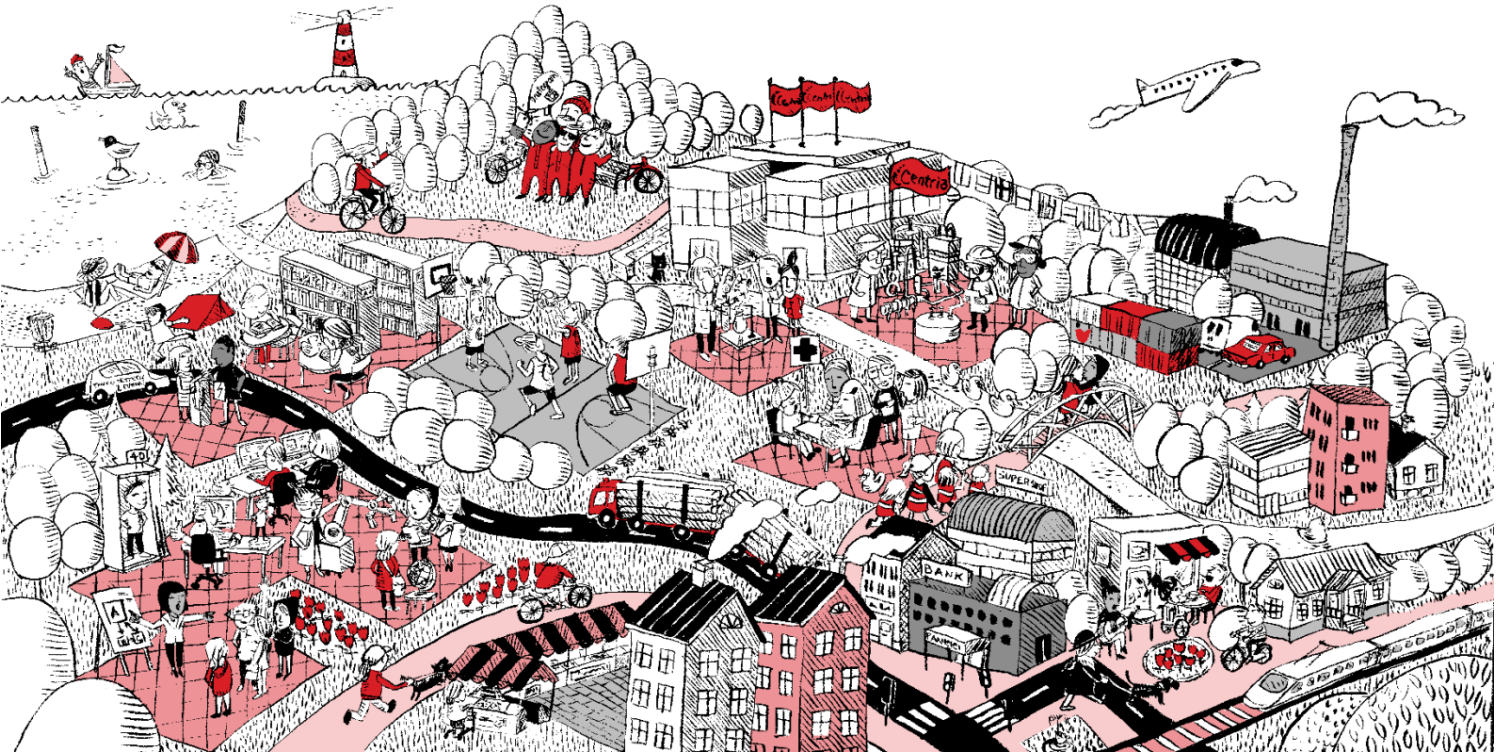
Arthur Maya

STEGANOLOGY AND INFORMATION HIDING

Stegop2py: embedding data in TCP and IP headers

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Bachelor of Engineering, Information Technology
August 2021



ABSTRACT

Centria University of Applied Sciences	Date August 2021	Author Arthur Maya
Degree programme Bachelor of Engineering, Information Technology		
Name of thesis STEGANOLOGY AND INFORMATION HIDING. Stegop2py: embedding data in TCP and IP headers		
Centria supervisor Aliasghar Khavasi		**Pages** 52 + 2

In the time when the explosion of electronic and digital technologies happened, communication technologies concurrently suffered the same growth. Nowadays, for two parties to communicate, it is as easy as dialling a number, or opening an application on a mobile phone provided it is connected to the internet. The benefits provided by such technologies are immense, but in the same manner, so are the privacy concerns.

Advances in cryptography have allowed engineers to develop a vast number of algorithms and tools that can be used to secure communication channels, and even though their security relies upon the fact that a lot of computational power is required for guessing and testing of keys for decryption, they still do not provide total-privacy or secrecy, like the fact that an encrypted message is being sent is still present. Until recent years, steganography has not received the same attention as cryptography, due to the apparent differences in practicality, but that is steadily changing. This research thesis develops on the topic of steganography as a means for secret communication and lays out some examples of the historical use of steganographic techniques in rudimentary media, as well as the use of more modern techniques adapted to more modern technologies.

Even though the topic's domain of application is extremely wide, the primary aim of this work is to express the fundamental knowledge on steganographic techniques and their applications in physical and digital media. A current information-theoretic steganographic communication model is also presented, along with a way to test the performance of a passive opponent in such a system using hypothesis-testing.

As a secondary aim, a proof-of-concept, stegop2py, for using the fields in the TCP/IP headers as a means of secret communication is presented to give the reader more context for the application of these techniques for understanding the wide applicability of these ideas.

Key words
Steganography, Steganalysis, Information Theory, Covert Channels, TCP/IP

ABSTRACT
CONTENTS

FIGURES

PICTURES

TABLES

1 INTRODUCTION

Suppose two friends are communicating some information to each other someplace private. While they speak, an outsider around the corner is eavesdropping on their conversation. This eavesdropping may pose no importance to Alice and Bob or be of great importance if they would not want any outsiders knowing and understanding the contents of their messages. The conversation in this imaginary situation does not necessarily have to be through speech since communication and eavesdropping can take place through many different channels.

The methods and tools that can be used to communicate thoughts, knowledge, and anything for that matter, have been evolving ever since humans started communicating. Technology has opened the doors to increasingly more channels for the transmission of information. Examples of this can be found throughout history: the invention of cuneiform writing initially allowed people to keep track of livestock and other items, and later developed to be used to express richer ideas, like ancient Egyptians' hieroglyphs; later in history, the invention of paper and printing offered the possibility to store knowledge for long periods in the form of books; the discovery of electromagnetic waves gave scientists and engineers the fundamentals for building devices, like radio transmitters that allowed people to communicate through large distances.

There is no doubt that information is one of the most important things in the modern world and it is obvious that these advancements open more channels for humans to express themselves, but this also creates more ways for eavesdropping. Taking the previous examples of communication media: if a clay tablet or book is stolen, the thief can simply read the contents; or if one were to stand within range of a radio transmission with a receiver tuned to the same frequency as the transmitter, it is possible to intercept the transmission.

Ever since there has been the need to hide information from outsiders, regardless of the channel, different techniques have been developed to achieve this. In the same fashion, information hiding has come a long way along with the advancements of technology. Depending on the methods and on the resulting transmissible object, these techniques are widely divided into 2 categories: cryptography and steganography. This academic work focuses on the latter.

The primary aim of this work is to lay out the fundamental knowledge on steganography and to explore its domain and various techniques that are used in the digital era. Complimenting this, the secondary goal is to present a theoretical framework in an example application of steganographic techniques in a digital computer network.

Chapter two elaborates on the history of steganology. A few examples are given on how steganographic techniques have advanced along with technology, and a few applications of these techniques with physical and digital media are shown. In the same manner, the concept of steganalysis is explained shortly. The two different types of steganalysis, distortion and detection, are shortly explained to better understand the applications and the wide domain of application.

When two parties communicate secretly by using steganographic techniques, regardless of the media used, it is possible to describe the security of the communication system, a stegosystem, by using information theory. In chapter three a deeper, more technical dive into what constitutes a stegosystem is portrayed along with an information-theoretical model of a stegosystem proposed by Christian Cachin (2004). The different types of opponents of such a system are also explained and, in addition, the performance measurement with hypothesis-testing is explained.

Due to the broad use of internet services, the possible applications of steganographic techniques are described in chapter four. For this, the fundamentals of the OSI model are presented, as well as the idea behind the TCP/IP protocol stack. More specifically, the possibilities of embedding information in the headers present in the TCP and IP protocols. Afterwards, the proposal of a proof of concept, stegop2py, which uses these concepts to create a stegosystem that can allow two users to communicate over a subliminal channel is presented.

Finally, conclusions on steganography as a means of secrecy in different domains as well as conclusions on the proof that information hiding is possible within the headers of the TCP and IP protocols are drawn in chapter five. Likewise, possibilities for future research parting from this work are given as well as ideas for the development of tools and countermeasures to steganographic methods.

2 STEGANOLOGY

In older times, when the digital era was not even imaginable, apart from speech, information was transmitted via analogue media, i.e., wax tablets, scriptures and writings, art, and other objects. To hide information, the techniques that were used involved the modification of these mediums. Some of the first documentations describing the use of these techniques to hide information are two ancient Greek tales from Herodotus's "Histories" traced back to 440BC (Petitcolas, Anderson & Kuhn 1999).

The first story relates about Histiaeus, the Greek ruler of Miletus. Histiaeus wanted to send a message to his vassal Aristagoras. To hide his message, he proceeded to shave the hair off the head of his best, most trusted servant. Once shaved, he "marked" the message on his scalp. Once his hair had regrown, Histiaeus sent him on to Aristagoras instructing him, once he gets to Miletus, to shave his head and show Aristagoras. (Petitcolas, Anderson & Kuhn 1999.)

The second story relates about the time when king Xerxes of Persia wanted to invade Greece, Demaratus sent a warning message to notify Sparta of the incoming attack. To avoid the message being captured during the journey, he scraped the wax off a wax tablet used for writing, wrote his message on the tablet and then covered it with wax again. This way, the wax tablet would pass inspection by guards that would cross paths during the journey. (Petitcolas, Anderson & Kuhn 1999.)

In circa 1499 Johannes Trithemius, abbot of The Order of St. Benedict, coined the term steganography in his piece "Steganographia" and defined it as: "the sure art of disclosing the intention of one's mind to those who are absent through secret writing" (Trithemius 1499). The book relates about magical spells and spirits summoning out of thin air within its pages, while the "real" message is embedded within puzzles in the book using steganographic and cryptographic methods. The concept of steganography was not known at the time, so it was considered an act of magic and heresy, so it was later added to the Index of Prohibited Books in 1609 (Carbonero y Sol 1880, 309).

For the purposes of this work, the term "steganology" refers to the conjunction of steganography and steganalysis, similar to how "cryptology" refers to cryptography and cryptanalysis (Cox, Miller, Bloom, Fridrich & Kalker, 2007). In a way, it is safe to consider steganography similar to cryptography; both try to add the element of confidentiality or secrecy to a means of communication; basically, to protect

messages from being picked up and understood by outsiders. The methods used, and products that are obtained from them differ, nevertheless.

Cryptography's main aims are to achieve confidentiality using encryption, that is, prevent an eavesdropper from understanding a message being transmitted between two entities, often called plaintext, using an encryption function, which's product is called a cyphertext (Ferguson, Schneier & Kohno 2010, 23-24). Steganography, on the other hand, has the objective to hide or conceal the presence of a message in an innocent, legitimate communication medium, generally called cover-object (appropriately changing the name to cover-text, cover-image, cover-audio, according to the medium), by adding the hidden message to the cover-object using different methods and techniques, thus resulting in a stego-object (Petitcolas et al. 1999, 1063).

Watermarking – a similar idea, is defined as the practice of imperceptibly altering a medium to implant some information about the said medium. In contrast, but similarly, steganography is defined as the practice of undetectably altering a medium to implant a secret message unrelated to the cover-object. (Cox et al. 2007, 2). This concept is out of the context of this academic work but is very useful to contrast the meaning of steganography and to emphasize its aim for secrecy.

TABLE 1. Information hiding system categories (adapted from Cox, et al. 2007 p.5)

	Cover-object has importance	***Cover-object has no importance***
Message presence not known	Covert watermarking	Covert steganographic communication
Message presence known	Overt watermarking	Overt embedded communication

TABLE 1 shows a clear separation of information hiding systems into 4 categories according to the context. The categories are divided by the importance or dependence and value of the cover-object, that is, if the message contains information about the cover-object, and by the knowledge of the existence of a hidden message. Covert watermarking refers to when the cover-object carries some value to the actors of the system and the presence of a message about the cover-object is not known, overt watermarking when the cover-object carries some value to the actors of the system and the presence of a message about the cover-object is known, covert steganographic communication when the message is unrelated to the cover-object and is not known to be hidden within it, and overt embedded communication when the message is unrelated to the cover-object and is known to be embedded in it. (Cox et al. 2007.)

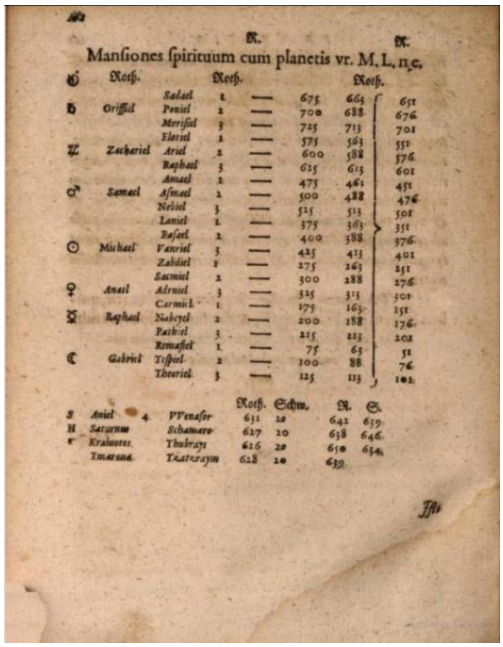
It is usual for the addition of a watermark to any medium to be known, thus discouraging forgery or illicit copying of said medium. This knowledge of the presence of a watermark is allowed because it is the cover-object that carries the primary value, the reason for which it is required for the watermark to be imperceptible. Steganography requires undetectability because it is the embedded message that carries all the value. (Cox et al. 2007, 425-426)

2.1 Steganography

The term steganography is derived from the New Latin word "steganographia", which combines "stegein", Greek for "to cover", and "graphia", Latin for "writing", and as described by Cristian Cachin (2004), "steganography is the art and science of communicating in such a way that the presence of a message cannot be detected". In other words, it is the transmission of hidden data through a carrier that appears to be nothing else than what it is to an outsider. The main goal is to conceal the existence of the data in its entirety (Johnson, Duric & Jajodia 2001, 1). The concept does not have a definite domain of application, since anything that can carry a message can become a stego-object. In the next subchapters, a few examples of the use of steganography are explained, divided by the type of cover-objects.

