

Master Thesis
Software Engineering
Thesis no: MSE-2002:24
August 2002



Information Hiding

Steganographic Content in Streaming Media

Peter Bayer
Henrik Widenfors

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Authors:

Peter Bayer

E-mail: peter.bayer@erotechtelub.se

Henrik Widenfors

E-mail: henrik.widenfors@erotechtelub.se

External advisors:

Jan Jönson

AerotechTelub AB

SE-291 39 Kristianstad, Sweden

Phone: +46 44 20 86 05

E-mail: jan.a.jonson@erotechtelub.se

Magnus Andersson

AerotechTelub AB

SE-251 89 Helsingborg, Sweden

Phone: +46 42 18 22 43

E-mail: magnus.b.andersson@erotechtelub.se

University advisor:

Prof. Rune Gustavsson

Department of Software Engineering and Computer Science

Department of

Software Engineering and Computer Science

Blekinge Institute of Technology

Box 520

SE – 372 25 Ronneby

Sweden

Internet : www.bth.se/ipd

Phone : +46 457 38 50 00

Fax : +46 457 271 25

ABSTRACT

For a long time, information hiding has focused on carriers like images and audio files. A problem with these carriers is that they do not support hiding in new types of network-based services. Nowadays, these services often arise as a consequence of the increasingly demand for higher connection speed to the Internet.

By introducing streaming media as a carrier of hidden information, hiding in new network-based services is supported. The main purposes with this thesis are to investigate how information can be hidden in streaming media and how it measures up compared to images and audio files. In order to evaluate the approach, we have developed a prototype and used it as a proof of concept. This prototype hides information in some of the TCP/IP fields and is used to collect experimental data as well.

As reference, measurements have been collected from other available carriers of hidden information. In some cases, the results of these experiments show that the TCP/IP header is a good carrier of information. Its performance is outstanding and well suited for hiding information quickly. The tests showed that the capacity is slightly worse though.

Keywords: Steganography, streaming media, information security, information threats, secret communication.

ACKNOWLEDGEMENTS

First of all, we would like to thank Jan Jönson at AerotechTelub AB for giving us the opportunity to perform our master's thesis at the company.

The daily link between the company and us has been with one of the technical experts within the information security group, our advisor Magnus Andersson. Thanks for many profitable conversations and good advice during the spring. We are looking forward a future work together with you and the company.

We would also like to thank the rest of the information security group for their shown interest concerning our research and the valuable comments given.

At the institute, we would like to thank our supervisor Prof. Rune Gustavsson for the support and encouragement he has given us.

TABLE OF CONTENTS

| | |
|---|------------|
| ABSTRACT | I |
| ACKNOWLEDGEMENTS..... | II |
| TABLE OF CONTENTS..... | III |
| LIST OF FIGURES | V |
| ABBREVIATIONS | VI |
| 1 INTRODUCTION | 1 |
| 1.1 GENERAL | 1 |
| 1.2 AIM, OBJECTIVES AND EXPECTED RESULTS..... | 1 |
| 1.3 THESIS OUTLINE | 2 |
| 2 BACKGROUND | 3 |
| 2.1 GENERAL | 3 |
| 2.2 DEFINITIONS | 3 |
| 2.3 INFORMATION HIDING..... | 3 |
| 2.4 STEGANOGRAPHY | 5 |
| 2.5 STREAMING MEDIA | 8 |
| 2.6 SUMMARY | 8 |
| 3 IMAGES AND AUDIO | 9 |
| 3.1 INTRODUCTION | 9 |
| 3.2 GENERAL ABOUT IMAGES | 9 |
| 3.3 IMAGE – JPEG..... | 10 |
| 3.4 IMAGE – PALETTE-BASED..... | 11 |
| 3.5 IMAGE TOOLS | 12 |
| 3.6 AUDIO | 12 |
| 3.7 AUDIO TOOLS | 13 |
| 3.8 SUMMARY | 14 |
| 4 STREAMING MEDIA | 15 |
| 4.1 INTRODUCTION | 15 |
| 4.2 STREAMING DELIVERY TECHNIQUES..... | 15 |
| 4.3 STREAMING FORMATS AND PROTOCOLS..... | 15 |
| 4.4 STREAMING TOOLS | 16 |
| 4.5 TRADE-OFFS | 17 |
| 4.6 POSSIBLE HIDING TECHNIQUES | 17 |
| 4.7 SUMMARY | 18 |
| 5 MEASUREMENTS AND EVALUATION | 19 |
| 5.1 ATTRIBUTES | 19 |
| 5.2 INTERPRETATION OF THE DIAGRAMS | 19 |
| 5.3 DETECTABILITY ATTRIBUTES..... | 20 |
| 5.4 STEGHIDE..... | 21 |

| | | |
|----------|--|-----------|
| 5.5 | JPHIDE | 24 |
| 5.6 | OUTGUESS 0.2..... | 25 |
| 5.7 | HIDE4PGP..... | 26 |
| 5.8 | DISCUSSION | 26 |
| 5.9 | SUMMARY | 27 |
| 6 | EVALUATION OF TCP/IP HEADER AS COVER..... | 29 |
| 6.1 | INTRODUCTION | 29 |
| 6.2 | MOTIVATION | 29 |
| 6.3 | HIDING INFORMATION IN THE TCP/IP HEADER..... | 30 |
| 6.4 | EXPERIMENTAL ENVIRONMENT | 31 |
| 6.5 | THE PROTOTYPE | 32 |
| 6.6 | THE EXPERIMENTS | 33 |
| 6.7 | DETECTABILITY DISCUSSION | 35 |
| 6.8 | SUMMARY | 35 |
| 7 | CONCLUSIONS | 37 |
| 7.1 | GENERAL..... | 37 |
| 7.2 | RECOMMENDATIONS | 37 |
| 7.3 | FUTURE WORK..... | 38 |
| 8 | REFERENCES..... | 39 |
| | APPENDIX A | 41 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: Information hiding. | 4 |
| Figure 2: Security layers..... | 6 |
| Figure 3: Trade-offs for steganography..... | 6 |
| Figure 4: Trade-offs for streaming media..... | 17 |
| Figure 5: An example of the efficiency diagram used..... | 20 |
| Figure 6: The IP header. | 31 |
| Figure 7: The TCP header. | 31 |
| Figure 8: A physical view of the experimental environment. | 32 |
| Figure 9: The prototype. | 32 |

ABBREVIATIONS

| | |
|-----------|--|
| A/D | Analogue-to-digital. |
| AU | An audio file format, created by Sun. |
| Blob file | Binary large object (of any file format). |
| BMP | Bitmap Format. |
| CIA | Central Intelligence Agency. |
| D/A | Digital-to-analogue. |
| DCT | Discrete Cosine Transform, an algorithm used for reducing bit-rates in digital images. |
| FTP | File Transfer Protocol. |
| GIF | Graphics Interchange Format. |
| IP | Internet Protocol. |
| JPEG | Joint Photographic Experts Group. |
| LAN | Local Area Network. |
| LSB | Least Significant Bit. |
| LZW | Lempel-Ziv-Welch, a data compression technique. |
| MIME | Multipurpose Internet Mail Extensions. |
| MMS | Multimedia Messaging Service (in Chapter 1). Microsoft Media Server. |
| MMST | Microsoft Media Server using TCP. |
| MMSU | Microsoft Media Server using UDP. |
| MP3 | MPEG-1 Audio Layer-3 is a standard technology and format for compressing a sound sequence into a very small file (about one-twelfth the size of the original file) while preserving the original level of sound quality when it is played. |
| NIC | Network Interface Controller. |
| NSA | National Security Agency. |
| OSI | Open Systems Interconnection. |
| PCX | PC-PaintBrush, a bitmap format. |
| PGM | Portable Graymap, a bitmap format. |
| PICT | Picture, an image format for Macintosh. |
| PKI | Public Key Infrastructure. |
| PNG | Portable Network Graphics, a bitmap format. |
| QoS | Quality of Service. |
| RAM | Random Access Memory. |
| RLE | Run Length Encoding, a simple form of data compression encoding. |
| SMS | Short Message Service. |
| SYN | Synchronization. |
| TCP | Transport Control Protocol. |
| TG File | Traffic Generating File. |
| TOS | Type of Service. |
| UDP | User Datagram Protocol. |
| VOC | An audio file format, created by Creative Labs. |
| WAV | A WAV file is an audio file format, created by Microsoft. |

1 INTRODUCTION

1.1 General

Today, when broadband is being installed all over the world, it becomes easier for people to send and receive information between each other. The amount of data packages increases when new services are offered and used via the Internet. The larger amount of data packages, the more information is let through and new possibilities to hide extra information, i.e., in the cover of something else, may be introduced.

The threatening picture against companies and authorities increases and new possibilities for hostile and illegal actions are easier concealed in a large information flow. In a mobile context, the possibility of sending MMS, as a complement to SMS, offers information to be hidden in images and audio as well as in the actual text message. The support for embedding objects in e-mails also gives the encoder several chances to hide information. The encoder can make use of the appearance of different kinds of media in the hiding process.

Business secrets may be easier hidden and sent out via seemingly innocent e-mails or embedded in the normal outgoing network traffic flow. The possibilities of detection are very limited because of the large set of algorithms for hiding information. Today's engineers, within the area of information security, have to struggle against apparently impossible threats. Steganography, the hiding of information, is not only interesting from a security perspective because of the vulnerabilities it may bring, but also the possibilities. In this thesis, examples of methods, possibilities, threats and vulnerabilities concerning hidden context will be given.

This master's thesis is written during the spring of year 2002. The authors are both students within Software Engineering at Blekinge Institute of Technology in Ronneby, Sweden. The thesis has been rooted in industry by an exchange with AerotechTelub AB, a company in the Saab Technologies Group. In Sweden, AerotechTelub has about 2,650 employees and the main customer is the Swedish National Defence.

1.2 Aim, objectives and expected results

The aim is to investigate how steganography can be used to hide information in binary media with a focus on streaming media.

The objectives with the research are to:

- Summarize the maturity of technology for steganography and *state-of-the-art*.
- Identify suitable methods for evaluation in respect of efficiency and performance.
- Evaluate different kinds of information carriers, i.e., file formats, in respect of efficiency and performance.
- Identify and evaluate how information can be hidden in streaming media.

- Identify detection methods for different carriers and media.

The expected results with this thesis are to illustrate the possibilities and threats with steganography. The elaborative part of the thesis shall deliver measurements about the effectiveness and capacity for hiding data in different kind of media. The result shall give an indication about the strength of hiding data in streaming media.

1.3 Thesis outline

Chapter One gives the reader basic information about the thesis. The scope, aim, objectives and the expected result with the research are given. *Chapter Two* will introduce the reader to the area of steganography and streaming media and offer important knowledge and background history for the understanding of the following chapters. *Chapter Three* is the review part of the thesis that handles the *state-of-the-art* knowledge found in literature concerning images and audio. In *Chapter Four*, information about streaming media and its characteristics are discussed in more detail, e.g., streaming techniques and streaming protocols. Tests and measured results on static media are documented in *Chapter Five*. *Chapter Six* gives a proof of concept for hiding in streaming media and the strength with it. Also, test results are included and a discussion about these.

To conclude the thesis, *Chapter Seven* highlights the major findings of the thesis and gives examples of future works in the area. The rest of the thesis contains information about referenced articles and books, but also an appendix with test results. Lists of abbreviations and figures are found just before Chapter one.

2 BACKGROUND

2.1 General

The purpose with this chapter is to introduce a basic knowledge and give an overview about steganography and streaming media that is required when reading the rest of the thesis. Different authors use modified or narrowed definitions of some of the concepts in this thesis.

2.2 Definitions

There are a variety of definitions found in literature concerning steganography and streaming media, especially for the last one. There are authors claiming that streaming media is limited to include only audio and images, like a movie for example. In this thesis a broader definition of the term is used and it is stated below.

2.2.1 Steganography

Steganography is a collection of ways of embedding secret messages. Binary steganography means that binary information is hidden in the cover of another binary kind of media.

2.2.2 Streaming media

Streaming media means a continuous transaction of information, i.e., all data is not needed before the receiver can take part of the information. The parts of the stream are independent on each other.

An example

A TCP/IP header contains all necessary information a packet needs to be delivered to its destination. The headers are independent of each other even though the contents in the packets may be dependent on the other ones. According to the definition, the headers can be seen as streaming media.

2.3 Information hiding

When discussing different hiding techniques, the following restrictions and features are desirable [[Bender et al. 1996](#)]:

The hidden information...

- shall be very little perceptible.
- should be directly encoded into the media, rather than into a header or a wrapper.
- should not be lost if modified by conversion, lossy compression, re-sampling, etc.
- should be embedded using asymmetrical coding (the use of public and private keys), which makes the exchanges of keys easier.
- should include error correction codes since manipulation of the cover media often leads to problems with the data integrity.

- should be self-clocking or arbitrarily re-entrant. This means that if only a part of the cover media is available, the hidden information within that part should be possible to extract.

In the area of information security, information hiding can be scheduled as a tree with branches like in the figure below.

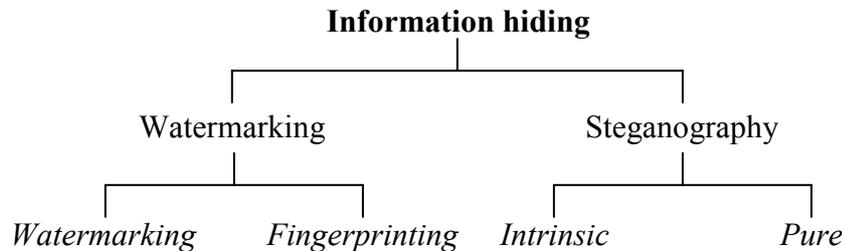


Figure 1: Information hiding.

