A study in how to inject steganographic data into videos in a sturdy and non-intrusive manner

En studie i hur steganografisk data kan injiceras i videor på ett robust och icke-påträngande sätt

JULIUS ANDERSSON

DAVID ENGSTRÖM
A study in how to inject steganographic data into videos in a sturdy and non-intrusive manner

En studie i hur steganografisk data kan injiceras i videor på ett robust och icke-påträngande sätt

Julius Andersson
David Engström

Examensarbete inom Datateknik/Elektroteknik, Grundnivå, 15 hp
Handledare på KTH: Jonas Wåhslén
Examinator: Ibrahim Orhan
TRITA-CBH-GRU-2019:027

KTH
Skolan för kemi, bioteknologi och hälsa
141 52 Huddinge, Sverige
Abstract

It is desirable for companies to be able to hide data inside videos to be able to find the source of any unauthorised sharing of a video. The hidden data (the payload) should damage the original data (the cover) by an as small amount as possible while also making it hard to remove the payload without also severely damaging the cover. It was determined that the most appropriate place to hide data in a video was in the visual information, so the cover is an image. Two injection methods were developed and three methods for attacking the payload. One injection method changes the pixel values of an image directly to hide the payload and the other transforms the image to cosine waves that represented the image and it then changes those cosine waves to hide the payload. Attacks were developed to test how hard it was to remove the hidden data. The methods for attacking the payload where to add and remove a random value to each pixel, to set all bits of a certain importance to 1 or to compress the image with JPEG. The result of the study was that the method that changed the image directly was significantly faster than the method that transformed the image and it had a capacity for a larger payload. The injection methods protected the payload differently well against the various attacks so which method that was the best in that regard depends on the type of attack.

Keywords
Steganography, Steganalysis, Watermarking, Information hiding, Cryptography, Compression, Non-intrusive, Sturdy, JPEG
Sammanfattning

Det är önskvärt för företag att kunna gömma data i videor så att de kan hitta källorna till obehörig delning av en video. Den data som göms borde skada den ursprungliga datan så lite som möjligt medans det också är så svårt som möjligt att radera den gömda datan utan att den ursprungliga datan skadas mycket. Studien kom fram till att det bästa stället att gömma data i videoar är i den visuella delen så datan göms i bilderna i videoen. Två metoder skapades för att injektera gömd data och tre skapades för att förstöra den gömda datan. En injektionsmetod ändrar pixelvärdet av bilden direkt för att gömma datat medan den andra transformerar bilden till cosinusvågor och ändrar sedan de vågorna för att gömma datat. Attacker utformades för att testa hur svårt det var att förstöra den gömda datan. Metoderna för att attackera den gömda datan var att lägga till eller ta bort ett slumpmässigt värde från varje pixel, att sätta varje bit av en särskild nivå till 1 och att komprimera bilden med JPEG. Resultatet av studien var att metoden som ändrade bilden direkt var mycket snabbare än metoden som först transformerade bilden och den hade också plats för mer gömd data. Injektionsmetoderna var olika bra på att skydda den gömda datan mot de olika attackerna så vilken metod som var bäst i den aspekten beror på vilken typ av attack som används.

Nyckelord
Steganografi, Steganalys, Vattenmärkning, Kryptografi, Kompression, Icke störande, Stabil, JPEG
Acknowledgements

This paper is a result of a thesis project at the Royal Institute of Technology within the field of computer engineering on behalf of June AB.

We want to thank everyone in June and especially Zoran Tolic, our mentor at June for his guidance and support. We would also like to thank our mentor at KTH Jonas Wåhslén for his support and suggestions. Last we want to thank Boris Asadanin & Daniel Eriksson at Eyevinn for a detailed overview of the subject.
## Glossary

<table>
<thead>
<tr>
<th>Word</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover Image</td>
<td>An image in which data is embedded</td>
</tr>
<tr>
<td>Payload</td>
<td>The data that is embedded into the Cover Image.</td>
</tr>
<tr>
<td>RGB</td>
<td>A method for representing pixels. It uses three numbers to describe how red, green and blue a pixel is.</td>
</tr>
<tr>
<td>Bit</td>
<td>The smallest measurement of information possible. Can represent two different states, for example 1 or 0, true or false, up or down.</td>
</tr>
<tr>
<td>Bit plane</td>
<td>All the bits in an image with the same amount of significance.</td>
</tr>
<tr>
<td>Sensor</td>
<td>A device that detects some measurement. A camera is the sensor which is often used to record visual information.</td>
</tr>
<tr>
<td>JPEG</td>
<td>JPEG is a standard developed by the Joint Photographic Experts Group. It describes how images can be stored and compressed.</td>
</tr>
</tbody>
</table>
1 Introduction

People can nowadays easily share files with each other due to the Internet's ability to connect large numbers of people. The Internet has many positive benefits, but as with most things, it also comes with drawbacks. One such drawback is that videos can easily be spread around the world without the approval of its owners, and without the owners knowing who is spreading the videos illegally. This is a big concern for the movie industry because it can lead to economic loss for video producers. It is, therefore, a big challenge to try and protect the movies from illegally sharing.

1.1 Problem specification

Digital watermarking is a method for solving problems that involve the unauthorised spread of data. Watermarking achieves this by firstly injecting a pattern into the data being shared, enabling to tie the shared video instance to a specific source i.e. a person who has bought and downloaded a video from the video producer. The data being shared can then be automatically tested for the presence of the pattern. If the pattern is found and the data is not shared with the consent of the owner then action can be taken to stop the unauthorised sharing. It may be desirable to convey more information than merely the presence of a pattern, such as which of the distributors of a video was the one a specific copy came from and in extension who is legally responsible for the unauthorised sharing, and that would then be considered steganography. This study aims to explain, measure and discuss various watermarking and steganography methods that exist for video. Additionally, the study also involves creating methods that detect and delete information previously hidden by watermarking and steganography techniques.

1.2 Goals

A more detailed description of the study's goals can be seen below.

1. Gather information
   ○ Determine in what or which mediums of the video it would be optimal to conceal the data.
   ○ Find appropriate steganography and watermarking techniques for the chosen medium.
   ○ Find the methods that are commonly used for detecting and erasing the hidden data.

2. Implement the chosen techniques
   ○ Create or use already existing programs that implement the methods chosen in step 1.
   ○ Setup ways to measure the performance of the steganography/watermarking techniques when used against the methods that find and erase the hidden data. The main measurements will be the amount of quality that the implementation must sacrifice to hide/remove the data.
3. Analyse and improve the system
   ○ Use the test results to optimize the performance of the prototypes.
   ○ Discuss and analyse the results of the study and determine how it can improve and what future work can be done in the area.

1.3 Constraints
The images extracted from videos that are used to hide data will be created by sensors rather than being synthesized by a computer, i.e. video from cameras rather than animated footage.

The quality loss that is incurred when the image is changed will be measured by objective value changes rather than the subjective perception influenced by the human eye and mind. The fact that humans perceive certain colors more than others, could, for example, be used to hide changes to an image more effectively, but that will not be investigated in this study.
2 Background and Theory

This chapter aims to explain how videos are represented by computers and how it is possible to hide data in them. It will also explain how images are stored and how it is possible to hide data in them as the videos are made up of a series of images and those were deemed the most appropriate place to hide data in videos. An overview of the methods that hide data in images will then be given alongside the methods for removing this data. This chapter will also examine how cryptology and compression are related to the injection and deletion of hidden data.

2.1 Theoretical background

The Internet has had a big success connecting people to each other and making it easy to share files. The digital world has evolved and continues to do so and together with the Internet's success, copyright protection is becoming more and more important. This is because digital data is very simple to copy and to share without loss of quality[1]. The paper “Detecting illegal file sharing in Peer-to-Peer networks using fuzzy queries”[2] claims that the economic loss every year due to piracy is 12.5 billion dollars. There could also be much information not shared with the public because of the risk of illegal sharing. As can be seen, a solution to this problem could contribute positively to our society.

The authors also believe this study can help in achieving the following goals in the United Nations sustainable development goals:

11. Sustainable cities and communities - Sharing files without the approval of its owners can harm the sustainability of society. This study aims to produce a solution to this problem.

