NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

MASTEROPPGAVE

Kandidatens navn: Andreas Grytting Furuseth

Fag: Datateknikk

Oppgavens tittel (norsk):

Oppgavens tittel (engelsk): *Digital Forensics*: Methods and tools for retrieval and analysis of security credentials and hidden data.

Oppgavens tekst:

The student is to study methods and tools for retrieval and analysis of security credentials and hidden data from different media (including hard disks, smart cards, and network traffic). The student will perform practical experiments using forensic tools and attempt to identify possibilities for automating evidence analysis with respect to e.g. passwords, cryptographic keys, encrypted data, steganography, etc. The student will also study the correlation of identified security credentials from different media in order to improve the possibility of successful identification and decryption of encrypted data.

Oppgaven gitt:	28. Januar 2005
Besvarelsen leveres innen:	1. August 2005
Besvarelsen levert:	15. Juli 2005
Utført ved:	Institutt for datateknikk og informasjonsvitenskap Q2S - Centre for Quantifiable Quality of Service in Communication Systems
Veileder:	André Årnes

Trondheim, 15. Juli 2005

Torbjørn Skramstad
Faglærer

Abstract

Steganography is about information hiding; to communicate securely or conceal the existence of certain data. The possibilities provided by steganography are appealing to criminals and law enforcement need to be up-to-date. This master thesis provides investigators with insight into methods and tools for steganography.

Steganalysis is the process of detecting messages hidden using steganography and is examined together with methods to detect steganography usage.

This report proposes digital forensic methods for retrieval and analysis of steganography during a digital investigation. The result is the following list of methods to defeat steganography:

- Physical crime scene investigation
- Steganalysis
- Detection of steganography software
- Traces of steganography software
- Locating pairs of carrier/stego-files
- Key word search and activity monitoring
- Suspect's computer knowledge
- Unlikely files
- Locating steganography keys
- Hidden storage locations

These proposed methods are examined using scenarios. From the examination of steganography and these cases, it is concluded that the recommended methods can be automated and increase the chances for an investigator to detect steganography.

Preface

This thesis is a result from work done as the part of a master's degree from the Department of Computer and Information Science, NTNU. The title of the project is "Digital Forensics: Methods and tools for retrieval and analysis of security credentials and hidden data" and has been proposed by André Årnes from Q2S - Centre for Quantifiable Quality of Service in Communication Systems. Årnes also did the supervising of this master thesis.

The focus of the project is on studying *steganography* in the context of digital forensics. The thesis covers tools and methods for steganography and how these can be detected.

The objective is to create digital forensic methods to detect steganography, which can be used in the process of a digital investigation.

I would like to thank everybody who supported me and inspired me to work on this project, especially André Årnes and Professor Torbjørn Skramstad for guidance and motivation.

Trondheim, July 15, 2005

Andreas Grytting Furuseth

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

This chapter contains a short motivation for this master thesis, an introduction to the important concepts of digital forensic and steganography, as well a presentation of the project scope. Finally an overview of this document is given.

1.1 Motivation

A trend in today's society is an increasing amount of assets existing only in the virtual world. Where there is something of value, there is also potentially crime.

Forensics is often though of as a crime scene, where police investigators are methodically collecting possible evidence for further analysis. The process of gathering evidence, interrogation of witnesses, identifying a suspect and building a case against this possible perpetrator is old as the introduction of a legal system with legislative, judicial and executive powers.

With new technology, both legal and illegal processes evolves. Computer crime and the methods to counter and investigate it are in an arms race. Old legislations are changed or adapted to answer to these new cases of digital investigations.

The keeping of secrets is an old doing. There are basically two ways of keeping something a secret. One is to hide an object, hoping that nobody finds this hiding place. The second way is to store the secret in a way that is only accessible to some, e.g. a safe. An example of the later from the digital world, secrecy can be achieved with the usage of cryptography. Cryptography is relatively mature and well known.

In some cases, there is a need of keeping the presence of a secret hidden. The rumor of a hidden treasure will for sure bring numerous treasure hunters. In the same way as the presence of cryptography can raise unwanted attention or suspicion.

Returning to the two ways of keeping a secret from above, the first one is called *steganography*. Steganography is about keeping something hidden, lit-

erally meaning "covered writing". The usage of steganography goes back in history to the ancient Greece. Examples of its use from World War II are secret ink and microdots[1]. In the digital world, steganography is hiding data inside other data.

Steganography can be a mean for criminals (and others) to hide information in the digital world. Some indications exists that this is the case, but no clear statistics exists. Investigators refusing to focus on the possibilities given by steganography, yields "security through denial" and is not a good alternative.

This master thesis looks into methods and tools for steganography, as well as how to deal with steganography in digital forensics.

1.2 Introduction to digital forensics

Digital forensics is about taking the forensic experiences and methods of the physical world to the virtual one. Acquisition of evidence, analyzing it and presenting findings is also needed when investigating a crime where a computer is involved, though the specific techniques used are quite different from more traditional forensics.

The term digital forensics comprises a wide range of computer activity. Not just evidence from computers (i.e. disk drive and computer memory), but including all sorts of generic digital media, including cell phones, memory sticks, PDAs , network traffic etc.

The methodologies from physical forensics are adapted into digital forensic, specific forensic software are created and comprehensive knowledge is obtained by digital forensic specialist to defeat digital criminality.

1.3 Introduction to steganography

Steganography is about covert communication; to hide the existence of a message from a third party [25]. The word steganography is derived from the Greek words *steganos* translating to covered and *graphein* meaning writing [web15], translating steganography to covered writing. Steganography consists of a variety of techniques, and all are not directly linked to the computer. Microdots and tattooed messages on human heads covered with hair regrowth [web53], to mention some.

In computer science, steganography is hiding data within nonsecret data, e.g. a data file of some sort. Steganography is based on the fact that data files can be slightly altered without losing its original functionality and human senses are not sensitive enough to discover the small changes in the altered files. These principles are illustrated in Figure 1.1.

[1] Microdots are text or images reduced in size so that they are not discovered by unintended recipients [web50].

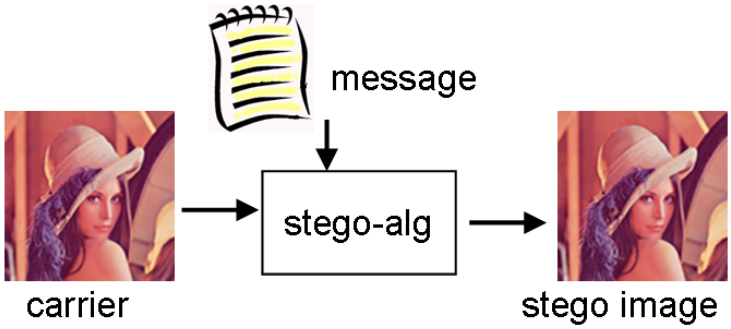


Figure 1.1: Simple presentation of the principle of steganography.

Figure 1.1 shows an example of steganography. A suitable image, called the carrier, is chosen. The secret message is then embedded into the cover using the steganographic algorithm, in a way that does not change the original image in a human perceptible way. The result is a new image, the stego-image, that is not visible different from the original. From an observer's view, the existence of a secret message is (visibly) hidden.

1.3.1 The use for steganography

Cryptography transform structured and intelligible data, like a text file, into a stream of random-looking data [52]. All digital data are ordered in well defined structures, like protocols, file types, hierarchical models etc. Hence, there basically exist no random data. Except encrypted data, which will stand out among other types of data. The purpose of steganography can be said to be the opposite of cryptography [web17]: "to mix random-looking data with decoy information", where *mix* is the steganography algorithm, *data* is the message to be embedded and *decoy information* is the carrier.

1.4 Interpretation of scope

Steganography is data hiding, hence methods and tools for retrieval and analysis of steganography are studied in this master thesis. Security credentials are then comprehended as passwords and keys used with steganography software.

Identification of data is the discovery of hidden data, i.e. the detection of steganography. The thesis description also includes "decryption of data". The related action in steganography is data extraction, i.e. to obtain the embedded message. Detection of steganography and message extraction is targeted in this master thesis, with the objective of automating evidence analysis with focus on steganography.

Digital forensics is well documented, in contrast to the digital forensic aspects of steganography. This thesis is not a review of digital forensics, but is limited to target the particular effects steganography has on digital forensics.

1.5 Document organization

The organization of this document is as follows. There is an introduction to the concept of digital forensics in Chapter 2. A methodology is presented, which later is applied to example scenarios. Some tools aiding the investigation of digital evidence is also presented.

Subsequent, in Chapter 3, the reader is introduced to steganography. The terminology used when describing steganography algorithms is presented, followed with a classification of information hiding techniques. A formal method for modeling steganography is treated, together with different methods for steganography. After the introduction to steganography, different steganography tools are studied by the author in Chapter 4.

Steganalysis is then introduced (Chapter 5). It contains a scenario, called "The Prisoner's Problem", describing the context of steganalysis and presents different methods for steganalysis. This is followed by a study by the author of tools for steganalysis (Chapter 6).

Digital forensics and steganography are brought together in Chapter 7. New and old forensic methods to defeat steganography are proposed by the author, including steganalysis, identification of steganography software and security credentials.

Then digital forensic scenarios are presented in Chapter 8. The forensic methods described earlier are used, treating cases dealing with steganography. Tools presented in the thesis are used in the scenarios.

Results from the cases and studies of tools are summarized and discussed in Chapter 9, with focus on the forensic aspects of steganography and steganalysis.

The conclusion is found in Chapter 10, with some outlines for future work on the subject.

Appendix A provides a database of hash signatures used to identify steganography tools treated in this master thesis.

Chapter 2

Digital Forensics

This chapter introduces the area of digital forensics. First, a definition of the term digital forensics is presented, followed by a section describing digital forensics in the context of general forensics. A methodology for digital forensics is then presented, which is used with the digital forensic cases in Chapter 8. Finally digital forensic software is treated.

2.1 Forensic Science

Forensic science, or just forensics, is defined in [web1] to be: "the application of science to law. Forensic science uses highly developed technologies to uncover scientific evidence in a variety of fields". The history of using scientific methods to identify and prosecute criminals is old. It dates back to the 12th century, when King Richard I established the "Office of the Coroner" [web1]. This led to the application of medical science to law. Forensic science covers a wide range of scientific methods. Well-known techniques include fingerprints and DNA analysis. These methods are results from scientific work, adapted to the court of law. In the same way as fingerprints met skepticism on their uniqueness and usability in court of law, investigation methods from the digital world suffer the same struggle to become accepted as evidence in the court of law.

Information is stored and processed using computers, some of the information only existing in the virtual world. This information is often important, and where there is something of value, there is criminality. The computer can be the tool aiding the criminal or the sole scene where the illegal act takes place.

In the same way as humans leave marks, like fingerprints and DNA-samples in the real world, actions leave traces on the computer. *Digital forensics* is about investigating crimes with the possibility of existing digital evidence. Examples of this are log files on a compromised (hacked) web server or the confiscated hard disk from a suspected child pornography possessor.

Analogous to burglars using gloves to prevent leaving fingerprints on the crime scene, criminals use various methods and tools to prevent detection. Data and

communication can be encrypted or (attempted) deleted; persons can operate behind anonymous services or, as in this master thesis, communication can be covert.

The forensic aspect of steganography is twofold, hidden data storage and hidden communication. Data, say child pornography, can be hidden inside innocent looking images or from surveillance of communication between criminals and their unknown associates. With the proper knowledge and tools, the investigator can try to break the steganography algorithms used, i.e. detect the presence of hidden data.

2.2 Digital Forensic

A definition of digital forensic science often referred to in the literature is the following [33]:

> the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation, and presentation of digital evidence derived from digital sources for the purpose of facilitation or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.

There exist other definitions of digital forensics, and other related expressions. An overview of different classifications of digital forensic related terms are presented in [14]. In this master thesis, *digital forensics* is used for its comprising of the terms computer, internet and network forensics. *Digital crime* is preferred for the same reason, comprising computer, cyber-, electronic crime etc. This follows the convention from [14].

The process of dealing with digital crime has to follow principles and methodology allowing evidence to be accepted in court. Section 2.3 presents the method used in this master thesis. Digital forensics encounters different technologies and fields from digital science, like disk drives, cell phones etc., various low-level file systems and network traffic, and a range of different software. All these areas call for special knowledge, and expert witnesses can be required to testify about the results.

2.3 Forensic methodology

A digital forensics methodology is wanted. Avoiding ad hoc approaches will add effectiveness and integrity to the results from forensic analysis. Truthfulness of the resulting evidence is vital when presented in court.

RFC 3227 [web5] presents "guidelines on the collection and archiving of evidence". This is only a small part of the total digital forensics process. The acquired data are typically raw format and difficult to comprehend, called the Complexity Problem [7]. To cope with this, layers of (data-) abstraction are

used. Similar abstractions can be done regarding tools, strategies and incidents, leading to a general methodology for digital forensics.

There exist different efforts to define methodologies for digital forensics. A simple model, presented by [27], called the *three As*: Acquisition, Authentication and Analysis. A more complement model of the forensic process is wanted. The Cyber Tools On-Line Search for Evidence (CTOSE) project aims to develop such a methodology [9]. The First digital forensic reseach workshop (DRFWS) [33] created a fundamental process model, recommending further research. [44] analyses four models and present their own, trying to address shortcomings of previous models. A formal model for analyzing and constructing forensic procedures is presented in [28].

Setting a side the slightly different terminology and phases in the methodologies from the literature (some listed above), consensus on a common set is within reach. Lacking this universal agreement, a choice has to be made. This master thesis has found the methodology from [6] to best suit its needs, described next.

2.3.1 An integrated digital investigation process

[6] defines a digital investigation methodology with close connections to the physical investigation. It reuses existing theory from physical crime investigations, and is not depending on specific products and procedures.

2.3.1.1 Digital crime scene

An important concept introduced is the digital crime scene. An example from [6] illustrates the concept. A physical crime scene might consist of traces of blood or fingerprints. These evidences are processed to show identity information. A computer is also physical evidence, which can be processed to yield valuable digital evidence. The computer is treated as a secondary crime scene: "the computer is a door that leads the investigators to a new room".

2.3.1.2 Process organisation

The process model is divided into 17 phases, spread over five groups, as shown in Figure 2.1 and described below. All phases are not explained in detail, hence the reader is referenced to [6] for more details.

The following descriptions of the different groups of phases are based on [6], with some additional pointers:

Readiness phases Deals with preparations to ensure functional operations and infrastructure prior to incidents. For digital forensic, this includes practicing with different tools, certifications and staying up to speed with changes in cyberspace criminality, like possible increased usage of steganography.

Deployment phases Provides mechanisms to detected and conform incidents, including authorization like search warrants.

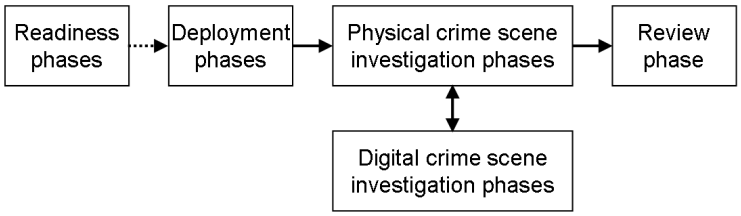


Figure 2.1: The five groups of the investigation process, with their phases. Adapted from [6].

Physical crime scene investigation phases In this phase physical evidence are collected and analyzed.

Digital crime scene investigation phases Closely linked with the physical crime scene phase, these phases treat the physical computer as a crime scene and searches it for evidence. The results are given back to the physical crime scene investigation phases. as shown in Figure 2.2. This group is treated more in Section 2.3.1.3.

Review phase Evaluation of the investigation to identify areas of improvements. This can result in the need for more training or tool improvement to better deal with the evolving criminal activities.

2.3.1.3 Digital crime scene investigation phases

There are 6 phases of the digital crime scene investigation grouping, as shown in Figure 2.2. They closely follows the steps from the physical one. Each of them are described below[1].

The different phases of the digital scene are:

Preservation of digital scene Involves securing and preserving the digital scene, including volatile data if possible. Tools are used to create identical images of data for further investigation later.

Survey for digital evidence Finding the obvious pieces of evidence. This is based in the information from the Deployment phase, i.e. the nature of the case. For instance, if the case is child pornography, images would be collected and illicit ones used as evidence.

Document evidence and scene Documenting the evidence from the previous phase, according to the abstraction layer [7] of the evidence. Example of abstraction layer is when viewing the file raw, shows ones and zeros. With the ASCII layer of abstraction, the numerical values are mapped to characters. If the file is a HTML document, the HTML layer of abstraction treats the evidence as viewed in a web browser [7]. In the case of information hiding in HTML comments, the HTML layer of abstraction misses this. Providing cryptographic hash values to prove integrity, where MD5 and/or SHA-1 is normally used.

[1] All explanations are adapted from [6], again with additional pointers.

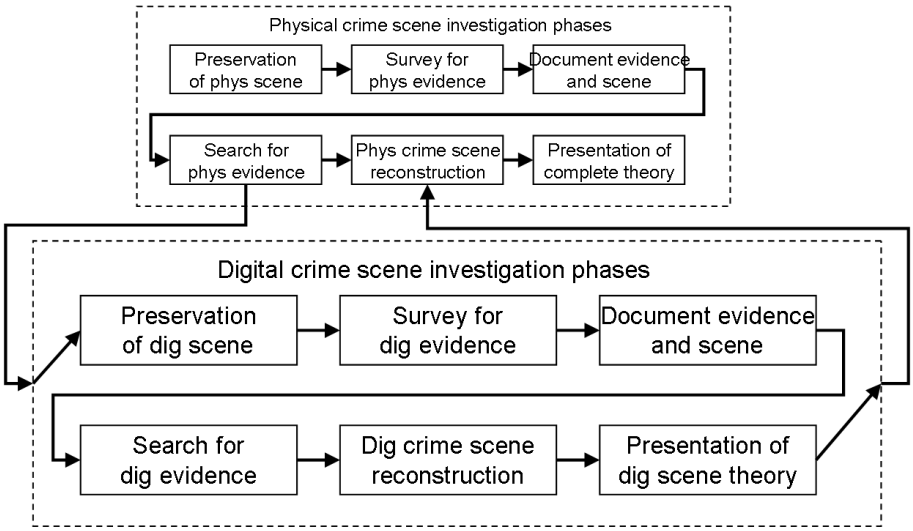


Figure 2.2: Digital crime scene investigation phases, also showing the connections to the Physical crime scene investigation phases. Adapted from [6].

Search for digital evidence A more thorough analysis of the digital scene. The results from the survey phase shows which additional analysis to focus on. E.g if the pornographic images suspected, but not found, during the survey, searches after steganography software, hash values or hidden data. More on this in the chapters on steganography (Chapter 3) and steganalysis (Chapter 5) and in Chapter 7, proposing forensic methods to defeat steganography.

Digital crime scene reconstruction Putting the pieces together, testing and rejecting/accepting theories. A good reconstruction might be visualized and used in the presentation of the digital crime scene theory [54].

Presentation of dig scene theory Presenting the digital evidence found, feeding the results back to the physical crime scene investigation.

2.3.2 Chain of Custody and Integrity documentation

Documentation is essential to the investigation. To quote [27]:

> The key to any investigation, particularly a computer crime investigation, is documentation.

For evidence to be reliable in court, integrity has to be preserved. Safe storage and tamper protection is needed, so is also the documenting of handling, i.e. who has accessed the evidence while it was in custody. Chain of custody prevents accusation in court that the evidence has been tempered with.

Evidence need to be identified and labeled as soon as it is collected. All actions performed by the investigator should be documented, including the reasons for doing so. In digital forensics, this means logging all actions and integrity checks.

A hash value comparison of the original evidence and the working copy, can yield the copy to be identical to the original.

2.4 Digital forensic tools

There exists a variety of tools that can be used during a digital investigation, some are specialized toward forensics. The different phases from the digital forensic methodology present in Section 2.3, need different hardware and software tools providing the investigator means to collect and analyse.

2.4.1 Acquisition tools

Tools that can help in viewing the digital scene and collecting possible digital evidence are needed. They range from expensive commercial applications targeting digital forensics, to free, common available programs.

EnCase [web44] from Guidance Software is a Windows-based comprehensive and complete forensic application[2]. A trial version of Encase Version 4 was obtained for this master thesis. The trail version is restricted to the included evidence file (i.e. disk image), so there is a limited usability when specializing on steganography. It has means to collect identical disk images, but following the tradition from the academic society of using open-source and its limited dataset for testing, EnCase is not used.

dd is a standard tool following *-nix distributions. It can access block devices directly, allowing the creating of byte-exact copies of entire disk driver or partitions. There exist an extended version of dd called *dcfldd* [web18], which allows taking the hash value while copying data and a convenient progress bar[3].

Other tools creating bitwise identical copies can be used for the acquisition of data. In some cases, like RAM on locked live systems and cell phones, special hardware is required to collect data.

2.4.2 Documenting evidence

In digital forensics, hash functions are used to document evidence. The authenticity of evidence presented to the court is critical, and it is necessary to trust the hash value to uniquely identify digital evidence [53]. Currently available forensic software like Encase and The Sleuth Kit (see below) are using MD5 [web41] or SHA1 [web12]. National Software Reference Library (NSRL) [web31] distributes its signatures as MD5 and SHA1. Tools creating these signatures are available on both *-nix and Windows. *md5deep*[4] is a cross-platform set of programs to compute MD5, SHA-1 and other hash values. md5deep allows for recursive operation, in contrast to the original *MD5sum* and *SHA1sum* tools.

[2] See `www.encase.com` for more information on EnCase.

[3] Transferring all data from a disk drive can take several hours, as will be seen in Chapter 8, where the dd tool is used.

[4] md5deep.sourceforge.net/

The signatures of steganography software found in Appendix A are created with a small Java program[5], which creates output as both text files with an identical syntax as the md5sum, sha1sum and md5deep tools, and tables to be used with LaTeX.

2.4.2.1 Notes on MD5 and Sha1-collision

There is discovered a collision weakness with the MD5 algorithm [53, 56] and with SHA1 [57]. From a forensic viewpoint, *weak collision resistance* is important. With a message x with hash signature $H(x)$, weak collision resistance states it is computationally infeasible to find a message y, $y \neq x$ and hash signature $H(y)$, with the property $H(y) = H(x)$ [52]. I.e. to find a second message with the same signature as the first message. *Strong collision resistance* states it is computationally infeasible to find any x and y with $H(y) = H(x)$ [52]. If $H(y) = H(x)$ occurs, it is called a collision. The attack on MD5 and SHA1 attacks the strong collision resistance of the algorithms.

The consequences of the detected collisions are different for cryptographers and digital forensics. It is still computationally infeasible to modify the contents of a message, such that the new message matches a pre-determined hash value, collisions in the National Software Reference Library (NSRL) data set are not likely and small changes to the evidence will change the hash value [53]. NSRL have published a note regarding SHA1-collisions [web30] and found it not necessary to change their hash algorithms.

Having both MD5 and SHA1 signatures gives additional security. The two signatures are linearly independent, so when used together they give a (128 [MD5] + 160 [SHA1] =) 288 bit signature. It is unlikely for x and y to have both $H_{MD5}(x) = H_{MD5}(y)$ and $H_{SHA1}(x) = H_{SHA1}(y)$. Also taking advantages of compatibility[6], MD5 and SHA1 are the algorithms used in this master thesis.

2.4.3 Analysis tools

There exist various digital forensics tools aiding the investigator. Some, like EnCase [web44], are quite expensive and others, like F.I.R.E (Forensic and Incident Response Environment) [web43], are freely available.

The Coroner's Toolkit (TCT) [web13] is primarily assisting the examination of Unix systems. The Sleuth Kit (TSK) is a collection of command line tools based on TCT. Autopsy is a graphical interface to the command line tools in TSK. TSK runs under Linux, but also supports investigations of FAT32 and NTFS file systems[7].

F.I.R.E is a Linux distribution bootable cd containing useful forensic tools, like Autopsy and The Sleuth Kit [web9]. It also ships with StegDetect for detection

[5] Available on the cd following this master thesis, *md5Sha1.(java|class)*

[6] Datakrimavdelingen at Nye Kripos uses MD5, NSRL [web31] uses MD5 and SHA1 for their database of software signatures and the forensic tool used in this master thesis, Autopsy, uses MD5.

[7] See www.fish.com/tct/ for more information on The Coroner's Toolkit. Autopsy and The Sleuth Kit are found at www.sleuthkit.org/.

of steganographic images. More on StegDetect in Section 6.3. F.I.R.E is freely available, and the containing tools are mostly GNU General Public License (GPL). More on F.I.R.E later, as it will be used in the forensic case in Chapter 8.

A more comprehensive list of forensic tools and toolkits can be found at www.forensics.nl/

2.4.4 Automatic identification of known software and files

The digital crime scene today normally exist of gigabytes of data. A frequently used method to reduce the amount is to automatically remove known files, e.g. remove static operation system files. It can also be useful to identify interesting files, e.g. steganography software. This is done by comparing hash signatures of encountered files with databases of known signatures. The National Institute of Standards and Technology (NIST) maintains a list of digital signatures of software applications called the "National Software Reference Library" (NSRL) [web31]. This list also contains steganography software. There exists a "Steganography Application Fingerprint Database" (SAFDB) [web3][8], claiming to contain signatures for 230 data-hiding applications.

2.4.5 Tool summary

Table 2.1 gives an overview of the tool presented in this chapter.

Name	Description	Reference
Encase	Complete forensic software	[web44]
dd	Acquisition	Unix distro
dcfldd	Acquisition	[web18]
The Coroner's Toolkit	Forensic toolkit	[web13]
Autopsy / The Sleuth Kit	Forensic toolkit	[web9]
F.I.R.E	Forensic distro	[web43]
md5sum	Documenting	Unix disto
sha1sum	Documenting	Unix distro
National Software Reference Library	Signature database	[web31]
Steganography Application Fingerprint Database	Signature database	[web3]

Table 2.1: Digital forensic tools

[8]This list is [web3]:"... free to qualifying US law enforcement, military, government, and intelligence agencies". After inquires to the company, it was not possible to obtain (a subset of) this database.

Chapter 3

Steganography

This chapter gives an introduction to the concept of *steganography*. A presentation based on this chapter was presented at the "Forskningsmessige utfordringer innen dataetterforskning og elektroniske spor" conference at *Nye Kripos* [18]. First, a short repetition of the introduction to steganography from Section 1.3 is given. The terminology used in the literature is then presented, together with a study connecting steganography with closely related concepts like cryptography and digital watermarking.

Some thoughts and sources speculating on the usage of steganography is presented, without proving the existence of steganography in the wild. A classification of information hiding follows, succeeded by an attempt to formalize steganography. A section on different methods for message embedding serves as a transition to the following chapter treating steganography software.

3.1 Introduction to steganography

An analogy to steganography from the legendary book "The Codebreakers" by David Kahn[22] is criminal behavior, where plotters attempt to do things in a secret way and not an overt way[1]. An example of steganography on digital media is given in Figure 3.1. The motive of the used image is of no importance, it serves only as a carrier for the hidden message.

Shannon stated in his legendary paper from 1949 that [50]: "concealment system[2] are primarily a psychological problem". The material presented in this master thesis shows this is not the case. There exist known steganographic systems, which require sophisticated steganalysis methods to be detect. More in this later, first the terminology of steganography will be addressed.

[1] David Kahn also presents the history of steganography specially at "Information Hiding, the First international workshop" [23].

[2] Shannon defined concealment system as : "methods in which the existence of the message is concealed from the enemy" [50], in other words steganography.

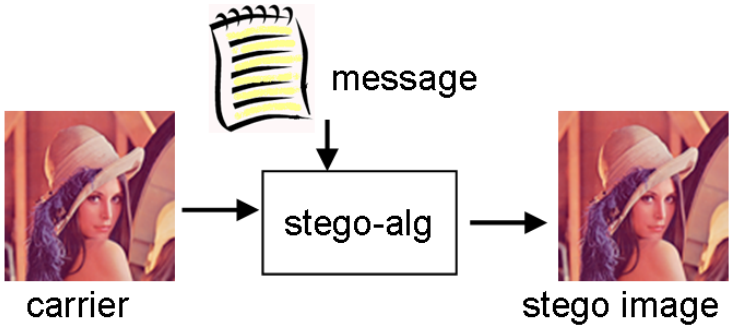


Figure 3.1: Example of steganography. A secret message is embedded into an innocent looking image. The embedding, i.e. steganography algorithm, tries to preserve the perceptive properties of the original image.

3.2 Terminology

In computer science, steganography is hiding secret data within nonsecret data, e.g. a data file of some sort. Steganography is based on the fact that data files can be slightly altered without losing its original functionality and human senses are not sensitive enough to discover the changes in the altered files. This can be stated with a simple equation (Equation 3.1 [35]). A measurement of the human imperceptibility threshold for a media, say an image, is assumed. Let t be the portion of the image that can be manipulated without causing perceptible changes to the image, and p the part yielding perceptible changes if manipulated. A possible carrier, C, for a hidden message can then be presented as in Equation 3.1.

$$C = p + t \tag{3.1}$$

It is assumed that both user and attacker of the steganography algorithm knows t. Usage of steganography is not perceived by human senses, since there exists a t' where $C' = p + t'$ with no perceptible differences between C and C'.

A conceptual overview of steganography is shown in Figure 3.2.

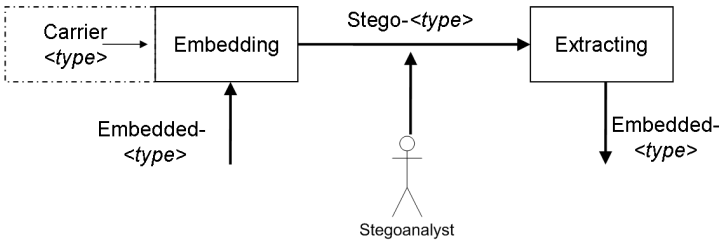


Figure 3.2: Conseptual view of steganography. Adapted from [40]

The hiding data is called the *carrier* [27] or cover message [web53]. Following the naming convention from Figure 3.2, carrier-<*type*> will be used. <*type*> is a

general expression capturing the different media types used with steganography, e.g. image, message, file, signal etc. Today multimedia files, like pictures or sound, are most common [27], but other types of carrier files can and are being used. This will be apparent when different steganography tools are examined in Chapter 4.

The data that is to be kept a secret is called embedded-<*type*> and the process of hiding is called embedding, i.e. running the steganography algorithm. The embedding process results in the stego-<*type*> and the recovering of the embedded-<*type*> is called extracting.

The carrier-<*type*> is of little or no importance, e.g. the picture or theme of the image carries no information. But it will be apparent when steganalysis is addressed in Chapter 5, that specific properties of the carrier-<*type*> is wanted to prevent detection.

In the rest of this master thesis, *message* will be used instead of <*type*> for readability.

3.2.1 Simple steganography

Simple or pure steganography [8] is based on keeping the method for embedding secret. Figure 3.2 shows this.

Simple steganography is however breaking with Kerckhoffs' principle, it is not wise to only rely on the secrecy of the steganography algorithm. Early steganography methods and software belong to this category. Consider the secrecy of secret ink and microdots, when the adversary knows these methods. Later, when different steganography software are treated in Chapter 4, software examples of simple steganography are treated. Section 3.7 mention general methods for embedding.