Watermarking means that some kind of identification code is permanently embedded, e.g., in the cover of an image or an audio file. This can, for example, be used in industry when one wants to prove the copyright of a work. In software industry, it can be used to limit simultaneous use of licenses and to trace the source of distribution when illegal copies of software products are found. When a unique identifier is hidden, like a user id, it is called fingerprinting (or sometimes mentioned as *traitor tracing*).

Steganography is used to hide messages in the cover of something else. In intrinsic steganography a secret key (pass phrase) is used when embedding a secret message in a cover and therefore also in the extracting process. The difficulty is that there must be an exchange of secret keys. This differs between intrinsic and pure steganography, because pure steganography does not require these kinds of exchanges, i.e., no keys are needed [Korjik & Morales-Luna 2001].

When hiding information, in for example a JPEG image, there exist different algorithms that take advantage of the way in which the image is stored. More information about this is found in Chapter 3. Researchers around the world have focused on hiding and detecting hidden content in images. The research of audio has been less published, most common are research papers concerning the MP3 and WAV file formats.

2.3.1 State-of-the-art

There exist methods for detecting hidden content for some algorithms used in a specific media. When suspecting an image to be manipulated, it can in some cases be guaranteed at a level of 97% [Farid 2001]. When a file is known to be manipulated, there is still a problem knowing where to start decoding it from and which bits that are manipulated in the cover file. If the algorithm used for hiding the message uses a pass phrase, it is today extremely difficult to find it and get a good result within reasonable time, e.g., by applying brute force. A survey, searching two million images for hidden information, found out that 17,000 of the images might include something extra. There were no guarantees given or

evidence found saying that the suspicious images actually included hidden information, since nothing could be extracted from the suspected set [Provos & Honeyman 2002].

2.4 Steganography

2.4.1 History

The need of hiding information has existed a long time back in history. Steganography literally means *hidden writing*. The ancient Greeks and the Romans shaved slaves' crowns and hid messages on their heads. Then, when the hair had grown out, each slave was sent to a receiver, who shaved the crown again to read the message. In China, they hid a code ideogram at a prearranged place in a dispatch. The hidden code where then uncovered by putting a template over the message. The list of examples when some kind of steganography has been used is long and can be fun reading for those fond of history. Obviously, these kinds of information hiding techniques are not optimal and are not applied directly in today's applications.

The armed forces have used steganography ever since the first appearance of it in history. A more uncertain matter is what knowledge secret services like NSA know about it. There are probably several internal classified documents describing this in detail. This thesis is therefore based on public reports in the area.

Some authors will state the scientific study of steganography to be when Simmons formulated the "Prisoners' Problem" in 1983 [Anderson & Petitcolas 1998]. The problem takes place in prison where two prisoners are trying to devise schemes for an escape. The third party is a warder that may read each message before he delivers it. The idea is based on how the prisoners can hide information in open letters and keep their secret communication channel remained secret. The problem was considered both with an active and a passive warder and with and without public key steganography. The result was that public key steganography was possible in some cases even if the warder was active.

The first international workshop about information hiding was held in Cambridge 1996 and steganography was a separate part of the workshop [Anderson 1996]. The art of information hiding has, as previously mentioned, an ancient history though.

2.4.2 Why steganography?

Cryptography is used when someone wants to hide information from being read as plain text. If a piece of text looks suspicious, it is easy to suspect that someone wants to hide something for the reader. When having an encrypted message, different kinds of decrypting methods can be applied, e.g., dictionary attacks or more time-consuming brute force methods. With steganography, the opportunity to suspect that there is something hidden, is not given. This means that the level of security has increased by at least one step (a hiding layer). Also, another layer can be motivated, i.e., a scatter layer (see Figure 2).

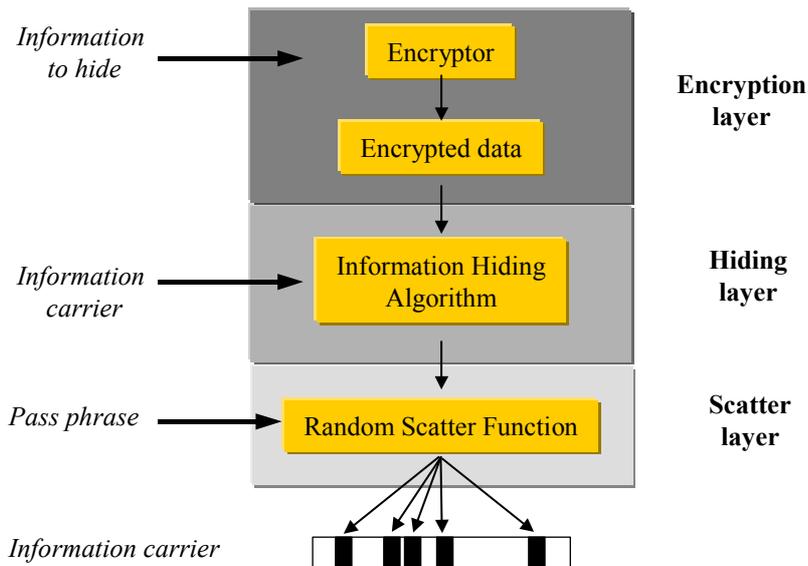


Figure 2: Security layers.

An example of an intrinsic way of hiding (cp. Figure 1).

Suspecting, discovering, extracting and then in some cases trying to decrypt hidden messages are today very problematic. Since information is hidden in the cover of something else, it becomes more difficult to find it. For example, if a secret message includes 10,000 characters and is hidden in an image with a size of 100 kB, the algorithm used must be known and probably also a pass phrase that the algorithm uses when hiding the information.

2.4.3 Possibilities

Steganography offers many possibilities that can be either positive or negative. The positive characteristics are discussed under this section and the negative characteristics in Section 2.4.4.

When talking about possibilities, the important trade-offs for the strength of using steganography need to be considered. There are dependencies between *detectability*, *robustness* and *capacity* (see Figure 3). Performance is also an important trade-off attribute, especially when trying to detect hidden content.

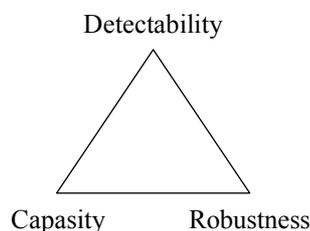


Figure 3: Trade-offs for steganography.

The higher the requirements for large capacity are, the easier it is to detect that something is divergent with visual or statistical methods. The robustness against transformations, e.g., A/D and D/A conversion, compression, scaling and

cropping, is also of importance. A higher capacity makes the information more sensitive for transformations and then information may be lost.

The possibilities of detection for different static information carriers are discussed in Chapter 3.

Examples of applications for steganography:

- Digital watermarking, e.g., copyrights and licenses.
- Tamper-proofing, i.e., to guarantee that a media has not been modified (e.g., hiding a hash value).
- A secret chat channel via web radio.
- Authentication of a user, e.g., by means of an image.
- Use of existing communication channels for sending authorization information without suspicion.
- Hide PKI-communication.
- Building a transparent file system. One way is to hide the file system in a seemingly innocent blob file. Another example is that files in a transparent file system are split up and set out according to a specific pattern (algorithm).

2.4.4 Threats

There are always people and organisations that use technologies for illegal purposes. Software programs that help to hide something are therefore not an exception. Criminals take favour of this in the binary world in a similar way as in the real world. If a thief grabs a jacket, he will surely hide it in a bag or behind his other clothes. While walking towards the door, the shop assistant may suspect the crime and stop him. Since the thief cannot prove his innocence by a written receipt, he is caught red-handed. In the binary world, the time for suspecting is shorter and even if the binary thief is caught, the stolen binary jacket is hidden so well that a search through the body cannot prove what and if something has been stolen. How this should be handled by a detection mechanism is very dependent on in which context the mechanism works.

According to CIA, the three greatest fears in America today are bio-attacks, nuclear attacks and steganography [UP 2001].

Internal espionage may easier succeed, because it might be easier to send out information from an intranet and avoid suspicion. The risk of detection is often lower than if a person tries to walk out of a well-secured company carrying a floppy disk or a CD.

The restriction of exporting cryptography tools in, e.g., US, may strengthen steganography as an alternative.

After 11 September 2001, military experts believe that the Taliban leader *Osama bin Laden* has used steganographical techniques to communicate with terrorists all over the world [CNN 2001]. They suspect that he has used web images as cover for secret instructions. On the other hand, a research report by Niels Provos and Peter Honeyman did not result in any evidence that holds for this theory [Provos & Honeyman 2002]. The results of this survey are mentioned in Section 2.3.1.

Secret communication between criminals and terrorists is a threat against the whole world and if they use common information carriers as cover, it will be difficult to detect their communication channels.

2.5 Streaming media

A common field of applications for streaming media is with the most used clients, i.e., *RealPlayer (RealNetworks)*, *Windows Media Player (Microsoft)*, *QuickTime (Apple)* and *WinAmp (Nullsoft)*. It has become popular to listen to web radio and to see trailers and pre-recorded news summaries as a consequence of the increased access to the Internet. The streaming information does not have to be completely downloaded before the user can take part of it and often the user can choose between different qualities of the streaming information as well. To set up a server that sends out streaming media requires a lot of bandwidth, especially when a larger number of simultaneous listeners are connected.

2.6 Summary

The use of steganography can be traced a long time back in history, but as a science it is not very old. In a binary context, steganography offers some new fields of applications. The techniques can also be used in an illegal manner, not least as a consequence of the increasing amount of information available and information exchanged. An increased accessibility to the Internet has been important for this evolution.

Streaming media is a new concept and the definition about what it really is differs. In this thesis, the concept is not limited to include audio and video, but any kind of information that is sent and can be handled in real time fits the definition.

Next chapter summarizes today's work in the area of steganography including different hiding techniques in different media.

3 IMAGES AND AUDIO

3.1 Introduction

In the middle of 1990 several authors began presenting their work about digital steganography. With their work a new method for confidentiality was born. As mentioned in previous chapter, the fundamental of digital steganography is to hide the very existence of information for potential eavesdroppers. This approach is completely unlike cryptography where information is scrambled, thus unreadable for an eavesdropper. In a matter of years, techniques for digital steganography were developed and applied to information carriers. At first these techniques were considered to hide information in an undetectable fashion. The goal of perfect confidentiality seemed to have been reached.

There are two sides involved in information hiding; one side focusing on hiding the information and the other focusing on breaking the secret communication by revealing the existence of the hidden information. A good analogy is the struggle between cryptographers and cryptanalysts, which have been going on for longer than the past two millenniums. For an historical overview of this struggle Simon Singh [[Singh 1999](#)] and David Kahn [[Kahn 1996](#)] provides an overview. Ever since methods for information hiding were developed there has been people dedicated at trying to break them. And soon after the first hiding techniques were presented, the steganalysts began their work. At first, focus of their interest were tools developed in order to apply and explore the known hiding techniques. Since the information hiding at first solely concentrated on images as information carrier, so was the steganalysis.

In this chapter, the most common carriers and hiding techniques will be described, how they have been applied to some of the most well used information carriers and how they have been defeated by the steganalysts. Since research focusing at images and audio, these are the two carriers being discussed. With images, three well-known formats will be handled: JPEG, GIF and BMP. In the section concerning audio, a more general discussion is given.

3.2 General about images

With information hiding in digital images, three elements are required. First, the information to hide, often referred to as the *message*. The message is hidden (or embedded) in, what is called, a *cover image*, i.e., the information carrier. During the embedding phase an *algorithm* is required to determine how the message is embedded. This algorithm can be more or less advanced, ranging from simple LSB embedding in the spatial domain to bit scattering in the frequency domain.

The actual hiding process starts with embedding bits of the message into the cover image. The result is an image, called *stego image* and contains the original image with the embedded message inside.

A distinction is made between palette-based and non palette-based images. An example of the latter is the Joint Photographic Experts Group (JPEG) format. The

Graphical Interchange Format (GIF) and Bitmap Image Format (BMP) are examples of the palette-based format.

3.3 Image – JPEG

3.3.1 General

JPEG is a standardized lossy image compression method commonly used on the Internet. The term lossy compression refers to that some of the original information is lost during the compression process. With JPEG, known limitations of the human eye are exploited to determine which and how much information can be lost until the human eye sees a difference.

Since JPEG is commonly used on the Internet it is a good candidate for carrying hidden information. These types of compressed images have got a lot of attention in research and numerous applications exist that offer information hiding in this format. JSteg, JPHide and OutGuess are examples of such steganographical software tools.

3.3.2 Hiding techniques

To understand the technique used for information embedding in a JPEG compressed file, basic knowledge about the compression process is required. The process starts with a decomposition of the image into blocks of 8x8 pixels. By applying a discrete cosine transformation for every group, 64 DCT coefficients are produced for each group. These coefficients are then quantized and rounded off to integers. Finally, the integers are compressed with Huffman coding.

The most common approach of data embedding in a JPEG compressed image uses LSB embedding. The embedding is performed before the compression starts and is applied on the quantized coefficients.

3.3.3 Detection

There is no evidence supporting visual detection of messages in JPEG images. This is of course a victory for the steganography community as they have succeeded with their goal of hiding information in a way that avoids visual detection. The steganalysts early realized that visual attacks were not a good detection method for JPEG images. The major cause of disbelief was that a visual attack would be insufficient when the number of images to analyse increased. In other words, the required resources would be too large and the detection process would consume too much time. Instead, they concentrated on an automatically and computer based detection mechanism. Such an automatically detection mechanism would clearly improve its practical use with respect to time and costs.