13. Climate action - By evaluating the energy needs of the solutions, those who use one of them have the opportunity to use what affects the climate the least.

2.2 Digital storage of videos

A video is made up out of a series of image frames, an audio track and some metadata. The images almost always make up a large majority of the data in the file and it is therefore also the most appropriate place to hide data[3]. The different frames types that a video can consist of are I, B and P-frames. I-frames are simply normal images, B and P-frames dictates what should be shown by describing what changes have been made from the previous or the next I-frame. I-frames are the only necessary part of these as B and P-frames are dependent upon I-frames while I-frames can be viewed independently. I-frames also uses the largest amount of storage space in comparison with B and P-frames. I-frames are therefore the best frames to hide data in. The large amount of storage space leads to the biggest chance of unnecessary details in which data can be hidden. The necessity of the I-frames also makes it so that anybody trying to remove the hidden data cannot simply remove the frames as the could have done with B or P-frames. This means that good video hiding techniques are also image hiding techniques as an I-frame is practically the same thing as an image[3].
2.3 Digital storage of images
A common way to store images in computers is as a grid of pixels. Every pixel in an image is made up of three numbers that describe how that pixel should be displayed. There are many conventions for what the numbers represent, but the most common one is to have the numbers describe how red, green and blue the pixel should be, this method is called RGB. Every number is made up of a number of bits where more bits allows the color to be represented with more precision. Every additional bit describes the value by half as much of the previous bit, in the same way that every additional character in a number is worth a tenth of the previous one when using the normal base 10 number representation system. Images with many bits per pixel can, therefore, have several bits whose effect on the image is almost invisible to a human[4]. The bit that has the biggest impact on the pixel’s color is called the most significant bit (MSB) and the bit that has the smallest impact is called the least significant bit (LSB).

An image stored as pixels can retain all of its details which makes it a lossless storage method. The problem with pixel storage is that it requires a large amount of space which caused other storage methods to be developed. A form of lossy storage, i.e. some details of the image is changed, is to simply not store some of the LSBs and thus making the image require less space. There are however more efficient image storage methods that make fewer changes to the image and also require even less storage space. These more efficient methods often function by transforming the image into different domains. The two most common methods used to transform images are the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT)[5]. An example the efficiency which can be gained can be seen in JPEG, an implementation of DCT, which is able to compress an image to 10 % of its original size with little perceptible loss in image quality[6].

At the very base level, there are two ways in which a digital image can be created. They can be synthesized by a computer or acquired through a sensor[5]. Computer generated images are generally created with a set of deterministic rules. Data that has been hidden in a computer generated image could be found by a program that investigated whether the rules could create the altered image. Any technique for finding hidden data that uses the specific rules of an image generating program would only be useful for images from that program[5] This is the reason for the constraint in chapter 1.3 that states that only images generated by sensors would be investigated in this study.

2.3.1 Discrete cosine transform
Discrete cosine transformation is a way to transform an image from the spatial domain to the frequency domain[7], it stores images as a combination of different cosine waves with weights. The image is split into small squares which are 8x8 pixels in size, padding is added if necessary to make the image divisible by 8 in both directions. Every square is then converted into numbers describing how similar the squares are to a set of predetermined cosine waves. JPEG, for example, uses this transformation.
A visual representation of the waves can be seen in Figure 2.1 and the result of example transforms can be seen in Table 2.1 and 2.2. Waves near the top left describe the broad strokes of the square while those near the bottom right describes the fidelity and details. The waves in the bottom right generally have a low absolute weight in images created by cameras, Table 2.2 shows an example of this. The weight difference alongside the fact that humans have difficulty perceiving differences in small details means that those waves can be considered superfluous. This method uses the same amount of storage space to store the same data but the important and unimportant parts of the data are now concentrated in different locations, the top left and the bottom right waves respectively. A common compression approach is therefore to remove the weight of a number of the waves close to the bottom right[5].

![Figure 2.1](image.png)

Figure 2.1[8] - The 64 waves the JPEG implementation of DCT uses visualized as squares.

Table 2.1 - The left square is the RGB representation and the right is the DCT representation. The RGB square is completely filled with the relevant color is therefore described perfectly by the top left cosine “wave”. The “wave” is a flat line which is described as a cosine wave with a frequency of 0.

<table>
<thead>
<tr>
<th>255</th>
<th>255</th>
<th>255</th>
<th>255</th>
<th>255</th>
<th>255</th>
<th>255</th>
<th>255</th>
<th>1024</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2.2 - The left square is the RGB representation and the right is the DCT representation. The RGB square is taken from a typical photo. Note the high absolute weight of the waves near the top left and the low absolute weight of the waves near the bottom right.

<table>
<thead>
<tr>
<th></th>
<th>62</th>
<th>55</th>
<th>55</th>
<th>49</th>
<th>49</th>
<th>47</th>
<th>55</th>
<th></th>
<th>-370</th>
<th>-29.7</th>
<th>-2.6</th>
<th>-2.5</th>
<th>-1.1</th>
<th>-3.7</th>
<th>-1.5</th>
<th>-0.08</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>57</td>
<td>54</td>
<td>52</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>53</td>
<td></td>
<td>-231</td>
<td>44.9</td>
<td>24.5</td>
<td>-0.3</td>
<td>9.3</td>
<td>3.9</td>
<td>4.3</td>
<td>-1.4</td>
</tr>
<tr>
<td>61</td>
<td>60</td>
<td>52</td>
<td>49</td>
<td>48</td>
<td>48</td>
<td>49</td>
<td>54</td>
<td></td>
<td>32.8</td>
<td>8.5</td>
<td>-7.6</td>
<td>-2.7</td>
<td>0.3</td>
<td>-0.4</td>
<td>0.5</td>
<td>-0.6</td>
</tr>
<tr>
<td>63</td>
<td>61</td>
<td>60</td>
<td>60</td>
<td>63</td>
<td>63</td>
<td>68</td>
<td>65</td>
<td></td>
<td>12.5</td>
<td>14.6</td>
<td>-3.5</td>
<td>-3.4</td>
<td>2.4</td>
<td>-1.3</td>
<td>2.7</td>
<td>-0.4</td>
</tr>
<tr>
<td>67</td>
<td>67</td>
<td>70</td>
<td>74</td>
<td>79</td>
<td>79</td>
<td>91</td>
<td>92</td>
<td></td>
<td>-4.9</td>
<td>-3.9</td>
<td>0.9</td>
<td>3.6</td>
<td>0.1</td>
<td>5.1</td>
<td>1.1</td>
<td>0.5</td>
</tr>
<tr>
<td>82</td>
<td>95</td>
<td>101</td>
<td>106</td>
<td>114</td>
<td>114</td>
<td>112</td>
<td>117</td>
<td></td>
<td>-0.5</td>
<td>3.1</td>
<td>-1.4</td>
<td>0.2</td>
<td>-1.1</td>
<td>-1.5</td>
<td>-1.1</td>
<td>0.9</td>
</tr>
<tr>
<td>96</td>
<td>111</td>
<td>115</td>
<td>119</td>
<td>128</td>
<td>128</td>
<td>130</td>
<td>127</td>
<td></td>
<td>4.4</td>
<td>2.3</td>
<td>-1.7</td>
<td>-1.6</td>
<td>1.1</td>
<td>-2.7</td>
<td>1.1</td>
<td>-1.4</td>
</tr>
<tr>
<td>109</td>
<td>121</td>
<td>127</td>
<td>133</td>
<td>139</td>
<td>141</td>
<td>140</td>
<td>133</td>
<td></td>
<td>-10.2</td>
<td>-1.8</td>
<td>5.9</td>
<td>-0.4</td>
<td>0.3</td>
<td>0.4</td>
<td>-1.0</td>
<td></td>
</tr>
</tbody>
</table>

2.4 Steganography

The three major scientific fields of message security are cryptography, steganography and watermarking, their relation to each other can be seen in Figure 2.2. A message is any collection of data sent between two parties. Cryptography hides the meaning of a message but not the message itself. Steganography hides the existence of a message behind another message. Watermarking adds a message to another message. The line between steganography and watermarking is at times unclear or subjective[3]. Some of the methods in this study could be considered a form of robust imperceptible watermarking rather than steganography, the difference could be decided by the precise implementation of the methods.