3.2.2 Secret key steganography

Systems better than *simple steganography* assume that sender and receiver share a secret key [3]. Secret key steganography algorithms us a key to seed a cryptographic keystream generator, which is used to select locations where to embed the secret message. I.e. which pixels or sound samples to alter. Figure 3.3 extends the conceptual view of simple steganography to secret key steganography.

A secret key steganography system is understood such that only the possessors of the secret key can detect the presence of an embedded message. To all other, the extracted message(-s) would be just noise. The key used is called *stego-key*, to distinguish it from cryptography keys.

3.2.3 Public key steganography

With public key cryptography, only the private key can decode the message. Public key steganography is then interpreted such that only the possessor of

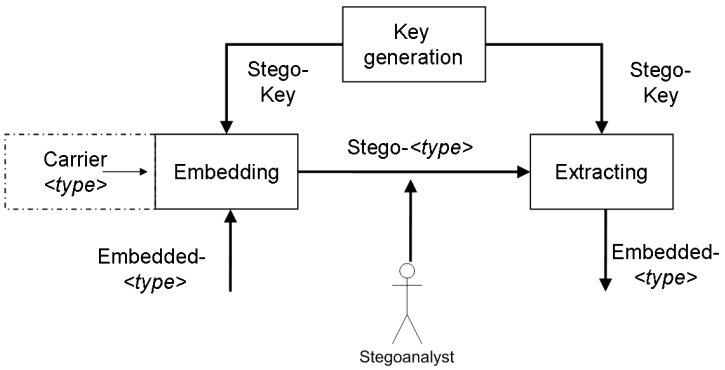


Figure 3.3: Conseptual view of secret key steganography. Adapted from [40]

the private key is able to detect the presence of an embedded message and extract it.

Public key steganography (PKS) is possible, as stated by several [37, 3, 8, 2]. With PKS, the covert message is first encrypted using the public key of the recipient. The embedding of the message, i.e. the steganography algorithm, then alters the parity of bit blocks to encode a pseudorandom bit string, i.e. the covert message. The adversary can not check a cover/stego-message by detecting the presence of a pseudorandom bit string, since a suitable parity check function will yield a pseudorandom looking bit string from all carriers where a message can be embedded [2]. All recipients must then try to extract a message from the received, and only the owner of the correct private key will succeed.

One thought of PKS is that the public key is used for encryption, i.e. creating the pseudorandom bit string. A steganography key is, as known, used to decide which bits to possible alter in the carrier-message. Hence, it could be discussed whether this key is a steganography or cryptography key. The literature is not precise on this and normally just presents it as a public key steganography scheme.

From the explanation of PKS above, parity blocks are altered to embed the message. So when stated as "change the parity to *odd* when embedding '1' and *even* when embedding '0' ", it can be though of as deciding where to add '1'-s and the public/private key pair can be named a public/private steganography key pair.

3.2.4 A formal model of steganography

To better be able to evaluate steganography methods, a more formal model is researched for. In the same way a theory for secrecy systems was defined by Shannon in "Communication Theory of Secrecy Systems" [50], information theory and entropy could be used with steganography. Such a modeling is attempted in [60, 32] and others. Another way of modeling steganograpy is by complexity-theory [19], as used in cryptography. The concept will then be to

define secure steganography system so that a stego-message is computationally impossible to differentiate from a cover-message. This work is not finished and is not targeted by this master thesis.

3.3 Steganography and cryptography

Steganography is, as stated above, about hiding information. Steganography is not to be mistaken for cryptography. They differ where adversaries in the case of cryptography know the existence, but not the content, of a secret message. Cryptography obscures the message to prevent disclosure. Examples of cryptography might be "E0F7E3AC" and of steganography invisible inks. A possible application of steganography is when trying to defeat censorship [22], where an encrypted message most likely would not go through the censor.

However, steganography and cryptography can be used together, illustrated in Figure 3.4.

$$\underline{\textit{Steganography}} \quad + \quad \underline{\textit{Cryptography}}$$
$$\textit{gives}$$
$$\textit{Secret communication} \quad \textit{of} \quad \textit{Secret information}$$

Figure 3.4: Steganography and cryptography. Adapted from [12].

The combination of steganography and cryptography has two positive effects, from the steganographer's point of view. It leads to additional secrecy and the cryptographic functions can distill entropy [web6], making steganalysis harder (See Chapter 5 for more on steganalysis.).

From a forensic view, knowing that communication takes place can be essential. So for an investigator, presence of hidden communication is an important discovery. Steganography can provide traffic flow confidentiality, i.e. concealing source and destination, message length, or frequency of communication [web32]. When the presence of steganography is revealed, its purpose is defeated. Even if the message content is not extracted or deciphered [55].

3.4 Digital watermarking

Digital watermarking is a technique which allows to add copyright notices or other verification messages to digital audio, video, or image signals and documents [web48]. Watermarking is closely connected to steganography, but in the same time somewhat different.

When dealing with steganography, the value or importance of the carrier is insignificant. In some cases, the stego-message does not depend on a carrier at

(a) The orignial "pepper.tif" (b) Watermarked "pepper.tif" (c) The watermark

Figure 3.5: Watermarking an image using AiS Watermark Pictures Protector [web46].

all and is created synthetically. This is a natural understanding of the concept of steganography; the only purpose of the stego-message is to communicate the embedded (hidden) message. The example from Figure 3.1 shows a stego-message, where the overtly message is of no importance.

With watermarking, the carrier is the important signal and the embedded message is just present to give some information about the carrier [20]. The watermark can be considered attributes to the cover. Figure 3.5 shows "pepper.tif" before and after adding a visible watermark.

Hence, steganography and digital watermarking are different by definition. With the latter, the carrier is the object being communicated, steganography communicates the hidden message.

The history of paper watermarking is old, with the oldest instance being from 1292 [35]. Typical applications today are visible logos on images and video and hidden copyright notices.

Consider watermarking used to give proof of ownership of digital media; it is obvious that it should be robust to attempt to remove it. Moreover, even if a hidden watermark is identified, it should be hard to remove. Watermarking can use more perceptible areas of a carrier, due to the reduced requirement to stay hidden. In contrast to steganography, where the algorithm is defeated when the hidden message is discovered.

Digital watermarks are, as stated above, closely related to steganography. The Information Hiding workshops [1, 4, 36, 31, 39] also addresses watermarking. Books also often cover both, e.g [48, 20, 35]. Digital watermarking is not specifically treated in this master thesis. However, due to overlap between watermarking and steganography concepts and techniques, it is nevertheless mentioned.

3.5 Usage of steganography

There exist sources speculating in the different usage of covert communications, i.e. steganography. An article from Newsweek [web29] discussing terrorism and

11th September attacks, quotes Neil F. Johnson: "I'd expect it [steganography] to have been used". There exist more speculation on steganography usage. [12] gives a nice presentation of steganography linked to terrorism in the media, mostly speculations and no evidence. Some examples are [web16, web23, web29]. One article from USA Today speculates :"Hidden in the X-rated pictures on several pornographic Web sites ... lie the encrypted blueprints of the next terrorist attack against the United States" [web23]. [3]

But there exist some concrete examples of usage. During a blackmailing attempt, the perpetrator tried to stay anonymous by using steganography[4] [29]. NTB is reporting of an incident in 2003 [web33], where CIA supposedly canceled 30 flights due to suspected hidden messages[5] of terror attacks in subtitles on the Al Jazeera TV station. Whether or not this was actually terrorist communications, it still can be said that CIA is looking for steganography.

As a response to the news articles above, Niels Provos analyzed two million images from the Internet auction site eBay using Stegdetect[6] and Stegbreak[7], no hidden messages were reported found [41]. Provos and Honeyman also searched one million images from Usenet, with the same empty result. After the publication of [41], Provos detected an image [web39] from an ABC News report covering steganography. But still no strong evidence or indicators of terrorists using steganograhpy.

3.5.1 Steganography encountered in digital forensics

The very nature of steganography is to stay hidden, so it is hard to speculate on it's extent. There are some attempts to get information of steganography encounters from investigators [web4, web24], without achieving publicly available statistics[8]. By ignoring steganography due to lack of statistics is "security through denial"[9] and is really not a good alternative.

It is natural to assume that steganography will or could be used, due to its characteristic of concealment, which should appeal to criminals. Therefore, if criminals are not already using steganography, the future will most likely see adoption of steganography as a tool for cyberspace criminals.

[3] Whether it likely or not that al-Qaeda are hiding messages inside pornography, can of course be discussed. As [web16] puts it: "[likelihood is] roughly the same as their likelihood of hiding them in pig carcasses". It is also interesting to note that Jack Kelly, who wrote many of the articles linking steganography and terrorism and often referred to by others, was caught and fired for making up his stories.

[4] The perpetrator was caught since he used is own PC and a anonymity service that revealed his true identity. And since the police knew where the stego-message was, they could monitor and examine all activity [29].

[5] Dates, flight numbers and geographical coordinates

[6] Stegdetect is treated in Section 6.3

[7] Stegbreak is treated in Section 6.4

[8] The results from [web4] was promised made available by the initiators, but after email enquires the results are still not provided.

[9] The term "security through denial" is from [25]

3.6 Classification of information hiding

The literature does not agree on a classification of different methods of information hiding[10]. Without going into to much detail on differences among papers and authors, below is an overview and explanations of the definitions used in this master thesis. The definitions are based on material from frequently appearing authors from the *International information hiding workshops* [1, 4, 36, 31, 39].

According to [37], information hiding is the top domain containing disciplines of fingerprinting, covert and subliminal channels, cryptography etc. Figure 3.6 shows the different consepts. Next follows an explanation of the different terms in the figure.

$$Information\ hiding \begin{cases} Covert\ channels \\ Steganography\ \{\ Continues\ in\ Figure\ 3.7. \\ Anonymity \\ Copyright\ marking \\ Cryptography \end{cases}$$

Figure 3.6: Information hiding methods. Adapted from [40].

Covert channels are based on the (mis-)use of existing shared resources [40]. In other words "to transfer information with a non-standard method" [45], where the communication goes unnoticed (obscured). Example of covert channels can be: to send information over error messages in operating system call interfaces [40] or eavesdrop on electomagnetic interference from video display units [11].

Copyright marking differs from steganography on two aspects. First, steganography needs to be kept secret, while the presence of copyright marking should certainly be detected. Secondly, robust steganography can be a wanted property, but for copyright marking, it is a necessity.

Anonymity is to avoid identification or traffic analysis by hiding locations or addresses [40].

Steganography is the main concern for this master thesis. When studying the various definitions above and in other literature, it can be understood that there is not always a clear separation of the different methods for information hiding.

Steganography is defined earlier in Section 3.1. In Figure 3.7, *steganography* from Figure 3.6 is further classified, later used to categorize specific tools and methods for steganography.

The term *subliminar channel* is left out from Figure 3.6 and Figure 3.7. Subliminar channel is defined in [47] as "the real message is hidden in the message the observer is observing". Subliminar channel and steganography are in [47] defined in the same way, they only differ in the amount of information exchanged. Therefore, steganography and subliminal channels can and often are used interchangeably. And subliminal channel is therefore left out from the above mentioned figures.

[10] For an example, compare [45] with [37].

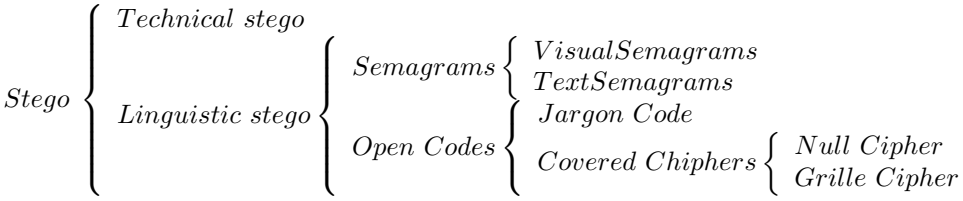


Figure 3.7: Classification of steganography techniques. Adapted from [25]. *Stego* used as abbreviation for steganography to reduse space usage.

The differences between covert channels, subliminal channels and steganography are not easy to identify. According to [web42], subliminal and covert channels are the same ting. In the rest of this master thesis, the definitions mention above will be used. The literature does not uniformly agree on the classification and category definitions for different steganography techniques. Figure 3.7 is frequently appearing and next follows an explanation of the terms in the figure.

Technical steganography uses scientific methods to hide a message [25]. The <*type*> of the carrier is non-text and often a tool, steganographic or photographic, is used in the embedding and extracting of the secret message. Examples are microdots and invisible inks. The steganography software from Chapter 4 are most often technical steganography.

Linguistic steganography uses text as carrier message. And is decomposed further into semagrams and open codes.

Semagrams is to hide the embedded message using symbols or objects, divided into two:

Text Semagrams embeds information by graphically altering the text, i.e. visual text conceals the real message [46]. Examples can be typefaces and spacing.

Visual semagrams use the appearance of physical objects. Examples are the ordering in a deck of cards or the ordering of items on a website [25].

Open codes embeds messages in a legitimate carrier in a way that is not obvious to the observer [25], there is a subliminal channel of information. Open codes is further divided into jargon code and covered ciphers.

Jargon code uses a secret language or phrases expressed in it [web49]. An example is warchalking [11] [25].

Covered ciphers embeds messages openly in the carrier message, so that anyone that knows the procedure can extract the embedded message [25]. Covered ciphers are divided into Grille cipher and Null cipher.

Grille cipher uses a cardboard with holes to extract the hidden message from the stego-message. See Figure 3.8 for a simple example.

[11] Warchalking is the drawing of symbols in public places to advertise an open Wi-Fi wireless network [web54]

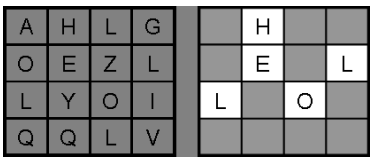


Figure 3.8: Simple example of a grille cipher.

In the grille cipher example in Figure 3.8 the stego-message is "AHLGOEZLLY-OIQQLV" The grille shown in gray to the right of the figure yields the embedded message: "HELLO".

Null cipher hides the message according to some rule like "read every first character of each word". A popular example found repeating in steganographic literature and actually sent by a German Spy in WWII [22] can be seen in Figure 3.9. Spam Mimic (See Section 4.4) generates text with the characteristics of spam.

> Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

Figure 3.9: Example of null cipher. Reading the second letter (in red) of each word, yields the embedded message.

The null cipher example in Figure 3.9 yield the following embedded message when reading the first characters from each word: "PERSHINGSAILSFROM-NYJUNEI". With some added spaces it becomes the real message: "PERSHING SAILS FROM NY JUNE I".

3.7 Different methods for embedding

There exist different steganography algorithms, each using different locations in digital data to hide the secret message. Methods for embedding are treated next. These methods are all examples of technical steganography. The different methods and their explanations are collected from the literature of steganography, listed in the Bibliography. However, other methods might exist which are not documented in the literature. Users of such rely on algorithm secrecy for security. From the forensic viewpoint, there could be homegrown, unpublished steganography algorithms used by criminals. Some of the methods mentioned here, have been successfully broken by steganalysis. These attacks on steganography will be treated in Chapter 5.

3.7.1 Data appending

Data appending is a simple form for steganography. This method relies on secrecy of the algorithm, since it simply embed the message by adding it to the

end of the carrier-file. This works for instance for some image file formats, like JPEG and BMP, since the file header contains a field indicating the total amount of data (BMP) or data after the "End of Image" marker (JPEG). The stego-file is perceptually not different from the cover-file, since most image viewer ignores the additional data.

3.7.2 Adding comments

Many file formats allows for optional comments. Various source codes allow comments to aid the understanding of the code, which are ignored by the interpreters. E.g., HTML files have a comment tag, which is ignored by browsers. These comments can easily be viewed in most browsers, by selecting an option of viewing the HTML source code. Hence, such comment can serve as hiding places for information.

3.7.3 File headers

Various data structures have header information, where some fields in the header are not mandatory or their values are not significant. Such fields can be utilized to communicate covertly. E.g., TCP/IP packets have unused space in the packet headers.

3.7.4 Spatial domain

A spatial domain example is embedding data into the least significant bit (LSB) plane of images. This is based on the assumption that the LSB of the image can be thought of as random noise. The actual embedding takes several approaches: sequential changes, random walk using a pseudo-random generator, parity functions etc.

3.7.5 Transform domain

In the transform domain, the most common embedding method is to utilize the discrete cosine transform (DCT) used with JPEG compression. The embedding is done by altering the DCT coefficients, but with different approaches: LSB changes, different permutations of the coefficients etc. The steganography software F5 [58] and Outguess (Section 4.6) uses embedding in transform domain.

3.7.6 Statistics-aware embedding

It has been noted by several ([17, 26, 59, 20] etc.), that embedding methods alter the statistical properties of the carrier-message. For example with LSB embedding in images, the frequency of colors change. This fact is used in steganalysis and is treated in Section 5.4.2.4. Statistics-aware embedding consider this and use a model of the carrier-message to preserve these characteristics.

Both spatial and transform domain embedding methods are known to preserve statistic properties.

3.7.7 Pseudo-random embedding

Some steganography software falls into the category of secret key steganography. These use a pseudo-random generator, as explained in Section 3.2.2, to select locations for the actual embedding. Both examples of spatial and transform domain methods can use pseudo-random embedding.

3.8 Classification of steganography software

There exist a lot of steganography software. [web3] has identified 230 tools for information hiding, other sources claim there exist not quite that many, but still enough to justify an attempt to classify them. Steganography software can be divided according to their maturity and embedding sophistications, i.e. classified into *generations*. The algorithms can also be classified according to their strength, i.e. they are considered broken, weak, strong or secure. The availability of the software is also interesting information.

3.8.1 Steganography software generations

The earliest steganography software were quite simple. Appending data after the end of images is a simple method for steganography. So is also hiding data in file headers and comments. Examples are adding data in comment fields of JPEG images and between HTML comment tags. These methods do not alter the perceptive properties of the carrier, but are easily detected. The next generation software do the embedding in the least significant bits (LSB). Palette images like GIF and BMP are carriers used for this spatial domain embedding. After successful attacks against LSB methods, steganography algorithms started using frequency domain embedding. Examples from the frequency domain are embedding data with the Discrete cosines transform (DCT) used with JPEG compression and the mp3 encoding of WAV-files.

The frequency domain methods are quite robust against perceptive inspection. However, the embedding introduces statistical changes to the carrier, resulting in stego-messages with different statistical properties then cover-messages. To prevent statistical attacks, the last generation of steganography algorithms preserves the original statistical properties of the carrier.

There exists steganography software bothdoing spatial and frequency domain embedding utilizing pseudo-random techniques.

3.8.2 Steganography software strength

Until there exist true secret or public key steganography, there will be a cat and mouse game between developers of steganography algorithms and steganalysts.

Current steganography software are considered *broken*, if there exist known tools that defeat them. Other software are using embedding methods for which a detection tool can easily be created, say appending data to image files.

The complexity of the embedding algorithms are increasing with generation. There is not a direct relation between steganographic strength, in the same understanding as cryptographic strength, and steganography software generation. However, late generations have adapted from previous weaknesses and is considered stronger. Secret key and public key steganography are per definition *secure*.

3.8.3 Steganography software availability

There exist commercial steganography software ranging from a few dollars to about $50. Some are freeware and others open source.

3.8.4 The classification

The steganography software in Chapter 4 is classified according to the above mentioned methods. Table 3.1 summaries the classification.

Generation	**Embedding methods**	**Example carriers**
Gen. 0	Appending, comments, file headers, ...	HTML comments, white spaces
Gen. 1	Spatial domain (LSB embedding)	GIF, BMP and WAV
Gen. 2	Spatial domain w/ pseudo-random emb.	BMP
Gen. 3	Frequency domain (DCT embedding)	JPEG, MP3
Gen. 4	Frequency domain Statistics-aware & pseudo-random emb	JPEG

Table 3.1: Classification of steganography software.

Chapter 4

Analysis of steganography software

4.1 Introduction

This chapter discusses some of the steganography software available from the web. It is important to note that the list presented here is not complete. The book "Investigator's guide to steganography" [26] gives a long list of steganography software and companies working with steganograhpy (incl. watermarking). Other listings of steganography software is found at [web21, web22]. The longest list of such software are from Backbone Security [web3], the Steganography Application Fingerprint Database (SAFDB). It claims to provide hash values from 230 data-hiding applications[1].

Analysis of steganography software is also done by others. [web17] presents as good overview of twelve steganography software. Eleven of them are additional to the ones treated in this master thesis. The analysis presented here is my own. Where information are obtained from elsewhere, this is clearly referenced.

Not all steganography software could possibly be assessed during this master thesis. The selected tools are listed in Table 4.1. They represent examples of free and licensed, various embedding methods, and from academic, professional and layperson. Due to the varied selection, experiences from a wide range of steganography software is presented.

[1]This list is "... free to qualifying US law enforcement, military, government, and intelligence agencies". After inquires to the company, it was not possible to obtain (a subset of) this database.

Name	Section	Why selected
EzStego	4.2	Pallete, LSB embedding
Mandelsteg	4.3	No carrier
Spam Mimic	4.4	Linguistic steganography
Snow	4.5	White space appending
Outguess	4.6	Academic work
appendX	4.7	File appending
Invisible Secrets	4.8	Licenced

Table 4.1: Steganography software treated in this chapter.

When analyzing each tools, how to use it is first addressed. Then follows a discussion of what the embedding does to the carrier-message, i.e. how a stego-message might be detected. Message extraction is for *simple steganography* software straight forward, just running the identified algorithm in reverse. With *secret steganography*, the stego-key has to be identified. However, not all tools using a stego-key are secure. Message extraction is addressed for all tools.

4.2 Description of EzStego

Name	EzStego 2.0b4
Licensing	Open source
Generation	1
Classification	LSB embedding
Carrier type	GIF image
URL	www.stego.com/ezstego/ezstego2b.zip

The EzStego tool uses GIF images as carrier files. To understand how this is done, some background on the GIF-file format is needed. Graphics Interchange Format (GIF) is a bitmap image (raster image), widely used on the Internet. The RBG color model is used[2], where each color is represented with a combination of red, green, and blue color. Using 8 bits for each of the tree base colors yields a total of $256^3 = 16777217$ possible colors. The GIF-file format used a palette, which is the list of the RGB colors used in the GIF image. The palette is limited to 256 different color values. The image itself is then a grid, where each cell (pixel) points to the appropriate position of the palette. So when rendering the picture, the color for each pixel is looked up in the palette.

EzStego adds the message into the least significant bit (LSB) of pixels and works in the following steps:

1. Create a copy of the palette, rearranging it so colors close to each other in the RBG color model are close in the palette.

2. Do as long as there are more bits in the message:
   Find the index, i, of this pixel's RGB color in the sorted palette
   Replace LSB of i with bit from message, creating i^*
   Find the RGB color i^* points to in the sorted palette

[2]See en.wikipedia.org/wiki/RGB for more resources on RGB.

Find the index of this new RGB color in the original palette
Change the pixel to this index

Figure 4.1 shows graphically the embedding method used with EzStego To summarize, the principle of EzStego is based on the similarity of colors in the palette. For a pixel, choose color a if message bit is 0, and color b if message bit is 1.

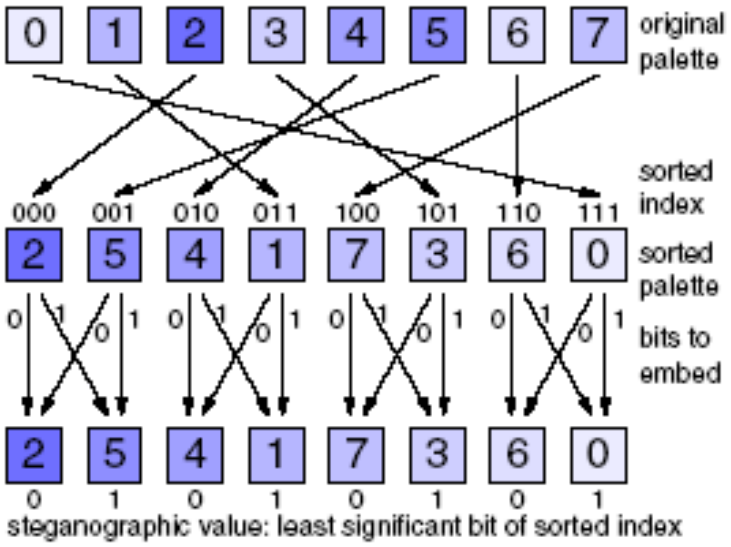


Figure 4.1: Embedding method of EzStego [59]

To recover the hidden message from the carrier, just find the index of the pixel's color in the sorted palette. The least significant bit is the embedded bit. When extracting the message from the carrier file, EzStego does not know the length of the original message. So the result is padded with garbage.

4.2.1 Usage of EzStego

EzStego can be run from the command line or with the GUI that comes with the tool. Figure 4.2 shows that there are no perceptible differences between carrier- and stego-image.

When testing the software on a larger number of images on a Windows machine, it is quite useful to write a small batch file[3]. Listing 4.1 shows a batch file which will embed a file, README, into all GIF-images in the current folder. How to extract a message is also shown. The extracted file contains the embedded message padded with "noice" from the image, i.e. the unused space.

4.2.2 Detection of EzStego

A naive method to detect messages embedded with EzStego, is to run all files through EzStego with the -unsteg option and check the result. Is the embedded

[3]A good primer on batch files is `http://www.computerhope.com/batch.htm`.

(a) Cover-image (b) Stego-image

Figure 4.2: Demonstration of EzStego using Lena image.

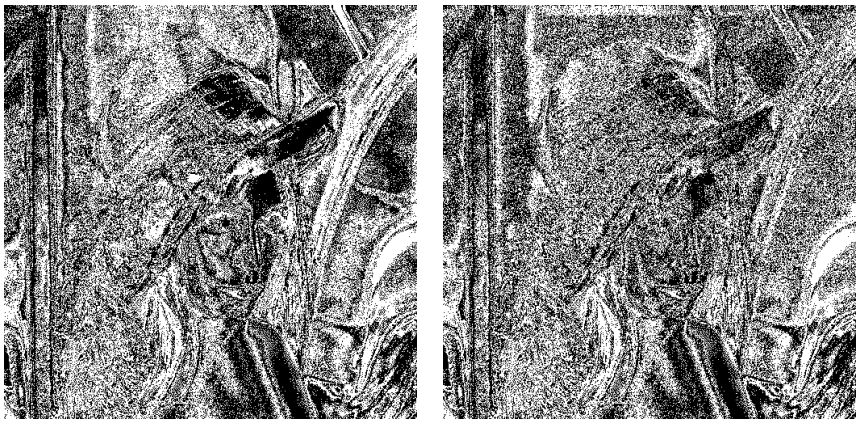
```
0 @echo off
  REM Creating Stego-images
2
  FOR %%i IN (*.gif) DO (
4         java EzStego -nogui -verbose -image %%i -input README -output
              %%~niStego.gif
  )
6 REM Extracting a message
  java EzStego -nogui -verbose -unsteg -image LenaStego.gif -output
      READMEsteg
```

Listing 4.1: Batch file used when running EzStego. All GIF-images in the folder will be used as carrier, when embedding the README file. Stego-images are named *Stego.gif.

message plain text it is readable, and if it is encrypted, header information can indicate this. Both these cases will break EzStego.

If the encrypted message has been stripped for header information, the extracted message will be a pseudo-random bit string These bits will not seam different from a bit stream extracted from a carrier-image. A method to still detect stego-messages created with EzStego is called *Visual attack* [59] and is described in Section 5.4.2.3, followed with a statistical attack that also breaks EzStego in Section 5.4.2.4.

There are no apparent changes to the carrier file. Figure 4.3 shows the LSB from Figure 4.3, and no abnormalities can be seen. The palette stays the same as the original, and no other strange artifacts are added to the stego-file when embedding. However, as mentioned above, EzStego can still be successfully attacked.



(a) LSB of cover-image (b) LSB of tego-image

Figure 4.3: LSB of images from Figure 4.2.

4.2.3 Message extraction

To recover the hidden message from the stego-image, just find the index of the pixel's color in the sorted palette. The least significant bit is the embedded bit. When extracting the message from the carrier file, EzStego does not know the length of the original message. So the result is padded with garbage. Just for clarity, EzStego is not using a steganography key and the message extraction is straight forward. Hence the attack on ExStego is reduced to prove whether or not the extracted message is just noise or a real message.

4.3 Description of Mandelsteg

Name	Mandelsteg
Licensing	Freeware
Generation	0
Classification	Image creation (Fractal)
Carrier type	GIF image
URL	ftp.univie.ac.at/security/crypt/steganography/MandelSteg1.0.tar.gz

The Mandelsteg tool differs from the other steganography tools in that it does not use an existing carrier. It creates stego-images based on Mandelbrot fractals.

4.3.1 Usage of Mandelsteg

Mandelsteg comes with a readme-file, describing usage and a short discussion of the security of the tool.The belonging tool *GIFExtract* is used to extract the message from the stego-image. It simply extracts the specified bit plane from the stego-image.

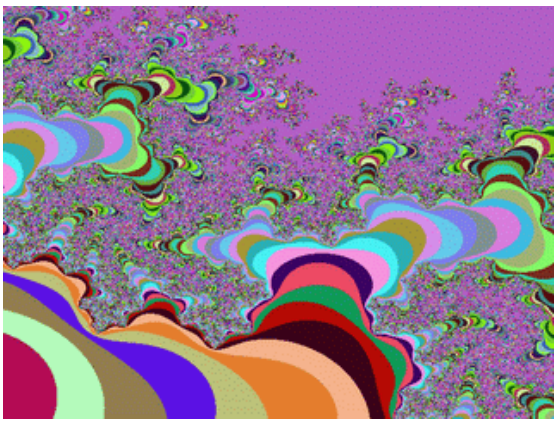


Figure 4.4: Mandelbrot image containing the text from Listing 4.2

4.3.2 Detection of Mandelsteg

As mentioned above, the readme-file from mandelsteg describes some security aspects.

All images from the mandelsteg tool have 256 palett entries in the color index and all have 128 unique colors with two palette entries for each color [21]. Picture 4.5 shows the palette of the mandelsteg image in Figure 4.4.

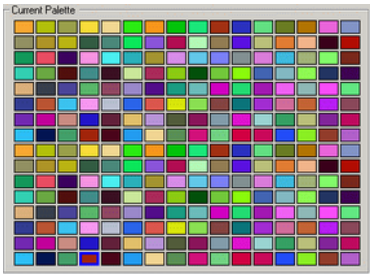


Figure 4.5: Palette from the Mandelbrot fractal image in Figure 4.4. Observe the repeating of the first 128 colors.

Mandelsteg is, from the forensic viewpoint, not a good alternative to hide messages. The use of a mandelbrot fractal image is too unusual. A visual inspection of images on seized data which detects the presence of mandelbrot fractal images, would lead to the suspicion of steganography usage.

4.3.3 Message extration

To extract a message from Mandelsteg, a tool called *GIFExtract* is included. It simply extracts the bit plane, which number is supplied as a command line option.

For an adversary to extract the message, it is only a matter of running GIFExtract. A brute-force attack is possible, due to the fact that the different command line options only provides 16^4 different possibilities. If the embedded message is not encrypted or known encryption headers are identified, the presence of steganography is detected, hence the tool is defeated. When the message is encrypted and stripped of headers, it would appear pseudo-random and can not be differed from other messages (noise) extracted.

4.4 Description of Spam Mimic

Name	Spam Mimic
Licensing	Freeware
Generation	0
Classification	Text generation (spam)
Carrier type	Text (Email)
URL	www.spammimic.com/

4.4.1 Usage of Spam Mimic

According to The Register [web26] , there is at last a positive usage of spam. Spam Mimic is a steganography tool that uses spam as stego-media.