In [Pfitzmann & Westfeld 1999] another detection method is presented. This method detects data embedded by the LSB technique and is not limited to palette-based image formats. The core of the method is that there exist statistical dependencies between colour frequencies in an image. Another prerequisite is that encrypted data is random, i.e., at a bit level there are as many zeros as ones. By combining these facts, the probability of detecting embedded information can be

calculated for different areas of an image. With this method, detection was first limited to images where data was embedded sequentially. By applying small adjustments to the method, random scattered embedded data can be detected as well. These adjustments and a way of defending against the Pfitzmann and Westfeld detection method is presented in [Provos 2001]. Provos shows that it is possible to preserve the colour correlation by embedding the data in a smart way. His work is applied to the JPEG format only, but might also be applicable to palette-based formats.

In [Fridrich et al. 2001] another approach is taken. The approach is different from the one proposed by Pfitzmann and Westfeld as it is not able to detect information hidden in the LSB of the quantized DCT coefficient. Instead it can be used when the cover image previously have been saved in the JPEG format. What actually happens when an image is saved as a JPEG file is that special fingerprints are introduced with the file as a result of the compression. When information is embedded these fingerprints are disturbed. By disturbing the fingerprints the JPEG compatibility is destroyed. Now, by checking for this compatibility it can be determined if the image contains an embedded message or not. Even small quantities such as one embedded bit can result in incompatibility, thus making it detectable.

3.4 Image – Palette-based

3.4.1 General

GIF is a lossless image format that uses a palette and compression to produce a small output file. The maximum colour depth is limited to 256 colours, i.e., 8 bits. The compression algorithm used is LZW (Lempel-Ziv-Welch).

BMP is another palette-based image format with a colour depth of 8 to 24 bits. 4-bits and 8-bits RLE (Run Length Encoding) compression are supported, but seldom adopted. A palette is often used when the colour depth is below 24 bits. In images with a colour depth of 24 bits, no palette is used. Instead, each pixel is represented as three bytes (one each for red, green and blue).

3.4.2 Hiding techniques

The most commonly used hiding method, together with palette-based image formats, is LSB embedding. With this method, great care must be taken or otherwise the final stego image may be vulnerable to visual attacks. The problem arises when colours in the palette are very unlike each other. At one extreme, two completely different colours may be swapped in the stego image. Imagine a two-coloured image of a red filled quadrant with black borders. If at one extreme the colours were swapped in this case, the result would be a black quadrant with red borders.

To solve this problem there are methods available [Johnson & Jajodia 1998]. Johnson and Jajodia propose two such methods. The first method involves sorting the palette thus making the colour swapping more robust. The second method proposed extends the palette by introducing new adjacent colours. For the steganalysts, both approaches are good as they introduce recognisable patterns. The pattern introduced in the first method is a sorted palette, which is rare in the

normal case. With the second method many adjacent colours will exist. If any of these patterns are found in a palette-based image format, the probability of embedded information is high. An extension to Johnson and Jajodias' first method is proposed in [Fridrich 1999]. This extension shows a better way to calculate adjacent colours, thus making the colour swapping more robust.

3.4.3 Detection

Visual detection of LSB embedding in palette-based images is easier than with JPEG since the information is hidden in the spatial domain [Provos & Honeyman 2002]. When an 8-bit colour depth is used, the palette-based image formats may become highly vulnerable to visual attacks. This applies only if the colours in the palette differ a lot. As discussed in the previous section, special methods exist that can handle this problem. However, these methods introduce recognisable patterns making them detectable.

In 1999 Pfitzmann and Westfeld [Pfitzmann & Westfeld 1999] presented a detection method that was not based on visual observations. Instead their method used statistical constructs to reveal hidden information in an image (see Section 3.3.3 for a more detailed description of the method).

3.5 Image tools

There exist a lot of steganographical tools for message embedding in images. A random selection of available tools has been made and is presented in the table below. As with the steganographical research, most of the available tools allow embedding in JPEG and BMP/GIF images.

| Software products | Author/Company | Image type(s) supported |
|--|----------------|-------------------------|
| Steganos Security Suite 4 (Shareware) | Steganos GmbH | BMP |
| S-Tools (Freeware) | Andrew Brown | BMP, GIF |
| Stego (Freeware) | Romana Machado | GIF |
| StegoDos (Freeware) | Back Wolf | GIF, PCX |
| EzStego (Freeware) | Romana Machado | GIF, PICT |
| JPHIDE/JPSEEK (Freeware) | Allan Latham | JPEG |
| JSteg (Freeware) | Derek Upham | JPEG |
| OutGuess (Freeware) | Niels Provos | JPEG, PNG |
| Piilo (Freeware) | Tuomas Aura | PGM |

3.6 Audio

3.6.1 General

Information hiding in audio is very challenging because of the wide range of the human auditory system. An average human audible frequency spectrum ranges between 80-20,000 Hz and the human ears are good at hearing small variations in pieces of music and speech. The possibility that the human ears hear some kind of noise or echo and associate it with information hiding is, on the other hand, to be considered as very low. It can be a common disturbance or a conscious effect by a music maker as well.

The sampling rate has a direct connection with the amount of information that can be hidden per second. The higher sample rate the greater amount of information can be hidden. The transmission environment is also of importance for the robustness of the hidden information. If audio is sent over the air or re-sampled in any way, parts of the hidden information may be easily lost.

3.6.2 Hiding techniques

Low-bit coding is the easiest way to hide information in audio and works in a similar manner as for images, i.e., the LSB is changed. The capacity is good, e.g., 44 kbits can be hidden in a 44 kHz sampled sequence for each second [Bender et al. 1996]. The robustness is worse. A transformed audio sample loses parts of its hidden content easily.

From a robustness perspective, *phase coding* is better. First, the sound signal is divided into segments and each segment is transformed into a phase and a magnitude. By calculating the difference between phases each phase is modified and combined with the original magnitude, building a new segment. The method is more complicated than low-bit coding, but offers a better protection for the hidden data.

Spread spectrum hiding spreads out information in the frequency spectrum and is based on a pass phrase. As much of the frequency spectrum as possible is used and this leads to that uncontrolled noise may appear. Also, by not limiting the frequency spectrum for hidden data, more bandwidth is required [Manamalkav 2002].

Finally, a method called *echo data hiding* embeds data by introducing an echo. Three of the parameters are changed; *initial amplitude*, *decay rate* and *offset*. By manipulating the delay between the original sound and the echo, a one or a zero can be hidden. The method is quite complex but has shown to be good on audio files where there is no additional degradation, such as from line noise or lossy encoding, and where there are no gaps of silence [Manamalkav 2002].

3.7 Audio tools

In comparison to images, there are considerably fewer software products on the market focusing on audio. In the table below, examples of software tools for hiding information in audio media are listed and also which audio types they support.

| Software products | Author/Company | Audio type(s) supported |
|-----------------------------------|-------------------|-------------------------|
| Data Stash (Shareware) | Guan Inc. | Any binary |
| Hide4PGP (Freeware) | Heinz Repp | WAV and VOC |
| Invisible Secrets Pro (Shareware) | NeoByte Solutions | WAV |
| MP3Stego (Freeware) | Fabian Peticolas | WAV → MP3 |
| Scramdisk (Freeware) | Sam Simpson | WAV |
| Steganos (Shareware) | Steganos GmbH | WAV and VOC |
| StegHide (Freeware) | Stefan Hetzl | WAV and AU |
| StegoWav (Freeware) | Peter Heist | WAV |
| S-Tools (Freeware) | Andrew Brown | WAV |
| SureSign (Shareware) | Signum Tech. | WAV |

As can be seen, tools that hide messages in standard WAV files are dominating. Only some of the tools support other audio types than WAV. MP3Stego is one of these and it can be freely downloaded from Fabian Petitcolas's web page [Petitcolas 2002]. The tool hides a text message in an MP3 file during the compression process, i.e., WAV to MP3. The software is published as a proof of concept for power of parity and related to the article [Anderson & Petitcolas 1998]. The source code is written using Microsoft Visual C++ and included when shipped.

3.8 Summary

Even though digital steganography is a young science it has been examined quite thoroughly in some special areas. Most of the work has been concentrated at hiding information in images and other information carriers have more or less been left out of the discussions. Lately, a few authors have realized that there are gaps to fill in the research about information carriers besides images.

One conclusion is that the importance of steganalysis should not be underestimated. After all, steganalysis is concerned with detecting the hidden information thus making the hiding techniques obsolete. Most certainly, this results in that techniques are tweaked with and modified in a way that makes them undetectable again. This struggle or kind of evolution brings potential for the coming research in the area of digital steganography. Steganalysis can help deriving more secure steganographical systems as it will look for and attack weak parts of such systems.

Another observation is that almost all research has been concentrated on a few information carriers, i.e., images and audio. In these areas, the struggle continues and hiding techniques and detection mechanism will continue to be enhanced. Research about information hiding in other carriers, e.g., different kinds of word processing documents, would have been interesting. New possibilities and problems arise and are often unique for different carriers when one wants to hide information.

LSB, as a technique, can be used for both images and audio. Otherwise, audio manipulating is more about using specific properties for audio, e.g., the frequency spectrum and audio effects like echoes.

So far, the thesis has concentrated on static information carriers with a well defined start and stop (the beginning and the end of the file). Next chapter discusses what happens if the information is streaming between two computers. Then the start and stop problem must be taken care of. In the literature finding phase, just one article about hiding techniques for streaming media has been found, which seems to indicate that today's research literature seems to be lacking in this area.

4 STREAMING MEDIA

4.1 Introduction

Information hiding combined with streaming media will be a future combination to count on. Research of today focus on information hiding in static content and methods for hiding and detecting hidden information have been developed. Streaming media adds more possibilities to the hiding process and should therefore be taken into consideration. The streaming material can be seen as a static binary file if it is pre-recorded and algorithms for hiding information in a static context can be applied. It is not less streaming if the information is pre-recorded, as long as it is seen as a stream for the receiver.

A more interesting approach is to hide information sequentially in real time in a stream. This way of hiding brings other problems to the hiding process. For example, how can the receiver (decomposer) know from where to start looking and when has the information reached the receiver? To be able to apply a real time hiding technique, one possibility is that the algorithm has to change the information at a packet level (TCP/IP). This means that both the Transport layer and the Network layer of the OSI model are concerned. How this can be done is discussed in Chapter 6. If the hiding algorithms would work in the Application layer and hide information in a protocol like RTSP, they would be easier to distribute and implement since no changes would be necessary into the operating system's source code.

4.2 Streaming delivery techniques

There are two major streaming delivery techniques, i.e., *unicast* and *IP Multicast*.

Unicast is a two-way-communication, i.e., the client can communicate with the server simultaneous as the streaming is going on. The server only sends the information to those clients requesting it. A less good effect concerning streaming is that this method requires a great deal of network bandwidth.

IP Multicast is better on minimizing the bandwidth. A server sends a single copy of a stream over a network and there can be an unknown number of listeners. In contrast to broadcasting, multicasting only sends the information to interested users. Broadcasting sends the information to all users on the network. When multicasting, the clients have no control over the data stream because of the connectionless technique.

4.3 Streaming formats and protocols

TCP and UDP can be used for streaming information, but each of them suffers from some drawbacks. TCP has a built-in feature that guarantees that no packets are lost during transfer. This is time-consuming and may lead to synchronization problems when streaming. UDP has no control of the transport of the packets nor if the receiver gets the packets or not. This may lead to huge loss of data.

Some commonly used protocols today are RTP/RTSP and MMS¹. This thesis will just give a brief introduction to some of the central concepts with these protocols.

4.3.1 RTP/RTSP

RTP (Real time Transport Protocol) provides end-to-end network transport functions for applications transmitting real time data over a network [RFC 1889]. RTP is augmented by a control protocol (RTCP) to monitor the quality of service (QoS). A UDP network environment may experience some problems, like lost packets, jitter, and out of sequence packets. This is taken care of when RTP and RTCP are combined. RTP can be seen as an extended UDP, with a timestamp and a sequence number added to the header. This makes it possible for the client to re-order the packets in a buffer before they become visible for the receiver. The protocol support IP Multicast.

RTSP (Real time Streaming Protocol) is an application-level protocol for control over the delivery of data with real time properties [RFC 2326]. A standard HTTP or MIME parser can parse RTSP and security mechanisms, e.g., basic and digest authorization, can be directly applied. It is also a flexible protocol in the way that new methods and parameters can be added easily. The streams controlled by RTSP may use RTP, but the operation of RTSP does not depend on the transport mechanism used to carry continuous media [RFC 2326].

Both RTP and RTSP are based upon open standards and the specifications are therefore easily accessible.

4.3.2 MMS

Microsoft has a closed protocol for streaming media called MMS. It has mechanisms for both data delivery and control of packets. It works in the application layer on top of UDP (MMSU) and TCP (MMST). It is hard to find detailed information about MMS since Microsoft has the copyright.

4.4 Streaming tools

4.4.1 RealServer

RealNetworks, Inc. has a commercial variant but also a free version of their streaming server RealServer. The server supports RTP and RTSP for streaming information and IP multicasting is set as default. To produce streaming information, another product is needed, i.e., RealProducer. Its purposes are to convert information from a web camera or from a file according to the streaming protocol and send it to a RealServer. It is also possible to for RealProducer to stream to a web server, e.g., Apache, but then with fewer functionality supported.

A software client, e.g., RealPlayer, can view the streaming information. The information is also possible to view as an embedded object in a web browser, i.e., with a suitable plugin installed.

¹ MMS is here an abbreviation for *Microsoft Media Server*.

4.4.2 Windows Media Services

To convert and produce a stream of live or pre-recorded information, Microsoft uses Windows Media Encoder, which is free to download. For broadcasting to more than 50 simultaneous users, a special Windows Media Server is required. The streaming information is easiest accessed via HTTP with a software client like Windows Media Player. Microsoft also offer a plugin for MS PowerPoint that makes it possible to publish slides, images and video created by the program. This plugin is called Producer and is free to download as well.

