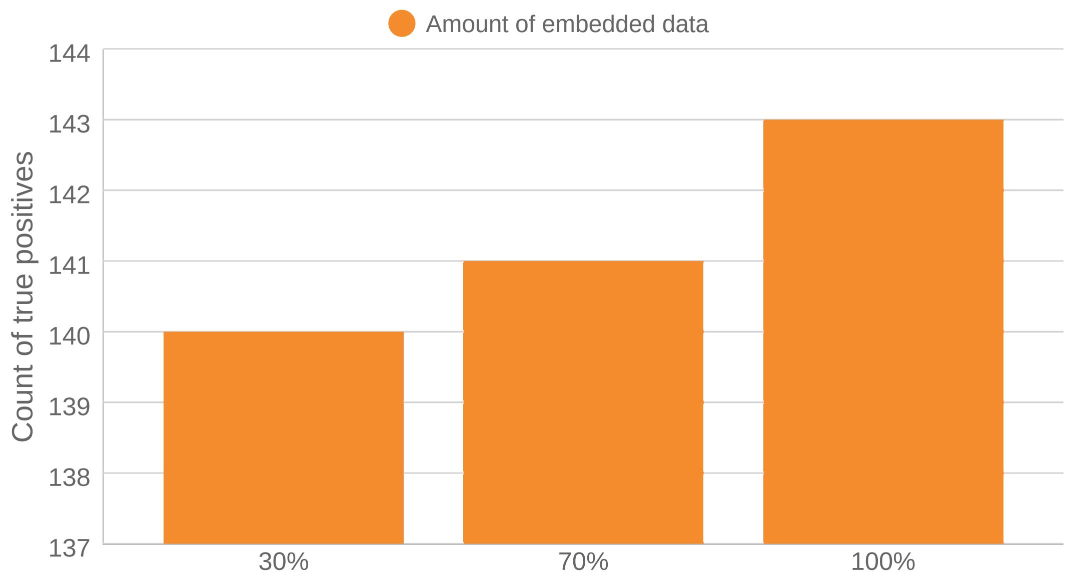


Figure 4.2: True positive rate on the first data set

It is worth pointing out that there are certain images within the first image set which are completely resistant to the proposed method. It has absolutely no effect on the histogram of the image, therefore it does not raise suspicion to the detector. A question which arises, is whether we are able to filter out such images. First of all, Chi square test seems to accurately analyze most of the images that we tested, specifically those with large homogeneous regions. However, if it contains many colors and small smooth regions, the results become less reliable.

In the following, we demonstrate the accuracy of the Chi square analysis, as well as use pie charts to exemplify the ratio of wrongly and correctly judged samples. Figure 4.3 demonstrates how accurately the proposed method is able to determine if the input image is a cover or not. The same applies to figure 4.4 too, except that each image is a stego.

Obtained values are then substituted into Equation 4.1, resulting 82.30% accuracy.

