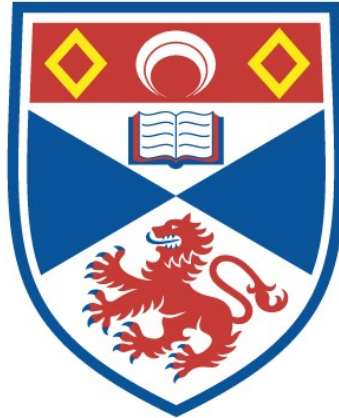


SOLVING THE THREAT OF LSB STEGANOGRAPHY  
WITHIN DATA LOSS PREVENTION SYSTEMS

Yunjia Wang

A Thesis Submitted for the Degree of MPhil  
at the  
University of St Andrews



2017

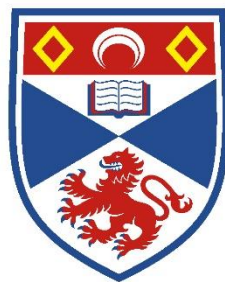
Full metadata for this item is available in  
St Andrews Research Repository  
at:  
<http://research-repository.st-andrews.ac.uk/>

Please use this identifier to cite or link to this item:  
<http://hdl.handle.net/10023/15662>

This item is protected by original copyright

# Solving the threat of LSB Steganography within Data Loss Prevention Systems

Yunjia Wang



University of  
St Andrews

This thesis is submitted in partial fulfilment for the degree of MPhil/PhD  
at the  
University of St Andrews

26 January 2017

## Abstract

With the recent spate of data loss breaches from industry and commerce, especially with the large number of Advanced Persistent Threats, companies are increasing their network boundary security. As network defences are enhanced through the use of Data Loss Prevention systems (DLP), attackers seek new ways of exploiting and extracting confidential data. This is often done by internal parties in large-scale organisations through the use of steganography. The successful utilisation of steganography makes the exportation of confidential data hard to detect, equipped with the ability of escaping even the most sophisticated DLP systems. This thesis provides two effective solutions to prevent data loss from effective LSB image steganographic behaviour, with the potential to be applied in industrial DLP systems.

### **Keywords:**

Data Loss Prevention (System); Steganography; Steganalysis; Histogram.

## Acknowledgements

First and Foremost, I would like to express my gratitude and sincere appreciation to my supervisor Ishbel Duncan, for your consistent aid in directing me throughout the all year. I could not have gotten through this year without you and you will forever be in my thoughts.

To Blair Fyffe, I would like to thank him for his consistent support the proof reading throughout this thesis.

Finally, I have to thank School of Computer Science and the University of St.Andrews for their teaching and support both academic and enjoyable life. Moreover, I would like to thank all friends in St Andrews for their support along the way.

Thank You

# Declaration

## 1. Candidate's declarations:

I, Yunjia Wang, hereby certify that this thesis, which is approximately 34983 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in January, 2016 and as a candidate for the degree of MPhil in January, 2017; the higher study for which this is a record was carried out in the University of St Andrews between 2016 and 2017.

Signature of candidate \_\_\_\_\_

26 January 2017

## 2. Supervisor's declaration:

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of ..... in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Signature of supervisor \_\_\_\_\_

26 January 2017

**3. Permission for publication:** *(to be signed by both candidate and supervisor)*

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Signature of candidate \_\_\_\_\_

Signature of supervisor \_\_\_\_\_

26 January 2017

# Contents

<b>Abstract</b> .....	2
<b>Acknowledgements</b> .....	3
<b>Declaration</b> .....	4
<b>1. Introduction</b> .....	15
<b>1.1. Research Background</b> .....	15
<b>1.2. Glossary</b> .....	18
<b>2. Literature Review</b> .....	20
<b>2.1. Data Loss Prevention</b> .....	20
<b>2.2. Steganography</b> .....	26
2.2.1. General Algorithms .....	26
2.2.2. Advanced Algorithms .....	31
2.2.3. Summary of Spatial Domain Image Steganography .....	50
2.2.4. Frequency Domain and its Stego-Algorithms .....	54
<b>2.3. Steganalysis</b> .....	56
2.3.1. General Detections .....	56
2.3.2. Advanced Detections .....	58
2.3.3. Summary of Steganalysis .....	65
<b>3. Methodology</b> .....	66
<b>3.1. The relevant challenges</b> .....	66
3.1.1. The Diversity of Steganography Algorithms .....	66
3.1.2. The Limitation of Steganalysis Methods .....	67
<b>3.2. Preliminary Investigations</b> .....	68
3.2.1. Network Transmission .....	69
3.2.2. Concealment of Data by Concatenation .....	71
<b>3.3. The Research Hypothesis</b> .....	74
<b>3.4. Experimental Architecture</b> .....	75
<b>4. Implementation</b> .....	80
<b>4.1. Developed Tools</b> .....	80
4.1.1. Text Steganography Tool .....	80
4.1.2. Text Steganalysis Tool .....	87
<b>4.2. Experimental Procedure</b> .....	89
4.2.1. Experimental materials and measurement units .....	90
4.2.2. Phase 1: The Implementation of LSB Replacement Algorithm .....	92
4.2.3. Phase 2: Determine the potential visual Threshold .....	105

4.2.4. Phase 3: Verify the applicability and reliability of solutions .....	112
<b>4.3. Analysis .....</b>	<b>125</b>
<b>5. Evaluation .....</b>	<b>128</b>
<b>6. Conclusion .....</b>	<b>137</b>
<b>7. Future Works .....</b>	<b>139</b>
<b>Bibliography.....</b>	<b>140</b>
<b>Appendix A .....</b>	<b>145</b>
<b>Appendix B.....</b>	<b>146</b>
<b>Appendix C.....</b>	<b>160</b>



## List of Figures

Figure 1: 2015 Exposed Records by Threat Vector .....	15
Figure 2: Lemonade steganography comparison between without (left) and with (right) a high temperature environment.....	16
Figure 3: Timeline of the development of the carrier in steganography.....	17
Figure 4: The illustration of Least Significant Bit in the colour channel .....	18
Figure 5: The illustration of Most Significant Bit in the colour channel .....	18
Figure 6: The ranking of top DLP vendors [13] .....	20
Figure 7: The classification of network attacks in 2015. Source: McAfee Labs 2015 [22]......	23
Figure 8: The illustration of LSB replacement algorithm (LSB 1) in the colour channel. ....	27
Figure 9: The illustration of LSB matching in the colour channel .....	28
Figure 10: The visual comparison between LSB 1 (left) and LSB 6 (right) .....	30
Figure 11: The illustration of Hash-LSB algorithm in the colour channel .....	33
Figure 12: The illustration of PVM algorithm in the embedding procedure [37] .....	38
Figure 13: The illustration of PVM algorithm in the extraction procedure [37].....	38
Figure 14: The description of l-h level and l-m-h level .....	40
Figure 15: The illustration of pixel blocks .....	42
Figure 16: The noise difference histogram between the original cover image and the stego image (without table) [43].....	45
Figure 17: The noise different histogram comparison between without (top) and with (down) the presented stego table [43].....	46
Figure 18: The resistance results comparison between the LSB matching and new proposed method [44] .....	47
Figure 19: The equation of Expected Number of Modifications Per Pixel [31] .....	48
Figure 20: The resistance result comparison in LSB matching and variable n in G-LSB-M [31] .....	49
Figure 21: The illustration of JPEG compression [58] .....	55
Figure 22: The visual detection comparison between the original image and the stego image .....	56
Figure 23: The histogram detection comparison between the original image and the stego image (with a replacement algorithm).....	57
Figure 24: The illustration of reduction block [50]. ....	61
Figure 25: Ratio of pattern counter comparison between a cover image (a) and the stego image (b) [50] .....	62
Figure 26: The histogram comparison between the original image and the LSB 1 stego image in Lena.bmp .....	68

Figure 27: The details of target PC.....	69
Figure 28: The detail of captured email packets.....	69
Figure 29: The explanation of IP: 216.58.198.197 .....	70
Figure 30:The specific detail of Secure Sockets Layer in the captured packet.....	70
Figure 31: The command of hiding data by the concatenation method in a Windows system .....	71
Figure 32: The extracted result of a target file by using the 7zip File Manager .....	71
Figure 33: The visual comparison between the original image and the merged image.....	72
Figure 34: The details of experimental files.....	72
Figure 35:The details of experimental files; the secret file is compressed to a zip file .....	72
Figure 36: The target hex result in Hex Fiend .....	73
Figure 37: The binwalk search result when the secret file was a TXT file .....	74
Figure 38: The binwalk search result when the secret file was a .zip file.....	74
Figure 39: The visual comparison between the original cover image and LSB 3 stego image, in StABridge.png .....	77
Figure 40: The visual comparison between the original cover image and LSB 3 stego image, in Lena.png.....	77
Figure 41: The Tools of this project .....	80
Figure 42: The details of process track in system console.....	81
Figure 43: The GUI layout in Steganography Tool .....	82
Figure 44: Details of the developed classes and packages in the steganography tool.....	82
Figure 45: The code explanation, convert decimal to string, fill the extra 0s until the length meets 8 .....	83
Figure 46: The code explanation, write an extra line break during reading the secret message.....	83
Figure 47: The code explanation, convert the secret message to 8-bit binary format. ....	84
Figure 48: The code explanation, the equation of quantity of expected replacing pixels .....	84
Figure 49:The code explanation, the procedure of replace the bits and generate a new pixel.....	85
Figure 50: The code explanation, the method of obtaining the random position in the cover image.	85
Figure 51: The code explanation, the procedure of re-combine a new pixel.....	86
Figure 52: The code explanation, the procedure of repaint a stego image from the new pixel matrix. .....	86
Figure 53: The visual comparison between the cover image and the LSB 5 stego image, in StAbaridge.png.....	86
Figure 54: The GUI layout in Steganalysis Tool .....	87
Figure 55: The details of process track in system console.....	88

Figure 56: The code explanation, the equation of quantity of replaced pixels .....	88
Figure 57: The code explanation, the method of identify the randomly hidden position .....	89
Figure 58: The details of experimental secret files .....	90
Figure 59: The methods of reading the cover image and the stego image in MatLab .....	91
Figure 60: The PSNR result between the cover image and the stego image .....	91
Figure 61: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with very short size data file. ....	93
Figure 62: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.bmp, with very short size data file. ....	93
Figure 63: The stego image results in LSB 1, LSB 3 and LSB 5, in Tiger.bmp, with very short size data file. ....	93
Figure 64: The stego image results in LSB 1, LSB 3 and LSB 5, in Cathedral.jpg, with very short size data file. ....	94
Figure 65: The stego image results in LSB 1, LSB 3 and LSB 5, in Edinburgh.jpg, with very short size data file. ....	94
Figure 66: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with very short size data file. ....	94
Figure 67: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.png, with very short size data file. ....	95
Figure 68: The stego image results in LSB 1, LSB 3 and LSB 5, in StABridge.png, with very short size data file. ....	95
Figure 69: The position of hidden (very short size) message file in the cover image, StABridge.png.	95
Figure 70: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with a small size data file. ....	96
Figure 71: The stego image results in LSB 1, LSB 3 and LSB 5, in MondrianTree.bmp, with a small size data file. ....	96
Figure 72: The stego image results in LSB 1, LSB 3 and LSB 5, in Tiger.bmp, with a small size data file. ....	96
Figure 73: The stego image results in LSB 1, LSB 3 and LSB 5, in Cathedral.jpg, with a small size data file. ....	97
Figure 74: The stego image results in LSB 1, LSB 3 and LSB 5, in Edinburgh.jpg, with a small size data file. ....	97
Figure 75: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with a small size data file. ....	97

Figure 76: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.png, with a small size data file. ....	98
Figure 77: The stego image results in LSB 1, LSB 3 and LSB 5, in StAaBridge.png, with a small size data file. ....	98
Figure 78: The distortion areas of the stego image in LSB 3.....	99
Figure 79: The distortion areas comparison in LSB 3, LSB 5 and LSB 8.....	99
Figure 80: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with a large size data file. ....	99
Figure 81: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.bmp, with a large size data file. ....	100
Figure 82: The stego image results in LSB 3 and LSB 5, in Tiger.bmp, with a large size data file. LSB 1 is not enough for concealing the secret message.....	100
Figure 83: The stego image results in LSB 3 and LSB 5, in Cathedral.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file. ....	100
Figure 84: The stego image results in LSB 3 and LSB 5, in Edinburgh.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file. ....	101
Figure 85: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with a large size data file. ....	101
Figure 86: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.png, with a large size data file. ....	101
Figure 87: The stego image results in LSB 3 and LSB 5, in StABridge.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file. ....	102
Figure 88: The stego image results in LSB1, LSB 3 and LSB 5, in Lena.bmp, with a large size data file with a random embedding algorithm. ....	102
Figure 89: The stego image results in LSB 3 and LSB 5, in Cathedral.jpg, with a large size data in randomly embedded. LSB 1 is not sufficient to conceal the secret message file.....	103
Figure 90: The stego image results in LSB1, LSB 3 and LSB 5, in Mondrian.png, with a large size file with a random embedding algorithm. ....	103
Figure 91: The stego image results in LSB 3 and LSB 5, in StABridge.png, with a large size data file with a random embedding algorithm. LSB 1 is not sufficient to conceal the secret message file. ...	103
Figure 92: The stego image visual result in LSB 5, with a large size of data in vertically embedded. ....	104
Figure 93: The stego image visual result in LSB 5, with a large size of data file with a random embedding algorithm .....	104

Figure 94: The PSNR distribution of experimental images, embedded a very short size data file. (20 bytes) .....	108
Figure 95: The PSNR distribution of experimental images, embedded a small size data file. (42 KB)	108
Figure 96: The PSNR distribution of experimental images, embedded a large size data file. (119 KB) .....	109
Figure 97: The MSE distribution of experimental images, embedded a very short size data file. (20 bytes) .....	111
Figure 98: The MSE distribution of experimental images, embedded a small size data file. (42 KB).	111
Figure 99: The MSE distribution of experimental images, embedded a large size data file. (119 KB) .....	112
Figure 100: The visual comparison between the original image and the stego image in LSB 1 .....	113
Figure 101: The histogram comparison between the cover image and stego image in LSB 1 .....	113
Figure 102: The extracted result from the stego image by using the stego key .....	114
Figure 103: The visual comparison between the Original stego image and the <b>Transformed</b> stego image.....	114
Figure 104: The histogram comparison between the Original stego image and the <b>Transformed</b> stego image.....	115
Figure 105: The extracted result from the Transformed stego image by using the same stego key .	115
Figure 106: The extracted result after transform back to the previous image format. ....	116
Figure 107: The visual comparison between the Original stego image and the <b>Modified</b> stego image .....	117
Figure 108: The histogram comparison between the Original stego image and the <b>Modified</b> stego image.....	117
Figure 109: The extracted result from the Modified stego image by using the same stego key .....	117
Figure 110: The visual comparison between the cover image and the stego image in LSB 2 .....	118
Figure 111: The histogram comparison between the cover image and the stego image in LSB 2 .....	118
Figure 112: The histogram comparison between the Original stego image and the <b>Transformed</b> stego image.....	119
Figure 113: The comparison of extracted result between the Original stego image and the <b>Transformed</b> stego image by using the same stego key .....	119
Figure 114: The visual comparison between the Original stego image and the <b>Modified</b> stego image .....	120
Figure 115: The histogram comparison between the Original stego image and the <b>Modified</b> stego image.....	120

Figure 116: The extracted result in the <b>Modified</b> stego image by using the same stego key.....	121
Figure 117: The visual comparison between the cover image (Lena.bmp) and the stego image (Lena.png) in LSB 2.....	121
Figure 118: The histogram comparison between the cover image (Lena.bmp) and the stego image (Lena.png) in LSB 2.....	122
Figure 119: The visual comparison between the Original stego image and the <b>Converted</b> stego image.....	122
Figure 120: The histogram comparison between the Original stego image and the <b>Converted</b> stego image.....	123
Figure 121: Comparison of the extracted result between the Original stego image (LSB 2) and the <b>Converted</b> stego image (LSB 2) from using the same key.....	123
Figure 122: The visual comparison between the Original stego image (LSB 2) and the Modified stego image (LSB 2).....	124
Figure 123: The histogram comparison between the Original stego image and the Modified stego image.....	124
Figure 124: The extracted result in the Modified stego image by using the same stego key.....	125
Figure 125: The PSNR comparison between the PDT solution and CMA solution in StABrdige.png with LSB 1.....	126
Figure 126: The MSE comparison between the PDT solution and CMA solution in StABridge.png with LSB 1.....	126
Figure 127: The PSNR comparison between the PDT solution and CMA solution in StABrdige.png with LSB 2.....	126
Figure 128: The MSE comparison between the PDT solution and CMA solution in StABridge.png with LSB 2.....	127
Figure 129: The PSNR comparison between the PDT solution and CMA solution in Lena.bmp with LSB 2.....	127
Figure 130: The MSE comparison between the PDT solution and the CMA solution in Lena.bmp with LSB 2.....	127
Figure 131: The experimental image.....	135
Figure 132: Cathedral.jpg.....	136

## List of Tables

Table 1: The comparison of LSB replacement algorithm and LSB matching algorithm.....	31
Table 2: The details of experimental images .....	90
Table 3: The result of PTR with different LSB replace bits, for a short size data file. ....	105
Table 4: The result of PTR with different LSB replace bits, for a small size data file. ....	106
Table 5: The result of PTR with different LSB replace bit, for a large size data file. ....	106
Table 6: The result of PSNR with different LSB replace bit, for a short size data file. ....	107
Table 7: The result of PSNR with different LSB replace bit, for a small size data file. ....	107
Table 8: The result of PSNR with different LSB replace bit, for a large size data file. ....	107
Table 9: The result of MSE with different LSB replace bit, for a short size data file.....	109
Table 10: The result of MSE with different LSB replace bit, for a small size data file. ....	110
Table 11: The result of MSE with different LSB replace bit, for a large size data file. ....	110
Table 12: The PSNR result in PDT solution with variable replaced bits, in StABridge.png .....	128
Table 13: The PSNR result in CMA solution with variable replaced bits, in StABridge.png .....	129
Table 14: The MSE result in PDT solution with variable replaced bits, in StABridge.png.....	129
Table 15: The MSE result in CMA solution with variable replaced bits, in StABridge.png .....	130
Table 16: The PSNR result in PDT solution with variable replaced bits, in Edinburgh.jpg.....	130
Table 17: The PSNR result in CMA solution with variable replaced bits, in Edinburgh.jpg .....	130
Table 18: The MSE result in PDT solution with variable replaced bits, in Edinburgh.jpg .....	131
Table 19: The MSE result in CMA solution with variable replaced bits, in Edinburgh.jpg .....	131
Table 20: The PSNR result in PDT solution with variable replaced bits, in MondrianTree.bmp .....	132
Table 21: The PSNR result in CMA solution with variable replaced bits, in MondrianTree.bmp .....	132
Table 22: The MSE result in PDT solution with variable replaced bits, in MondrianTree.bmp .....	132
Table 23: The MSE result in CMA solution with variable replaced bits, in MondrianTree.bmp.....	132
Table 24: The PSNR result in PDT solution with variable replaced bits, in Lena.bmp .....	133
Table 25: The PSNR result in CMA solution with variable replaced bits, in Lena.bmp.....	133
Table 26: The MSE result in PDT solution with variable replaced bits, in Lena.bmp.....	134
Table 27: The MSE result in CMA solution with variable replaced bits, in Lena.bmp .....	134
Table 28: The PSNR and MSE results in the experimental images after executing the PDT solution	136
Table 29: The PSNR and MSE result in Cathedral.jpg image after executing the PDT solution.....	136

# 1. Introduction

## 1.1. Research Background

In today's digital economy, company information is stored and transferred in digital formats. The rapidly developing Internet era has entirely changed company business models, primarily by making it both easier and faster to transfer information. However, nothing is safe on the Internet. Companies constantly fall victim to data loss, and high-profile data leakage involving sensitive personal and corporate data are frequent [1]. According to the Risk Based Security Data Breach QuickView incidents, there were 3930 incidents reported during 2015 exposing 736 million records [2].