---

Figure 2.2 [3] - The different disciplines of message security. The arrow indicates an extension and the boldface indicates the focus of this study.
This study will revolve around steganography in the digital images that make up a video (as described in chapter 2.3), it will likely have a large overlap with steganography in other digital images. The image that is used to carry the hidden data will in this paper be referred to as the cover image. The data that is being concealed will be referred to as the payload.

Steganography techniques are classified into three distinct groups[9].

- The spatial domain technique is where you hide the payload by inserting it into the cover image by modifying the pixel values. This technique is often less complex to implement and has a high capacity for the size of the data being injected. One common method that utilizes this technique is the least significant bit substitution method.

- The transform domain technique uses various algorithms and transformations on the cover image to hide the payload inside the result of the transform. The idea is to hide information in areas in the image where it does not get exposed to compression and cropping, which otherwise would have harmed the payload. The transform domain coefficients are then used to encode the messages instead of changing the pixel values as described above in the spatial domain technique. It is usually complex to implement.

- The fractal domain technique is where you hide the payload by identifying patterns similar to the payload in an image. It has a low capacity and is highly computationally expensive. Many images might lack these patterns and therefore this technique is not suitable for general use.

2.4.1 Least significant bit substitution
A common approach to inserting data into images is to change the LSBs of the image to represent the hidden data instead of their original value. An example of the method in action can be seen in Figures 2.3, 2.4, and 2.5 which has had 100 000 random numbers between 0 and 10 000 injected into the image. Figure 2.3 is the original Image, 2.4 is the image with the LSBs altered and 2.5 is the image with the most significant bits altered. Altering the MSBs in Figure 2.5 is done only to illustrate the method is doing to the LSBs, it is not considered a reasonable steganographic method.
Figure 2.3 [10] - The unmodified cover image, included for comparison with Figure 2.4 and 2.5.

Figure 2.4 - The cover image with the payload injected into the least significant bits. The changes are almost undetectable by the human eye.
The least significant bit substitution method always stores the payload in a manner that is the least disruptive to the image quality of the cover image. An actor trying to prevent the transmission of hidden messages would have difficulties spotting the difference with the naked eye. Detection via computer programs would be easier but it is possible to render the payload undetectable by sacrificing a sufficient amount of information capacity (see chapter 2.5). The real weakness of LSB substitution is that any attempt to scrub the cover image of its payload would be likely to succeed with minimal effort. Not only would any intentional attempt probably succeed, but any changes to the image, like lossy compression or resizing, would also be likely to remove the payload[3], this weakness can be mitigated by using more significant bits.
2.4.2 Discrete cosine transform coefficient replacement

If the cover image is stored as DCT cosine waves (as described in chapter 2.3.1), then the weight of those waves can be altered to represent the payload. The waves are roughly ordered from most to least important for human perception, which enables a great granularity when choosing between the amount of quality lost contra the risk of payload erasion. This makes it easy to avoid damage from compression methods which implements DCT. The avoidance of the DCT compression can be achieved by simply finding to which cosine wave the compression method erases data and inject the payload into an earlier wave, this makes the payload immune to damage from the compression. Other types of data erasure methods will generally disrupt the bottom right waves from Figure 2.1 first and then disrupt earlier and earlier waves more as the changes to the cover image becomes larger and larger[5].

As described in chapter 2.3.1 the image is split into small squares which are 8x8 pixels in size. According to “Implementation of Image Steganography Algorithm”[11] the 8x8 pixel block can be divided into 3 areas (see Figure 2.6). There are 6 low-frequency coefficients, 22 coefficients in the mid-frequency range and 36 coefficients in the high frequency. The low-frequency coefficients are not suitable to be altered because human eyes can detect even small changes in low-frequencies in an image. This makes changes to those coefficients more noticeable but it also makes it likely that changes humans make to images will line up and make a disproportionate change to one of these coefficients. The very most top left coefficient is for example very unlikely to change much when the changes to the RGB image are random but if the image is brightened or darkened then that coefficients would be changed a lot. The high frequencies coefficients are highly volatile so any information hidden there is likely to be removed. The mid-frequency coefficients are instead a good middle ground and therefore the coefficient to modify should be chose from them[11].

![Figure 2.6][11] - The different areas of frequencies. $F_L =$ low frequency, $F_M =$ mid frequency, $F_H =$high frequency

---

[11]: Reference to the source of the information.
2.5 Steganalysis
Steganalysis is the scientific field that studies how steganographic data can be detected and deleted. Steganography data hidden with spatial domain techniques can be found by analysing the statistical properties of the image, e.g., first-order statistics (Histogram) or second order statistics (correlation between pixels, distance, direction). Data hidden with transform domain techniques can be found by using a JPEG double compression and then analysing the discrete cosine transform coefficients for irregularities[3].

Spatial steganography generates unusual patterns that steganalysis tools can pick up on. The patterns can be found in the sorting of color pallets, the relationship between indexed colors and/or an exaggeration of the “noisiness” of the LSBs. The LSB substitution method is considered to be “very fragile” and steganalysis, therefore, have an easy time scrubbing its payload. Any kind of filtering or manipulation of an image is very likely to destroy the payload. It is also possible to assign a new value to every pixel in the LSB plane with “very little change in the perceptual quality of the modified[5] ” but with certainty that the entire payload has been removed[3]. A LSB substitution program could disperse the payload over a larger area to hide these patterns, sacrificing part of its capacity to conceal the payload better. The paper “Reliable Detection of LSB Steganography in Color and Grayscale Images”[12] claims to be able to reliably detect the presence of hidden data in a message which has been altered by more than 0.03 bytes per pixel, so a program would have to sacrifice more than 92% of its capacity (3 colors in every pixel, \( \frac{3}{8} \times (1 - X) = 0.03 \), \( X = 92\% \)) to avoid detection from similar methods.

The three steganalysis programs used in this study were the Steganography Obliterator, JPEG conversion and noise adding. The programs all have a setting that describes by how much they damage the cover image and this will be referred to as their strength setting in this paper.

2.5.1 Steganography Obliterator
This method is based on the program described in the paper “Steganography Obliterator: An Attack on the Least Significant Bits”[13]. The method simply sets all bits in a bit plane to 1. The strength setting of the obliterator dictates which of the bit planes is being scrubbed.

2.5.2 JPEG conversion
JPEG is an implementation of DCT and it is not created to be a steganalysis tool, but it can still function as one. A program that was intentionally created to remove hidden messages could presumably be better at that task but JPEG conversion is still often used. The major benefit of this method are that it is easy to acquire and use, it is not illegal or suspicious to own as some steganalysis methods can be and the deletion of the hidden data can be excused as an accident[5]. JPEG transformers the image to the .jpg file format and then compresses it. Its strength setting is the amount of compression that it performs where 100 is no compression and 0 is maximum compression.
2.5.3 Noise adding
This method is based on the program described in the paper “Image Watermark Attacks: Classification & Implementation” [14]. Noise adding is similar the obliterator method but instead adds or subtracts a random number from each pixel value. Its strength setting is the noise amplitude which is the maximum value for the random number.

2.6 Image quality damage measurement
It is necessary to have some methods for measuring how damaged an image has been by various operations and the measurement chosen in this study were the Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM).

MSE is a measurement method that is the most used in steganographic contexts to measure how likely it is for a computer program to detect changes to the image. The MSE is calculated by totaling the squared difference between every color channel of every pixel and then dividing it by the number of pixels. This method produces an error per pixel measurement where a higher value is considered worse [15]. The equation for the method can be seen in Equation (1).

\[
MSE = \frac{\sum_{MN} [I_1(m,n) - I_2(m,n)]^2}{MN}
\]  

(1)

PSNR is another way to measure the quality of an image. It is the ratio between the maximum power of a signal and the power of distorting noise that affects the representation. It contains no more information than MSE because of its definition, which can be seen in Equation (2), R is the highest color value a pixel can have, 255 for most pictures.

\[
PSNR = 10 \log_{10} \left( \frac{R^2}{MSE} \right)
\]  

(2)

SSIM is commonly used to measure the similarity between images. Its measure is correlated to the human visual system. The formula can be seen in Equation (3).

\[
SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{\mu_x^2 + \mu_y^2 + c_1 \sigma_{xy}^2 + c_2}
\]  

(3)

2.7 Compression and cryptography
The size of the payload is directly proportional to how easy it is to detect and how vulnerable it is to deletion. A larger payload necessitates a larger change to the cover image which is easier to detect and more pixels leads to a bigger chance that a part of them will be hit by random changes, either
from intentional steganalysis or from other changes to the image like lossy compression. It is because of this desirable to compress the payload before injecting it[5]. The danger of using compression is that it might make the payload take on a more ordered structure which is easier to predict and detect[5]. This potential drawback can be nullified by also encrypting the payload.