Spam Mimic works similar to Mandelsteg (Section 4.3), on the fact that they do not need an existing carrier. The output from Spam Mimic is text with the characteristics of looking like spam. Listing 4.2 shows an example spam message generated with Spam Mimic.

The idea behind Spam Mimic is that there is sent a lot of spam. For an adversary it would not raise suspicion that the subject receives spam, i.e. email gets a (one-way) subliminal channel.

```
Dear Friend ; We know you are interested in receiving cutting-edge
announcement. If you are not interested in our publications and
wish to be removed from our lists, simply do NOT respond and
ignore this mail ! This mail is being sent in compliance with
Senate bill 1627 ; Title 3 , Section 305 . This is NOT
unsolicited bulk mail . Why work for somebody else when you can
become rich within 14 days . Have you ever noticed society seems
to be moving faster and faster and most everyone has a cellphone!
Well, now is your chance to capitalize on this ! WE will help YOU
decrease perceived waiting time by 130\% plus sell more . You can
begin at absolutely no cost to you ! But don't believe us. Mr Jones
who resides in Georgia tried us and says "I was skeptical but it
worked for me" . We are licensed to operate in all states ! For
God's sake, order now! Sign up a friend and you'll get a discount
of 90%. God Bless !
```

Listing 4.2: Output from Spam Mimic with input "Steganography".

There is a web interface of Spam Mimic at the homepage of the authors.

[4]Could also be more than 16 possibilities, but they do not affect the possibility do detect the embedded message

4.4.2 Detection of Spam Mimic

An apparent challenge with Spam Mimic is that it by nature is one way. Let us say William is monitoring traffic coming to and from Alice. Inbound spam should not raise suspicion, spam originating from Alice probably would.

Spam Mimic is freely available, so the algorithm for decoding and encoding has to be considered known by William. A forensic investigation, monitoring traffic to and from a suspect will detect traffic going to the Spam Mimic homepage, and obviously alert the investigator and hence defeat the steganography.

4.4.3 Message extraction

Spam Mimic is *simple steganography*, hence running the algorithm in reverse, using the freely available tool, will yield the embedded message.

4.5 Description of Snow

Name	Snow
Licensing	Open source
Generation	0
Classification	White space appending
Carrier type	Text
URL	www.darkside.com.au/snow/index.html

Snow (Steganographic nature of Whitespace) is very simple steganography software. It appends white spaces to lines in ASCII text files, where the embedded-message is encoded as space and tabulator characters. These white spaces at the end of lines do not change the appearance of the file in normal text viewers, hence the resulting stego-message is not visibly different from the original carrier.

4.5.1 Usage of Snow

Listing 4.3 shows the contents of the bat-file used to run Snow. In this case, encryption and compression is not used for simplicity. In a real-word example at least encryption should be used for additional protection of message contents.

```
1 @echo off
  REM Usage: snow [-C][-Q][-S][-p passwd][-l line-len] [-f file | -m
      message] [infile [outfile]]
3 REM Display the approximate amount of space available for hidden message
  snow -S -l 100 cover.t
5 REM Embedd msg.t into cover.t
  snow -l 100 -f msg.t cover.t stego.t
```

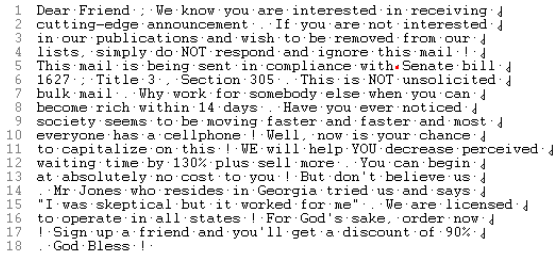
Listing 4.3: Bat-file to run Snow

The $-lline - len$ option gives the max line length in the output. If Snow is not able to append spaces and tabular within this limit, a warning is given and new lines are appended to the carrier-message. This will clearly alter the visual characteristic of the stego-message and should be avoided. The output from running the bat-file is showed in Listing 4.4.

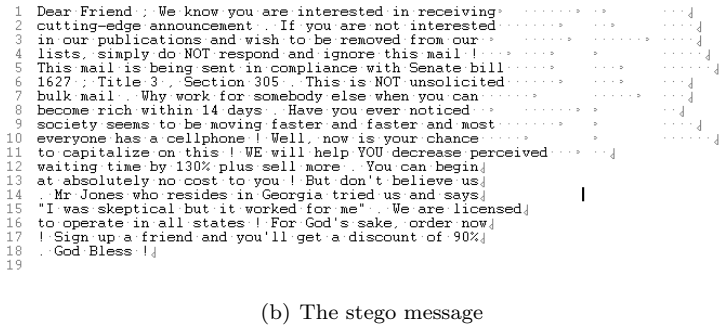
```
File has storage capacity of between 281 and 386 bits
Approximately 41 bytes.
Message used approximately 31.42% of available space.
```

Listing 4.4: Output from running the bat-file from Listing 4.3

Figure 4.6 gives an example of the use of Snow. The carrier-text is taken from the Spam Mimic tool presented in Section 4.4 and repeated here with visible white spaces. When tabular and spaces are shown, the presence of embedded data is clearly visible.



(a) The carrier message



(b) The stego message

Figure 4.6: Usage of the Snow tool. White spaces are shown to visualize the added spaces.

The Snow tool gives a method to conceal the message; the users need to find a way to distribute the stego-message without suspicion. The *Snow web-page encryption/decryption* [web34], while misleadingly named, still gives an easy to use interface and idea to use web pages as carrier-messages. A quick glance at the resulting web page or source code will not break the tool. Although the steganographic strength of Snow has to be considered *weak*, as the next section will show.

4.5.2 Detection of Snow

The Snow tool is not very sophisticated and leaves clear indicators of its presence, but only if they are searched for. The trailing spaces and tabulators of lines are not normal and is a give-away. It can be detected with visual inspection of all text files, but a more feasible solution is to create some tools that can automate this process. And in this case, a good detection algorithm is achievable.

In the same way as when the Snow tool tries to detect the start of embedded data, the detection algorithm can also search for this trailing tabulator. And with the presence of this and more trailing tabulators and spaces, there is a strong probability of the presence of an embedded message.

4.5.3 Message extraction

When Snow has been detected, it is straight forward to extract the embedded message. As stated earlier, this embedded message has to be assumed encrypted. The Snow tool even ships with a possibility for encryption with a 64-bit block cipher called ICE (developed by the same author as Snow). There is of course also the possibility to use other encryption algorithms together with Snow.

4.6 Description of Outguess

Name	Outguess 0.2
Licensing	Open source
Generation	4
Classification	DCT w/ statistics-aware embedding
Carrier type	JPEG
URL	www.outguess.org

Outguess was created by Niels Provus [web38]. It is meant as a framework for information hiding, not depending on the data type of the carrier. But a handler has to be created for each type, and currently Outguess supports PNM and JPEG image formats. Outguess is resulted from a professional researcher from the academic society.

4.6.1 Usage of Outguess

To use Outguess efficiently on several images, a small script is created. Listing 4.5 shows the script. It creates two stego-images for each cover-image, one with statistic preservation option and one without. The naming convention is to add “Steg” to the stego-image with preservation and “StegF” to the other. The script is based on *seek_script* following Outguess, originally used to locate the best carrier-image in a directory for a given message.

Listing 4.6 shows an extract from the log when running the *outguess_script* from Listing 4.5.

```
#!/bin/sh
# A very simple script using OutGuess to find an image that yields
# the best embedding.
# (C) 1999 Niels Provos
FILES=*.jpg
#MESSAGE=/tmp/fortune
MESSAGE=msg.txt
TMPNAME="STEG.jpg"
TMPNAME2="STEG_f.jpg"
ARGS="-d $MESSAGE -k test123"
ARGS2="-d $MESSAGE -k test123 -F-"
OUTGUESS="../outguess/outguess"
BEST=0
WORST=0
NAME="no name"

if [ ! -f "$MESSAGE" ] ; then
        echo "The file $MESSAGE does not exist"
        exit
fi

for name in $FILES
do
        echo -n "$name "
        $OUTGUESS $ARGS $name $name$TMPNAME
        $OUTGUESS $ARGS2 $name $name$TMPNAME2
done


```

Listing 4.5: Script running Outguess, called *outguess_script*. Embedding a message in all *.jpg files in the current directory.

```
Script started on Sat Jun 11 17:33:01 2005
# ls
arctic_hare.jpg  f14.jpg   log.script  peppers.jpg
bear.jpg         lena.jpg  msg.txt     seek_script
# ./outguess_script
arctic_hare.jpg Reading arctic_hare.jpg....
JPEG compression quality set to 75
Extracting usable bits:   24145 bits
Correctable message size: 11083 bits, 45.90%
Encoded 'msg.txt': 1344 bits, 168 bytes
Finding best embedding...
     0:   666(48.4%)[49.6%], bias   681(1.02), saved:     0, total:
          2.76%
     3:   674(49.0%)[50.1%], bias   631(0.94), saved:     0, total:
          2.79%
     6:   657(47.7%)[48.9%], bias   634(0.96), saved:     1, total:
          2.72%
     7:   637(46.3%)[47.4%], bias   594(0.93), saved:     4, total:
          2.64%
    35:   655(47.6%)[48.7%], bias   572(0.87), saved:     2, total:
          2.71%
    92:   632(45.9%)[47.0%], bias   550(0.87), saved:     5, total:
          2.62%
92, 1182: Embedding data: 1344 in 24145
Bits embedded: 1376, changed: 632(45.9%)[47.0%], bias: 550, tot: 24053,
    skip: 22677
Foiling statistics: corrections: 274, failed: 0, offset: 54.351648 +-
    165.267897
Total bits changed: 1182 (change 632 + bias 550)
Storing bitmap into data...
Writing arctic_hare.jpgSTEG.jpg....
```

Listing 4.6: Running *outguess_script* from Listing 4.5

(a) Original image (cover-image)

(b) Message embedded (stego-image)

(c) Zoom 1000% cover-image

(d) Zoom 1000% stego-image

Figure 4.7: Before and after running Outguess on "pepper".

The files listed in Listing 4.7 are the carrier-images and stego-images from the *outguess_script*

```
# ls
arctic_hare.jpg              f14.jpg              log.script
arctic_hare.jpgSTEG_f.jpg    f14.jpgSTEG_f.jpg    msg.txt
arctic_hare.jpgsteg.jpg      f14.jpgsteg.jpg      peppers.jpg
bear.jpg                     lena.jpg             peppers.jpgSTEG_f.jpg
bear.jpgSTEG_f.jpg           lena.jpgSTEG_f.jpg   peppers.jpgsteg.jpg
bear.jpgsteg.jpg             lena.jpgsteg.jpg     seek_script
```

Listing 4.7: Files created with *outguess_script*.

Figure 4.7 shows one of the images before (4.7(a)) and after (4.7(b)) running Outguess. The original TIF image was converted to JPEG, with setting on "best quality". The original is 223 KB and the stego-image 40 KB. Creating cover with similar size as the stego-image from Figure 4.7 and running Outguess on this image, presents no strange artifacts. Cover- and stego-image are perceptually identical. When zoomed in 1000% (figures 4.7(c))and 4.7(d)) , differences between the images can be seen. But it can not be told which is cover-image or stego-image, by just looking at the images.

4.6.2 Detection of Outguess

As shown in Figure 4.7, Outguess can not be visibly detected. Outguess 0.13b is detected by *Stegdetect*[5]. The detection of outguess 0.13b is done based on a statistical attack, described in Section 5.4.2.4. OutGuess 0.2. defeats Stegdetect by preserving the statistics from the carrier-image.

Outguess is a result from the academic community and is one of the more sophisticated steganography software available. Outguess 0.2 has also been attacked by the academic community and broken [16]. Section 5.4.2.4 has more on this steganalysis method which defeats Outguess 0.2.

4.6.3 Message extraction

With knowledge of the correct key, it is no problem to extract a message embedded with Outguess, as shown in Listing 4.8

```
# ../outguess/outguess -k "test123" -r peppers.jpgsteg.jpg out.txt
Reading peppers.jpgsteg.jpg....
Extracting usable bits:   41477 bits
Steg retrieve: seed: 119, len: 168
# cat out.txt
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
# cat msg.txt
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
Test message for embedding.
```

Listing 4.8: Extracting message embedded with Outguess.

However, if the key is unknown, message extraction is more problematic. Listing shows the two possible outcomes of a wrong password; a *floating point exception* or noise.

There exist a tool called *Stegbreak* which performs a dictionary attack against Outguess. This tool is treated in Section 6.4. It is clear that the floating point exception can be used as a wrong password indicator. When the extraction succeeds, the message need to be checked. With chipertext, this is done with the detection of file headers in the extracted message. If the header is stripped of, Stegbreak can not determine if the tried password is successful.

[5]Treated in Section 6.3

```
# ../outguess/outguess -k "test" -r peppers.jpgsteg.jpg out.txt
Reading peppers.jpgsteg.jpg....
Extracting usable bits:   41477 bits
Steg retrieve: seed: 44872, len: 3569
Floating point exception
# ../outguess/outguess -k "t" -r peppers.jpgsteg.jpg out.txt
Reading peppers.jpgsteg.jpg....
Extracting usable bits:   41477 bits
Steg retrieve: seed: 3242, len: 2707
# xxd out.txt | head
0000000: 977f 816f aa4f f42c e649 2d95 50a8 d659   ...o.O.,.I-.P..Y
0000010: 6cae e8b7 ae78 d2ef 5262 fa8e df85 011d  l....x..Rb......
0000020: b722 b1ed 4385 2292 fe42 6dc0 45b6 e35f  ."..C."..Bm.E.._
0000030: bf29 5ea3 d399 61ad c106 c229 20d1 e82f  .)^...a....) ../
0000040: 997f 2a3c 9b28 6094 f143 52ca 5367 7f41  ..*<.(`..CR.Sg.A
0000050: f843 06aa a6fd 20a4 f1c1 e031 33ee bcda  .C.... ....13...
0000060: 4e6c 221a 6c5b b07f ac6c 25da af4a bd00  Nl".l[...l%..J..
0000070: e434 b00f 1e06 3169 09de 5af1 4d10 2621  .4....1i..Z.M.&!
0000080: b100 bb3e 0fa8 28ee 661b 4ef9 c14a 9ee0  ...>..(.f.N..J..
0000090: 45e1 1700 47f2 585a 62a9 affc ac26 c720  E...G.XZb....&.
# exit

Script done on Sat Jun 11 17:44:52 2005
```

Listing 4.9: Brute-force attempt against Outguess. This listing shows the two possible outcomes with a wrong password. The contents of the extracted message is shown using *xxd* and is just noise.

4.7 Description of appendX

Name	appendX 0.4
Licensing	Open source
Generation	0
Classification	Data appending
Carrier type	PNG, JPEG, GIF, ...
URL	www.unet.univie.ac.at/ a9900470/appendX/

appendX is a simple steganography tool. The embedding method is simply appending data to the end of the carrier-file.

4.7.1 Usage of appendX

As stated, appendx is quite simple. It is written in perl, and Listing 4.10 shows the options and usage of the tool. As long as perl is available on the system, there are no required installations. The message to be embedded is read from stdin and is not compressed.

appendX supports PGP-header stripping. When a message is encrypted with PGP, a header is added to the chipertext. This header clearly identifies the hidden data as chipertext. When this header is stripped, the appended data looks like noise. Listing 4.11 shows the continuation of running appendX.

The end of the embedded message and the stego-message are also shown in Listing 4.11. Observe the additional spaces and string (3ad) representing 0x3ad, the length of the embedded message.

```
$ ./apX
appendX 0.3
syntax apX [command] [option] [infile] [outfile]
Commands are: help extract restore append
Options:
        -s strips/adds the pgp header (can be combined with extract or
            append)
        -Z supresses compression/uncompression. (use this to communicate
             with a version <=0.4
written by mihi, i don't care what you use it for
ABSOLUTLY NO WARRANTIES OF WHAT THIS SKRIPT DOES OR DOESN'T
$ ./apX append -Z lena.jpg leneSteg.jpg < msg.txt
Your ... please:
Dear Friend ; We know you are interested in receiving
cutting-edge announcement . If you are not interested
in our publications and wish to be removed from our
```

Listing 4.10: Usage of appendX

```
! Sign up a friend and you'll get a discount of 90%
. God Bless ! $
$ xxd msg.txt | tail
0000310: 7420 776f 726b 6564 2066 6f72 206d 6527  t worked for me'
0000320: 2720 2e20 5765 2061 7265 206c 6963 656e  ' . We are licen
0000330: 7365 6420 0a74 6f20 6f70 6572 6174 6520  sed .to operate
0000340: 696e 2061 6c6c 2073 7461 7465 7320 2120  in all states !
0000350: 466f 7220 476f 6427 7320 7361 6b65 2c20  For God's sake,
0000360: 6f72 6465 7220 6e6f 7720 0a21 2053 6967  order now .! Sig
0000370: 6e20 7570 2061 2066 7269 656e 6420 616e  n up a friend an
0000380: 6420 796f 7527 6c6c 2067 6574 2061 2064  d you'll get a d
0000390: 6973 636f 756e 7420 6f66 2039 3025 200a  iscount of 90% .
00003a0: 2e20 476f 6420 426c 6573 7320 21          . God Bless !
$ xxd lenaSteg.jpg | tail
0000320: 2720 2e20 5765 2061 7265 206c 6963 656e  ' . We are licen
0000330: 7365 6420 0a74 6f20 6f70 6572 6174 6520  sed .to operate
0000340: 696e 2061 6c6c 2073 7461 7465 7320 2120  in all states !
0000350: 466f 7220 476f 6427 7320 7361 6b65 2c20  For God's sake,
0000360: 6f72 6465 7220 6e6f 7720 0a21 2053 6967  order now .! Sig
0000370: 6e20 7570 2061 2066 7269 656e 6420 616e  n up a friend an
0000380: 6420 796f 7527 6c6c 2067 6574 2061 2064  d you'll get a d
0000390: 6973 636f 756e 7420 6f66 2039 3025 200a  iscount of 90% .
00003a0: 2e20 476f 6420 426c 6573 7320 2120 2020  . God Bless !
00003b0: 2020 2020 3361 64                            3ad
$
```

Listing 4.11: Usage of appendX, continued from Listing 4.10. The output of the message from appendX is skipped.

4.7.2 Detection of appendX

The hidden data is appended to the end of the carrier. For a JPEG data file[6], this would mean data after the End of Image (EOI) marker, (hex) ffd9. For a BMP data file, it would mean more data than stated in the BMP header. appendX is classified as *weak*, since it is relatively easy do detect the presence of embedded data.

appendX also creates some sort of file signatures. After the embedded message, the length of the embedded message is padded, from the left, until ten characters and appended. This is not adding more needed knowledge to detect steganography, but can be a help in identifying the software used.

4.7.3 Message extraction

Once appendX has been identified, it is straightforward to extract the additional data at the end of the file. And the problem is reduced to decryption and decompression, if used.

4.8 Description of Invisible Secrets

Name	Invisible Secrets 4.0
Licensing	Licensed
Generation	0 / 1
Classification	LSB (BMP, WAV), Comment insertion (JPEG, PNG), append space (HTML)
Carrier type	BMP, WAV, JPEG, PNG, HTML
URL	www.invisiblesecrets.com/

Invisible Secrets is available from NeoByte Solutions [web45]. They provide a software security package with encryption, safe deletion (overwriting deleted data) and steganography. The steganography part can hide information inside JPEG, BMP, PNG, HTML and WAV- data files. The corresponding techniques are LSB embedding, comment insertion and white space appending. The software is easy to use, well documented and integrates into the windows shell and start menu by default. Invisible Secrets has a single user license fee of 40$. However, the embedding techniques available are not more sophisticated than freely available steganography tools. But it is very user friendly.

4.8.1 Usage of Invisible Secrets

Figure 4.8(a) shows the results of embedding with Invisible Secret the result of Spam Mimic in the web page wwww.ntnu.no. The embedded message is the result from Spam Mimic (Listing 4.2). The resulting stego-message is similar to the one from SNOW (See Figure 4.6).

[6]More info on the JPEG file format can be found at `http://www.obrador.com/essentialjpeg/headerinfo.htm`

Figures 4.8(b), 4.8(c) and 4.8(d) displays the hidden message in the comment field of a JPEG image.

4.8.2 Detection of Invisible Secrets

Invisible Secrets is not hard to detect. The methods used to embed a message are well known. For white space appending, the same applies as for SNOW (Section 4.5.2). LSB embedding is addressed with EzStego (Section 4.2.2). With Invisible Secrets, the embedding can be compressed and encrypted. But adding comments to the JPG image, in plaintext or encrypted, as in Figure 4.8, stands out from cover messages. The same goes for PNG images.

An interesting feature of Invisible Secrets is the possibility to create bogus stego-messages. Doing so could increase the difficulty to detect a specific hidden message. But more stego-messages would increase the possibility of steganography usage detection, hence the covert channel is defeated. This is especially correct when using simple steganography software, which are fragile to algorithm exposure.

4.8.2.1 Earlier versions of Invisible Secrets

Earlier version of invisible Secrets had additional properties that could be used for steganalysis. With LSB embedding, *Invisible Secrets 2002* padded the unused areas with all 0's or all 1's [web17]. Clearly, this is not normal for images.

4.8.3 Message extraction

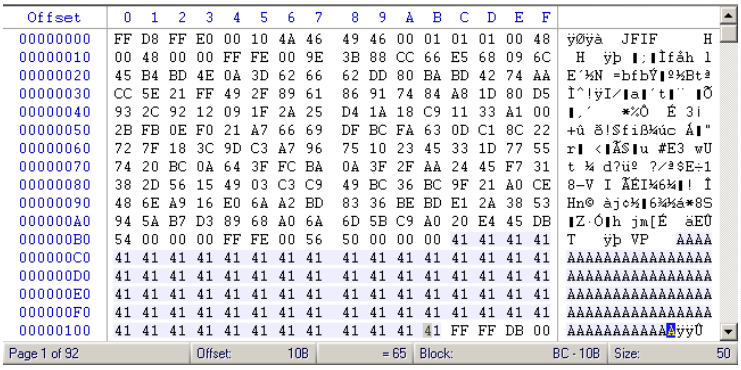
The extraction of a message embedded with Invisible Secret is possible, and varies of course with the different methods used for embedding.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel='stylesheet' href='emeny000.css'>
<title>NTNU - Norges teknisk-naturvitenskapelige universitet</title>
</head>