Generally, data loss mainly comes from malicious attacks, which come in the form of either internal attacks and external attacks. External attacks represent a compromise, often of servers, which are subjected to malicious attacks from outside of the company network boundaries. External attacks are blocked as much as possible with the use of firewalls and security patching, combined with ethical hacking to address potential issues. Another large threat comes in the form of the internal attack. These stem from intentional (staffs transferring data on purpose by, for example, email or USB) or unintentional accidents to malicious exploitations of company data, often exported through the network or USB devices. Vormetric Data Security indicates that globally, around 89% of respondents felt their organization has suffered a risk from an insider attack, with 34% classifying themselves as feeling extremely vulnerable to attacks in the future [3]. Similarly, Risk Based Security, a cyber-security company, stated 49% of threat vectors were from "inside the organization" activities [2], as show in Figure 1:

Threat Vector	Records Exposed
Outside	375,768,013
Inside-Accidental	236,404,844
Inside-Malicious	73,814,654
Inside-Unknown	49,431,541
Unknown	769,798
<b>Total</b>	<b>736,188,850</b>

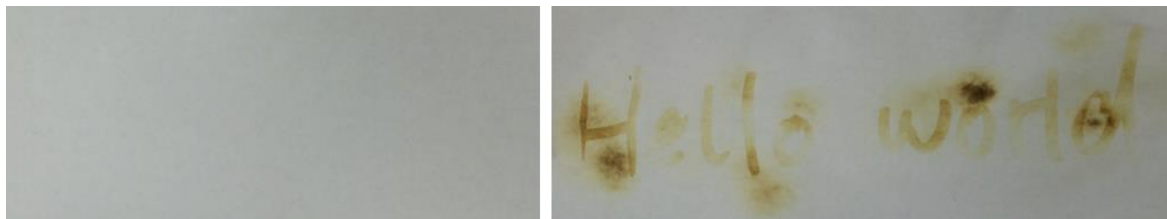
*Figure 1: 2015 Exposed Records by Threat Vector*

For an enterprise, all forms of data loss (whether leakage, disappearance or damage) could cause a variety of consequences, at the cost of not only financial but also reputational damage. However, due to Data Loss Prevention systems, these risks and threats are greatly mitigated. Data Loss Prevention systems, better known as DLP systems, protect and monitor confidential enterprise data in three different states: data at rest, data at endpoints, and data in motion [4]. Any data breach which is



caused by malicious behaviours will be flagged and blocked in this system. Nevertheless, there is no all-encompassing silver bullet. For instance, If the secret data is attached in cover file images through steganography, it will escape DLP defence systems monitoring and consequently this is a major concern.

Steganography is an ancient science for hiding the information in the communication. The inspiration and source of steganography came from the natural animal or plant world [5], where an organism's abilities of camouflage and impersonation often significantly improve its chances of life and success. This "art" originated in Greek, meaning "covered writing" [6]. In early steganography, for example, the secret message was written on a man's shaved head, this message was hidden naturally after his hair had grown back [7]. During the World War I and II, the application of invisible ink also utilized the concept of steganography [7]. A similar basic example is commonly known as lemonade steganography, which is demonstrated in Figure 2 below. Firstly, a juice was squeezed from a lemon. Then the juice was used as ink to write the secret message on paper and at this point the writing would be invisible to the naked eyes. However, this writing will reappear once this paper is placed in a high temperature environment.



*Figure 2: Lemonade steganography comparison between without (left) and with (right) a high temperature environment*

So far, steganography is defined as a technique for concealing information [8], and it is applied in the area of the computers and networks widely. In Figure 3 below, the chronological timeline of the development of the different carrier within steganography has been listed [5]. Unlike in cryptography, the steganography has unique features against relevant detections, e.g. higher resistance and imperceptibility [9]. Hence, the detection of steganography is often regarded as a problem worth researching.

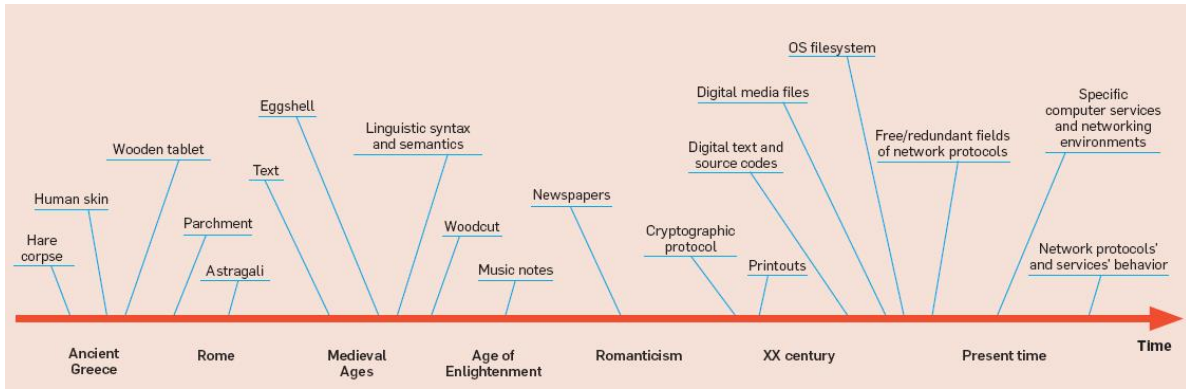


Figure 3: Timeline of the development of the carrier in steganography

Moreover, security often goes against user experience. Higher security results in lower user flexibility. For companies, absolute network segregation is a way to protect the confidential data in some cases, but it will bring more inconvenience to the employees. Although, the majority of existing steganalyses (solutions) can be used to identify the corresponding steganography algorithms, the implementation of these solutions are not reliable in DLP systems due to the limitation of each steganalyses method and the diversity of steganography algorithms (this will be discussed in Section 3). Also, many images are normally sent in normal company network traffic. Therefore, an adaptive solution is necessary.

This thesis examines spatial image steganography, and presents a novel approach to prevent the data loss threat, with the potential to be applied within industrial DLP systems. A research hypothesis is presented, which is that by slightly modifying the pixel values of spatial domain images, secret messages hidden inside cover images will be destroyed, while the quality of the image will be preserved. On the condition that there is no visible distortion of the image, two effective solutions are presented and verified. One is an adapted solution, called Presentation Domain Transform (PDT), which uses the irreversible JPEG compression procedure to affect the original pixels' value. The other is a novel solution, named Chrominance Modification Algorithm (CMA), which performs random manipulations in the chrominance value to change the associated pixel value. In the second chapter, the literature will be reviewed. The methodology and implementation to prove the reliability and applicability of presented solutions are described in the third and fourth chapters respectively. The evaluations of these solutions are compared to verify the most effective one in the fifth chapter. Finally, the conclusion and future work chapter complete this thesis.

## 1.2. Glossary

Prior to reviewing the existence literature, the relevant terminologies will be explained in this sub-section.

### **Data Loss Prevention Systems**

In this thesis, Data Loss Prevention (DLP) is defined as a protection system that prevents potential data breaches [10].

### **Steganography**

Steganography is an ancient art for hiding secret messages under the carrier (cover). In this thesis, the secret message was hidden in digital images. The relevant hiding algorithms will be discussed in the next Literature Review section.

### **Steganalysis**

Steganalysis is a reversing technique of steganography, which is used to prove the existence of a secret message in the cover image. The relevant detection methods will be introduced in the next Literature Review section.

### **Least Significant Bit (LSB)**

The range of colour intensity which can be represented in 8-bit binary form is from 0 to 255. The LSB of a colour pixel is often defined as the 8<sup>th</sup> bit in the colour channel. As illustrated in Figure 4 below:



Figure 4: The illustration of Least Significant Bit in the colour channel

### **Most Significant Bits (MSB)**

The MSB is often defined as the 1<sup>st</sup> bit in the colour channel [11]. As illustrated in Figure 5 below:

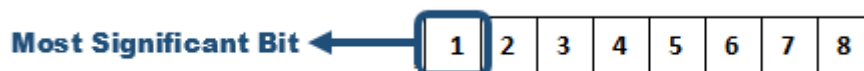


Figure 5: The illustration of Most Significant Bit in the colour channel

### ***Robustness***

The quality of steganographic algorithms can be examined by three factors: robustness, embedding capacity and the visual quality in the steganographic image. Robustness is the ability of an algorithm to defend against systems that endeavour to examine the potential hidden data [12].

### ***Histogram***

In this thesis, the histogram is used to examine the frequency of colour intensity in the target image, and this is called an image histogram. The distribution in an image histogram represents the amount of each colour value in the target image.

## 2. Literature Review

Prior to presenting the solution in the thesis, the relevant literature was thoroughly researched and reviewed from three aspects; Data Loss Prevention, Steganography and Steganalysis.

### 2.1. Data Loss Prevention

Data loss prevention (DLP) is a protection system that prevents the occurrence of data breaches. It can also be seen as a strategy for making sure that confidential data cannot be transmitted out of the company network boundary by either an insider employee or outsider competitors [10], whether accidentally or maliciously. However, with the occurrence of increasing data breaches, computer security vendors have to develop correspondingly sophisticated protection software to govern these risks. Hence, the term DLP is also used to describe the software protection products that help the company to keep their confidential data under their control as well as out of the public domain. So far, there are a lot of security vendors that support the deployment of DLP systems. The top companies that provide this were ranked in “SelectHub” according to their popularity [13], as shown in Figure 6:

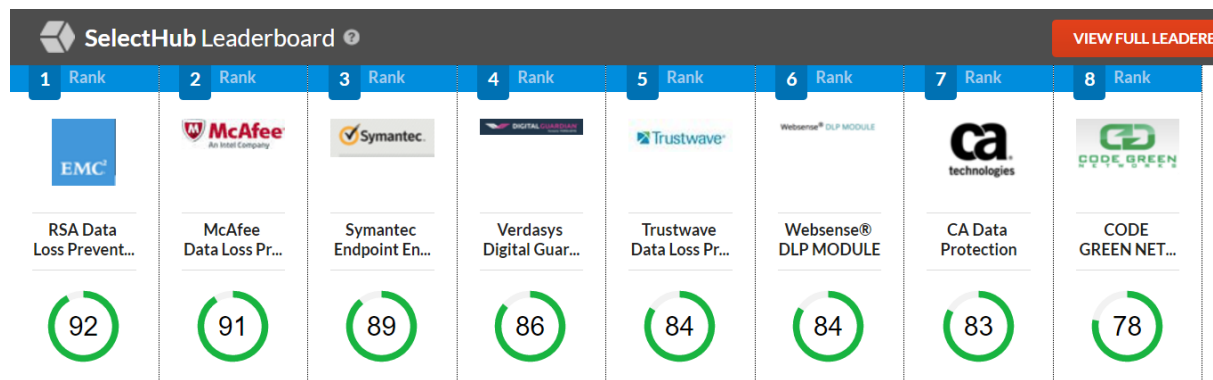


Figure 6: The ranking of top DLP vendors [13]

### Case Studies

Although the protection systems are deployed within the enterprise, many data loss incidents still continually occur. As reported in many online articles, the most infamous data breaches in the UK have been listed [14], such as:

**1. The “Three Mobile” event, 2016. [14]**

The customer upgrade database was accessed illegally using an employee’s login identification. Although the company announced the financial information was safe and was not accessed, the corresponding customer personal information was still at risk.

**2. The “Tesco Bank” event, 2016. [14]**

In early November, the British supermarket giant, Tesco, was subject to criminal activity, causing up to 40,000 customer’s data records to be compromised, and almost half of them had money stolen from their personal online bank account. Also, Tesco had to shut down all of their online banking operations; the customers could not perform any online transactions until the situation was under control.

**3. The “Sage” insider access event, 2016. [14]**

In Sage Group plc, a FTSE-100 firm in the UK, an insider illegally accessed a large volume of users’ data putting them at risk. This event will become one of the most important data breach cases in the UK’s history if the affected scale is confirmed.

**4. The “Mumsnet” event, 2014. [14]**

Mumsnet, one of the UK’s largest website for parents, suffered from the Heartbleed SSL attack due to software vulnerabilities. This allowed up to 1.5 million users’ data to be accessed freely by hackers. Although nothing was leaked that was financial, the reputation of the brand has been called into question.

**5. The “Sony PlayStation Network” event, 2011. [14]**

This damage was referred to as the largest data breach in history at the time, with almost 77 million users’ data affected. Incredibly, both personal details or transaction records were dropped (removed) from the database due to this outsider attack. The company’s system was offline for 23 days, and subsequently the influences and repercussions were felt worldwide.

Data breaches are a very common phenomenon with the majority of companies having suffered this threat. Besides UK companies, these problems still exist in internationally well-known IT companies, even if they have good security groups within their company, such as Snapchat, Facebook, Apple. As the report has shown:

**1. Getting Scammed and Leaking Employee Data in Snapchat. [15]**

In 2016, their official blog identified employee information having been compromised, though the users’ data was safe. The main reason behind the compromise was from a phishing email that was presented as being sent by the CEO, but actually imitated by a hacker. Neither the security system,

nor the employees identified its non-authenticity. Afterwards, the employee data was completely exposed on a website.

However, a similar data breach occurred in Snapchat before. In late 2013, due to system vulnerabilities caused by their security group, up to 4.6 million accounts had been disclosed and the data could be downloaded directly from SnapchatDB.info [16].

## **2. *Data breach - exposing 6 million users in Facebook.* [17]**

As the giant of the world's social networking, Facebook did not escape from having a data loss incident and being breached. The official announcement stated that almost 6 million users' information was exposed to an unauthorized viewer due to system bugs in 2012.

## **3. *The infamous celebrity breach in Apple – iCloud hack***

In 2014, almost 500 private nude pictures of various celebrities were exposed from the iCloud server and disclosed on various websites. The security of cloud services has been called into question, bringing security to the forefront of everyone's minds.

Although Apple has denied the security breach, the verification procedures of iCloud were subsequently increased, and "two-step verification" checking has been activated to protect customers' stored online data [18][19].

Consequently, data breaches frequently feature prominently in the news. No company seems to be immune to this phenomena, whether large or small. These malicious behaviours are normally caused by two vectors seen in the examples above; external attack and internal attack. The external attack vector is almost always implemented through exploiting the corresponding vulnerabilities, such as SQL injection, XSS, etc. Confidential data can be exposed completely to any unauthorized viewer once the relevant attack command has been injected.

In addition, as the statistics from 2015 showed, around 84% of enterprises said that the most common attack experienced by their company is SQL injection [20]. However, with the constant upgrading of protection software such as Intrusion Detection Systems (IDS), some vulnerabilities can be repaired or detected much more quickly. Compared with previous attack incidents, the attacks which involve exploiting system vulnerabilities have reduced appreciably. From the 2015 statistics, the common network attack types are now 'flood attacks' rather than based on attacking vulnerabilities, as shown in Figure 7 [22]:

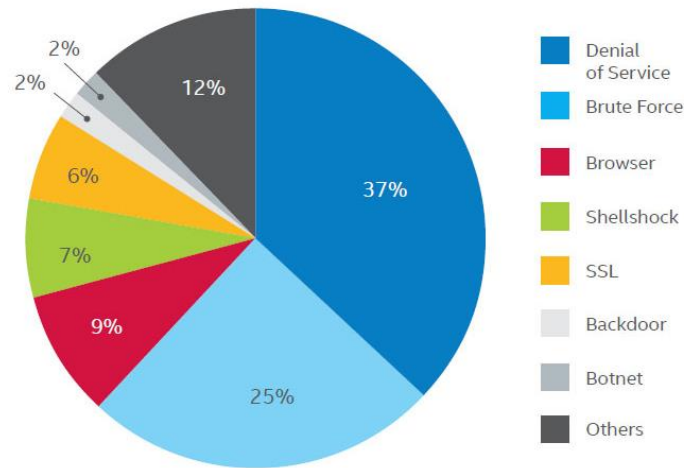


Figure 7: The classification of network attacks in 2015.  
Source: McAfee Labs 2015 [22].

Nowadays, the greatest threat of data loss is that of an internal attack [2]. The malicious actors are often insider employees as data may be accessed from the company database either intentionally or unintentionally. As reported by Ernst & Young, the corresponding malicious activities are given as examples [23]:

- Unsuitable rights assigned to access confidential data.
- Exploitation of limitations in the database's development environment.
- Betrayal of trust among developers.
- Front office without supervision.
- Employee discontent.
- Trading secret information with insiders.

For the cybersecurity teams in the enterprise, the preventative solutions are becoming increasingly challenged. Especially, with the rapid development of mobile devices, the data can be quickly accessed and extracted by physically removing a device, such as an iPhone or an iPad. Although flexibility has increased significantly, the loss risk has also increased proportionally. The laptop & mobile loss cases are also very frequent. In addition, with the increasing trend of cloud computing usage, the centralised storage that is referred to as cloud storage has been widely used by start-ups and medium enterprises alike resulting in data that is accessible everywhere which makes it much more difficult to track data storage and access within the enterprise [23]. To sum up the above challenges, the biggest trouble is that corporate data is not under their control. Prior to preventing data breaches effectively, the cybersecurity employee has to understand and identify the following fundamental questions [23]:



1. *What is the data being protected?*
2. *Where is the data?*
3. *Where does the data need to go/never go?*

A Data Loss Prevention systems is an integrated defence system, which is implemented to prevent potential data breaches both internally and externally. With respect to the movement of enterprise data, this definition consists of three aspects: Data at Rest (resides in database, or other storage centres); Data at the Endpoint (resides at an external device, such as laptops, USB); and Data in Motion (communication traffic, such as email, P2P) [24].

Computer security companies, such as Symantec and McAfee, discuss the implementations of DLP systems and divide the work into three predominant processes: Discover, Monitor and Protect. The first step is to discover and locate where the company's confidential data resides. The following process is to monitor how this data is being used, who is using it, and where it is going. Finally, the relevant permissions should be adjusted and corresponding policies set up to prevent any data loss [25] [26].

With the expansion of enterprises, the complexity of managing traffic rises, especially due to the amount of endpoint devices which are constantly added to internal networks. Data loss becomes easier as an attacker can hide data within larger volumes of traffic in the network or in a higher frequency of resources requests, either in cloud or in database centres. Consequently, an effective DLP program should adopt appropriate comprehensive techniques to cover all of the organization's potential loss models, rather than consider any single part only [27].

From the introduction in the Symantec DLP system white paper, the specific protection procedure is [25]:

- ***Explore data in Content-Aware detection***

Exploiting a combination of advanced techniques to accurately detect the location of all of the confidential data, whether at rest, in motion, or in use, and also the type of data and file can be identified by using a vector machine learning method.

- ***Monitor and Protect Cloud-Based Email and Storage.***
- ***Ensure the Security in Endpoints***

As both modules of endpoint discovery and prevention are enabled by a single scalable agent in the Symantec DLP system, the specific operation in the module is, for example, to perform a multi-tiered

local scan either in a laptop or in a desktop, and to monitor the real-time stream and notify the user with a pop-up window.

- ***Ensure the Security in Mobile Devices (BYOD)***

Extending the monitoring and protection to all of your devices, the Mobile Email module is used to detect when confidential email is downloaded into mobile devices, also the Mobile Prevent module is used to monitor the transmission of confidential data via the native mail client, browser, and other apps.

- ***Locate and Protect the Unstructured Data***

Unstructured data can be controlled through four essential DLP modules, such as Symantec DLP Network Discover, Network Protect, Data Insight and Data Insight Self-Service Portal. It is possible to implement the discovery, protection, and management of the confidential data across virtually any storage system.

- ***Monitor and Protect the Data in Motion.***

A wide range of network protocols can be monitored and protected through three more network modules. These are Symantec DLP Network Monitor, Network Prevent for Email, and Network Prevent for Web.

As demonstrated in the above procedure, DLP systems are an integrated defence system, designed and implemented by multiple function modules. Admittedly, confidential data loss is of utmost concern to the companies, DLP system offer products with a wide range of functionality and capability to protect the company's data. However, none of these produces can detect insider use of steganography [69].

Laboratory of Cryptography and System Security [68] indicated that globally, a worrying trend is that of using steganography to send the information over the Internet, and this has been done by some malwares already, such as Duqu. Not only companies, but countries have been the victim of steganographic loss. For example, in 2008, sensitive financial information was hidden inside JPEG images and sent out of US Department of Justice [68]. Hence, data loss and steganography have been the subject of concern by security researchers.

## 2.2. Steganography

### 2.2.1. General Algorithms

Prior to discussing the image steganography algorithms and steganalysis detections, the basic format of the digital image should be noted. Jessica Fridrich stated [28]: “Digital images are commonly represented in four basic formats – raster, palette, transform and vector”. The most popular images are based on raster, palette and transform, which represents BMP, PNG and JPEG respectively.