2.1.1 **Physical**

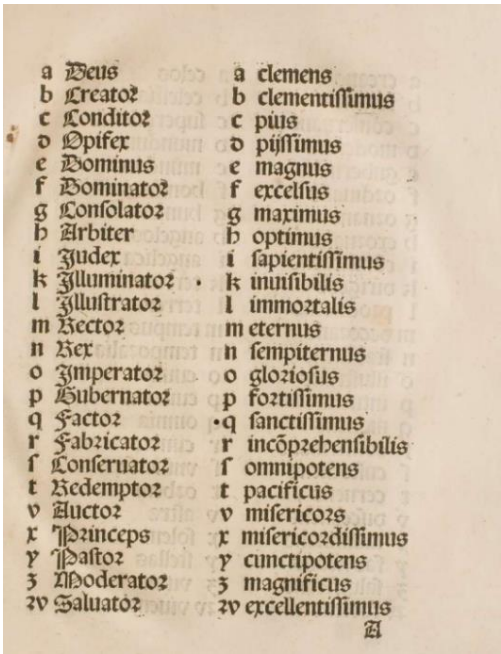
In books I and II of Steganographia, Johannes Trithemius writes about spirits, their names, the astrology related to them in the form of short, poetic, and sometimes nonsensical stego-texts. The plaintext, found within the mass of the contents, can be obtained by taking a sequence of letters from the text depending on the system used, for example, every other letter of every other word. In the works, these systems are literally represented as the summoning of the spirits. In book III, he depicts various tables, like in PICTURE 1, with apparent astronomical information, which then would need to be converted to numbers which in turn would represent letters. This system would be used to retrieve the contents of the book. (Reeds 1998.)



Mansiones spirituum cum planetis vr. M, L, n, c.

PICTURE 1. The first of eight tables in book III of Steganographia (Trithemius 1499).

Steganographia was not the only work published on the matter. In 1518 another work he named Polygraphiae, published posthumously, became the first known printed work about cryptology (Kahn 1968, 133). In this work, Trithemius developed a cypher called “Ave Maria”. The cypher spanned most of the book and consisted of two columns per page. Each column would have the Latin alphabet, and each letter – a representative word next to it as seen below (PICTURE 2). The way these tables were used to encipher a plaintext was by talking the words associated with the conforming letters from consecutive tables. These words make sense when put together.


a Deus
b Creator
c Conditor
d Opifex
e Dominus
f Dominator
g Consolator
h Arbiter
i Iudex
k Illuminator
l Illustrator
m Rector
n Rex
o Imperator
p Gubernator
q Factor
r Fabricator
ſ Conſeruator
t Redemptor
v Auctor
x Princeps
y Paſtor
z Moderator
zv Saluator

a clemens
b clementiſſimus
c pius
d piiſſimus
e magnus
f excelſus
g maximus
h optimus
i ſapientiſſimus
k inuiſibilis
l immortalis
m eternus
n ſempiternus
o glorioſus
p fortiſſimus
q ſanctiſſimus
r incõprehenſibilis
ſ omnipotens
t pacificus
v miſericors
x miſericordiſſimus
y cunctipotens
z magnificus
zv excellentiſſimus

A

PICTURE 2. First page depicting the "Ave Maria" cypher in Polygraphiae (Trithemius 1518).

Textual steganographic techniques can be regarded as the simplest form of steganography, although the techniques applied may also involve a hint of cryptography, like in the previous examples. Steganographic techniques in this domain, therefore, imply hiding plaintexts through cyphering in a visually and intelligibly innocent cover-text. Cyphering can be done in any way imaginable, and in fact, this proves that the less known the method is, the harder and less likely it is to discover.

In contrast to the mystical-at-the-time works by Trithemius, an interesting account of the application of steganography with a very well-known cypher is that of Jeremiah Danton during the Vietnam War. Jeremiah, Commander of the United States Navy was taken as prisoner-of-war in North Vietnam after his aeroplane was shot down in the South China Sea. During a televised propaganda interview he was forced to participate in, he was asked about his position on the actions of the United States government, and while speaking onto the camera, he used Morse code to spell "torture" with his eyes by blinking. This one word confirmed the intelligence reports that the prisoners of war were being tortured, despite denial. (Bredhoff 2006.)

With these examples, it is easy to illustrate the importance of steganography in comparison to cryptography. In the case of the works by Trithemius, the cyphers he used were completely unknown; the real message was hidden in his books employing puzzles he created himself. If the readers of the book would know that there was a hidden message in the books by these means, the readers would still

not have known the messages, since the puzzles would still have to be solved, the puzzles served as a second protection layer, per se. If the knowledge of the presence of a hidden message would have been known in the televised interview with Jeremiah Danton, then further actions would have been probably taken for the worse, for example, not releasing the recording, or re-recording the interview, or something much worse since Morse code is a widely used international standard language. This proves that steganography hides the plaintext in the stego-object imperceptibly.

Emphasizing the fact that anything can be used as a cover-object for the transmission of hidden information, during the harsh times of World War II, a technique was employed by spies to send messages through mail across countries. During these times it was especially difficult to get messages across the borders of the countries that were directly involved because it was widely known that spies were used by many countries; incoming and outgoing mail was inspected to prevent them from communicating back to their headquarters. The technique in question is the use of Microdots to hide paragraph-sized texts in a tiny space. The way Microdots were created was by taking a picture of some text using a film camera, reducing its size to the size of a film frame, roughly the size of a post stamp. To reduce its size further another picture was taken, but by using a reverse microscope, the paragraph's size was reduced to that of a dot of about one millimetre in size. This dot was cut out of the film and then put on a period or similar symbol, or the postage stamp in the cover-text in the correspondence. (Kipper 2003.)

With these examples, it becomes evident that steganographic techniques, from the simplest to the most complex, can take any form and are only limited by technology and one's imagination. With the creation of computers and the arrival of the digital era, the world of steganography suffered the same expansion. All the information in a digital device, from text to images and audio, is encoded with bits of 0's and 1's – a language that computers can interpret. The way these bits are organized to encapsulate the information constitute the different protocols, filetypes, and such that a computer uses to store and process information and communicate, either within the components of itself or with other devices when connected over a network.

2.1.2 Digital

The inception of rich computer media consumption, as well as the internet, gave way to a plethora of new media that people use to communicate daily, and with them, a whole new dimension of methods

and applications for steganographic applications. In the same way that textual steganographic techniques involves the modification, reordering or replacing of character in the text, the same can be done with digital media and the underlying bits of data they constitute. The way computers interpret and store different media gives way to vast possibilities of information hiding within them. (Johnson et al. 2001.)

Images and video are perhaps the most popular domain of all that encompasses the topic. Hiding information within images implies the manipulation of the bit-representation used to interpret the intensity of each pixel. Typically, pixels are represented as blocks of 8 or 24 bits encompassing the "amount" of colour for each channel: red, green, and blue. (Johnson et al. 2001.) Taking into account that digital imaging, and the displaying of such images, takes place with the additive colour model.

Ignoring other colour bit-depths, in a 24-bit image the representation of each channel lies in the value of 1 byte, hence the value can range from 0 to 255, 0 being 0% intensity, and 255, being 100% intensity. In this way, white is represented with all 3 bits set to value 255 as RGB(255,255,255), and black with the 3 bits set to value 0 as RGB(0,0,0). The binary representations, are (11111111,11111111,11111111) and (00000000,00000000,00000000) respectively. The total number of colour combinations possible therefore is $2^{24} = 16777216$. (Johnson et al. 2001.)

Supposing one has a 24-bit image with a high resolution of 1920x1080 pixels, the total amount of pixels in the image results to 2073600 pixels, and if each pixel is represented by 24-bits, the reproduced file size will amount to 49766400 bits, or slightly less than 6 Megabytes. This size may seem viable, but for purposes of reducing the total data during the transmission of such data, and possibly to lower storage space when a large number of images must be stored, many lossless and lossy compression algorithms have been created to achieve this (Johnson et al. 2001).

There are many ways information can be hidden in a digital image. A popular method for hiding information within images is by manipulating the noise, that is, adding information that would rather be imperceptible to the eyes. Taking the above information into consideration, the way this can be achieved is by encoding information into the Lowest Significant Bits (LSBs) of the pixels in each image. (Johnson et al. 2001.)

If we have a 24-bit image, since each pixel has 1 byte per channel, it is possible to store 3 bits of information per pixel. In a 1920x1080 pixel image then, about 6220800 bits or about 760 kilobytes of

information can be encoded in such an image. The resulting stego-image is changed so slightly, that it is virtually imperceptible to the naked eye (Johnson et al. 2001).

FIGURE 1. Cover-image of a water tower (left), a stego-image with an embedded message (centre), the normalized difference between both images (right)

FIGURE 1 shows an example of the use of LSB manipulation as a steganographic technique. In the cover-image on the left, a 2488x2488 JPEG image was embedded the full text of Shakespeare's Othello tragedy play in text format through graph-theoretic LSB exchange. In essence, the algorithm exchanges the LSB of one pixel with the LSB of another one that would result in the embedding of the message. The centre image is the resulting 2488x2488 stego-image. The right image is the resulting difference between the original cover-image and the stego-image. The difference image was normalized to highlight the difference further, since the initial resulting difference was too small to visualize, yielding a nearly black image.

Many more techniques for hiding information within images exist. Not all techniques involve adding noise to an image, but also by manipulating the compression algorithms, or the format of the image (Cox et al. 2007.)

Similarly, for audio steganography, techniques can exploit the sensitivity of the human auditory system by adding signal noise at levels of audio that are imperceptible (for example, LSB manipulation), manipulation of compression algorithms, or modification of audio file formats. (Johnson et al. 2001).

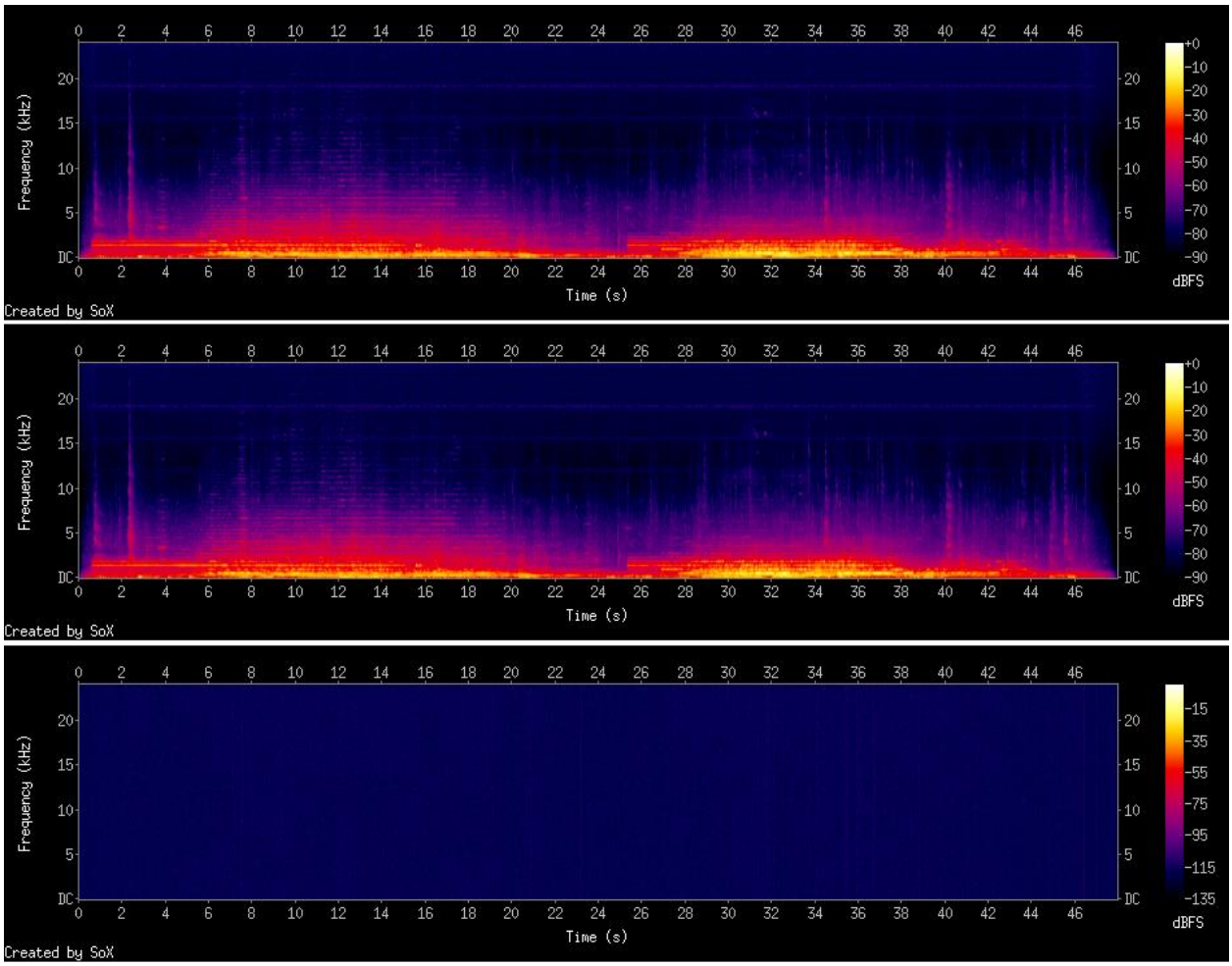


FIGURE 2. A cover-audio spectrogram (top), a stego-audio with embedded message spectrogram (middle), a spectrogram showing the differences (bottom)

FIGURE 2 shows an example of the use of LSB manipulation as a technique used against an audio file. The cover-object, a 47-second-long WAVE audio file of a symphony tuning instruments before a concert, was embedded with the same text of Shakespeare's Othello tragedy play via LSB manipulation. It becomes hard to judge the differences by looking at the respective spectrogram, but it is urged to note the range of the decibels relative to full scale (dBFS) in the spectrogram of the calculated differences and compare it to those of the cover- and stego-audios. The resulting audio file of the differences is silent to human ears.

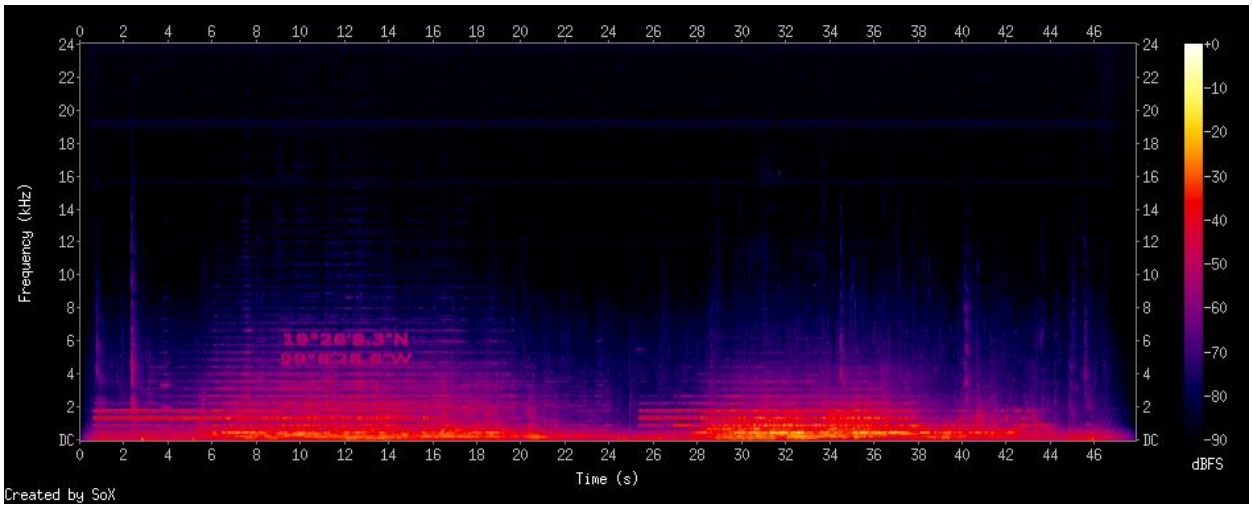


FIGURE 3. Spectrogram of audio with coordinates embedded in visual form

Another form of steganography is the embedding of messages in the form of a visual representation in a spectrogram. This is achieved by creating an audio file that plays the specific frequencies needed to create such a spectrogram. FIGURE 3 shows the same orchestra tuning audio spectrogram as in FIGURE 2 mixed with the audio file created with the message. This technique is employed sometimes by music artists, to watermark their music tracks.