The message sender might want to use encryption to prevent the payload from being read even if it is found but there are also other benefits to encryption. A common feature of encryption is that the result appears to be random. Random data is easier to hide as there are no patterns or statistical anomalies that can be exploited to detect the payload. The payload is however still detectable as the cover image is presumably not random and thus still different from the payload, it just makes harder to detect. It can be assumed that the cover image is not random or encrypted as it would then be indistinguishable from an encrypted payload which would remove the need to hide anything. Encryption is likely to be helpful for all non-random payloads but it is especially important for compressed payloads as they tend to be more structured and that structure tends to be more well known which means that it is more often searched for[5].
3 Methodology

This chapter will describe the methods chosen to achieve the goals of the study and the reasoning behind them. A literature review was performed to determine which algorithms should be used for the steganography and steganalysis methods. Computer application prototypes were developed based on the chosen methods. A quantitative study was performed on the discrete cosine transform coefficient replacement method to determine which cosine coefficients were most appropriate to host the payload. Various measurements were developed to measure and test the steganography and steganalysis implementations.

The methods used in this thesis were:

1. Development of steganography prototypes.
2. Find steganalysis methods, implement prototypes or use existing implementations.
3. Test the steganography prototypes under the chosen attacks.
4. Analysis of data from all tests.

3.1 Compression and cryptography

While chapter 2.7 describes the desirability of compression and encryption of the payload, neither of these will be used when testing the implementations. Encryption is only helpful if any of the steganalysis methods are looking for patterns and none of the chosen methods do that. The reason that no steganalysis methods look for these patterns is that encryption already counters this approach perfectly so it was deemed unnecessary to investigate this steganography, steganalysis interaction further. Compression would improve the results of the steganography implementations but it was not used either. This is because the purpose of the test was to measure the implementations under a set of predictable and consistent circumstances rather than a set of optimal but variable circumstances. If the compression method reduced the size of the payload by different percentages for different payloads then the results could be skewed and the test would misrepresent the actual performance of the steganography and steganalysis methods.
3.2 Reasoning behind the choice of methods
Two steganography methods and three steganalysis methods were chosen and also 3 formulas for measuring the amount of quality degradation that is created when an image is changed.

3.2.1 Steganography
The three major methods that are used in steganography are the least significant bit (LSB) substitution, the discrete cosine transform (DCT) coefficient replacement, and the discrete wavelet transform (DWT) coefficient replacement[16]. This study chose to investigate the first two of these methods. DWT is considered to be similar DCT and they are both considered complex to implement and computationally demanding according to the paper “Analysis of techniques involving data hiding and watermarking”[9]. So DWT was not included as the implementation would be likely to require a large amount of time, it would have significantly increased the time required to perform tests and the result would be unlikely to be significantly different from DCT. DCT and DWT had many of the same drawbacks so the reason DCT was chosen over DWT is that JPEG implements a transformation that is similar to DCT and their interaction was deemed to be potentially interesting.

3.2.2 Steganalysis
Scientific papers that describe steganalysis implementation are fairly rare so both of the methods that were found that had scientific backing were implemented and tested, i.e. the steganography obliterator[13] and noise adding[14]. A fairly common concern in steganography is that compression will damage the payload[5] so another tool that was used for steganalysis was JPEG conversion.

3.2.3 Prototype evaluation measurements
The measurement used to determine the amount of damage the implementation did to the images was mean square error, peak signal to noise ratio and structural similarity as they are considered common measurements for determining the quality difference between two images[17]. The time of the programs was also measured as some steganography and steganalysis implementation can differ significantly in the amount of computational resources they require.
3.3 Steganography implementations
The chosen implementation of the steganography method was LSB substitution, which is a spatial
domain technique, and discrete cosine transform which instead is a transform domain technique.

3.3.1 Bit substitution method
The basis for this method was the code from the program that the user William_Wilson presented on
the forum dream in code\(^1\). Figure 2.3 and 2.4 shows the result of this method. The program
substitutes a bit of the top left pixel of the cover image with the first bit of the payload and the
second bit is substituted into the pixel on the right. This pattern continues until the entire payload
has been injected, if the rightmost edge of the image is reached then the next pixel is the leftmost
pixel one row down. The program then uses the more significant bits when all the bits in a bit plane
are substituted but it is unlikely that any ID is of such a size that it would not fit into a bit plane of
any image which was relevant to this project. The program indicates which bits are a part of the
payload by adding four bytes to the start of the payload which describes how many bytes long the
rest of the payload is. This solution limits the payload to 4 294 967 295 bytes (~4 GB) which was
once again deemed larger than any reasonable ID implementation and also larger than the entirety
of most images. The biggest possible payload size in bits an image can hold when being injected by
bit substitution is \((\text{Height of image}) \times (\text{Width of image}) \times (\text{Amount of colors (3 for RGB)})\).

3.3.2 DCT coefficient replacement method
The program divides an image into 8x8 blocks as described in chapter 2.3.1. Each pixel is
represented with RGB and thus separating each pixel into 3 values, every 8x8 pixel block will turn
into three 8x8 blocks. The discrete cosine transformation is then applied to every 8x8 block to
transform it from the spatial to the frequency domain. The new 8x8 block now consists of DCT
coefficients. Which coefficients and what bit of that coefficient that will be used can be dictated by
the user. The payload will be hidden in those coefficients and then transformed back to the spatial
domain to be able to view it as a regular image. The image can later be transformed back into the
frequency domain and the altered coefficients can then be read to reconstruct the payload. The
method was made to use the same four-bytes-for-payload-length solution as the bit substitution
method (BSM) to ensure parity when comparing methods. This once again limits the size of the
payload to 4 294 967 295 bytes (~4 GB). The size also likely more harshly limited by the amount of
8x8 squares in an image and the number of bits in coefficients that are used to represent the payload
in each square. The exact formula to calculate the maximum payload size in bits for this method is
\((\text{Height of image} / 8 \text{ rounded down}) \times (\text{Width of image} / 8 \text{ rounded down}) \times (\text{amount of bits used in}
each square}) \times (\text{Amount of colors (3 for RGB)})\). The code for the discrete cosine transform and its
inverse was taken from a GitHub repository written by Dr. Joren Six at IPEM, University Ghent\(^2\).

---

\(^1\) https://www.dreamincode.net/forums/topic/27950-steganography/

\(^2\) https://github.com/JorenSix/TarsosDSP/blob/master/src/core/be/tarsos/dsp/mfcc/DCT.java
3.4 Comparing prototypes

The images the tests were performed upon were all 1280 times 872 pixels. The pixels were stored as an RGB in the .png format and each of the colors was represented by 8 bits. All the tests were performed on 30 different cover images which were created using a cellphone camera. The payload used in the tests was made up of a byte repeated a variable number of times, half of the bits of the byte were 1 and the other half were 0. The payload size were 1000 bytes long in all the tests were the size is not explicitly stated to be otherwise.

The various steganography and steganalysis programs that are tested in this study were developed in Java. The Java virtual machine optimises some parts of the code after running it a certain amount of time so every program was run on an image as a warmup before the performance of the program was measured. The Java garbage collector was invoked between every measurement to avoid it being a source of time difference.
4 Results

The purpose of the first tests performed was to determine where the payload should be embedded in the DCT coefficients. Tests were performed to determine how much the steganographic methods distorted the cover image in relation to the size of the payload. It was also measured how much of the payload and the cover image was distorted when subjected to the steganalysis methods. Finally, the time requirement of the various methods was measured. All figures in this chapter are collected at the end of the chapter.