The most common image steganography algorithm is the Least Significant Bit (LSB) method. However, during the long developmental history of steganography, this ‘war’ between the two sides (steganography and steganalysis) has been continuously progressing for the past decade. The embedding methods have evolved and upgraded while the detection method (steganalysis) has also been updated continually, such as LSB,  $\pm 1$ , JPEG, F3, F4 and F5 [67].

Although the algorithms have been classified and proposed in many formats, the categories based on their algorithms can be divided into two areas; the Spatial Domain and the Transform (Frequency) Domain. In [29], the difference in the techniques between these two domains is explained. In spatial domain imagery, the message is embedded into the intensity of the pixel directly through replacing the LSB value with the cover image. In transform domain imagery, the message is embedded into the frequency table of the previously transformed image by using the same algorithm. Therefore, for these two algorithms, the spatial domain techniques change the pixel colour intensity value (RGB) to implement the message hiding; but in transform domain techniques, the frequency is altered rather than the intensity of colour.

Whether steganographic techniques are in either the spatial domain or the transform domain, they have been improved and presented constantly in recent research papers. In this thesis, the research is focused on the spatial domain imagery as the transform domain imagery is more complicated, and can be analysed in future research.

The most common algorithm in the spatial domain image is the LSB algorithm, which utilizes colour redundancy in the human eyes to replace the Least Significant Bits of pixel bytes where hidden data is stored in a cover image. With the updating of the algorithm, LSB can be classified into four areas [30]: LSB Replacement, LSB Matching, LSB matching revisited (LSBMR) and LSBMR-based edge-adaptive (LSBMR-EA). However, LSBMR and LSBMR-EA belong to the extension of the LSBM algorithm in their classification as they are still very much related to LSB matching. Therefore, in this research, the LSB algorithm will be classified into two areas to be considered: LSB replacement and LSB matching.

### LSB Replacement Algorithm

LSB replacement algorithm is also called as LSB embedding or LSB substitution algorithm. This algorithm conceals the data through replacing the LSBs of cover images with embedded message bits directly. As the illustration in Figure 8 shows:

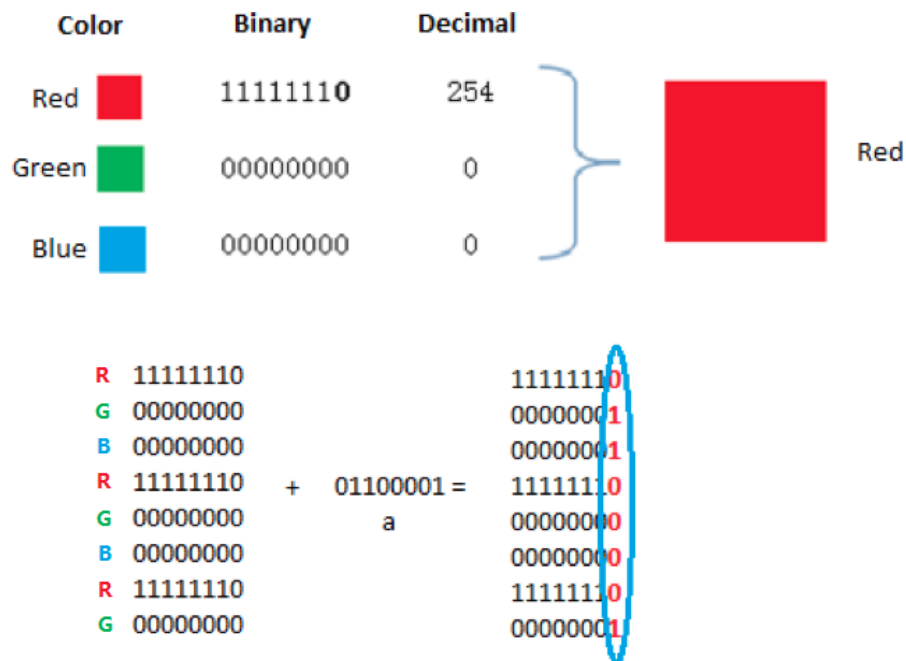


Figure 8: The illustration of LSB replacement algorithm (LSB 1) in the colour channel.

The colour “Red” has the values [red: 254, green: 0, blue: 0] in an array; this is transformed into a binary code of [11111110, 00000000, 00000000]. The letter “a” is represented in binary by “01100001”. Thus, if the user wants to hide the message into the cover image where the background of this cover image is pure “Red”, the LSB 1 will take the last 1-bit from the colour array, and exchange it with the data values. The newly generated value could combine a new “Red” that is similar with the previous “Red”.

(In this case, 3 pixels were picked<sup>1</sup> for concealing the letter “a”. Before the embedding, they were, in binary [11111110, 00000000, 00000000], [11111110, 00000000, 00000000] and [11111110, 00000000, 00000000]; in decimal [254, 0, 0], [254, 0, 0] and [254, 0, 0].

After the embedding, they were updated to: in binary [11111110, 00000001, 00000001], [11111110, 00000000, 00000000] and [11111110, 00000001, 00000000]; in decimal [254, 1, 1], [254, 0, 0] and [254, 1, 0].)

<sup>1</sup> Because each pixel can conceal one character by using the LSB 1 algorithm and the letter consists of 8 characters in binary, thus it requires 3 pixels to implement.

### LSB Matching Algorithm

Unlike the LSB replacement algorithm, LSB matching does not replace the message within the cover image directly. It embeds the data through comparing the confidential message with LSBs in the cover image first. If the message bit matches the LSB in the cover image, this pixel will stay the same. Otherwise, this pixel will be randomly either added or subtracted by 1 [30], as the illustration in Figure 9 shows:

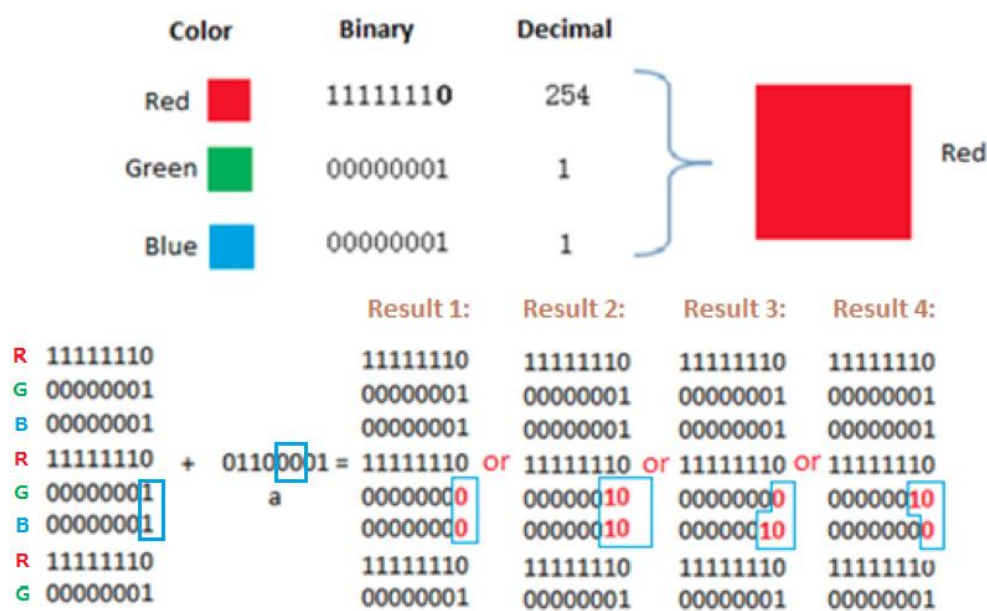


Figure 9: The illustration of LSB matching in the colour channel

Assume the colour "Red" has the values [red: 254, green: 1, blue: 1] in an array. This was transformed to a binary code of [11111110, 00000001, 00000001]. The letter "a" is "01100001" in binary. If the user wants to hide the message in the cover image where the background of this cover image is a single colour, in this case **Red** in the LSB matching algorithm, the message data (binary bit) will be compared with the LSB of each pixel in the cover image first rather than being replaced with them directly. The pixel in the cover image was randomly incremented or decremented by 1 when the message bit is not the same as the corresponding LSB in the cover image. So, as the display of the above figure shows, only the fifth and sixth bits were modified (marked by the blue rectangles). The new generated values should have 4 possibilities, all of them could form a new "Red" that is similar with the previous **Red** as well.

(In this case, 3 pixels were picked for concealing the letter "a". The different fifth and sixth characters of message data corresponded with the Green and Blue colour channels in the second pixel of the cover image. It has 4 possibilities due to the randomness of the LSB matching algorithm. Therefore:

In Result 1: Green minus 1, Blue minus 1, the second pixel is, in binary [11111110, 00000000, 00000000]; in decimal [254, 0, 0].

In Result 2: Green plus 1, Blue plus 1, the second pixel is, in binary [11111110, 00000010, 00000010], in decimal [254, 3, 3].

In Result 3: Green minus 1, Blue plus 1, the second pixel is, in binary [11111110, 00000000, 00000010]; in decimal [254, 0, 3].

In Result 4: Green plus 1, Blue minus 1, this pixel is, in binary [11111110, 00000010, 00000000]; in decimal [254, 3, 0]).

### ***The extraction procedure of both algorithms***

During the process of extraction data, in the LSB replacement algorithm, the last  $n$  bit of each pixel will be chosen from the target image according to the specific stego-key (which includes the exchange bit and the location of exchanged pixel). Next, these single characters can be recombined into the 8-bits binary codes, and the plaintext can be transformed from these binary codes.

However, in the LSB matching algorithm, only the last bit of each pixel will be chosen from the target image due to their random manipulation during the embedding process. Then perform the same work flow as the replacement algorithm to recombine and transform the message from these picked bits.

### ***The difference between LSB replacement and LSB matching***

During the process of embedding data, in the LSB replacement algorithm, the data is replaced with the LSB of the cover image directly. Normally, the last bit of each colour channel in the cover image is used for concealing the message data, we called it "LSB 1" or "1-bit LSB". However, with a constant increase in data size, the maximum bits available within "LSB 1" is occasionally not enough to hide all of the data. Generally, in order to digest large volumes of data, it is easier to utilize more colour bits to take part in the replacement, and this is called "LSB  $n$ ". The integer  $n$  can be determined by the sender according to the size of the message data and cover image (LSB 2, takes the last 2-bits; LSB 3, takes the last 3-bits; etc.). With an increasing  $n$ , the quality of the image will be degraded. As Figure 10 conveys, the LSB 1 algorithm and the LSB 6 algorithm for hiding the same message on a cover image have very different results with the LSB 6 producing an image of a noticeably reduced quality.



*Figure 10: The visual comparison between LSB 1 (left) and LSB 6 (right)*

In the LSB matching algorithm, only the last bit of the RGB pixel in the cover image will be changed only when the message bit does not match, also the changed result will match the message data completely whether by adding or subtracting 1 as there are only '0' and '1' in binary code. For example, if the pixel in the cover image is 11111110 (254), and the bit of the message is 1, using the LSB replacement algorithm, the message bit will be replaced with this pixel directly to obtain the new pixel value 11111111; the message is stored in the last bit in the pixel. In the LSB matching algorithm, this pixel will be added or subtracted by 1 randomly as the last bit of this pixel is '0' which does not match with the message bit '1'. However, whether adding (the result of new pixel will be 11111111) or subtracting (the result of new pixel will be 11111101), the message will be stored in the last bit as well. From this example, in an LSB 1 replacement, all of the changed bits are based on: when even, add 1, and when odd, subtract 1. This regularised modification will make this algorithm easily recognised by histogram detection [31]. Instead, in the LSB matching algorithm, the random operations can make these new pixel values distributed more evenly in the histogram rather than using a set pattern. Notably, the LSB matching algorithm can only be used in the last bit of the RGB value in the cover image, otherwise the message data would not be stored in the cover image correctly.

In the process of extracting the data, in the LSB replacement algorithm, the receiver needs a stego key which contains the value of replaced bit  $n$  and the length (size) of message bits (in binary form). The message can be extracted in binary form through this stego key, then the plaintext can be displayed after transforming into text form. However, in the LSB matching algorithm, the key consists of the length (size) of the message bits (in binary form) as only the last bit has changed during the embedding process, otherwise the extracted result will be unreadable. Finally, the message can be extracted in binary form directly according to the key's length and the plaintext can be displayed after further conversion.

(NB: In these cases, the replaced / matched pixels start from the coordinate (0,0) in the cover image. Other situations are not considered as the key is required to import the location of changed pixels.)

### **Evaluation (LSB replacement & LSB matching)**

Comparing these two algorithms, the LSB replacement algorithm has a greater embedding capacity, although the quality of image can be reduced with the increase of replaced bits  $n$ . The robustness is lower due to its constant modification (even add 1, odd subtract 1), the stego image can be identified by histogram detection, especially when the embedding message has a large volume size. During the extraction process, the stego key consists of two parameters. The length (size) of the binary message bit is an essential parameter; the second parameter is the value of replaced bits  $n$ , otherwise the extraction data will be unreadable.

Instead, in the LSB matching algorithm, although it has lower space for concealing the message data as only the last bit can be used, the random manipulation (of pixel placement) can increase the robustness, so that the stego image cannot be identified easily even in histogram detection. Also during the extraction process, the stego key only needs to contain the length (size) of binary message bit.

A summary of the difference and evaluation is shown in the Table 1:

	Embedding Space	Robustness	Stego Key
LSB replacement	Higher	Lower	N, Length of message
LSB matching	Lower	Higher	Length of message

*Table 1: The comparison of LSB replacement algorithm and LSB matching algorithm*

(NB: In this table, the “Higher” and “Lower” is only relative to each other)

In the next sub-section, the relevant advanced algorithms, improvements in both the LSB replacement and the LSB matching will be introduced.

### 2.2.2. Advanced Algorithms

**“FPGA Hardware of the LSB Steganography Method.”** By Mohd, Abed etc.[32]

Firstly, three different measurement units were mentioned to measure the imperceptibility of Steganography, they were **Mean Squared Error (MSE)**, **Bit Error Rate (BER)**, and **Peak Signal-to-Noise**



Ratio (PSNR). A higher value of MSE indicates dissimilarity (difference) between the compared images. The value of PSNR indicates the quality of image, with a higher value indicating a better quality.

Next, a new algorithm was proposed by Mohd et al, which is concealing confidential data in spatial domain images through manipulating the pixel bit values in the cover image. Unlike the LSB replacement algorithm, this algorithm is based on the combination of LSB 2 and LSB 3 rather than modifying the constant bit(s) in each colour channel. The data is embedded into the pixel with 2 bits in the red channel, 3 bits in the green channel and 3 bits in the blue channel. Thus, it is also called **LSB 2/3**. The benefit of this algorithm is that a complete character can be embedded in each individual pixel. Also the measurement results of MSE and PSNR are between the LSB 2 and the LSB 3, there have:

In MSE,  $LSB\ 2 < LSB\ 2/3 < LSB\ 3$

In PSNR,  $LSB\ 3 < LSB\ 2/3 < LSB\ 2$ .

In the conclusion a set of images including the famous "Lena", were tested with the LSB 2/3 algorithm to verify its performance. The resultant stego image not only has a good image quality, but also requires relatively lower memory for computation.

***"A Secure Image Steganography Based on RSA Algorithm and Hash-LSB Technique."*** By Kumar and Sharma [33]

In order to enhance the security of steganography, a comprehensive method was proposed by Kumar and Sharma, which was a combination of cryptography and steganography. The authors recommended implementing the RSA algorithm and Hash-LSB technique at the same time.

In their proposed system, the confidential data has to be converted to cipher text using the RSA algorithm to enhance the secrecy of the message before the embedding process, then the novel insertion technique Hash-LSB which is derived from the LSB replacement algorithm was implemented.

In the Hash-LSB algorithm, the position of the LSB in the cover image for concealing the confidential data was determined through a hash function, as shown in the equation below. *"Where  $K$  is the position of RGB within the pixel,  $p$  denotes the position of each hidden image pixel and  $n$  is the number of bits of LSB, which is 4 for this present case"*.

$$K = p \% n$$

During the embedding process, the confidential data is converted into binary form. Each 8 bits of the message data is embedded in the LSB of the cover image pixel directly in the sequence of 3, 3 and 2 independently (3 bits in red, 3 bits in green and 2 bits in blue) as “*the chromatic influence of blue colour to the human eye is more than red and green colours*”. Also, it means that each pixel can be embedded a complete character in binary form, as shown in Figure 11:

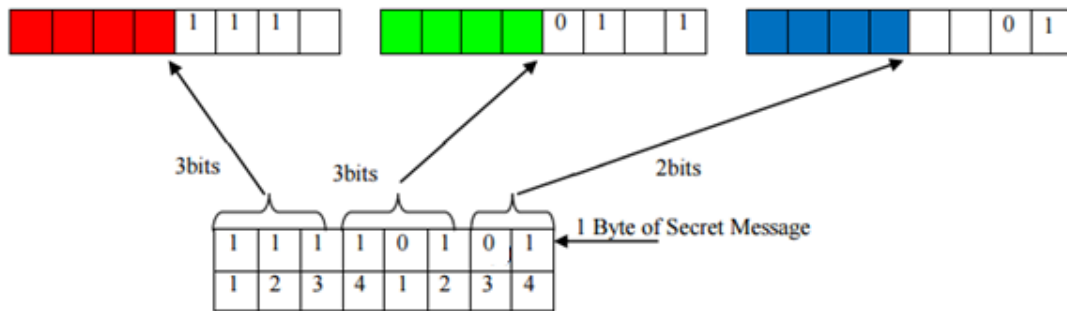


Figure 11: The illustration of Hash-LSB algorithm in the colour channel

In this case, the sequence of 3, 3 and 2 was implemented to embed the message in the LSB of the RGB pixel respectively. The position can be obtained from the above equation, the value of  $K = 1, 2, 3$  in red channel,  $K = 4, 1, 2$  in Green channel, and  $K = 3, 4$  in Blue channel. Therefore, the 1-byte message data “11110101” was split to “111” in the Red channel with position 1,2,3; “011” in the Green channel with position 1,2,4; and “01” in the Blue channel with position 3,4.

In the final part, they calculated and compared the results of PSNR and MSE between LSB with RSA and Hash LSB with RSA. The Hash LSB has a higher PSNR value and a significantly lower MSE, meaning that the stego image has a good quality and a small dissimilarity.

**“A proposed Method for Image Steganography using Edge Detection.”** By Arora and Anand[29]

Arora and Anand proposed an alternative spatial domain embedding technique by implementing the image edge detection. The system model is divided into four phases: edge detection, randomization of edge pixels, embedding of confidential data and extracting confidential data.

The embedding process is implemented in three steps. Firstly, the image edge will be detected through scanning in a 3 x 3 pixels window, the edge pixels are in an array (*Array\_Cover*). Secondly, in order to enhance the security of steganography, the sorting approach is implemented to randomize these pixels in *Array\_Cover*. Meanwhile, the confidential data is stored into another Array (*Array\_Message*). Thirdly, according to the details of the *Array\_Message*, the LSB replacement algorithm is executed

between the binary value of *Array\_Message* and the blue component of *Array\_Cover*, to generate a new Array (*Array\_Stego*). The new stego image will be made from the *Array\_Stego*.

With this method, an advantage comes from utilising the principle of Image Edge Detection to increase the message embedding capacity in the cover image. As the edge areas are harder to detect changes than the smooth areas using human eyes only, this means that more data can be stored in the edge areas without increasing the possibility of image distortion. However, the disadvantages are obvious as well, because in the extraction process, the receiver is required to possess a symmetric key for the randomization value for *Array\_Stego*, otherwise, the replaced edges cannot be identified.

**“A Novel Approach for Data Hiding using LSB on Edges of a Grayscale Cover Images.”** By Chaturvedi[34]

A novel approach was proposed by Chaturvedi to conceal data in a Grayscale Cover Image through LSB replacement algorithm on Edge Detection, which is called **EG-LSB**. In order to extract edges, two parameter values (*Mean* and *Standard Deviation*) were used and they were assigned to the first two pixels in the cover image, and then using the edge detection algorithm all of the edge pixel values were obtained. Unlike Arora, S and Anand, S' s algorithm [29], this method did not use the randomization method to locate the positions for embedding data. Instead, it utilised a horizontal scanning method, pixel by pixel to determine the position of the edge areas. The LSB 2 algorithm will be implemented in this pixel once its edge value is '1'. Moreover, the embedding process was not started at the first point as the first two pixels were reserved to hide the information of these parameters, *Mean* and *Standard deviation*. Otherwise, during the data extraction process, the receiver cannot obtain the same edge array to extract the message.