Other methods for embedding messages into audio are possible: adding echoes and delays, masking low and subtle sounds with higher and more dominant sounds, and other methods which are not mentioned to keep brevity. (Johnson et al. 2001.)

If a cover-object has a specific form, order, or arrangement, it is possible to embed messages by altering them in a specific way. Hardware and Software are prime examples of this. The way hardware is organized in a circuit board, for example can be done in such a way that it functions as a watermark, thus identifying it. If this is done correctly, then modifying the layout of the board can be difficult to do without rendering it non-functional. Software code can be organized in the same way. (Johnson et al. 2001.)

Expanding on this idea, it is possible to send messages with hardware by removing or adding specific electronic components in a way that can be interpreted as a message by a receiver. Although the usefulness of such a method is questionable, it could be practical in very specific scenarios. Other very similar methods can be employed with software; the order in which functions are called, or the hierarchy

of polymorphic classes can be arranged in such a way that a message can be extracted by a receiver, given the same algorithm.

Due to the way operating systems allocate files, oftentimes not all the storage space is used. This space, called slack space, varies depending on the formatting of the disk drive where the file is stored, and it can be used to store information. Hidden partitions in a disk can also be used to store information; in fact, some operating systems use a specific form of this idea for securing data. (Johnson et al. 2001.)

Given the need for transferring data and media, files are digitally represented in specific formats that allow them to be read, processed, and serialized by electronic devices; this allows virtually anything in the digital world to become a carrier of a secret message if applied an information hiding technique; the technologies involved are the only limitation.

2.2 Steganalysis

If the goal for steganography is the transmission of messages in a secret way, through innocent media, then if the presence of the message becomes known in any way, it is safe to say, that the method or technique employed has failed. In the same way that cryptanalysis aims to decipher encrypted information as well as break the cryptographic codes and algorithms used to do this, steganalysis aims to discover the presence of embedded information in a carrier or to process the transferred media in a specific way (according to known methods and techniques) to remove or invalidate the embedded information. (Johnson et al. 2001.)

When embedding a message to some cover-object, be it physical or digital, it always involves some form of manipulation which in turn degrades the cover-object a certain amount (Johnson et al. 2001). Depending on the cover-object and methods used for embedding, this degradation can either be easily detected or if the embedding method is very robust, the information can be so integrally embedded into the cover-object that detection is very hard.

There are two categories for steganalysis attacks that can be done to objects that are suspected to be carrying hidden information, and they are divided according to the end goal of the attack (Johnson et al. 2001).

2.2.1 Distortion

The steganalysis techniques that involve distortion refer to the manipulation of the stego-object in such a way to remove the presence of a message in the carrier, thus avoiding the transmission of said message in the covert channel. If it is suspected that a message is invisibly embedded in a carrier, it is possible to make invisible alterations to it to achieve this. Depending on the steganographic technique that was employed to embed data in an object, the distortion of such an object can be easy or require various iterations of alteration to effectively remove the presence of a message. (Johnson et al. 2001.)

The ability to remove an embedded message from a cover-object depends greatly on the robustness of the technique that was employed to hide it. In some cases, the embedding function can be effective to the extent that the message becomes an integral part of the stego-object. In such cases, distorting the stego-object to remove the presence of a message may not be possible without destroying the stego-object itself. (Johnson et al. 2001.)

Depending on the cover-object used, it may be easy and cheap to distort a stego-object to attempt to remove an embedded message. From the previous chapter, if an analyst were to try and distort Trithemius' works to remove the messages from them, it would be very hard or costly. The books are written with ink on paper, therefore the only solutions would be to remove the specific markers that were employed to hide the information in the book, like symbols, tables, and other tools that need to be used by the reader to discover the message. This could be achieved by rewriting the book in its entirety and omitting these markers, but even then, the hidden messages would not be removed completely due to a few of these messages being hidden within the text itself. Due to this, the methods used by Trithemius can be considered robust.

When digital media is used as a carrier for hidden information, distortion can also be employed according to the kind and format of the media. In the example with the JPEG files, the data was hidden by LSB manipulation. Due to the way lossless and lossy formats work, it is possible to convert the file to different formats and back. Doing this will change the way the image data is compressed, therefore completely overwriting the embedded information. (Johnson et al. 2001.)

Taking the example of the microdots added to letters from the previous chapters, applying steganalysis one could use some scraping tools to remove any microdots pasted onto the paper. Rewriting the letter

when it is in transit is another rather invasive possibility. Similarly, the audio files that were used in the previous examples of audio steganography were saved in lossless WAVE format. By converting this audio to a lossy format, like MP3, the data hidden in the LSBs would be erased.

In general, distortion attacks on digital media mostly involve the modification of the underlying bits of information that make up the data; but not exclusively, given that steganographic methods may also modify the data itself as well as the structures they conform to. For example, in the case of the message embedded visually in the spectrogram of the audio file, converting and recompressing the file will not help to remove the image, unless the compression algorithm removes so much information that the audio becomes noise. The point of compression algorithms is to reduce the file size of the media while hindering the quality as little as possible; since the message is represented by audio (and not the bits of data), the message is imperceptibly degraded.

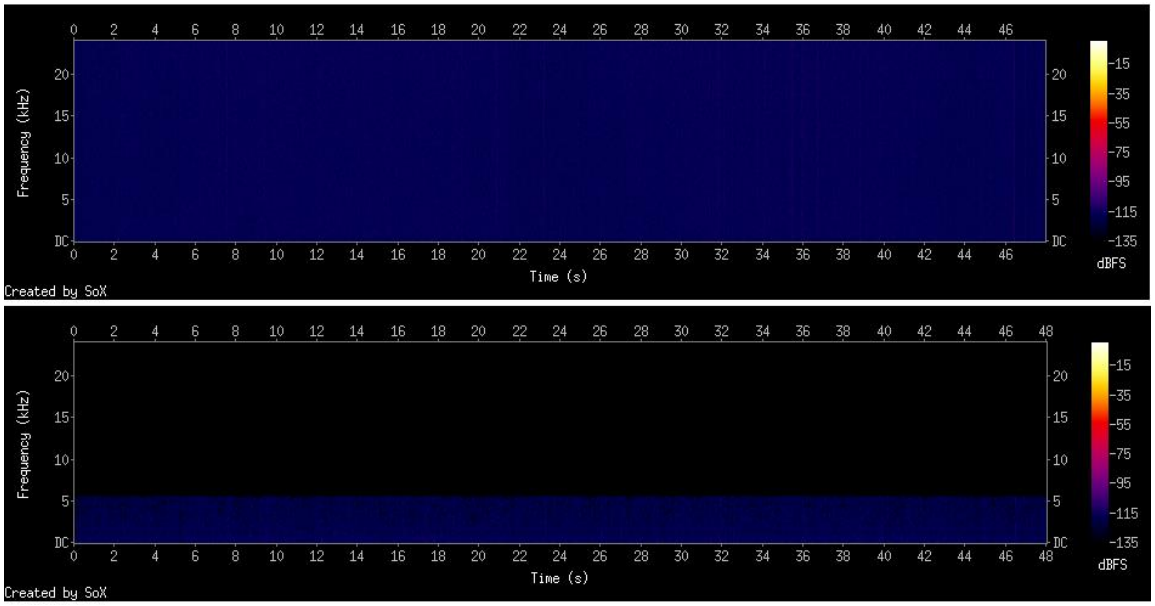


FIGURE 4. Difference between a stego-audio in uncompressed WAVE format (top), and compressed MP3 format (bottom)

FIGURE 4 shows an example of distortion of an audio file. The top shows the difference between a cover- and stego-objects. The resulting WAVE file was compressed to an MP3 file with a 16Kbps bitrate and downsampled to 6Khz. The result is the removal of a great part of the embedded data, which renders the contained text unrecoverable.

2.2.2 Detection

Detecting a message in a stego-object involves applying some method to extract the message from a suspected carrier. When attempting to use detection steganalysis methods, it is also possible to use cryptanalysis methods where the message could be enciphered, as this is usual practice in some domains of steganographic applications. In the same way as distortion techniques, results may vary depending on the robustness of the embedding algorithms, and the steganography objects that are available to the attacker at the time. (Johnson et al. 2001.)

When attempting to attack a stego-object, it is possible that the attacker might not have any information on what the original cover-object is, the algorithms that were used, whether cryptography was used in conjunction, and even if the message itself is known. Likewise, it is possible to have knowledge of all or any combination of these. TABLE 2 describes the different types of attacks divided according to these conditions.

TABLE 2. Steganographic attack types (Johnson et al. 2001.)

Type of attack	***Description***
Stego-only attack	This is a blind attack since only the stego-object is available to the attacker.
Known cover attack	Along with the stego-object, the original cover-object that was used is also known to the attacker.
Known message attack	The message is known to the attacker. In these cases, it is possible to use steganalysis to the stego-object, given the message is known, to find patterns and signs of steganographic use that can aid future attacks, for example, with early recognition of the presence of a message, since the knowledge of a message does not equate to knowledge of the methods, and if the methods are very robust, it still may be very hard to detect in the future.
Chosen stego attack	The steganographic algorithm that was used is known to the attacker.
Chosen message attack	In this kind of attack, different steganographic methods along with a chosen message are applied to detect patterns that may point to the use of a specific steganographic method.

Known stego attack	The steganographic method is known, as well as the original cover-object.

There are many methods, techniques and tools for detection steganalysis. Generally, for physical objects, it is enough to do some sort of physical inspection. Microdots, for example, can be viewed under a microscope to reveal the information contained within (Kipper 2003). To decode the contents of the books written by Trithemius, one must read and study the book to find the patterns and hidden markers to further discover the hidden messages (Kahn 1968, 133).

Detection steganalysis against digital media refers to the analysis of the small distortions that are caused by the methods used to hide information (Johnson et al. 2001). This can be considered as a whole different field of study and is out of the scope of this paper due to a large number of media and formats available.

In the case of FIGURE 1 in the previous chapter, where information was hidden through LSB manipulation, a Known Stego attack was employed to view the difference between the cover- and stego-images, thus revealing the noise caused by the embedding method. For the audio example in FIGURE 2 and FIGURE 3, two methods were used for embedding a message. With the first method, which involved LSB manipulation of the WAVE file, the cover- and stego-audios were compared, and the differences were isolated; the resulting audio had imperceptible artefacts. The second method involved embedding a message visually in the spectrogram of the audio, which was discovered by creating the spectrogram of said audio.

There are a variety of tools freely available for detecting embedded messages in different file formats. One of the most notorious ones, which can be employed in these cases of audio and image steganography, is Binwalk. The original purpose of the tool was to analyse firmware images and attempt to recover files and executables that could be embedded. Nevertheless, it can be used with almost any binary file (ReFirm Labs 2015).

3 STEGOSYSTEMS AND THE SUBLIMINAL CHANNEL

When steganography is used as a means of concealing information, the interaction of the parties, the medium used, the channel, and everything else involved in the process can be described as a system, similar to how a communication system is described. One of the best descriptions of a communication system where secrecy is needed is the famous Prisoner's Problem, written by Gustavus Simmons (1984). This work is one of the first to describe a system where steganography can be used to communicate.

The problem consists of an imaginary situation where two characters, Alice and Bob, are about to be locked in separate prison cells for a crime they committed. A third character, Eve, is a warden in such a prison. The prisoners will not be able to communicate in person once in the cells. instead, Eve will allow both to communicate by sending messages that are delivered to the other via the warden's trusted guards because her aim is to trick them into receiving forged messages making them believe it is a legitimate one sent by the other. Eve allows this only if every message is open for her to read because she is certain the prisoners will try to coordinate a plan to break out of the prison. Alice and Bob want to escape, so they accept the warden's terms. (Simmons 1984.) In this story, the prisoners must find a way to communicate with each other to plan and coordinate their breakout, while accepting the risk of deception. In this situation,

Simmons (1984) argues that the prisoners will only communicate with the other if the warden allows them to authenticate the messages to avoid deception. This story is an analogy to the problem stegosystems are good at solving. Alice and Bob, also called the transmitter and receiver, need to establish a secret communication channel over an existing channel that is being monitored. If Eve, the opponent, finds any signs of the prisoners devising a plan to escape, she will send them both to maximum security prisons where it is impossible to escape; for them to succeed it is critical for them to be able to communicate and plan their escape without Eve ever knowing about their intentions.

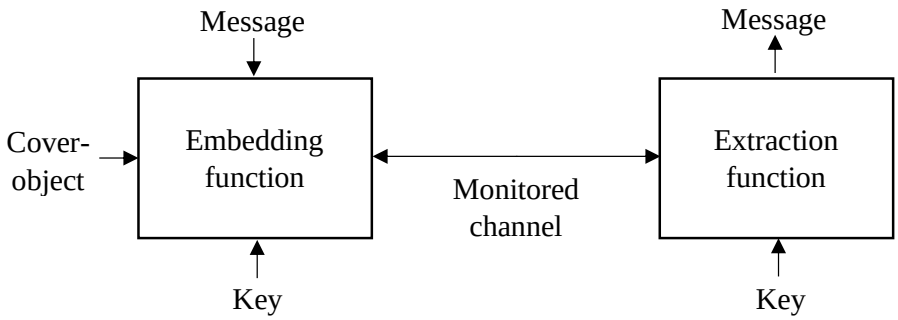


FIGURE 5. A simple stego-system model (adapted from Cox et al. 2007, 427)

FIGURE 5 shows a very simple model of the communication system established by the prisoners. The middle line represents the communication channel that is monitored by the warden. Simmons names this channel "the subliminal channel" because the message they intend to transmit is somehow hidden in the cover-object.

The actions of the warden define the channel's properties. As a matter of fact, in this situation Eve can take 3 stances for acting upon the events in the channel: passive, active, and malicious (Cox, et al. 2007). TABLE 3 describes the different types of stances Eve can take as a warden.

TABLE 3. Types of opponents in the Prisoner's Problem (Cox, et al. 2007)

Type of warden	***Description***
Passive	Alice sends a message to Bob. The message first goes through Eve for inspection. If Eve does not find anything suspicious, Eve can forward the message to Bob. Alternatively, if Eve realises that the prisoners are attempting to communicate secretly, she can stop the communication completely, without modifying the message. Detection steganalysis is mostly employed by this kind of warden.
Active	In this stance, every time either of the two sends a message to the other, Eve applies steganalysis techniques to attempt to remove the presence of a message in its entirety. The methods applicable

	by an active opponent are primarily done to degrade the message slightly to attempt to remove steganographic messages, without making the message unintelligible. Distortion steganalysis is mostly employed by this kind of warden
Malicious	A warden is called malicious when the actions taken have the main goal to discover secret communications between the transmitters. Both detection and distortion steganalysis is employed by this kind of warden.