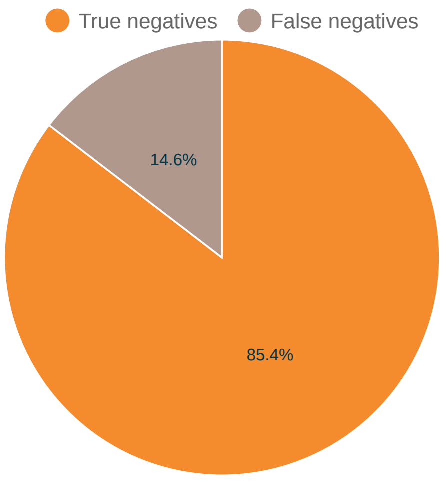


Figure 4.3: Cover image detection rate

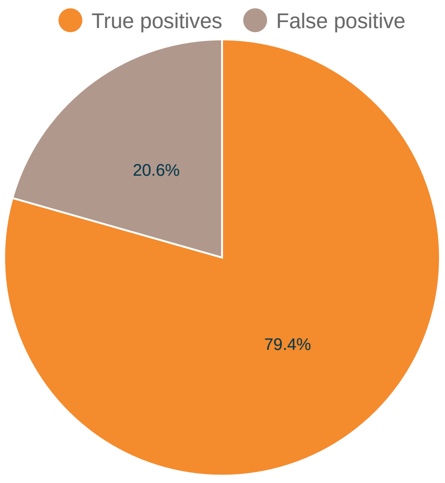


Figure 4.4: Stego image detection rate

Results obtained from these tests are not considered to be bad at all. However, we have to take into consideration that the internal structure of the steganographic algorithm was known. The success of the steganalysis attack also relied on the very fact that the message was encoded in the same order that it was re-scanned by χ^2.

Authors of the Chi square analysis also claim that if the embedded message is randomly spread over the image, the results become inaccurate [20]. This behavior was also tested on the second data set(see sect. 4.1) which consists of various embedding methods including pseudo random generators. Results obtained from this experiment confirmed the previously disputed statement.

For this purpose, 60 images were put to use, where each of them has been altered by randomised LSB. Note that 14% of the image's capacity was utilized. Unfortunately though, none of them was successfully marked as a stego image. If we used 100% of its entire capacity we would certainly be able to notice it. However, when utilizing every single LSB in an image(using 100% of the image's capacity) we lose the meaning of a PRNG generator.

With these facts in mind, we can declare that the proposed Chi square algorithm is not considered to be highly precise.

4.4 Experimental results - RS analysis

In the following, we present the experimental results of the proposed RS analysis method, outlined in subsection 3.3.2. The set of experiments, obtained from the first and second data set consists of both RGB and grayscale images, deeply discussed in section 4.1.

First of all, we will take a look at the same grayscale images which have also been mentioned during Chi square analysis. It is considered to be much more reliable and accurate than the χ^2 test was. Accordingly, we had to prove this assumption. Each experiment from a set of stego images was noticed by our steganalysis detector **without exception**. It should be emphasized that the entire set of images consisted of 3 x 178 images which is remarkably promising. In order to fully trust the given method, it was also tested on a set of cover images. Unfortunately though, 13 cover images were classified as false positives -

falsely detected. However, if we take into account that it also contained a large number of samples then the accuracy value is considered to be quite satisfying, 98.17%.

So far we have only dealt with 8 bit grayscale images, thus it is time to move on to the RGB ones.

Next, we will evaluate results related to stego images which have been altered by 12 different steganographic methods. It is worth pointing out that we had absolutely no knowledge of the employed methods, excluding their names(see sect. 4.1). The true negative rate is 83.33%. Note that the lower limit, from when an image is considered to be a stego was set to 1%. This is why the ratio of false positives is as high as 16.66%. Most of the cover images which have been marked as stegos go over the lower limit only by 1-2%.

In figure 4.5 we use a vertical bar chart to represent a set of numerical data provided by each method. Y-axis is numerical, furthermore describes in percentage the amount of true positives acquired by a given method.