As shown in the following pseudocode:

(Assuming the size of cover image is  $M * N$ , the value of  $i$  and  $j$  indicate the current coordinate value of pixel.  $0 \leq i \leq M$ ,  $0 \leq j \leq N$ )

```
for i = 1; i <= M; i ++
    for j = 1; j <= N; j ++
        if (Edge (Cover Image (i, j)) == 1)
            execute LSB 2 in cover image (i, j)
        end
    end
end
```

Thus, in the EG-LSB algorithm, all of the first row and the first column pixels are reserved and the rest of pixels is used to embed the secret data, each 2 bits of data is replaced by 2 bits in the LSB of the cover image if this pixel is identified in the edge areas.

In their conclusion, the comparison between the EG-LSB algorithm and the LSB replacement algorithm was evaluated, and the measurement unit PSNR was set to a reference to verify the newly generated stego image. As the experimental results proved, the quality of images using EG-LSB is better than the LSB, although the value of PSNR has a slight increase.

***“A steganography method for images by pixel-value differencing.”*** By Wu and Tsai [35]

There are various ways to determine the areas of edges and smoothness in the images. Besides the several regular edge detection algorithms, the method of **Pixel Value Differencing (PVD)** is a good mechanism to distinguish these parameters as well. This was first proposed in 2002.

Wu and Tsai proposed a new steganographic method for grayscale images using Pixel Value Differencing. In a grayscale image, each pixel contains 8 bits grey-values, totalling 256 grey-values which indicates the colour from 0 (Dark) to 255 (White). In the PVD method, the cover image will be split into a number of non-overlapping two-pixel blocks. Then, these blocks will be categorized according to the difference of the two pixel values. A higher difference value indicates this block is located in the edge areas, whereas a lower difference value indicates the area is smooth. In this method, the same principle was used to conceal the data as mentioned in [29], the tolerance in the edge parts is more than smooth parts, thus, more data will be embedded into the edge areas.

During the embedding process, the pixels in the cover image will be scanned in a “**zigzag**” manner, and split into non-overlapping two-pixel blocks. The difference  $d$  can be computed from two consecutive pixels in each block. Assume those two pixels are  $p_i$  and  $p_{i+1}$ , and the value of these pixels are  $g_i$  and  $g_{i+1}$ . So,  $d = |g_{i+1} - g_i|$ . Using the value of  $d$  to classify boundaries into a number of contiguous ranges, the value of the lower bound  $l_i$  and the value of upper  $u_i$  can be obtained according to their Difference Range Table. Next, the number of bits  $n$  which is going to be embedded in this block is calculated by the equation  $n = \log_2 (u_i - l_i + 1)$ . The  $n$  bit(s) data is taken from the secret data and converted into a decimal format. A new difference  $d'$  is calculated from the value of  $d$  and  $l_i$ . Finally, the new grayscale pixel can be obtained through a new function which works on the new difference value  $d'$ . The stego image can be generated from these new pixel values in the cover image.

In conclusion, Wu and Tsai listed the different experimental results which was based on the different ranges width. The stego image not only has a very good result in both PSNR and MSE, but also can escape the detection from some advanced steganalysis methods, such as the RS attack detection [47].

**“Colour Image Steganography Based on Pixel Value Differencing in Spatial Domain.”** By Mandal and Das [36]

Subsequently, various image steganography approaches which are based on the PVD method have been proposed. In 2012, the authors used the PVD algorithm on a spatial domain colour image. Unlike the algorithm in [35], the pixel value is separated and stored into three different colour matrixes: Red matrix, Green matrix and Blue matrix. Each block’s pixel is stored related to each colour matrix respectively, from the beginning until the end. The value of difference  $d$  is computed for each block of two consecutive non-overlapping pixels, assume the pixels are  $p_i$  and  $p_{i+1}$ ,  $d = |p_i - p_{i+1}|$ . Then, the optimal range  $R_i$  can be obtained through their Difference Range Table<sup>2</sup>, and the amount of confidential data ‘ $n$ ’ which will be embedded into the cover image is computed.

Also, the data is not embedded into all of the blocks in this method, the different thresholds are set for each colour matrix as the reference. Next, the value of  $n$  in each colour matrix is used to determine whether the message data can be embedded in this block through comparison with the related references. Such as: In the Red matrix, if  $n \leq 5$ ,<sup>3</sup> then execute the next step, otherwise, no data will be embedded in this block. In the Green matrix, if  $n \leq 3$ , execute the next step, otherwise, no data will be embedded in this block. However, in the Blue matrix, whatever the  $n$ ’s value is, execute the next step directly. The pseudo code was shown in below:

```
In Red:
    If  $n \leq 5$ 
        LSBfunction();
In Green:
    If  $n \leq 3$ 
        LSBfunction();
In Blue:
    LSBfunction();
```

---

<sup>2</sup> as same as the previous Difference Range Table in paper [12].

<sup>3</sup> The authors defined the reference value 5 in the red matrix, and 3 in the green matrix.

Here, pick  $n$  bit(s) of data from the secret data and convert them into a decimal value  $b$ . The new difference value  $d'$  can be computed from the equation:  $d' = l_i + b$ , also the new pixel  $(p_i', p_{i+1}')$  (after the data embedding) can be generated through the function of the original PVD method. The previous pixel  $(p_i, p_{i+1})$  is replaced by this new pixel  $(p_i', p_{i+1}')$  and will generate a new block. In the final step, in order to keep the information whether  $n$  bits or  $n-1$  bits have been embedded, the relevant add or subtract operation is executed from the value of LSB in  $p_i$  and  $p_{i+1}$ , which then generates the new stego-image.

In the final part of this paper, the evaluation of this algorithm was presented which compared the original PVD method and this method through calculating the value of PSNR. In this method, although the algorithm can be executed for both the grayscale images and colour images, the value of PSNR has a trivial decrease.

***“Colour Image Steganography based on Pixel Value Modification Method Using Modulus Function.”*** By Nagaraj, Vijayalakshmi and Zayaraz [37]

One more colour image steganographic algorithm was proposed in 2013, which was based on using a modulus function to implement the method of Pixel Value Modification (PVM). Unlike other steganography algorithms, the confidential data is not converted to binary form in this case. Also, the value of the cover image pixel can be implemented the embedding process in decimal form directly without any conversions.

During the process of embedding the data, the cover image is separated into three different colour matrices to store the related colour pixels, from the first to the last. The confidential data is converted into base 3 with values of (0, 1, 2) rather than in binary form; it is called  $d$ . Then, all the values within the different colour matrices execute the function modulo 3, and the result is recorded as, say,  $f$ . Next, the value of  $f$  and  $d$  is used to evolve a new value of this pixel. For example, in the red colour matrix, assume the value of this pixel is  $P_r$ , if  $f = d$ ,  $P_r$  will stay the same value as previously. If  $f < d$ ,  $P_r$  will increase by 1 to obtain the new value  $P_r'$ ; if  $f > d$ , the new value  $P_r'$  can be generated through decreasing by 1. Other colour matrices follow the same rules to obtain the new value through modifying the previous value. Finally, the new pixel can be combined from those matrices, to generate a stego-image.

As shown in Figure 12, converting the secret data  $d$  into base 3 gives the value of “10212...”. The value of an individual pixel is, R: 226, G: 137, B: 126. Using the modulus function to calculate the result of  $f$

respectively, then combining the *result* (0,2,1) with the converted secret data *d* (in this example only the top three values were used, 1, 0, 2) and the result is the new pixel, R:226, G:136, B:127.

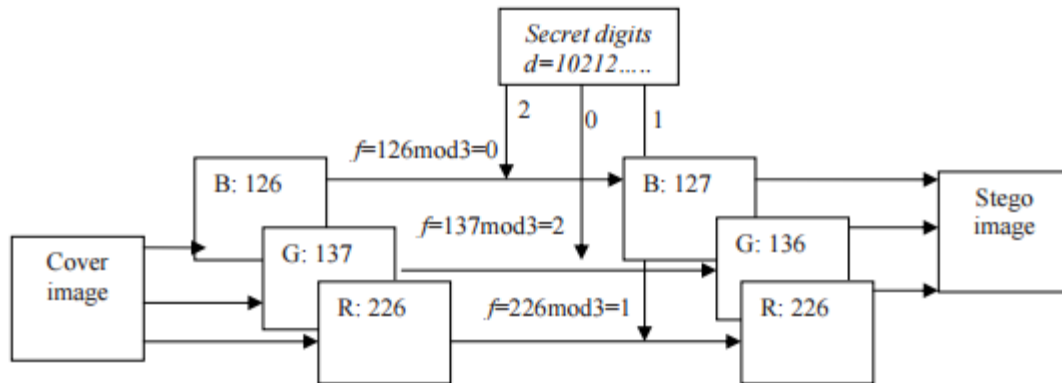


Figure 12: The illustration of PVM algorithm in the embedding procedure [37]

During the procedure of data extraction, the data can be extracted directly using modulo 3 in the value of each colour channel. As shows in Figure 13, the data (1, 0, 2) can be calculated through modulus 3 directly from the new value, R:226, G:136, B 127.

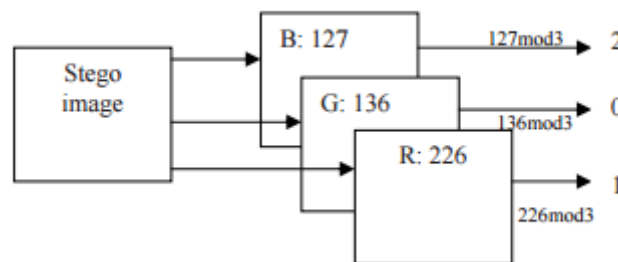


Figure 13: The illustration of PVM algorithm in the extraction procedure [37]

The evaluation was listed comparing both the values of Payload (embedded capacity) and PSNR with the original PVD. This method has an outstanding quality and capacity, but also has an efficient computation time as the procedure of the conversion of the binary form in cover image has been eliminated.

**“A Novel Image Steganography Method with Adaptive Number of LSB Modification Based on Private Stego-Keys.”** By Jain and Ahirwal [38]

A novel LSB substitution algorithm based on the utilization of private keys was proposed for grayscale images. In this method two private keys were mentioned; the first key ( $K_1$ ) is used in both the

procedures of the data embedding and extraction, and the second key ( $K_2$ ) is used to verify the integrity of the confidential data for the receiver.

Firstly, the private  $K_1$  consists of **five** different grey value ranges that are selected from the range  $0 \sim 255$  randomly, say  $K_1: \langle S_1 - E_1, S_2 - E_2, S_3 - E_3, S_4 - E_4, S_5 - E_5 \rangle$ . Where,  $S_1$  indicates the first range of the start value;  $E_1$  indicates the first range of the end value. The difference  $D$  can be obtained by equation:  $D = E_i - S_i$  ( $1 \leq i \leq 5$ ). The value of  $D$  is a reference to confirm the correctness in each range, because the difference of two consecutive ranges cannot be less than 32, also anyone range cannot overlap others.

Next,  $n$  bits of the secret data can be determined for insertion into each pixel. Firstly, according to the details of  $K_1$ , the related number of pixels can be counted in each range from the cover image, denoted by  $\langle N_1, N_2, N_3, N_4, N_5 \rangle$ . Then by sorting by value, the largest range can have 5 bits of data replaced, the second largest can be embedded in 4 bits data and so on.

For example, assume the counting of pixel in each range is  $\langle N_1 = 250, N_2 = 430, N_3 = 90, N_4 = 3500, N_5 = 100 \rangle$ , then after sorting these ranges, the order is:  $N_4 > N_2 > N_1 > N_5 > N_3$ . Therefore, the first range  $S_1 - E_1$  can replace 3 bits of the message data in the pixel, the second range  $S_2 - E_2$  is 4 bits, the third range  $S_3 - E_3$  is 1 bit, the fourth range  $S_4 - E_4$  is 5 bits and the fifth range  $S_5 - E_5$  is 2 bits.

The LSB substitution algorithm will be executed subsequently, the related bit(s) will be replaced with message data in each pixel. However, the newly generated pixel has to meet one of these ranges, meaning that the new value cannot exceed the boundary (end value) of the range, otherwise, one more adjusting algorithms will be used for modifying the value to fall within the range.

Finally, the signature  $K_2$  is confirmed to verify the integrity of the message through a simple **XOR** method with a random key of 140 bits, and this is appended with the message.

In conclusion, the evaluation of this method was listed through comparison with other algorithms, such as *LSB 4* and other adaptive methods. In the grayscale image, their proposed method not only has a superb image quality with the highest PSNR value, but it also has the largest embedding capacity than all the others. However, the major drawback is the security issue that occurs from the transmission of both private keys.



**“Adaptive data hiding in edge areas of image with spatial LSB domain systems.”** By Yang, Weng, Wang and Sun [39]

A derived algorithm from the original PVD method was implemented for concealing data in edge areas. In this method, the Pixel Value Differences are still used to determine the position of edge areas and smooth areas. Three levels are defined to identify the different ranges, the **Lower level**, the **Middle level** and the **Higher level**. The LSB substitution algorithm is executed on all of two consecutive pixels; the value of a replaced bit  $n$  is subject to the classification of the level as a higher level can be embedded with more characters.

The colour intensity range  $[0, 255]$  is divided into two different categories, which are called **lower-higher (l-h)** level and **lower-middle-higher (l-m-h)** level, as shown in Figure 14.

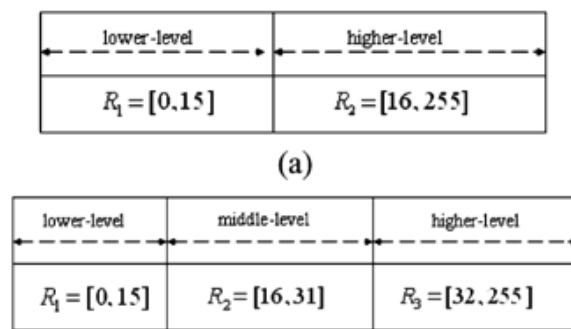


Figure 14: The description of l-h level and l-m-h level

Hence, under *l-h* level,  $R_1 \in [0, 15]$ ,  $R_2 \in [16, 255]$ ; under *l-m-h* level,  $R_1 \in [0, 15]$ ,  $R_2 \in [16, 31]$ ,  $R_3 \in [32, 255]$ . The length of range  $|R|$  can be obtained, for example, in *l-h* level,  $|R_1| = 16$ ,  $|R_2| = 240$ . Then, the value of replaced bit  $n$  can be calculated by the below equation:

$$n \leq \log_2 |R|$$

For each different level, the number of  $n$  has a different result. In *l-h* level,  $n_1 = 4$ ,  $n_2 = 7$ . In *l-m-h* level,  $n_1 = 4$ ,  $n_2 = 4$  and  $n_3 = 7$ . In the last step, in order to avoid the distortion in a stego image, the adjusting phase is executed to reduce the value of this  $n$ . Finally, in the division of *l-h* level,  $n_1 = 2$ ,  $n_2 = 3$ . In the division of *l-m-h* level,  $n_1 = 3$ ,  $n_2 = 4$  and  $n_3 = 5$ .

During the embedding procedure, every two non-overlapping pixels are scanned in a raster way (horizontal direction scanning) from the cover image, all of the pairs of pixels are stored into different blocks, and the differences  $d$  in these blocks can be gathered. Next, all of the blocks are defined into relative levels due to the value of  $d$ , and the number of replaced bits  $n$  can be confirmed for each pixel. By using the LSB substitution algorithm, to pick the corresponding  $n$  bit data from the message, and replacing it with the cover image to generate the new block. The new pixel difference  $d'$  in this new

block must belong to the same level with previous difference  $d$  in the division. Otherwise, an adjusting algorithm will be executed to alter the new block to fall within the previous level, then the stego image can be generated from all of these new blocks.

In their conclusion, the quality of the stego-images was evaluated through the comparison of PSNR that is based on the variable  $n$ . As their report showed, with a decrease in the value  $n$ , the result of PSNR rises significantly. In the division of  $l-h$  level, the best image quality is when  $n_1=2$  and  $n_2=4$  (represent by 2-4). Also, the results in the original PVD,  $l-m-h$  division (3-4-5) and  $l-h$  division (3-4) were compared, the quality of stego-image is in the sequence:  $l-h > l-m-g > PVD$ .

***“Image Data Hiding Method Based on Multi-Pixel Differencing and LSB substitution Methods.”*** By Jung, Ha and Yoo [40]

In 2008, one more derived algorithms from the original PVD method was proposed by Jung, Ha, and Yoo, which was based on the combination of the optimized Multi-Pixel Differencing (MPD) and LSB replacement methods. Unlike previous Pixel-Differences algorithms, in this method, the blocks are collected into groups of four consecutive non-overlapping pixels from the grayscale cover image, and the pixel value is sorted from the lowest to the highest. Assume a block consists of pixels with  $p_i, p_{i+1}, p_{i+2}$ , and  $p_{i+3}$ , the grey value of them are  $g_0, g_1, g_2$  and  $g_3$  respectively, also the sequence is  $g_3 > g_2 > g_1 > g_0$ .

During the embedding procedure, the differences  $d$  in each pixel-pair of four consecutive non-overlapping pixels are computed first, as shown in equation below:

$$d_i = |g_i - g_0|, \text{ Where } i \in [1, 2]$$

In the second step, the areas between edge and smooth will be categorized through comparing the sum of differences  $D$  in the block with a defined threshold. Where the sum  $D = d_1 + d_2 + d_3$ , and the threshold value is set at  $T$ .<sup>4</sup> If  $D < T$ , then this block is in the smooth areas, otherwise, it is defined in the edge areas.

Next, the  $n$  bits of confidential data can be confirmed to be embedded into  $g_i$  by using the LSB replacement algorithm, where  $i \in [1, 2]$ . Two cases should be considered:

---

<sup>4</sup> The Author defined this threshold value, which is 6.

- **Case 1:** if this block is located in the smooth areas, pick 9 bits of data from the confidential message, also execute *LSB 3* algorithm to replace the corresponding pixel in the cover image, and generate a new block.
- **Case 2:** if this block is defined in the edge areas, the number of  $n$  can be computed with a defined range *Tables* by equation:  $n = \log_2(u_i - l_i + 1)$ .  $u$  indicates the upper bound, and  $l$  denotes the lower bound.

Then, the value of new difference  $d'$  can be computed by equation:  $d'_i = d_i + b_i$  for each  $g_i$ , where  $b_i$  is the decimal value for the embedded confidential message bits, and  $i \in [1, 2]$ . Finally, the optimize function is executed in the edge block, and then a new block will be obtained, and the stego-image can be generated from these new blocks.

In the conclusion, this algorithm was evaluated through comparing the value of PSNR and embedded capacity with the LSB 2, original PVD and MPD. As the report showed, the largest embedding capacity is in their optimized algorithm, although the PSNR is the lowest in all four of them.

**“Image Steganography using Pixel-Value Differencing.”** By Hanling, Guangzhi and Caiqiong [41]

In 2009, another algorithm based on Pixel Value Differencing steganography method was presented by Hanling, Guangzhi and Caiqiong. The amount of the insertion bits refers to the depth distribution of the cover image, the message data in the edge area can be concealed more than in the smooth areas. However, unlike the other PVD methods, this method is only modifying one target pixel for each block rather than all of the pixels.

During the embedding procedure, the pixel blocks are gathered by using raster-scan in the cover image, each block consists of four non-overlapping pixels, one target pixel and three neighbouring pixels.

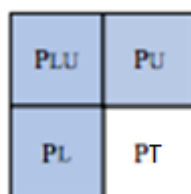


Figure 15: The illustration of pixel blocks

---

<sup>5</sup> The Author also defined this range Table, where  $R_1[0, 7]$ ,  $R_2[8, 15]$ ,  $R_3[16, 31]$ ,  $R_4[32, 63]$ ,  $R_5[64, 127]$  and  $R_6[128, 255]$ .