In the Prisoners' problem, Alice and Bob had to communicate in the presence of a malicious opponent with the chance of deception, giving the need for (explicit) authentication without secrecy. Simmons (1984) describes the creation of a subliminal channel within the techniques used for authentication. These scenarios are out of scope. This academic paper focuses primarily on stegosystems where the users are faced with passive opponents since most practical stegosystems for covert communication face this kind of opponent.

The embedding functions can fundamentally be of 3 types, depending on the cover-object and its modification, or lack thereof. Steganography by cover-object lookup happens when the cover-objects exist in a finite set, that is, they pre-exist and the embedding function aims to select specific cover-objects that when applied some function, for example, a hashing function, the result is the intended message – in the end, it is only the cover-object that gets transmitted since both sides use the same functions. Steganography with cover synthesis happens when the cover-objects do not exist: the steganographic algorithm aims to synthesize the stego-object given certain parameters, for example, in the previous examples of Johannes Trithemius's "Polygraphiae", where the stego-object is formed by concatenating the words that are represented by the letters of a message. Steganography by cover-object modification is by far the most popular and complex type of steganographic function – it is used when the cover-object exists in a finite set, and the aim is to embed the message in the cover-object by modifying it. (Cox et al. 2007)

To make the system more secure, a key can be added to the mix. This key is a redundant piece of information agreed upon beforehand or constantly changed according to a protocol. This key can be used

in various ways, for example as a seed for randomness, or the offset location for the embedding of the message in a stream of bits. (Cox et al. 2007.)

$\boldsymbol{E}: \boldsymbol{C} \times \boldsymbol{M} \rightarrow \boldsymbol{S}$;
$\boldsymbol{D}: \boldsymbol{S} \rightarrow \boldsymbol{M}$

In this way, the set of resulting stego-objects are expressed in the following expression:

$$\boldsymbol{S} = \{(\boldsymbol{c_1}, \boldsymbol{m_1}), (\boldsymbol{c_2}, \boldsymbol{m_2}), \dots, (\boldsymbol{c_n}, \boldsymbol{m_n})\} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_n}\}$$

The embedding and extracting process can be expressed as functions of $\boldsymbol{E}$ and $\boldsymbol{D}$ respectively:

$\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}) = \boldsymbol{s}$; $\boldsymbol{D}(\boldsymbol{s}) = \boldsymbol{m}$;
$\therefore \boldsymbol{D}\big(\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m})\big) = \boldsymbol{m}$

This must follow a rule that for every different cover-object combined with a different message, there should not be any intersections (Konakhovich & Puzyrenko 2006):

if $\boldsymbol{m_a} \neq \boldsymbol{m_b}$, where $\boldsymbol{m_a}, \boldsymbol{m_b} \in \boldsymbol{M}$ and likewise $(\boldsymbol{c_a}, \boldsymbol{m_a}), (\boldsymbol{c_b}, \boldsymbol{m_b}) \in \boldsymbol{S}$,
then $\boldsymbol{E}(\boldsymbol{c_a}, \boldsymbol{m_a}) \cap \boldsymbol{E}(\boldsymbol{c_b}, \boldsymbol{m_b}) = \emptyset$.

Also, if a function that compares similarity is expressed as $\mathbf{sim}(\boldsymbol{C}) \rightarrow (-\infty, 1]$ such that for any $\boldsymbol{c_a}, \boldsymbol{c_b} \in \boldsymbol{C}$ it is true that $\boldsymbol{c_a} = \boldsymbol{c_b} \therefore \mathbf{sim}(\boldsymbol{c_a}, \boldsymbol{c_b}) = 1$, and $\boldsymbol{c_a} \neq \boldsymbol{c_b} \therefore \mathbf{sim}(\boldsymbol{c_a}, \boldsymbol{c_b}) < 1$, then a stegosystem can be considered trustworthy when (Konakhovich & Puzyrenko 2006):

$\mathbf{sim}(\boldsymbol{c}, \boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m})) \approx 1$ for each $\boldsymbol{c} \in \boldsymbol{C}$ and $\boldsymbol{m} \in \boldsymbol{M}$

Given this, a stegosystem can be expressed as the sum of all its constituents (Konakhovich & Puzyrenko 2006):

$$\sum(\boldsymbol{E}, \boldsymbol{D}, \boldsymbol{C}, \boldsymbol{M}, \boldsymbol{S})$$

By adding a key ($\boldsymbol{k} \in \boldsymbol{K}$) to the mix it is simply implied that (Konakhovich & Puzyrenko 2006):

$\boldsymbol{E}: \boldsymbol{C} \times \boldsymbol{M} \times \boldsymbol{K} \rightarrow \boldsymbol{S}^{\boldsymbol{K}}$;
$\boldsymbol{D}: \boldsymbol{S}^{\boldsymbol{K}} \times \boldsymbol{K} \rightarrow \boldsymbol{M}$

Alternatively:

$\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k}) = \boldsymbol{s}^{\boldsymbol{k}}$; $\boldsymbol{D}(\boldsymbol{s}^{\boldsymbol{k}}, \boldsymbol{k}) = \boldsymbol{m}$;
$\therefore \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k}), \boldsymbol{k}) = \boldsymbol{m}$

Resulting in a secret-key stegosystem:

$\sum(\boldsymbol{E}, \boldsymbol{D}, \boldsymbol{C}, \boldsymbol{M}, \boldsymbol{K}, \boldsymbol{S}^{\boldsymbol{K}})$

The secret-key stegosystem relies on a shared secret key between Alice and Bob. In reality, this stego-key exchange is a security risk if it must be done via a transmission protocol. Eve can get this key and use it to her advantage by deciphering the message and reading the contents, thus realizing the prisoner's escape plans, or using it for forging a message pretending to be either one of the prisoners tricking them into planning a fake escape plan. (Konakhovich & Puzyrenko 2006.)

Thanks to the invention of public-key cryptography, it is possible to avoid this by implementing private-public key pairs, instead of a single, secret key (Ferguson et al. 2010). This implies that Alice and Bob each have their public/private key pairs. To encode a message Alice uses Bob's public key, to decode the message, Bob uses his private key, and vice versa. This proves to be secure because the public key is only used for encoding a message, decoding must be done with the private keys, which are not shared over the channel. (Konakhovich & Puzyrenko 2006.)

The use of public-key ($\boldsymbol{k_o}$) and private-key ($\boldsymbol{k_c}$) key pairs in steganographic transformation functions would result in the following (Konakhovich & Puzyrenko 2006):

$\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k_o}) = \boldsymbol{s}^{\boldsymbol{k}}$; $\boldsymbol{D}(\boldsymbol{s}^{\boldsymbol{k}}, \boldsymbol{k_c}) = \boldsymbol{m}$;
$\therefore \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k_o}), \boldsymbol{k_c}) = \boldsymbol{m}$

The resulting stegosystem:

$$\sum(\boldsymbol{E}, \boldsymbol{D}, \boldsymbol{C}, \boldsymbol{M}, \boldsymbol{K} = (\boldsymbol{k_o}, \boldsymbol{k_c}), \boldsymbol{S^K})$$

The security of a stegosystem does not only rely on the embedding and extraction functions and the stego-key used; it is also based on the choice of cover-objects as well as the set of possible messages. Cristian Cachin (2004) defines the security of a stego-system by modelling a secret-key stegosystem using information theory and the theory of hypothesis testing. He describes a stegosystem (FIGURE 6) where the opponent is passive and assumes a secret-key is used for security and Alice can be in one of two states: active – when she is sending messages secretly, and inactive – when she is sending messages without steganographic messages. (Cachin 2004.)

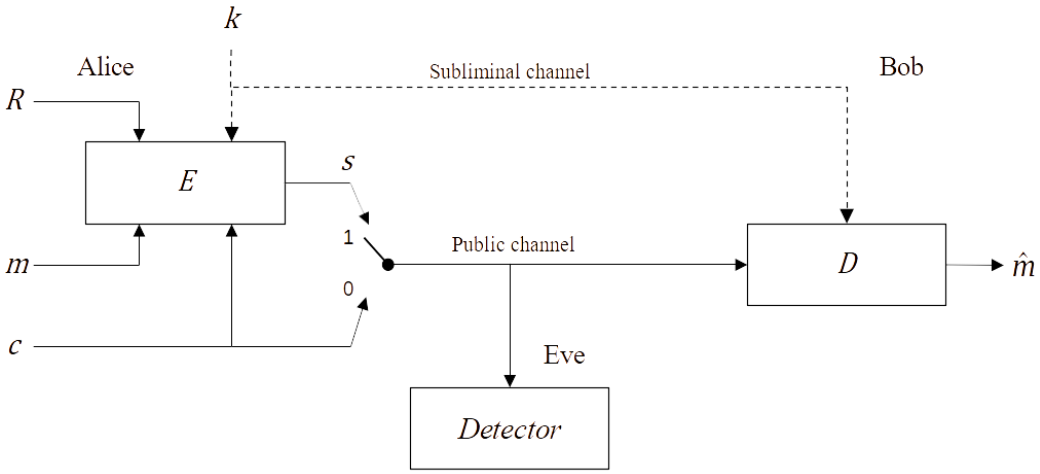


FIGURE 6. Cachin's model on a secret-key stegosystem where Alice is active (adapted from Cachin 2004)

When Alice is inactive the message sent to Bob, the cover-object containing no hidden information is taken (or generated) from a probability distribution of all cover-objects $\boldsymbol{P_C}$. Eve knows this distribution because that is the only way she will allow them to communicate. When Alice is active, she is sending a stego-object $\boldsymbol{S}$, which is created by embedding a message $\boldsymbol{m}$ with function $\boldsymbol{E}$ by also inputting a stego-key $\boldsymbol{k}$ and some randomness $\boldsymbol{R}$ only known to Alice. Cachin assumes the messages are a random variable, from the system's point of view, taken from a set **M**. Given the probability distribution of cover-objects the resulting stego-objects, computed from cover-objects in $\boldsymbol{P_C}$, is therefore expressed also as a probabilistic distribution $\boldsymbol{P_S}$. When the message is sent to Bob through the public channel, Eve and Bob

both read $\boldsymbol{S}$. Bob deciphers the message with the extraction function $\boldsymbol{D}$ and key $\boldsymbol{k}$, resulting in the message $\widehat{\boldsymbol{m}}$ which is supposed to give Bob some information about $\boldsymbol{m}$. (Cachin 2004.)

Cachin (2004) models this stegosystem utilizing the entropy of probability distributions, and the amount of mutual information between $\boldsymbol{m} \in \boldsymbol{M}$ and $\widehat{\boldsymbol{m}} \in \widehat{\boldsymbol{M}}$.

The entropy H of a probability distribution $\boldsymbol{P_X}$ on the values that a random variable X can take from an alphabet $\boldsymbol{\chi}$, is expressed as:

$$H(X) = -\sum_{x \in \chi} P_X(x) \log P_X(x)$$

With this, it is possible to express the entropy of a message $\boldsymbol{m}$ over the probability distribution of messages $\boldsymbol{P_M}$, or in other words, the expectancy or surprisal of a message $\boldsymbol{m}$ when choosing from a distribution $\boldsymbol{P_M}$ as:

$$H(M) = -\sum_{m \in M} P_M(m) \log P_M(m)$$

The conditional entropy of a variable X given a variable Y is expressed as (Cachin 2004):

$$H(X|Y) = \sum_{y \in Y} P_Y(y) H(X|Y = y)$$

where $H(X|Y = y)$ is the expectancy or surprise of X given the condition that $Y = y$, namely the entropy of the conditional probabilistic distribution $P_{X|Y=y}$ (Cachin 2004). With this, it is possible to describe the mutual information between variables X and Y as (Cachin 2004):

$$I(X:Y) = H(X) - H(X|Y)$$

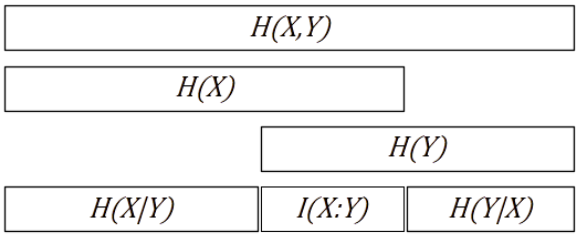


FIGURE 7. The relationship between mutual information and marginal, conditional and mutual entropies (adapted from MacKay 2003)

FIGURE 7 makes it is easier to see the relationship between the entropies of two random variables, and how the mutual information is the reduction of entropy that Y provides about X. With these notions, Cachin (2004) defines a stegosystem (FIGURE 6) as a pair of algorithms $\boldsymbol{E}$ and $\boldsymbol{D}$ if there are random variables $\boldsymbol{K}$ and $\boldsymbol{R}$ such that for all of $\boldsymbol{m} \in \boldsymbol{M}$ where $\boldsymbol{H}(\boldsymbol{m}) > 0$, namely when drawing a message $\boldsymbol{m}$ from $\boldsymbol{M}$ the value is not always expectable, it is true that $\boldsymbol{I}(\hat{\boldsymbol{m}}:\boldsymbol{m}) > 0$, or in other words, that the resulting message after extraction has some information, or resembles, the original message before being embedded. The latter expression implies that the stegosystem is useful to Alice and Bob since the message contents can be interpreted on the other side.

From Eve's side, she does not know whether Alice is active or not, therefore whenever there is a message being transmitted, she uses hypothesis testing to find out whether the transmission contains a hidden message or not. Eve evaluates the message and must agree on one of two hypotheses. The first one dictates that the transmission $\boldsymbol{s}$ does not contain a hidden message; the second one – that $\boldsymbol{s}$ does contain a hidden message. If after testing, Eve finds out that the message does not contain a hidden message, then $\boldsymbol{s}$ is taken from the distribution $\boldsymbol{P_C}$; conversely, if she finds out a message indeed is hidden, then $\boldsymbol{s}$ is taken from the distribution $\boldsymbol{P_S}$. (Cachin 2004.)

Given these distributions, it is possible to mathematically model the security of a stegosystem by means of the relative entropy between these distributions, also called the discrimination between distributions $\boldsymbol{P_C}$ and $\boldsymbol{P_S}$ (Cachin 2004):

With the definition of relative entropy between two probabilistic distributions of random variables:

$$D(P_{Q_0} \parallel P_{Q_1}) = \sum_{q \in Q} P_{Q_0}(q) \log \frac{P_{Q_0}(q)}{P_{Q_1}(q)}$$

Cachin (2004) argues that the secret-key stegosystem (FIGURE 6) is considered perfectly secure if

$$D(P_C \parallel P_S) = \sum_{c \in C} P_C(c) \log \frac{P_C(c)}{P_S(c)} = 0$$

This simply means that a stegosystem is perfectly secure against a passive opponent monitoring all traffic if it is not possible to distinguish between any possible cover-objects and any possible stego-objects. Moreover, Cachin (2004) describes a stegosystem that is ϵ-secure when:

$$D(P_C \parallel P_S) \leq \epsilon$$

As mentioned earlier, it is Eve's task to figure out whether there is a message or not in the transmission from Alice to Bob. This is modelled with hypothesis testing. Supposing Eve makes an observation Q via her detector out of all possible observations $\mathcal{Q}$, the first hypothesis H_0 states that the transmission does not contain a hidden message, and the second hypothesis H_1 states that the transmission does contain a hidden message. Given this, it is possible to define the possible outcomes for both non-stego-object observations and stego-object observations as part of probabilistic distributions. If after making the observation Q hypothesis H_0 is true, then Q is taken from $\boldsymbol{P_C}$; likewise, if H_1 is true, then Q is taken from $\boldsymbol{P_S}$. (Cachin 2004.)