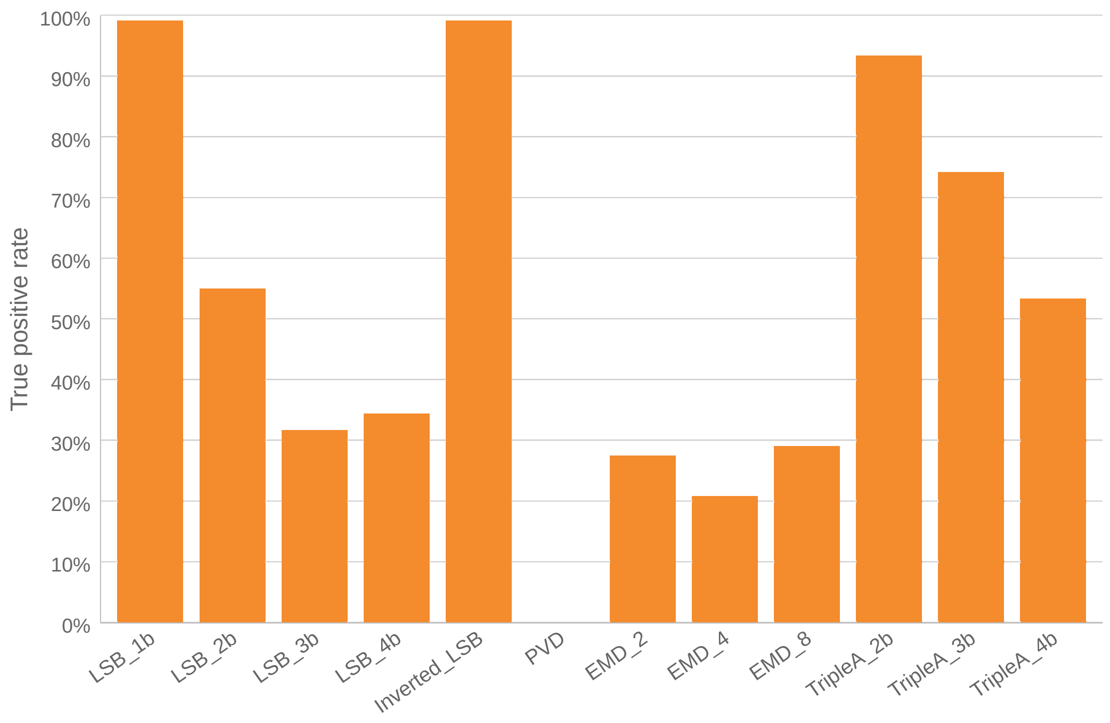


Figure 4.5: True positive rate on the second data set

In terms of accuracy, it can be clearly seen that LSB embedding algorithms are much more perceptible. It is worth pointing out that both sequential and non-sequential approaches have been used. In other words, it proves the assumption that RS analysis is able to detect even randomly distributed messages.

Next, let's take a look at column *TripleA 2b*, the third most perceptible method. It utilizes the same principles as LSB, where secret bits are stored in the least significant bits. Nonetheless, it also enhances its performance by using PRNG[1] and an encrypting algorithm, AES[2] [1]. It is well known that targeted steganalysis is designed to counteract only a certain steganographic algorithm. A question which arises is why were we able to detect even *TripleA* with an accuracy of 93.33%.

The answer is quite simple - each embedding technique which utilizes the same principles of LSB is relatively sensitive to RS analysis.

Another interesting phenomenon is the inaccuracy of column *LSB 4b*. Although it uses the same approach as *LSB 1b*, results are quite unsatisfying. In order to demonstrate the meaning of this, we are going to use visual steganalysis, mentioned in 2.6.1.1. Figure 4.6, altered by *LSB 1b* does not raise suspicion to the naked eye, meanwhile the upper side of figure 4.7 which was altered by *LSB 4b* would definitely call out the observer's attention.