Therefore, assume, in this block, the four pixels are  $p_t$  (target pixel),  $p_u$  (the upper of  $p_t$ ),  $p_l$  (the left of  $p_t$ ), and  $p_{lu}$  (the upper-left of  $p_t$ ) as shown in the Figure 15 above. The corresponding grey values are denoted as  $g_t, g_u, g_l$  and  $g_{lu}$  respectively. Next, the differences  $d$  between the Maximum and Minimum in this block can be computed through the equation below:

$$d = \text{Max}(g_u, g_l, g_{lu}) - \text{Min}(g_u, g_l, g_{lu})$$

then, the amount of insertion bits 'n' can be obtained from the value of d, as shows in formula below:

$$\text{If } d \in [0, 1], n = 1$$

$$\text{If } d \in (1, +), n = \log_2 d$$

In order to avoid the distortion in the stego images,  $n$  will be set 4 when the value of  $\log_2 d > 4$ ; picking the corresponding  $n$  bits from confidential data and converting them to decimal form, denoted in  $b$ . Then, the new pixel  $g_t'$  can be calculated by the equation:

$$g_t' = g_t - g_t \bmod 2^n + b$$

Finally, before generating the stego-image, in order to reduce the **Embedding Error** between  $g_t$  and  $g_t'$  within a defined range<sup>6</sup>, an optimal adjustment process will be executed to alter the newly generated pixel.

In their conclusion, the evaluation of this method was compared with Chang's [61] method and Park's [62] method respectively through calculating the PSNR and Embedding Capacity. This novel method not only has a highest PSNR, but can also embed the most data out of the three techniques.

***“High payload steganography mechanism using hybrid edge detector.”*** By Chen, Chang and Le [42]

The hybrid edge detection mechanism was presented to locate the position of edge areas and be used with the LSB replacement algorithm to conceal secret data in the grayscale image. The authors implemented the relevant edge detection experiment based on three different detection algorithms, such as *Fuzzy Edge Detector*, *Canny Edge Detector* and *Hybrid Edge Detector*. Through comparing the amount of edge pixels to state the reason of using hybrid detection to conceal data, as in the edge transforming rates, the order was denoted as *Hybrid > Canny > Fuzzy*.

During the embedding procedure, there are three phases:

---

<sup>6</sup> The author defined this range to  $(-2^n < g_t < 2^n)$ .

- **Phase 1:** using a hybrid detection algorithm to generate an edge image from the original grayscale cover image.

- **Phase 2:** dividing the edge image into a set of blocks, and assume each block contains  $N$  pixels, from  $p_1$  to  $p_N$ . Also, in each block, the first pixel ( $p_1$ ) is used to store the edge information of remaining pixels (besides  $p_1$ ) in this block by using the LSB replacement algorithm.

For instance, assume a block contains four pixels,  $A=[p_1, p_2, p_3, p_4]$ , where  $N=4$ . If  $p_2$  and  $p_4$  are both located in edge areas only, the edge information (besides  $p_1$ ) is defined to "101". Then this value '101' will be replaced into  $p_1$  through the LSB substitution algorithm. Generally, in order to increase the embedding capacity and protect the quality of stego image, the  $N$  is suggested to be set as [3,5].

- **Phase 3:** according to the edge information, the replaced bit  $n$  can be considered respectively, as the edge areas can be embedded more data than smooth areas in the image. However, in order to avoid causing the distortion in the stego image, the  $n$  is set to [3,5] in edge areas and [1,2] in smooth areas. Then, the corresponding  $n$  bits LSB replacement algorithm will be executed in the remaining pixel (beside  $p_1$ ) for each block, if the pixel is located in the edge areas, the range of  $n$  will be picked as [3,5], otherwise, [1,2] will be chosen.

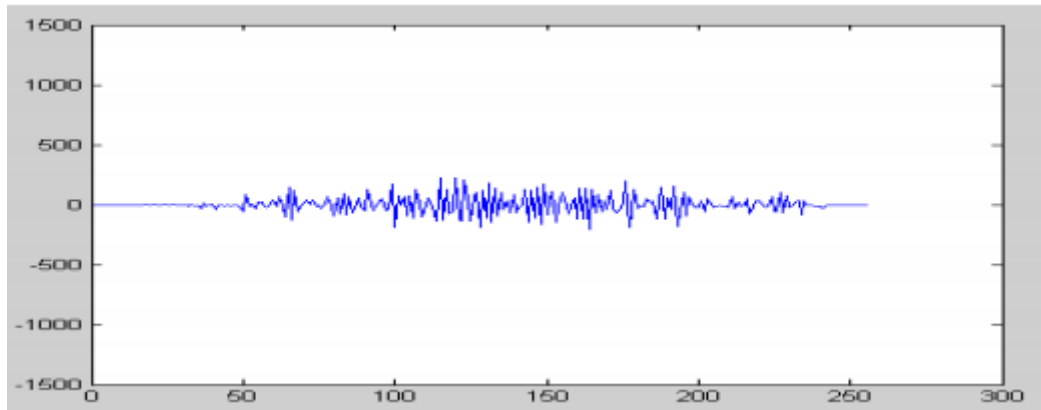
In their conclusion, the comparison between this method and original LSB replacement method was evaluated, the proposed method has a higher Embedding Capacity and better PSNR value. Especially in the original LSB replacement algorithm, the stego image has an appreciable distortion since the replaced bits  $n$  is up to 5 or 6. However, for the proposed method, the quality of stego image is kept consistent, even if the value of replaced bit  $n$  is up to 5 in edge areas. Also, the evaluation showed that  $n$  was optimal on [3,5] in edge area, [1,2] in smooth area. Otherwise, the stego image will be caused perceptible distortion.

***"A New LSB Matching Steganographic Method Based on Steganographic Information Table."*** By Qiudong and Liu [43]

A novel LSB matching algorithm based on Steganographic information table was presented. In the beginning of this paper, the limitations of LSB Matching algorithm were analysed, although the LSB matching has a higher robustness in human visual detection and its histogram change is more imperceptible than the result of the LSB replacement algorithm. The histogram of the stego image is still distorted from the cover image if the embedding rate is too high a value, and the peak value has a significant smoothness over the cover image. In this proposed method, the previous limitation of

LSB matching algorithm is improved by using a steganographic information table to reserve all of the histogram characters.

In the original LSB matching algorithm, the resultant differential noise between the cover image and stego image has an obvious 'ragged' distribution in the histogram, as shown from the paper's Figure 16 [43],



*Figure 16: The noise difference histogram between the original cover image and the stego image (without table) [43]*

In order to generate a stable noise histogram between the cover image and the stego image as much as possible, the 'Steganographic information record table' was presented to record the result of pixel changes during the algorithm embedding procedure. In this table, all operations where the pixel was changed would be recorded dynamically step by step. The table will be accomplished after all of the confidential data are embedded into the cover image.

In this proposed method, during the embedding procedure, if the bit of message is equal to the LSB of the cover image, this pixel will stay the same in the cover image. Otherwise, it will be added or subtracted by one according to the resultant differential noise in the 'Steganographic information record table', which guarantees to reduce the difference of noise histogram to the lowest in each time operation.

In the conclusion, the noise histogram result between original LSB matching and new improved proposed method was compared, as shown in Figure 17. Although the proposed method has a negligible difference between the cover image and stego image, the computation time is very large as the steganographic information table has to update constantly in each time injection operation during the embedding procedure.

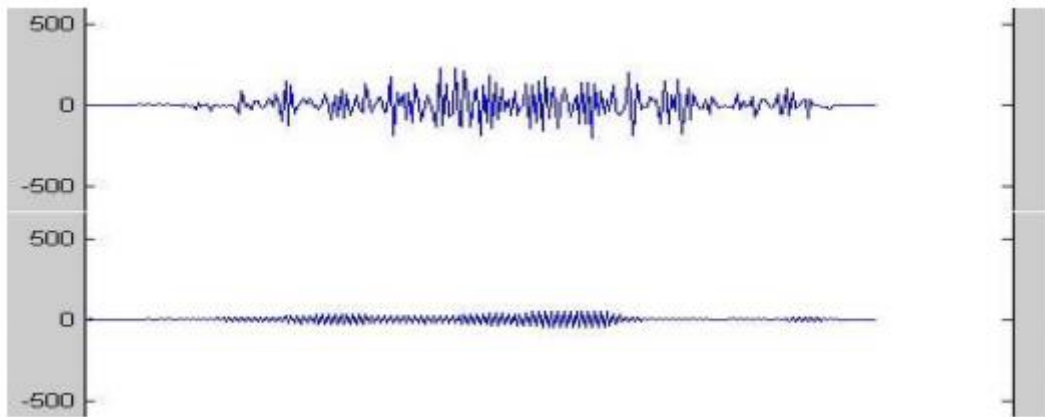


Figure 17: The noise different histogram comparison between without (top) and with (down) the presented stego table [43]

**“LSB Matching Revisited.”** By Mielikainen [44]

For the LSB matching algorithm, there are only a few detections proposed before 2006. However, in order to improve the original LSB matching algorithm for increasing the robustness in these detections, a new matching method based on two pixels at a time was presented which updated the original LSB matching algorithm.

The new method is used in the grayscale image. During the insertion procedure, the minimum unit is a pair of bits either in the cover image (two pixels) or the confidential data (two bits), and which would be picked to execute relevant operations each time in the embedding operation. Also, this new embedding algorithm cannot be embedded on saturated pixels, such as when the value is 255, or 0, which is the same as the original LSB matching algorithm. The specific operation is shown by the below pseudocode:

NB: two confidential message data denoted as  $M_i$  and  $M_{i+1}$ ; two cover image pixels are referred as to  $p_i$  and  $p_{i+1}$ .  $p_i'$  and  $p_{i+1}'$  indicate the new pixel value in the stego-image. Also, the function  $F(p_i, p_{i+1})$  can be evolved by the equation below:

$$F(p_i, p_{i+1}) = LSB(\lfloor \frac{p_i}{2} \rfloor) + p_{i+1}$$

(NB: [...] indicates the largest integer less or equal to the value that is in these two brackets.)

```

if  $M_i = LSB(p_i)$ 
    if  $M_{i+1} \neq F(p_i, p_{i+1})$ 
         $p_{i+1}' = p_{i+1} \pm 1$ 
    else

```

```

         $p'_{i+1} = p_{i+1}$ 
    end
     $p'_i = p_i - 1$ 
else
    if  $M_{i+1} \neq F(p_i - 1, p_{i+1})$ 
         $p'_i = p_i - 1$ 
    else
         $p'_i = p_i + 1$ 
    end
     $p'_{i+1} = p_{i+1}$ 
end
end

```

In their conclusion, the robustness was compared between the original LSB matching algorithm and this proposed method through implementing the relevant LSB matching detection in both algorithms with 1,000 images. As the result shows, this new method has a relatively lower probability of detection curve, as shown in Figure 18:

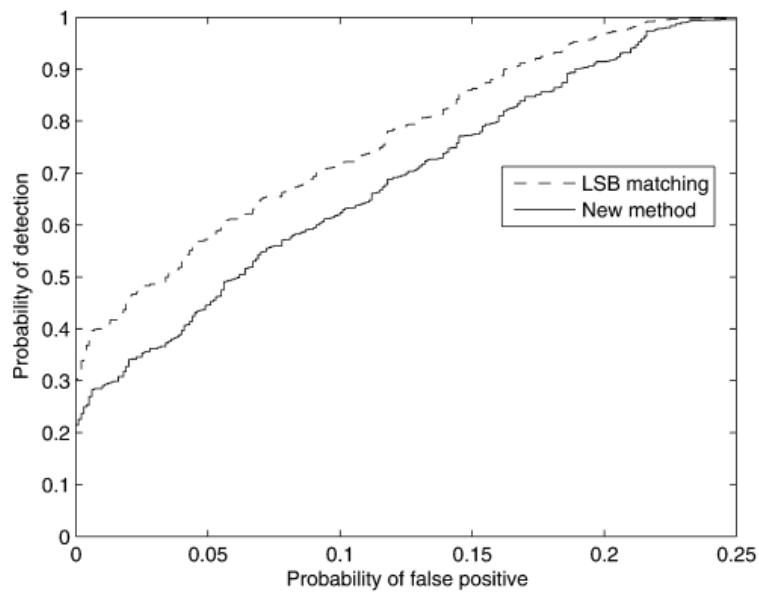


Figure 18: The resistance results comparison between the LSB matching and new proposed method [44]



**“A Generalization of LSB Matching.”** By Yang, Cheng and Zeng [31]

A novel LSB matching algorithm that generalized the LSB matching and LSB matching revisits was first proposed in 2008, and called the generalised LSB matching scheme (*G-LSB-M*). Also, in this paper, the concept of the *Expected Number of Modification Per Pixel (ENMPP)* has been proposed to identify the resistance in relevant steganalysis. As shown in Figure 19:

$$\mathbb{E}(f_{emb}^n, f_{ext}^n) = \frac{1}{n2^n 254^n} \sum_{x \in I_{0,n}} \sum_{w \in \mathbb{Z}_{2^n}} \frac{\sum_{y \in f_{emb}^n(x,w)} \|y - x\|_{l^1}}{\#f_{emb}^n(x,w)}$$

Figure 19: The equation of Expected Number of Modifications Per Pixel [31]

The embedding procedure was compared in the *G-LSB-M* with other algorithms. In the original LSB matching algorithm, the pixel can be added or subtracted randomly by one while the message data does not match with this pixel. The considered unit is every single pixel, and the value of ENMPP is [0.5.7](#)

However, in the LSB matching revisits algorithm, each time the embedding picks two bits of the message data, it compares them with two LSB pixels in the cover image; the operation on the pixel, whether added or subtracted, depends on the algorithm’s regulation table. Thus, the implementation units are up to a pair either in the cover image or in the secret message; the value of ENMPP is decreased to [0.375](#).

In this new *G-LSB-M* algorithm, the value of ENMPP was reduced again through increasing the consideration unit, which extended up to 3. Three bits are embedded in a ternary pixel array, the relevant operations of each pixel, whether added or subtracted, depends on its algorithm regulation function as well. The value of ENMPP was verified to be [1/3](#). Therefore, the new method *G-LSB-M* can generate a more secure stego image with a higher robustness for detection through reducing the value of ENMPP.

In their conclusion, the resistance is verified through implementing 5000 images in different LSB matching detections. As the experiment showed, the *G-LSB-M* algorithm has a lower probability of positive detection especially with the value of consideration unit (both in the cover image and the secret image) increasing, the value of probability of detection will be reduced. This is shown in Figure 20:

---

<sup>7</sup> This value was computed by author in his paper, as well as the 0.375 in LSB matching revisits algorithm.

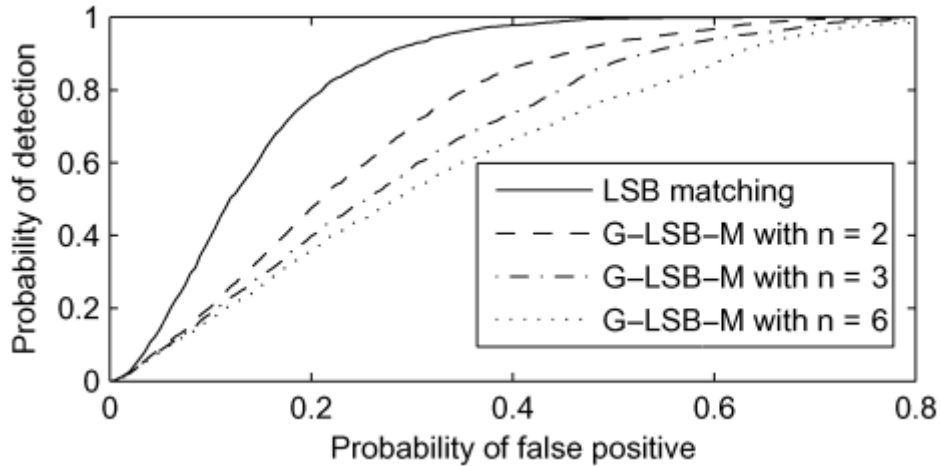


Figure 20: The resistance result comparison in LSB matching and variable  $n$  in G-LSB-M [31]

(NB: in this Figure 20, the  $n$  indicates the value of the considered unit)

**“Lossless Data Hiding Scheme Based on LSB Matching.”** By Quan and Zhang [45]

In 2013, another simple reversible LSB matching algorithm was proposed through implementing the concept of parity feature to a pair subset in the cover image. This new algorithm not only can be applied with a lower implementation complexity, but also can be executed in multiple cover files, such as image, video and audio.

Prior to discussing the embedding procedure, a secret key  $K$  was defined, and it will be shared between the sender and receiver for the process of data extraction. The cover image can be divided into two equal subsets ( $A$  and  $B$ ) due to the information in  $K$ . In the subset  $A$ , the corresponding embedding message pair is defined as  $(2k, 2k+1)$ . Instead, the corresponding message pair is defined as  $(2k-1, 2k)$  in the subset  $B$ .

During the embedding procedure, the pixel can only be modified when the message bit does not match (equal) the LSB of cover image. The relevant operation whether added or subtracted is considered by the position of this pixel. If the pixel is in the subset  $A$ , the odd pixel value  $2k + 1$  is decreased by one, and the even pixel  $2k$  is increased by one. Instead, in the subset  $B$ , the odd pixel value  $2k - 1$  is increased by one and the even pixel value  $2k$  is decreased by one, as shown in the pseudocodes below:

(Assume:  $M_i$  indicates the embedding message bit;  $p_i$  indicates the value of target pixel;  $p_i'$  indicates the value of stego-image pixel)

```

if  $M_i \neq \text{LSB}(p_i)$ 
  if  $M_i = 1$ 
    if  $p_i \in A$  (subset)
       $p_i' = p_i + 1$ 
    else
       $p_i' = p_i - 1$ 
  else
    if  $p_i \in A$  (subset)
       $p_i' = p_i - 1$ 
    else
       $p_i' = p_i + 1$ 
end

```

The resistance and quality were compared between this algorithm and the generalized LSB matching algorithm. As the experiment result displayed, the proposed algorithm has a higher quality in the stego-image as the value of *PSNR* and *MSE* are stable around 51.45 and 0.5 respectively. Moreover, for this new algorithm, the resistance in *RS* detection (*RS* detection will be explained later) has a significant increase, only with a quite low probability of detection.

### 2.2.3. Summary of Spatial Domain Image Steganography

According to the explanation of the algorithms above, a summary was concluded as below:

Algorithm Name	Principle	Benefits	Drawbacks	Author
<b>LSB 2/3</b>	Red channel exchanges 2 bits, Green and Blue channel exchange 3 bits separately.	Good quality in stego image, and lower memory for computation during the embedding procedure.	Only work on colour image	Mode, Abed et al.
<b>Hash-LSB</b>	Message is encrypted, then perform hash function to identify the exchange bits in each colour channel.	Stego image has good quality and small dissimilarity. Message is more secure as encrypted by RSA.	Only work on colour image.	Kumar and Sharma.

<b>Randomization edge detection</b>	Use edge detection to sort the edge pixels through 3 * 3 pixels windows, randomize these pixels and perform LSB algorithm	More data can be stored in edge areas without image distortion, edge areas are harder to be detected than the smooth areas.	A symmetric key is required in the extraction, as the edge pixels were randomized.	Arora and Anand.
<b>EG-LSB</b>	Identify edge pixel by using horizontal scanning method, perform LSB 2 in edge pixels, reserve first two pixels to store the Mean and Standard Deviation.	Good image quality in stego image, and the measurement unit PSNR is better than LSB replacement algorithm.		Chaturvedi.
<b>Pixel Value Difference. (PVD)</b>	The image is split into a number of non-overlapping two-pixel blocks, according to the difference value in each block, to identify the exchange bit.	Very good measurement unit result in both PSNR and MSR, even the stego image can escape the detection of RS attack.	Only works on grayscale image.	Wu and Tsai.
<b>Colour Pixel Value Differencing.</b>	The same as PVD, but the pixel value is separated and stored into three different colour matrixes (RGB), according to the value difference in each block in each colour, to identify the exchange bit	Improved the original PVD algorithm, to make it can work on both grayscale and colour image.	Measurement unit PSNR has a trivial decrease.	Mandal and Das.
<b>Pixel Value Modification. (PVM)</b>	All the pixel value within the different colour matrices perform the function modulo 3, according to the result, to identify the exchange bit.	The stego image has a better quality and embedding capacity than PVD algorithm, and the computation time is lower.	Only works on colour image.	Nagaraj, Vijayalakshmi and Zayaraz.