4.1 Comparing DCT coefficients

The DCT Coefficient Replacement Method (DCRM) requires one or more coefficients to be chosen as the ones to contain the payload. The amount the coefficients were changed when the cover image was altered by common steganalysis tools were measured to determine which coefficient was the safest to store the payload in. The percentage change of a coefficient is used to determine which coefficient is the most stable rather than the absolute change.

The amount the DCT coefficient changed in an image which was subjected to the steganographic obliterator described in chapter 2.5.1 and the JPEG conversion algorithm described in chapter 2.5.2 was measured. The values in Table 4.1 are the average of the result of the obliterator and JPEG attacks, the individual results can be seen in Appendix 3. Each block in Table 4.1 represent one coefficient. A visual representation of the waves corresponding to these coefficients can be seen in Figure 2.1. The table was used to determine which of the coefficients should be the hiding place for the DCRM. The coefficient chosen was the one with the lowest sum of the median and the 95% coefficient margin of the valid coefficients. The coloured coefficients are the ones considered valid, the basis for this can be read in chapter 2.4.2. This means that the chosen coefficient on average changes by ~27% of its original value when subjected to these two attacks and that its 95% confidence interval is between 11.77 % and 41.83 %.
Table 4.1 - The average of a Noise attack and a JPEG attack. The grey and green colored squares are those that a coefficient should be chosen from and the green colored one is the coefficient chosen. The first value is the mean and the second is by how much the 95% confidence interval extends around the mean.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.265</td>
<td>15.3</td>
<td>25.74</td>
<td>26.80</td>
<td>31.675</td>
<td>37.90</td>
<td>62.1</td>
<td>75.91</td>
</tr>
<tr>
<td>+/- 0.066</td>
<td>+/- 6.330</td>
<td>+/- 11.55</td>
<td>+/- 15.03</td>
<td>+/- 19.05</td>
<td>+/- 28.09</td>
<td>+/- 48.31</td>
<td>+/- 94.88</td>
</tr>
<tr>
<td>19.38</td>
<td>21.68</td>
<td>26.23</td>
<td>25.93</td>
<td>24.69</td>
<td>31.9</td>
<td>52.29</td>
<td>67.00</td>
</tr>
<tr>
<td>+/- 8.220</td>
<td>+/- 12.43</td>
<td>+/- 16.05</td>
<td>+/- 19.46</td>
<td>+/- 24.94</td>
<td>+/- 36.09</td>
<td>+/- 78.21</td>
<td>+/- 115.6</td>
</tr>
<tr>
<td>31.65</td>
<td>31.3</td>
<td>31.21</td>
<td>30.56</td>
<td>33.01</td>
<td>46.38</td>
<td>62.12</td>
<td>71.08</td>
</tr>
<tr>
<td>+/- 16.02</td>
<td>+/- 18.10</td>
<td>+/- 20.71</td>
<td>+/- 27.11</td>
<td>+/- 42.42</td>
<td>+/- 70.06</td>
<td>+/- 106.9</td>
<td>+/- 138.0</td>
</tr>
<tr>
<td>40.65</td>
<td>37.55</td>
<td>37.47</td>
<td>40.09</td>
<td>46.73</td>
<td>58.69</td>
<td>70.36</td>
<td>78.69</td>
</tr>
<tr>
<td>+/- 23.86</td>
<td>+/- 26.67</td>
<td>+/- 32.33</td>
<td>+/- 42.08</td>
<td>+/- 69.48</td>
<td>+/- 102.1</td>
<td>+/- 128.5</td>
<td>+/- 149.9</td>
</tr>
<tr>
<td>44.03</td>
<td>41.44</td>
<td>44.27</td>
<td>49.01</td>
<td>59.86</td>
<td>68.77</td>
<td>77.24</td>
<td>90.07</td>
</tr>
<tr>
<td>+/- 36.01</td>
<td>+/- 41.52</td>
<td>+/- 55.75</td>
<td>+/- 71.47</td>
<td>+/- 105.3</td>
<td>+/- 136.8</td>
<td>+/- 156.8</td>
<td>+/- 172.3</td>
</tr>
<tr>
<td>54.83</td>
<td>53.45</td>
<td>60.29</td>
<td>69.02</td>
<td>69.54</td>
<td>78.04</td>
<td>84.25</td>
<td>97.88</td>
</tr>
<tr>
<td>+/- 47.26</td>
<td>+/- 62.80</td>
<td>+/- 84.47</td>
<td>+/- 119.5</td>
<td>+/- 143.0</td>
<td>+/- 152.9</td>
<td>+/- 175.0</td>
<td>+/- 194.2</td>
</tr>
<tr>
<td>70.28</td>
<td>67.69</td>
<td>74.56</td>
<td>84.19</td>
<td>85.80</td>
<td>86.78</td>
<td>114.425</td>
<td>119.62</td>
</tr>
<tr>
<td>+/- 74.10</td>
<td>+/- 92.60</td>
<td>+/- 125.4</td>
<td>+/- 140.8</td>
<td>+/- 159.8</td>
<td>+/- 172.9</td>
<td>+/- 203.1</td>
<td>+/- 208.4</td>
</tr>
<tr>
<td>78.32</td>
<td>87.09</td>
<td>93.79</td>
<td>98.12</td>
<td>103.56</td>
<td>112.54</td>
<td>132.98</td>
<td>138.24</td>
</tr>
<tr>
<td>+/- 76.75</td>
<td>+/- 118.0</td>
<td>+/- 145.9</td>
<td>+/- 145.9</td>
<td>+/- 169.2</td>
<td>+/- 196.4</td>
<td>+/- 212.1</td>
<td>+/- 248.5</td>
</tr>
</tbody>
</table>

4.2 The steganographic programs

The amount by which the BSM and the DCRM altered the cover image were measured alongside the time these methods required. The DCRM was also tested for if and by what amount the rounding of its weights damaged the payload. It was decided that showing the MSE, PSNR, and SSIM for every test were surplus so the MSE is shown for every test and the PSNR or SSIM is added if those values were deemed interesting. The unused PSNR and SSIM values can be found in Appendix 1.

4.2.1 Quality tests

Figure 4.1 illustrates the amount of MSE that is created when the payload is injected using the BSM and the DCRM and Figure 4.2 illustrates the SSIM of the same test. Figure 4.3 also illustrates the amount of MSE created when injecting the payload but only for the BSM. The X-axis of Figure 4.3 has the scale 200:1 when compared to 4.1, the purpose of the zoomed out view is to show what happens when the BSM uses additional planes. These results show that the damage to the cover image grows linearly in relation to payload size and that the DCRM damages the cover image by a significant amount more than the BSM. The amount the payload is altered by the rounding of values in the DCRM can be seen in Figure 4.4.
4.2.2 Time tests
The amount of time required to inject a payload of a certain size was measured in accordance with the methods described chapter 3.4 and the results of these tests can be seen in Figure 4.5. The time requirement for the DCRM was larger than for the BSM and it appears to grow linearly in relation to payload size while the BSM’s time appears to be constant.

4.3 The steganalysis programs
The amount by which the various steganalysis methods altered the cover image were measured alongside the time these methods required.

4.3.1 Quality tests
Figure 4.6 describes the amount by which the cover image is altered when various bit levels of the image are removed by the steganography obliterator program described in chapter 2.5.1. This result is presented in table form due to the small number of data points and the large difference between the value of the data points. Figure 4.7 describes the amount by which the cover image is altered when the noise adding method is used for various amounts of noise and Figure 4.8 does the same for JPEG.

4.3.2 Time tests
The amount of time the steganalysis methods were measure, again in accordance with the methods described chapter 3.4. The results of these tests can be seen in Figure 4.9. The time requirement was determined to be independent of any strength setting used in the methods, i.e. the obliterator level, noise amplitude, and JPEG quality setting made no difference for the amount of time the methods required.