4.5 Trade-offs

In Section 2.4.3, the dependencies between detectability, robustness and capacity are discussed. These attributes primarily concern steganography as a whole, but they have also some affects on streaming media. More primarily trade-offs for streaming media are, e.g., *bandwidth (server and client), compression, streaming technique, storage, buffer and quality* (see Figure 4).

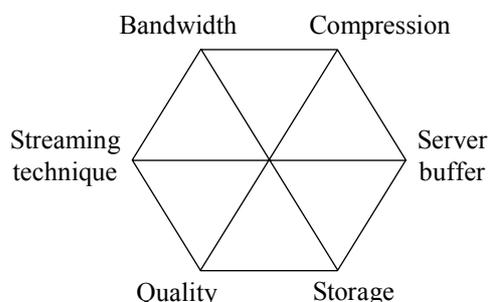


Figure 4: Trade-offs for streaming media.

A change in one of the attributes does not necessarily affect all the other attributes.

These trade-offs are good to keep in mind when discussing streaming media, because they throw light upon some of the problems that may appear when setting up a streaming server environment. The RAM memory and clock speed of the server also play an important role. At least 256 MB RAM and a speed of 700 MHz is desirable for sending real time video in a LAN.

4.6 Possible hiding techniques

As an information carrier, streaming media offers many opportunities to hide information. Streaming media can be considered in two states, i.e., as pre-recorded material or live produced. This makes it possible for two different kinds of hiding algorithms, one that is applied on a static file and one that has to work sequentially (in real time). The algorithm used on pre-recorded material may be more effective since it can affect characteristics of the streaming protocol and therefore hide more information in a given size of the streaming media. Also, the start and stop signals for the hidden information are given by the beginning and the end of the pre-recorded streaming file. A more difficult approach is to hide information in real time, e.g., when sending live. The possibilities to affect properties concerning bit rate and frames/second are difficult. Also, some kind of signals for 'hidden information begins' and 'hidden information ends' should be included. There can also be more advanced approaches to solve these problems,

e.g., added functionalities to the server's and the decoder client's OS kernels (more about this in Chapter 6).

4.7 Summary

IP-telephony, videoconferences and interactive television are concepts that are supposed to be common with the rampaging of broadband. The Internet and internal LANs offer these possibilities if the bandwidth is large enough. The Swedish ICT-Commission recommends a bandwidth of 5 Mbps for a simultaneous use of all activities [[SICTC 2002](#)].

The use of streaming media is expected to increase as the bandwidth increases and this offers new possibilities. It is today possible for anyone to download software and set up an own streaming environment. To produce and send streaming media requires a lot from the hardware as well. By experiments, it can be concluded that the RAM memory is of major importance.

Since this thesis shall discuss information hiding and focusing the research part to information security, it is in place to give an example of how streaming media can be misused.

A possible scenario is that a hacker uses the increased amount of data and applies steganographical methods on it. Since streaming media often consist of large amount of data, it requires a lot of bandwidth. This is of interest for a hacker. If the hacker can take advantage of the great amount of streaming packets and manipulate them, he can hide a lot of information. In the laboratory environment, we will investigate the possibilities to hide information in a streaming context and measure the efficiency of it. This is the main part of Chapter 6. To have something to compare with, next chapter (Chapter 5) evaluates efficiency for static media.

5 MEASUREMENTS AND EVALUATION

This chapter discusses the output from a couple of tests of hiding algorithms in which efficiency and capacity were focused. It also evaluates the advantages and disadvantages between the algorithms and the media. Only algorithms that hide information in a static context, i.e., in a file, have been included in this chapter. Other requirements are that the algorithms are published as open source and possible to compile and execute under Linux.

In other words, the purpose with the tests is to get some reference figures that can be used when comparing the effectiveness in proportion to hiding techniques for streaming media (see Chapter 6). All tests are accomplished in a closed environment with a 450 MHz Pentium II running Linux, 256 MB RAM and 99% of the CPU power available. Each test is iterated three times and there was no clear divergence between the executions. Therefore, the values are considered to be stable and representative.

The cover files used in the tests are all original images and audio files taken and recorded for the purpose of this thesis. The images are in different resolutions and saved with different compression rates. The audio files are of different length in time, but all sampled in a quality of 16 bits and 44.1 kHz.

5.1 Attributes

The following attributes and derived attributes were recorded and calculated during each test:

- Size of the cover file (bytes).
- Size of the stego file, i.e., the cover file after embedding the information (bytes).
- Size of the hidden text. The test files included plain text of different length (5, 10, 20, 30, 40 and 50 kB). A text file containing 50 kB of text is approximately the same as 17 A4-pages of a non-formatted MS Word document (using text font Courier New, 12p).
- Measured time, i.e., the algorithm's execution time excluding the reading and writing to file (ms).
- Ratio between hidden text and the cover file (%).
- Ratio between the stego file and the cover file (%).
- Average time for each byte to be hidden (μ s/byte).

Calculated in percentages, 50 kB of hidden information is a large amount (up to 90%) compared to the smallest cover file for some of the JPEG files, but for audio files less than 4%. This may look strange, but the focus has been set to limit the hiding of a specific amount of bytes in different kinds of media. The media differs in file size, but are all considered representative for their kind.

5.2 Interpretation of the diagrams

Efficiency has been in focus for each test and is measured as the time for hiding each byte of a text file in a cover file (μ s/byte).

X-axis: Time ($\mu\text{s}/\text{byte}$).
Y-axis: Size of cover file (kB or MB).
Dot: A test case. If all test cases are successfully executed for a cover file with a specific file size, six dots are horizontally visible (see Figure 5).

The expected tendency for an algorithm should look like in the diagram below (see Figure 5), i.e., it is acceptable that it takes more time per byte to hide information in a large cover file if the algorithm uses the whole capacity of the cover file in the hiding process. The scatter plot for the dots are expected to be as collected as possible for each cover file.

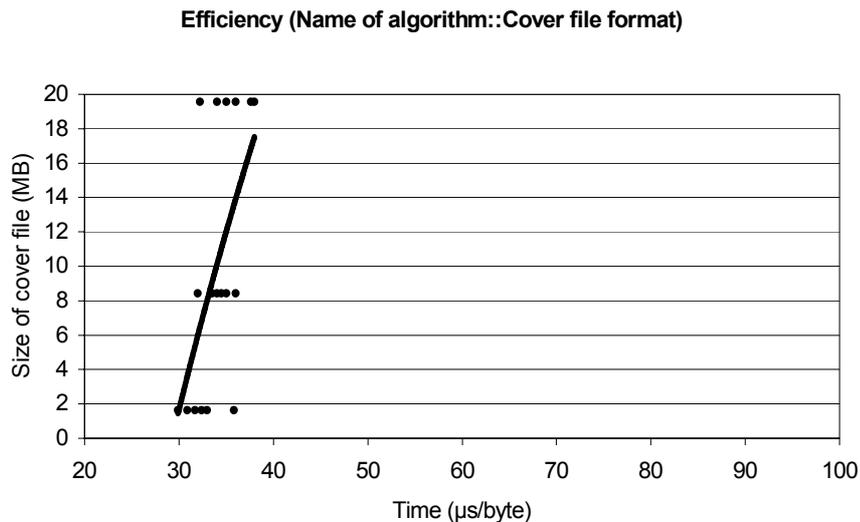


Figure 5: An example of the efficiency diagram used.

Also, the change in size between the cover file (original) and the stego file (manipulated) will be shown, but not graphically.

5.3 Detectability attributes

An important attribute for stego files is that they shall be as less perceptible as possible (see Section 0). The strength of each algorithm has been tested in different ways, i.e., visually or audiovisually and statistically when possible. If hidden information is suspected in an image by just looking at it, it is breaking against the rule of being little perceptible. Searching for hidden information in images, by just looking at them, is a very time-consuming activity that is not applicable for greater sets of images. In other words, it is hard to automate. Therefore, a better approach would be to develop a software application to perform the job in a different manner. A statistical analysis of the image and its characteristics can expose that there may be something hidden. A statistical analysis is easier to automate as well.

StegDetect and StegBreak² are two software applications that use statistical methods to prove exceptional characteristics for JPEG-images and apply brute-force to get the pass-phrase used by the algorithm when hiding. In the latest version (ver. 0.5) of the applications, hidden information can be exposed for five

² For more information, see <http://www.outguess.org/detection.php> (last visited 5 June, 2002).

common stego algorithms, i.e., OutGuess 0.13b, JSteg, JPHide, Invisible Secrets and F5 (see Section 3.5). StegDetect can also expose added information in the header or at the end of JPEG-images. The applications are written by Niels Provos, who is a successful researcher in the area. He is also the author of the stego algorithm used by OutGuess, which is published in two releases (version 0.13b and 0.2). The latest version is very good and cannot be exposed by statistical methods or by visually attacks [Provost & Honeyman 2002]. At least not by any published method.³

In this chapter, StegDetect has been used to test the stego files' statistical characteristics.

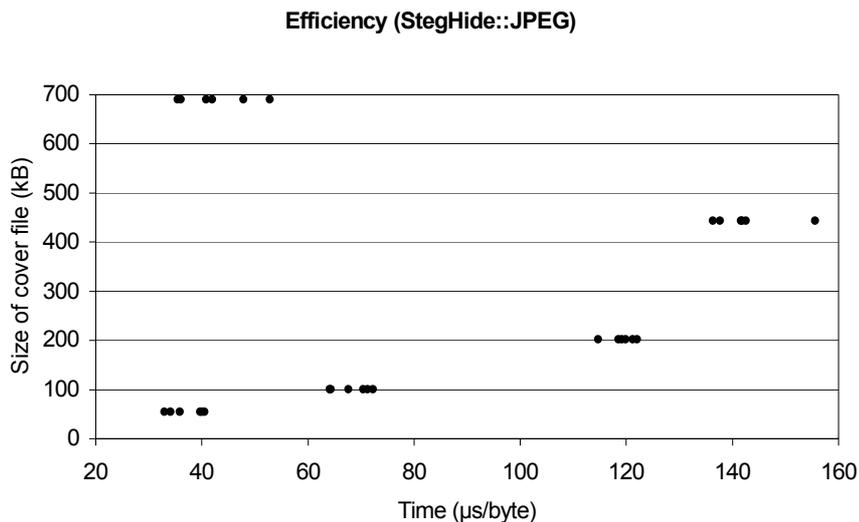
The visual tests have been performed by opening the stego files in Microsoft Photo Editor and zooming in 200% and 400%. Another obvious visual attribute is, e.g., to see if the stego file differs from the cover file in file size. If so, a stego file can be easily exposed when compared to the original file (cover file). The results are summarized under each algorithm's section.

An application similar to StegDetect has not been found for steganalysis of audio files, but research is going on about this⁴. The audiovisual effects are tested by listening for crackles and other strange noises compared to the cover file. This test method is included to give some kind of a rough hint of the strength of stego algorithms for audio files.

5.4 StegHide

StegHide is a software tool that supports hiding techniques for images and audio, i.e., JPEG, WAV and AU. This makes it the most flexible tool in the tests.

5.4.1 JPEG



³ Jessica Fridrich claims in an e-mail to us that her research team recently have found a method for breaking OutGuess 0.2. The e-mail is dated 7 June 2002.

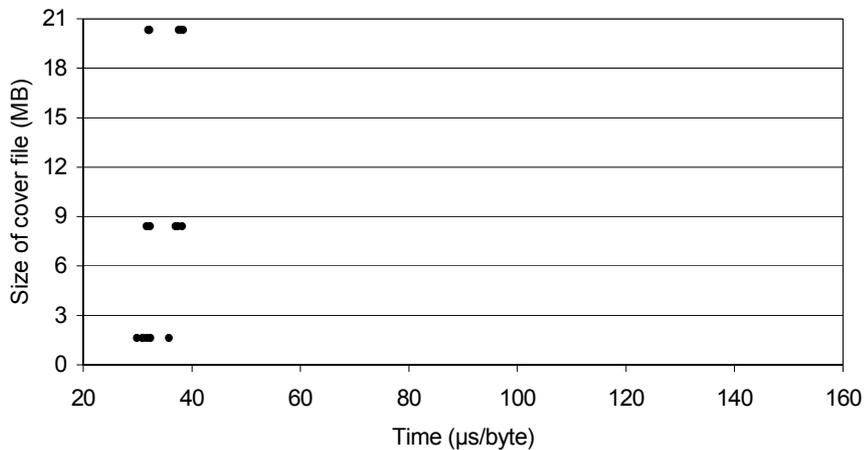
⁴ Stephen P. Mahoney states this in a report "*Audio Steganography and Steganalysis*", Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, 11-13 June 2002 at John Hopkins University.