<b>LSB modification based on private stego-keys</b>	Two keys are defined, one is used in embedding and extraction procedure; the other is used to verify the data integrity. The exchange bit is identified by the range of colour value difference.	Compare to LSB 4, this algorithm has a superb image quality with the highest PSNR value.	Only works on grayscale image, and the major security issue that is the transmission of both private keys	Jain and Ahirwal.
<b>A derived algorithm from PVD.</b>	Define three different level to identify the exchange bit in edge areas.	Has a better image quality than original PVD algorithm.		Yang, Weng, Wang and Sun.
<b>Multi-Pixel Differencing. (MPD)</b>	Use for consecutive non-overlapping blocks to collect the pixels from the image, according to the difference of pixels value in each block, to identify the exchange bit.	Compared to LSB 2, original PVD and MPD algorithm, this algorithm has largest embedding capacity.	The measurement unit PSNR is the lowest in all four of them.	Jung, Ha and Yoo.
<b>Another algorithm based on PVD.</b>	The pixel blocks are gathered through four non-overlapping pixels, and define one target pixel and three neighbouring pixels in each block. Only the target pixel will be exchanged according to the difference between the Max value and Min value in each block.	Good quality in the stego image. Compared to Change's [61] and Park's [62] algorithm, this algorithm has the highest PSNR and a largest embedding capacity.		Hanling, Guangzhi and Caiqiong.
<b>Hybrid edge detection algorithm</b>	Edges will be gathered into a set of blocks uniformly by using a hybrid detector algorithm. Reserve the first pixel to record the exchanging statement, and	This method has a higher embedding capacity and better PSNR value than original LSB		Chen, Chang and Le.

	the rest pixels perform the LSB algorithm	replacement algorithm.		
<b>Stego information Table.</b>	Use a stego information table to record and consider the matching manipulation each time.	Compared to original LSB matching algorithm, this algorithm has a better PSNR, and reduced noise in the histogram as much as possible	Very large computation time during the embedding procedure.	Qiudong and Liu.
<b>LSB Matching Revisited</b>	Unlike original LSB matching algorithm, two pixels are considered in each matching manipulation in this new algorithm.	Improved the robustness from the original LSB matching algorithm,	Only works on grayscale image.	Mielikainen.
<b>Generalised LSB matching scheme. (G-LSB-M)</b>	Improved the embedding manipulation from LSB matching revisits. The consideration unit is increased up to 3. Three bits are embedded in a ternary pixel array.	The stego image has a lower probability of positive detection and it has a higher robustness than LSB matching revisited algorithm.		Yang, Cheng and Zeng.
<b>Lossless data hiding based on LSB matching.</b>	Define and share a secret key between sender and receiver. Image is divided into two equal subsets duo to the key. Then perform the relevant matching manipulation according to the message bit and target pixel value.	Compare to G-LSB-M algorithm, this method has a higher quality in both PSNR and MSE. Also, this method can be executed in multiple cover file, such as video and audio.	The transmission of secret key is a security issue.	Quan and Zhang.

Spatial image steganography algorithms were reviewed within this sub-section. Although the algorithms are based on different ways to conceal the secret data, all of them utilise the redundancy

of colour intensity in human eyes. Some algorithms embed the data to the identified edge areas in the image by using relevant edge detections. Certain algorithms are using the difference of consecutive non-overlapping pixels to hide data accordingly. Also, some algorithms are implemented based on the feature of the colour intensity. For example, the author *Parvez* [46] offered a novel algorithm based on the Intensity of the colour image; the embedded data can be stored more in the lower colour value areas due to its negligible influences. As the details of his Distortion Ratio experiment showed, the difference between the higher intensity and lower intensity has been examined, the change of higher intensity value colour has an obvious distortion ratio than that of the lower value colour.

Therefore, for all spatial image steganography algorithms, the common principle is based on changing the pixel value in the cover image to ensure effective data hiding.

#### 2.2.4. Frequency Domain and its Stego-Algorithms

According to the category of image presentation domain, the image can be presented both through the Spatial domain image and Frequency domain image systems. Normally, the spatial domain image consists of BMP, PNG and GIF; the frequency domain image predominantly contains JPEG, which is the most common online transmission image format [28].

For the spatial domain image, the digital content is stored in accordance with the format of the images colour intensity (colour value), ranged from 0 to 255, which is denoted as [0,255]. Colours are most frequently represented as additive combinations of RGB (Red, Green, Blue) in each pixel which is the fundamental unit for the image [57].

In the frequency domain, an image such as a JPEG image is presented as a transform domain, this format stores the coefficients of the cosines, representing the frequency of a value rather than storing the colour intensity that is assigned to each pixel. In other words, it means that the frequency domain image utilizes luminance and chrominance signals instead of RGB to represent an image. This is referred to as YUV. Y represents luminance, and the chrominance components (U and V) are the differences, as shown in the formula below [28]:

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = R - Y$$

$$V = B - Y$$

However, in order to adjust all of those components to the same range [0,255] by using 8 bits, the chrominance components (U and V) are further linearly transformed to Cr and Cb as they fall into the range [-179, 179], obtaining the YCrCb colour model [28].

Unlike the spatial domain, a JPEG image has a completely different compression process. As explained in the following, the procedure can be categorized into six phases [28]:

1. Colour transformation, from RGB to YCrCb. (Although this step is unnecessary as JPEG can also work directly with the RGB representation). It is typically used because it enables higher compression ratios at the same fidelity).
2. Divide image into 8 by 8 blocks.
3. Perform DCT (Discrete Cosine Transform) in 8 by 8 blocks.
4. Quantization of the DCT blocks.
5. Entropy coding.
6. Generate the JPEG image.

Thus, the steganography of a frequency domain image, requires an advanced algorithm to embed the data by using the LSB algorithm under the frequency value of each pixel rather than implementing the LSB algorithm under the colour intensity directly, such as with the JSTEG, F3, F4, F5 algorithms.

However, the JPEG compression can be seen as a lossy procedure, the part of redundancy data within the digital information will be lost in the phase of quantization, as shown in Figure 21 below [58]. Therefore, these based-frequency algorithms are often implemented to embed data into the quantized DCT table after the quantization phase. Furthermore, hiding data before the entropy coding stage (after the quantization stage) would not affect the data, as the process of “DCT” and “Quantization” is lossy, and “Entropy Coding” is lossless.

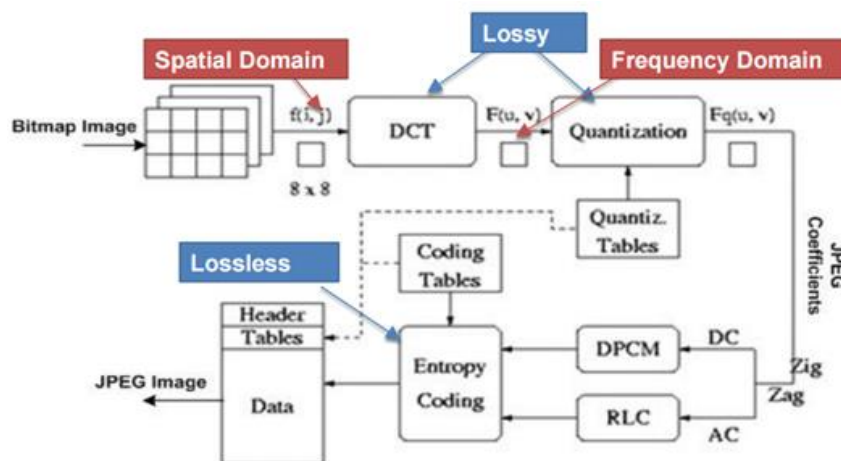


Figure 21: The illustration of JPEG compression [58]



## 2.3. Steganalysis

### 2.3.1. General Detections

Steganalysis is a solution that is implemented to detect the hidden messages in cover files. For image steganalysis, the common detection mechanism is mainly divided into two ways; visual and histogram analysis.

Visual detection compares the visual difference between an original image and a stego image. However, for advanced algorithms, it could utilize image redundant parts effectively, so that the difference cannot be detected by human eyes directly.

Histogram detection (similarly known as statistics detection) can be represented with a histogram chart. This chart represents the various values of each of the RGB (red, green and blue) colour values. The X axis represents the colour's value, from the minimum 0 to the maximum 255. The Y axis, represents the frequency of colours value. For example, in a 200 x 200 image, the pixel resolution is 40,000. This means that the amount of RGB variations contains a total of 40,000 pixels for an image, as shown in the formula below:

$$R: \sum_{i=0}^{255} Y(x = i) = 40,000$$

$$G: \sum_{i=0}^{255} Y(x = i) = 40,000$$

$$B: \sum_{i=0}^{255} Y(x = i) = 40,000$$

Therefore, the generated histogram counts the amount of each RGB colour from 0 to 255, and also presents the tonal distribution in digital image. The histogram detection compares the distribution of the value of RGB to distinguish the difference between an original image and a stego image.



Figure 22: The visual detection comparison between the original image and the stego image

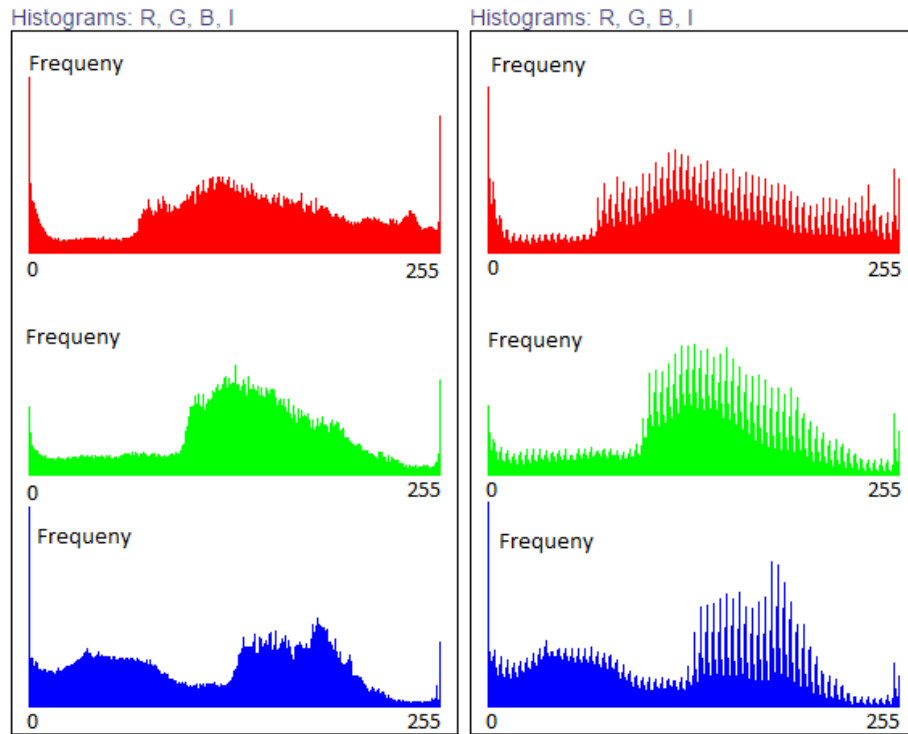


Figure 23: The histogram detection comparison between the original image and the stego image (with a replacement algorithm)

As shown in above Figures 22 & 23, a comparison can be seen between the original image and a stego image, the stego image concealed a TXT file with a file size of 100 KB. However, from the visual detection, it is hard to distinguish the difference, they are almost visually similar. From the histogram detection, the difference can be found absolutely, the hidden message has changed the colour's distribution significantly, as the histogram conveys a more 'ragged' result in its output for the stego image. Also the specific difference value can be obtained from the below formula. "Diff" represents the total differences, and  $diff_R$ ,  $diff_G$ ,  $diff_B$  represent the differences in red channel, green channel and blue channel respectively [43].

$$diff_R: \sum_{i=0}^{255} |Y_i - Y'_i|$$

$$diff_G: \sum_{i=0}^{255} |Y_i - Y'_i|$$

$$diff_B: \sum_{i=0}^{255} |Y_i - Y'_i|$$

$$Diff = diff_R + diff_G + diff_B.$$

In the Golfer's Bridge image in Figure 22, the difference in red is 59826, in green is 59050, in blue is 60478. Thus the total difference "Diff" is 179354.

Algorithms are being continually developed in steganography. The traditional histogram detection was not good enough due to the various advanced steganography algorithms, such as 'Steganographic

information record table' [43], which causes the embedding noise to be reduced, and makes the new histogram as much as possible with the original histogram. Therefore, new statistical methods are continuously being developed to account for better detection algorithms. In the next sub-section, the advanced steganalysis methods are discussed in more detail.

### 2.3.2. Advanced Detections

#### **“Reliable Detection of LSB Steganography in Colour and Grayscale Images.” (RS attack) By**

Fridrich, Binghamton, Goljan and Du [47]

A new reliable and extremely accurate steganalytic detection was presented for both colour and grayscale images. The concept of this method is based on the relationship between regular group pixels and singular group pixels, also it does not require the original cover image to be compared with the stego image.

During the detection procedure, in the target image, the pixels can be divided into different groups  $G$ , which consist of a constant number of the element  $n$  (NB: Number of pixels in each group). The discrimination function  $f$  can be used to distinguish the identification of each group  $G$ , as shown in the formula below: <sup>8</sup>

$$f(G) = f(X_1, X_2, \dots, X_n) = \sum_{i=1}^{n-1} |X_{i+1} - X_i|$$

Moreover, in order to set the category of each pixel group, the Flipping Function  $F$  was used before implementing the above discrimination function, the flipping result can be divided into three different situations due to the features of the LSB algorithm, such as  $F_1$ ,  $F_{-1}$  and  $F_0$ .

- In  $F_1$ :  $0 \leftrightarrow 1, 2 \leftrightarrow 3, 4 \leftrightarrow 5, \dots, 254 \leftrightarrow 255$ .
- In  $F_{-1}$ :  $-1 \leftrightarrow 0, 1 \leftrightarrow 2, 3 \leftrightarrow 4, \dots, 255 \leftrightarrow 256$ .
- In  $F_0$ :  $F_0(x) = x$ .

State a mask  $M$  with the value of  $[-1, 1]$ , import this  $M$  into each pixel group  $G$ . For example,  $G = (39, 38, 40, 41)$ ,  $M = (1, 0, 1, 0)$ ,  $-M = (-1, 0, -1, 0)$ . So:

$$F_m(G) = (F_1(39), F_0(38), F_1(40), F_0(41)) = (38, 38, 41, 41)$$

---

<sup>8</sup> This formula also represents the image smoothness of adjacent pixels

$$F_{-m}(G) = (F_{-1}(39), F_0(38), F_{-1}(40), F_0(41)) = (40, 38, 39, 41)$$

Next, the specific pixel category can be defined due to the above function  $f$  and operation  $F$ , as follows:

- **R**egular groups:  $G \in R \Leftrightarrow f(F(G)) > f(G)$
- **S**ingular groups:  $G \in S \Leftrightarrow f(F(G)) < f(G)$
- **U**nusable groups:  $G \in U \Leftrightarrow f(F(G)) = f(G)$

Finally, the parameters  $R_m$ ,  $R_{-m}$ ,  $S_m$  and  $S_{-m}$  can be obtained, where,  $R_m$  and  $R_{-m}$  indicate the number of regular groups for mask  $M$ ;  $S_m$  and  $S_{-m}$  indicate the number of singular groups for mask  $M$ . In this detection, the concept is based on the relationship between the number of the regular group and singular group.

In a normal image, there are:

$$R_m \cong R_{-m} \text{ and } S_m \cong S_{-m}$$

However, if the image has a message embedded, this relationship will be violated, and will change to:

$$R_{-m} - S_{-m} > R_m - S_m$$

Therefore, in this detection method, based on statistical detection, not only the result is accurate, but also the implementation process is quite easy as it does not require the cover image for reference.

**“Steganalysis for LSB Matching in Images with High-frequency Noise.”** By Zhang, Cox and Doerr [48]

A targeted steganalysis method was proposed to detect the LSM Matching algorithm. The concept is to calculate the absolute differences between the local extrema, because the maxima will be decreased and the minima will be increased after executing the LSB Matching algorithm in the image. Also the neighbouring pixels of these local extrema are affected, an obvious reduction reflected on the image histogram.

In the LSB Matching algorithm, there is a 50% probability that the pixel LSB will be changed. However, for an embedding rate  $p$ , the probability that pixels will not be modified in the cover image is  $(1 - p/2)$ . Hence, the histogram of the stego image can be represented by: ( $h_s$  indicates the stego image histogram;  $h_c$  indicates the cover image histogram)

$$h_s(n) = (1 - \frac{p}{2}) h_c(n) + \frac{p}{4} (h_c(n-1) + h_c(n+1))$$

Also, the local extremum  $n^*$  can be obtained and defined from this formula, as shown in following function:

$$h_s(n^*) = (1 - \frac{p}{2}) h_c(n^*) + \frac{p}{4} (h_c(n^*-1) + h_c(n^*+1))$$

$$= h_c(n^*) - \frac{p}{4} [(h_c(n^*) - h_c(n^*-1)) + (h_c(n^*) - h_c(n^*+1))] < h_c(n^*)$$

However, the absolute differences between the extremum and its neighbours has an appreciable attenuation in the histogram. The maxima decreases and the minima increases. Thus, the stego image's histogram is smoother than that of the original cover image. As the following function evidenced:

$$D_c = \sum_{n^*} |2 \cdot h_c(n^*) - h_c(n^*-1) - h_c(n^*+1)|$$

$$D_s = \sum_{n^*} |2 \cdot h_s(n^*) - h_s(n^*-1) - h_s(n^*+1)|$$

In the conclusion, this detection was compared with other steganalysis, Ker's detection [49] and Goljan's [63] detection. This method has the highest probability of detection in spatial images (never-compressed) in all three of them. However, if the dataset has been JPEG compressed, the testing result is not as good as expected.

***“LSB matching steganalysis based on patterns of pixel differences and random embedding.”***

By Lerch-Hostalot and Megias [50]

A novel steganalysis method based on the identification of patterns of pixel differences (PPD) was proposed in 2012. The main idea is to analyse the differences between neighbouring pixels before and after injecting random data by using the LSB Matching algorithm. The target image can be identified through verifying the presentation of the patterns' distribution.

During the implementation procedure, the patterns of pixel differences  $d$  can be created by gathering all of the pixel value from the cover image. In order to simplify the complexity of pattern differences, a threshold has been exploited to reduce this value. The threshold regulation is shown as follow:

$$d(x, y) = \begin{cases} S - 1, & \text{if } |x - y| > S - 1 \\ |x - y|, & \text{if } |x - y| \leq S - 1 \end{cases}$$

where,  $S$  indicates the number of possible value of the pixel differences,  $x$  and  $y$  represent the target pixel and reference pixel respectively.

In addition, as the experiments proved, pixel pairs from the same direction contain redundant information whether in the horizontal, vertical or diagonal. Thus, half of the pixel information of each pair can be neglected due to its intrinsic symmetry, as the example shown in Figure 24:



Figure 24: The illustration of reduction block [50].

(NB: the original 9 (3 x 3) pixels was simplified to 5 pixels. In the horizontal pair, ( $X_{22}, X_{21}$ ) was neglected; in the vertical pair, ( $X_{22}, X_{32}$ ) was neglected; in the diagonal pair ( $X_{22}, X_{11}$ ) and ( $X_{22}, X_{31}$ ) were neglected.)

Since the reference pixel can be chosen from any value in the reduction block, the rest of the target pixels can be defined to the corresponding position information which is related with the reference pixel; these are,  $l$ ,  $ul$ ,  $ur$  and  $r$ , representing 'left', 'upper left', 'upper right' and 'right' respectively. In these 5 pixels,  $N$  represents the array of the rest of target pixels, there are:

- $N[1]$  = left;
- $N[2]$  = upper left;
- $N[3]$  = upper right;
- $N[4]$  = right.

Hence, combining the above functions, the pattern of pixel difference can be represented as:

(NB:  $P$  indicates the pattern array;  $b$  indicates the reference pixel)

$$P[i] = d(b, N[i]), \quad i=[1,4].$$

Also the maxima and minima  $P$  can be identified according to the different values of the reference pixel, denoted to  $P_{min}$  and  $P_{max}$ .