With this, it is possible to measure Eve's detector's performance on the deduction of either hypothesis – a decision rule where one of the two hypotheses is assigned to each observation $q \in \mathcal{Q}$. The detector's results upon a measurement are represented as a bit of information (hence the logarithms in this chapter are implied to be base 2): 0 (false) if it is found that a message is a cover-object, or 1 (true) if it is found that a message is a stego-object. Supposing both scenarios, when Alice sends a cover-object $\boldsymbol{c}$ without a hidden message and when Alice sends a stego-object $\boldsymbol{s}$, two types of errors are possible: a type I error – a false positive, occurs when Eve asserts that the transmission contains a hidden message when it does not in reality (asserting H_1 when H_0 is true), and a type II error – a false negative, which occurs when Eve asserts that the transmission does not have an embedded message when it does in reality (asserting H_0 when H_1 is true). (Cachin 2004.)

The probability for Eve's deduction being a type I error is denoted as α. Likewise, the probability of a deduction being a type II error is denoted as β. Given that each result of Eve's detector's observations is binary, in the scenario where Alice only sends cover-objects the probability of the detector's result being 1 is α, and the probability for a 0 is $1 - \alpha$. Likewise, in the scenario where Alice only sends stego-objects the probability of the detector's result being a 1 is $1 - \beta$, and the probability for a 0 is β. (Cachin 2004.)

The distributions of the results conform to a pair of Bernoulli distributions in the form of $(p, 1 - p)$. The relative entropy of these distributions, namely the binary relative entropy, is derived from the relative entropy equation as:

$$D(P \parallel Q) = p_1 \log \frac{p_1}{q_1} + (1 - p_1) \log \frac{1 - p_1}{1 - q_1}$$

If the probability distribution of the detector's results given only cover-objects are sent is $P = (\alpha, 1 - \alpha)$ and the probability distribution of the detector's results given only stego-objects are sent is $Q = (1 - \beta, \beta)$, then the binary relative entropy for the detector's results is denoted as:

$$d(\alpha, \beta) = \alpha \log \frac{\alpha}{1 - \beta} + (1 - \alpha) \log \frac{1 - \alpha}{\beta}$$

Using the knowledge of the property of hypothesis testing that the entropy does not increase with the deterministic processing of data, Cachin (2004) asserts that the probabilities α and β of the detector's errors satisfies: $d(\alpha, \beta) \leq D(\boldsymbol{P_C} \parallel \boldsymbol{P_S})$.

With this inequality, it is possible to explain what it means for a stegosystem to be ϵ-secure. With the previous definition of $D(\boldsymbol{P_C} \parallel \boldsymbol{P_S}) \leq \epsilon$, it is implied that:

$$d(\alpha, \beta) \leq D(\boldsymbol{P_C} \parallel \boldsymbol{P_S}) \leq \epsilon$$

Cachin (2004) assumes a stegosystem where Eve's detector cannot make false-positive mistakes, rather, only false negatives (the goal for Alice and Bob). This implies that the probability of a type I error is 0.

By plugging this value into the binary relative entropy equation along with the other side of the inequality:

$\alpha = 0$

$\therefore \log\frac{1}{\beta} \leq D(\boldsymbol{P_C} \parallel \boldsymbol{P_S}) \leq \epsilon$

In this way, it is possible to calculate the lower bound on the probability of a type II error, that is, false negatives for an ϵ-secure stegosystem, by solving for β:

$\log\frac{1}{\beta} \leq \epsilon$

$\frac{1}{\beta} \leq 2^{\epsilon}$

$\beta \geq \frac{1}{2^{\epsilon}}$ or $\beta \geq 2^{-\epsilon}$

also applicable to the relative entropy: $\beta \geq \frac{1}{2^{D(\boldsymbol{P_C} \parallel \boldsymbol{P_S})}}$ or $\beta \geq 2^{-D(\boldsymbol{P_C} \parallel \boldsymbol{P_S})}$

This gives insight into what it means to have an ϵ-secure stegosystem. In essence, it can be interpreted such as to say that the smaller the value of ϵ is, and therefore the more similar the distributions $\boldsymbol{P_C}$ and $\boldsymbol{P_S}$ are, the more probability there is that Eve, a passive opponent, commits a type II error, in other words, the more likely it is that Eve commits a false-negative error. For example, if the distributions $\boldsymbol{P_C}$ and $\boldsymbol{P_S}$ were identical, then their relative entropy $D(\boldsymbol{P_C} \parallel \boldsymbol{P_S}) = 0$. (Cachin 2004.)

Using this value in the previous inequality would yield 1, meaning that Eve will always commit errors of type II when a stego-object is sent:

$\beta \geq 2^{-0}$

$\beta \geq 1$

$\therefore \beta = 1$

This information-theoretic model proposed by Cachin is impractical in scenarios like the one in the Alice and Bob analogy, nevertheless. In reality, Eve would not let the prisoners transmit random strings of

bits. (Cachin 2004.) Regardless, this model is very useful in describing the security of a stegosystem, and ultimately, all steganographic schemes try to comply with this model (Cox et al. 2007).

4 INFORMATION HIDING IN THE NETWORK AND TRANSPORT LAYERS OF THE OSI-TCP/IP MODEL

When a user consumes any kind of service on the internet, it may seem like a simple process. Say, accessing a forum to post a picture about a vacation trip is as easy as entering the URL in your browser's address bar, provided the device is already connected to the internet and uploading the picture that was transferred from the device that took the picture to the computer. As mentioned in the first chapters, technology provides the ability to communicate in ways that would not be possible otherwise. To make the users' lives even easier, these processes are covered with many layers that abstract the inner workings of the systems in use to only concentrate on the stuff that matters.

With the expanding world of electronic equipment, the International Organization for Standardization realized the need for a standard for systems that communicate information between each other, since manufacturers would design systems that used different protocols and conventions for their systems; a subcommittee for this issue was created under the name Open Systems Interconnection. The Reference Model of Open Systems Interconnection (OSI model hereon) was designed based on layered architecture, as this was quickly identified to be a viable solution for interconnectivity and compatibility across systems with the ability to grow and expand to adapt to further needs. (Zimmermann 1980.)

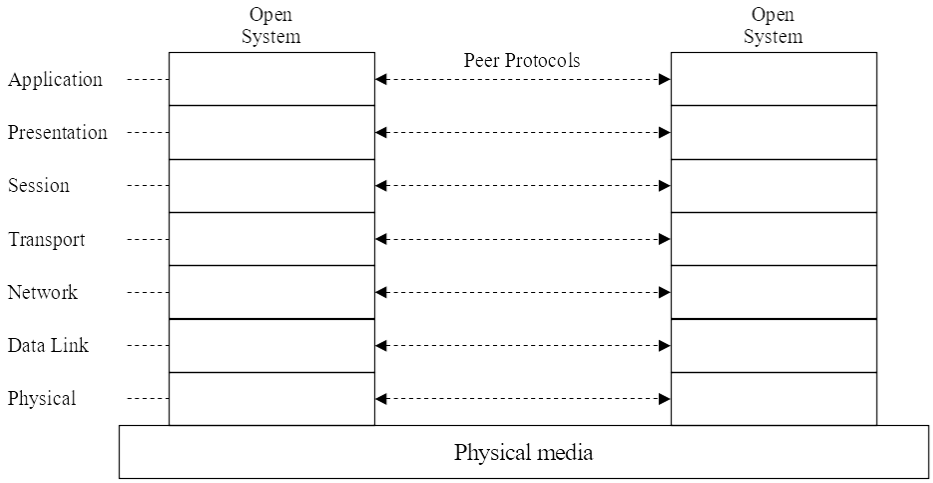


FIGURE 8. The OSI model layers as per ITU-T Recommendation X.200 (ITU-T 1994).

The main principle of the OSI Model is layering and encapsulation. In essence, the model is comprised of various stacked layers (FIGURE 8); each layer encapsulates the layers below and provides an

abstraction for the layer above independently providing a set of services; this allows for the functionality of one layer to be modified, while still providing the same service without breaking the connection to the layer above which depends on it. This abstraction allows dividing the problem of interconnectivity into smaller pieces.

Each layer in the model solves a small piece of the problem, namely, it contains a set of services that follow standard protocols for interacting with the same layer in another interconnected system that follows the same model (Zimmermann 1980).

This modus operandi can be compared to that of a mail and parcel delivery service. Considering that each country has its mail and correspondence system due to different street naming systems, sorting procedures, and other contributing factors, it would be a hassle to understand every one of these details just to send a letter. For this reason, layers of abstraction were created, making correspondence a lot easier. If someone wants to send a letter to their family member across the world, first they write the letter on a piece of paper, then they wrap the letter in an envelope. This envelope later is marked with a destination address and dropped off at a postal box, or the local postal office. Once the postal service processes the letter, it is scheduled for transport. In practice, the destination country is extracted from the address, and transported via air, ground, or across oceans if necessary. Once the letter reaches the destination country and is collected and processed by the destination's international postal arrival terminal, the destination city is extracted from the address, and later directed there via whatever service applies. Once in the destination city, the city's postal sorting facility directs the letter to the destination street, building number, and other identifiers, until collected by the recipient.

With this scenario, it is possible to see the abstraction. For example, if the recipient has a mailbox outside of their home, the letter is sent for delivery via the courier that travels the route where the recipient is located and when the courier reaches the address, the letter is dropped in the recipient's mailbox. If, on the other hand, the recipient lives in a city where correspondence must be picked up at the local post office with an official ID, then the letter is dropped off at the post office and a notification is sent to the recipient. This procedure of delivery is abstracted from the layers above. The service that transfers the letter from the international terminal to the destination city's postal sorting facility does not care how the letter will be delivered to the recipient. In the same manner, the service that transfers the letter between countries does not care how the letter will be transferred to the destination city. However, the layers below do provide a "service" for the layer above. For instance, the destination country has ports where the international correspondence is received and passed on to inter-city delivery service, and

likewise, each city's local distribution centre has reception locations where all the correspondence is collected for further processing.

The OSI model aims to create a similar structure. Each layer contains various similar protocols that provide the functionality to the layer's above, independently from the layers below, thus organizing the functions of the network in a logical structure. TABLE 4 describes the layers in more detail.

TABLE 4. Description of the layers of the OSI model (Tanenbaum & Wetherall 2010)

Layer	***Description***
1. Physical layer	The lowest layer represents the interaction between two nodes and the physical requirements and properties of data transfer. Connections in this layer imply physical connections between the entities, i.e., wired electrical, wireless, fibre-optical, and other types of connections for raw bit transfer.
2. Data Link layer	This layer is responsible for the data transfer and error correction between individual connected nodes on the network.
3. Network layer	The responsibilities found on this layer are the routing and forwarding of the data across the nodes of a network, regardless of the network scope.
4. Transport layer	Independent from the management of network connections, this layer manages and coordinates the actual sending of data across systems on the network. This layer is responsible for carrying the transfer of data and defines the connection parameters required like the source and destination addresses and ports, the data block size, transfer speed, and other parameters, as well as error correction in case of failures, and other functions.
5. Session layer	This layer can be thought of as the endpoint to the network. This layer provides initiation, negotiation, management and termination of connections with other systems.
6. Presentation layer	Independent from the data display to the user, this layer provides a "translation" service through mappings for applications that do not necessarily support the same data semantics and syntax as the data sent down the stack.
7. Application layer	This layer is the closest to the user. This layer takes input from the user for transmission through the rest of the model and displays incoming data to the user in an understandable format. Although the name might suggest

	otherwise, applications running on an end system are not part of the scope of the OSI model. This layer makes it easier for the end-user applications to establish connections with other user applications and services that provide functionality on the connecting end. This does not necessarily mean across the whole OSI model stack as different applications running on the same device can communicate across this layer. In general, it provides an interface for the end-user to the lower layers of the model.

This model describes a more general organization of a system for interconnection of open systems, but another model exists, the TCP/IP or IP model, which is more widely referred to in the internet community due to it being allegedly simpler, more accurate for modelling the Internet, and likely also due to the fact it was created before. This model precedes the OSI model, its main goals were to be able to connect multiple networks seamlessly and be impervious to the sudden loss of hardware between the source and destination machines in the network. (Tanenbaum & Wetherall 2010.)

Both models have received a fair amount of critique due to differing ideas and points of view and backgrounds, although both have much in common and neither is perfect. The OSI model clearly distinguishes the three main concepts – services, interfaces and protocols, in the general idea; services are performed at each layer for the layer above, interfaces inform the layer above how to access these services, and protocols are used by each layer at its scope. The TCP/IP model originally did not clearly distinguish these ideas. Also, the OSI model was designed before the underlying peer protocols were created, while the TCP/IP model was designed according to protocols that already existed, therefore the TCP/IP model fits the protocols perfectly, while the OSI model is more general, hence the criticism. (Tanenbaum & Wetherall 2010.)

Though the discussion on the differences of these models and a deeper dive into the protocols is out of the scope of this paper, a combination of both models is taken into account, as discussed by Tanenbaum and Wetherall (2010). This hybrid model uses five of the OSI model's layers, excluding the presentation and session layers, for presenting network topology and TCP/IP's protocol suite due to the wide community acceptance and historical use (Tanenbaum & Wetherall 2010).

There are more internet-connected devices now than ever before. For making sure all these devices are supported by the service and product providers on the internet, protocols are put in place which ensures that providers and consumers speak the same language. Mostly, the protocols in the TCP/IP suite are

responsible for enabling and making sure of the correct transmission of data across networks. The main two protocols in the stack are TCP and IP, hence the name. The TCP (Transmission Control Protocol) is found in the transport layer, and it is responsible for reliably delivering streams of bytes to devices that are connected over an IP network in an ordered way while preventing and correcting errors that arise along the way. The IP (Internet Protocol) complements the TCP and is the main protocol in the suite; it allows for blocks of data to be forwarded between network boundaries along the path from the source to the destination. (Tanenbaum & Wetherall 2010.)

In essence, the way these protocols achieve their goals of correct data transmission across networks and geographical distances is by means of fragmentation (splitting streams of data into blocks) and encapsulation (wrapping these blocks with a header and defining the size of the block). When a host application wants to transmit a stream of data to an application on another host, the data is prepended with a TCP Header (layer 4) which contains the source and destination ports of the communicating machines and other information used by the TCP resulting in a TCP Segment (Postel, RFC 793 1981). When the segment is ready for transmission across the network, the IP may divide (fragment) the segment into blocks if it is larger than the Maximum Transmission Unit (MTU) size, and prepends an IP Header (layer 3) that contains the source and destination addresses, along with information needed to reconstruct the TCP segment if fragmented, and other information specific to the IP, resulting in a packet (Postel, RFC 791 1981). The headers for IP and TCP are described in detail in RFC 791 and RFC 793 of the IETF, respectively.