This diagram shows an interesting divergence compared to an expected scatter plot and no obvious tendency can be drawn. The result can be explained by the differences in compression rate for the cover files. This algorithm is slower when a higher compression rate is used and this showed out to be unique for this algorithm. The ratio increased between the cover file and the stego file according to how much information that was hidden. The algorithm had no upper limit for how much information that could be hidden. The quality of the images suffered though.

| Average figures | |
|---------------------------|------------------------------------|
| Time for hiding | <i>81.9 μs/byte</i> |
| Size changed (stego file) | <i>+35%</i> |

5.4.2 WAV

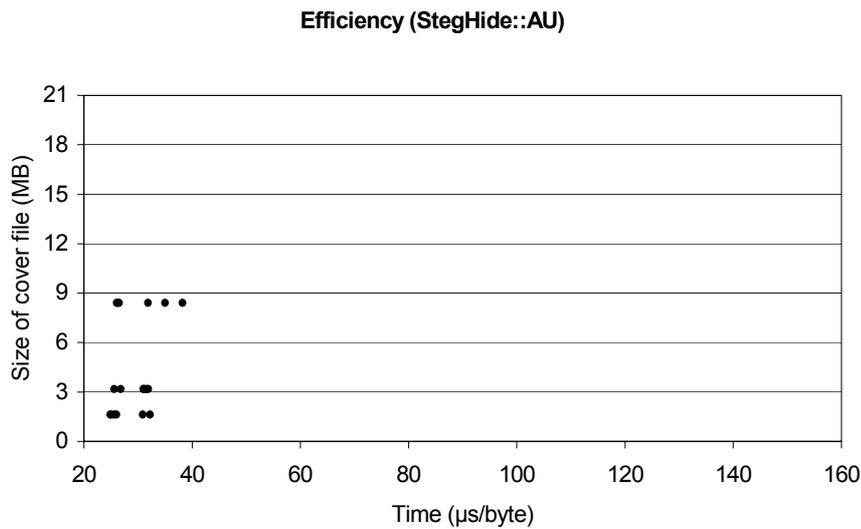
Efficiency (StegHide::WAV)



The scatter plot for each cover file contains well collected plots. This means that the algorithm seems to embed different sizes of information in a constant way. When making a comparison between WAV and JPEG, hiding information in audio is superior because of its speed and that the stego file does not change in size compared to the cover file.

| Average figures | |
|---------------------------|------------------------------------|
| Time for hiding | <i>33.9 μs/byte</i> |
| Size changed (stego file) | <i>None</i> |

5.4.3 AU



This algorithm is the only one in the test that can hide information in AU-files. The result has many similarities to the result for WAV-files. The average time for hiding one single byte was fastest for AU-files, which is interesting because of the small variations between the WAV and the AU files.

| Average figures | |
|---------------------------|---------------------|
| Time for hiding | <i>29.2 µs/byte</i> |
| Size changed (stego file) | <i>None</i> |

5.4.4 Detectability

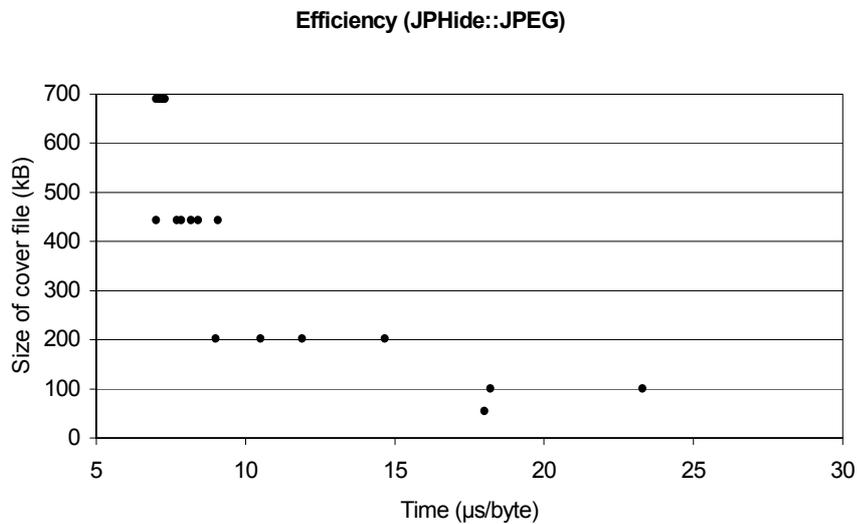
The steganalysis tool StegDetect was not able to find any suspicious characteristics during the analysis.

The visual detection resulted in obvious evidences for that something was hidden. Squares of 8x8 bits were easily found and increased as the size of the hidden information grew. This is typical for algorithms that hide information in the LSB [Fridrich & Goljan 2002]. Also, a larger fuzzy square was found in the upper left corner of each stego image. Evidence was harder to find when the cover file was rich of details and had a large file size, but still pretty obvious. The size for the stego file increased on average 35%, which is not good if the cover file is known.

No audiovisual oddity was noticed for the stego files in the WAV and AU formats.

5.5 JPHide

5.5.1 JPEG



This algorithm did not even try to hide information that was greater than 15% of the cover file and that is why some dots are missing for the three tests at the bottom of the diagram. The scatter plot was better collected for larger cover files and a reason to this is that the algorithm is dependent on the compression rate of the image. This test shows that the lower a compression rate is for the cover file, the faster the information is hidden.

| Average figures | |
|---------------------------|---------------------|
| Time for hiding | <i>10.4 µs/byte</i> |
| Size changed (stego file) | <i>-6%</i> |

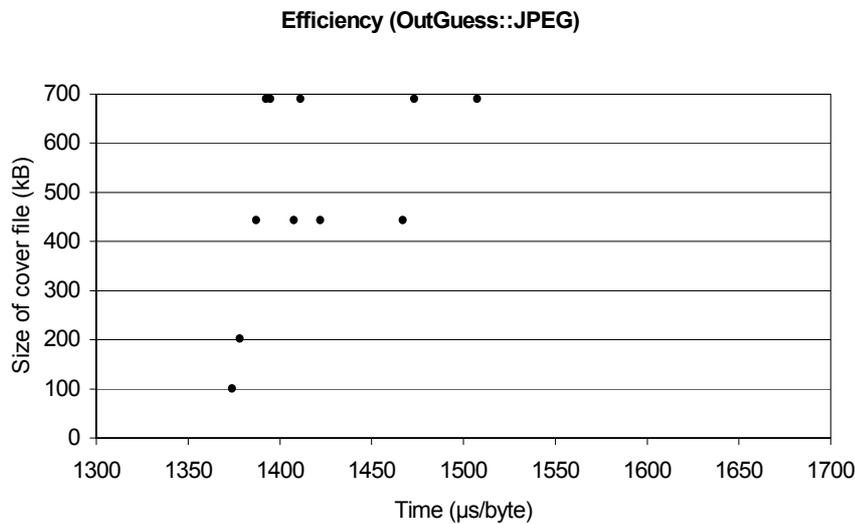
5.5.2 Detectability

The steganalysis tool StegDetect found exceptional characteristics for the stego file. In all probability, it could also state that JPHide had hid the information.

The visual detection was not as successful and resulted in no suspicion at all. The quality of the stego image was visually very good.

5.6 OutGuess 0.2

5.6.1 JPEG



OutGuess is the slowest hiding algorithm in the tests, but also considered as the best one for JPEG-files. The reason for being slow is that the algorithm includes advanced optimisation functions that are time-consuming. Just like JPHide, it does not even try to hide information files that are greater than 7% of the cover file. In other words, it is greedier than JPHide. Notice that none of the cover files could hide 50 kB of information (cp. Appendix A). To do that, a cover file of at least ~715 kB should be used. Hany Farid got a similar result in a test in which he only could hide a secret message in 219 of 500 images [Farid 2001].

The scatter plot for each cover file also shows large gaps between the dots. This can also be explained by the advanced hiding algorithm that tries to optimise the hiding of each byte, i.e., the more bytes to hide, the longer it will take for each byte because it is included in a larger set of bytes.

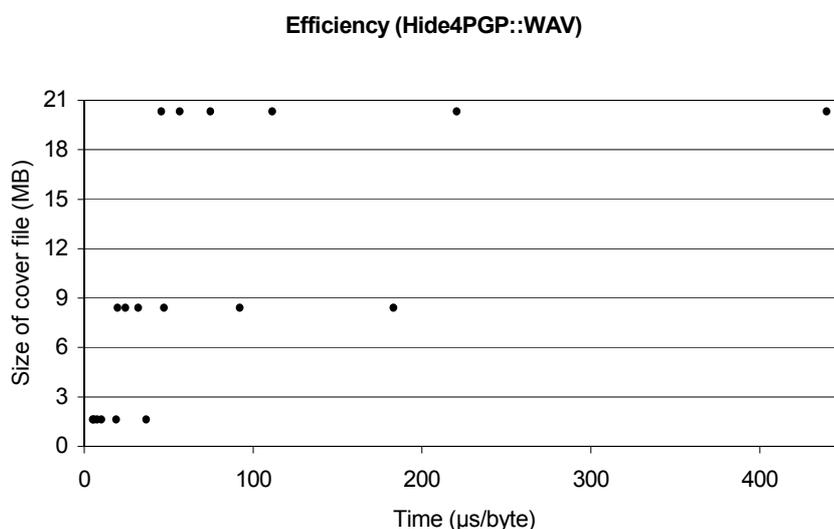
| Average figures | |
|---------------------------|----------------|
| Time for hiding | 1419.5 µs/byte |
| Size changed (stego file) | -5% |

5.6.2 Detectability

This algorithm is, as previously mentioned, very good. No statistical evidence was found. Nor any kind of obvious visual oddities could be observed. The lower quality compared to the cover file can be explained by the higher compressing rate used by the algorithm during transformation (a compression rate of 75% is used as default). This was only possible to observe for the cover file with the highest resolution.

5.7 Hide4PGP

5.7.1 WAV



The way Hide4PGP works differs compared to how the other algorithms work. In the scatter plot, the outer right dot represents the smallest information file in file size. The opposite holds for the other algorithms. In other words, the smaller the text to be hidden is in file size, the slower the algorithm works to hide each byte. A reason for this behavior is that the hiding of each byte is very fast, but the time to prepare the hiding is almost constant for all images of the same file size. This also results in that the gaps between the dots become smaller when the size of the hidden information increases. The larger a text to be hidden is, the larger amount of bytes there are to split the constant time over.

| Average figures | |
|---------------------------|--------------|
| Time for hiding | 79.5 µs/byte |
| Size changed (stego file) | None |

5.7.2 Detectability

As with the StegHide's algorithms that hide information in audio files, nothing strange could be heard.

5.8 Discussion

Even though these tests only include stego algorithms that execute under Linux, their way of working are representative for the set of algorithms on the market. According to Ross Anderson, there exist up to four generations of hiding algorithms [McCullagh 2001]. The number of open source algorithms published, often belongs to the first generation of hiding algorithms. It is those algorithms that are available to download from the Internet. In [McCullagh 2001] the author interviews Neil Johnson, whose research of developing steganalysis tools is sponsored by NSA. Neil Johnson says that there exists classified research for detecting hidden information in a second and a third generation of stego algorithms. These generations have increased characteristics for hiding information and making the chances of detection close to zero.

5.8.1 Recommendations

The test results show that OutGuess is the best choice for hiding information in JPEG files. If it is possible to choose audio as cover, this will increase the possibilities of avoiding detection, partially because there are no published statistical tools that can be used.

5.9 Summary

These tests were performed using static cover files and will act as base for next chapter's discussion. The results gave examples of how different algorithms work and their efficiency, capacity and performance in different media. All tests were iterated three times and gave similar outcome all of the times (only a deviation of 1-2% differed between the executions).

Only algorithms available as open source and executable under Linux were tested. The reasons for this were that it is easier to measure time (add timekeeping in the source code) and to see how the algorithms work when having access to the source code. All source code were written in C or C++.

The tested algorithms and the test results are summarized in the tables below. The first table lists the efficiency results for each algorithm and the second table lists how the file size changes of the stego file compared to the cover file (0% means that there was no difference in size between the files).

| <i>Time (μs/byte)</i> | JPEG | WAV | AU |
|--------------------------------------|--------|------|------|
| StegHide | 81.9 | 33.9 | 29.2 |
| JPHide | 10.4 | - | - |
| OutGuess 0.2 | 1419.5 | - | - |
| Hide4PGP | - | 79.5 | - |

| <i>Size (%)</i> | JPEG | WAV | AU |
|---------------------|------|-----|----|
| StegHide | +35 | 0 | 0 |
| JPHide | -6 | - | - |
| OutGuess 0.2 | -5 | - | - |
| Hide4PGP | - | 0 | - |

Next chapter describes the laboratory environment in which the possibilities and strength of hiding information in streaming media are evaluated and discussed.

6 EVALUATION OF TCP/IP HEADER AS COVER

6.1 Introduction

According to Section 2.2.2, the TCP/IP header can be considered as a type of streaming media. It was Craig Rowland that first introduced the approach of hiding information in the TCP/IP header. In a paper [Rowland 1996], he presents the approach together with different methods that can be used during the embedding of information. After his publication, no additional public research has been published.

Our intention is to evaluate the TCP/IP header as information carrier, by doing a series of experiments and measuring different attributes. The main purpose with these experiments is to answer two questions; is it practical possible to hide information in the TCP and IP headers? And, maybe even more interesting; how do protocol hiding measure up with the more well-known carriers, as images and audio files? The experiments will provide data, making it easier to draw valid and correct conclusions about these two questions. In those cases where no experiments can be carried out to test a particular part, a discussion will be held instead. This will, for example, be used with the detectability attribute.

Alas, Rowland's work alone cannot be considered to be of any greater practical use. Some flaws exist in his embedding process and at least one important issue has been left out of the discussions. Thus, before we can perform the experiments, these issues will be presented and proposals of solutions will be presented as well. These proposals are not in focus but only used as a way to create a practical useful approach, which hides information in the TCP and IP headers.

Since the TCP/IP approach will be compared in respect to other information carriers, we want the comparisons to be made as fair as possible. Therefore, a fully working prototype and proof of concept of hiding in TCP/IP has been implemented. This prototype is further explained and discussed in Section 6.5. Finally, the experiments and the measured results are presented and discussed.

6.2 Motivation

The motivation for evaluating the TCP/IP approach can be divided in three parts. First, the current research is heavily concentrated on images and audio files. This may result in unreliable security since revolutionary progress is likely to happen, decreasing the strength of the carrier. If this type of progress is made in the field of steganalysis, it will most likely not reach people, but agencies and other interests. While nobody knows about the progress, they think they are using a secure schema, which in fact is not totally true. Secondly, both images and audio files are based on static content. No respect is taken to the increasingly demand for new types of services becoming available, like web radio and web TV. Finally, there is an increasing demand for faster data transmissions. This may most likely result in that the average number of network packages sent over the Internet, in the next couple of years, will increase. With all these billions and billions network

packets passing through the cables every day, it would be interesting if they could be utilised to carry some hidden information.