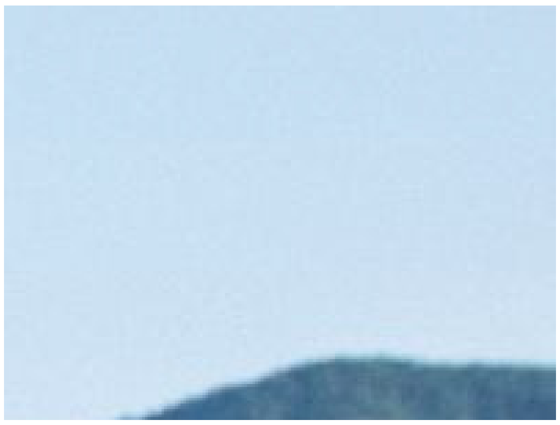


Figure 4.6: Using 1 bit of each pixel

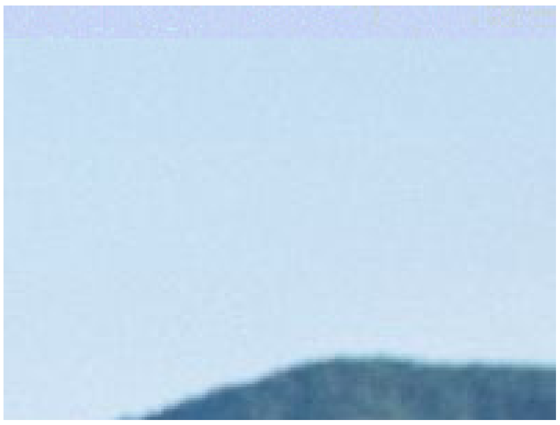


Figure 4.7: Using 4 bits of each pixel

What we witness here is what happens when instead of using only the least significant bit, someone makes a use of 4 bits of each pixel. It does raise suspicion to the naked human eye, meanwhile the proposed RS analysis is ineffective. This can be explained by the fact that during implementation phase, we were focusing only on the least significant bit. With this in mind, if a steganography user makes use of more than 1 bit from each pixel then final results become much more inaccurate.

As it was mentioned in 2.6.1, targeted steganalysis approaches are accurate, nevertheless they cannot be extended to work against multiple embedding techniques. This can be proved by taking a look at the column *PVD* and each variation of *EMD*. None of them relies only on the least significant bit, instead of that an appropriate combination of LSB substitution and pixel value differencing(PVD) or exploiting modification directions(EMD) is applied [15].

[1]PRNG: pseudorandom number generator
[2]AES: Advanced Encryption Standard

Estimated message length

Regarding embedded message length, we will a give a comprehensive overview related to comparison of achieved and true(real) results. In the following example, we are given a set of 120 images. Each of them has been altered by *LSB 1b*, utilizing 14% of its entire capacity. The message size estimated by RS steganalysis is presented in figure 4.8. It is worth pointing out that the average value is 16.04% which barely differs from the real one.

The differences in the message size estimation between sequential and pseudo-random stego images were also found to be negligible. Another interesting phenomenon is that results obtained from *LSB Inverted* method are basically the same as shown in figure 4.8. This is because it also utilizes the same principles as *LSB 1b* method. Not surprisingly, the average message length is 16.63%.