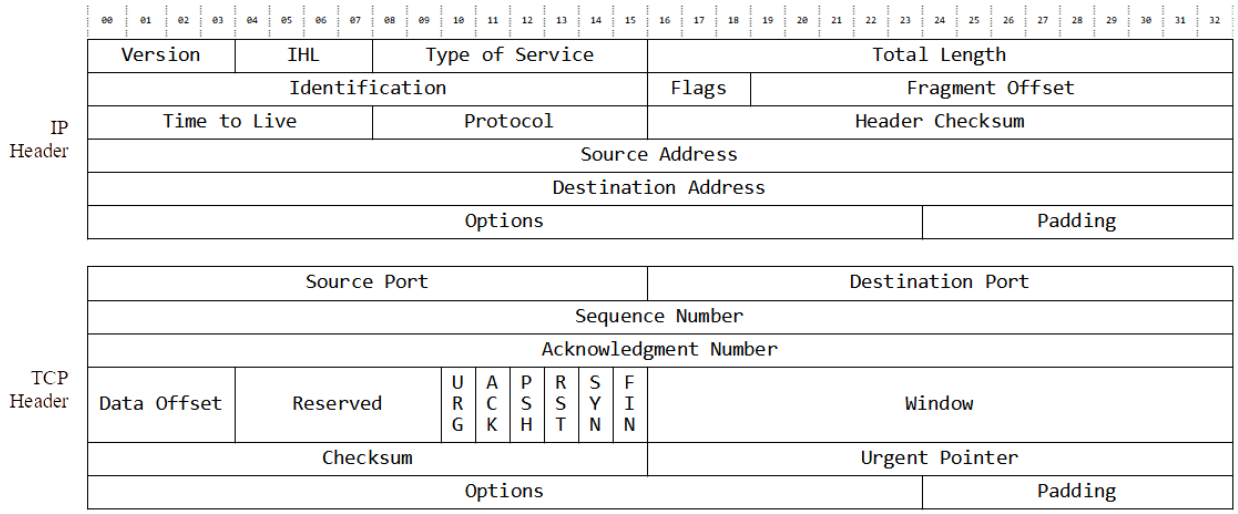


FIGURE 9. IP and TCP header formats as per RFC 791 *(Postel, RFC 791 1981)* and RFC 793 *(Postel, RFC 793 1981)*

Without loss of generality, the TCP and IP protocols are selected for study due to their widespread use in most internet services. FIGURE 9 depicts the structure of the IP and TCP headers that are prepended to the respective Protocol Data Units (PDU, i.e. segment or packet) during data transmission. Murdoch and Lewis (2005) elaborate on the application of steganography to fields in TCP/IP headers and identify those that could be employed for embedding messages considering a passive opponent. The authors point out that the header fields that can be used for embedding information are those that can take an arbitrary value without compromising the validity of the header, in other words, those fields that if modified the warden would not be able to detect the modification. In the IP header, the fields that are usable for steganographic purposes are Type of Service, Identification, Flags, Fragment Offset, and Options (Murdoch & Lewis 2005), and those in the TCP header are Sequence Number and the Timestamp option in the Options.

The IP Type of Service header field is 8 bits long and is used to indicate the abstract parameters of the quality of service desired along the path of the packet such as delays, reliability and throughput (Postel, RFC 791 1981). The issue with using this field for embedding messages is that the opponent can easily detect if this is being used, since it is set to zeroes by default in most systems, and are rarely used (Murdoch & Lewis 2005).

The IP Identification field contains a value set by the sender that identifies the packet a fragment belongs to, aiding in the reconstruction of the packet at the receiving end, in other words, when a packet is fragmented, this field is used to identify the fragments that form one packet from the fragments of another (Postel, RFC 791 1981). Given that the only constraint in this field is that the value must be unique in a certain timespan to avoid the fragments of one packet being reassembled onto another one, there currently are implementations that embed random data into this field although in some cases their use can be detected since that the data in this field is not random, and may follow some detectable patterns (Murdoch & Lewis 2005).

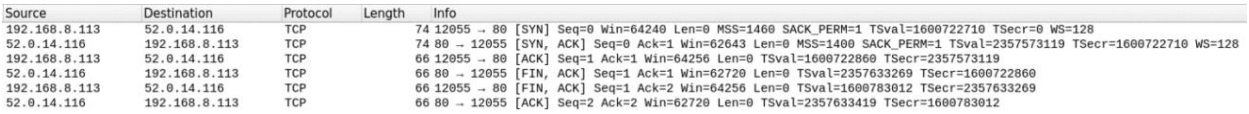
Referring to IP packet fragmentation, in the header, there exists a 3-bit field for Flags that are used as control parameters. The leftmost bit is reserved, therefore always zero. The middle bit (DF, for Don’t Fragment) is used to indicate if a packet may fragment (if set to 0) or if it should be discarded if it is too large (if set to 1), and the rightmost bit (MF, for More Fragments) is used, if the packet is fragmented, to indicate wether the packet is the last one in the sequence (if set to 0, also used to indicate if the packet has not been fragmented), or if there are more packets to be transferred in the sequence (if set to 1)

(Postel, RFC 791 1981). This field is not very useful for steganographic applications because the opponent can predict the correct values for these bits given the context of the communication (Murdoch & Lewis 2005).

Also, referring to IP packet fragmentation, the Fragment Offset field is used to indicate the correct offset in octets location where the packet belongs, aiding the receiving host with the reconstruction of the sequence (Postel, RFC 791 1981). It is possible to embed information in this field by controlling the sizes of fragmented packets, therefore, controlling the value in this field, but it is easily detectable since packet fragmentation is unusual in systems where Path MTU Discovery (PMTUD) is used (Murdoch & Lewis 2005).

The IP Options is an optional field in the IP Header that allows for extra control functions, in the likes of timestamps, security and special routing options that may be useful, and sometimes necessary in some scenarios (Postel, RFC 791 1981). In normal scenarios, though, this field is not needed so it is extremely rarely used, and thanks to this it is not very useful for steganographic purposes as it results to be very easily detected, even though some implementations exist that exploit the timestamp option in this field (Murdoch & Lewis 2005).

To achieve reliability of data delivery, TCP has two 32-bit (4 octets) fields in its header, the Sequence Number field and Acknowledgement field, which are used by the hosts on a network to correctly verify that all sequences of data have been received since every octet of data is "assigned" a sequence number. When a connection (two communicating sockets) is created, the first host (hereon host A) sends an SYN segment, where the TCP header has the SYN (for synchronize) flag set to 1 and the Sequence Number field set to a generated initial sequence number (ISN hereon). When the second host (hereon host B) receives this sequence, its task is to reply with a confirmation of the initiation of the connection and to synchronize with host A its ISN. To achieve this, host B replies with an ACK segment, where the ACK (for acknowledge) is set to 1 and the Acknowledgment Number field is set with host A's next sequence number (host A's Sequence Number field + 1); this is done in conjunction with setting the SYN flag to 1 and host B's generated initial sequence number to the Sequence Number field. When host A receives host B's segment, host A replies with an ACK segment by setting the ACK flag to 1, and host B's next segment number (host B's Sequence Number field + 1) to the Acknowledgement number. (Postel, RFC 793 1981). When finalizing a connection, a similar procedure is executed, only instead of the SYN flag, the FIN (for finalize) is set to 1. (Postel, RFC 793 1981)



Source	Destination	Protocol	Length	Info
192.168.8.113	52.0.14.116	TCP	74	12055 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1600722710 TSecr=0 WS=128
52.0.14.116	192.168.8.113	TCP	74	80 → 12055 [SYN, ACK] Seq=0 Ack=1 Win=62643 Len=0 MSS=1400 SACK_PERM=1 TSval=2357573119 TSecr=1600722710 WS=128
192.168.8.113	52.0.14.116	TCP	66	12055 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1600722860 TSecr=2357573119
52.0.14.116	192.168.8.113	TCP	66	80 → 12055 [FIN, ACK] Seq=1 Ack=1 Win=62720 Len=0 TSval=2357633269 TSecr=1600722860
192.168.8.113	52.0.14.116	TCP	66	12055 → 80 [FIN, ACK] Seq=1 Ack=2 Win=64256 Len=0 TSval=1600783012 TSecr=2357633269
52.0.14.116	192.168.8.113	TCP	66	80 → 12055 [ACK] Seq=2 Ack=2 Win=62720 Len=0 TSval=2357633419 TSecr=1600783012

PICTURE 3. Visualization of the synchronization (SYN) and finalization (FIN) of a TCP conversation

In general, when transmitting a stream of data, each TCP sequence assigns the first octet's sequence number to the Sequence Number field in the header; when a host replies, it sets the next expected sequence number to the Acknowledgement Number field in its response. PICTURE 3 shows an example of the initiation and finalization of a TCP conversation between a local machine (IP address 192.168.0.113 on port 12055) and the servers for NASA (nasa.gov, IP address 52.0.14.110 on port 80). APPENDIX 1 shows the breakdown of the IP Packet and TCP segment for the first incoming frame (the second entry in PICTURE 3).

When a TCP conversation is started, the ISN must be generated with much care, to avoid overlapping sequence numbers with multiple instances of connections in the system and minimize security risks of attacks to the TCP stack; the implementation varies between operating systems as there is no recommended implementation (Postel, RFC 793 1981). Rather, given the security constraints, the RFC 6528 proposes an ISN generation algorithm that takes into account a counter (M) that is incremented every 4 microseconds and a pseudo-random function (F()) that takes the source and destination addresses and ports, and a secret key for the generation of an ISN that is difficult to guess for a host that is not part of the conversation (Gont & Bellovin 2012):

$$\text{ISN} = \text{M} + \text{F}(localip, localport, remoteip, remoteport, secretkey)$$

A few implementations that exploit the ISN number have been developed, but given that the ISN generation function implementations for different operating systems are known, and the ISNs themselves are not random as they follow certain structures, they are detectable (Murdoch & Lewis 2005). The characteristics of the resulting ISN number vary for each system and are constantly updated.

Within the Options field in the TCP header, Timestamps can be added by the host systems for measuring the total time it takes for a round-trip between hosts for aid when channel bandwidth is low or delays are high. This field can be used to implement a covert channel by modifying the LSB of the 32-bit timestamps. This is detectable, though, by studying the randomness (or entropy) of the LSBs, or by

calculating the ratio of the number of different timestamp values observed in the TCP segments where the transmission rate is higher than the timestamp update rate. (Murdoch & Lewis 2005.)

Considering the statements in the previous chapter, in a stegosystem where messages are embedded in the headers of the underlying network protocols of the network environment, a passive opponent would know the generating functions for the ISN and IP ID fields that each host is using. Even though the ISN and IP ID numbers should not be deducible from the context, they do follow specific characteristics. Given this, an implementation that simply encodes the information directly into the fields has the weakness that the patterns are not followed. This allows the warden to easily detect that the fields contain anomalies. (Murdoch & Lewis 2005.)

5 STEGOP2PY

A proof of the concept of networking steganography, Stegop2py, is proposed[1]. By exploiting the IP Identification and the TCP Sequence number fields, it is possible to create a subliminal channel that can be used to transmit data secretly, from one host machine to another across networks. It is a peer-to-peer chat program, that utilizes the concepts in the previous chapters to secure messages being sent by embedding them in chains of random bits at random offsets.

The initial idea behind the program was to embed messages solely into the TCP Sequence number field. The first issue with this method is that the only time when a message will be embedded is on the very first packet sent to the other host (during 3-way-handshake), given that the sequence number grows according to the number of bytes that were sent with a packet – an implementation like this would require the initiation of various TCP connections for every message. The second issue is that the message could be a maximum of 32 bits long, which is very short for any realistic data transfer. These issues made this method unviable to send messages in real-time while making the TCP communication seem legitimate. This was averted by not using the TCP Sequence Number field as the message carrier. Instead, the encrypted message is embedded into a stream of random data (FIGURE 10).

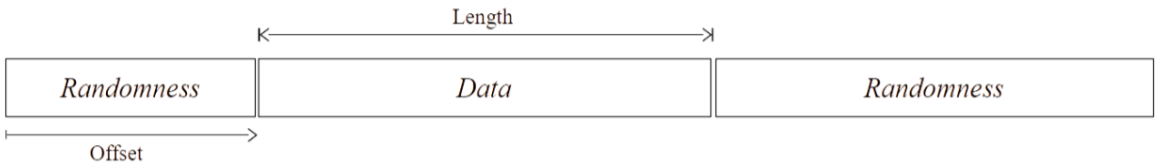


FIGURE 10. Depiction of the data embedded in the chain of randomness

5.1 Architecture

To be able to use the TCP and IP headers as cover-texts for the stegosystem, the program should be able to send out forged packets including the TCP and IP headers, as well as receive them also including the TCP and IP headers. When programming sockets in Linux, one must choose the base protocol on which

[1] The source code for stegop2py can be found at: https://github.com/kn1dar1an/stegop2py

it will run – if opening a Stream Socket (TCP protocol), the kernel will handle the TCP functionality for you. Using these sockets, however, does not allow you to send forged sockets, therefore are not viable for this program. By using raw sockets in Linux it is possible to take control of layers 3 and 4 and use them to send and receive forget packets. Referring to the abstraction layers of the OSI Model discussed in the previous chapter, the Network and Transport layers are abstracted from the programs running in the layers above when using normal sockets, therefore, when using raw sockets, these abstraction layers disappear along with the services that ran in them, eliminating socket functions like accept() or connect(). Thanks to this, IP and TCP functionalities must be handled manually.

Stegop2py uses raw sockets to receive incoming forged packets, but since raw sockets are connectionless and are not bound to a port, the data must be filtered out from all incoming data at layer 3 (IP). The filtering is basic and is done by comparing the destination IP address, and destination port of the packets to the IP given by the user and predefined port number; this should be enough for these purposes. Handling of the packets, like parsing and forging them, is done with the help of Scapy, a packet manipulation library for Python. Scapy helps to parse incoming data into scapy.Packet objects which contain the IP and TCP headers and their fields' values, and to forge and send them at the TCP/IP layer 3.

For handling incoming data at the same time as waiting for the user to input a message and sending it, threads were used. One thread is spawned for listening for incoming messages and decoding them once the connection is "established", and another thread for a window manager which handles the user's input and the displaying of the messages in the terminal through the NCurses console management wrapper for Python.

5.2 Stegosystem

For steganographic purposes, the messages are embedded into chunks of random bits at a constant, per-connection offset. This is done to mimic the transfer of encrypted data, which looks like random data, such as with TLS. Even though this will not make the sent packets look like TLS encrypted data when analysed, the idea is to prove the point, because implementing TLS would require writing a TLS implementation from scratch, which is out of the scope of this work.

For extracting the data, the receiving side must know the offset that was used to embed the message and the length of the message. These are considered as the stego-keys for the system. The length key is transferred to the user by embedding it into the lowest significant byte of the IP Identification field, allowing for values from 0 to 255. In the case of the per-connection offset, it is randomly generated at the start of each session, and embedded into the Initial Sequence Number when handling the 3-Way-Handshake. The forged headers are then appended to the message data.

Before embedding, though, the message is first encrypted with a password (which is assumed to be a pre-shared password between the users of the system) via AES. Once encrypted, the salt that was used for encryption is appended to the message for later decryption. The IV (Initialization Vector) and key for the AES algorithms are derived from the password and a randomly generated salt every time a new message is encrypted. For decrypting, the same salt and password are used to derive the same IV and key. The salt is randomly generated every time a message is sent, therefore it can be public. If the password is different to the one that was used for encryption, the decrypted data will be corrupt.

5.3 Classes

The present classes in the Stegop2py application are described in detail in TABLE 5. A detailed class diagram with the class methods and attributes and the relationship between them can be found in APPENDIX 2.