6.3 Hiding information in the TCP/IP header

As previously mentioned in the introduction, Rowland's approach must be modified to be of any greater practical use. As we see it, there are two fundamental parts lacking in his approach. First, no discussion is held about how the receiving part knows when to interpret the received TCP/IP header as a header with embedded data or not. Secondly, the proposed way to embed information is inefficient in respect of capacity and time. Also, it is a requirement that no packet fragmentation will occur.

Our solution proposal to the first problem is to introduce the use of a stego connection. This type of connection involves two tasks; connection establishing and connection termination. When established, the connection acts as a logical covert connection between sender and receiver. Once established, a message can be hidden in the header of TCP/IP network packets. Upon completion of sending a message, the connection is terminated. A more technical and detailed description is presented in the next section.

The solution to the second problem is straightforward and quite obvious. To solve it, no information is hidden in fields related to packet fragmentation, as it will make it harder to rearrange packets later on. More about our embedding method can be found in Section 6.3.2.

6.3.1 Connection establishing phase

Within this phase, the server sends a four bytes long start signal to the client. The signal is hidden in the same way as a message is hidden, which is described in Section 6.3.2. The purpose of sending a start signal is to make the client aware of that a transfer of hidden data is requested from the server. Thus, it ensures that the client only interpret data from the TCP/IP header as a message from the sender when it actually is.

Each packet arriving to the client is examined for the presence of the start signal. If found, an acknowledgement signal is sent back to the sender. This signal ensures that the server does not unnecessarily embed and send information to the client. As the sender receives the acknowledgement signal, a stego connection is established between sender and receiver. When a network transfer (TCP/IP) takes place between sender and receiver, hidden information is embedded and sent together with the transferred data.

6.3.2 Embedding phase

After the establishing of a stego connection, the *embedding phase* is next. It is here the actual message is embedded in the IP and TCP headers (see shaded areas in Figure 6 and Figure 7). The fields used for the embedding process are eight bits of the TOS (Type of Service) field in the IP header and eight bits of the sixteen bits long destination port of the TCP header. Thus, the capacity is sixteen hidden bits in each TCP packet. The actual embedding process involves copying the message bits to hide into the packet to be sent. The extraction process is like the

embedding process, but with the distinction that it copies the message from the incoming network packet and inserts it into a receive buffer.

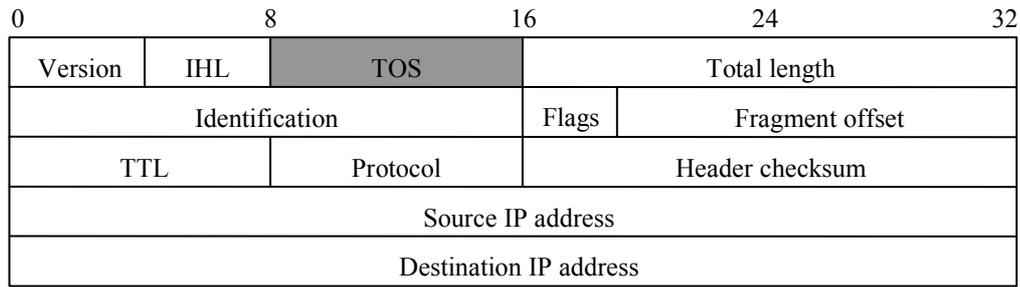


Figure 6: The IP header.

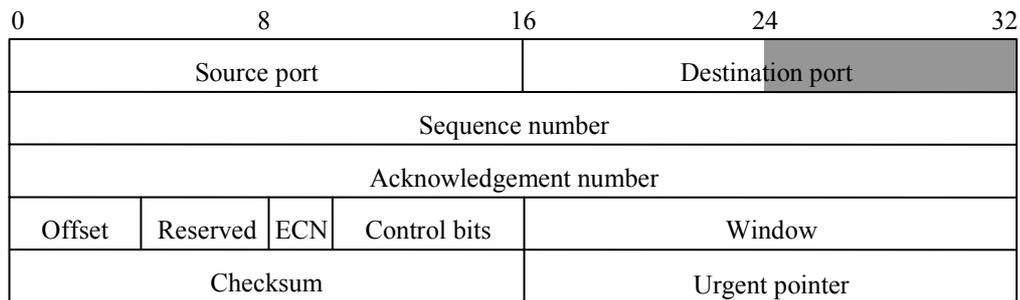


Figure 7: The TCP header.

6.3.3 Connection termination phase

As soon as the message transfer is completed, the stego connection needs to be terminated. This is accomplished by sending an end signal. As with the start signal, the end signal is four bytes long and hidden in the same way as the message. When the client receives the end signal, the stego connection is terminated and has to be established once again, in order to be used for data transfer.

6.4 Experimental environment

The actual experiments were conducted with two computers, a client and a server. Both client and server used a 450 MHz Intel Pentium II processor, 128 MB RAM memory and a 10/100 Mbit/s NIC (Network Interface Controller). On both computers, Red Hat Linux 7.0 was installed. A modified version of the kernel (version 2.2.16) was installed as well. The modifications in the kernel were done in the networking code, to invoke the prototype (described in the following section). In order to connect the computers, a 100 Mbit/s network hub was used during all experiments. No other computers were connected to the hub during the execution of any of the experiments to avoid disturbing network traffic. Both computers were in single user mode state and a minimal number of processes ran in order to minimise interference from other processes during the tests. For a physical view of the experimental environment, see Figure 8.

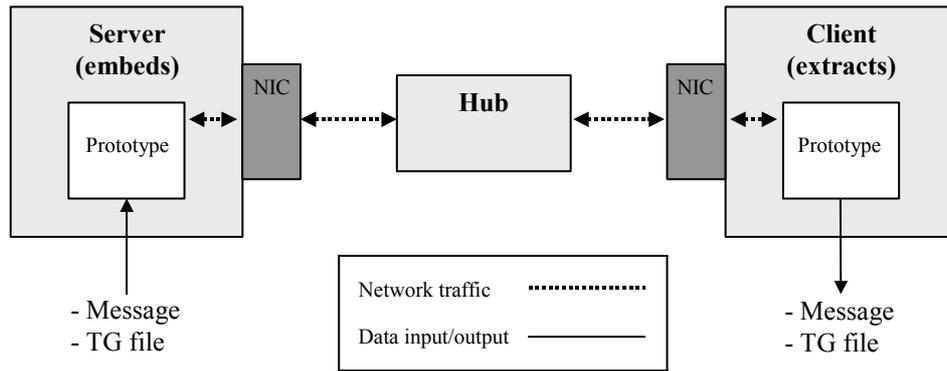


Figure 8: A physical view of the experimental environment.

6.5 The prototype

The prototype is installed and looks the same on both the client- and the server-sides (see Figure 9). The server part is only concerned with embedding a message into outgoing network packets and the client of extracting it from incoming network packets. In both client and server, the networking code in Linux's kernel has been modified to redirect all IP packets to the stego module. The calling of the stego module is done differently in the server and the client. In the server, the call is done just before the checksum is calculated. By doing this, no redundant checksum calculation is performed. In the client, the call is done within the IP code.

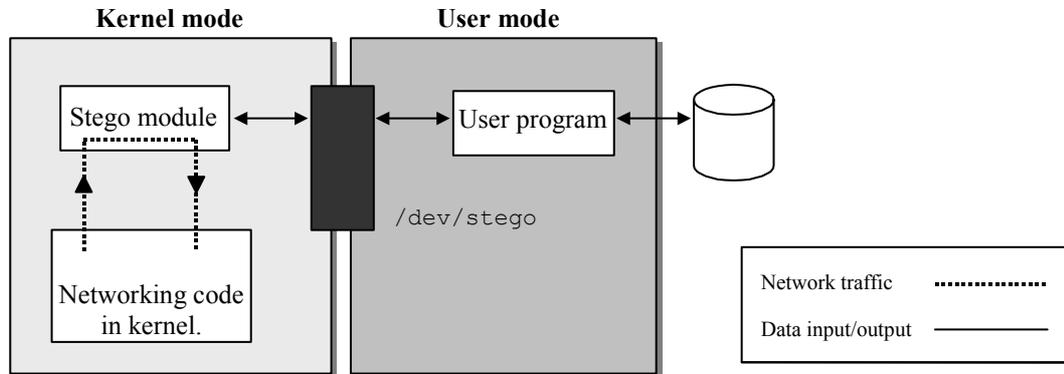


Figure 9: The prototype.

The stego module is implemented as a Linux loadable module, i.e., it runs in kernel mode and can be added or removed to/from the kernel in run-time. A reason for using a module is that it is practically, since the kernel does not have to be re-compiled, installed and booted after every modification of the module.

In the stego module, all network packets' source- and destination addresses and port numbers are examined. If they match those of the client's and the packet is outgoing, the message is read from a buffer and added to the packet. In this case, the stego module is said to operate in server mode, i.e., it embeds the message in the outgoing packets. On the other side, if the module operates in client mode, the message is extracted from incoming packets. A receive buffer is then used to hold the extracted message.

A character device, i.e., `/dev/stego`, handles the exchange of data between kernel and user mode. The device exposes a standardised interface to user processes, e.g., write, read, open and close. In our implementation, a user program reads the message to hide from a file system and sends it through `/dev/stego` into the stego module's buffer.

6.6 The experiments

All experiments began by loading the stego module at the server and the client. The stego module at the server was then loaded with a 5,000 – 50,000 bytes large text file. This file contained the message to be hidden. A 100 MB large file, called TG file, was used to generate network traffic (something to hide in).

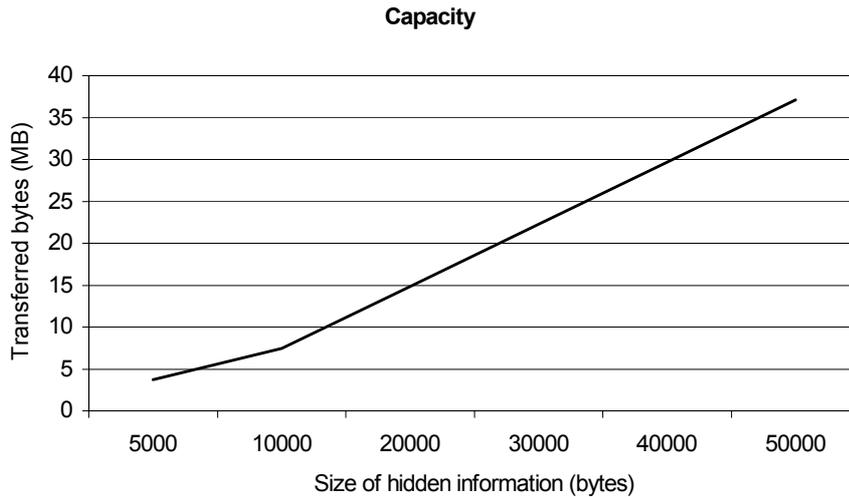
Next, an FTP connection was opened between the server and the client. The TG file was transferred to the client. As the FTP network packets left the server, they were redirected in to the stego module. There they were embedded with data from the message before they continued their way to the client. When the client received one of these network packets, it redirected the packet to its stego module, which extracted the data and stored it in a buffer. The FTP process, at the client's side, finally received the network packet.

Three different types of experiments have been performed, i.e., capacity, performance and consistency. The capacity experiment answers the question of how many TG bytes that are required to hide one byte of the message. In the performance experiment, the time for embedding is measured, i.e., the time required for the algorithm to perform the embedding. This way of measuring the performance is the same as used for the other investigated carriers and can therefore be used when comparing these approaches. The last experiment is concerned with consistency and is not involved in the actual evaluation of the TCP/IP approach. Instead, it reveals the presence of corrupted data, as a result of the stego process. Its purpose is to ensure that none of the transferred data packets have been corrupted. Both the message and the TG file were examined in this experiment and it was conducted after the capacity and performance experiments.

The three types of experiments are all described in more detail in the following sections below. All experiments were conducted five times.

6.6.1 Capacity experiment

This experiment was conducted by performing an FTP transfer from the server to the client as described in Section 6.6. The message file was continuously embedded into the headers. Inside the stego modules, a counter was used to measure the number of bytes embedded. Finally, a consistency test on the transferred data was carried out to ensure its correctness.

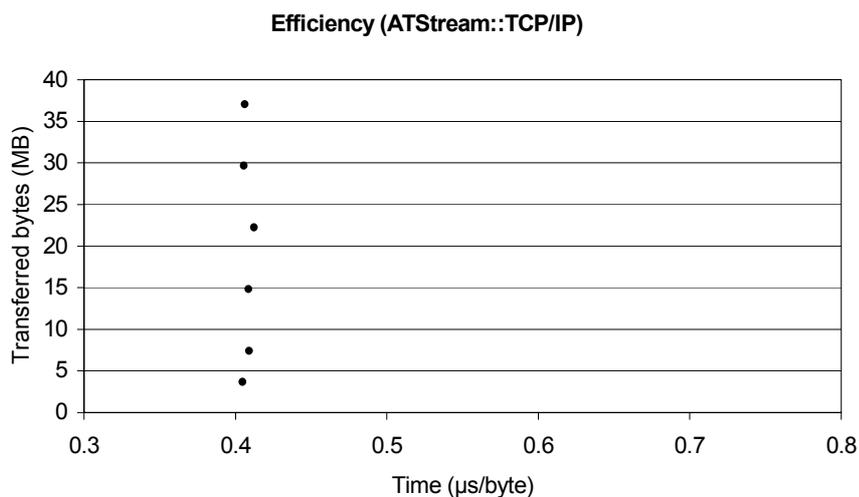


As seen in the chart above, the average of the five tests is almost linear. The main reason for this is that the capacity is directly related to the number of network packets transferred. Since the length of each packet may vary from time to time, the number of packets varies and thus the number of transferred TG bytes.

$$\text{The capacity quota by experiment} = \left(\frac{\text{TotalBytesHidden}}{\text{TotalTGBytesTransferred}} \right) = 0.13\%.$$

6.6.2 Performance experiment

Since the embedding is performed in kernel mode and the time required is short, a software timer construct available in the kernel was used. The timer is started when the stego module has been invoked from the kernel's network code and stopped right after it returns.