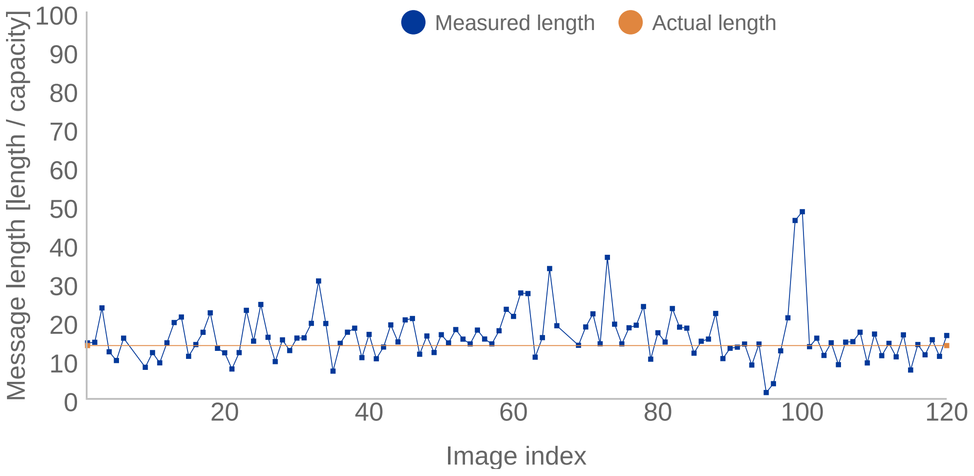


Figure 4.8: Comparison of measured and true message lengths

4.5 Comparison

So far, during the evaluation of experimental results, we were focusing only on a certain steganalytic method. Because the advantages of RS analysis over Chi square test have not been formally substantiated, we will take a look at it from different perspectives. For this purpose the first set of grayscale images will be put to use (see sect. 4.1). Each test image was produced by sequential LSB embedding method with message sizes ranging from zero to the maximal capacity. Since each image was altered with different message sizes, we had repeated the detection ten times for each image and averaged p values. Results are shown in figure 4.9. On the y-axis the relative measured length of the message, and on the x-axis the true length. Note that the line chart consists of only images marked as true positives.

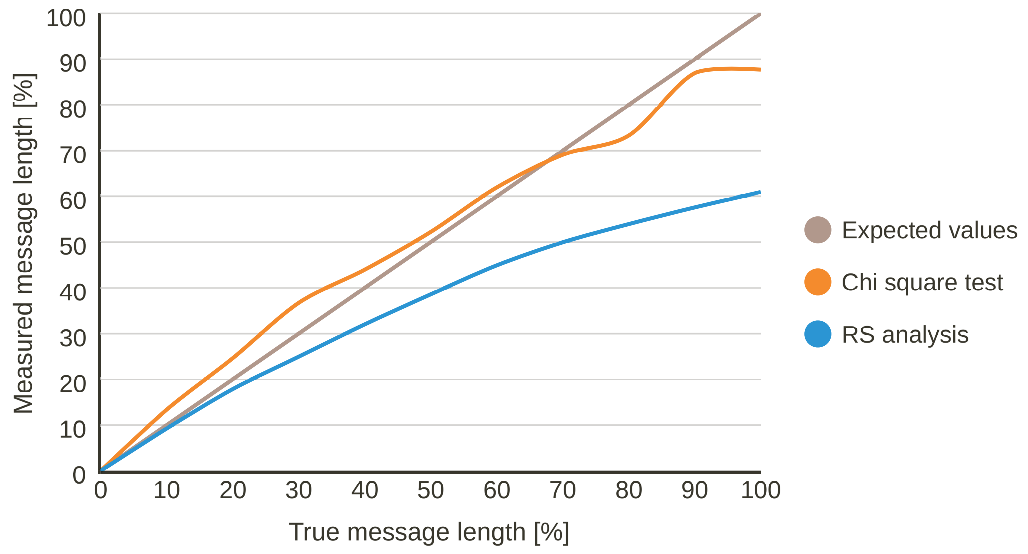


Figure 4.9: Comparison of Chi square and RS analysis according to the detected message size