<body>

<script type='text/javascript'>
```

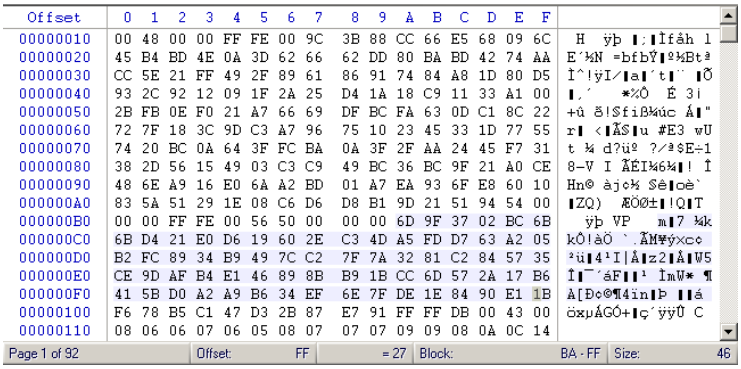
(a) HTML steganography with appending spaces



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	48
00000010	00	48	00	00	FF	FE	00	9E	3B	88	CC	66	E5	68	09	6C
00000020	45	B4	BD	4E	0A	3D	62	66	62	DD	80	BA	BD	42	74	AA
00000030	CC	5E	21	FF	49	2F	89	61	86	91	74	84	A8	1D	80	D5
00000040	93	2C	92	12	09	1F	2A	25	D4	1A	18	C9	11	33	A1	00
00000050	2B	FB	0E	F0	21	A7	66	69	DF	BC	FA	63	0D	C1	8C	22
00000060	72	7F	18	3C	9D	C3	A7	96	75	10	23	45	33	1D	77	55
00000070	74	20	BC	0A	64	3F	FC	BA	0A	3F	2F	AA	24	45	F7	31
00000080	38	2D	56	15	49	03	C3	C9	49	BC	36	BC	9F	21	A0	CE
00000090	48	6E	A9	16	E0	6A	A2	BD	83	36	BE	BD	E1	2A	38	53
000000A0	94	5A	B7	D3	89	68	A0	6A	6D	5B	C9	A0	20	E4	45	DB
000000B0	54	00	00	00	FF	FE	00	56	50	00	00	00	41	41	41	41
000000C0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
000000D0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
000000E0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
000000F0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
00000100	41	41	41	41	41	41	41	41	41	41	41	41	FF	FF	DB	00

Page 1 of 92 | Offset: 10B = 65 | Block: BC - 10B | Size: 50

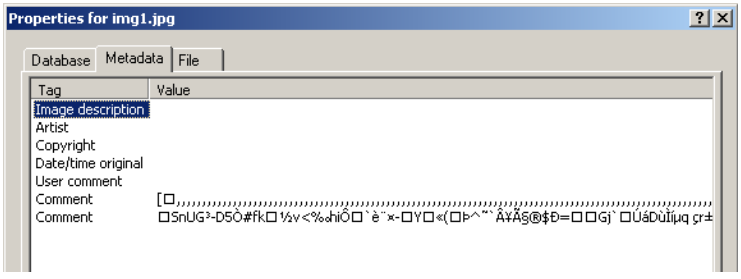
(b) Hiding "AAAA...." inside JPG comment without compression and encryption



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000010	00	48	00	00	FF	FE	00	9C	3B	88	CC	66	E5	68	09	6C
00000020	45	B4	BD	4E	0A	3D	62	66	62	DD	80	BA	BD	42	74	AA
00000030	CC	5E	21	FF	49	2F	89	61	86	91	74	84	A8	1D	80	D5
00000040	93	2C	92	12	09	1F	2A	25	D4	1A	18	C9	11	33	A1	00
00000050	2B	FB	0E	F0	21	A7	66	69	DF	BC	FA	63	0D	C1	8C	22
00000060	72	7F	18	3C	9D	C3	A7	96	75	10	23	45	33	1D	77	55
00000070	74	20	BC	0A	64	3F	FC	BA	0A	3F	2F	AA	24	45	F7	31
00000080	38	2D	56	15	49	03	C3	C9	49	BC	36	BC	9F	21	A0	CE
00000090	48	6E	A9	16	E0	6A	A2	BD	01	A7	EA	93	6F	E8	60	10
000000A0	83	5A	51	29	1E	08	C6	D6	D8	B1	9D	21	51	94	54	00
000000B0	00	00	FF	FE	00	56	50	00	00	00	6D	9F	37	02	BC	6B
000000C0	6B	D4	21	E0	D6	19	60	2E	C3	4D	A5	FD	D7	63	A2	05
000000D0	B2	FC	89	34	B9	49	7C	C2	7F	7A	32	81	C2	84	57	35
000000E0	CE	9D	AF	B4	E1	46	89	8B	B9	1B	CC	6D	57	2A	17	B6
000000F0	41	5B	D0	A2	A9	B6	34	EF	6E	7F	DE	1E	84	90	E1	1B
00000100	F6	78	B5	C1	47	D3	2B	87	E7	91	FF	FF	DB	00	43	00
00000110	08	06	06	07	06	05	08	07	07	07	09	09	08	0A	0C	14

Page 1 of 92 | Offset: FF = 27 | Block: BA - FF | Size: 46

(c) Hiding "AAAA...." inside JPG comment with compression and encryption



(d) Viewing image metadata

Figure 4.8: Using Invisible Secrets to hide messages. Similar effects with HTML carries as with SNOW in Figure 4.6. The embedded data can with the JPEG images be seen with a hex viewer, as in figures 4.8(b) and 4.8(c), or using an image viewer which can show image metadata (Figure 4.8(d)). Observe how the encrypted data stand out.

4.9 Discussion

As this chapter has shown, various steganography tools exist. Different carrier-types and embedding methods are used. For all tools, it is possible to detect their usage. Each tool is examined with respect on how to detect it and this is the foundation for the following chapter on *steganalysis*; how to separate cover-message from stego-messages.

Chapter 5

Steganalysis

5.1 Introduction

This chapter presents the concept of *steganalysis*. First, the term steganalysis is described and the scenario *The Prisoner's Problem* is presented. Attacks on steganography are then linked with digital forensics, and various steganalysis methods are presented. Extraction of hidden information is treated, followed by a brief presentation of disabling hidden information.

5.2 Introduction to steganalysis

The process of detecting hidden messages is called steganalysis. The definition of steganalysis is limited to the detection of an embedded message, and not message extraction. The detection of a hidden message can identify the method used for the embedding. When the tool or method has been identified, it might be possible to extract the message.

Steganalysis is used to detect stego-messages among cover-messages. Other forensic methods can be used to defeat steganography. For instance, detection of steganography software on the suspect's computer. Chapter 7 suggests techniques aiding investigators in defeating steganography, where steganalysis is one of these methods.

To describe steganalysis, it is useful to use a scenario called *The Prisoner's Problem* illustrating steganography with different parameters.

5.2.1 The Prisoner's Problem

Steganography is often seen described using the *The Prisoner's Problem* [51]. Using the well established names for the different participant, the problem is displayed in Figure 5.1. There are two prisoners, Alice and Bob, trying to communicate with each other. The warden, named William, allows the prisoners to

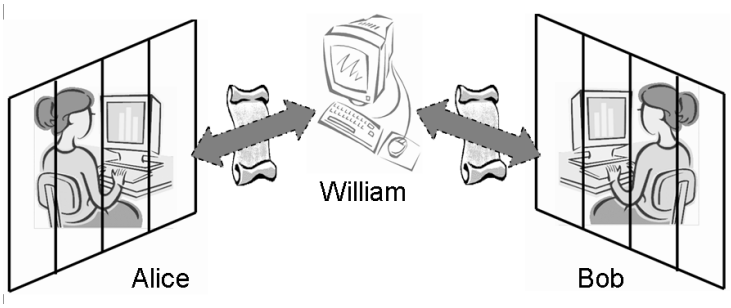


Figure 5.1: The Prisoners Problem. Figure adapted from [51]

communicate, but will intercept and give punishment if messages are discovered to contain illegal information (e.g. escape plans). The prisoners are willing to accept this risk and need to establish a way of communicating secretly in the message exchange, i.e. establishing a "subliminal channel".

To relate The Prisoner's Problem to the terminology from Figure 3.2, the legal communication that William, the warden, will allow to pass is *carrier-messages*. If William can detect the presence of a *stego-message*, Alice and Bob will be in trouble.

The scenario can be described with a passive or active warden. William can be passive, reading and allowing messages in which he can not detect the presence of a subliminal channel. An active William can alter messages at his will, where in the strictest scenario he changes all messages, in an attempt to prevent unwanted data exchange.

Hence attack on steganography can be separated into the two general cases:

Active wardens The adversary can alter the stego-message to wipe out the hidden message. From the prisoner's problem, the warden intercepts and changes all messages.

Passive wardens The adversary can intercept the stego-message and analyse its contents, i.e. the warden reads all messages.

The case with an active warden requires a robust steganography method, for Alice and Bob to communicate using a subliminal channel. But it is important to notice that the warden does not need to detect the presence of an embedded message to be active.

5.3 Description of steganalysis

Steganalysis could be described as a method to prevent steganography. However, there are other attacks on steganography. For example, attacking the end hosts of the steganography algorithm by searching for security credentials is not steganalysis. Therefore, digital forensics encompasses more methods than solely steganalysis to attack steganography.

The target for digital forensics is detection of steganography. The objective of steganalysis is "detecting messages hidden using steganography" [web52]. In other words, steganalysis is about separating cover-messages from stego-messages.

5.4 Attacks on steganography

Attacks and analysis of steganography might take different forms, called the "three D's" of defeating steganography [web11]: *Detection*, *Decryption* and *Destruction*. [web11] explains decryption as used with cryptography and cryptanalysis. A more correct form would be *extraction*, i.e. to separate the embedded message from the stego-message. After the hidden (assumed) encrypted message has been extracted, this chipertext an be attacked using cryptanalysis techniques, or other forensic techniques, e.g. interrogate to obtain the password.

5.4.1 Steganalysis and digital forensics

Steganalysis is defined above to be the detection of hidden messages. Message extraction can then follow after successful steganalysis. Hence, steganalysis can be said to be an action taken during digital forensics. It is however natural to also include other forensic activities aiding the detection and extraction of hidden data. The following phases could then take place[1]:

Identification of digital media to be analyzed. This is based on input to the digital crime scene from the Physical crime scene investigation phases.

Development of algorithms to detect stego-messages. An important part of the Readiness phases is the creation of tools to detect stego-messages. Such tools are discussed in Chapter 5.

Identification of embedding method. How messages can be embedded are knowledge searched in the Readiness phases, and detected during the Digital crime scene investigation phases. E.g. spatial or frequency domain embedding, simple or secret key steganography.

Determining the steganography software. Obtain databases of steganography software signatures in the Readiness phases and search for them in the digital crime scene, or other means to detect steganography software.

The following phases are taking place in the Digital crime scene investigation phases.

Searching for steganography keys and message extraction. If the identified steganography software uses keys, these are needed to extract the embedded data.

Cryptanalysis to obtain the secret message The embedded message is probably chipertext.

[1] Adapted from [17] to the digital forensic process in Section 2.3.

5.4.2 Steganalysis: Detection of stego-messages

This section presents different methods for steganalysis. Most steganalysis algorithms and tools targets specific steganography software. *StegSpy* treated in Section 6.2 is an example. Such tools relay on specific signatures left in the stego-message. The embedding of a message can give a specific statistical property, which is another method used to detect stego-messages. Universal blind detectors are starting to emerge, mostly as theoretical algorithms and not as available tools, yet. *Stegdetect v0.6* (Section 6.3) supports linear discriminant analysis.

5.4.2.1 File signatures

Some steganography software add specific signatures to stego-messages. For example, the string "CDN" is always present when using Hiderman [web40]. Such file signatures can be used to detect stego-messages.

5.4.2.2 File anomalies

Some simple steganography software embed messages by appending data to the end of the carrier file. Hiderman, appendX (Section 4.7) and Invisible Secrets (Section 4.8) are examples of such steganography software. When these files are read by software, e.g. image viewers, the amount of data read depends on the file length defined in the file header. Hence the appended data is not read and the changes to the carrier are not perceptive.

When using Invisible Secrets 2002[2] and LSB embedding in BMP images, the bits not used for embedding are all set to 0 or to 1. I.e. the unused LSBs have an irregularity.

Such file anomalies can be detected when examining steganography software and used for steganalysis.

5.4.2.3 Visual attacks

It as been assumed that LSB of luminance values are be random[3]. This is however shown in [59] to be wrong and a summary is presented here.

The idea is to remove all parts of the image covering the message and use the human eye to decide whether there is a potential message or still image content. Using EzStego, it is recalled from Section 4.2 that the colors of each pixel, as defined by the palette, determines the embedded message. The filter for the visual attack, Figure 5.2, graphically presents the values of each pixel, i.e. the stego-message.

[2] Invisible Secrets 2002 is an earlier version of Invisible Secrets examined in Section 4.8.

[3] Twelve steganography software assuming this are referenced by [59], among them EzStego from Section 4.2

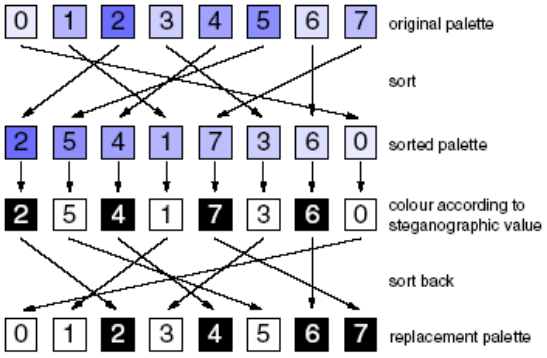


Figure 5.2: Visual attack filter: Assigning new colors to the palette. I.e. replacing even index in the sorted palette with black and odd with white [59].

Figure 5.3 shows the results from applying the visual attack. Figure 5.3(b) is the filtered LSB of the carrier-image in Figure 5.3(a). The image data is clearly visible. An embedded message is clearly identified in Figure 5.3(b), as well as the message length.

Visual attacks only succeeds when the cover-image has clearly structured contents. For instance, image textures typically withstand this type of attack. Figure 5.3(f) shows a filtered image, but it can not be told of this indicates an embedded message or just image content.

5.4.2.4 Statistical attacks

Steganography algorithms tries to embed messages in areas of the carrier that will not introduce perceptible changes, recall Eq. 3.1. However, there is discovered statistical changes to the carrier. I.e. there are statistical differences between C and C'. These differences can be use to break steganography algorithms.

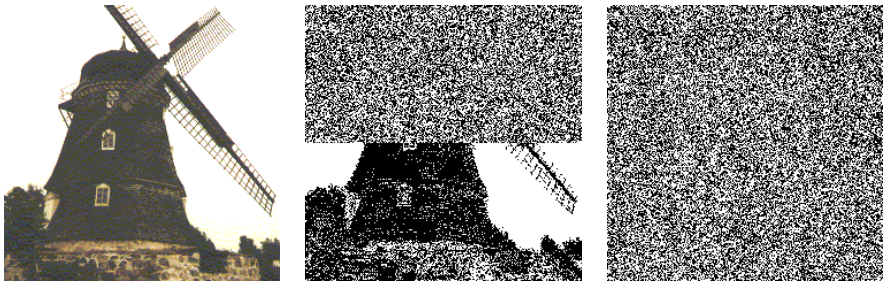
For example, by creating a norm for images, i.e. possible carriers, stego-images will derive from this norm. Some tests are independent of data format and measure only the entropy of redundant data. Stego-images are expected to have higher entropy [26].

There exist various methods for statistical steganalysis. Next follows a presentation of various methods in the literature.

The spatial domain LSB embedding described in Section 3.7.4 is done by several steganography software, e.g. EzStego. The assumption of the LSB being random noise in not correct. Hence, the statistical properties of the stego- and cover-image are different and can be detected, as proven by [59].

[15] attacks steganography systems that use JPEG images as carriers. The JPEG algorithm leaves distinctive fingerprints in JPEG images, and the steganalysis method from [15] uses these as a “fragile watermarks”. Are these watermarks destroyed, presence of steganography is assumed. The authors claim

(a) Carrier-image (b) Stego-image with 50% embedding. (c) Carrier-image 2



(d) Filtered (a) (e) Filtered (b) (f) Filtered (c)

Figure 5.3: Visual attacks on EzStego. Figure 5.3(e) shows a successful visual attack, and Figure 5.3(f) shows an inconclusive visual attack. All images from [59].

to detect embedded changes as small as modifying LSB of one random pixel. This method works well against spatial domain embedding, but not against steganography algorithms using discrete cosine transform (DCT) coefficients.

In additional to the visual attack mentioned above, [59] presents the *Chi-square Attack*. They introduce the Pair of Values (PoVs) concept. Consider EzStego and the LSB embedding, this process yield pairs of values only differing in their LSB. For a cover image, the color histogram is unevenly. After embedding of a message, with equally distributed bits, in all LSB, the occurrence of each PoVs becomes equal. When not all LSB of the carrier is used, a change in the statistics is observed at the end of the message. The PoVs can be all pairs, which are changed into each other when embedding the message.

The same statistical attack from [59] is done on DCT of JPEG images by [42]. This work has resulted in the Stegdetect tool, treated in Section 6.3.

–

5.4.3 Extracting hidden information

After the present of hidden information is known or suspected, methods trying to extract the embedded message would be the next step. This might be using the discovered steganography algorithm or tool. With *simple steganography*,

this is normally relatively easy. In the presence of a *stego-key*, this key is needed to succeed with the extraction.

Even though not all users of steganography might encrypt their message, this could be expected. Stripping headers from encrypted files results in an embedded message indistinguishable from noise.

PGP Stealth[4] is a tool which strips of all headers of a PGP encrypted message. The complexity of brute force search is then much greater, since for each *stego-key*, K_s, all encryption keys need to be tried. Only a successful cryptanalysis of the embedded message will show if the currently tried K_s is the correct one, or so it is assumed. However, [17] shows a technique that is $O(|K_s|)$, hence the stego-key search is independently of the cryptographic key size. But [17] also provides countermeasures to the presented stego-key search, which could be adopted by the steganography algorithm.

5.4.4 Disabling hidden information

Going back to Equation 3.1, defining a part t of the carrier C which can be altered without perceptible changes to C. This can also be used by the steganalyst to prevent embedded messages, i.e. by altering t for all C. From the Prisoner's problem, William is here an active warden and changes all messages, not needing to care whether the communication is a illegal stego-message or innocent and legal cover-message.

There are several ways to disable hidden information. A drastic step is to disallow all communication, e.g. intercept the communicated image. When this is not possible or wanted, changes to the (assumed) stego-message can be made. E.g. change file format, perform image processing like blur, crop etc. or add noise to the LSB of GIF images.

The original embedded message will probably be lost by the changes. However, it should be noted that there exist watermarks, which are quite robust and can withstand multiple changes. There is a trade-off between robustness and amount of hidden information. Adding error-correction to the message reduces the amount of information it carries.

Disabling hidden information is normally not a goal for digital forensics and is not treated in this master thesis.

[4] www.cypherspace.org/adam/stealth/

Chapter 6

Analysis of steganalysis software

6.1 Introduction

There exist some tools that are able to detect the presence of steganography, called steganalysis software. Some are open source, others are quite expensive. This thesis is limited to testing freely available tools, hence only treat licensed alternatives based on public available descriptions. The analysis presented here is the author's own. Where information are obtained from elsewhere, this is clearly referenced.

After the detection of hidden information, the embedded message can be tried extracted. In some situations, this means running the identified steganography software. Sometimes, a key is needed to extract the message. A brute-force or dictionary attack can be performed on such systems. An example of software aiding dictionary attacks against steganography software is also mentioned. Table 6.1 gives an overview of the software for steganalysis tested in this chapter.

Name	**Section**	**Licensing**
StegSpy	6.2	Free
Stegdetect	6.3	Open source
Stegbreak	6.4	Open source
Stego Suite	6.5	Licensed
StegAnalyzer	6.6	Licensed

Table 6.1: Steganalysis software treated in this chapter.

Usage of each tool is first described. Then the tools are studied, mostly addressing their limitations.

6.1.1 Disabling hidden information

Software for disabling hidden information is not treated. Often the embedding method is not robust to even small changes to the stego-message. For images, this could be lossy compression, resizing etc. Preventing the disabling of hidden information is a goal for watermarking schemes, for instance Digital Rights Management (drm) depend on this. In the case of the active warden from Section 5.2, the warden William could apply methods to destroy potential hidden information. From a forensic point of view, this is most likely not very interesting and software for disabling hidden information is left out.

6.2 Description of StegSpy

Name	StegSpy V2.1
Licensing	Free
Software detected	Hiderman, JPHideandSeek, Masker, JPegX and Invisible Secrets
URL	www.spy-hunter.com/stegspydownload.htm

StegSpy V2.1 is freely available steganalysis software developed by Michael T. Raggo. It claims to detect Hiderman, JPHideandSeek, Masker, JPegX and Invisible Secrets [web40]. The author as presented StegSpy at InfoSec 2004, BlackHat 2004 and DefCon 2004.

6.2.1 Usage of StegSpy

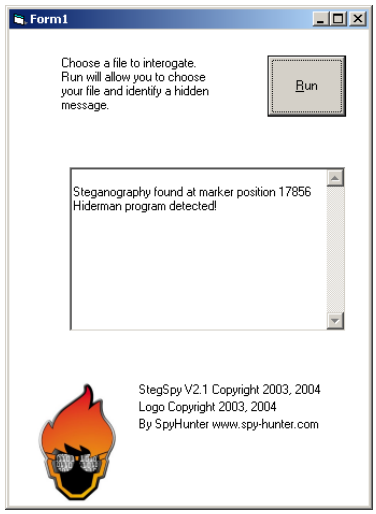
The current version, v2.1, of StegSpy is written in Visual Basic. It has a graphical interface, allowing the user to manually select a file to be examined. A screen shot is shown in Figure 6.1(a). The picture in Figure 6.1(b) is from a presentation of steganography and steganalysis by M. T. Raggo at BlackHat 2004.

StegSpy indicates in Figure 6.1(a) that there is information hidden with Hidermann , starting at position 17856. The stego-file is BMP image data. The BMP header contains a field indication the size of the file [web27]. Figure 6.2 shows top and bottom sections of the image in a hex viewer. Figure 6.2 clearly indicates that there is appended data.

Also seen in Figure 6.2(b) is the string “CDN”. This string is always present when using Hiderman [web40]. CDN is then a signature of Hidermann and is used by StegSpy to detect the steganography software used for the message embedding.

6.2.2 Examination of StegSpy

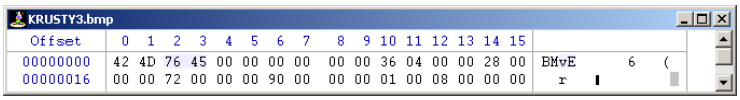
According to the author [web40], StegSpy is doing signature-based steganalysis. More is not known of how StegSpy works, but it could be assumed that it follows a similar approach as described in sections 5.4.2.2 and Sec:stega:FileSignatures,



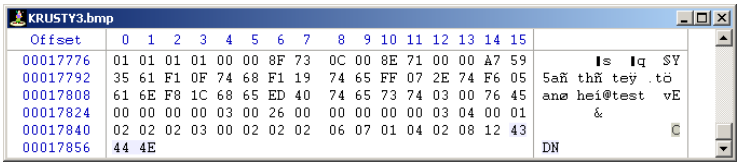
(a) Screen shot of StegSpy v2.1

(b) Stego-image of Krusty the Clown (KRUSTY3.bmp)

Figure 6.1: Using StegSpy. The tool is used to detect the presence of hidden data in *KRUSTY3.bmp*, starting at offset 17856. The steganography software is identified as Hiderman.



(a) Beginning of the header of BMP image file (KRUSTY3.bmp)



(b) End of the (KRUSTY3.bmp)

Figure 6.2: Viewing KRUSTY3.bmp in a Hex viewer. The header in 6.2(a) indicates that the size of the file is 17782 (0x4576) bytes. However, there is data past this offset, as seen in the bottom figure.

detecting file anomalies and signatures. However, when changing the "CDN"-signature to "000", StegSpy fails to detect steganography in KRUSTY3.bmp. So it seems StegSpy only relies on file signatures and ignores file anomalies for detection. And after the detection of a known signature, perform some action to locate the position of the embedded data, e.g. detecting end of image based on header information.

The graphical user interface does not allow to such a selection of files for steganography. Hence it is not very convenient for searching for steganography among a large number of possible files.

Based on the observations above, the forensic utility value of StegSpy is low. But the knowledge it contains, i.e. signatures, are quite useful and a possible action could be to create a more forensic friendly tool using these signatures.

6.3 Description of Stegdetect

Name	Stegdetect 0.6
Author	Niels Provos
Licensing	Open source
Software detected	jsteg, jphide, invisible secrets, Outguess 0.13b, F5 , appendX and camouflage.
URL	www.outguess.org/detection.php

Stegdetect is open source steganalysis software developed by Niels Provos. It can detect presence of a message embedded with Jsteg, jphide (unix and windows), Invisible Secrets, Outguess 01.3b, F5 (header analysis), AppendX and camouflage [web37].

6.3.1 Usage of Stegdetect

Listing 6.1 shows the output of running Stegdetect on a image, where a message is embedding using Invisible Secrets. Stegdetect indicates the certainty of the results with $*$-s, the more the better.

```
1 # ./stegdetect-0.6/stegdetect img1.jpg
  img1.jpg : invisible[7771](***)
```

Listing 6.1: Running Stegdetect

6.3.2 Examination of Stegdetect

How Stegdetect works is presented in Section 5.4.2.4, with a more through description of the statistical analysis by Provos found in [42]. Stegdetect is a result from the academic community. Newer theoretical steganalysis algorithms

have been suggested[1], but there exist no publicly known steganalysis software supporting these[2].

Stegdetect 0.6 also supports linear discriminant analysis to detect any JPEG based steganography system [web37]. Carrier-images and stego-images are used as a training set. A linear decision function is automatically created based on the test set and used to classify new images as stego-images or cover-images. This functionality is not examined.

6.4 Description of Stegbreak

Name	Stegbreak
Author	Niels Provos
Licensing	Open source
Software detected	JSteg-Shell, JPHide and OutGuess 0.13b
URL	www.outguess.org/detection.php

6.4.1 Usage of Stegbreak

Stegbreak is developed by the same author as Stegdetect, Niels Provos. It is however not software to detect the presence of steganography, but for message extraction. It tries dictionary attacks against JSteg-Shell, JPHide and OutGuess 0.13b [web37]. The success of Stegbreak is of course closely related to the quality of the password and the dictionary. The rules to permute words in the dictionary are also closed related to the success[3].

6.4.2 Examination of Stegbreak

From a forensic point of view, the investigation can present clues of the password, where permutations can be the password used. These clues, like name, birthday, a pet's name etc., can be added to the dictionary used by Stegbreak.

Stegbreak need a method to verify that the extracted bit string is an embedded message and not just noise. This is done by identifying file headers in the extracted bit string, as presented in Section 5.4.3.

[1] E.g. [13] detects Outguess 0.2 using higher-order statistical attacks

[2] Again, the performance of the licensed steganalysis software have not been examined

[3] Newer versions of Stegbreak does not contain rules for dictionary permutations and dictionaries. Stegdetect 0.4 for Windows comes with some rules, that can be used.

6.5 Description of Stego Suite

Name	Stego Suite
Author	Wetstone Technologies
Licensing	Licensed
Software detected	Unknown
URL	www.wetstonetech.com/

6.5.1 Usage of Stego Suite

Wetstone Technologies offers *Stego Suite*, consisting if the detection tools Stego Watch and Stego Analyst and a password cracker, Stego Break. They also offer training in using these tools, among others at the Black Hat USA 2005 conference.

How these tools perform steganalysis is not clear, also which steganography tools it detects is not known.

6.5.2 Examination of Stego Watch

Without access to the tools, it is not possible to examine them. Other sources discussing Stego Watch was not found, however, as stated above, these tools will be presented at the Black Hat USA 2005 conference.

6.6 Description of StegAnlyzer

Name	StegAnlyzer
Author	Backbone Security
Licensing	Licensed
Software detected	Unknown
URL	www.sarc-wv.com/products.aspx

6.6.1 Usage of StegAnalyzer

Backbone Security provides two version of StegAnalyzer. StegAnalyzer AS can search file systems for traces of known steganography software. StegAnalyser SS includes the functionality to detect known stego-files signatures.

6.6.2 Examination of StegAnalyzer

A copy of this software is not available, but still some thought can be made. StegAnalyzer is quite expensive; it is sold at around $2000. However, its functionality is available from other tools. Detection of file hash signatures is supported with free available tools (e.g. Sleuthkit /Autopsy), the same is detection

of known stego-files signatures. So the real value of StegAnalyzer is in the database with software- and stego-file signatures. The quality of this database is not known.

6.7 Discussion

This chapter has treated four tools for steganalysis and one for dictionary-attack against steganography. Two are licensed; hence, they were not available to the author. The two free steganalysis tools, StegSpy and Stegdetect, have been tested with varying results. StegSpy targets file signatures, i.e. it is looking for known hex-values, while Stegdetect has a more sophisticated approach. A limitation of StegSpy has been identified and an improved solution including searching for file anomalies has been suggested.

Chapter 7

Digital forensics and steganography

The very nature of steganography is to stay hidden. There are some attempts to get information of steganography encounters from investigators [web4, web24], without achieving good publicly available statistics. By ignoring steganography due to lack of statistics is "security through denial" and really not a good alternative.

It is natural to assume that steganography will or could be used, due to its characteristic of concealment, which should appeal to criminals. Therefore, if criminals are not already using steganography, the future will most likely see adoption of steganography as a tool for cyberspace criminals.

During a digital investigation, it is possible to encounter steganography, Hence, investigators should prepare for it as a part of the *Readiness phases* (from the forensic methodology from Section 2.3). To find something, you need to know what to look for. By applying automatic routine procedures searching for (hints of) steganography, otherwise undiscovered evidence might be collected from various digital media.

This chapter addresses methods to defeat steganography. It is important to note that this is not limited to pure steganalysis, other strategies, some already well-known to the digital investigator, can also be applied.

7.1 Defeating steganography

The process to defeat a steganography algorithm, steganalysis, is similar to cryptanalysis. It tries to defeat the algorithm by looking for weaknesses, these attacks are defined in Section 7.1.2. Some approaches are to use statistical properties to look for abnormalities in files, e.g. strange palettes in gif-images or other known signatures in stego-messages.

However, the investigator has other tools than just steganalysis at his disposal.

Cryptography algorithms are weak at the end host. Here security credentials are stored and algorithms (software) executed. The same holds for steganography: keys are exposed and possible carriers and software are present here. Items from the physical scene can provide security credentials in the form of written down passwords etc. Locating these artifacts can help defeat steganography.

Digital forensic investigators are, by definition, experts at finding information from the digital crime scene. Well-known forensic methods are to search for keys and passwords, known key words, recover deleted data etc. Discussion of these methods including steganalysis and their use when attacking steganography follows. It closely follows the phases introduced in Section 5.4.1. Below is an overview of the methods.

List of methods to defeat steganography:

- Physical crime scene investigation
- Steganalysis
- Detection of steganography software
- Traces of steganography software
- Locating pairs of carrier/stego-files
- Key word search and activity monitoring
- Suspect's computer knowledge
- Unlikely files
- Locating steganography keys
- Hidden storage locations

7.1.1 Physical crime scene investigation

The Physical crime scene investigation phases from the model of digital forensic in Section 2.3 is the foundation for the digital crime scene. Not just providing the digital media for further examinations, but the collection of notes and markings can yield security credentials. The famous post-it sticker with the no-longer secret passwords, can clearly healp defeat steganography.

It is also important to note that interrogation of the suspects can give away passwords. The focus for this master thesis is on digital forensics, so this and other techniques from the physical crime scene is not addressed. The following methods concentrate on the digital crime scene.

7.1.2 Steganalysis

During an investigation, the usage of steganography can be suspected or it could be routine work to look for it. Already having mentioned the importance to prepare for steganography, steganalysis can be conducted during the *Digital crime scene investigation phases*. After the Survey for digital evidence, hints of

steganography might be found or it could be suspected. Maybe the suspected evidence was not found at all. Then steganalysis should be conducted in the *Search for digital evidence*.

Section 5.4.2 presents different methods for steganalysis, they are

- File signatures
- File anomalies
- Visual attacks
- Statistical attacks

7.1.3 Detection of steganography software

An absolute indication of steganography is the discovery of steganography software. It could be located on the suspect's disk drive, or hidden away on some memory stick or cd. To automatically search of known steganography software, can be done by maintaining a database of cryptographic hash values for known components of such software. Appendix A contains such a hash database for the steganography software used in this thesis. There exists a "Steganography Application Fingerprint Database" (SAFDB) [web3], claiming to contain signatures for 230 data-hiding applications. The National Institute of Standards and Technology (NIST) maintains a list of digital signatures of software applications called the "National Software Reference Library" (NSRL) [web31]. This list also contains steganography software.

7.1.4 Traces of steganography software

When steganography software is not found, traces of its use might still be discovered. For example, the list of recently used files in winzip or winrar[1] can present evidence of recently extracting *EzStego.zip*, which is the steganography software mentioned in Section 4.2. Similar traces can be found elsewhere.

7.1.5 Locating pairs of carrier/stego-files

As known, steganography software often creates stego-messages based on an original carrier-message. Files with different hash values, but with the same perceptional properties are potential carrier/stego file pairs. E.g. two images looking similar, but with slightly different LSB-planes. Even if the carrier file was deleted, it can in some cases be undeleted using forensic tools.

Characteristics of images containing child pornography are collected and used for automatic detections. The algorithms calculating these distinctiveness's are probably more robust than just cryptographic hash values, hence small modifications will not render the images unrecognizable. Therefore, the modifications

[1] winzip and winrar are software to extract compressed files.

from steganography software yielding two slightly different versions can be detected automatically using the same algorithms. The same principle can be applied to other media types used for steganography.

7.1.6 Key word search and activity monitoring

Similar to the database of hash signatures for steganography software, a dictionary of key terms can be compiled and searches can be done to try to locate these on the seized data. The search for key words is not something only done with steganography, the contents of the dictionary decides the target.

The rate of false positives will depend on the words used, but good candidates are software names like *Outguess* and words like carrier, cover, etc [web2]. To create a part of the database of key words, *strings*[2] could be used to extract words from steganography software binaries.

Besides searching for specific key words, internet activity of the suspect can provide some answers. History logs in the web browser might show visits to steganography web sites. Therefore, the digital forensic team could keep a list of such web sites.

7.1.7 Suspect's computer knowledge

It might be tempting to use the believed computer knowledge of the suspects to assume whether or not usage of steganography is likely. This is however a dangerous thing to do, since most steganography software, as some demonstrated in Chapter 4, are fairly easy to use. On a general basis, instead of assuming knowledge too low for steganography, it is more tempting to state that the suspect's computer knowledge and resources are at such a level that steganography has to be suspected.

The suspect's computer knowledge might then again be used when speculating whether homemade steganography tools or algorithms are being used. A suspect with high computer skills, might make the investigators extra alert toward hidden data. Unknown software encountered during the investigation should of course be looked into to discover their functionality.

7.1.8 Unlikely files

Some steganography software uses or creates uncommon file types. The mandelsteg tool (See Section 4.3) uses no existing carrier, but creates a mandelbrot image based on the message to be embedded. The investigator should ask himself what use the suspect might have of such images. They clearly stand out among vacation images. A similar example is inconsistencies between religion, interests, etc. and file types. Consider the speculation of Al-Quida hiding information in pornographic images. Such images are against the Muslim religion and findings of this kind have to be considered unlikely and suspicions.

[2] See man pages for more on *strings* or www.rt.com/man/strings.1.html

7.1.9 Locating steganography keys

Some steganography software uses steganography keys (stego-key) to seed a pseudo-random number generator used when selecting locations for bit manipulations. During the Physical crime scene investigation phases, handwritten notes or markings would be collected. These could be passwords used by the suspect, and should be checked.

Method of guessing bad passwords with dictionary attacks and other brute force methods from cryptanalysis can be applied to steganalysis. Section 6.3 treats *Stegdetect* which tries a dictionary attack against some steganography systems. One problem encountered by the steganalyst (and Stegdetect), is when the embedded encrypted message has been stripped of headers (see Section 5.4.3). Then the extracted message is a random looking bit string, hence it is difficult to asses whether it is valid chipertext to be treated further or just noise.

A method to brute force the stego-key independently of the cryptographic key (crypto-key) is shown in [17]. However, the same article provides techniques to prevent this method, making it possible to adapt the steganography algorithm.

Encryption of the message to be embedded is sometimes provided by the steganography software. These keys used for encryption are more correctly called a crypto-key, since the embedding of the message, done by the steganography algorithm, does not depend on this key. How to handle an encrypted message, cryptanalysis, is not treated in this master thesis.

7.1.10 Hidden storage locations

There exist steganographic file systems and otherwise hidden data partitions and alternate data streams[3]. This should be looked for when examining the file structure of the seized storage media [43].

7.2 Anti-Forensics

Not trying to provide means for illegal behavior, it is useful to be aware of techniques applied when attempting to withstand digital forensic analysis. This best practice for anti-forensic is not necessary unique to steganography. This list is not attempting to be complete, just presenting some ideas.

7.2.1 Choice of passwords

The literature is full of ideas on which passwords are most secure. Weak passwords can be attacked in a brute-force matter, and there exist software supporting such attacks on steganography. As stated above, steganography is weak at the end hosts and the location of the password(-s) is contribution to this. Using

[3] Alternate Data Streams or 'ADS' are special data streams existing in NTFS data streams. These ADS can store files that are invisible to the user [web47].

good passwords, following points from the literature like no yellow post-it stickers and pet's name, and keeping it secret is necessary to withstand the simplest investigation.

7.2.2 Remove the carrier-message

The carrier-message should be removed completely from the system, since the comparison of cover-message with stego-message breaks steganography and also could give away the embedded message.

7.2.3 Hide the existence of steganography software

A problem arises when the computer is confiscated and found to contain steganography software. All traces of the software and usage of it should be removed. Examples of easily forgotten locations are the Windows registry, in the recently used file list in tool used for software extraction and the history of the web browser. Running the tool from a floppy or USB dongle, which is hidden when not in use, could be a good idea.

7.2.4 Remove headers from encrypted messages

All messages should be encrypted prior to embedding. However, encryption schemes add a header to the encrypted message. This header could contain data like encryption algorithm etc. This plaintext can aid steganalysis by simply stand out amount random looking data or be used to identify a successful brute force attack on a secret key steganograpy system. For instance, *Stegdetect* relies on detecting known headers in the extracted message to signal success.

Long steganography keys should be used and the steganography algorithm should adapt the countermeasures described in [17] to defeat the stego-key search also described in [17].

7.3 Summary

The methods to defeat steganography are repeated in Table 7.1. These techniques are proposed by this master thesis as a result from applying known forensic methods to steganography and from the particular properties of steganography.

	Name of method
1.	Physical crime scene investigation
2.	Steganalysis
3.	Detection of steganography software
4.	Traces of steganography software
5.	Locating pairs of carrier/stego-files
6.	Key word search and activity monitoring
7.	Suspect's computer knowledge
8.	Unlikely files
9.	Locating steganography keys
10.	Hidden storage locations

Table 7.1: Forensic methods to defeat steganography

Chapter 8

Digital forensic cases

Based on the forensic methodology defined in Section 2.3, investigation cases will be examined in this chapter. Forensic software and processes will be used, addressing their value to detect steganography software and usage of such. Some of the steganalysis software described in Chapter 6, will also be demonstrated.

8.1 Introduction to the cases

In the scenarios, the digital crime scene investigation phases are the important phases. It is assumed that the investigator is prepared with both knowledge and tools from the Readiness phases, as described in Chapter 7. The Deployment phases is assumed finished, so is the Physical crime scene investigation phases i.e. the material is handed over to the digital forensic expert, ready for processing. These phases of however mentioned, since they provide useful background information on each case.

When dealing with each case, the Document evidence and scene phase is not treated. The process of bringing evidence to court is extensive, and requires a lot of documentation and knowledge of the legal system. The accounting of each case here does not try to be this extensive. The purpose is to address issues regarding steganography and not legal matters.

8.1.1 Summary of methodology and tactics

Some of the information from previous chapters are summarized here.

From the digital forensic model, The Digital crime scene investigation phases are repeated below:

Preservation of dig. scene Involves securing and preserving the digital scene.

Survey for dig. evidence Finding the obvious pieces of evidence.

Document evidence and scene Documenting the evidence from the previous phase.

Search for dig. evidence A more thorough analysis of the digital scene.

Dig crime scene reconstruction Putting the pieces together, testing and rejecting/accepting theories.

Presentation of dig scene theory Presenting the digital evidence found.

Chapter 7 addresses methods for the investigator to defeat steganography. The list of methods to defeat steganography consists of the following:

- Physical crime scene investigation
- Steganalysis
- Detection of steganography software
- Traces of steganography software
- Locating pairs of carrier/stego-files
- Key word search and activity monitoring
- Suspect's computer knowledge
- Unlikely files
- Locating steganography keys
- Hidden storage locations

The methods above will be the foundation for investigating the presence of steganography, i.e. to defeat it. After the detection of steganography, methods to extract the embedded message will be attempted.

8.2 Digital forensic case 1 Honeynet Scan of the Month 26

This case serves two purposes. First as an introduction to the combination of the forensic methodology from Section 2.3 and forensic software from Section 2.4. Secondly, due to the supplied background material for the challenge, it introduces forensics and steganography together. Hence, challenge 26 from the Honeynet project is a perfect start. It is also worth noticing, that analysis logs and results from other participants are available [web10, web25, web7]. These answers to the challenge provides evaluation material and hints.

8.2.1 Introduction to "Scan of the Month"

The Honeynet Project is a non-profit organization dedicated to improving the security of the Internet by providing cutting-edge research for free. The strive to raise awareness, provide teaching (e.g. the Scan of the Month) and research tools and methods. The "Scan of the Month" (SotM) challenges provide sample

cases for the security community to improve their forensic and analysis skills [web19].

8.2.2 Challenge 26

Challenge 26 (SotM 26) is a continuance of a previous challenge (SotM 24). In SotM 24, the task was to analyse a floppy disk recovered from a drug dealer (Joe Jacobs). There is a police report explaining the situation of SotM 26, with the challenge for the digital forensic investigator is to analyse another floppy, this time recovered from Joe's "computer savy" supplier, Jimmy Jungle [web36].

The resulting report shall try to answer the following questions:

1. Who is the probable supplier of drugs to Jimmy Jungle?
2. What is the mailing address of Jimmy Jungle's probable drug supplier?
3. What is the exact location in which Jimmy Jungle received the drugs?
4. Where is Jimmy Jungle currently hiding?
5. What kind of car is Jimmy Jungle driving?

And there is a bonus Question: "Explain the process that was performed so that there were no entries in the root directory and File Allocation Table (FAT), yet the contents of each file remained in the data area?"

8.2.3 Investigating the case

Next follows the account of the digital investigation of SotM 26, following the phases from the forensic methodology used in this thesis.

8.2.3.1 Deployment phases

The results from the Deployment phases are explained in the description of SotM 26. Based on the results from previous investigations (i.e SotM 24), warrants have been given to search the apartment of Jimmy Jungle. This is described in the police report following SotM 26.

8.2.3.2 Physical crime scene investigation phases

Also described in the police report is the results from the search in Jimmy Jungles apartment. The police found a floppy disk, labeled with the writing: *dfrws.org*. The floppy is treated as a digital crime scene for further analysis.

8.2.3.3 Digital crime scene investigation phases

The Digital crime scene investigation phases is the main target of this master thesis. It is in these phases the knowledge the thesis provides comes in use and can be evaluated.

8.2.3.3.1 Preservation of dig. scene An image of the floppy is created, i.e. downloaded from honeynet.org [web36]. Cryptographic hash values are created to make sure the image is identical to the original.

A new case is created with Autopsy, named "scan26". A host is added, "floppy", and the image "scan26" is added as a FAT12 partition, with drive letter "A:". At the same time, the known md5 value of the image is added and verified after copying to the evidence storage (figures 8.6(a) and 8.6(b)).

8.2.3.3.2 Survey for dig. evidence The case being marked to contain steganography, a quick search for known steganography software is tempting, together with a scan with steganalysis software for stego-messages.

To be able to read the contents of the image, it is mounted as a read-only loop device[1], as shown in Listing 8.1. However, when navigating into the directory it appears to be empty. When looking at the floppy image through "file analysis"-view in Autopsy, this is confirmed, Figure 8.6(c).