TABLE 5. Stegop2py classes and their description

Class (Filename)	***Description***
Client (Client.py)	This class creates and contains the instances for the socket connection and the window manager. It is also responsible for managing the shutdown of the threads in question. Lastly, this class contains the callbacks for queuing a message when the user inputs one, and for printing system messages when something goes wrong. The callables are passed to the WindowManager and the Connection classes respectively as constructor parameters.
Connection (Connection.py)	This class is responsible for everything related to the connection to the other host. At start-up, the client tries to connect to the host address given by the user as a command-line argument. If the connection is unsuccessful, then

	the connection simply listens for incoming connections at the address provided by the user as a command-line argument. The 3-way-handshake is also managed by this class, both for initiating them and handling incoming attempts.
Stegocoder (Stegocoder.py)	As the name suggests, this class manages the embedding of messages into randomness. This includes the generation of the local Initial Sequence Number containing the offset that will be used for embedding, storing of the offset that was used by the other host for extraction, and encrypting and decrypting of the messages. The encrypting of messages is done before embedding with a password. A salt is used in conjunction with the password to generate the necessary Initialization Vector (also called nonce) and key for the AES-256 block cypher; this is achieved by using the PBKDF2 key derivation algorithm. The AES cypher is instantiated every time a new message is sent, to avoid using the same salts and keys. These algorithms are implemented by using the PyCryptodome library.
WindowManager (WindowManager.py)	This class is responsible for showing the formatted messages in the terminal. It also runs on a separate thread since it runs in a loop, which constantly checks for new incoming messages from the queue and outgoing messages input by the user. For Python, there is a built-in module, Curses, which wraps the original ncurses C library for terminal manipulation.

5.4 Prerequisites

Stegop2py was built using Python 3.9.5 on Arch Linux with kernel version 5.10. Logically, the Scapy[2] and PyCryptodome[3] must be installed via pip.

There is an issue when receiving TCP segments with raw sockets. When packets arrive at a port that does not have an active stream socket at that port, the kernel responds to all requests by sending TCP

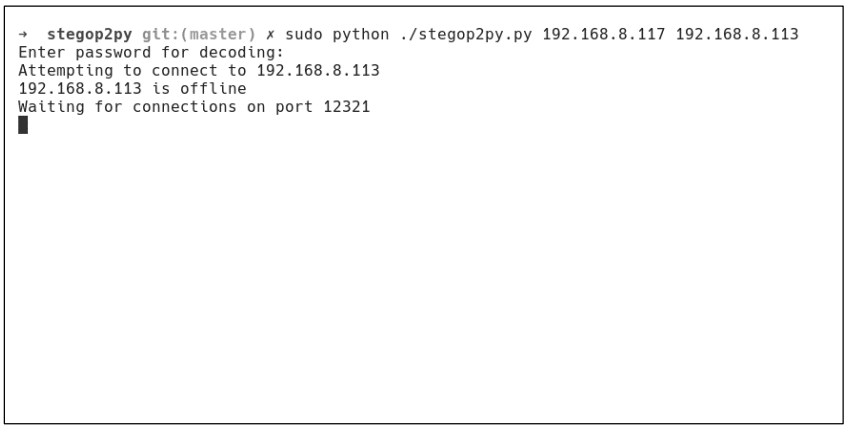
[2] Documentation for the Scapy library can be found at: https://scapy.readthedocs.io/en/latest/
[3] Documentation for the PyCryptodome library can be found at: https://www.pycryptodome.org/en/latest/

RST segments. To go around this issue easily, it is possible to add a firewall rule entry to drop the outgoing TCP RST segments by using iptables[4].

5.5 Usage

The program can be run on most modern terminal emulators on a Linux distribution. The program can be started by launching the stegop2py.py Python script. The script requires 2 positional arguments, the first is the local IP address where the program will be serving (listening), the second is the other host's IP address to try to connect to. Given that raw sockets can only be created and used by superuser in most Linux distributions, the Python interpreter must run as root, for example by using sudo:

```
# python ./stegop2py.py 102.142.34.94 92.43.184.24
```



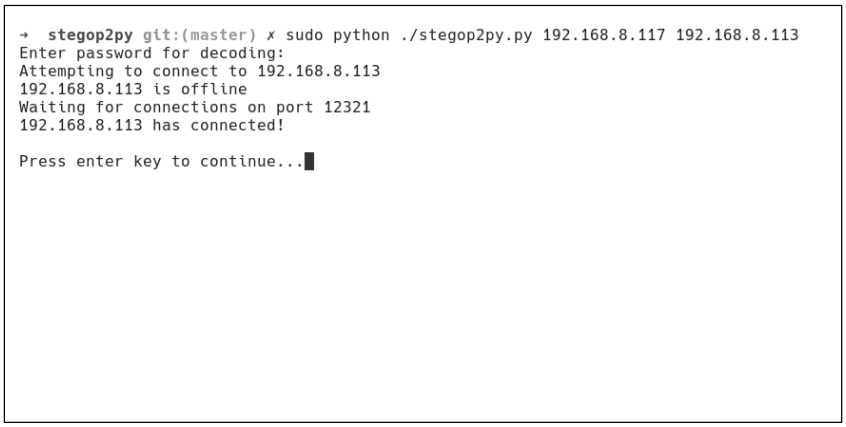
PICTURE 4. Stegop2py instance listening for incoming connections

When running the script, the program will prompt the user for a password. This password will be used for encrypting and decrypting the messages in the session. It is assumed that both hosts enter the same password, to allow the messages to be decrypted correctly. As seen in PICTURE 4 after the password prompt, the program attempts to connect to the host indicated by the script's second positional argument on port 12321 by initiating a 3-way-handshake, expecting the target host to reply to the requests, as per the TCP standard. This will be attempted 5 times with a timeout of 2 seconds per request in the case when no response is received.

---

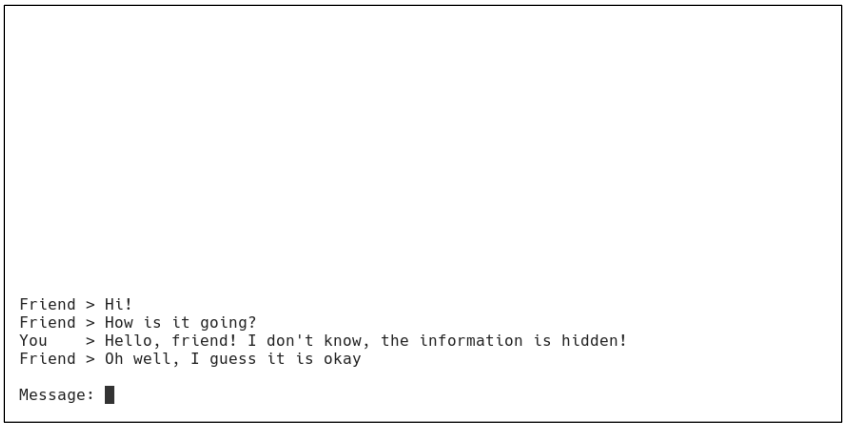
[4] Example iptables command: iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP

If the target host is offline, the program will listen to all traffic, waiting for a connection request with port 12321 as the destination port. Once a connection request (TCP SYN packet) is received, the program will handle the request by responding to the 3-way-handshake initiation. Once the connection is established, the program will ask to press enter to begin sending messages as seen in PICTURE 5



PICTURE 5. Stegop2py instance receiving a connection request and connecting to the host

As soon as the user presses the enter key, messages can be sent and received. As pictured in PICTURE 6, the messages are displayed on the terminal window in chronological order, and a message input line is also present. As the user types the message, it will be echoed in the terminal as expected. When sent and received messages are displayed on the screen, the source of the message is prepended to each line. If the message was sent by the user the word “You” will be prepended, if the message was received the word “Friend” will be prepended, if the message is an internal, system message the word “System” will be prepended.

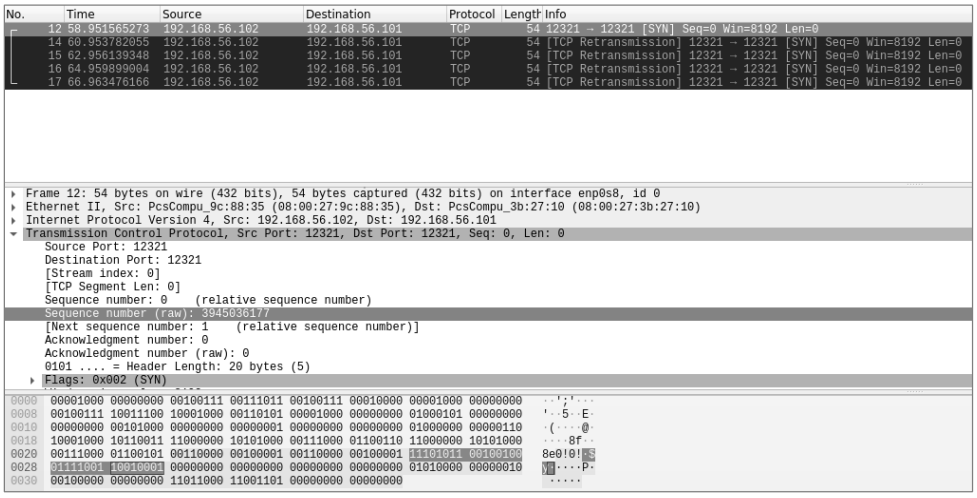


PICTURE 6. A Stegop2py instance having a conversation with the connected host

5.6 Packet analysis

When analyzing the network traffic generated by Stegop2py with virtually any packet capture software, it is possible to visualize the 3-way-handshake in action, as well as the messages being transmitted from host to host.

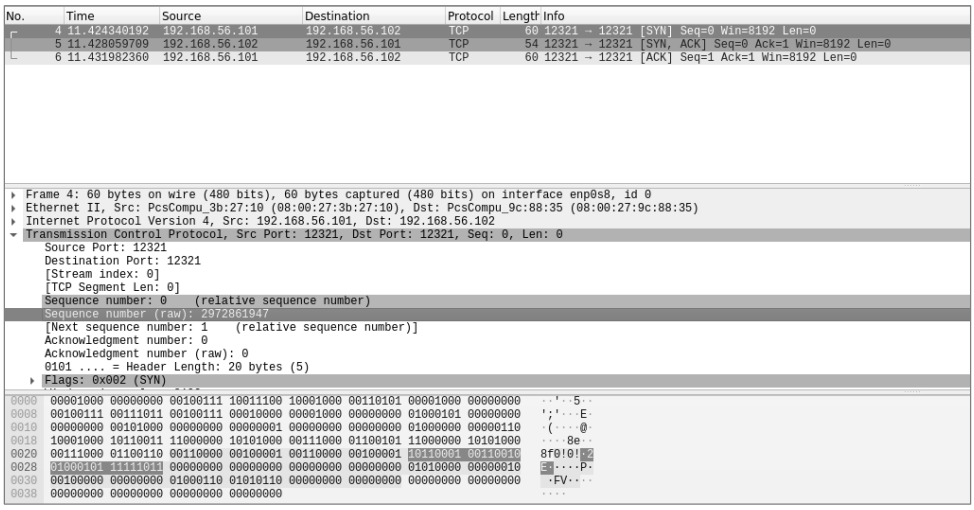
When initially attempting to connect to a host, Stegop2py attempts to connect 5 times, with a 2-second delay each time. As seen in PICTURE 6, Wireshark shows the last 4 attempts as TCP retransmissions, since the destination host did not reply. When connecting, the data offset number is transmitted to the destination host via the randomly generated initial sequence number (ISN), embedded in the least significant byte, as highlighted in the bottom panel in PICTURE 7 ($10010001_2 = 142_{10}$, i.e., 142 bits from the beginning of the payload. When a connection is unsuccessful, Stegop2py stops attempting, and starts listening for connections.



PICTURE 7. Wireshark packet capture showing initial attempt to connect

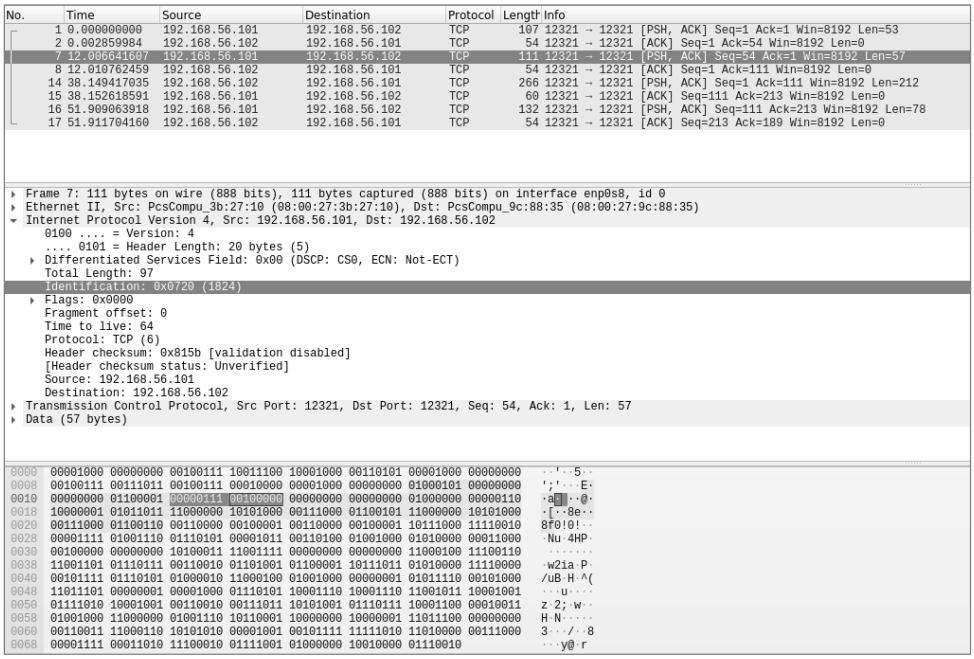
When listening for connections, Stegop2py filters out incoming SYN packets from the rest of the traffic. As soon as a host attempts to connect, the 3-way-handshake is handled. PICTURE 8 shows the packet capture of a successful 3-way-handshake. The connecting host initiates the handshake by sending an SYN packet with the SYN flag set and providing their ISN; the other host replies with an acknowledgement also initiating the connection on their side by setting the SYN flag and providing their ISN. Finally, the first host replies with an acknowledgement of its own. At this point, it is safe to say

that both hosts are connected. Once stgeop2py is connected, having “negotiated” keys with the other host, it can start sending embedded encrypted messages.