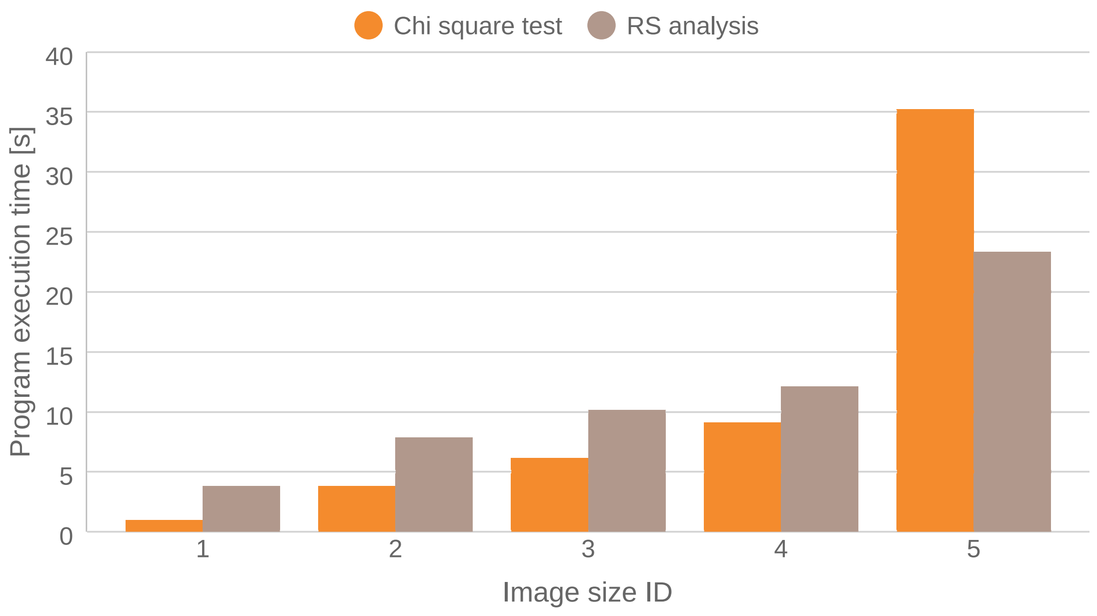
In an ideal situation results would look like the *Expected values* gray straight line. Deviation between the two mentioned methods is not negligible at all. In case of sequential embedding, authors of the RS analysis claim that results become less accurate when the length of the message becomes comparable with the number of pixels in the image [7]. Figure 4.9 also proves this assumption.

Another aspect which needs to be examined is the program execution time. It solely depends on the particular image size. For this purpose, 5 different sizes were put to use ranging from 400x300 to 4032x3024.

Image size ID	1	2	3	4	5
Image size	400x300	1200x900	1600x1200	2000x1500	4032x3024
Chi square test	0.97s	3.82s	6.15s	9.1s	35.23s
RS analysis	3.82s	7.88s	10.16s	12.14s	23.34s

Table 4.2: Averaged system execution time

The following figure gives a more comprehensive overview about the execution time. Y-axis represents the amount of time(in seconds) needed to analyse an image with a particular size, furthermore x-axis shows the ID of the image size to which the value reffers to. Image size ID can be found in table 4.2.



An interesting phenomenon on which the author of the thesis would like to draw attention is that on a theoretically small image Chi square test significantly performs better than RS analysis, however, after a certain image size the difference between them sharply drops off and the RS analysis becomes more powerful in terms of speed. As it was mentioned in 2.6.1.3, RS analysis needs a certain amount of pixels to approximately determine the number of regular and singular groups. If these numbers are below a certain threshold limit, final results might become inaccurate. In case of smaller images to avoid this unpleasant situation, relatively more pixels are being processed and therefore it slows the system to some extent.

Chapter 5

Conclusion

In the last few decades, using digital images as cover mediums in a steganographic communication has gained a wider audience due to the suspicion that the technology may have been used for malicious purposes. Given this fact, detecting hidden communication has become a major issue. One of the objectives of the work was to highlight the need for progress in stegoanalysis.

The aim of this thesis was to introduce the theoretical and technical background of both dital image steganography and digital image steganalysis. After the research of already existing solutions and available technologies, two powerful methods have been implemented, including Chi square test and RS analysis. We also reviewed a number of different steganographic techniques which have been proposed recently, like different variations of LSB. Besides that, we successfully identified some of the requirements of a good steganographic and steganalytic algorithm. At the beginning LSB seemed to be unbreakable but as we moved forward in understanding its overall structure most of these algorithms have been successfully steganalysed.

In terms of steganalysis, the thesis mainly dealt with one of the basic approaches, targeted steganalysis. Since given an unknown image we may not know the embedding method being used. Taking into consideration the fact that more and more steganographic methods are being proposed day by day, targeted steganalysis might become less reliable in the future and it will show a need for blind steganalysis. Specific detection methods will always provide more accurate results, nevertheless, universal detection is more flexible because it can adjust to unknown steganographic algorithms too. From this point of view, the battle between steganography and steganalysis is unlikely to end in the near future. A possible extension of this work from theoretical and practical perspectives would consist of making a research of already existing universal steganalytic methods.