```
12 # mount -ro,noatime,loop scan26 /mnt/scan26
   # ls -la /mnt/scan26/
14 total 15
   drwxr-xr-x  2 root root 7168 Jan  1  1970 .
16 drwxr-xr-x  6 root root 4096 Jun 13 16:57 ..
```

Listing 8.1: Mounting the floppy image as a "read only" loop device. Listing the content with *ls*, the floppy appears to be empty.

Based on above, the floppy is empty or contains only hidden data. Techniques that are more sophisticated are needed, these are provided through Sleuthkit. More on this in the "Search for dig. evidence" phase.

The floppy was labeled with "dfrws.org". This is a possible url and when tried, `http://dfrws.org/` leads to the web page of Digital Forensic Research Workshop (DFRWS). An interesting web page, considering the circumstances. But no clues regarding this case where discovered when surveying the web page.

8.2.3.3.3 Document evidence and scene All actions and findings need to be documented, i.e. Chain of Custody. This extensive process of documenting actions and findings for court is not treated, since the focus for this scenario is on the technical aspects and not the legal.

8.2.3.3.4 Search for digital evidence A more thorough search for digital evidence will be done here. The Autopsy toolkit will be used, the different screen captures are shown in Figure 8.6 and referred to in the following account.

Starting of with a closer inspection of the image, with the "image details"-view. This presents details of the file system on the floppy and Listing 8.2 shows the truncated results.

[1] The loop device is a device driver that allows an image file to be mounted as though it was a block device.

```
1  FILE SYSTEM INFORMATION
   File System Type: FAT12
3
  OEM Name: RVRbIHC
5  Volume ID: 0x16da0644
   Volume Label (Boot Sector): NO NAME
7  Volume Label (Root Directory):
   File System Type Label: FAT12
9
   Sectors before file system: 0
11
   File System Layout (in sectors)
13 Total Range: 0 - 2879
   * Reserved: 0 - 0
15 ** Boot Sector: 0
   * FAT 0: 1 - 9
17 * FAT 1: 10 - 18
   * Data Area: 19 - 2879
19 ** Root Directory: 19 - 32
   ** Cluster Area: 33 - 2879
21
  METADATA INFORMATION
23 Range: 2 - 45554
   Root Directory: 2
25
  CONTENT INFORMATION
27 Sector Size: 512
   Cluster Size: 512
29 Total Cluster Range: 2 - 2848

31 FAT CONTENTS (in sectors)
```

Listing 8.2: File system details of the floppy image. It is a FAT file system and the sectors for each part is identified, as described in Figure 8.1.

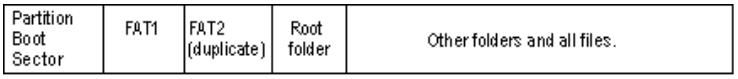


Figure 8.1: FAT file system organisation of a volume [web14].

Some knownledge of the FAT file system is needed at this point [2]. Figure 8.1 shows the organisation of a volume with FAT.

With the knowledge from Figure 8.1 and Listing 8.2, a closer analysis of the floppy image can be done. Using the "Data unit"-view to examine the different sectors from the image. Primary Fat, sector 1-9 yields only f0ffff00 0000000 ... I.e. media id, fill and the rest 0's. The secondary FAT is, as expected, identical and show in Figure 8.6(e). The Root section, sector 19 -32, is also empty, i.e. only 0's. So far, an empty floppy.

When looking at sector 33, the Data area, there is finally some data. The Hex display is shown in Figure 8.6(f) and yields strings like JFIF, SNuG etc. This data is not allocated. Autopsy can extract data from unallocated sectors of an image. Running the *strings* tool on the extracted data, yields all strings found.

Running the keyword search with a regular expression[3] matching all strings,

[2] More information on Fat file system can be found at `http://www.ntfs.com/fat-systems.htm`

[3] A regular expression (abbreviated as regexp, regex or regxp) is a string that describes or matches a set of strings, according to certain syntax rules [web51].

`[[:alnum:]{4,}]`, on the unallocated data strings yields 452 hits. This list is quite long and, so searches limiting the results can be made or other text editors can be used to view the results. Listing 8.3 shows the result from *tail* on the file created when running *strings* on the unallocated data.

```
# tail scan26-0-0-fat12.unalloc-dls.asc
      31772 " "
      31777 " "
      31782 " "
      31787 " "
      31862 h%ad
      31945 H:qV
      32006 kcpkt
      32183 kA$4
    1210704 pw=help
    1385824 John Smith's Address: 1212 Main Street, Jones, FL 00001
```

Listing 8.3: Output from running *strings* on unallocated data. Using *tail* on the file containing the results.

The results from Listing 8.3 are quite interesting. *pw* could be a password and we got John Smith's address.

Returning to the data found in the Data Area-view. The string "JFIF" has been encountered earlier, when looking at images with a hex-viewer[4]. Searching Google for JFIF it confirmed that JFIF indicated a jpeg-file, i.e. an image. Figure 8.6(f) also indicates jpeg image data. It would be interesting to view this image.

"File type" view uses the *sorter* tool to extract files and organize them according to file type. A useful option in our case is to extract (save) graphic images and make thumbnails. The tool also validates file extension with file type. And it will alert of files found to be in the "Alert Hash Database", e.g. if the files belongs to known steganography software. The screen shot is found in Figure 8.6(g).

The results are 3 files (unallocated), but they were skipped as "non-files". So the *sorter* tool did not help. But the string "JFIF" indicated the presence of a file. So the next attempt is to extract this file from the image using *dd*.

Viewing scan26 in a hex-editor (Figure 8.2) and searching for JFIF yields it at 0x4206h. Asking Google provides information about JPEG file format[5]. A JPEG file begins with (hex): ffd8 ffe0 0010 4a46 4946 and ends with ffd9. This helps locating the start of the file at 0x4206h and the end at 0xc158h.

Listing 8.4 shows the *dd*-tool to extract the JPEG file. The values has to be converted from hexadecimal to decimal. The first 16896 bytes are skipped and a total of 32602 is read. The image extracted is shown in Figure 8.3(a).

[4]When testing different software for steganography for Chapter 4, cover-and stego-images were examined with a hex-viewer to study changes.

[5]More info on the JPEG file format can be found at www.obrador.com/essentialjpeg/headerinfo.htm

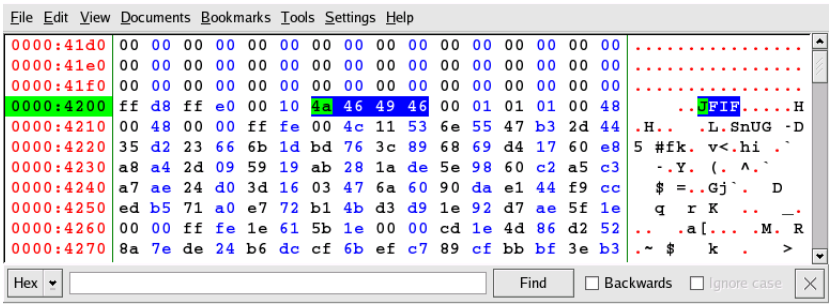
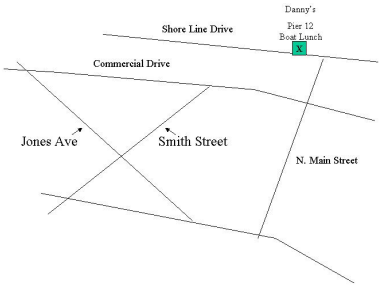


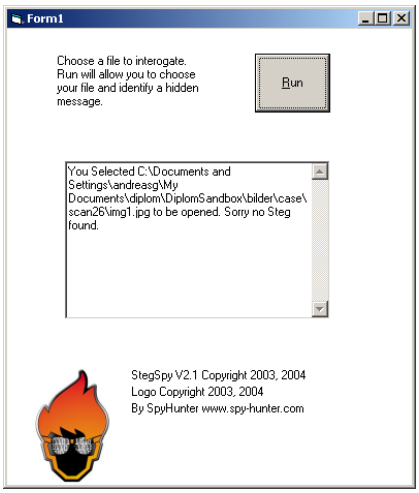
Figure 8.2: Viewing the unallocated data from a hex-editor.

```
608 # dd if=~dust/scan26 ibs=1 of=img1.jpg skip=16896 count=32602
    32602+0 records in
610 63+1 records out
```

Listing 8.4: Extracting an image file from scan26. The first 16896 bytes are skipped and a total of 32602 is read.



(a) The first image extracted: *img1.jpg*



(b) Running StegSpy on *img1.jpg*

Figure 8.3: The first image extracted: *img1.jpg* is shown in Figure 8.3(a). The image provides a map and a location: "Dannie's Pier 12 Boat Lunch". Figure 8.3(b) shows a negative result of running StegSpy V 2.1 on *img1.jpg*.

Detecting Steganography Finally having an image, *img1.jpg*, steganlysis software from Chapter 6 can be put to use. StegSpy v2.1[6] is tried, with the result of not finding presence of steganography (Figure 8.3(b)). Stegdetect[7] is tried with the results in Listing 8.5.

```
# ./stegdetect-0.6/stegdetect img1.jpg
img1.jpg : invisible[7771](***)
```

Listing 8.5: Running Stegdetect on *img1.jpg*.

Stegdetect strongly indicated in Listing 8.5 that the image is a stego-image and that the embedding is done with *Invisible Secrets*. An interesting observation is that StegSpy claims to detect messages hidden with Invisible Secrets.

Message extraction To extract the message, *Stegbreak*[8] could be tried. Of course, the success of brute-force depends on a weak password. But it is worth a try. The problem is just that stegbreak does not support attacking Invisible Secrets. Manually trying to extract the message with all algorithms provided by Invisible Secrets and the possible password "help", is not successful.

A hint after examining the submissions to SotM 26, was to check out the source code for dfrws.org[9]. Listing 8.6 shows what was found.

```
<!-- 100 guest rooms have been reserved at a special conference rate of
     -->
<!-- Invisible Secrets --><!-- $149.00 per night for non-government-->
     <!-- http://www.invisiblesecrets.com -->
<!-- PW=lefty -->Please honor this pricing arrangement and
<!-- Algorythm= twofish -->In order to ensure room availability
```

Listing 8.6: Extract from the HTML source of dfrws.org.

Armed with more possible passwords: lefty and right, and the Twofish algorithm, more attempts with Invisible Secrets are done. The successful result with the password "lefty" was the file "john.doc". The word document needs a password and "help" was the correct one. The complete text from the document is shown in Listing 8.8. The following quote from "john.doc" is interesting:

> [Jimmy Jungle:] "take a look at the map to see where I am currently hiding out."

So far, this map is not located.

The second file After the end of the JPEG file, there are 0's until "42 4d" is encountered (Figure 8.4). Searching for "42 4d file forensic" with Google , gives

[6] Described in Section 6.2
[7] Described in Section 6.3
[8] Described in Section 6.4
[9] Hiding of information in HTML comments was mentioned as a simple example at the authors presentation of steganography at [18]. It was however not considered when surveying the web page. The passwords are no longer available, and the source listing is from one of the supplied answers to SotM 26 [web7]

us knowledge of the file type: BMP [web27]. The next four bytes gives the total size of the file, little Endian. 0x11cc77 is 1166454 in decimal. Once again *dd* is used to extract the image, as shown in Listing 8.7.

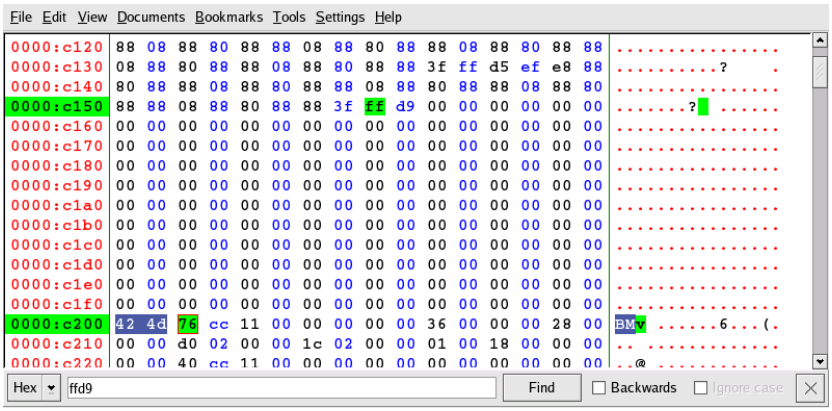


Figure 8.4: Hex view of data. Showing the end of the JPEG file and start of the BMP file.

```
# dd if=scan26 ibs=1 of=img2.bmp skip=49664 count=1166454
1166454+0 records in
2278+1 records out
```

Listing 8.7: Extracting the second image from scan26. Based in information from Figure 8.4, the first 49664 (0xc200) bytes are skipped and a total of 1166454 (0x11cc76h) bytes are read.

The BMP image extracted is shown in Figure 8.5. It it similar to img1.jpg, but this one contain the location X marked as a hideout. This could be the hideout Jimmy Jungle referred to in "john.doc".

Detecting Steganography The search for steganography continues, but Stegdetect can not target BMP files. Stegspy was tried, but could not detect the presence of steganography. Steganography software, Invisible Secrets, has already been identified, so it is worth a try to extract an embedded message.

Message extraction With Invisible secrets already being used and an unused password, it had to be tried. Using the same algorithm (Twofish) and the password "right", yields "Jimmy.wav". When played, Jimmy says there is a meeting at the pier tomorrow and that he is driving a 1978 Blue Mustang.

8.2.3.3.5 Dig crime scene reconstruction The crime scene is not reconstructed.

8.2.3.3.6 Presentation of dig. scene theory Based on the above documented findings, answers to the questions from SotM 26 are given.

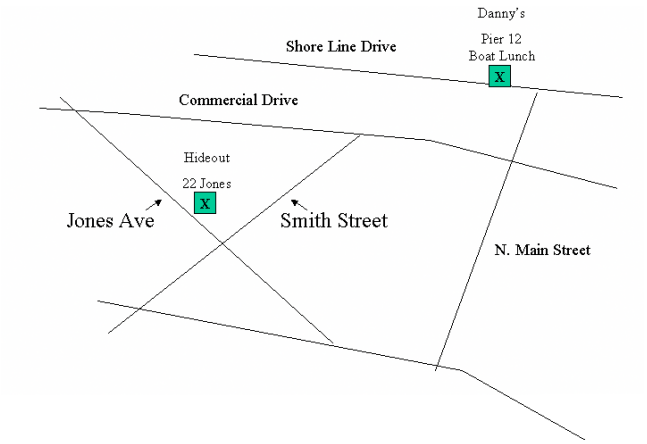


Figure 8.5: The second image, *img2.bmp*, indicates a hideout at 22 Jones Ave.

Who is the probable supplier of drugs to Jimmy Jungle? Based on the letter extracted, john.doc, the probable supplier is John Smith. Complete letter shown in Listing 8.8.

What is the mailing address of Jimmy Jungle's probable drug supplier? Listing 8.3 presents the address of John Smith: "1212 Main Street, Jones, FL 00001".

What is the exact location in which Jimmy Jungle received the drugs? Based on the data found, i.e the letter and the map (img1.jpg, Figure 8.3(a)) , the drugs are delivered to "Dannie's Pier 12 Boat Lunch".

Where is Jimmy Jungle currently hiding? From the letter, Jimmy Jungles hiding place is indicated on a second map (img2.bmp, Figure 8.5). The address of the hideout is "22 Jones Ave".

What kind of car is Jimmy Jungle driving? The last file, *Jimmy.wav*, is a recording where Jimmy is saying he drives a 1978 Blue Mustang.

To answer to the bonus question, knowledge of how formating of a disk is done. When performing a "quick format" of a floppy under Windows, only the root directory and fat entries are deleted, and the real data is not [web10]. I.e. the pointers telling where files are allocated on the floppy are removed, but the data can still be manually retrieved, as demonstrated above.

8.2.4 Discussion and summary of SotM 26

All questions stated in the case description are answered, after some hints [web7, web10, web25, web28]. The forensic methodology defined earlier is put to use, and it serves it purpose well. The quick survey for evidence did not present any evidence, but it identified the next necessary steps. Files was identified and extracted, and with the help of steganalysis software, identified as a stego-message. The steganography software used to hide a word document was also identified and together with the found passwords, successfully extracted.

```
1 Dear John Smith:

3 My biggest dealer (Joe Jacobs) got busted. The day of our scheduled
      meeting, he never showed up. I called a couple of his friends and
      they told me he was brought in by the police for questioning. I'm
      not sure what to do. Please understand that I cannot accept another
       shipment from you without his business. I was forced to turn away
      the delivery boat that arrived at Danny's because I didn't have the
       money to pay the driver. I will pay you back for the driver's time
       and gas. In the future, we may have to find another delivery point
       because Danny is starting to get nervous.

5 Without Joe, I can't pay any of my bills. I have 10 other dealers who
      combined do not total Joe's sales volume.

7 I need some assistance. I would like to get away until things quiet down
       up here. I need to talk to you about reorganizing. Do you still
      have the condo in Aruba? Would you be willing to meet me down there
      ? If so, when? Also, please take a look at the map to see where I
      am currently hiding out.

9 Thanks for your understanding and sorry for any inconvenience.

11 Sincerely,

13 Jimmy Jungle
```

Listing 8.8: The complete letter to John Smith from Jimmy Jungle.

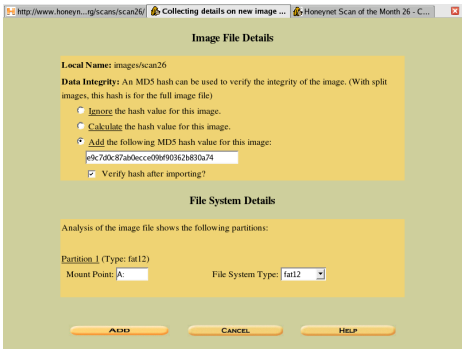
Passwords was also tried hidden, but once the method was discoved, it was easily broken.

Some of the techniques for defeating steganography was used in this scenario, but not all. An interesting observation can be done from the source code from dfrws.org (Listing 8.6), notice the name of the steganography software. The "Key word search" method (Section 7.1.6) would most likely, or even should, contain names of known steganography software. Performing such a search on all discovered media, could in some cases indicate the presence of steganography or other useful information. Like in this case, where it would have identified possible passwords, algorithm and steganography software.

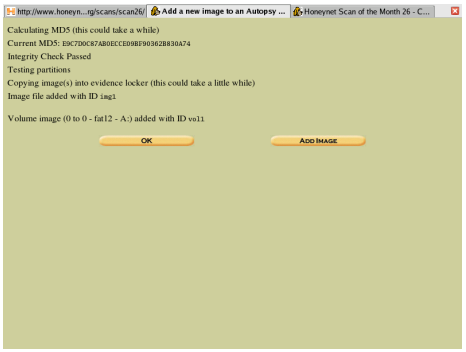
When dealing with larger amounts of data, some of the techniques used in this case need to be improved. Manually searching through a 40 gigabyte disk image with a hex viewer is time consuming. The next scenario encounters this challenge.

In this case, StegSpy and Stegdetect was tried with different results. StegSpy claims to detect Invisible Secrets, but fails. This can be due to an update of Invisible Secrets. The usability of StegSpy on larger amounts of data, say pictures by hundreds, is low. The user can only select one image at the time for analysis. Stegdetect performs better, at least against the encounted steganography software, and has better support for multiple images.

Autopsy was used in this case, but without any great success. More basic tools, like the hex viewer and *dd*, were enought to be successful.

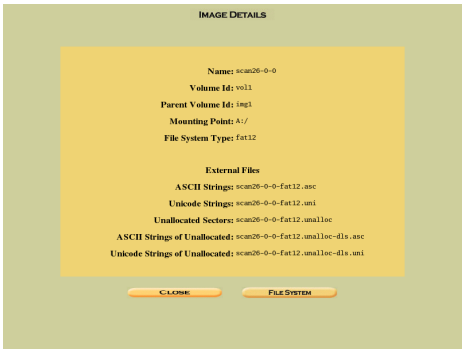


(a) Adding the floppy image to the case

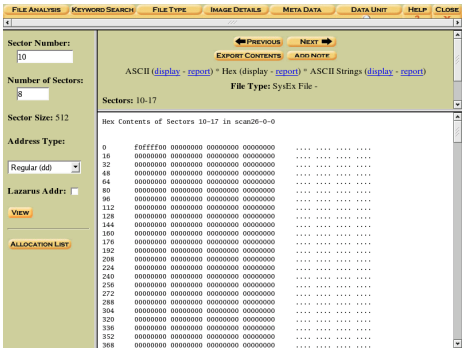


(b) Results of adding the floppy image

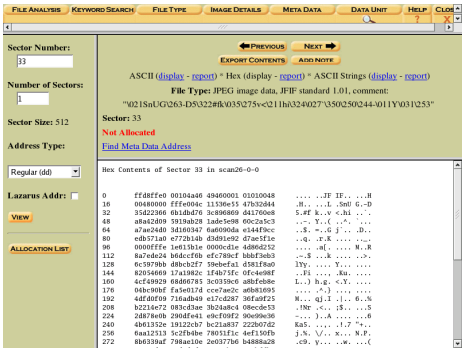
(c) File analysis yielding a seemingly empty floppy



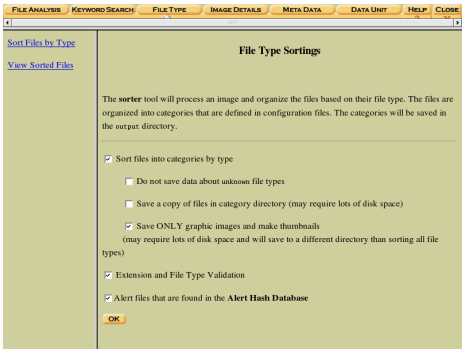
(d) Image details after extracting using *strings*



(e) Data unit viewer, Hex contens sector 1018



(f) Data unit viewer, Hex contens sector 33. Indicating jpeg image data



(g) "File type"-view uses the *sorter* tool to extract files. It is possible to limit the extraction to images and create thumbnails.

Figure 8.6: Using Autopsy for case 1: Honeynet Scan of the Month 26.

8.3 Digital forensic case 2

Case 2 will differ from the previous in data volume. The scenario involves a laptop owned by the author. During the work resulting in this master thesis, steganography software has been tested on this laptop and web pages with steganography contents have been visited. The traces in the laptop creates a useful scenario for testing the methods from Chapter 7.

This scenario is thought a part of a bigger case. There is some criminal activity and, without going into details, this results in the necessary warrants to the seizure of physical components, and the computer components is handed over to digital investigation, as illustrated in Figure 8.7.

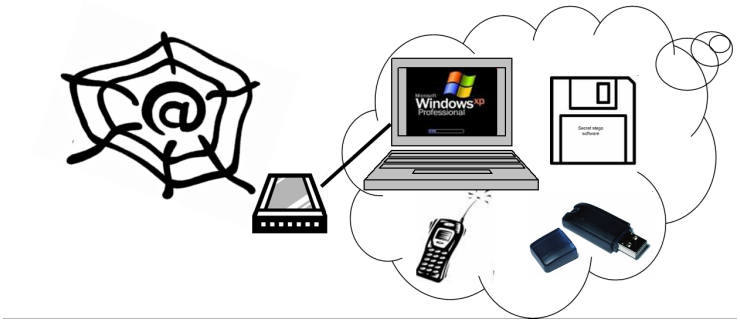


Figure 8.7: Digital forensic case 2.

Figure 8.7 shows a laptop running Windows XP. It has a connection to the Internet and there seized material also contains a usb memory stick, cell phone and a floppy. This is material which has to be expected, so methods on how to deal with them have had to have been addressed in the Readiness phases.

8.3.1 Case limitations

Some limitations are made from the case description above and in Figure 8.7. These are addressed here.

The spiderweb in Figure 8.7 represent the Internet. That would mean examination of network traffic, if this were logged or monitored. As mentioned in Section 3.7, there exist steganography software which hide communication in TCP/IP header fields. This is not addressed here, other than detecting software or other hints of such activity on the laptop itself.

External storage media like floppies, memory sticks or similar are normal. They come in many forms and some have storage capacity measured in gigabytes. Detecting and processing such media is important. Software can be run directly from these, leaving few (if any) traces on the laptop. But the methods used to analyse them are not different from analysing a normal disk drive.

Newer cell phones also comes with unelectable storage space, and MMS can be used to exchange illegal material. With the proper equipment and knowledge,

the memory on such devices can also be threated [34]. However, this is not done here.

8.3.2 Investigating the case

Next follows the account of the digital investigation of *Case 2*, following the phases from the forensic methodology used in this thesis. The account is not as extensive as for *Case 1*, due to the amount of space this would occupy in the thesis.

8.3.2.1 Deployment phases

This case is a created scenario and tries to capture general aspects of forensic cases. Hence a comprehensive description of the setting is not given.

8.3.2.2 Physical crime scene investigation phases

Figure 8.7 shows the physical components from the physical crime scene. Limitations are already described in Section 8.3.1.

The output from the Physical crime scene investigation could yield security credentials. For it to yield a password known to be a steganographic key, would surely indicate the presence of steganography. Not addressing how a password could be known to be a stego-key, it is stated that possible passwords should be documented and tried as passwords for encountered encryption and steganography. And not forget that the same password could be used more than one time.

In this case, there exist no such password clues. Nevertheless, several suspicion books were identified as treating steganography [20, 26, 35, 48] and several proceedings from Information hiding workshops [1, 4, 31, 36, 39]. These findings clearly indicated the suspect's interest for steganography. They do not however state which, if any, steganography tools are used.

8.3.2.3 Digital crime scene investigation phases

The following accounts for the digital crime scene investigation phases.

8.3.2.3.1 Preservation of dig. scene In the scenario of *Case 2*, the plug has been pulled, i.e removing the battery on the laptop. Whether or not to pull the plug is an important choice, and depends on the setting. [27] has more on this subject. Examples of when not to turn of the system could be when computer memory also is wanted collected or there is a high demand on system uptime.

It is important to make the copy identical to the original, and proving the fact. The *dd*- tool can be used to create images of file systems. It allows reading and

writing to disk devices directly, without mounting the device first. Comparing the hash values of the original disk and the copy image can prove identical versions.

The original plan was to boot with the F.I.R.E (Forensic and Incident Response Environment) distribution [web43] with ships with an extended version of dd called *dcfldd* [web18]. dcfldd has additional features that are useful for forensics, like hashing on the fly and status output.

There are alternatives to using F.I.R.E. Other cd-rom or floppy bootable distributions, like Knoppix, Trinux and PLAC can be used. But F.I.R.E has the advantage of shipping with more security tools. Knoppix has better hardware support then F.I.R.E, and was needed to detect and format the external hard disk used in the forensic case. F.I.R.E also does not ship with the newest version of some programs (i.e. Autopsy/Sleuthkit).

The data need to be transfered from the confiscated system to a storage media, from where the data can be examined further. As stated in this scenario, the seized equipment is a laptop.

A overview of the situation is given in Figure 8.8, where an external storage media is connected to the laptop, The disk drive of from the laptop can also removed from the laptop and connected directly to a forensic workstation for imaging or use special hardware like the Image MASSter Solo Forensic unit. Due to limited hardware resources, the drive image is created with the disk still in the laptop.

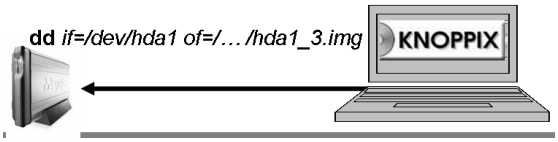


Figure 8.8: Collection and preservation of possible evidence.

Listing 8.9 shows the *dd* command creating an image of the acquired hard disk on an external disk drive. *hda1* is the block device representing the disk of the laptop and *sda1* is the Maxtor external disk drive. *hda1_3.img* tells that this image of hda1 is version three.