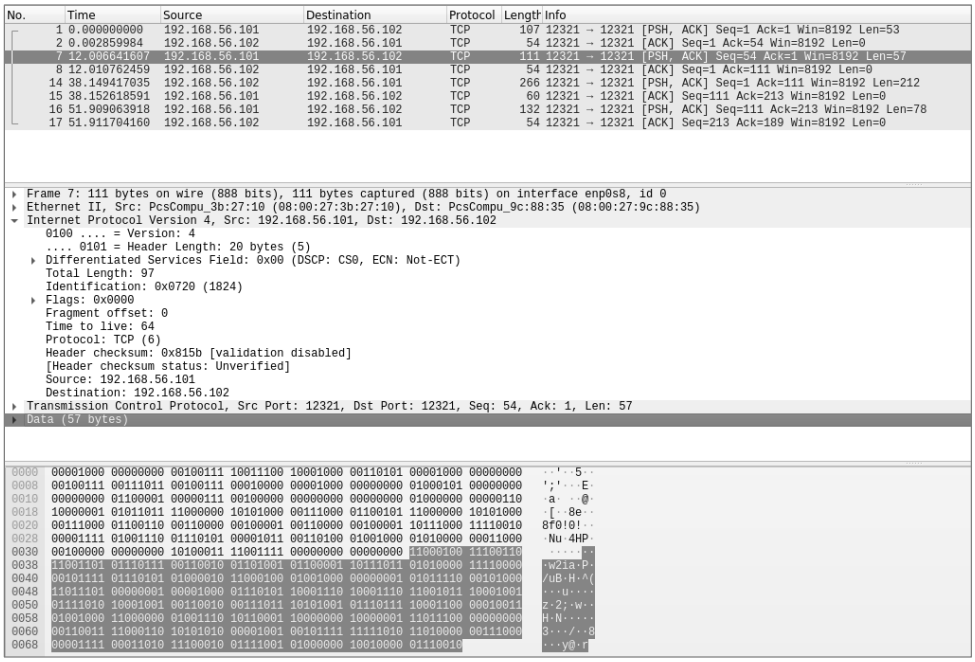
PICTURE 8. Wireshark packet capture showing 3-way-handshake. Note the sequence (Seq) and acknowledgement (Ack) numbers, as well as source and destination IP addresses

The messages that the user inputs in the console window will be encrypted with the provided password. The ciphertext will then be embedded into an arbitrary number of bytes between 2 to 3 times the size of the input message, plus the session’s previously generated data offset, to randomize the total payload length but ensure that the message can be embedded. In this phase, Stegop2py acknowledges every incoming packet as per TCP protocol. PICTURE 8 shows the packet capture of the session mentioned in the previous subheading.



PICTURE 9. Wireshark packet capture showing two hosts communicating with Stegop2py; the IP Identification field for one message is highlighted.

When inspecting a packet captured from the conversation, it is possible to analyze the value of the IP identification field. For the other host to be able to extract the message from the payload, apart from the data offset, the message length must also be known. Stegop2py embeds this information in the lowest significant byte of the IP identification field. PICTURE 9 shows that the message length is $100000_2 = 32_{10}$. PICTURE 10 shows that the payload size is 57 bytes; the 32-byte message is embedded in the 57-byte payload at a defined offset from the beginning. In the bottom panel in PICTURE 10 the highlighted data corresponds to the payload; it is possible to see that the ASCII conversion to the right does not convert to legitimate text, i.e., the encryption allows the message to blend in seamlessly.



PICTURE 10. Wireshark packet capture showing two hosts communicating with Stegop2py; the payload for one message is highlighted.

6 CONCLUSION AND FURTHER DEVELOPMENT

Thanks to the advancements of communication technology and the nature of communication, it is possible to establish communication channels with virtually anything, and due to this, anybody can eavesdrop on these channels given the field and tools to do so. Privacy is something that everyone should strive for, be it from merchants that track your data, or from anyone for that matter. Cryptography provides a way to make the contents of a message unintelligible, while steganography completely hides the presence of a message within another, innocent one. Steganographic techniques have been around for centuries; in the second chapter, a few examples of physical and digital applications of information hiding are presented. With these examples, it is understood that anything can be a covert carrier. A vast amount of research effort has been put into cryptography, and steganographic techniques may indeed seem impractical in comparison, but research in steganography for security may follow the same path as cryptography; there already is lots of interest in information hiding as an as art.

As a primary goal, this thesis aims to provide the fundamental knowledge needed to understand steganography as an art and to present an information-theoretic model for stegosystems with current research that helps to measure the usefulness of stegosystems to its users, as well as the performance on detection of a passive opponent in such a system. This is done in the hopes that the reader understands the wide reach and domain that steganographic techniques can be applied to, and possible attacks and countermeasures to prevent the creation of subliminal channels. As a secondary goal, stegop2py is presented as a proof-of-concept, showcasing the possibilities of embedding messages in TCP/IP packets and creating stegosystems in network environments.

The vast number of applications and uses of interconnected computer systems gives way to a vast number of protocols that are employed; these protocols can be manipulated in different ways to achieve covert communication. The TCP/IP protocol stack is a very interesting field for studying possible steganographic techniques that can be applied, as they are the most widely used.

Stegop2py allows two users to communicate over networks by exploiting the headers that are appended to data before sending. The encrypted messages are embedded into randomness to mimic encrypted data, similarly to that of communication over TLS. For embedding and extracting messages, the stegosystem requires a set of stego-keys: the offset where the data being sent starts (within the randomness), and the length of the actual message. Both keys are transferred to the other party, via steganography. The static

per-session offset is embedded into the lowest significant byte of the initial TCP sequence number field (ISN), and the length is likewise embedded into the lowest significant byte of the IP Identification field. With the packet capture of the session shown in the previous chapter, it is possible to see the initial 3-way TCP handshake with the first three packets, where the two clients negotiate their ISNs with their randomly chosen offset value where the plaintext (from the steganographic point of view) is embedded. Once the 3-way handshake is complete, the two clients are "connected" and are ready to send and receive messages. It is also possible to see the packets where the messages are sent and, following the TCP protocol, their corresponding acknowledgements. It is worth noting that the length of the packets varies according to the length of the messages; this proved to be the reason for which the TCP sequence number field was employed to embed the offset rather than the message length: the TCP sequence number can only be manipulated once, with the ISN, given that the sequence number grows according to the message length. This is visible when observing the (relative) sequence numbers.

This paper provides a proof-of-concept for creating subliminal channels, as depicted by Simmons (1984), in a slim area of network environments. The underlying protocols that help internet-connected devices around the world to communicate in the same "language" are a viable field for embedding messages. The constant development of these protocols indeed helps to avoid their exploitation, but new methods can still be developed; it can be safely said that to avoid covert communication channels to be built upon these protocols in their entirety, the ability to connect to the internet must be taken away.

Murdoch and Lewis (2005), provide great research on different applications that have been developed that exploit the fields in IP and TCP headers. The authors also develop an implementation of ISN generation for using them as steganographic carriers that is resistant to detection, based on the actual implementations present in the OpenBSD and Linux operating systems. The versions that were employed, though, are not current anymore, therefore some research can be put into the review of those methods against the current ISN generation implementations.

In the future, it would be interesting to further research into other important protocols present in the TCP/IP stack, like the Internet Control Message Protocol (ICMP). Likewise, research must be put into techniques for embedding messages (and their feasibility) into protocols that are higher in the OSI model, like HTTP, Web-Sockets, SSH, and others, as well as protocols lower in the OSI model, like Ethernet and IEEE 802.11 (Wi-Fi), and other less popular ones.

Moving away from communication networks, it would also be interesting to investigate the possibilities of embedding messages into protocols that employ blockchain technology, like Bitcoin and Ethereum, to study the use of such stegosystems given their rise in popularity in recent years.

With this knowledge, different stegosystems can be built to further prove these concepts with a tangible and testable product in the future. In the same manner, attacks can be developed to prevent these subliminal channels where it is deemed necessary. In the same manner, countermeasures to these attacks can also be studied to understand the weaknesses in different systems. In addition to all this, the ethics of creating and using stegosystems can be assessed in-depth, as the concept of steganography and its applications can be topics of great controversy.

Regarding the proof-of-concept, stegop2py, it is imperative to improve the program to make the forged packages truly indistinguishable from legitimate ones, for any real-world usage. This implies enhancing the method of the ISN generation, adding more TCP protocol functionalities such as finalizing connections gracefully (via FIN segments), adding a TLS protocol implementation to better mimic encrypted data transfer, improving error handling, and finding more possibilities for embedding plaintexts and sending (or deriving) any required stego-keys. Finally, a study on the efficiency and security of such a stegosystem to measure the general usability would be of benefit.

REFERENCES

Bredhoff, Stacey, ed. 2006. *Eyewitness: American Originals from the National Archives.* London, England: Philip Wilson.

Cachin, Christian. 2004. "An information-theoretic model for steganography." *Information and Computation* 192 (1): 41-56. doi:10.1016/j.ic.2004.02.003.

Carbonero y Sol, León. 1880. *Indice de libros prohibidos mandado a publicar por su santidad el papa Pio IX [Index of forbidden books ordered to be published by His Holiness Pope Pius IX]*. Madrid: Impr. de D.A. Perez Dubrull. https://archive.org/details/indicedelibrosp00solgoog.

Cox, Ingemar J., Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker. 2007. *Digital Watermarking and Steganography.* Burlington, Massachusetts: Elsevier Science & Technology. https://ebookcentral.proquest.com.

Ferguson, Niels, Bruce Schneier, and Tadayoshi Kohno. 2010. *Cryptography Engineering: Design Principles and Practical Applications.* Indianapolis, Indiana: Wiley Publishing, Inc.

Gont, F., and S. Bellovin. 2012. "Defending against Sequence Number Attacks." *Request for Comments.* RFC Editor, February. doi:10.17487/RFC6528.

ITU-T. 1994. "Recommendation X.200. Information technology - Open Systems Interconnection - Basic Reference Model: The basic model." https://www.itu.int/rec/T-REC-X.200-199407-I/en.

Johnson, Neil F., Zoran Duric, and Sushil Jajodia. 2001. *INFORMATION HIDING: Steganography and Watermarking - Attacks and Countermeasures.* Norwell, Massachusetts: Kluwer Academic Publishers.

Kahn, David. 1968. *THE CODEBREAKERS: The Story of Secret Writing.* New York: The Macmillan Company. https://archive.org/details/B-001-001-264.

Kipper, Gregory. 2003. *Investigator's Guide to Steganography.* Philadelphia, PA: Auerbach Publications.

Konakhovich, G. F., and A. Yu. Puzyrenko. 2006. *Kompyuternaya steganografiya. Teoriya y praktika [Computer steganography: Theory and practice]*. Kyiv: MK-Press.

MacKay, David. 2003. *Information Theory, Inference, and Learning Algorithms.* Cambridge, UK: Cambridge University Press.

Murdoch, Steven J., and Stephen Lewis. 2005. "Embedding Covert Channels into TCP/IP." Edited by Mauro Barni, Jordi Herrera-Joancomartí and Stefan Katzenbeisser. *Information Hiding.* Berlin, Heidelberg: Springer Berlin Heidelberg. 247-261. doi:10.1007/11558859_19.

Petitcolas, Fabien A. P., Ross Anderson, and Markus G. Kuhn. 1999. "Information Hiding - A Survey." *Proceedings of the IEEE* 87: 1062-1078. doi:10.1109/5.771065.

Postel, Jon, ed. 1981. "Internet Protocol." *Request for Comments.* RFC Editor, September. doi:10.17487/RFC0791.

Postel, Jon, ed. 1981. "Transmission Control Protocol." *Request for Comments.* RFC Editor, September. doi:10.17487/RFC0793.

Reeds, Jim. 1998. *Solved: The Ciphers in Book III of Trithemius's Steganographia.* Florham Park, New Jersey: AT&T Labs - Research.

Simmons, Gustavus J. 1984. "The Prisoners' Problem and the Subliminal Channel." In *Advances in Cryptology: Proceedings of Crypto 83*, by David Chaum, 51-67. Boston, MA: Springer US.

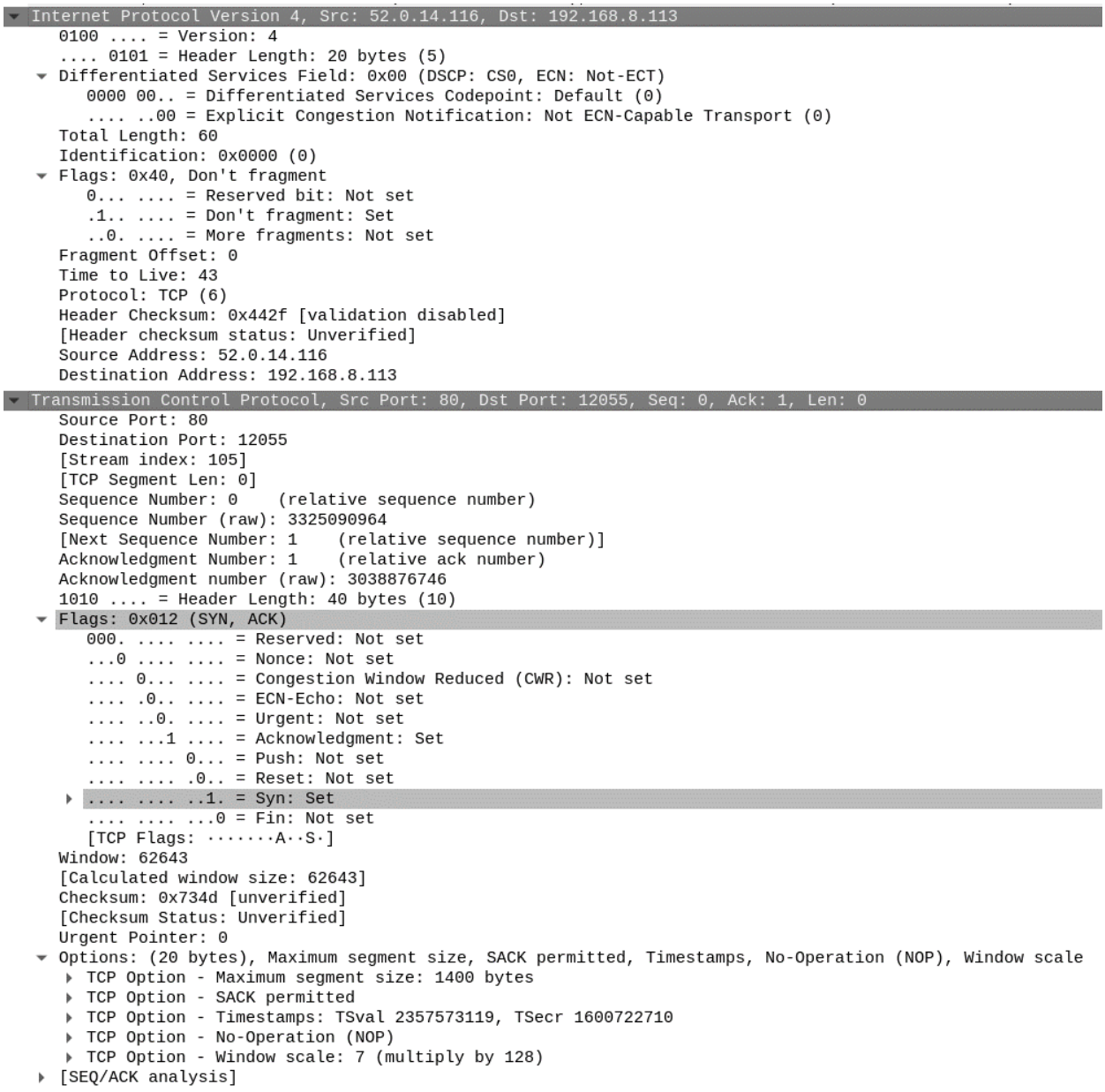
Tanenbaum, Andrew S., and David J. Wetherall. 2010. *Computer networks.* 5. Upper Saddle River, NJ: Pearson.

Trithemius, Johannes. 1499. *Steganographia.*

Zimmermann, Hubert. 1980. “OS1 Reference Model - The IS0 Model of Architecture for Open Systems Interconnection.” *IEEE TRANSACTIONS ON COMMUNICATION* COM-28 (4): 425-432.

APPENDIX 1

Wireshark screen capture showing the fields of the IP and TCP headers of a captured packet during a TCP conversation.



APPENDIX 2

Class diagram representing the classes present in Stegop2py and their relationship. The member variables and functions and their parameters are shown.