Bibliography

[1] Adnan Gutub, A.; Abdulaziz Tabakh: *Triple-A: Secure RGB Image Steganography Based on Randomization.* [Online; Accessed 01.04.2019].
Retrieved from: https://www.researchgate.net/publication/224503189

[2] Ahmed Hussain Ali, M.; Loay E. George: *A Review on Audio Steganography Techniques.* [Online; Accessed 14.11.2018].
Retrieved from: https://www.researchgate.net/publication/292361595

[3] Avadhesh KUMAR Gupta, V.; Dr. Munesh Trivedi: *Analysis of different text steganography techniques: A survey.* [Online; Accessed 13.11.2018].
Retrieved from: https://www.researchgate.net/publication/306301164

[4] Bartosz Jankowski, W.; Krzysztof Szczypiorski: *PadSteg: introducing inter-protocol steganography.* [Online; Accessed 16.11.2018].
Retrieved from:
https://link.springer.com/content/pdf/10.1007%2Fs11235-011-9616-z.pdf

[5] Christy A. Stanle: *Pairs of Values and the Chi-squared Attack.*
Retrieved from:
https://orion.math.iastate.edu/dept/thesisarchive/MSCC/CStanleyMSSS05.pdf

[6] Dr. Ahmad T. Al-Taani; Zaid Al-Omari: *A Survey on Digital Image Steganography.* [Online; Accessed 20.10.2018].
Retrieved from: https://www.researchgate.net/publication/278817774

[7] Jessica Fridrich, M.; Rui Du: *Detecting LSB Steganography in Color and Gray-Scale Images.* [Online; Accessed 05.01.2019].
Retrieved from: https://pdfs.semanticscholar.org/c0f2/84378f906cb4531ee1d2047895be376486f5.pdf

[8] Kanchan Patil, R.; Gajendra Singh: *Digital Image Steganalysis Schemes for Breaking Steganography.* [Online; Accessed 20.12.2018].
Retrieved from: https://users.cs.fiu.edu/~fortega/df/research/Stenography%20II/Statistical%20Steganalysis.pdf

[9] Kwangsoo Lee, A.; Sangjin Lee: *Category Attack for LSB Steganalysis of JPEG Images.* [Online; Accessed 03.01.2019].
Retrieved from: https://pdfs.semanticscholar.org/5e70/0c8bb6f1914949e6e61734f5879492641973.pdf

[10] Madhavi B. Desai; Dr. S.V. Patel: *Survey on digital image steganalysis.* [Online; Accessed 28.02.2019].

Retrieved from: https://pdfs.semanticscholar.org/9846/7716b084abb843305b5a6a1726d23bef9467.pdf

[11] Media4x: *jpg2png online tool for converting images to PNG format.* Retrieved from: https://jpg2png.com/

[12] Mehdi Kharrazi, H.; Nasir Memon: *Image steganography: Concepts and Practice.* [Online; Accessed 04.11.2018]. Retrieved from: http://sharif.edu/~kharrazi/pubs/ims04.pdf

[13] Miroslav Goljan; Jessica Fridrich: *Practical Steganalysis of Digital Images – State of the Art.* [Online; Accessed 05.01.2019]. Retrieved from: https://pdfs.semanticscholar.org/ccd6/0437d8577e190d879a7de22bd156b3a80ffd.pdf

[14] Rainer Böhme: *Principles of Modern Steganography and Steganalysis.* [Online; Accessed 12.11.2018]. Retrieved from: https://www.researchgate.net/publication/251223942

[15] Raja Sekhar Krovi, G.; Anita Pradhan: *Digital image steganography using LSB substitution, PVD, and EMD.* [Online; Accessed 02.04.2019]. Retrieved from: https://www.researchgate.net/publication/319644580