```
# date
Tue May 24 19:50:15 EDT 2005
# uname -a
Linux Knoppix 2.4.27 #2 SMP Mo Aug 9 00:39:37 CEST 2004 i686 GNU/Linux
# dd if=/dev/hda1 of=/mnt/sda1/img/hda1_3.img bs=1024k noerror,sync
38146+1 records in
38146+1 records out
39999504384 bytes transferred in 50073.867797 seconds (798810 bytes/sec)
```

Listing 8.9: Transferring data using *dd*. This is also the start of the log file, hence the data and identification of the system. *noerror* and *sync* ensures that *dd* continues after read error and pad error blocks with 0's to match the input[11]

.

[11] [web8] has more information on using *dd*

The Maxtor disk is connected with the IEEE-1394 FireWire specification. Even though this is a fast connection, transmitting data at up to 400 megabits per second, it still takes some time to transfer 40GB. Remember that the hole disk, including slack space, unallocated areas and swap files, is transfered, and not just the individual files. Deleted and otherwise hidden files are also of interest to the investigator.

There exist other methods and tools for gathering digital evidence. They all have their cons and pros, depending on the situation. Some of these are introduced earlier in Section 2.4.1.

The copied evidence need to be 100% equal to the original. To verify this, a hash algorithm as described in Section 2.3.1.3, can be used. To be compatible with autopsy/sleuthkit, MD5 is used. SHA1 is introduced to have a stronger alternative due to the collision weakness mentioned in Section 2.4.

Listing 8.10 shows the output of running md5sum and sha1sum on the original block device and the acquired image.

```
# md5sum /mnt/sdc1/img/hda1_3.img; md5sum /dev/hda1
7ef0df1423342b5936992a9eb37927eb  /mnt/sdc1/img/hda1_3.img
7ef0df1423342b5936992a9eb37927eb  /dev/hda1
# sha1sum /mnt/sdc1/img/hda1_3.img; sha1sum /dev/hda1
0175d4aec11b203eb16cb175c5d4e381a8e05e1c  /mnt/sdc1/img/hda1_3.img
0175d4aec11b203eb16cb175c5d4e381a8e05e1c  /dev/hda1
```

Listing 8.10: Authentication of the transferred data using hash signatures.

For the analysis of the acquired image, there exist a variety of tools[12]. Going back to the original plan of using the F.I.R.E. distribution could be done, with the disadvantage of not using the latest versions of Autopsy/Sleuth kit. A master thesis on reconstruction [54] presents *VMware* [web20] as an alternative environment. So, creating a Linux virtual machine in VMware to run the forensic tools is favorable due to the limited number of computers.

Another interesting observation is that USB 2.0 is not supported by the available Linux distributions and VMware does not support FireWire. Since the amount of data being analyzed is huge (40GB), a fast connection is preferred. From the available hardware, the solution was to boot with Knoppix and install the newest version of autopsy/sleuthkit.

After the intensive (both cpu and disk communication) operations have been performed and stored, VMware is convenient for the later stages.

8.3.2.3.2 Survey for dig. evidence Before starting the examinitaion the lab has to be set up. In this case, the forensic workstation is virtual. A virtual machine with Fedora Core 3 is created in VMware and the newest versions of Sleuthkit and Autopsy are installed. The case created with Knoppix is made available and the survey can begin.

It is sometimes difficult to separate the survey phase from the search phase. Usage of steganography is by it self not illegal[13], hence it is a supporting tool.

[12] Some analysis tools are addressed in Section 2.4.3

[13] Might not hold for all countries.

The survey detects illegal pictures, while stegnalysis detects the hidden ones. This would indicate that techniques to detect steganography belongs to the search phase.

Listing 8.11 shows the commands to mounting the hard disk image. First the external disk, *sda1*, containing the image, and then the image it self as a loop device. The mounting of the image is not required when using autopsy.