4.3 Steganography vs steganalysis
The final round of testing measured the performance of the steganographic methods when the cover image was subjected to a steganalysis method between the injection and the extraction of the payload. They were performed for various strength setting for the steganalysis methods which is represented as the X-axis of the graphs and for various bit levels of the steganographic methods which are represented by different lines in the graph and the percentage amount of the payload that was removed is represented by the Y-axis of the graph. For example, in the graph LSB vs Noise, a point in the line “Bit depth 5” with a noise amplitude of 10 and payload loss of 34 % should be read as: 34 % of the payload is removed when the noise adding method is used with a noise amplitude of 10 when the payload is injected in the bit which is 5 steps more important than the least significant bit.

The performance of the steganography methods against the steganalysis methods can be seen in Figure 4.10 - 4.14. The only pairing not measured was the BSM against the obliterator and the reasoning for this can be read in chapter 5.3.1.
Figure 4.1 - A graph showing the MSE in relation to the size of the payload for bit substitution and DCT injection. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.

Figure 4.2 - A graph showing the SSIM in relation to the size of the payload for bit substitution and DCT injection. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.
Figure 4.3 - A graph showing the MSE in relation to the size of the payload for bit substitution injection. Lines have been added to illustrate where a new bit plane is being used. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.

Figure 4.4 - A graph showing how much the payload is damaged by the rounding of weights in the DCT coefficient replacement method. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.
Figure 4.5 - A graph showing how much time an injection requires in relation to the size of the payload for bit substitution and DCT. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.

Figure 4.6 - The amount the cover image is altered when subjected to various levels of the steganography obliterator. The first value is the mean and the second is by how much the 95% confidence interval extends around the mean.
Figure 4.7 - A graph showing how much the MSE difference is between an image and the same image after it has been subjected to various levels of noise. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.

Figure 4.8 - A graph showing how much the MSE difference is between an image and the same image after it has been converted by JPEG and then compressed by the amount specified by the JPEG quality setting. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.
Figure 4.9 - The amount of time that is required for the three chosen steganalysis methods. The first value is the mean and the second is by how much the 95% confidence interval extends around the mean.
Figure 4.10 - A graph showing how much of the payload is altered when it was injected by the bit substitution method and attacked by the noise adding method. The line is the average of the value and the area inside the bars are the 95 % confidence interval of the value.

Figure 4.11 - A graph showing how much of the payload is altered when it was injected by the bit substitution method and transformed and compressed by JPEG. The line is the average of the value and the area inside the bars are the 95 % confidence interval of the value.
Figure 4.12 - A graph showing how much of the payload is altered when it was injected by the DCT coefficient replacement method and attacked by the noise adding method. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.

Figure 4.13 - A graph showing how much of the payload is altered when it was injected by the DCT coefficient replacement method and transformed and compressed by JPEG. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.
Figure 4.14 - A graph showing how much of the payload is altered when it was injected by the DCT coefficient replacement method and attacked by the obliterator method. The line is the average of the value and the area inside the bars are the 95% confidence interval of the value.
5 Analysis and Discussion

This chapter will analyse the result from chapter 4, discuss any potential limitations or error sources for those results and then discuss how this study could affect society at large.

5.1 Selection of coefficient
The location to hide the payload is straightforward to find in the BSM method which contrasts the DCRM. In chapter 4.1 the coefficient for DCRM was chosen because it was inside the favorable area described in chapter 2.4.2 and it was the least likely to be changed by the stenographic obliterator or by JPEG conversion. The effect of the noise adding method was not a part of the calculation because of its similarity to the obliterator, two similar methods would potentially have skewed the choice away from a coefficient appropriate for JPEG. Only one coefficient was chosen because it represents the best value achievable. The DCRM is, therefore, specialized against just the chosen attacks while BSM is not.

5.2 Steganographic prototypes analysis
In this chapter the prototypes will be analysed according to time, capacity, quality, and data loss created by rounding errors.

5.2.1 Time
The time difference between the methods can be seen in Figure 4.5. The BSM is much faster than the DCRM. In the BSM the value is altered directly which makes it fast while in the DCRM it is needed to apply the DCT to an 8x8 pixel block before altering the DCT coefficient. After altering the DCT coefficient the inverse DCT is applied on the 8x8 DCT block to be able to represent it back to pixel values. The transformation and its inverse are what make the DCRM slow. It might exist faster DCT and DCT inverse algorithms than the one implemented. For example the JPEG compressor performs a simplified version of its transformation which trades some accuracy for speed.

5.2.2 Data loss
The BSM can inject a payload and then extract it without any part of it being lost if no other method has altered the cover image. This is unfortunately not the case for the DCRM as seen in Figure 4.4. Transforming an image from the spatial to the frequency domain and vice versa using the DCT and its inverse can be done without losing information if allowing decimal values. A pixel being represented with the RGB system where each color can have an integer value between 0-255 and cannot be represented with a decimal value. Therefore decimal values must be rounded to the closest integer and there exist a risk that information is lost because of this rounding.
Figure 4.4 shows that there is a higher payload loss when using the bit with less significance than the bits with more significance to hide the payload. There will however always be some amount of risk of payload loss even when using the most significant bits. The next thing would then be to ask if there is a way to prevent this information loss. A way to solve this is to preemptively look if the information could be lost. If that is the case two approaches could be chosen.

1. The easiest way is just not to hide data there. This is a solution but this solution would decrease the maximum size of the payload.
2. The harder way is to modify the cover image such that no chance of losing information exists. This is a solution but would affect the quality of the image in a negative manner.

Further work must be done into how to determine if a risk exists such that information is lost to be able to solve the problem.

5.2.3 Quality
The authors expected that the MSE would be higher for the DCRM, higher being worse, than the BSM because changing one DCT coefficients can change several pixel values, this prediction was affirmed by the results. The authors, however, theorised that the SSIM might be higher for the DCRM, higher being better, than the BSM because its measure is correlated to the human visual system and the implemented DCRM uses a mid-frequency coefficient which should make a smaller difference visually, this was discovered to be incorrect. The related result is presented in Figure 4.1 and 4.2. The BSM was better than the DCRM in both regards. One explanation for this could be that the coefficient chosen is the one closest to the low-frequency coefficients. A coefficient from the higher frequencies (closer to the bottom right in Figure 2.1) would be more vulnerable to the steganalysis methods used but it might produce a higher SSIM and therefore be more pleasing to humans.

5.2.4 Capacity
The process for calculating the payload capacity for bit substitution and DCT is described in chapter 3.3. The images used in this study were all, as previously mentioned, 1280 times 872 pixels. The capacity is therefore 418560 byte for BSM and 6540 byte for the DCRM. The current implementation of the DCRM only uses one coefficient but this could be changed for increased payload capacity. Using many coefficients would affect the quality of the image more, presumably by about double the amount, it would also probably make it more than twice as vulnerable to steganalysis attacks as any additional coefficients would have to be a worse option than the first, assuming that the initial tests correctly identified the optimal coefficient choice.
5.3 Steganography vs steganalysis
In this chapter, DCRM and BSM will be compared when being attacked by obliterator, JPEG conversion, and noise adding.

5.3.1 Stenographic obliterator
The BSM was not tested against the stenographic obliterator because the results were computable by analysing the programs so practical tests were unnecessary. If the chosen bit in the BSM was different than the chosen bit in the obliterator attack the payload would be unaffected. If the bits chosen instead were the same the payload would be affected and all information would be lost. All bits of the message would not be altered as the ones with a value of 1 would remain 1 but it would be impossible to differentiate between the altered bits and unaltered.

The interesting part to test were instead the DCRM which result can be seen in Figure 4.14. The obliterator starts at bitplane 0 and then increases. The payload loss for all different bit levels used in the DCRM initially increases with the bit level affected but then decreases after somewhere between the first and fifth bit plane for all DCT bit levels. The authors theorize that the payload loss would continue to rise for an image with random values but the fact that pixel values for real images are often similar to its neighbors creates this effect. The similarity between neighbors means that the more significant bits representing the pixel are often the same in an entire 8x8 pixel block. Therefore when the obliterator is used on the more significant bit on those blocks it will either change most of the bits in that plane or keep most of them. If the bits are kept there are no changes to the pixel block and the payload will be kept. If the bits instead are changed then some part of the DCT coefficient must be changed but such a uniform change is much more likely to affect the low-frequency coefficients than the medium frequency coefficient that the payload is hidden in. The steganographic obliterator can be a simple way to remove the payload from the BSM but the DCT coefficient substitution method appears to be more robust.