All results show better performance values compared to those for images and audio files in Chapter 5. Compared to images, embedding in the TCP/IP header is at least 25 times faster. These results were expected since the embedding and

extraction process is simple and does not involve any complex tasks like optimisation, etc.

6.6.3 Consistency experiment

The experiment was conducted as described at the beginning of Section 6.6. After transferring the stego file including a message to the client, the received message was compared to the original message at the server. All comparisons were conducted with the Linux `diff` utility command. No inconsistency could be found, neither after the capacity experiment nor after the performance experiment.

6.7 Detectability discussion

At first glance, the strength of the security for TCP/IP header as information carrier seems weak. Since the TOS field in the IP header most often is not used at all, a check is what is needed to suspect and detect hidden information. It is a bit trickier with the byte hidden in the destination port number, as a simple check is not sufficient. Instead, another detection mechanism is required. An idea would be to investigate the incoming packages, looking more closely on the sequence and acknowledgement number in the TCP header. By looking at these, the detection mechanism should be able to identify that a packet belongs to a certain transfer, without looking at the port number at all. Finally, a comparison of the expected destination port number and the actual port number is needed before the package can be suspected to carry hidden information.

The strength of the security for an information carrier must not be poor only because detection mechanisms exist. It is still of major importance that this type of threat is successfully investigated. If the threat is seen as a vulnerability, countermeasures should be developed and regularly maintained. Another issue involves how to handle a packet that is suspected to carry hidden information. What if the TOS field actually is used for a particular transfer containing no hidden information? What would be the correct way to handle this? If the packet is dropped, a legal transfer has been disturbed. Maybe the use of the TOS field should be forbidden in network traffic within networks where the possibility of finding hidden information is high. These are some of the issues that must be dealt with from an information security perspective.

6.8 Summary

Compared to images and audio files, the time of embedding was clearly impressive with the TCP/IP header approach. The embedding process was at least 25 times faster compared to the tested algorithms for embedding in images. An expected limitation was the limited capacity of the approach. The capacity experiment showed that an average of only 0.13% of the total TG file could be utilised. This cannot be compared to images and audio files in which about 7 – 15% can be utilised (cp. Chapter 5). A more correct comparison would be 5% since each header of 40 bytes can hide two bytes. With a capacity of 0.13%, a TG file of approximately 769 MB is needed to transfer a message of 1 MB.

Even though non-complex detection mechanism can be developed, this alone does not solve the potential vulnerability of information hiding in the TCP/IP header. Countermeasures require that a policy is established and followed. It might, for

example, involve decreasing the functionality in the networks, e.g., by restricting the use of certain header fields in network communication.

However, it is worth mentioning that the TCP/IP approach presented in this chapter is not optimal. For example, by adjusting the hidden bytes per packet or other parameters, e.g., the maximum length for a network packet, the capacity can be greatly improved. If the packet length decreased to maximum 100 bytes, we could increase the capacity to 2%. But, a short packet length may raise suspicion and is therefore not good from a steganographical point of view.

Anti-detection mechanism might also be added to make it more difficult to detect. More about this and other future works are presented in the following chapter together with general conclusions about our research.

7 CONCLUSIONS

In this chapter, the work of the thesis is summarized and recommendations that discuss what type of cover to use in different situations are given. At the end, examples of future work are mentioned.

7.1 General

The aim and objectives with this thesis can all somehow be associated and work as base for the main question, i.e., how steganographical methods can be used to hide information in streaming media. The literature research part consists of an investigation about how information can be hidden and detected in static media, e.g., an image file or an audio file. Also, a brief background to steganography and its possibilities, the threats and vulnerabilities it may bring, have been studied and documented.

A laboratory environment was set up and implemented for efficiency tests and to practically study the possibilities for hiding information in the TCP/IP header, which we by definition see as streaming media. The focus was set on hiding in ordinary network traffic using fields in the TCP/IP header. Fundamentally it is the same thing when hiding in UDP headers, which is preferred when streaming live video for example.

The efficiency tests for hiding in a TCP/IP header resulted in very good results concerning performance, but less good in respect of capacity. The performance was around 25 times faster (0.41 μ s/byte) compared to the fastest stego algorithms tested for static media. In comparison with static media, the capacity would be 5% since each header of 40 bytes can hide two bytes. Our algorithm for hiding in the TCP/IP header, called *ATStream*, is limited to hide in the TCP/IP header. Despite a capacity of 5%, our experiment shows that huge amounts of data (TG bytes) have to be sent.

ATStream is a fully working algorithm for its purpose. No other implemented algorithms for hiding information in streaming media have been found within the open source community. This is our way to illuminate the strength in such algorithms.

The whole idea with steganography is to avoid suspicion. If revealed, steganography in itself does not offer enough protection. The possibility of using a flow of ordinary information as cover for information increases the chance to succeed to avoid suspicion. To detect and protect against a use of steganography in ordinary files and streaming media may be difficult and time-consuming. The *state-of-the-art* research is focused on detecting statistically divergences in characteristics in stego images.

7.2 Recommendations

All algorithms tested that hide information in images increase the file size and often result in visual defects compared to the original image file. Most of the algorithms on the market use images as cover, but the variations in strength

among those differ a lot. By using statistical analysis, hidden information can be revealed at a large percentage of the stego files. There exist a couple of tools for detecting hidden information in images. Even if they cannot always prove which stego algorithm that has been used, a general statistical analysis is often enough and thereby the purpose of using steganography is destroyed.

Audio as cover is not as usual as images. None of the algorithms tested leave any obvious signs, neither in quality nor file size. There is also a lack of detection tools for finding hidden information in audio, which is good from a steganographical point of view.

To hide in streaming media means great opportunities and we believe the techniques for hiding will be adjusted in a near future.

Our recommendations for what cover to use depends on the purpose of hiding and the size of the information to hide. Images are very common on homepages on the Internet and are therefore a good alternative for hiding smaller messages and make them public available. WAV files have a good capacity, but are less common on the Internet and may therefore be suspected. MP3 files are more used, but the only public algorithm for hiding in these kinds of files (MP3Stego) requires a WAV file as input, which means a more time-consuming hiding process. We find streaming media as the most interesting one because it can take advantage of an ordinary stream of data packets for example. Even if the capacity is lesser than for static media, it is faster. Today there does not exist any known hiding algorithm for streaming media, but we believe it is for this the future heads. The use of streaming media is growing and offers great opportunities to hide information, but the hiding techniques have to be refined. Our research shows up some of its possibilities.

7.3 Future work

We find the area of steganography for binary media very interesting and during the work on this thesis some future work proposals have been found. For example, it would be interesting to develop our algorithm further to make it more efficient in the hiding process and better at avoiding detection.

Another interesting idea had been to investigate how a changed packet length affects capacity, ease of detection, performance, etc.

An investigation of how semantics can be applied to the time between packets would also be of interest.

Even if the challenge of finding and constructing an algorithm for hiding may be tempting, we believe a focus on detecting hidden information should be prioritised. Steganography is sometime considered as one of the greatest threats against democracy. It is of major importance to be able to reveal the existence of hidden information for any kind of cover. If no methods exist for decoding a suspected cover, it can always be totally destroyed and thereby lose its purpose. This way is an emergency solution and should not be accepted in a longer time perspective. It is more interesting to know what information that is hidden if a suspicious kind of media is found.

8 REFERENCES

- [Anderson & Petitcolas 1998] Anderson, R., Petitcolas, F., “*On The Limits of Steganography*”, IEEE Journal of Selected Areas in Communications, 16(4):474-481, 1998.
- [Anderson 1996] Anderson, R., “*Information Hiding: First International Workshop*”, Lecture Notes in Computer Science, vol. 1174, Springer-Verlag, 1996.
- [Bender et al. 1996] Bender, W., Gruhl, D., Morimoto, N., Lu, A., “*Techniques for data hiding*”, IBM Systems Journal, vol. 35, NOS 3&4, 1996.
- [CNN 2001] Sieberg, D., “*Bin Laden exploits technology to suit his needs*”, CNN.com, 21 Sep., 2001.
- [Farid 2001] Farid, H., “*Detecting steganographic messages in digital images*”, Technical report, Dartmouth College, Hanover, 2001.
- [Fridrich & Goljan 2002] Fridrich, J., Goljan, M., “*Practical Steganalysis of Digital Images – State of the Art*”, In SPIE Photonics West, Electronic Imaging, San Jose, CA, 2002.
- [Fridrich 1999] Fridrich, J., “*A new Steganographic method for palette-based images*”, IS&T PICS Conference, Savannah, Georgia, April 25-28, 1999.
- [Fridrich et al. 2001] Fridrich, J., Goljan, M., Du, R., “*Steganalysis Based on JPEG Compatibility*”, Special session on Theoretical and Practical Issues in Digital Watermarking and Data Hiding, SPIE Multimedia Systems and Applications IV, Denver, CO, August 20-24, 2001.
- [Johnson & Jajodia 1998] Johnson, N., Jajodia, S., “*Exploring steganography: seeing the unseen*”, IEEE Computer, vol. 31, no. 2, pp. 26-34, February 1998.
- [Kahn 1996] Kahn, D., “*The codebreakers*”, Simon & Schuster Inc., 1996.
- [Korjik & Morales-Lunar 2001] Korjik, V., Morales-Luna, G., “*Information Hiding through Noisy Channels*”, Lecture Notes in Computer Science, vol. 2137, pp. 42-50, Springer-Verlag, 2001.
- [Manamalkav 2002] Manamalkav, M., “*Audio file Steganography*”, available from Internet <www.cise.ufl.edu/~smanamal/steganography.htm> (9 April 2002), 2002.
- [McCullagh 2001] McCullagh, D., “*Secret Messages Come in .Wavs*”, newspaper article published in Wired News, February, 2001.
- [Petitcolas 2002] Petitcolas, F., “*mp3stego*”, available from Internet <www.cl.cam.ac.uk/~fapp2/steganography/mp3stego/> (9 April 2002), 2002.
- [Pfitzmann & Westfeld 1999] Westfeld, A., Pfitzmann, A., “*Attacks on Steganographic Systems*”, In proceedings of Information Hiding – Third International Workshop, Springer Verlag, pp. 61-76, September 1999.
- [Provos & Honeyman 2002] Provos, N., Honeyman, P., “*Detecting Steganographic Content on the Internet*”, ISOC NDSS’02, California, 2002.

- [Provos 2001]** Provos, N., *“Defending against statistical steganalysis”*, 10th USENIX Security Symposium. Washington, DC, August 2001.
- [RFC 1889]** RTP, available from Internet <www.ietf.org/rfc/rfc1889.txt> (3 June 2002), 1996.
- [RFC 2326]** RTSP, available from Internet <www.ietf.org/rfc/rfc2326.txt> (3 June 2002), 1998.
- [Rowland 1996]** Rowland, C., *“Covert Channels in the TCP/IP Protocol Suite”*, available from Internet <www.firstmonday.dk> (11 June 2002), 1996.
- [SICTC 2002]** The Swedish ICT Commission, *“General guide to a future-proof IT infrastructure Report 37/2001”*, available from Internet <www.itkommissionen.se> (3 June 2002), 2001.
- [Singh 1999]** Singh, S., *“The code book”*, Fourth Estate, 1999.
- [UP 2001]** Hatfield, J., *“The Crazy Gang”*, UP-Magazine, Issue 2, 2001.

APPENDIX A

This appendix includes figures from the tests performed in the thesis. Each test table represents average figures from tests iterated three times.