Finally, an integer  $M$  which represents the pattern frequency value can be gathered by mapping these patterns' values, the function is as shown below:

$$M(P, S) = P[1]S^3 + P[2]S^2 + P[3]S^1 + P[4] + 1,$$

Whether the target image contains a hidden message can be verified through counting the result of the pattern ratio. In this experiment, in order to compare the different ratio result between the cover image and the stego image, the random message was embedded into both of them. In the Histogram

Pattern, both images have the same result. Although the frequency decreases after the random message is embedded in the cover image, the magnitude is not obvious. However, in the Ratio of Pattern counters before and after random data embedding, the results of both images are extremely different. As shown in Figure 25:

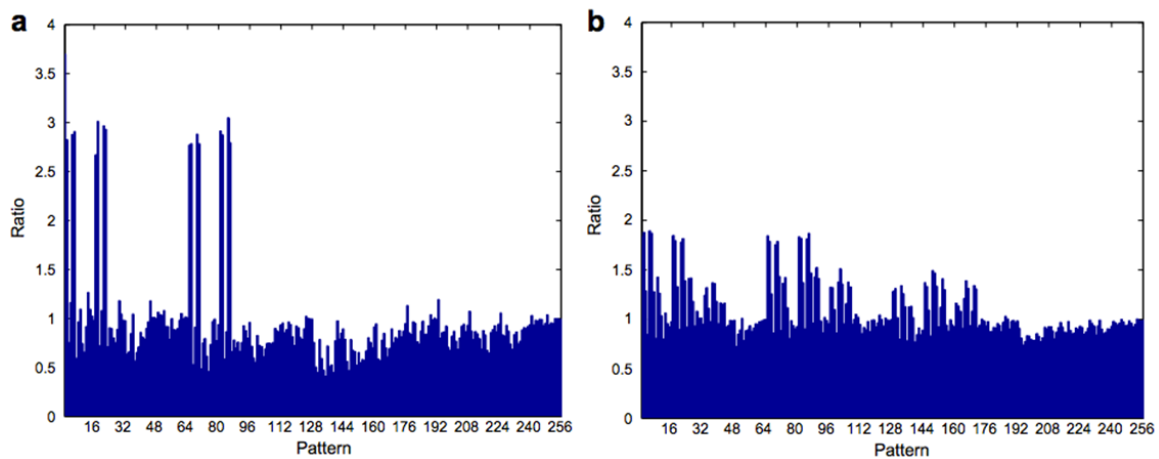


Figure 25: Ratio of pattern counter comparison between a cover image (a) and the stego image (b) [50]

(NB: In this case,  $S$  was equal to 4. In the stego image, the values of the ratio are much more stable than in the cover image.)

In this method, the stego image can be identified clearly through injecting random data into the target image with the LSB matching algorithm. If the Ratio of Pattern has a stable distribution, the target image can be identified as stego image, otherwise, it is a typical image. Although the computation process of this analysis method is complicated, it can be used widely to identify other advanced steganographic algorithms, even in JPEG steganography.

**“Steganalysis of additive noise modelable information hiding.”** By Harmsen and Pearlman [51]

Another steganalysis method based on the identification of additive noise was presented in 2003, and it can be applied to detect multiple image steganography algorithms, such as LSB replacement, LSB matching, other frequency-based algorithms. Several theorems have been separately proved to distinguish the stego image from the cover image, because the stable correlation in the original cover image can be disturbed due to the injection of the secret data.

During the steganography procedure, the generation of the stego image can be seen as the injection of (stego) noise to the cover image. Hence, the histogram of the stego image is equal to the convolution of the cover image histogram and stego noise, as shown in [Formula 1](#):

$$h_s[n] = h_c[n] * f_\Delta[n]$$

(NB: Where  $f_\Delta[n]$  is the probability mass function, which indicates the probability that a pixel will be altered by  $n$  after inject the noise. The **equation 1** is :  $f_\Delta[n] \triangleq p(X_s - X_c = n)$ .  $X$  indicates the pixel value.)

In order to simplify the process of analysis, the above **equation 1** and **Formula 1** are evolved to the frequency domain presentation by using Discrete Fourier Transform (*DFT*), also the result is defined to the Histogram Characteristic Function (*HCF*), which is a representation of the image histogram in the frequency domain, as shown in the following **equation 2**.

$$H_s = F_\Delta[k]H_c[k] \Leftrightarrow DFT(h_s[n]) = DFT(f_\Delta[n]) DFT(h_c[n])$$

The Histogram Characteristic Function Centre of Mass (*HCF COM*) is defined to the measurement of the *HCF* distribution, and the result is useful to identify the existence of the stego noise. As shown in **equation 3** following:

$$C(H[k]) \triangleq \frac{\sum_{k \in \mathcal{K}} k |H[k]|}{\sum_{i \in \mathcal{K}} |H[i]|}$$

(NB: Where,  $\mathcal{K} = \{0, \dots, \frac{N}{2} - 1\}$  and  $N$  is the length of *DFT* )

After injecting the stego noise to the cover image, the result of *HCF COM* may reduce or stay the same. As shown in **Formula 2**:

$$C(H_s[k]) \leq C(H_c[k])$$

Therefore, in the detection procedure, the stego image can be identified from *HCF COM*, the expected distribution is absolutely lower in the stego image. In their conclusion, this detection method has been demonstrated in different imagesets, the detected result has a high correct rate, especially in the identification of the LSB replacement algorithm.

**“Steganalysis of LSB Matching in Grayscale Images.” (Ker Detection).** By Ker [49]

As Harmsen did not implement his steganalysis on the grayscale image, the limitation was presented by Andrews D. Ker soon after. Although the original theorem stated that it should be available, the result of one-dimensional image steganography is far away from the expected result, which is much bigger. Hence, two novel detections based on the improvement of the *HCF* were proposed two years after Harmsen’s publication [51].



Several pieces of evidence proved the unreliability of  $HCF$  in grayscale images, such as unknown  $C(H_c[k])$ <sup>9</sup> and sparse distribution in grayscale image histogram<sup>10</sup>.

In order to address the problem of unknown  $C(H_c[k])$ , a downsampling method was considered to use to gather the pixel, as shown in the follows **Equation 1**:

$$p'_c(i,j) = \left\lfloor \sum_{u=0}^1 \sum_{v=0}^1 \frac{p_c(2i+u,2j+v)}{4} \right\rfloor \quad \text{Equation 1}$$

(NB: Where,  $i$  and  $j$  represent the pixel coordinates in the image)

As the massive experiment proved [49], the distribution between the original  $HCF\ COM$  and downsampling  $HCF\ COM$  is a linear with a correlation bitrate in 0.9967, which means that both results are very similar. So, the **Equation 2** is as shown below:

$$C(H'_c[k]) \approx C(H_c[k]), \quad \text{Equation 2}$$

Also, after inject the stego noise (secret data) by using the LSB Matching algorithm, the **Equation 3** is presented:

$$C(H_c[k]) - C(H_s[k]) > C(H'_c[k]) - C(H'_s[k]) \quad \text{Equation 3}$$

Therefore, according to the above equations 2 & 3, the **Formula 1** can be evolved:

$$C(H[k]) < C(H'[k])$$

The LSB Matching algorithm is subject to the above **Formula 1**, the existence of stego noise can be proved if the target image meets this theorem. Hence, the first detection method changes the pixel by using downsampling, and then observes the relationship between  $HCF\ COM$  and *downsampling*  $HCF\ COM$  in the target image.

In the second detection method, the problem of sparse histogram distribution in the grayscale image has been fixed. The two-dimensional adjacency histogram is used instead of the colour intensity histogram, which indicates how often the pixel values have been observed next to each other in horizontal direction, making the histogram distribution much closer. This is shown in the following **Equation 4**:

$$h_c^2(m, n) = |\{(i, j) | p_c(i, j) = m, p_c(i, j+1) = n\}| \quad \text{Equation 4}$$

Next, the *two-dimension HCF COM* can be formed by the following **Equation 5**:

<sup>9</sup> During the detection procedure, the cover image may be unknown.

<sup>10</sup> In three-dimensional colour image, the distribution is  $N^3$ ; in one dimensional grayscale image, it is only  $N$ .

$$C^2(H^2[k, l]) = \frac{\sum_{i,j=0}^n (i+j)|H^2[i,j]|}{\sum_{i,j=0}^n |H^2[i,j]|} \quad \text{Equation 5}$$

Finally, the stego noise can be identified by observing the *two-dimension HCF COM* relationship between the cover image and stego image. Also, this method can be combined with the above **Formula 1** during the detection process without the related cover image.

### 2.3.3. Summary of Steganalysis

The development of steganalysis has been continually progressing. New detection methods are constantly being substituted and are evolving constantly. The majority of reasons for this are:

- The detection result is subject to some parameters, such as the embedding rate. If the embedding rate is lower, the result will be not as good as expected.
- The limitations of the detection method. For example, some of the methods only work on grayscale images.
- The long computation time for the detection procedure, especially if a higher complexity algorithm is used.

So far, there have been some novel steganalysis methods proposed, but most were updated soon after their development for the above reasons.

In 2008, a novel detection method based on the grayscale level co-occurrence matrix (GLCM) was proposed by Abolghasemi et al[52]. By observing the pixel concentration distribution, the stego image can be distinguished from the cover image as the injection of secret data can affect the distribution along the diagonal direction in the co-occurring pixel matrix. However, the limitations of this method were proved soon after. Firstly, this method was only available for grayscale images. Secondly, the accuracy was not as good as expected. Subsequently, this detection was improved by H.B.Keker in 2011 [53]. Although this new method can be executed for colour images and the result is accurate, the complexity of computation still exists.

Therefore, the two sides (steganography and steganalysis) are in a constant battle for the most effective detection methods. However, it is difficult to know which is best due to the diversity of steganography algorithms, especially for DLP systems. Many other factors need to be considered when developing DLP systems to prevent the potential risk from the steganography, such as the computational time.

## 3. Methodology

### **Research Question:**

This research aims to investigate an effective solution in DLP systems to prevent the threat of image steganography.

### **Objectives:**

In order to address the above research question, in this section, the specific methodology will be described from three aspects.

- Firstly, the challenges / limitations of the DLP system should be analysed, to consider:
  - How can the confidential data, embedded in the cover file via Steganography, bypass the DLP system monitor?
  - Why the current steganalysis mechanisms can not detect the existence of steganography in DLP systems effectively?
- Secondly, the preliminary investigation and research hypothesis will be discussed and analysed.
- Finally, in order to prove the reliability and applicability of the presented solutions, the architecture of the experiments will be introduced.

### 3.1. The relevant challenges

Both algorithm and detection mechanisms are under constant evolution and optimisation. Especially, within a DLP system, some of the reliable steganalysis solutions are not suitable for implementation. The major challenges stem from two aspects, the diversity of the advanced algorithms and the limitation of the corresponding steganalysis.

#### 3.1.1. The Diversity of Steganography Algorithms

The rapid development of steganography algorithms is the first challenge. Either in the spatial domain or in the frequency domain, the sophistication of steganalysis solutions and steganography algorithms are both advancing in a cat and mouse fashion, rendering many previous algorithms analysis and solutions obsolete. Often the novel steganography algorithms are one step ahead of steganalysis, making detection extremely difficult. For example, as mentioned by author Lou [30], the proposed algorithm has a good resistance to both *RS* attack and  $X^2$  attack which are the most effective steganalysis solution so far.

For a specific algorithm, different parameters can often generate the different stego image matrices such as the embedding rate according to traditional LSB algorithms. As the statistics from the above section show [31][38][39][40][41], the probability of detection and embedding rates are inversely proportional, the imperceptibility of the stego image can be increased by reducing the embedding rate. Therefore, the most serious challenge is based on the embedding rate during the application of the corresponding steganographic algorithm. The majority of novel proposed algorithms may bypass the detection of effective steganalysis when their embedding rate has a low value.

There are many open source tools on the Internet <sup>11 12</sup>, for both image and text based steganography. The complete procedure of steganography that involves the embedding and extracting process can be implemented by anyone, which means company data can be hidden easily by any internal staff, including non-technical personnel.

### 3.1.2. The Limitation of Steganalysis Methods

As discussed previously, steganalysis can be divided into visual detection and statistical detection in two separate ways. Often, the statistical detection method is the most effective solution to steganography, and the most common method is the histogram detection.

However, the most important problem is the size of the (stolen or secret) company data (embedding rate) to be hidden. Detection techniques may display a minute difference in the cover image which cannot be distinguished by human eyes, or even in histogram charts, especially if this confidential data is just a few characters. Even for the most effective steganalysis, such as an RS attack, the detection result may not be reliable when the embedding rate is lower than 0.005 [30]. An example of LSB replacement algorithms can be seen in Figure 26 below (this is the histogram result between an original “Lena” image and LSB 1 stego “Lena” image which conceals hidden textual data, “this is for test !!!”).

---

<sup>11</sup> Text Steganography Tool: <http://manytools.org/hacker-tools/steganography-encode-text-into-image/>

<sup>12</sup> Image Steganography Tool: <http://incoherency.co.uk/image-steganography/>

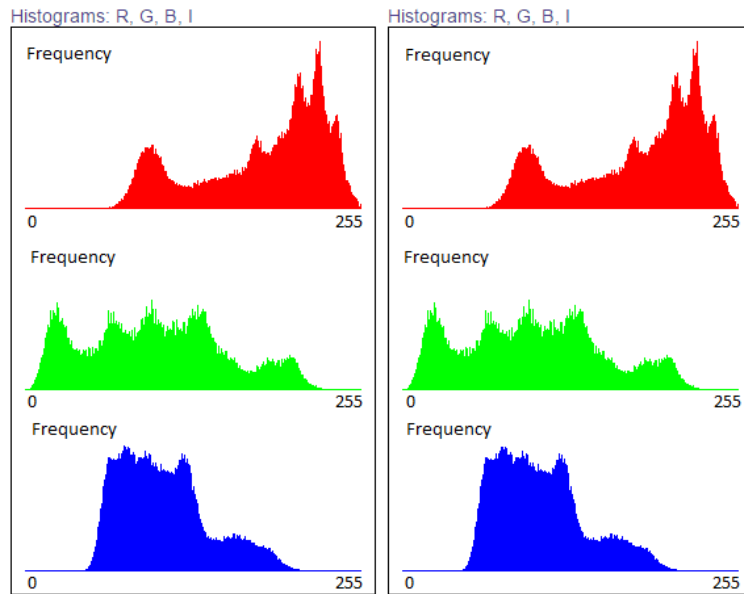


Figure 26: The histogram comparison between the original image and the LSB 1 stego image in Lena.bmp

As Figure 26 above represents, there is an insignificant difference between LSB 0 and LSB 1. Therefore, in industry, DLP systems are faced with serious challenges, particularly in defence steganography. With the increase of steganography utilisation, it is very difficult to determine which solution can verify all algorithms effectively, as unreliable detection is wasting time and money in industry only. Secondly, within any DLP system, it is impossible for security staff to detect each single image in the company boundary traffic as they would have to be able to download the original image, and then compare it to a stego image for possible detection. Thirdly, the most important is the content-aware work principle in the DLP system. An image file is identified as an image data stream in the network transmission rather than the text data stream. This can then lead to the successful exportation of sensitive company information by a malicious actor residing within the organisation, potentially causing serious reputable and financial damage.

### 3.2. Preliminary Investigations

In this section, the preliminary investigations will be introduced, which lead to the subsequent hypothesis and research solutions. These investigations are based on two aspects:

- **Network transmission**, to discover the (next) destination of an email leaving the company boundary.
- **Concealment of data by concatenation**, a simple method for the concealment of data by using file concatenation in a Windows system and its solution will be discussed.

### 3.2.1. Network Transmission

Prior to preventing data loss within an enterprise network, it is necessary to determine the route of the network stream, and how DLP can identify the content of the transmission stream. As mentioned in the above DLP system section, the DLP system is an integrated defence system, designed and implemented by multiple function modules. For example, the vendors may have provided a cloud service to protect the email transmission, meaning that the vendors could receive and identify the email content before the email arrives at the target destination. Although a vendor white paper [25] [26] did not introduce this, it was proved by the following experiments:

*(NB: This experiment was implemented on a Windows system, the email service used Googles Gmail, while the packet capturer was implemented by Wireshark.)*

The tested PC IP address was 138.251.207.129, located in the Jack Cole Building of the St Andrews Computer Science department, as shown in Figure 27 below:

```
Wireless LAN adapter Wireless Network Connection:

Connection-specific DNS Suffix . . : cs.st-andrews.ac.uk
Link-local IPv6 Address . . . . . : fe80::d5c1:be46:dd6a:6064%10
IPv4 Address. . . . . : 138.251.207.129
Subnet Mask . . . . . : 255.255.255.0
```

Figure 27: The details of target PC

The network packets were captured by Wireshark after sending a simple email to a Hotmail hosted email service. Notably, the Wireshark did not capture any mail transfer protocol packets, such as POP or SMTP. Instead, the packets only contain TLSv1.2 and TCP even after filtering packets from the source IP 138.251.207.129, as shown in Figure 28 below:

86	42.502650000	138.251.207.129	216.58.198.197	TLSv1.2	566 Application Data
87	42.503033000	138.251.207.129	216.58.198.197	TLSv1.2	100 Application Data
88	42.503091000	138.251.207.129	216.58.198.197	TLSv1.2	1183 Application Data
93	42.772124000	138.251.207.129	216.58.198.197	TCP	54 62487-443 [ACK] Seq=1688 Ack=106 win=65084 Len=0
96	43.091421000	138.251.207.129	216.58.198.197	TCP	54 62487-443 [ACK] Seq=1688 Ack=608 win=64958 Len=0
99	43.091701000	138.251.207.129	216.58.198.197	TCP	54 62487-443 [ACK] Seq=1688 Ack=1546 win=65205 Len=0
100	43.092587000	138.251.207.129	216.58.198.197	TLSv1.2	100 Application Data
104	43.383414000	138.251.207.129	216.58.198.197	TCP	54 62487-443 [ACK] Seq=1734 Ack=1976 win=65097 Len=0

Figure 28: The detail of captured email packets

Several packets were sent to the destination address 216.58.198.197. After tracking this IP address, it can be determined to belong to Google, Inc, as shown in Figure 29 below:

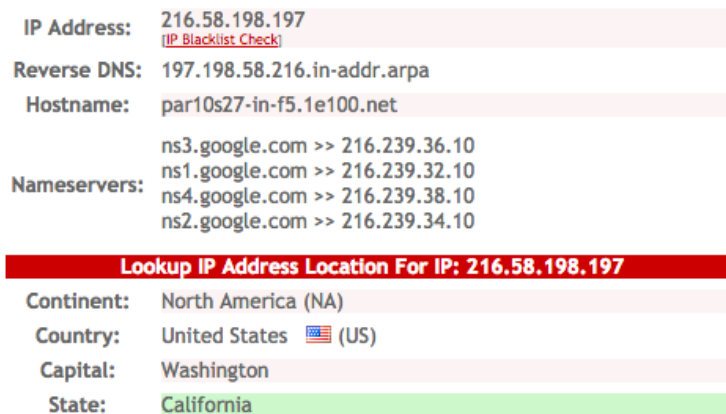


Figure 29: The explanation of IP: 216.58.198.197

This is due to St-Andrews utilising Google’s Gmail as their email hosting platform up until 2015. Hence, this email was sent to the Google server first after it left the St Andrews network boundary. In addition, the email was sent via web browser, the inside of these packets were encrypted by the TLS protocol, which has been used for providing confidentiality and data integrity between two applications [54], the data is therefore unreadable, as shown in Figure 30 below:

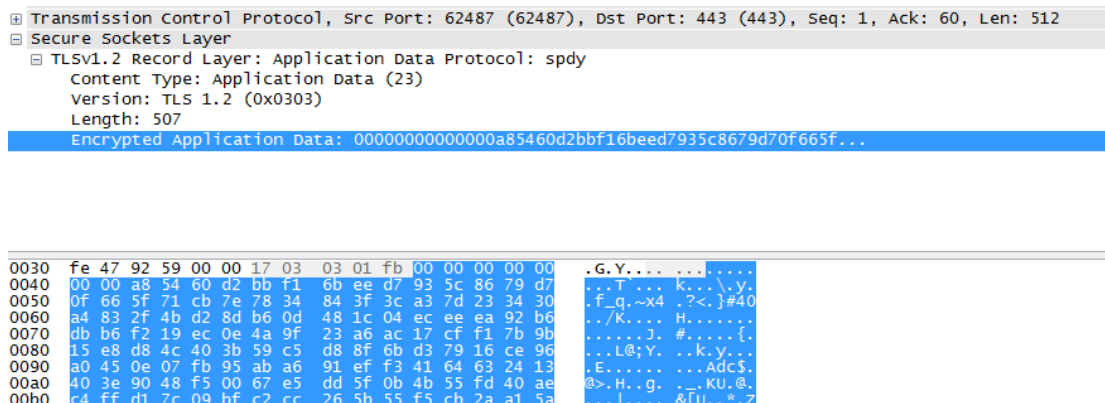


Figure 30: The specific detail of Secure Sockets Layer in the captured packet

It should be noted that in Oct 2014, a vulnerability was found in SSL 3.0, so relevant products use the TLS protocol in Google [55]. For the vendor email service a remote cloud system will receive all of the email stream once it leaves the company network boundary. During the stream transmission, the data will be split into different packets depending on the capacity of a packet. Subsequently, this data will be encrypted in accordance with the transmission algorithm, and transform it to a human unreadable format. The data can be readable within remote vendor cloud systems; which means that any confidential data can be monitored and identified before forwarding to the destination address, and any malicious data leakage can be terminated in the cloud system.