5.3.2 JPEG conversion
In Figure 4.11 and 4.13, the result from the bit substitution and the DCRM when attacked by a JPEG conversion are shown. The BSM seems to perform poorly against this attack. When using the most significant bit it has a payload loss of about 42% and for the least significant bit, it has a payload loss of about 75% at JPEG quality setting 100. The only positive aspect is that it seems to not increase much when the JPEG quality is decreased. The DCRM, on the other hand, seems to handle the JPEG conversion much better with a near 0% payload loss for at a JPEG quality setting off 100 for most of the more significant bits and the payload loss for the DCMR constantly stays below the BSM payload loss for most values. In this test DCRM outperforms BSM. However, JPEG conversion is similar to DCT which could be an advantage of the DCMR. Nevertheless, JPEG is a very common compression method and is still an important test case.
5.3.3 Noise adding
When faced with a noise adding attack, the BSM (seen in Figure 4.10) outperforms the DCRM (seen in Figure 4.12) except in some instances when both methods payload loss is above 70%. This means that we consider the BSM to be the most resistant to noise adding, by a significant amount even, as it often had half of the payload loss of the DCT coefficient method.

The explanation to this might be that in the BSM every pixel gets affected separately and therefore it exists a probability for every pixel that its payload will be destroyed. DCRM instead divides the image into 8x8 pixel blocks. When adding noise to those 8x8 pixel block there is a high probability that the coefficients must be changed to describe this new image. This is because there are 64 pixels in each block and every one of them can affect the whole block. Because of the randomness of the noise adding there is a high risk that at least some of the 64 pixels will get affected which will lead to changes in the DCT coefficients and later might lead to payload loss.

5.4 Error sources
A potentially significant source of error in this study is the pictures the prototypes were tested on. They were all taken with the same camera and the motif of the images was arbitrarily chosen. It is possible another set of images would have produced a non-trivial change to the results. The coefficient chosen for the DCRM was chosen by analysing what the chosen steganalysis methods did and it is, therefore, custom fitted for those specific attacks. It is possible that other attacks might have been weaker to other choices of coefficients. The BSM is in contrast not fitted for the steganalysis methods and therefore probably more generally consistent.

5.5 Sustainable development
The DCRM is much slower than the BSM. The amount of energy needed to hide the payload will, therefore, be much higher for the DCRM. When hiding a payload in a large number of images there will be a significant difference in energy consumption between the two methods. Higher energy consumption has a negative impact on the environment. From an economic perspective it is also advantageously to use a method with less energy consumption.

5.6 Payload loss
Even with payload loss, there could be ways to retrieve the payload. Data recovery is a subject which creates methods to facilitate the recover data that has been damaged. An example of this could be to inject the payload multiple times into the same cover image. After the payload was damaged every part of the extracted message could be calculated by the mean of all copies of the message which could repair some of the damage. So a complete message could be extracted from a payload of multiple slightly damaged copies of the message. This method could benefit from a steganographic method with higher capacity. Further work has to be done into how much payload loss there can exist while being able to restore the payload.
6 Conclusions

This study has shown that it is possible to inject data into videos in such a way that the injected information is at least partially protected from methods commonly used to remove this information. The attacks could generally easily cause some damage to the injected data message but a data recovery technique could be used to mitigate this damage. The DCRM was more resistance to the obliterator and JPEG conversion while BSM was more resistance to noise adding. The DCRM is however much slower than the BSM, it damages the image much more, it has a much lower capacity and it risks damaging its own payload even in the absence of any attack. A positive aspect for the DCRM is that it has several clear opportunities for improvement with further research while the BSM is unlikely to be improvable. For both DCRM and BSM, the general result was that there was less payload loss when using higher bits than lower. However, the more significant bit used the poorer the image quality becomes as can be seen in Appendix 2. It, therefore, becomes a consideration between payload loss and quality loss.

6.1 Topics for further study

- This study chose to not optimize the data injection for computer-generated images, a possible extension to the study would, therefore, be to examine in which ways the prototypes fail for those images and to modify the prototypes to deal with these shortcomings.
- The aim of the prototypes is to make the injected data indistinguishable from the small imperfections created by cameras, it would therefore also be interesting to investigate if different cameras produce different imperfections and then develop steganography and steganalysis prototypes that exploit these differences.
- Another area of study would be to investigate what amount of extra payload capacity would be required to reconstruct messages that are damaged by different amounts.
- The heavy time requirement of the DCT transformation could probably be improved using a faster but imprecise calculation in a similar manner to JPEG. This could probably improve the DCT program a lot as JPEG is able to convert an image in less than a hundredth of the time of DCT with a loss of quality of less than 5 MSE.
- The coefficient chosen for the DCRM is the one most resistance to steganalysis but it still might still not be the optimal choice, a solution to this could be to test all the coefficients and investigate their effects. This is based on the discussion in chapter 5.2.3 that proposes that a coefficient with a lower frequency might damage the quality of cover images less. Any such tests should preferably be done after the DCT transformation is optimised or done with a large amount of computing power as the estimated time required to run tests for all coefficients with the computer used in this study would be very large.
References


Appendix

Appendix 1 - SSIM and PSNR

SSIM and PSNR for bit substitution injection with no steganalysis
SSIM and PSNR for DCT and bit substitution injection with no steganalysis

SSIM and PSNR for obliterator

<table>
<thead>
<tr>
<th></th>
<th>plane 1 (LSB)</th>
<th>plane 2</th>
<th>plane 3</th>
<th>plane 4</th>
<th>plane 5</th>
<th>plane 6</th>
<th>plane 7</th>
<th>plane 8 (MSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-46.85 ± 0.216</td>
<td>-39.96 ± 0.147</td>
<td>-34.36 ± 0.195</td>
<td>-28.29 ± 0.169</td>
<td>-22.32 ± 0.157</td>
<td>-16.26 ± 0.249</td>
<td>-11.55 ± 0.830</td>
<td>-4.89 ± 0.929</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9987 ± 0.000184</td>
<td>0.9951 ± 0.000614</td>
<td>0.9823 ± 0.002482</td>
<td>0.9468 ± 0.006407</td>
<td>0.8764 ± 0.012966</td>
<td>0.7447 ± 0.031166</td>
<td>0.6623 ± 0.048325</td>
<td>0.5410 ± 0.072004</td>
</tr>
</tbody>
</table>
SSIM and PSNR for Noise

![Graph showing the relationship between noise amount of change and PSNR.]
SSIM and PSNR for JPEG
Appendix 2 - Images for all levels of bit substitution and DCT
Bit substitution method injecting 400 000 bytes using bit 0

Bit substitution method injecting 400 000 bytes using bit 1

Bit substitution method injecting 400 000 bytes using bit 2
Bit substitution method injecting 400 000 bytes using bit 3

Bit substitution method injecting 400 000 bytes using bit 4
Bit substitution method injecting 400 000 bytes using bit 5

Bit substitution method injecting 400 000 bytes using bit 6
Bit substitution method injecting 400 000 bytes using bit 7

DCT coefficient replacement method injecting 6000 bytes using bit 0
DCT coefficient replacement method injecting 6000 bytes using bit 1

DCT coefficient replacement method injecting 6000 bytes using bit 2
DCT coefficient replacement method injecting 6000 bytes using bit 3

DCT coefficient replacement method injecting 6000 bytes using bit 4
DCT coefficient replacement method injecting 6000 bytes using bit 5

DCT coefficient replacement method injecting 6000 bytes using bit 6
DCT coefficient replacement method injecting 6000 bytes using bit 7
Appendix 3 - DCT coefficient change for individual attacks