```
# mount /dev/sda1 /mnt/sda1
# mkdir /mnt/img
# mount -t ntfs -o ro,noatime,noexec,loop /mnt/sda1/img/hda1_3.img /mnt/
    img/
```

Listing 8.11: Mounting image of acquired hard disk for analysis.

Autopsy is a HTML front end to the Sleuth kit toolkit. After starting Autopsy, a url is given to access the software through a web browser. After defining a case with this tool, it is ready to perform analysis like string searches and file identification with known hash values. Screen shots from Autopsy and Case 2 is shown in Figure 8.10.

8.3.2.3.3 Document evidence and scene All actions and findings need to be documented. The extensive process of documenting actions and findings for court is not treated.

8.3.2.3.4 Search for digital evidence A more thorough search for digital evidence is performed here. The proposed techniques from Chapter 7 are tested for their efficiency on *Case 2*, i.e. on large data volumes.

Physical crime scene investigation As described earlier, the search of the physical crime scene did not provide any clues of passwords, but provided hints of steganography usage. Hence, an increased alertness for steganography.

Steganalysis Steganlysis software from Chapter 6 is here put to use. StegSpy v2.1[14] has only a graphical interface, allowing the user to select one image at the time. It does not support command line arguments, so a batch file cannot be created and used. Hence, the usability of StegSpy on large data volumes is limited.

Stegdetect[15] is more useful. It can process all images in a directory or a script can be created, recursively searching through the whole disk. Autopsy can extract all images from the disk image (Figure 8.10(c)), yielding a total of 22996 images. These images can be tested using Stegdetect. This is however done in *Case 1* and in the earlier treatment of StegDetect, and is not repeated here.

The other steganalysis software from Chapter 6 are unfortunately not available for testing. Theoretical steganalysis algorithms from the academic community, e.g. [16], are also not available as tools.

[14] Described in Section 6.2
[15] Described in Section 6.3

Detection of steganography software The database containing file signatures of known steganography software from Appendix A is used to help detect such software. Autopsy supports this and the results are shown in Figure 8.10(c)[16], yielding 108 hash database alerts. Steganography software, i.e. known files, has been detected. Figure 8.9 shows a subset of the identified signatures.

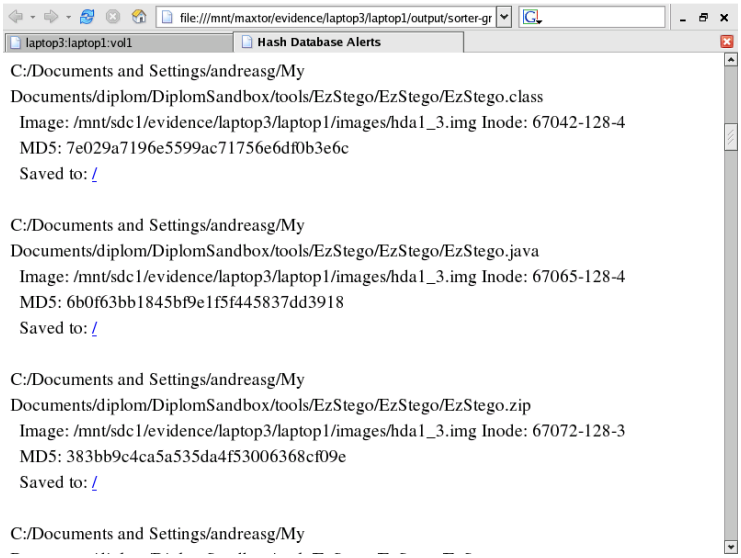


Figure 8.9: Hash alert database results. This figure shows a subset of the identified alert file signatures, which are files indicating EzStego.

Traces of steganography software The search for traces of steganography requires automated tools. These traces are often strings identified in the search for key words. Hence, identification of traces is reduced to the key word search and presented below.

Locating pairs of carrier/stego-files Locating pairs of original carrier-files and the belonging stego-files can be automated, as described in Section 7.1.5. No tools are available to perform this, but the principle can be demonstrated for image files. Autopsy allows for the automatic extraction of image files, also deleted ones. These can then be visually compared to help identify perceptually equal images, but with different hash signatures.

Extraction of images with Autopsy yields a total of 22996 images (Figure 8.10(c)), with thumbnails. This set can be reduced by elimination small images. Viewing these images in a suitable program, will located similar images. This is however not shown here, as it would only display two similar looking images.

This method would possible provide some false-positives, but this depends clearly on the algorithm used. Even if it is not illegal to have such file pairs, it

[16] Only a subset of the database is used in this scenario

can be used to reduce the dataset and as an indicator for further analysis, e.g. with a hex viewer or steganalysis software.

Key word search and activity monitoring Autopsy allows for key word searches. Strings can be extracted and hash signatures created. These signatures can then be compared to the dictionary containing steganograhpy key words.

Each word need to be manually searched in Autopsy, however it is possible to use the Sleuthkit tools directly to performe the searches. The results can even be stored so that they are available from Autopsy afterwords. Figure 8.10(d) shows the results of running a search when the dictionary contains the words: "steg*´´ and "mandelsteg´´ as ascii and "mandelsteg´´ as unicode.

These findings can be different meanings. Each location of the key words must be examined and identified. Is it part of the source code of a steganography software, name of a recently extracted zip archive, Windows registry key etc. Such findings can detect traces of steganography usage.

Suspect's computer knowledge Speculating on the suspects computer knowledge can be a dangerous path, since usage of steganography software requires little training and understanding. In this scenario, it is also hard to be objective on the assumed computer knowledge of the laptop owner.

Unlikely files Looking for unlikely files among 40 GB of data, is like looking for the famous needle in the haystack. However, some tricks are available. E.g. when looking for unlikely images, like mandelbrot fractals, thumbnails can be created with Autopsy for all images. Reducing the set of thumbnails by image file types and minimum file size, and the rest can be browsed manually.

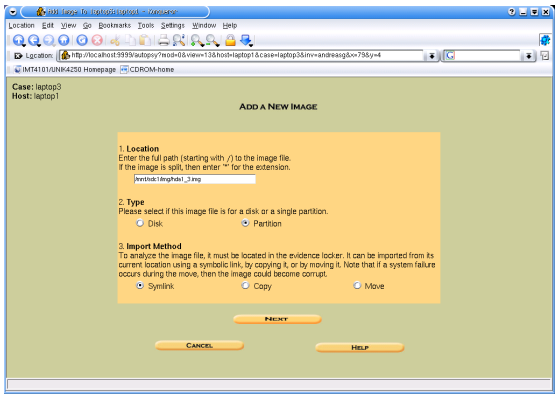
Hidden storage locations When investigation the disk image, no hidden partitions or steganography file systems where found.

8.3.2.3.5 Dig crime scene reconstruction In this scenario, there is no defined crime which need to be reconstructed.

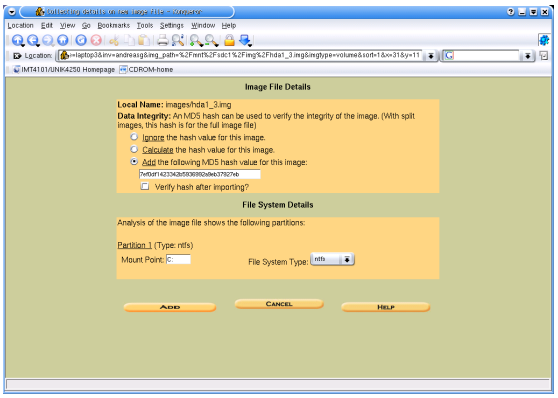
8.3.2.3.6 Presentation of dig. scene theory This presentation depends on the nature of the crime. Without losing the general view, the result from this case clearly states that steganography has been used. Stego-messages has been identified together with steganography software.

8.3.3 Discussion and summary of Case 2

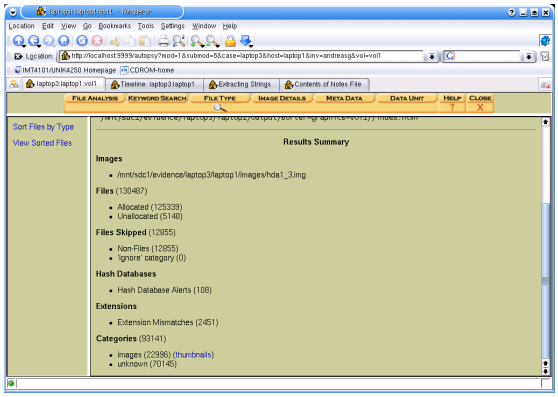
This case deals with large data volumes and techniques from Case 1 has to be adapted, e.g. it is not feasible to go through the hole disk with a hex viewer.



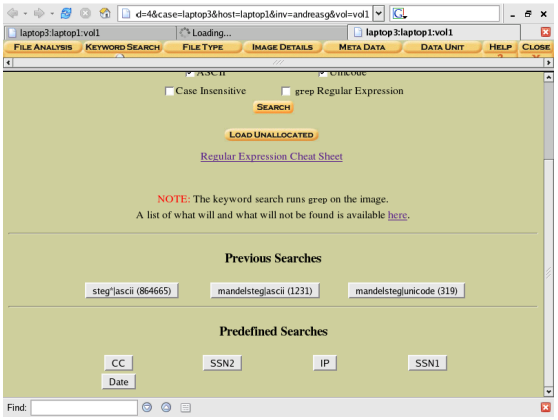
(a) Adding the disk image to the case



(b) Adding information about the disk image



(c) Result for sorting files.



(d) Results of key word searched

Figure 8.10: Using Autopsy for case 2.

But, using the methods from Chapter 7, detection can be automated and the amount of interesting data can be reduced.

There exist some forensic tools aiding the techniques proposed in this master thesis to detect steganography. These has been presented in this scenario. It is apparent that to handle large data volumes, automated tools are needed.

From *Case 2*, it can be stated that the detection techniques works. However, more tools are needed and existing need to be continuously improved. The "locating pairs of carrier-/stegi-files" need to be automated to be useful on large data volumes.

Autopsy and Sleuthkit shows their usefulness when the data volume gets large. Automatic searches for hash signatures, creating of image thumbnails, key word searches in both allocated and unallocated data etc. are necessary tools to cope with ever larger disk capacities.

Chapter 9

Discussion

In this chapter the use and need of steganography is fist considered. A discussion of the currently achievements in steganography and steganalysis follows. Finally thoughts around the proposed methods to defeat steganography are presented.

9.1 The use and need of steganography

Section 3.5 comments on the discussion of whether or not steganography is used by criminals or terrorists. There is no evidence that terrorists are using it, nor is good statistics from investigators publicly available. This master thesis tries not to answer this question, it is however a result from law enforcement and academic community interest in the topic of steganography.

At a conference at Nye Kripos [34], a question was raised, debating that the progress and interest of steganography are from the academic and law enforcement community, providing tools aiding criminal activity. This discussion conforms to the similar debates concerning cryptography and anonymity.

If cryptography and steganography are made illegal, then only criminals will be willing to use these methods. They are already performing illegal activities and one more crime to hide the others, looks like a nice alternative. Business and personal interest can be argued to employ cryptography and steganography to good purpose. This discussion is often very personal and not treated. It is however the case, that law enforcement need to understand and know how to handle both cryptography and steganography. Hence, the work performed by this master thesis is relevant and valuable.

9.2 State-of-the-art steganography

Different tools for steganography are discussed in Chapter 4. What is available today of known steganography software, the author will characterize as usable. This software is not mathematically proven secure. There is an arm's race in

the development of steganography and steganalysis algorithms. The academic community is working towards achieving secret key steganography, which in the end will lead to steganography software following Kerckhoffs' principle.

9.3 State-of-the-art steganalysis

Steganography algorithms often create artifacts in the stego-message, like signatures or altered statistics. There are sophisticated steganography algorithms, like Outguess and F5, developed by the academic community. The same community again attacks these, presenting a cycle, which will continue to evolve, possibly until there is *secure* steganography software.

State-of-the-art steganalysis is limited to attacking known steganography algorithms. Hence, in the case of unknown steganography software, "security by obscurity" might be successful.

Any algorithm, steganography and cryptography, is exposed at the end hosts. In the concept of digital forensics, at least one of the end-points can be a digital crime scene. The alertness for steganography and unknown software can break previous unknown steganography algorithms. Security by obscurity algorithms will eventually be broken. Even *secure* steganography can be defeated with the retrival of security credentials at the crime scene.

Stegspy was presented at the InfoSec 2004, BlackHat 2004 and DefCon 2004 conferences. Not attacking the quality if these conferences, the steganalysis technique used by Stegspy is not very sophisticated. There exist other similar sources [26, 43, web17] performing steganalysis and evaluation steganography software. They mostly attack the more simple steganography algorithms. The various statistical attacks and examinations from academical community (e.g. [48, 20, 35, 13]) are however more advanced. Based on publicly available sources, the academical community matches the current state of steganography software. The status of resourceful organizations not publishing their material is of course unknown and the speculation is left for others.

9.4 Methods for detecting steganography

Chapter 7 proposes some methods to detect steganography. As seen in Case 2 from Chapter 8, there is a need for automatisation tools aiding the investigator. The amount of data is otherwise not manageable. Due to the nature of steganography, there could be a need for such tools even on small data volumes. Recall Case 1 and the HTML comments in the web page, with a practice of automatically running tools, created based on knowledge from Chapter 7, these hidden data would be detected.

This master thesis does not provide these tools, only the outline for such. Some tools already exist, e.g. Stegdetect and other steganalysis software and the hash signature database in Appendix A is ready to be used.

During *Case 2*, it was made clear that the difference between identifying *traces* of steganography software and locating *key words* is not existing. The method to detect traces was performed as a search for certain strings. Originally, this method was added to include more special identification tools, like detecting traces in the Windows registry of in the recently used file list in winzip.

As long as there does not exist such specialized tools, this method is reduced to the search for key words, as described in Section 7.1.6. Therefore, the proposed methods could be reduced by combining *Traces of steganography software* and *Key word search and activity monitoring*, and simply make this different approaches based the level of data abstraction. E.g. using the *strings* tool on all data or a more sophisticated specialized tool. This modification is not implemented in the report.

9.4.1 Advantage of using the proposed methods

The proposed methods provide the investigator with the means to detect steganography. Without these, a likely consequence is failure to notice important evidence. Even the simplest steganography software will then go unnoticed. Both *Case 1* and *Case 2* showed that tools supporting these suggested techniques are desirable. Some exist and have already demonstrated their usefulness.

Some of the techniques are just minor expansions to already exist forensic procedures. For example, adding steganography code words to the dictionary for the key word search. Hence, adaptation of methods requires modest adjustments with low costs.

When discussion costs, an interesting approach is trying to address the cost of not using such methods. If it is publicly known that steganography is not treated by law enforcement, steganography is the perfect tool to help hide criminal activity.

Methods like performing steganalysis on all images and possibly some kind of brute force attack to extract messages are quite time and resource consuming. However, they are automated and can be left running for as long as desired.

9.4.2 Weaknesses with the proposed methods

The methods can provide hints of steganography usage, but they do not provide ultimate answers. Skilled investigators are still needed to put the pieces together and provide the theories.

Most of the the methods can be adapted without great costs. However, some of the tools for steganalysis and the Steganography Application Fingerprint Database from [web3] are quite expensive. Running some kind of brute force attack of some length requires a lot of CPU power, adding hardware costs. How to obtain fundings for tools to detect something which is not proved used by criminals, is not answered by this thesis.

With large data volumes, the methods can be quite time consuming. Strict time limits can then prevent discovery of steganography. But without these methods,

steganography detection probably would not be possible.

9.5 Real world digital crime scenes

It would be interesting to try the proposed methods in a real digital investigation. This would most likely identify improvements and additions. It is however hard to evaluate the quality of these methods, due to the nature of steganography. If these methods do not detect steganography in a digital crime scene, is it then the situation that there is no steganography? Or, do they merely fail to detect it?

The question above is hard to answer, but the proposed methods are proven useful in the two scenarios. Further use and improvement will strengthen the possibility to detect hidden data.

Chapter 10

Conclusion

The focus for this master thesis is steganography and the implications it has on digital forensics. Steganography has long been available to spies and communications during wartime, and has now been adopted by the information technology community.

This master thesis provides automated methods and resources aiding digital forensics detect steganography. The scenarios increased the understanding of steganography and forms the basis for the examination of the proposed techniques. The practical work shows that these methods can help defeat steganography. The methods add little extra costs and can with little effort be adapted to existing automatic forensic procedures.

The investigator will not know if steganography is used, until it is detected. The extend of steganography usage can be discussed, but it is better to be safe than sorry. In contrast to cryptanalysis and other methods attacking cryptography, which can be performed when cryptography first is discovered, steganography need to be looked for probably without any prior clues of its existence.

Claude Shannon's maxim states: "The enemy knows the system". It should be a goal for digital forensic investigators, the enemy of criminals, to know the system, steganography, in order to perform their assignment at the best possible measures. As long as steganography is not secure and/or weak at the end hosts, it can be defeated with the right knowledge and methods.

10.1 Future work

The suggested digital forensic methods from Chapter 7 are tried out on two scenarios in Chapter 8. These cases largely differ in data volume, 3.5" floppy vs. a 40 GB disk drive. It would have been of interest to have a case introducing data captured from network monitoring. For instance, capture a TCP session with *TCPDUMP*[1] or *Ethereal*[2], and test the various methods on this data.

[1] www.tcpdump.org/
[2] www.ethereal.com/

Steganography software working on network protocols would also have been desirable to examine.

The steganography software examined in Chapter 4 is merely a subset of publicly available software. Others would be interesting to examine, like *F5*[3], same generation as *Outguess*, and *StegFS*[4], a steganographic file system for Linux, are good candidates for investigation. To have made available trail versions of the licensed steganalysis software from Chapter 6, *Stego Suite* and *StegAnalyzer*, would made it possible to examine and compare them to the freely available ones.

There exists recent literature presenting a more theoretical approach to steganography. Information theory is used in a similar approach as with cryptography. This is left out from this master thesis due to its lack of maturity, hence it is not yet aiding digital forensic proposes. However, more work on this subject could yield a more solid understanding of steganography and could be targeted with further work on steganography.

A continuance of state-of-the-art steganography would be to develop steganography software according to the definition of *secure steganography*.

[3] wwwrn.inf.tu-dresden.de/w̃estfeld/f5.html
[4] www.mcdonald.org.uk/StegFS/

Bibliography

Web pages

[web1] Microsoft Encarta Online Encyclopedia 2005. Forensic science, encarta.msn.com Visited 20. Feb 2005.

[web2] Brad H. Astrowsky. Steganography: Hidden images, a new challenge in the fight against child porn. *American Prosecutors Research Institute*, 13(2), 2000. ndaa-apri.org/publications/newsletters/update_volume_13_number_2_2000.html Visited 20. Mar 2005.

[web3] Steganography Analysis and Research Center Backbone Security. Steganography application fingerprint database (safdb), 2004. www.sarc-wv.com/products.aspx Visited 10. Apr 2005.

[web4] Steganography Analysis and Research Center Backbone Security. Steganography examination and prevalence survey, 2004. www.sarc-wv.com/ Visited 10. Apr 2005.

[web5] D Brezinski and T Killalea. Rfc 3227 - guidelines for evidence collection and archiving, 2002. www.faqs.org/rfcs/rfc3227.html Visited 25. May 2005.

[web6] Jon Callas. Using and creating cryptographic-quality random numbers. 1996. www.merrymeet.com/jon/usingrandom.html Visited 30. mar 2005.

[web7] Brian Carrier. Scan of the month 26. submitted solution 2., http://www.sleuthkit.org/case/sotm_26/index.html Visited 10. Jun.

[web8] Brian Carrier. The sleuth kit informer, issue 11, www.sleuthkit.org/informer/sleuthkit-informer-11.html Visited 10. Apr 2005.

[web9] Brian Carrier. The sleuth kit & autopsy: Forensics tools for linux and other unixes, www.sleuthkit.org/ Visited 10. Feb 2005.

[web10] Nick DeBaggis and Eloy Paris. Scan of the month 26. submitted solution 1., www.honeynet.org/scans/scan26/sol/nick/ Visited 10. Jun 2005.

[web11] Elonka Dunin. Elonka.com - steganography, 2003. elonka.com/steganography/ Visited 25. May 2005.

[web12] D Eastlake. Rfc3174 us secure hash algorithm 1 (sha1), 2001. rfc.net/rfc3174.html Visited 1. Jun 2005.

[web13] Dan Farmer. The coroner's toolkit (tct), www.fish.com/tct/ Visited 1. Jun 2005.

[web14] NTFS.com fat file system. FAT32 FAT16 FAT12, www.ntfs.com/fat-systems.htm Visited 15. Jun 2005.

[web15] Brett Glass. Hide in plain sight. *PC Magazine*, Oct 2002. www.pcmag.com/article2/0,1759,543491,00.asp Visited 20. Mar 2005.

[web16] Thomas C. Greene. Al-qaeda said to be using stegged porn. *The Register*, 2003. www.theregister.co.uk/2003/05/12/alqaeda_said_to_be_using/ Visited 11. Apr. 2005.

[web17] GUILLERMITO. Analyzing steganography softwares, www.guillermito2.net/ Visited 20. Jun 2005.

[web18] Nicholas Harbour. dcfldd, dcfldd.sourceforge.net/ Visited 10. Feb 2005.

[web19] The honeynet project, www.honeynet.org/ Visited 10. Jun 2005.

[web20] VMware Inc. VMware - virtual infrastructure software, www.vmware.com Visited 1. Jun 2005.

[web21] Neil F. Johnson. Steganography and digital watermarking tool table, 2003. www.jjtc.com/Steganography/toolmatrix.htm Visited 10. Feb 2005.

[web22] Neil F. Johnson. Steganography tools and software, 2005. www.jjtc.com/Security/stegtools.htm Visited 10. Feb 2005.

[web23] Jack Kelly. Terror groups hide behind web encryption. *Newsweek*, Feb 2001. www.usatoday.com/tech/news/2001-02-05-binladen.htm Visited 11. Apr. 2005.

[web24] Gary Kessler. Stego practices and software*ldots* forensics mailing list, provided by securityfocus, 2004. www.securityfocus.com/archive/104/348223 Visited 10. Apr 2005.

[web25] Brenda Langedijk and Hans Van de Looy. Scan of the month 26. submitted solution 2., www.honeynet.org/scans/scan26/sol/brenda/ Visited 10. Jun.

[web26] John Leyden. Website combines spam with encryption. *The Register*, 2000.

www.theregister.co.uk/2000/12/15/website_combines_spam_with_encryption/ Visited 21. Apr. 2005.

[web27] Terrence V. Lillard. Stego forensics techniques, www.tlillardconsulting.com/images/DoD_2003_CyberCrime_Conference_Stego_Forensics.ppt Visited 15. Jun 2005.

[web28] Claus Lund. Scan of the month 26. submitted solution 3., www.honeynet.org/scans/scan26/sol/claus Visited 10. Jun 2005.

[web29] Peter McGrath. Coded communications, did the hijackers hide their messages in harmless-looking images on the internet? *Newsweek*, Sept 2001.
msnbc.msn.com/id/3067670/ Visited 11. Apr. 2005.

[web30] National Institute of Standards and Technology (NIST). Nsrl and recent cryptographic news), 2004.
http://www.nsrl.nist.gov/collision.html Visited 10. JUN 2005.

[web31] National Institute of Standards and Technology (NIST). National software reference library (nsrl), 2005.
www.nsrl.nist.gov/ Visited 10. Apr 2005.

[web32] Inc Network Sorcery. Ipsec, internet protocol security protocol suite, www.networksorcery.com/enp/topic/ipsecsuite.htm Visited 10. Apr 2005.

[web33] NTB. 30 fly innstilt på grunn av cia-feil, 2005.
www.vg.no/pub/vgart.hbs?artid=282199 Visited 28. Jun 2005.

[web34] Richard J. Perry. Snow web-page encryption/decryption, fog.misty.com/perry/ccs/snow/snow/snow.html Visited 15. Apr 2005.

[web35] Bart Preneel. Breaking the grille cipher, www.esat.kuleuven.ac.be/cosic/thesis/2005/mai/breaking_the_grille_cipher.html Visited 20. Apr 2005.

[web36] The Honeynet Project. Scan of the month 26, www.honeynet.org/scans/scan26/ Visited 10. Jun 2005.

[web37] Niels Provos. Steganography detection with stegdetect, www.outguess.org/detection.php Visited 15. Jun 2005.

[web38] Niels Provos. Outguess - universal steganography, www.outguess.org/ Visited 1. Jun 2005.

[web39] Niels Provos. First steganographic image in the wild. 2001. niels.xtdnet.nl/stego/abc.html Visited 19. Mai 2005.

[web40] Michael T Raggo. Spyhunter: Stegspy, www.spy-hunter.com/stegspydownload.htm Visited 15. Jun 2005.

[web41] R Rivest. Rfc1321 the md5 message-digest algorithm, 1992. rfc.net/rfc1321.html Visited 1. Jun 2005.

[web42] RSA. Rsa security - 7.15 what are covert channels?, www.rsasecurity.com/rsalabs/node.asp?id=2351.

[web43] William Salusky. F.I.R.E. Forensic and Incident Response Environment Bootable CD, fire.dmzs.com/ Visited 10. Feb 2005.

[web44] Guidance Software. Encase®forensic, www.encase.com/ Visited 10. Feb 2005.

[web45] NeoByte Solutions. Invisible secrets, www.invisiblesecrets.com/ Visited 1. Jun 2005.

[web46] watermarker.com. Ais watermark pictures protector, www.watermarker.com/ Visited 10. Jun 2005.

[web47] Wikipedia. Alternate data streams, http://en.wikipedia.org/wiki/Alternate_Data_Streams Visited 20. June 2005.

[web48] Wikipedia. Digital watermark, http://en.wikipedia.org/wiki/Digital_watermark Visited 10. Jun 2005.

[web49] Wikipedia. Jargon code, en.wikipedia.org/wiki/Jargon_code Visited 20. Mar 2005.

[web50] Wikipedia. Microdot, en.wikipedia.org/wiki/Microdot Visited 20. Mar 2005.

[web51] Wikipedia. Regular expression, en.wikipedia.org/wiki/Regexp Visited 1. Jun 2005.

[web52] Wikipedia. Steganalysis, en.wikipedia.org/wiki/Steganalysis Visited 20. Mar 2005.

[web53] Wikipedia. Steganography, en.wikipedia.org/wiki/Steganography. Visited 20. Mar 2005.

[web54] Wikipedia. Warchalking, en.wikipedia.org/wiki/Warchalking Visited 20. Mar 2005.

Publications

[1] Ross Anderson, editor. *Information hiding: First International Workshop IH'96*, volume 1174 of *Lecture Notes in Computer Science*. Springer, 1996.

[2] Ross Anderson. Stretching the limits of steganograpghy. *Information hiding: First international workshop [1]*, 1996.

[3] Ross J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications, Special Issue on Copyright & Privacy Protection.*, May 1998.

[4] David Aucsmith, editor. *Information hiding: Second International Workshop IH'98*, volume 1525 of *Lecture Notes in Computer Science*. Springer, 1998.

[5] Becker, Grunwald, et al. Wer suchet, der findet. *Kes − Die Zeitschrift für Informations−Sicherheit*, 1, 2003.
www.dn-systems.de/pdf/kes/ Visited 20. Feb 2005.

[6] B Carrier and E H Spafford. Getting physical with the digital investigation process. *International Journal of Digital Evidence (IJDE)*, 2(2), 2003.
www.ijde.org/docs/03_fall_carrier_Spa.pdf Visited 25. May 2005.

[7] Brian Carrier. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of Digital Evidence (IJDE)*, 1(4), 2003.
www.ijde.org/archives/02_winter_art2.html Visited 25. May 2005.

[8] R Chandramouli, M Kharrazi, and N Memon. Image steganography and steganalysis: Concepts and practice. *Digital Watermarking: First International Workshop [38]*, 2003.

[9] Cyber tools on-line search for evidence (ctose),
www.ctose.org/ Visited 25. May 2005.

[10] K Curran and K Bailey. An evaluation of image-based steganography methods. *International Journal of Digital Evidence*, 2(2), 2003.
www.ijde.org/docs/03_fall_steganography.pdf Visited 1. Jun 2005.

[11] Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Elsevier Science Publishers*, 1985.
jya.com/emr.pdf Visited 30. Mar 2005.

[12] Sophie Engle. Current state of steganography: Uses, limits, & implications. 2003.
wwwcsif.cs.ucdavis.edu/ẽngle/stego.pdf Visited 15. Apr 2005.

[13] H Farid. Detecting steganographic messages in digital images. *Technical Report, TR2001-412, Dartmouth College, Computer Science*, 2001.
www.cs.dartmouth.edu/f̃arid/publications/tr01.html Visited 1. Jun 2005.

[14] Espen Andre Fossen. Principles of internet investigations: Basic reconnaissance, geopositioning and public information sources. *Master thesis, Department of Telematics, NTNU*, 2005.

[15] J. Fridrich, M. Goljan, and R. Du. Steganalysis based on jpeg compatibility. *Special session on Theoretical and Practical Issues in Digital Watermarking and Data Hiding, SPIE Multimedia Systems and Applications IV*, August 2001.
www.ws.binghamton.edu/fridrich/Research/jpgstego01.pdf Visited 19. Jun 2005.

[16] J. Fridrich, M. Goljan, and D. Hogea. Attacking the outguess. *Proc. of the ACM Workshop on Multimedia and Security*, 2002.
http://www.ws.binghamton.edu/fridrich/Research/acm_outguess.pdf Visited 10. Jun 2005.

[17] Jessica Fridrich, M. Goljan, and Soukal. D. Searching for the stego key. *Proceedings of SPIE: Security, Steganography, and Watermarking of Mul-*

timedia Contents VI, 5306, 2004.
www.ijde.org/docs/IJDE-LeiglandKrings.pdf Visited 25. May 2005.

[18] Andreas Furuseth. Digital forensics. *Conference presentation at Nye Kripos [34]*, Mar 2005.

[19] Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. *Crypto 2002/CMU Tech report*, 2002.
www-2.cs.cmu.edu/ biglou/PSS.pdf Visited 1. Jun 2005.

[20] Niel F Johnson, Zoran Duric, and Sushil Jajodia. *Information hiding: Steganography and watermarking - Attacks and countermeasures*. Springer, first edition, 2001.

[21] Niel F. Johnson and Sushil Jajodia. Steganalysis of images created using current steganography software. *Second Information Hiding Workshop [4]*, Apr 15-17 1998.
www.jjtc.com/ihws98/jjgmu.html Visited 30. Mar 2005.

[22] David Kahn. *The codebreakers; The story of secret writing*. The Macmillan company, first edition, 1967.

[23] David Kahn. The history of steganography. *Information hiding: First international workshop [1]*, 1996.

[24] Richard Kemmerer and Giovanni Vigna. Intrusion detection, a brief history and overview. *IEEE Security & Privacy, Supplement to Computer magazine.*, 2002.

[25] Gary C. Kessler. An overview of steganography for the computer forensics examiner. *Forensic Science Communications*, 6(3), Jul 2004.
www.fbi.gov/hq/lab/fsc/backissu/july2004/research/2004_03_research01.htm Visited 15. Mar 2005.

[26] Gregory Kipper. *Investigator's guide to steganography*. Auerbach publications, first edition, 2003.

[27] W G Kruse II and J G Heiser. *Computer Forensics, Incident Response Essentials*. Addison-Wesley, first edition, 2001.

[28] R Leigland and A W Krings. A formalization of digital forensics. *International Journal of Digital Evidence (IJDE)*, 3(2), 2004.
www.ijde.org/docs/IJDE-LeiglandKrings.pdf Visited 25. May 2005.

[29] Jan Libbenga. Dutch internet blackmailer gets 10 years. *The Register*, 2004.
www.theregister.co.uk/2004/03/24/dutch_internet_blackmailer_gets/ Visited 11. Apr 2005.

[30] B McBride, G Peterson, and S Gustafson. A new blind method for detecting novel steganography. *Digital Investigation*, 2(1):45–49, Feb 2005.

[31] Ira S. Moskowitz, editor. *Information hiding: 4th International Workshop IH'01*, volume 2137 of *Lecture Notes in Computer Science*. Springer, 2001.

[32] Pierre Moulin and Joseph A. O'Sullivan. Information-theoretic analysis of information hiding. *IEEE TRANSACTIONS ON INFORMATION*

THEORY, 49(3), Mar 2003.

[33] Gary Palmer. A road map for digital foresic research. *DFRWS Technical Report*, 2001.
http://www.dfrws.org/dfrws-rm-final.pdf Visited 28. Apr 2005.

[34] PDS. Forskningsmessige utfordringer innen dataetterforskning og elektroniske spor, Mar 2005.

[35] F. A. P. Petitcolas and S Katzenbeisser, editors. *Information hiding techniques for steganography and digital watermarking*. Artech house, inc., first edition, 2000.

[36] Fabien A. P. Petitcolas, editor. *Information hiding: 5th International Workshop IH'02*, volume 2578 of *Lecture Notes in Computer Science*. Springer, 2002.

[37] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Information hiding a survey. *Proceedings of the IEEE, special issue on protection of multimedia content*, Jul 1999.

[38] F.A.P. Petitcolas and H.J. Kim, editors. *Digital Watermarking: First International Workshop*, volume 2613 of *Lecture Notes in Computer Science*. Springer, 2003.

[39] Andreas Pfitzmann, editor. *Information hiding: Third International Workshop IH'99*, volume 1768 of *Lecture Notes in Computer Science*. Springer, 1999.

[40] B Pfitzmann. Information hiding terminology. *Information hiding: First international workshop [1]*, 1996.

[41] Niels Provos and Peter Honeyman. Detecting steganographic content on the internet. *Network and Distributed System Security Symposium (NDSS'02)*, 2001.
www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/provos.ps Visited 19. Mai 2005.

[42] Niels Provos and Peter Honeyman. Hide and seek: An introduction to stegangography. *IEEE Security & Privacy Magazine*, May/Jun 2003.

[43] Michael T Raggo. Identifying and cracking steganography programs. *Black Hat 2004 conference*, 2004.
www.blackhat.com/html/bh-media-archives/bh-archives-2004.html Visited 30.Mar 2005.

[44] M Reith, C Carr, and G Gunsch. An examination of digital forensic models. *International Journal of Digital Evidence (IJDE)*, 1(3), 2002.
www.ijde.org/archives/02_fall_art2.html Visited 25. May 2005.

[45] Russ Rogers. The keys to the kingdom. *Black hat Japan*, 2004.
blackhat.com/presentations/bh-asia-04/bh-jp-04-pdfs/bh-jp-04-rogers.pdf Visited 30 Mar 2005.

[46] Thomas J Rude. Steganography, disappearing cryptography. *GMU 2000 - Computer Crime Symposium, George Mason University, Fairfax,*

Virginia, August 14-18 2000.
www.crazytrain.com/rudedude.pps Visited 4. Apr 2005.

[47] B Schneier. Secrets & lieas: Digital security in a networked world. *Weiley Computer 2000*, 2004.

[48] H Sencar, M Ramkumar, and A Akansu. *Data hiding fundamentals and applications*. Elsevier, first edition, 2004.

[49] C . E. Shannon. A mathematical theory of communication. *Bell System Technical Journal* Vol. 27, 1948.
www.cs.ucla.edu/j̃kong/research/security/shannon1948.pdf Visited 10. Apr 2005.

[50] C . E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal* Vol. 28, 1949.
www.cs.ucla.edu/j̃kong/research/security/shannon1949.pdf Visited 10. Apr 2005.

[51] Gustavus J Simmons. The prisoner's problem and the subliminal channel. *Advances in Cryptology: Proceedings of CRYPTO 83*, 1983.
dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C83/51.PDF Visited 30. Mar 2005.

[52] William Stallings. *Network Security essentials*. Pearson Hall, second edition, 2003.

[53] Eric Thompson. Md5 collisions and the impact on computer forensics. *Digital Investigation*, 2(1):36–40, Feb 2005.