StegHide :: JPEG

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (µs) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------|
| 55515 | 72263 | 5000 | 179 | 9.0% | 1.30 | 35.80 |
| 55515 | 82048 | 10000 | 405 | 18.0% | 1.48 | 40.50 |
| 55515 | 94821 | 20000 | 799 | 36.0% | 1.71 | 39.95 |
| 55515 | 102235 | 30000 | 1191 | 54.0% | 1.84 | 39.70 |
| 55515 | 106197 | 40000 | 1361 | 72.1% | 1.91 | 34.03 |
| 55515 | 116006 | 50000 | 1645 | 90.1% | 2.09 | 32.90 |
| 100465 | 114690 | 5000 | 338 | 5.0% | 1.14 | 67.60 |
| 100465 | 123256 | 10000 | 723 | 10.0% | 1.23 | 72.30 |
| 100465 | 133587 | 20000 | 1424 | 19.9% | 1.33 | 71.20 |
| 100465 | 139181 | 30000 | 2112 | 29.9% | 1.39 | 70.40 |
| 100465 | 141335 | 40000 | 2573 | 39.8% | 1.41 | 64.33 |
| 100465 | 150582 | 50000 | 3204 | 49.8% | 1.50 | 64.08 |
| 202260 | 224226 | 5000 | 606 | 2.5% | 1.11 | 121.20 |
| 202260 | 238913 | 10000 | 1199 | 4.9% | 1.18 | 119.90 |
| 202260 | 261064 | 20000 | 2382 | 9.9% | 1.29 | 119.10 |
| 202260 | 279366 | 30000 | 3555 | 14.8% | 1.38 | 118.50 |
| 202260 | 293706 | 40000 | 4587 | 19.8% | 1.45 | 114.68 |
| 202260 | 310794 | 50000 | 6100 | 24.7% | 1.54 | 122.00 |
| 443217 | 472108 | 5000 | 688 | 1.1% | 1.07 | 137.60 |
| 443217 | 495133 | 10000 | 1425 | 2.3% | 1.12 | 142.50 |
| 443217 | 534247 | 20000 | 2726 | 4.5% | 1.21 | 136.30 |
| 443217 | 566195 | 30000 | 4255 | 6.8% | 1.28 | 141.83 |
| 443217 | 594551 | 40000 | 5663 | 9.0% | 1.34 | 141.58 |
| 443217 | 619169 | 50000 | 7780 | 11.3% | 1.40 | 155.60 |
| 690597 | 717250 | 5000 | 180 | 0.7% | 1.04 | 36.00 |
| 690597 | 742416 | 10000 | 419 | 1.4% | 1.08 | 41.90 |
| 690597 | 784639 | 20000 | 709 | 2.9% | 1.14 | 35.45 |
| 690597 | 819118 | 30000 | 1584 | 4.3% | 1.19 | 52.80 |
| 690597 | 849412 | 40000 | 1632 | 5.8% | 1.23 | 40.80 |
| 690597 | 878193 | 50000 | 2392 | 7.2% | 1.27 | 47.84 |

StegHide :: WAV

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (µs) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------|
| 1634348 | 1634348 | 5000 | 162 | 0.3% | 1.00 | 32.40 |
| 1634348 | 1634348 | 10000 | 317 | 0.6% | 1.00 | 31.70 |
| 1634348 | 1634348 | 20000 | 618 | 1.2% | 1.00 | 30.90 |
| 1634348 | 1634348 | 30000 | 930 | 1.8% | 1.00 | 31.00 |
| 1634348 | 1634348 | 40000 | 1432 | 2.4% | 1.00 | 35.80 |
| 1634348 | 1634348 | 50000 | 1495 | 3.1% | 1.00 | 29.90 |
| 8423468 | 8423468 | 5000 | 191 | 0.1% | 1.00 | 38.20 |
| 8423468 | 8423468 | 10000 | 323 | 0.1% | 1.00 | 32.30 |
| 8423468 | 8423468 | 20000 | 640 | 0.2% | 1.00 | 32.00 |
| 8423468 | 8423468 | 30000 | 1123 | 0.4% | 1.00 | 37.43 |
| 8423468 | 8423468 | 40000 | 1265 | 0.5% | 1.00 | 31.63 |
| 8423468 | 8423468 | 50000 | 1851 | 0.6% | 1.00 | 37.02 |
| 20330540 | 20330540 | 5000 | 192 | 0.0% | 1.00 | 38.40 |
| 20330540 | 20330540 | 10000 | 380 | 0.0% | 1.00 | 38.00 |

| | | | | | | |
|----------|----------|-------|------|------|------|-------|
| 20330540 | 20330540 | 20000 | 644 | 0.1% | 1.00 | 32.20 |
| 20330540 | 20330540 | 30000 | 962 | 0.1% | 1.00 | 32.07 |
| 20330540 | 20330540 | 40000 | 1504 | 0.2% | 1.00 | 37.60 |
| 20330540 | 20330540 | 50000 | 1597 | 0.2% | 1.00 | 31.94 |

StegHide :: AU

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (μ s) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------------|
| 1634332 | 1634332 | 5000 | 161 | 0.3% | 1.00 | 32.20 |
| 1634332 | 1634332 | 10000 | 260 | 0.6% | 1.00 | 26.00 |
| 1634332 | 1634332 | 20000 | 512 | 1.2% | 1.00 | 25.60 |
| 1634332 | 1634332 | 30000 | 926 | 1.8% | 1.00 | 30.87 |
| 1634332 | 1634332 | 40000 | 997 | 2.4% | 1.00 | 24.93 |
| 1634332 | 1634332 | 50000 | 1245 | 3.1% | 1.00 | 24.90 |
| 3186988 | 3186988 | 5000 | 134 | 0.2% | 1.00 | 26.80 |
| 3186988 | 3186988 | 10000 | 319 | 0.3% | 1.00 | 31.90 |
| 3186988 | 3186988 | 20000 | 631 | 0.6% | 1.00 | 31.55 |
| 3186988 | 3186988 | 30000 | 767 | 0.9% | 1.00 | 25.57 |
| 3186988 | 3186988 | 40000 | 1244 | 1.3% | 1.00 | 31.10 |
| 3186988 | 3186988 | 50000 | 1549 | 1.6% | 1.00 | 30.98 |
| 8423452 | 8423452 | 5000 | 191 | 0.1% | 1.00 | 38.20 |
| 8423452 | 8423452 | 10000 | 265 | 0.1% | 1.00 | 26.50 |
| 8423452 | 8423452 | 20000 | 637 | 0.2% | 1.00 | 31.85 |
| 8423452 | 8423452 | 30000 | 782 | 0.4% | 1.00 | 26.07 |
| 8423452 | 8423452 | 40000 | 1400 | 0.5% | 1.00 | 35.00 |
| 8423452 | 8423452 | 50000 | 1308 | 0.6% | 1.00 | 26.16 |

JPEG::Average time 81.9 μ s/byte
WAV ::Average time 33.9 μ s/byte
AU ::Average time 29.2 μ s/byte

JPHide :: JPEG

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (μ s) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------------|
| 55515 | 57999 | 5000 | 90 | 9.0% | 1.04 | 18.00 |
| 55515 | | 10000 | | 18.0% | 0.00 | 0.00 |
| 55515 | | 20000 | | 36.0% | 0.00 | 0.00 |
| 55515 | | 30000 | | 54.0% | 0.00 | 0.00 |
| 55515 | | 40000 | | 72.1% | 0.00 | 0.00 |
| 55515 | | 50000 | | 90.1% | 0.00 | 0.00 |
| 100465 | 77671 | 5000 | 91 | 5.0% | 0.77 | 18.20 |
| 100465 | 80409 | 10000 | 233 | 10.0% | 0.80 | 23.30 |
| 100465 | | 20000 | | 19.9% | 0.00 | 0.00 |
| 100465 | | 30000 | | 29.9% | 0.00 | 0.00 |
| 100465 | | 40000 | | 39.8% | 0.00 | 0.00 |
| 100465 | | 50000 | | 49.8% | 0.00 | 0.00 |
| 202260 | 178952 | 5000 | 45 | 2.5% | 0.88 | 9.00 |
| 202260 | 179247 | 10000 | 105 | 4.9% | 0.89 | 10.50 |
| 202260 | 181717 | 20000 | 238 | 9.9% | 0.90 | 11.90 |
| 202260 | 190362 | 30000 | 440 | 14.8% | 0.94 | 14.67 |
| 202260 | | 40000 | | 19.8% | 0.00 | 0.00 |
| 202260 | | 50000 | | 24.7% | 0.00 | 0.00 |
| 443217 | 419800 | 5000 | 35 | 1.1% | 0.95 | 7.00 |
| 443217 | 420237 | 10000 | 84 | 2.3% | 0.95 | 8.40 |
| 443217 | 420310 | 20000 | 157 | 4.5% | 0.95 | 7.85 |
| 443217 | 424308 | 30000 | 231 | 6.8% | 0.96 | 7.70 |
| 443217 | 429162 | 40000 | 327 | 9.0% | 0.97 | 8.18 |

| | | | | | | |
|--------|--------|-------|-----|-------|------|------|
| 443217 | 431252 | 50000 | 453 | 11.3% | 0.97 | 9.06 |
| 690597 | 670285 | 5000 | 35 | 0.7% | 0.97 | 7.00 |
| 690597 | 670604 | 10000 | 71 | 1.4% | 0.97 | 7.10 |
| 690597 | 670905 | 20000 | 143 | 2.9% | 0.97 | 7.15 |
| 690597 | 671385 | 30000 | 216 | 4.3% | 0.97 | 7.20 |
| 690597 | 672852 | 40000 | 292 | 5.8% | 0.97 | 7.30 |
| 690597 | 672976 | 50000 | 364 | 7.2% | 0.97 | 7.28 |

JPEG::Average time 10.4 μ s/byte

OutGuess 0.2 :: JPEG

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (μ s) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------------|
| 55515 | | 5000 | | 9.0% | 0.00 | 0.00 |
| 55515 | | 10000 | | 18.0% | 0.00 | 0.00 |
| 55515 | | 20000 | | 36.0% | 0.00 | 0.00 |
| 55515 | | 30000 | | 54.0% | 0.00 | 0.00 |
| 55515 | | 40000 | | 72.1% | 0.00 | 0.00 |
| 55515 | | 50000 | | 90.1% | 0.00 | 0.00 |
| 100465 | 78475 | 5000 | 6870 | 5.0% | 0.78 | 1374.00 |
| 100465 | | 10000 | | 10.0% | 0.00 | 0.00 |
| 100465 | | 20000 | | 19.9% | 0.00 | 0.00 |
| 100465 | | 30000 | | 29.9% | 0.00 | 0.00 |
| 100465 | | 40000 | | 39.8% | 0.00 | 0.00 |
| 100465 | | 50000 | | 49.8% | 0.00 | 0.00 |
| 202260 | 123313 | 5000 | 6890 | 2.5% | 0.61 | 1378.00 |
| 202260 | | 10000 | | 4.9% | 0.00 | 0.00 |
| 202260 | | 20000 | | 9.9% | 0.00 | 0.00 |
| 202260 | | 30000 | | 14.8% | 0.00 | 0.00 |
| 202260 | | 40000 | | 19.8% | 0.00 | 0.00 |
| 202260 | | 50000 | | 24.7% | 0.00 | 0.00 |
| 443217 | 517307 | 5000 | 7335 | 1.1% | 1.17 | 1467.00 |
| 443217 | 517414 | 10000 | 14219 | 2.3% | 1.17 | 1421.90 |
| 443217 | 518102 | 20000 | 28153 | 4.5% | 1.17 | 1407.65 |
| 443217 | 518811 | 30000 | 41610 | 6.8% | 1.17 | 1387.00 |
| 443217 | | 40000 | | 9.0% | 0.00 | 0.00 |
| 443217 | | 50000 | | 11.3% | 0.00 | 0.00 |
| 690597 | 598405 | 5000 | 7537 | 0.7% | 0.87 | 1507.40 |
| 690597 | 598967 | 10000 | 14732 | 1.4% | 0.87 | 1473.20 |
| 690597 | 600142 | 20000 | 28224 | 2.9% | 0.87 | 1411.20 |
| 690597 | 601417 | 30000 | 41841 | 4.3% | 0.87 | 1394.70 |
| 690597 | 602527 | 40000 | 55700 | 5.8% | 0.87 | 1392.50 |
| 690597 | | 50000 | | 7.2% | 0.00 | 0.00 |

JPEG::Average time 1419.5 μ s/byte

Hide4PGP :: WAV

| Coverfile (bytes) | Stegofile (bytes) | Hidden text (bytes) | Measured Time (ms) | Hidden text / Coverfile (%) | Stegofile / Coverfile | Time / byte (μ s) |
|-------------------|-------------------|---------------------|--------------------|-----------------------------|-----------------------|------------------------|
| 1634348 | 1634348 | 5000 | 183 | 0.3% | 1.00 | 36.60 |
| 1634348 | 1634348 | 10000 | 190 | 0.6% | 1.00 | 19.00 |
| 1634348 | 1634348 | 20000 | 201 | 1.2% | 1.00 | 10.05 |
| 1634348 | 1634348 | 30000 | 228 | 1.8% | 1.00 | 7.60 |
| 1634348 | 1634348 | 40000 | 225 | 2.4% | 1.00 | 5.63 |
| 1634348 | 1634348 | 50000 | 255 | 3.1% | 1.00 | 5.10 |

| | | | | | | |
|----------|----------|-------|------|------|------|--------|
| 8423468 | 8423468 | 5000 | 916 | 0.1% | 1.00 | 183.20 |
| 8423468 | 8423468 | 10000 | 921 | 0.1% | 1.00 | 92.10 |
| 8423468 | 8423468 | 20000 | 944 | 0.2% | 1.00 | 47.20 |
| 8423468 | 8423468 | 30000 | 957 | 0.4% | 1.00 | 31.90 |
| 8423468 | 8423468 | 40000 | 973 | 0.5% | 1.00 | 24.33 |
| 8423468 | 8423468 | 50000 | 985 | 0.6% | 1.00 | 19.70 |
| 20330540 | 20330540 | 5000 | 2198 | 0.0% | 1.00 | 439.60 |
| 20330540 | 20330540 | 10000 | 2207 | 0.0% | 1.00 | 220.70 |
| 20330540 | 20330540 | 20000 | 2227 | 0.1% | 1.00 | 111.35 |
| 20330540 | 20330540 | 30000 | 2241 | 0.1% | 1.00 | 74.70 |
| 20330540 | 20330540 | 40000 | 2260 | 0.2% | 1.00 | 56.50 |
| 20330540 | 20330540 | 50000 | 2280 | 0.2% | 1.00 | 45.60 |

WAV ::Average time 79.5 μ s/byte

ATStream :: TCP/IP

| Hidden text (bytes) | Transferred (bytes) | Measured Time (laps) | Measured Time (ms) | Hidden text / Transferred (%) | Time / byte (μs) |
|--------------------------------|--------------------------------|---------------------------------|-------------------------------|--|--|
| 5000 | 3708704 | 910406 | 2 | 0.1348% | 0.405 |
| 10000 | 7421008 | 1840363 | 4 | 0.1348% | 0.409 |
| 20000 | 14847064 | 3676687 | 8 | 0.1347% | 0.409 |
| 30000 | 22260624 | 5566147 | 12 | 0.1348% | 0.412 |
| 40000 | 29670296 | 7299715 | 16 | 0.1348% | 0.406 |
| 50000 | 37077568 | 9139244 | 20 | 0.1349% | 0.406 |

Average time 0.41 μ s/byte