The amount the DCT coefficients change when the LSBs are removed, expressed as a percent of the average of the coefficient in the unaltered cover image. The first number is the average change and the second number in parenthesis is the standard deviation of the change.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0,25</td>
<td>5,82</td>
<td>10,33</td>
<td>12,73</td>
<td>15,27</td>
<td>18,95</td>
<td>26,52</td>
<td>31,51</td>
</tr>
<tr>
<td>(0,11)</td>
<td>(7,60)</td>
<td>(13,97)</td>
<td>(19,53)</td>
<td>(27,25)</td>
<td>(34,47)</td>
<td>(47,88)</td>
<td>(60,63)</td>
</tr>
<tr>
<td>8,97</td>
<td>10,56</td>
<td>13,46</td>
<td>15,91</td>
<td>15,30</td>
<td>20,27</td>
<td>27,45</td>
<td>37,69</td>
</tr>
<tr>
<td>(11,45)</td>
<td>(15,00)</td>
<td>(20,40)</td>
<td>(26,21)</td>
<td>(30,36)</td>
<td>(40,64)</td>
<td>(54,43)</td>
<td>(82,67)</td>
</tr>
<tr>
<td>17,68</td>
<td>16,31</td>
<td>18,95</td>
<td>19,86</td>
<td>20,69</td>
<td>22,90</td>
<td>29,94</td>
<td>39,98</td>
</tr>
<tr>
<td>(23,94)</td>
<td>(24,98)</td>
<td>(28,57)</td>
<td>(35,15)</td>
<td>(41,09)</td>
<td>(46,37)</td>
<td>(60,75)</td>
<td>(86,43)</td>
</tr>
<tr>
<td>22,58</td>
<td>22,45</td>
<td>23,76</td>
<td>25,32</td>
<td>24,03</td>
<td>26,60</td>
<td>35,18</td>
<td>49,37</td>
</tr>
<tr>
<td>(34,03)</td>
<td>(37,32)</td>
<td>(41,54)</td>
<td>(45,95)</td>
<td>(52,45)</td>
<td>(60,09)</td>
<td>(78,88)</td>
<td>(114,50)</td>
</tr>
<tr>
<td>29,35</td>
<td>27,63</td>
<td>29,51</td>
<td>28,46</td>
<td>32,65</td>
<td>38,33</td>
<td>42,91</td>
<td>64,05</td>
</tr>
<tr>
<td>(52,09)</td>
<td>(54,36)</td>
<td>(60,98)</td>
<td>(62,33)</td>
<td>(71,32)</td>
<td>(85,38)</td>
<td>(104,85)</td>
<td>(147,14)</td>
</tr>
<tr>
<td>33,63</td>
<td>36,24</td>
<td>38,89</td>
<td>39,77</td>
<td>37,93</td>
<td>45,91</td>
<td>58,78</td>
<td>80,61</td>
</tr>
<tr>
<td>(64,15)</td>
<td>(74,94)</td>
<td>(79,27)</td>
<td>(89,32)</td>
<td>(92,49)</td>
<td>(109,51)</td>
<td>(142,61)</td>
<td>(201,16)</td>
</tr>
<tr>
<td>44,90</td>
<td>45,93</td>
<td>48,85</td>
<td>53,01</td>
<td>58,50</td>
<td>63,50</td>
<td>98,08</td>
<td>103,67</td>
</tr>
<tr>
<td>(85,75)</td>
<td>(98,21)</td>
<td>(102,99)</td>
<td>(118,05)</td>
<td>(142,79)</td>
<td>(151,76)</td>
<td>(238,96)</td>
<td>(275,25)</td>
</tr>
<tr>
<td>44,45</td>
<td>58,06</td>
<td>62,53</td>
<td>66,80</td>
<td>77,67</td>
<td>95,49</td>
<td>124,11</td>
<td>138,24</td>
</tr>
<tr>
<td>(89,00)</td>
<td>(128,21)</td>
<td>(141,71)</td>
<td>(156,48)</td>
<td>(189,23)</td>
<td>(239,65)</td>
<td>(309,08)</td>
<td>(363,06)</td>
</tr>
</tbody>
</table>
The amount the DCT coefficients change when JPEG compresses an image with a quality setting of 70, expressed as a percent of the average of the coefficient in the unaltered cover image. The first number is the average change and the second number in parenthesis is the standard deviation of the change.

| 0,28 (0,26) | 24,78 (27,78) | 41,14 (50,56) | 40,86 (64,50) | 48,08 (79,20) | 56,85 (122,53) | 97,58 (222,15) | 120,30 (469,67) |
| 29,79 (34,49) | 32,79 (54,48) | 38,99 (69,30) | 35,94 (82,57) | 34,07 (109,01) | 43,43 (161,08) | 77,12 (382,70) | 96,31 (563,68) |
| 45,61 (65,65) | 46,29 (76,16) | 43,47 (87,17) | 41,25 (116,36) | 45,33 (196,01) | 69,85 (345,20) | 94,30 (536,52) | 102,18 (685,01) |
| 58,71 (99,32) | 52,65 (111,76) | 51,17 (139,14) | 54,85 (189,21) | 69,42 (335,86) | 90,77 (510,31) | 105,54 (639,44) | 108,00 (723,33) |
| 58,70 (149,19) | 55,25 (177,73) | 59,02 (250,60) | 69,56 (337,09) | 87,07 (517,31) | 99,21 (678,96) | 111,56 (771,78) | 116,09 (815,59) |
| 76,03 (199,99) | 70,66 (276,09) | 81,68 (392,81) | 98,26 (578,65) | 101,15 (706,74) | 110,17 (744,84) | 109,72 (835,48) | 115,15 (884,45) |
| 95,66 (328,42) | 89,45 (419,31) | 100,26 (597,71) | 115,37 (668,81) | 113,09 (750,57) | 110,06 (814,78) | 130,77 (896,33) | 135,56 (889,46) |
| 112,18 (339,98) | 116,12 (531,07) | 125,05 (673,80) | 129,43 (719,89) | 129,45 (756,49) | 129,59 (857,78) | 141,84 (876,49) | 138,24 (1025,63) |
The amount the DCT coefficients change when 10 is added to each color value. The first number is the average change and the second number in parenthesis is the standard deviation of the change.

<table>
<thead>
<tr>
<th></th>
<th>418.69</th>
<th>22.40</th>
<th>11.65</th>
<th>7.62</th>
<th>5.08</th>
<th>4.55</th>
<th>3.95</th>
<th>3.64</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(699.61)</td>
<td>(97.63)</td>
<td>(50.47)</td>
<td>(33.40)</td>
<td>(23.97)</td>
<td>(20.77)</td>
<td>(18.17)</td>
<td>(17.17)</td>
</tr>
<tr>
<td>11.29</td>
<td>8.87</td>
<td>6.94</td>
<td>5.45</td>
<td>4.24</td>
<td>3.87</td>
<td>3.60</td>
<td>3.26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.01)</td>
<td>(38.02)</td>
<td>(29.52)</td>
<td>(23.15)</td>
<td>(18.55)</td>
<td>(16.70)</td>
<td>(15.94)</td>
<td>(14.97)</td>
</tr>
<tr>
<td>7.28</td>
<td>6.25</td>
<td>5.28</td>
<td>4.52</td>
<td>3.63</td>
<td>3.45</td>
<td>3.26</td>
<td>3.09</td>
<td></td>
</tr>
<tr>
<td>4.56</td>
<td>4.32</td>
<td>3.87</td>
<td>3.52</td>
<td>3.12</td>
<td>3.03</td>
<td>2.89</td>
<td>2.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(21.00)</td>
<td>(16.94)</td>
<td>(15.36)</td>
<td>(14.08)</td>
<td>(13.66)</td>
<td>(13.07)</td>
<td>(13.35)</td>
<td></td>
</tr>
<tr>
<td>4.29</td>
<td>3.91</td>
<td>3.68</td>
<td>3.44</td>
<td>3.08</td>
<td>3.03</td>
<td>2.78</td>
<td>2.66</td>
<td></td>
</tr>
<tr>
<td>3.81</td>
<td>3.56</td>
<td>3.37</td>
<td>3.17</td>
<td>2.85</td>
<td>2.81</td>
<td>2.70</td>
<td>2.60</td>
<td></td>
</tr>
<tr>
<td>3.62</td>
<td>3.25</td>
<td>3.09</td>
<td>2.96</td>
<td>2.71</td>
<td>2.59</td>
<td>2.51</td>
<td>2.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(16.81)</td>
<td>(13.69)</td>
<td>(13.16)</td>
<td>(12.34)</td>
<td>(11.86)</td>
<td>(11.52)</td>
<td>(10.99)</td>
<td></td>
</tr>
</tbody>
</table>