[54] Hildegunn Vada. Reconstruction of attacks on ict systems. *Master thesis* at *Department of telematics, NTNU*, 2004.

[55] Huaiqing Wang and Shuozhong Wang. Cyber warfare: steganography vs. steganalysis. *Commun. ACM*, 47(10):76–82, 2004.
portal.acm.org/citation.cfm?doid=1022597 Visited 6. Apr 2005.

[56] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. *Cryptology ePrint Archive, Report 199*, 2004.
eprint.iacr.org/2004/199.pdf Visited 19. Apr 2005.

[57] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. *CRYPTO 2005*, 2005.
www.infosec.sdu.edu.cn/paper/sha1-crypto-auth-new-2-yao.pdf Visited 1. Jun 2005.

[58] A Westfeld. F5-a steganographic algorithm: High capacity despite better steganalysis. *Information hiding: 4th International Workshop IH'01 [31]*, 2001.

[59] Andreas Westfeld and Andreas Pfitzmann. Attacks on steganographic systems. *Information Hiding. Third International Workshop, [39]*, 1999.
os.inf.tu-dresden.de/w̃estfeld/publikationen/ihw99.pdf Visited 10. Apr. 2005.

[60] J Zöllner, A Pfitzmann, A Westfeld, et al. Modeling the security of steganographic systems. *Second International Information hiding Workshop [4]*, Apr 1998.
os.inf.tu-dresden.de/w̃estfeld/publikationen/zoellner.et.al.ihw98.pdf Visited 25. Apr. 2005.

Appendices

Appendix A

Identified Signatures and Strings

This appendix contains hash signatures from steganography software. These signatures can be used to easily identify steganography software, as mentioned in Section 7.1.3 and demonstrated in Section 8.3.

Both MD5 and SHA1 signatures of the files are provided. MD5 signatures are compatible with Autopsy/Sleuthkit. The signatures of steganography software are created with a small Java program[1], which creates output as both text files with an identical syntax as the md5sum, sha1sum and md5deep tools and tables to be used with LaTeX.

A.1 Identified signatures of steganography software

This section contains hash signatures from the steganography tools examined in this master thesis. Both MD5 and SHA1 signatures are provided.

Element	*MD5*	*SHA1*
EzStego		
DynamicFilterInputStream.class	901fcfbbef66fcb72904983d0531423d	19193171ec8a62ad055a372ecad04306c831b2e8
DynamicFilterInputStream.java	938fe9c1b31d6f33df2ecb229cc0bbf7	f563c54a6e9b7609c12345a86b5eab03b185c407
DynamicPropertyInterface.class	11c5829f22436639b996640af1dcbe1e	22007972846006a7b9eb5dd41682b7b457deb9f6
DynamicPropertyInterface.java	b3f9e79b35f4c1077be27a794053ea16	c76f97e37f027f9d61076ac0f6862bee6db12c44
EndsWithFilter.class	d90ddb393fe6b9557b773b447cc420b0	f963f0166acc99474163c8636429bd88c1198bf9
EzStego.class	7e029a7196e5599ac71756e6df0b3e6c	4790776d70049917a91ee263dd8e3057a21e2ea9
EzStego.java	6b0f63bb1845bf9e1f5f445837dd3918	8054a97424a002039537e2619ee91d052cec1abe
EzStego.zip	383bb9c4ca5a535da4f53006368cf09e	b8a97ce3d0bdd512f20e99a5e0189fa5fbdc166d
EzStego._	d4ec432755ac0f4fe7570b7a5dba8bf2	318bdb51747e35dda9187f5082816c251194a48b
EzStegoEncoder.class	ebea3ba597f0f37116d1f93e86df4966	d74c891788d70948f9aa58a558ee8792a73b02bb
EzStegoEncoder.java	60b08b0c22986bc3795a81eb04414419	c7dcfd4feb298e15b1066f8ade2aa076564d7f59
GifEncoder.class	9c6276f351d9c74880164af68ccebe53	e1257e70eb759c3a7a969050cdb27a28b5e509d9
GifEncoder.java	1424fb65f2355c5b914e1fdc97e1476e	1253698d9ee672f67dcfdc239350634d558aa177

Continued on next page

[1] Available on the cd following this master thesis, *md5Sha1.(java|class)*

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
GifEncoderHashitem.class	ff26bb5f6f0f91699ef897aaf764da98	0e93d5dee6b5c7988142920fe61672791ad9c092
ImageEncoder.class	479f3aa312e8ac6f22ede65ee85c402d	a469f8bacf03a314c2d54065bca8a253b01df809
ImageEncoder.java	4d88773c5c81df3cbf8e2dc8f6f905cd	48a2fa55a1dfe53df371df60b7af650f73c7d0a4
ImageScrollbar.class	0431735c4fe128a71e8a069fbe61f82c	98e5ca9fbf811b1f62987cc011d481b25ad2abdc
README	4f30e391a41fa31e6512418458adab8d	89ccf7ddd955bc497c8f23f011494ec59f81e5b2
RGBPaletteSorter.class	3f93ff69a6e63ea462a7fea164499c79	d473fca67d71de521cd5485dce21d7435f90ca27
RGBPaletteSorter.java	0cd15a3b45cf922f9dbc54d8d0a65caf	09cd1ba30010c4358cde76a24df39d71fc2d5bc4
StegoCanvas.class	2159a7c4df2cdc2f5cb02b01e4b8bbc4	9e618d5093d0a7da37a6cd1dbcb67a3b2353527b
StegoFrame.class	716297bb967d098b04e8b13418214e24	ffa71419524690fcdff30d5d6e74f64b0a169094
StegoPanel.class	2d46e4c8794fe563f319bf75ba95aafc	2fd097c2638a27ec79b22941fa4a6154cbaedc7d
ToUpperFilterInputStream.class	f69757f30648d0ba2ea1d47a5fe14450	6b1f5ac74b0f4d95efa744e6f2483689591fea68
ToUpperFilterInputStream.java	52c7186b2ae89bfc6c0dfc1a120fda62	b9abc04898415c5efc3a13ac8aec560b2fa506ea
appendX 0.4		
apX	6e3c86b822c13581c8e2a93987342acc	7adca736d1c0cb11baee97e0b5d911262896cbef
apX.asc	ff8a510f4ff3aa08796aca7efc3614fd	98291d19fc64233295f3bc13149b7d8afe56fbc2
apX.bat	7d94d5db22424ea028a29e65ab5ebaf6	5505365672dcb9400e84ae011fdfc44f4720adf1
CHANGELOG	9ec4ed5fd39616e6cb86caccccb866b6	cc4fb0d010ba9689dd4b955362e49b8c987ecdb1
README	2cc8108494ca9f4a556adb1fec0be816	bb177690fe3cf2eaff5478739aba5e479b631ee8
hiderman		
Aide.exe	029b00c6e4715cde965a537dc397c04c	c6b972189f2c2e4af851e0f74bde7a12ca35a83c
aide.pdf	e94a612305ddff0fdaab76e162616edb	6f67b12446b7b90efbbf73cfd8077ca7bacf17bb
Corffre2.ico	68c9270c5775acc21f83aefc81a439a2	1e1b18c30a0dd3d6d7c5059a85c5a37156ae877f
data.bin	d41d8cd98f00b204e9800998ecf8427e	da39a3ee5e6b4b0d3255bfef95601890afd80709
Hiderman.exe	863b3370f9a454b2ba49a7dcad52364a	c7530f24b4e9c2eb8cd6e0fae3eeb1bd8a7d9362
Hiderman_us.exe	dac0059a2b9a0443b76da730f657c668	caedd035b82acbebe1928dbaef279c5fa355b301
Licence_Agrement_Us.txt	3b3bc010d69536965aabe432bfc1681d	e0edc43defa87e1c75b3624ef5820f48a8bad447
Reset_Hider.exe	a2c680728484ba436d8bf31473527350	b4f6b89e2ee4ae3aca35b5fed71639f642ac5fe5
Temp_Restored.bin	c1dcbfb57e1c7a9eceb98e708d9eb810	ae67c3ab93fae8b780b63a9d3d0f96d3062bb7e5
tipsus.txt	4b55645bb6c69f06844cde7f8f022d33	4ccc17f459073484413eb2093d587dad3701effc
uninstall.exe	8fde804d40de5e735c717470a494ec9a	735046d85f4f11db7273528819df42bd65c8bef4
uninstall.ini	37d4022f7aeeaab1d3f00e3b552ee3dd	277e62fec8a16e7a9923312b7a3dc37b53f7868c
Invisible Secrets 4		
blowfish.dll	ed7d721d0e29b215d02af84db9c67dfa	6fa174507fde66ba57cfffdaf226b688786f9c63
buynow.rtf	c5db5a720f7191cb1785732a6d23832e	6e5838df3bdca88b038c050db30bc8b370a68225
cast128.dll	2c864dc6ce1593bfc0fb8c87439a534f	0c2bcd2a02b0e0cf0fcdcb32ae9c37cdb25362e7
diamond2.dll	d851f6d039a09c6e21786663c16bf808	ce688ae4008972ce5e8d93408b8552952c9941de
directmessage.dat	66b86ab0232f8377c518f27ef9ae4be8	08e5ba8ab2c17ed0eb5cdd45c51f7391ea6190ff
directmessage.xml	3c3f68e7317a1936ccc3c7b6b853267a	6085a12f8c2d17e5dd045175e023ffa1c9dd83a1
english.lng	90be7f563643fcd33b7d313b2704b6e7	c939b4998edd163d7b91c7bf0c9e92b1b20b21de
feedback.htm	5ed448517a9beef57fe624f8bba9f048	4919ed0c4ddabf701f6faabe9f89004e3f69d8e6
gost.dll	12ab3fb71d00b7569944524a8559fb07	b1f45b860c2a64dfe226861e1b9cea7baf9f9c0a
INSTALL.LOG	9c005b8d4c3f15f8401f72744fae421c	6ff0abb6b69781ec79ee16f2066dc95d223f7155
invsecr.chm	c9c84970dcdb04ec3d3ea55933a4cdad	65ce9508628d7490c6675c92b8995b55143edc4b
invsecr.exe	32ecc9e19d03779ca2f24ab61295ba9b	4ced01c56897b04bfea739a011b90d34eb93d059
invsecr.url	a2c421a07f4248e9e49904c2f59f4c84	c58b28aa4ef9e59c8e942ea639af310fb9c0fa80
invtray.exe	9e535d292a3be910428fd03ee45972b4	15d3bff3054a059c7b08c6bb3a6df07422ad106c
iscm.dll	8267ec9c9295c6b864c516fb18c4c03a	ed46b2caacae0ddea4e7b920358259b2d7adca92
keycapt.dll	39927a3dd5a6454e139f1fcb4902fe39	e5a1e8e707464ca264d19128b6669a933a90b130
license.txt	2ac672f3d27f740ee72d6941542b5fbf	d6325174c102c2daa3e0b2f2b175bb6dcca38bc7
neobytesolutions.url	e87c39edb3f1065325f1b8f1c493d5a8	867cefc1b77cf6c1259a8b2af3f4da82d2031f00
purchase.url	040eb27a3f6ce8e12935c3d5a828dff7	9f20f3b2c59d4fe7c65341513f0533c5e7d7459b
rc4.dll	8088090c02e884a9fca9eb7941ee5c9a	d31d32707186aa022b1771ebfc84b63daa0eb5f9
ReadMe.txt	d692abba2742cb36539de9cc057a0242	75dd51e6a69d98996f14b2740dbe89941de0e48b
Rijndael.dll	16ba2a04575c792235ea3c81c044ece8	935478008d4eed8929825dd81014af45a6c507c9
sample.bmp	e7f64fc08c7b179f19bae25edac2d211	27c1dc448425a1b83aa80775ef296e00e496f5c8
sample.jpg	c4a8af3a38f7480893980a69f97b926a	5c717c829c30b503574d93aa0adef10b42e26c80
sample.png	2859626ff8e7c0eae3bbf6a91c736f89	007f33f438876925dd77e5ad6fdf399697f429d8
sample.wav	751f15b52381c6a7017bc8cca74b4e00	570f4eef5da7fcbeee291dfe315a83dbe57e45ef
sapphire2.dll	b88b5dec49947d36952107f7b8b0f725	494a5f688908d680d222b564ff295ef748d2f2ea
stopinv.exe	3f86748e36c77e1c6b6371d064fa1786	d1d3f7c2dda065b27503e502197a0a673ae34c11
twofish.dll	5a23616c88b53757ff3c51bb39b0b9cd	1b1504f7c5a72a4513e027da3a8bf153e6899978
uninstall.url	58bdf30053ee9df66f1c59b336799e00	b2d89a463d0a89373bcb9bc158f48cf5cf8e4159
UNWISE.EXE	973567b98cdfc147df4e60471d9df072	3c4735750c99c63e6861170a8c459a608594211e
ZipDLL.dll	164099bf5795c087aea45f8e3b192fdc	b9fb76fa61cbe05c0068df64f0c1e1d47867564c
_sfd.exe	5b0de04d6b7dc167364ccced6652d7e9	7e363db7477a80f4eb9d7f0e40d7c2547900c8ca

Continued on next page

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
jsteg		
ansi2knr.c	ba89cbf665907d7be83f131cafe5bf80	2f0c8a43bc14aa03fc03dcfa15487fbc10987734
architecture	c22979140ced9db7877d3ccb00ac386b	37556f52cf878f178543ac90183e67daa67fad07
CHANGELOG	f3a66676b74d772e8c712860f24465e7	39065bfa999c593c88ab0dacf4d33ca3c27c1794
cjpeg.1	9ce8dfb6c4bd5faed91b15414e48011d	8bf6e475919e496a84b9ba0c88d60d5be2e9bfd4
ckconfig.c	89f990c3488998b4b8c80c95fc2ef08c	1f06448a0f115cbeb0704bc21a138537c5a4ab29
codingrules	ada6e9f812f7233c1fd6432ddd907698	9d1663ad2b939be671353ea5666937447e99a8f6
djpeg.1	635ce2127a3dc839fbffca73138fe55a	ced98e8377801611a0f1753f25e9be39c3ebae29
example.c	c389e10d0279fceec8c70f0fd8a21ded	dbfb07452b35ec310bfbf1137696fb919c818c3b
jbsmooth.c	aa663c4dff4efa5640234f9ddef164bf	968d51dcf796b393e8b574be8f12f01f8a40587d
jcarith.c	b2de130bdbc4292d1e4ef4681b5aa1d6	89ba89fc6d9a72d821e368b19cb095609b74fb5e
jccolor.c	0221d7ee67d0b878daca693fbc3aa7d9	164e01ce52b596a74e5cc6423e8a30c7c9f712e0
jcdeflts.c	7092edcdcfdd070bb70479b278890053	317dd7f00fc123dd58487e6f269a767f1ece3e4d
jcexpand.c	50ca2f7eafc0a96231de67d51a34644f	e36aef64bacad8d047d4995e078fdce0ec4ad221
jchuff.c	c519be39b037574d6bb1e8147add3654	76427d8f0148d9343deab8ef63d67d2b92de131f
jcmain.c	39ae806f2abf0a92cc62c9d18cf0955a	356c087bc265b3700b0576c58a9939fa14619459
jcmaster.c	5adaade36034d9fcfa95ff3deee39830	ab7f3fbace301907f85425fa8a9e3e1013951838
jcmcu.c	dc8ca8b8151c5c2a2acdcdd4b498c563	8815603fa951f53f1de30a189abda4852c4268e0
jconfig.h	e3505c52d134a1a8aed4b6acb80d444a	1d32732f5eca65937d93e3219205b2f863b71320
jcpipe.c	4f563f9a1cffe86c5c70b8f5c7716ae6	51751cc30f2bfc4fda221d7a941c0bd8aa02adb3
jcsample.c	5fe0aba4b7b05ad9ab4229d588ec548d	937837bbc016d552bcf9a320920b3bfa197ef84a
jdarith.c	086a1e313a186628b388ca3d4210584f	9bd7af8a8eb96063466f6aad140374b0f973166b
jdcolor.c	5dc9b75d7e0e9d0c904038d9a895902b	b8bf89d7e85d5c4d146401061f56b68d5f8052f3
jddeflts.c	1535ddb3c5f6aaf993e54d8bd55f67ae	2684618aa6acb1144d2ba86eff52c644762946af
jdhuff.c	85080e234dd7e47812e6c7fbd9e74cf8	d12d5aea9779f96f4e7e1fcf37df30bec6b0f6b3
jdmain.c	ae3236f14f60f7c8bce6cafc3ecab1ff	140eef8926fdab43de6ab557469ee93a33d46c61
jdmaster.c	b77247cac8665178437a6085682df0c3	ba16acda66cb47c37bdc770d0b4bfa8fb158ca44
jdmcu.c	3f58225833b9a34606a26822e5c3a2a0	8aff3bb8c997214b06a703f914b74ae926a0c9d6
jdpipe.c	615a5da00f2bd90fe1e24de676b3c766	027bc05fd415543574ddab17ce71027335eb2269
jdsample.c	e34aa15535c09f7c5cf3c294a0c944a4	202a72ad1d7a94da9ee8c4ff68b650d34733ee1e
jerror.c	1ddd9477548fcd991e5194ef63c2fc7e	eb651394e1d809ccb8096154404a1d188b3983ce
jfwddct.c	0bd025e5a4b4700a49987438ecc7ed33	b5fa7f4c162a63fad2d77b789aad32ae0dafe7cf
jinclude.h	5bc5a36d416c1d48dcb74da1bb7da55f	c90b24ef55740609138906ad16eb911c219c5dab
jmemansi.c	62e149fce3b1c80328135d3db425d68e	886135c64e89debc685fc67f1717260bbf5728f6
jmemdos.c	46a7aca9e01ae381a5f17293deb7589d	031adcb007de4ba3634e612b56f5542c7774f063
jmemdos.h	65874374377bd16a19bf79a07d10a668	55fa67b632f82ddb9af7eb4455b4242246690d92
jmemdosa.asm	73a2757a18c9243f4b3451631b7e254e	4ff7b9486b8ac0b5bb42327b8f2400d1ab9c8d21
jmemmgr.c	134721377b2c13e61a7791dce8c4bbed	413228b81f36b489f969d241e07c7fe487c5738a
jmemname.c	00f3ada3ce048777ca60666cd5192f7e	bf3728ff4ba79350fdbd9a4d62d68ab52a2b3aea
jmemnobs.c	c88a293ffe3b0269255ddc070f9429d7	a7d458b2fd1e1e87eced49d13588d9b258ffcfd9
jmemsys.h	0071d83f173c813f4ad8949005916095	15e90f06bf25f1cf8ab634b9be011bae55255216
jpeg-jsteg-v4.diff	64663c1d3d2c8e0668d234342a08ccc3	c6a1e599f4af1b3f40668e6f7d3243402cecb001
jpeg-jsteg-v4.diff.gz	13f2aac0436c194c0f9b1993be8824a6	ba703068f8f8d7956a90476a111ffdcffbc06dd0
jpeg-v4.tar.gz	f3d31d69fb47679d40c0c8c14fdf159d	06b8e1bcb6a291351f0d92d10a2b0f9110a1f60e
jpeg.announcement	04446a0236852915aa900a053253dfa9	e1014eb3433ac70c2a09018870e3149ea5914373
jpeg.announcement.gz	8360b7901968a2b9640498dab91da981	81a5f50fda8f0e32c640cb79e6432ce645290ed8
jpegdata.h	18e43aa159e8479a1ab71f364cfdd00c	5da8b0b55ca08f8b15f3729e184a49e486b46c5c
jquant1.c	ad575c6f3d5372db6fb129c5514639a7	e6b0c34a556b95bbfb5d013db6c26b8276557f14
jquant2.c	2d2c5d3d45f12c62dceaac2ebe9c5b53	557c884775b07b82fcdabdda2554a4ffafd27ec7
jrdgif.c	2ff2c6ac5235538b8a39898b11e2eb04	f91c93749add94ec71fa7aff6d5ca56f8aab74ce
jrdjfif.c	ba1e39f173fdcd6106cb71b7eced3e93	6f933b76667d505cef52d3b844764edf7acd5403
jrdppm.c	fcc35cdece246249d090195e5203ad44	f36d5d2556eb9931a5c33ff9ecad53e587fc625c
jrdrle.c	8888f3448543f914c913950e743e82b4	58a82e305693fd53b09307ee1f4829b5b2e84880
jrdtarga.c	362b0b73fb4310d127449f0271b7a976	6088deec80d8032092d04ba8be680647c27cdb3d
jrevdct.c	aaf4cbd6b39155ee19e56cf7938ed32a	6fc9c78d6800624daf1db91a313d6d7e7b5ed943
JStegS1.CAB	465773adee38a29c15552c867285b379	156a619c2b4ef6bfee0d5e4036c3399b1ebc9a6f
JStegS2.CAB	a96d9775bb8259645a211a364d5683b3	bd4b4cfee41672dc0164cdccebb18a943aafd4d6
jstegshella.zip	f5abfbc71ebe3e186649635d1e7eab31	32106b76ac5c977970fe5586ef4198b2460a6ac5
jutils.c	81dd4a590cf6c8c62668fa9cf689cb38	c7476e86d771a6edbc12e5f5bf0e26ce412d8cf0
jversion.h	78ee9b5805fa925b3738c9dc22963c1e	bb4ecc7c67a224ea4ab4d7eacb4ee6ab30620bac
jwrgif.c	a6d0566b9ee03381d08683eb91b20102	29207293e77a974689160a6cebd2a0df9699a607
jwrjfif.c	1f616f631f2657afbc80b420b94b8603	f9ff3ca5f42c01498aae3f28089452f1a8f37c2d
jwrppm.c	09955c6b206b45d27e547e627f28938c	11e3ee2b38cb4e9a3a12ad08fcad0cb81bb16208
jwrrle.c	fd335a1e2308a5f817617721b4c0eb8e	b89952f24fed3128da8f601d2d90a36f1fbbc4e4
jwrtarga.c	a2c5873afe462c2edcfbd834b56a59a4	56cfc3bd5180bc9559afcbebd6d51c14d9b9803d
makcjpeg.st	07a3151da68003501518589ad8ad10e1	072dbac16936a4bdc0f7ae3d5070b771150fb006
makdjpeg.st	f17aee1cbe8686a75241f8a05106dd17	902e57b18964f3ed4156ed43d4977db278a06857

Continued on next page

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
makefile.ansi	4597e410e139012e45fe07e19e1574fd	36c37703e172ec89d6a7792ba8ef85b855e0b983
makefile.bcc	947b41bb733eae63dab1d59734055075	baf0ab17450ef6d3ffb89f8c82a07572a5939857
makefile.manx	8d09765a61ed7b9af35c054d44bc72e1	8fda87a018f7acccc0c1ad983c27dc6a0c859844
makefile.mc5	4baca2d453a42fff2edd75de49d20662	ccd5a762e9e7b20a807f87768a8761049b2bbf0c
makefile.mc6	f9d0072061e78e5a464b2996cd237b2c	ea34bbc8cf05898708684b210c1e1b9b20ce9528
makefile.mms	f9f6499620c0a7b4a4317c786ec93cb3	a37bcc9571215f688b9ac65a559ad9ef401ad2ba
makefile.sas	06f939d6aea766e11a4845b56b9ee81f	72a81050ce199f8a45bbe35c8c5460ce9b791541
makefile.unix	90b6a5c0436cc867ff1a559d1d717c26	45cde9e340b648b20c945dc595b88baa4c404bea
makefile.vms	7ae6744dc8991f37a42ccff23c503c6d	9f1a790090c3843db7ae93721555b79fbbb5dd5f
makljpeg.st	b53b5523b66bd63b9e104214b265514c	6e13fe12d0c7891aa3cb58443639b19dbe5fbe1a
makvms.opt	8024dad70d8b9edd012ac1189c1a1221	bb5de8a6c1049a949dc6d05f5ddcb94adb99f993
README	9e670b34d75436b814e8fb381505cc7a	4f64a917cf56a4a3fccb6c52255e5c60cbdc2404
SETUP	7c91f9a7b7d0a168cc5f05da2318eb43	2d3b79bd7241ad22293385dfc799fed5371adc63
setup.exe	b851276c65be17e04d363c7c606aa3b1	f6c857a99fd081a90dab6f2cb3cb8e1cef02e2b5
SETUP.LST	5e64d352666a7fa86bf5e4d8a2d33f7d	09bd5b2176301ba2cc5dbca1ea32369c077cfc86
testimg.gif	133dfd5b1d60753079f0d510b2861dfc	7d455e56e33324d0442b1b664e2a04069cdb0086
testimg.jpg	0a7366639f58a8a6b08483e41d92077e	4040f8aef4aad45b4df3b67d821bbc0279d10a53
testimg.ppm	80ca9932ec3ef7f25e8a6f08abd43008	218e007c9215da5da4e2dc9a0eb9aee65ffaadd2
testorig.jpg	e0ce598fa47e69c6224058cf3bf5b60f	a1dd3cf8b3d2e27e1c4780738bbe007415f9d7ff
USAGE	a79237c3a15f594b78fe9baadd94874c	3a47f6d1500572838fc1904b4f9ea46e8011f8b6
mandelsteg		
comp.bat	a18f817f7a822f6db9f1e2664d96f7a3	a6945a7b499e751d406a2a4481a3a75c3f611627
ext.h	3e109d6e9f4cc25a36f3dc159fdda28e	8cc82fab1bf74f226c4d70206a5a9e5cc2ff4b7d
gif.h	bbf869aa4e73ac94f57bc7c3a793a3cb	4ab4965dcf040b41c275a8a603f86e311c7cfb72
gifextr.c	0504f65eca5a193439a29abd3409f190	046ed275e3c91036b241c76aa2a09af398cf23fa
gif_comp.c	413ac84cc312a54e65e016b689f535e6	5787e5859b49b4e7aacfa75affcfc18be943de4f
makefile	403ecb43f3ef1ad56612f2137bd3aac3	7e619180e7408aa0ccb2d7e205a90f443f1e679d
mandsteg-10.tar.gz	6bb5e47f4c28e08d3d4702425242449a	fa19bb14d9dca4505f017053d537c6f2488a874f
mandsteg.c	3dbd42dfbd8f8185060ff57c0565da06	ae41ac2040cf46a866537640782a8e9b5a4bae5a
README	c088917cf179cebb99e06393ace5c157	e452ea9144a86385256f044bc84e8b922bf29e3a
outguess		
arc.c	4b288fb13c13c60bbfcd02c43d66e357	68626d783341765dcea1b4dfec894d037a172779
arc.h	5fda66ac7b458493201fa55d3df3c9bc	21536db2ddb2d86478449cfc40cd4388397852c2
ChangeLog	522da1211bc57f5cd1c9ddb1cde023ba	3d035aedf40cf8353b7d73c8edb85ea4ad958711
config.h.bot	1304918b9bec624eea2909dd4cfb1e0b	75e1ce492968fe412466e9c28d0003735117c900
config.h.in	52b10277f3fb4ae3ebad1959eb848140	755b7a7b7694fd77e4b4e3ff6bf78f02f021b932
configure	fc6939faccd58f721016b91b501845b5	533173c0e4e76df3bbac21f3ce989a721793f25d
configure.in	5b946a00c4ea6f42e7c3dbe149f0a956	79950915b2fa48dbd103a9b2fe4b9c4db353408f
fourier.c	d2e9e8c08786d8e0e78f08b3d2e30ce5	0e8e0498c648b05c60c22b93a705ceb5fdfe843f
fourier.h	8a49091488b1aa7fb1dbc8527940a4b5	9fe2a0e54ef3aac11d6a1c1a018bce3223825e6b
golay.c	2a7f07304d53ec3b06fe1cbd192fb7ad	a779c2ae50f749691abcaeb3d4ed51016548bbe4
golay.h	5ef38206ea2c005257bf809727097c5a	b71888500d4b0a35739ad4253334104440f8f4a7
histogram.c	4eb26143e6de993b38a0a9209a47394b	5b1ff3ac33bd181ead2496c3d35bd70f25602576
install-sh	9b9a3382dc6798b6cc9db374e5ca6c9e	ada1c3622f7fda4987bccd8b7c64f8fdfe919818
iterator.c	f9a72c8e27e501173bee391252b0591e	f9b366826b28226f1fb9997bd1391a5bca34a46c
iterator.h	286813dec9fe0a7c2b47004522c186ee	aae1c874e8012c3e2e7a08969d03ab779127d1df
jpeg-6b-steg.diff	9415744ee11168628ba3b7cf310cb6db	938dfbb96a15be37090515512a77fc8823790902
jpg.c	9b5ac68cbc3ca61d0dce2a7b4ff4988a	ce0835d22f8d64ae600550f04c79805339938965
jpg.h	97dd9c725585925c69d2918d2b932dd4	5571fba232db8998d8cdd62e63abd36f60dcfe1e
Makefile.in	2b7d640450984a13bc7ae3df6172a05c	85e80895c8c3f5440224c0302342b48b494d4d4d
outguess-0.2.tar.gz	321f23dc0badaba4350fa66b59829064	d8d7ff3d8f492c3fbb075ecd2c6e87ce7cf13b80
outguess.1	dcb82d262a14bc021d62892c25d6c068	e07b8da94664b9c74ce3017d117ed444af76f4a8
outguess.c	a2fc74251e4a679630023499aa6678df	9672149e9a0987808aa560895edef19324ae7023
outguess.h	bb84b6213acd7dc6fda08e2f79315e44	b748922e29e79d0d24734898c47147e12663be54
pnm.c	a8a091e4c7411dae79f7a428b81f3e84	0abe6f2579a21f8c7ef2b0356b7127bb89a0623a
pnm.h	2975ad88222258732c71cb54a0c61a35	9d8e7afa2d659f8b3e44b8a2d72ec98f68451738
README	3d733bc818bcbbf3ff89777fc03966e3	9a3ff8c927a3f35b17e1fd1787cf31f2f4c0c2a8
seek_script	a64675ab5c1005ed9279cbdf9719dd5a	a192337781b3803b0206d9b3c4d1cff4a208a180
STIRMARK-README	60bf80961c406870524a9175a355916c	94c726bf2fe4936f8022eef0c43f0c460be8a0dd
TODO	d41d8cd98f00b204e9800998ecf8427e	da39a3ee5e6b4b0d3255bfef95601890afd80709
ansi2knr.1	5e1899cee05f8b6f22a67923e8137999	dd505f18897d8f14613aa79936def0b0f3cbb839
ansi2knr.c	7b35c47837cc799eeb3653ee85f25ce1	4c7ed24f77d6dbe05b734cd73489c4af8c066adc
cderror.h	be7d9da751b8a0e2663af1bd6a91fe31	0f86ace2f8d088d04d47a1f0dbcfa5f19214464b
cdjpeg.c	4ffdd4eeafaad7234a527ba382381380	56549d8cc15f4a99f3b213d9b5e7330d95e722ae
cdjpeg.h	41b7e56406e2148542d9303216736585	d9941a32cbfb60d8d1d95b4b62158f46677d66c8
change.log	4f4c5be244fb2c70b7940914938c0dda	046ad3ce34021109011301fcd1ba0f5466d83bae4

Continued on next page

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
cjpeg.1	5690e9c297900db4cd50bc5ad8905fdd	1e2d759cd9353dffaa23c9154529f9d72c28dd4b
cjpeg.c	4dbdb32c41596544f500abc8636bdfd2	5e847dd0aa3d3694f98d25a058cd7dfadb6f5bcb
ckconfig.c	b440bc8a7f36d75cdbfc7d67a0544f72	1dc0387d584bf9c39dd28782d1304823579336 23
config.guess	755c3f68567ac6dc168086d8fc10bcbf	357867cfa4252275b17752363da1992c7be1ed88
config.sub	771e3460a0731ce5ff987925e40104af	15e65bec3b3dba99bfdd0e54c1d48a1026d2d495
configure	be58fa2f0073922c98d6760ee364ceb1	c5cb75c5eb0c976ac81f481d89baf18bf06f34c2
djpeg.1	52a74eef729c28448e56b731f45a3dcf	84f98ce0fe3c29cae85d885831d9cb78ffa23d28
djpeg.c	2eaf705a7f37e9c88eed36d666680735	226870b8db01ddd620d50b6a2909bb6dd80773bc
example.c	168af978e831a5fb7bc9ba5ebb20dc51	cf9603ede98c40f12ab187a3b78d19038c8ad50d
install-sh	9a98c7b799ad906d61a88d52b5790b1c	f70782d58c610b0f5ac3343a45282c429df45b0a
jcapimin.c	c2b39cff18077c2601c7f013bc461764	976ec8c49a81616a85466b04184f7ea8f4a8cbd2
jcapistd.c	98951837a74c22f4fe7bb1ff6c320116	2da43bbec11a814cde795fdb75a9cba6b6c31b7b
jccoefct.c	5c4997ec8ba5559dd1b4c87faab144d7	08d046aefc1227eeb5b575c55a4ea6ec3b578b09
jccolor.c	f92444f9ec363c2f9bc847b46a0f020b	6050333754a73ac6def0ff8860d80e4f93c01d95
jcdctmgr.c	bbf7883cacf29210d4adff73bf30ec6c	2b765927df06fc2f693a488083b83c82ce818515
jchuff.c	a76b65afe34ce454eadc30493df04932	bdf601f8e79245c899d821ca0a63b7540c9220fa
jchuff.h	3a359f3bf9cbe701951eb1ae7cd55075	721250f329f4ddaf806a77341d87b08a5cdb61ee
jcinit.c	539fb9ca6829be79f545c74bfb9526d7	42cdfe7f4732552edd909528481aaa8526479a09
jcmainct.c	642ae670269fac0832313b24be8f710e	e109512824f1401496caf0f78574d0ca5a0454ad
jcmarker.c	af0a5eabc655397fe50c5e55c210db2a	0c16a8d868961f41f74d84c8da1c36e43d45a71c
jcmaster.c	0ff8082a5856a0bd97e8efe0c03b974d	ead966537b7c089cab1448dd83ebfc22fdecbd81
jcomapi.c	ae25b8c654347d91ba5c393adde56355	838e835844115395472263153 2f1b39960bdf678
jconfig.bcc	ea27a698dbee176921f0a1e20cb0f7f6	766c615826ab30208df6121d17ff3adf52c37dba
jconfig.cfg	e6d25f793d2327403294c8c4b1a2be08	e96bfe9f16a330bb974d5ac2645cc6d56803b574
jconfig.dj	047cb327527d5523b4fee6ac12624b4b	f31c7a2c7517707453fe08c98934b5e7450161fe
jconfig.doc	d41d8cd98f00b204e9800998ecf8427e	da39a3ee5e6b4b0d3255bfef95601890afd80709
jconfig.mac	031b66003e61c0033ccd87c75434075e	7127759efcd3a6c8f9efd95dd4dea8968ee4c92e
jconfig.manx	4e1d7be4223d5b39cb040748fbc00b97	c06eb256405a3fee4d362f1615ee9db86e82e0c2
jconfig.mc6	31535c252c6f98fcb8501234f3128a10	8573024b08dd592fe4124b9daf3be832f670dd2c
jconfig.sas	e454db42fae301b21dc7ca7a75b17028	db050809e08c3f2022a0b1e16560ec8b09a3f7f8
jconfig.st	814cf32becbedc530766ea363e6747f8	963a32ecbb513179645884fbccafa96a68e5ee53
jconfig.vc	b666c488a717a339d636dc63bd300b73	93a39e212ccad6ddcf782af678d69ae57f6c34ae
jconfig.vms	918fb9d9d7450686f4df077cda420ef5	8335b0291345c46eeadbaa2e90b6fd88c86793bd
jconfig.wat	13ca4d6e947fb78fcc898247d3cce493	3da15b477c5f23dcf2a1df3dc921517f0a6ac015
jcparam.c	c1c24937d948fa4df69bcd29a0ecc33d	73460f27b28b6932a24ca76bff3b7df8d30484de
jcphuff.c	187e27b1f37fa51023d764c73b03c7a4	228c64e8696740ce20d49f66f7aefe310b1b0eda
jcprepct.c	a7e4032663d3d01ac2b7d5aae115c58c	7f52b0f67f9eebfa3b57347bb09fef16da452d5f
jcsample.c	ec3c4dcb82eec478cc53a84bd7d4f9e7	cc8d11224bd0f32b3b184b5fb26f03882a476452
jctrans.c	b7e298b8b8e10ca38d726130d5f119ff	595c4816ad931ca5a38a8d5ac2089ae967125dd3
jdapimin.c	763f5acd55337b21e55877316b5ba0c3	63eacafa0de15aa22f7ebd2ad074a25fadbb09e3
jdapistd.c	b8c2a9211a40b4e34a6bede0d768e729	cc762411f718e7d7607fb9ea1b844498c2967540
jdatadst.c	22e6fd5addf27184cbc19f4c435d22ec	f298be6137c08dd145c3556be8e84805094b64a9
jdatasrc.c	ea35cc0b876a765c3f09a24decdd21b8	33c637eec290444aa73ae90395070c864de033b7
jdcoefct.c	8f2e23341d49de9906aff854f97e2370	92bfd19fad0866a4aca814fbc96edca21cfb1c89
jdcolor.c	393943ca6017e3115bbe6f4aa2b56373	3fc78ed8563445ec8e83ac85e116969c0afdf856
jdct.h	18978d93cfb1aabf5618c3caa5dbcf5a	3c3716c51b41da36c50a13826e7d8d7a1f430b9c
jddctmgr.c	1caa1e0a42975992180b315855419dd6	4050307344f7bda78f89a1006c9ab4a3deadab62
jdhuff.c	6492c45efdb95b0edde47bd2c05de989	0ef3f97220b8c1e64abdc9019268d61153f7ba69
jdhuff.h	54c014bd2f681b2c0f221ef9f55a02e7	4518da41d2f0a845b34ced1632fc8636165a3d20
jdinput.c	3e4fc9a613c6a1586f2b0ed6b872a668	e851fdf7dd1616837aaeb925efae6905a3b164dc
jdmainct.c	b6a438d178c694400971a7463e5632de	c2f36ee10d793e1d7600931f34f9d8dbe5f735c6
jdmarker.c	0687ea922a828d5eda9060248fbb4513	90ee00358c3183bd0d71c38e3092a600e9b654c7
jdmaster.c	e66d4070f646b204ab4ae799b23c63cc	028ac3afa8a127329e382d1a17064053ff1aed10
jdmerge.c	955b27a18c0d3da682da3d4c43dc8ffc	031f37b551c132a5edc2594a07bdfc889f8a69c4
jdphuff.c	708491bf4ea3e387dd60a0443e6f8a4b	355e10fae0b238db49c6b180b287207f3b8aba94
jdpostct.c	73bad6582af500b1b146ac08e0d23844	47457796a19b3eb086d53b43374e1f43650f26ed
jdsample.c	4363ec24ab3d5df946f550eac583708a	77cb4e66286b6459e2a0c0744ea5aba52cef4d2d
jdtrans.c	1bd79c3a33c177ff07f41839a6d8e671	2b7d18d6ed2ec2a84b1fafac3226e7132728917b
jerror.c	af10a8c210ca453462f0e50441d9bd87	fca5c27f8f9f8dc911c22339192ddd76b3d85d2a
jerror.h	fd9e21c2cbb50822bc5cc94c50861467	a45a88f30583670c57b22225ee74aadd72ed942d
jfdctflt.c	6c0522409d7e8d81a2b2f0892c34d8ad	0a341b2fe95f937a0263e9b60f0f608005aab15f
jfdctfst.c	f806a6c964daef45c71ca78af55bc7db	8df22470a0a855fa1ce35d79d86e2f06aaeb934f
jfdctint.c	9cd92764a3a5e9640abbd29fc0a09c40	efc47fd6d78308e84c71d78edcea5a80b305f710
jidctflt.c	2db3e30a55bf8609a7ded7ea3c2ff99d	4686640dd8c013dd697609c2637dc5e3753983fb
jidctfst.c	52ffd9e905e785b8cf55a5966fb5d484	7f45205f64b70d9e05bf4f76b65a2dcee06440bd
jidctint.c	108ce4f1bbbca7d3af1a00166e2260ea	2e23a2ba721a9e7a11174d7939334edf99450e36
jidctred.c	a6ecd4e7da02cbd6972b713d8b925615	76e4143d8df8ff6bdf51ea833d0d2015f805def8

Continued on next page

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
jinclude.h	dbde79bc104a2caa9316cc2a9df7fd25	31ab682733b096b3a98c0a35f9b54a7936e480d5
jmemansi.c	5bb449eba31015a4dd7536c8b3d7b68d	b51ed0c5816ad800a5f8b6590ec7f59604f2de7c
jmemdos.c	f36b756e2ae25dff6c3b5a3ea0347b59	1df4d173f791221c0cbe023c8ea588a65cb21174
jmemdosa.asm	73a2757a18c9243f4b3451631b7e254e	4ff7b9486b8ac0b5bb42327b8f2400d1ab9c8d21
jmemmac.c	7977c0b00763a1b7373bdffcd03f229a	1f8aac84d843e74df128c6c0995f58072a7b2726
jmemmgr.c	04b36ee19b3049c72046e662aab68208	6f07be8c031773c510451ae06e997a6546e6c811
jmemname.c	719f6097de742d244c6680941795fa3e	92bffd9f31944ac63c39d6ef3f08aca1a544b4b8
jmemnobs.c	0fa7947ebfd9f703df89b0aabd7f58a2	be0a1e39cd9c9a715070e2e677e57178c81bfb48
jmemsys.h	23e479350ba09f9be9c09d5812165194	662d26fb621941839b91d39e5bdb05a453612212
jmorecfg.h	662009a2ec0646da33ce6f98bca91ebb	1a7c3577e7a5cb2eadae63988fdd14ac65ae7f3b
jpegint.h	cedf741244f17031bc40505160893af2	3427c4e1453ad34a9340022b051b22453407e97e
jpeglib.h	99e9c87c55c6f147d062be906fb0366c	6fc0a3cc8697cef0556e1c7465bf96ba1233fdde
jpegtran.1	f4a3b3f7def11be90cdf5c0017c556e3	333a44376cec9739e502a5939dc1be6d5fa4a73b
jpegtran.c	9fe8cb2b4235362ffadb19f6feff2a37	9fe0ca1ed9b8a1c0b474627781ede03a28dd60ed
jquant1.c	00c6590973efe2514d5d67526ac14fe1	fe6e415e102dffb92857bcdd0aa6d232a88be9c9
jquant2.c	2cb32492e6ec440c1c5a9dc6d7209173	abd65ed7c4efc6f121997e42d396c218c1fcb9a0
jutils.c	74fca8c86e91d12a9f688df153688633	052847eb62d4e89e3c6c8d7fadf0d8012f692bb4
jversion.h	b84525e18300494f16d52e6854aaaae2	55277083aff061fb5f9fe9a4336facf3a8b9aa68
ltconfig	ae90254d1eb8cacb496eabee7c26be48	aa6a94dfcdef98938ddd96f0d530c253b5925da3
ltmain.sh	00ee2a35bff2d110c964941a8b7911dc	354c900ca1078d31cf979dad860f2a8c22347a1e
makcjpeg.st	004fafcd450c9bfc742d30c7d427125b	f7fb98e1cac03dae91e8f7b582708b26179a31c1
makdjpeg.st	499a4cbc5628d5ee781dbc97eab74546	4205648ae6de953479b8a4fb96e1a9d4247f1156
makeapps.ds	9909b938be8895554a5555a533e5974b	d9c12d4492843601ebd29e751b852247388341ce
makefile.ansi	9d3f3d3465d373361e04ef3f13b78b59	3f985f009a5796be53ec11a9e305ff22607c09d0
makefile.bcc	9ce94b3862e13f40affa3a836bbb5bd6	423a31181a5b0283229d375281b7bd57e1318233
makefile.cfg	52b6b731571b2515a9b760530ef744d5	46c2546c5cfe7b1c79d74fbd6fa2f410e10fe281
makefile.dj	2f34b2df24219d0f99b1b4b97f1b3108	29e5c0a8aafa51fe526ea42084d377a21d015df0
makefile.manx	3d37997768bae69a46462f1f9954c51c	53fb8dfafa2424391b42f4a03ce7118c93f8d8cd
makefile.mc6	79e7b20768f1099690758377bdffa51f	7a4060c834e7579e0675e80fa917250eecd63b03
makefile.mms	c307a67f5bdf24e3551216e50a3c32d9	b23f22501e9408fd2c65aa91f4ee64d7621bcd70
makefile.sas	9ae0ea476a7500a270204b6afc52783c	bb533baaec00ffb39ddcdeeab53adfc3cbe34ad2
makefile.unix	4c3460aabefaaeff1b6fb59195022e08	f2273984dd4f0872407b69cfed6c2bc24e0c7a51
makefile.vc	d4cf40f892f4dc87f5eab111d12b8709	60d31300ba6d9ae09a6e7fbc87d570fa8043621c
makefile.vms	7ac4ecebcd2e91fa5dee514e90b84ed8	5653f6308b6f0828731a9fcde5e176316ccb4495
makefile.wat	d1eaf25fe6c5d49ee2648c5f7ce3ea8a	23163895fe1ea37db45be725d0adf44d452aca69
makelib.ds	60e89ab465a0e4b1780d579efac8e950	ac661399da0ebe606d8a5b7f9be465cd449b0b17
makeproj.mac	01c955d4b70dc9a686a046a42307dae2	2e624888015cd2696f589d74875a4fac2d879f22
makljpeg.st	3ef0ad9f9964ed3fa4ab6e62885ac36d	f52ece430c19b65f8a72f45569e624939d2ea18d
maktjpeg.st	f4ebbb314ffd63fcf7829166ae1f802d	6fb3b7a5639d785cd2f7f03b94f55017400964ec
makvms.opt	3ee02f68cb9a8ea9ab5bd6bc725de55e	a3e9bffd9a7c8b015a1236b7dd44204b2563a86a
rdbmp.c	c645c2f17288fccb2d81ddc00a5e56e8	a9adc20889778766a75e21ee78b8a402f3d39908
rdcolmap.c	d33306e5e40d81d4a18b97453a23de26	090c5f2dd503595969c98629beeaeb0a50d0f56d
rdgif.c	6be05e2a767c2d18328d766d0b4d450f	ebd0cc75eec11e39bdd1c815f0aeefa890198b79
rdjpgcom.1	714dee8cc30bab77ba21ad5d11c8bb8e	81938bbbb9cd8832ab8132d284d27a9e26eae151
rdjpgcom.c	341a84f5b038a63c1a917292ecd36983	0c0db69b639023b6d9151f6147c837b320bfdcbc
rdppm.c	76e70bebf457b0eb593bcb3ad30768ea	8a567288c15ed9d8eb50d99023e22d063dcb4f22
rdrle.c	4bcc8d13f2208b26d67f49a0f9ac0f60	816ee7c7840ea335224f1b8e47d240d93605e084
rdswitch.c	ab14f8994845aedcdf1a60d3ee63185f	2523378f4543a6215fd33abf15d48ecc40a28392
rdtarga.c	b522b32beda6a075f8d1bccfb66aa094	486a47d75f2057f6bd21fb00b6d87f213513c572
README	b2c4ba2a8ce80468db498a6b331f900d	ecdf7321b8fac879475f2a123c495468dd346ba8
transupp.c	9c88a00708fd1f01590de75ae9d6815d	0f89005cc19658ef3c41a6e36a77787c5e2330c3
transupp.h	af905ab0a4286224bfe1edff540924f0	69e1f02356fee118f5b1c2c51a4db14027714bac
wrbmp.c	7a9b95b40309d5212bb8f10090bb4b4e	0538c2427c6a5ff9f769efa67a2b4fd1de263d2b
wrgif.c	c3358daa681a28d173c011180fa3323a	f11260c01580b9b5327e779b92b74da27d1171c9
wrjpgcom.1	8ef6ec3d5c84fd25fd4a34ca8eff233d	230efe9f04d9f082eb5cffe302f58047399c835e
wrjpgcom.c	4f69aaa5a4a643bc852fb2247bd0c797	902da01b722840fff801c9addbc2ff322ecb722c
wrppm.c	a877118fd96c41d4e05e572fb84361b4	6bcd69d7998547a9d54c3851351068db1ced0c7d
wrrle.c	2d737ee0dffd7750bd06c6cdad5e5386	8c7e109fcfb1a57b8dd217845fa6c87db42d9c5d
wrtarga.c	aca4ae26407d3a990bcd9950cf9085b2	a524cb8b32cc0639d0c5a7ef5efa277cd375e24e
snow		
BitFilter.class	0660b1b23d795dfdc7f4c9db5e35e679	19a34ecd8e5738aff690a5e617d8f73d46c8817d
BitFilter.java	bd290be3f3049405f3633c05ac5c9d31	abd9fb7c46ea89f295db6038b166bbb051d3fa14
compress.c	be2ac54cbf8d01e28e12db8257147ee4	674b6ea45d4f555344df4b1dbdf58033cf362751
encode.c	1ba02ad662801c489dc41abef4b01096	fa142c9ef63a28b9d379e95dc08e75260bc585e6
encrypt.c	bf141f75995526065eb871b4ab38525e	d066443149447d7baff7f1cae81ceb772b15e16d
huffcode.h	eb707e027f0f7580f6944b86757c4b8d	74293a0a219813da7133b5a2ec89dc922d95da62
ice.c	8dfaf89f625ffbf2469fead2bb6c6343	a0cefa7c7dcb01cba32dd86e963c5b588b799ab4

Continued on next page

Table A.1 – continued from previous page

Element	*MD5*	*SHA1*
ice.h	1cd5158b18a89bd7f26943edf6ba5712	9baea5741ae987bd58adf2e924567666e77a577f
IceKey.class	713ea44ba4852be0f44dd86797277ed8	bbb5eacb906b0f9ebe21569fdf0b48ce395713f4
IceKey.java	00b87c93be3876ae8c1a8ff35fb5d123	e43cbedfcec730f4821b713b03d8fd07afd92d79
jsnow.jar	3c37e653d3b15f566008ca34989f5ebc	15a4937bbe61431953b7300e31cb184c15ea1974
jsnow.zip	531126c5266da5ebfa1acece135e9f22	c59421327f8b44ab1377c06b1b43fd222bb734a5
jsnowapp.html	1a0f0b2cbd1430b73196cadb0857abdd	71bdae3aba3cf01ee8075d9923cf7038dc5cad82
jsnowapp.zip	1b2b3cf0856169d0cd2011554a1e4b0b	52b835762c185dedba5e406965253d36864fc8e9
main.c	043ddc03211445aec064bddd086f0ee6	c4d2e212204cb9689948e33d43bc050cff498b06
Makefile	0fd3d98b3014e3a9b78933718dfc9585	64390177cb160744227bfc98d75e49ab7c9e333e
README	296f1ee78e504c8d724bd7befdf3a5c6	12f6525067f7866bdd47ccc7f64c30b035e38dea
snow.1	fa5c252a7c83258fcb94e0eadf309222	d97b643b3bf7ddfd7ec530fa7fd04f61d1db77a2
SNOW.DOC	6bad2928736aef5dd77a5ef8d0008668	02fe7d3d8c13626079574726836547991178be5c
SNOW.EXE	db3c070a2078b747f63737e8d2e5c8dc	414e1b75ca52a1474669bf0e63ee26ea0e265982
snow.h	fc0a11a78a27bf405c0c8c38f8e5e142	210f269e42af0ae043661bca0e6c9ae292d8e287
Snow.java	afcd92e9ee89a51b2ccdb7a2f7aca2a4	325dc16ddf6d54de6d499f31e7fd5240071f398f
snow.tar.gz	002780331bef06c50d650d45c359f32e	b947177df3f979fc573ba691a7aa1fa5bbb5d1b7
snow.zip	2d9d934b801b619d53fc727106a416b3	4fb651881d95b979b6b7fa5a096e674029050119
SnowCompress.class	1010a6a3a2ddffd1e1d8f942dd881e46	4b0dcba5ebf862c432fa5f62e1e1bdd30ddd3d9a
SnowCompress.java	e53ca34b39dc374f99a2dee051eaa2ed	24ebebbab10d69d52bdda4793ee2b0fbcc98c3c9
SnowEncode.class	136d50a658327d15c156337d38a32b72	d2a0b23f98299743adb22505b70d3d33883007d7
SnowEncode.java	1e4b1a67f47b160abb63bd29d5bccb88	37977cc3ac7cf495bef492c822ca908d7ea08693
SnowEncrypt.class	8f4357e3eb30a2f050d4f4cefdc0db34	9a54a5ab32fe927f10e672752c3b31f823473fa4
SnowEncrypt.java	1f2d4096b5cbff18800b20deab3266c1	e642efd4b63c8beebe58fa24a9708b6954b04dd2
SnowFront$VPanel.class	008ab273a344b1779e7c3914fa98ebfe	2f04fb41f381cdc28d4d282cde4b187a4902fe0e
SnowFront.class	d467bcb91efc78ac77258c2180785e2a	364ffb24e2edaef4fc2aafece62d1e86ea8f9c8d
SnowFront.java	939dda6bb4dd0c04a51953efd1218d1e	efa7602ac542347f5426129161012e45eb79a75a
SnowOutput.class	6d9a4259f3fcc7750bebf6a91b92c125	092269f7eb55e36806f9be7d4de09d81c106dbac
SnowOutput.java	26762ed9886f2f3a8d369097b12ed0fa	c59ae388c325c23748f942f00c9313111f6667e7
snwdos16.zip	8c5a1802a7d285aa548ae3e2faaed44a	54513f741948727bdc52a9d954632966718a28de
snwdos32.zip	927d51a0f5b2c967d58cbaf2d5f6a0f1	6410ece0a45bc7795754bbcdc8dfc7b7d34cbda2

Table A.1: Signatures of known steganography tools. The different signatures are also found on the cd following this master thesis as files formated as the output from the *md5sum* and *sha1sum* tools. They follow the following naming convention "toolName.(md5|sha1)".

g