[16] Rocha, A.: *Steganography and Steganalysis in Digital Multimedia: Hype or Hallelujah?* [Online; Accessed 19.10.2018]. Retrieved from: https://pdfs.semanticscholar.org/f60e/d5f0f609956d0715659c823506feea036a31.pdf

[17] Souvik Bhattacharyya, I.; Gautam Sanyal: *A Survey of Steganography and Steganalysis Technique in Image, Text, Audio and Video as Cover Carrier.* [Online; Accessed 01.12.2018]. Retrieved from: http://www.rroij.com/open-access/a-survey-of-steganography-and-steganalysis-technique-in-image-text-audio-and-video-as-cover-carrier-1-16.pdf

[18] Souvik Bhattacharyya, I.; Gautam Sanyal: *Statistical steganalysis.* [Online; Accessed 17.12.2018]. Retrieved from: https://users.cs.fiu.edu/~fortega/df/research/Stenography%20II/Statistical%20Steganalysis.pdf

[19] Thomas Boutell: *PNG (Portable Network Graphics) Specification.* [Online; Accessed 24.02.2019]. Retrieved from: https://tools.ietf.org/html/rfc2083

[20] Wen Chen: *Study of steganalysis methods.* [Online; Accessed 18.01.2019]. Retrieved from: http://archives.njit.edu/vol01/etd/2000s/2005/njit-etd2005-006/njit-etd2005-006.pdf

Appendix A

DVD content

Programs which have been written in Python3.6 require additional modules including *PIL* 1.1.7 and *scipy* 0.13.1. The attached DVD contains the following:

- "*Bachelor's Thesis - Text*" directory described in Section A.1,
- "*Bachelor's Thesis - Source Files*" directory described in Section A.2.

A.1 Bachelor's Thesis - Text

This folder contains source codes required to create the final PDF file and includes the following directories:

- "*Text*" - source codes in latex,
- "*PDF*" - PDF version of the thesis.

A.2 Bachelor's Thesis - Source Files

An example of executing steganography.py:

```
python3.6 steganography.py -i in.png -o out.txt -r leftToRight -m decode
```

• `-i INPUT, --input`	path to the input image file
• `-o OUTPUT, --output`	path to the output image file
• `-r ROUTE, --route`	algorithm start point
• `-m MODE, --mode`	encode or decode mode
• `-d MESSAGE, --message`	message or path to a valid text file
• `-h`	describes the usage of the program

Input and output image file must contain a valid path to the PNG image file. Route describes the location of the image where the algorithm starts to embed into the image, including possible options: leftToRight, rightToLeft, topToBottom and bottomToTop. Message can be either a string containing the hidden message or a path to a valid text which is readable.

An example of executing steganalysis.py:

```
python3.6 steganalysis.py -i in.png -o out.txt -a chisquare
```

• `-i INPUT, --input`	path to the input image file
• `-o OUTPUT, --output`	path to the output text file
• `-a ALGORITHM, --algorithm`	algorithm type (RSanalysis or chisquare)
• `-v, --verbose`	additional informations
• `-p, --plot`	results in form of a plot
• `-h`	usage of the program

Input image file must contain a valid path to the PNG image file. If the output text file is not specified, the standard output is going to be used. Additional information related to the results can be acquired by the `-v` optional argument. The `-p` optional argument creates a plot that helps to identify the altered area within the image. Note that it is only available for the Chi square algorithm. In case of using Chi square algorithm it either prints out that the image is a cover or a stego with some probability of embedding. However, in case of using RS analysis it results the percentage of flipped pixels. Analysing an RGB image requires more time due to it contains 3 channels.

An example of executing test.sh:

```
./test.sh input_folder algorithm output_file
```

• `input_folder`	path to a directory containing PNG images
• `algorithm`	algorithm type (RSanalysis or chisquare)
• `output_file`	path where results will be stored

Note that the above mentioned script requires `steganalysis.py` to be available in the same directory.