### 3.2.2. Concealment of Data by Concatenation

Prior to researching the solutions for the use of steganography algorithms within a DLP system, another efficient hiding method should be considered and solved. This method combines multiple files to implement the hiding of a target file. This experiment was implemented in a Windows O/S; the files were linked by system commands in the Windows “cmd”.

The hiding procedure is divided into three phases:

1. Prepare an image as the cover file,
2. Compress the data to a .zip file.
3. Type in the command “copy /b File1 + File2 File3” in CMD.
  - File 1 represents the cover image,
  - File 2 represents the secret data file,
  - File 3 represents the new generated file, and this file can be defined in any types.

An example of this steganographic technique can be shown in Figure 31 below,

```
C:\Users\yw43\Desktop\Link Files Example>copy /b StA.png + data.7z StegoWithZip.png  
StA.png
```

Figure 31: The command of hiding data by the concatenation method in a Windows system

(NB: The data file has to be linked behind the image file, otherwise, the newly generated image is unreadable. For more details, refer to Appendix A).

During the extraction procedure, the data can be removed through an unzip on ‘File 3’ directly. For example, in a Windows system, run the ‘7 zip File Manager’, click the StegoWithZip.png file and the data will be uncompressed and display the final result, as shown in Figure 32 below.

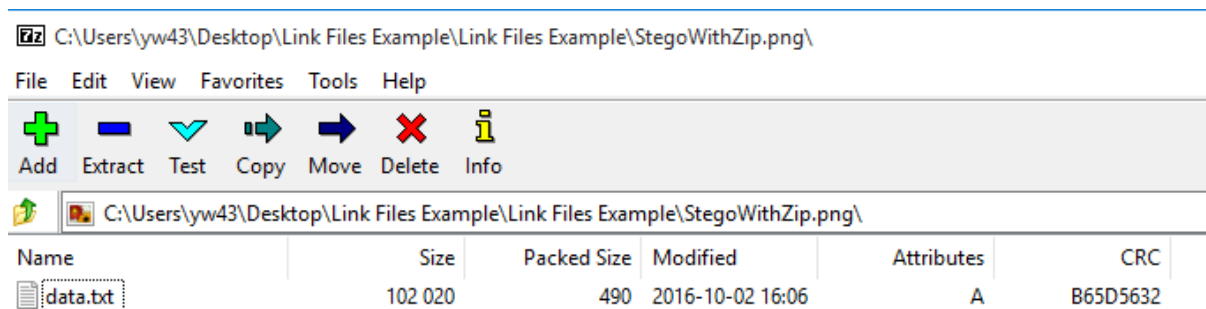


Figure 32: The extracted result of a target file by using the 7zip File Manager

Therefore, in this method, the merging steganographic technique can be implemented easily and the result is imperceptible to visual detection, as shown in Figure 33 below.



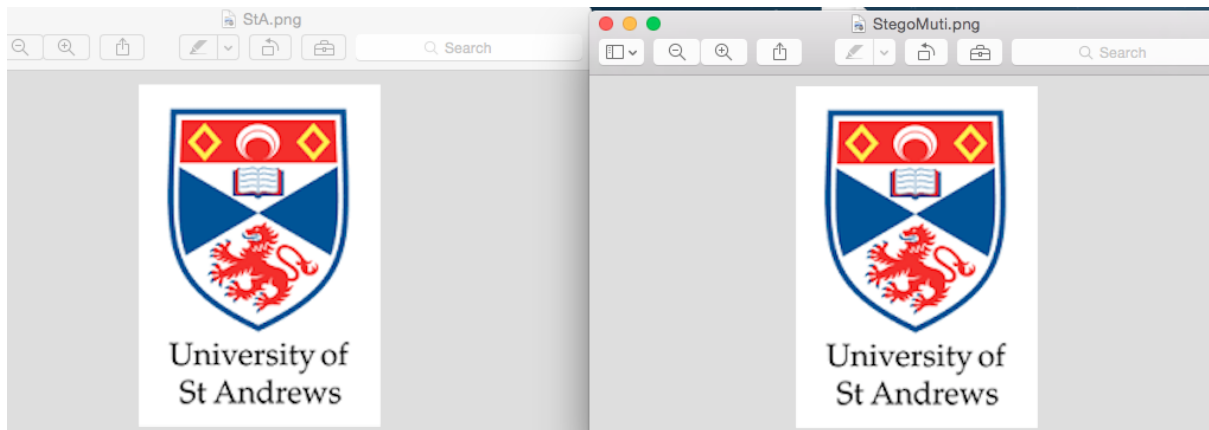


Figure 33: The visual comparison between the original image and the merged image

The first consideration is that of memory allocation. Unlike in a steganography algorithm, the size of the newly generated 'File 3' has been increased with the addition of the large data file ('File 2'), the size of 'File 3' is the combination of the size of both 'File 1' and 'File 2', as shown in Figure 34 below. Instead, in steganography, the size of generated stego image can be the same as the original cover image.

data.txt	2 October 2016 17:06	102 KB	text
StA.png	26 April 2016 00:58	7 KB	PNG image
stego1.png	Yesterday 18:14	109 KB	PNG image

Figure 34: The details of experimental files

An obvious result was produced in the example from Figure 34 above, the data file was a TXT file rather than .zip file, because the zip compression will reduce the size of file so significantly that the compared result is therefore negligible. Normally, in order to decrease the file size, the data file is compressed to a .zip file, which reduces the overall file size as much as possible. This is shown in Figure 35 below:

(NB: where 'File 1' is the above StA.png, and the data.txt ('File 2') is compressed to data.7z.)

data.7z	Yesterday 19:13	605 bytes	Keka...cument
StA.png	26 April 2016 00:58	7 KB	PNG image
StegoWithZip.png	Yesterday 19:14	7 KB	PNG image

Figure 35: The details of experimental files; the secret file is compressed to a zip file

Secondly, the most important vulnerability is to break the previous format signature. The file signature is also referred to as a magic number, which is used to identify or verify the content of a file. Every file has the different hex signature according to their different file extension [56]. For the most popular image files, such as BMP, PNG and JPEG (JPG), the hex signature are as follows:

- BMP: starts with "42 4D", the end hex digits are not defined.

- PNG: starts with “89 50”, the end is “60 82”.
- JPEG (JPG): starts with “FF D8”, the end is “FF D9”.

However, with the merging of the data files behind these images, the end part of file signature will be extended according to the linked data file extension, until all of the content of data file has been represented, as shown in Figure 36 below.

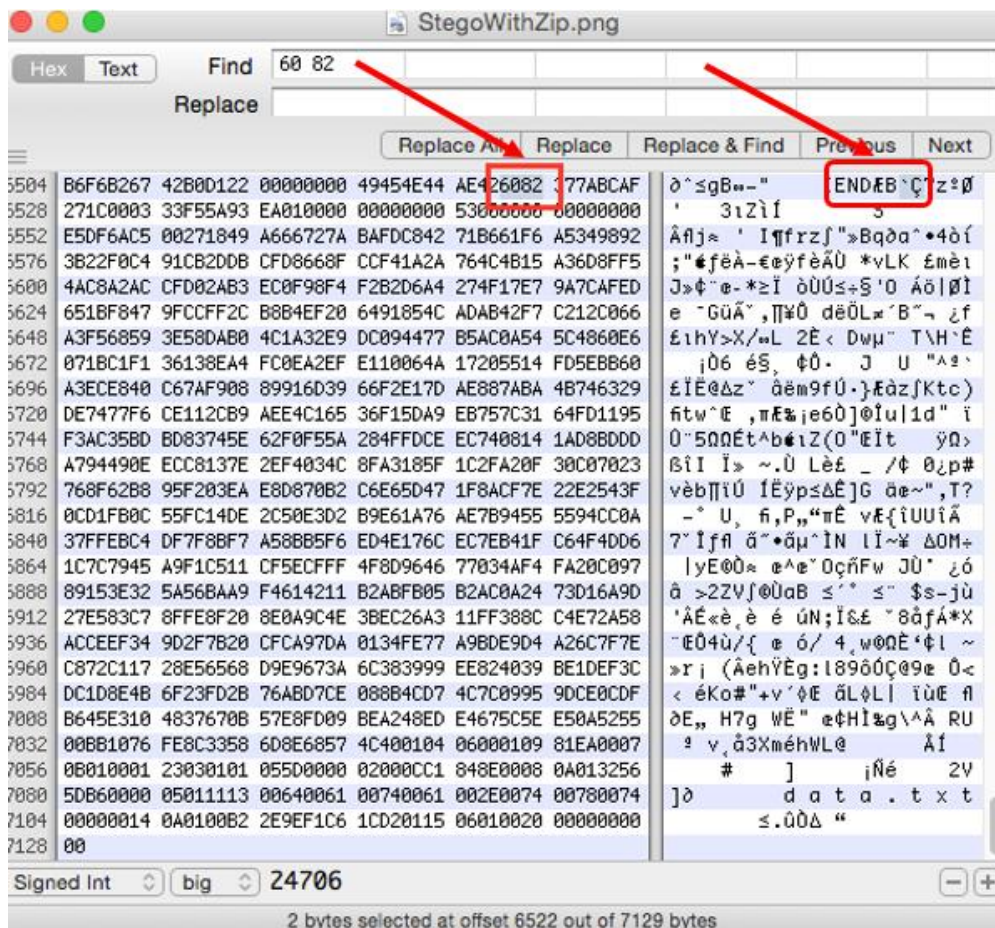


Figure 36: The target hex result in Hex Fiend

From this example, the result file is a PNG image generated by the combination of a PNG image with a .zip data file. The original PNG end signature is “60 82”, yet more hex data demonstrates the linked file contains the newly generated end signature value of “00 00”.

Therefore, a second vulnerability or limitation is the identification of the hex signature. Some effective tools are recommended to implement the corresponding identification. One of which includes checking the hex signature, which can be done by using some open source tools, such as WinHex, Hex Fiend, and how the content can reflect the file’s properties. In the example above, Figure 36, the screenshot results from the software “Hex Fiend” on a Mac system and the malicious data can be identified easily due to its file extension. However, for JPEG and PNG images, they both contain a definite signature either at the start or at the end. When it comes to a BMP image, the end part of the

signature is not defined, so that the identification is often unreliable in BMP image by using these hex editor tools.

Another effective tool that has been used to distinguish the malicious sub-file within a file is “Binwalk”. According to the different starting signature of each file, the specific sub-file can be identified as well as the corresponding description. As shown in Figure 37 below, the experiment result was from a BMP image that was linked a TXT file.

```
pc-131-68:Link Files Example $ binwalk stego1.bmp
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0         PNG image, 136 x 193, 8-bit colormap, non-interlaced
464         0x1D0      Zlib compressed data, default compression
```

Figure 37: The binwalk search result when the secret file was a TXT file

Also, as shown in Figure 38 below, the experiment result was from a BMP image which was linked to a .zip file.

```
pc-131-68:bmp $ binwalk StegoWithZip.bmp
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0         PC bitmap, Windows 3.x format,, 136 x 193 x 24
78798       0x133CE     7-zip archive data, version 0.3
```

Figure 38: The binwalk search result when the secret file was a .zip file

Therefore, for the file concealment method as a steganographic technique, although the implementation procedure is simple, it can be detected easily through identifying the corresponding signature within the file. In addition, the excessive file size can be seen as a malicious file directly, which may be caused by the merging of a large data file.

### 3.3. The Research Hypothesis

The concept of helping industry by preventing data from being exported with the use of the LSB algorithm predominantly stems from two aspects:

Firstly, an unsuccessful steganography process can be divided into three features [28]:

- “1. Prove the existence of the confidential message.
2. Extract the confidential message
3. Destroy the confidential message.”

Therefore, for the enterprise, there is nothing lost as long as the secret data is not readable. As their aim is to protect the confidentiality of data, it is unnecessary to prove the existence of confidential messages or even extract them, even destroying the secret data alone can meet their requirements. Also, the solutions to protect against corporate loss in a DLP system should be implemented efficiently and effectively, as many images sent are normal images in network traffic.

Secondly, as the summary of the LSB algorithm in the previously discussed Literature Review section showed, for the spatial domain image, the algorithms conceal the confidential data by modifying the value of image pixels, regardless of whether the algorithm is based on edge detection or value differences. The extraction phase is also based on the intensity of each image pixel and corresponding stego key. Consequently, on the condition that there is no visible distortion of the image, if some of the stego image pixel values are randomized, the confidential data can be destroyed and become unreadable even if extracted by a stego key.

Therefore, the Research *Hypothesis* to be presented in this thesis is:

- *By modifying the pixel values of spatial domain images slightly, secret messages hidden inside cover images will be destroyed, while the image quality would be preserved.*

### 3.4. Experimental Architecture

In order to verify the correctness of the above hypothesis, some experiments were designed and implemented to investigate the influences between original pixels and randomized pixels. Two effective solutions are presented; both are subject to the modification of image pixel value. One is based on the modification of image chrominance, which is the **Chrominance Modification Algorithm (CMA)**. The value of RGB in each pixel is affected by changing the value of chrominance. This is in YUV, where, the value of U and V will be changed, the value of Y is kept constant, rather than changing the colour intensity directly as luminance is more sensitive for human eyes [28].

The other solution is based on the conversion of the image frequency domain, which is called **Presentation Domain Transform (PDT)**. The compression of JPEG images is a lossy procedure, the quantization phase is an irreversible process, even if reverse engineered, the initial value could never be revealed [28].

These experiments were divided into two stages. The reliability and applicability of both solutions need to be verified first, then the results need to be analysed and evaluated. The specific experimental procedure is described as below:

### ***Required Tools***

In order to enhance the transparency of the experiments as well as deeply understand the experimental procedure, two tools were developed in Java with JRE System Library “JaveSE-1.7”. One tool is based on data hiding and the other tool is based on data extraction.

### ***Algorithm***

The LSB algorithm was executed in the embedding process. Similar to the majority of online open-source software, the algorithm (LSB 1) can be implemented through replacing data of the last bit of each pixel value with each message bit. In addition, in the developed tool, the replacement bit(s) in each pixel is controllable, from LSB 1 to LSB 8, to implement multiple bits embedded to increase the data hiding capacity. This implementation also can be used as a reference tool to detect the quality threshold that represents the image distortion for the traditional LSB algorithm.

### ***Measure Threshold***

With the constant increase of data size, the maximum bits available within LSB 1 is sometimes not enough to hide all of the data. Generally, in order to digest a large volume of data, increasing the number of bits replaced is an easier way to implement data hiding, although the image distortion becomes appreciable with the increase in replaced bits.

Therefore, the threshold of the traditional LSB algorithm should be measured for understanding its limitation further. As described by Fyffe [64], the visual threshold was identified at 3 bits LSB (LSB 3) through experimentation with 25 men and 25 women ranging from 20 - 60 years old. The subject viewed multiple visual examples of images with and without embedded hidden data.

Notably, as mentioned above, luminance is more sensitive for human eyes. Hence, the threshold result may not be very exact if an answer is given using visual detection only. The visual comparison between the original cover image and the stego image in “StAndrews Bridge.png” and “Lena.png”, with both images containing 100 KB of embedded data from a TXT file using the LSB 3 algorithm is shown in Figures 39 and 40. The data was embedded from the upper left corner of the image, and down the vertical direction. Obviously, in Figure 39, the stego image has appreciable blur, from the left side, especially in the sky part of the image. However, under same conditions with the same algorithm, the visual difference is negligible in Lena.png.



Figure 39: The visual comparison between the original cover image and LSB 3 stego image, in StABridge.png



Figure 40: The visual comparison between the original cover image and LSB 3 stego image, in Lena.png

Therefore, due to the influence of pixel intensity, the identification of a threshold is unreliable if the detection statistics is only from human visibility. In the next section, in order to enhance the credibility of the results, the threshold will be identified by using corresponding measurement units.

### **Measurement Units and Tools**

Two measurement units will be used to verify the quality of image in the next sections; there are the Peak Signal-to-Noise Ratio (PSNR) and the Mean Squared Error (MSE). As mentioned before, the quality of images and MSE are inversely proportional, the higher value of MSE indicates a dissimilarity

between the compared images. However, the value of PSNR indicates the quality of an image, with a higher value indicating a better quality.

Firstly, these measurement units will be used to identify the threshold of the LSB algorithm, through calculating the value between the original cover image and  $n$  bit ( $n \in [1, 8]$ ) LSB stego image. The significant point of detection can be seen as the threshold.

Next, these units can also be used to evaluate the performance of two presented solutions, to verify which one is better.

In addition, an online histogram tool [59] is selected to measure the distribution of the image colour intensity.

### ***Experimental Structure***

Prior to implementing the experiment, the relevant preparatory works were performed:

- Use different  $n$  to execute  $n$  bit(s) LSB algorithm in several cover images and use three different size TXT files as the secret (embedded) file. The cover images were selected from different frequency domain and spatial domain images.
- Collect all of the generated stego images and determine the threshold of LSB algorithms through statistical measurement. A histogram result is recorded for analysing the distribution change for the different sizes of embedding data.

In order to verify the applicability of presented solutions, the entire experimental procedure is divided into three phases.

- Phase 1: Pick all of imperceptible stego images in which the replaced bit  $n$  is under the range of the algorithm threshold, and implement presented solutions to these stego images respectively.
- Phase 2: Verify the applicability and reliability of these solutions through comparing data extraction results between the original stego image and a processed stego image.
- Phase 3: Identify the efficiency of these solution through benchmarking (measurement units).

In this experimental procedure, in order to confirm the distortion level of stego images using the LSB algorithm, the threshold should be researched first as the main risk often comes from the target image which is under this threshold. Otherwise, the distorted image will be identified malicious, and be blocked by defence system. Then, these images will be used to

examine the applicability of the solutions by observing the data extracted result after implementing the relevant solutions. If the result is unreadable and there is not a noticeable distortion in the image, it would prove the usability of these solutions and research hypothesis. Finally, accordingly through benchmarking the different solutions, the most effective solution will be identified.

### ***Analysis***

The evaluation of both solutions will be analysed through comparison with different parameters, such as histograms, PSNR and MSE. Finally, all of the results will be combined to verify the performance of each solution, and determine which one is the most effective solution for DLP systems to prevent the application of steganography.



## 4. Implementation

In this section, the implementation of the project will be explained from three separate aspects:

- Firstly, the developmental procedure of the utility tools will be introduced. The implementation of the core algorithm, *LSB replacement*, will also be explained in this part.
- Secondly, the specific experimental processes will be described. The reliability of the hypothesis and the applicability of the solutions will be verified.
- Thirdly, in order to determine and analyse the effect of the solutions on the image, the relevant measurement units will be computed.

### 4.1. Developed Tools

This project, developed in the Java programming language, consists of two different tools. The fulfilled text based steganography and steganalysis – embedding / extracting a secret text into a cover image, see Figure 41 below.

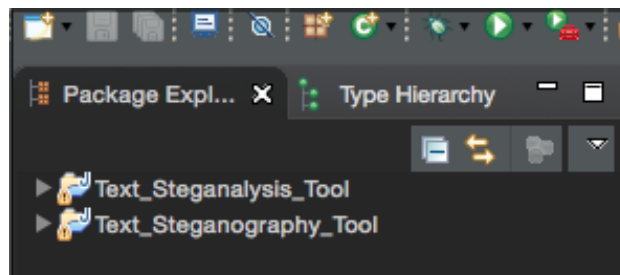


Figure 41: The Tools of this project

#### 4.1.1. Text Steganography Tool

##### **Class Description**

In this tool, the program consists of four packages: *FileProcess*, *MainPage*, *SteganProcess* and *UIDesign*.

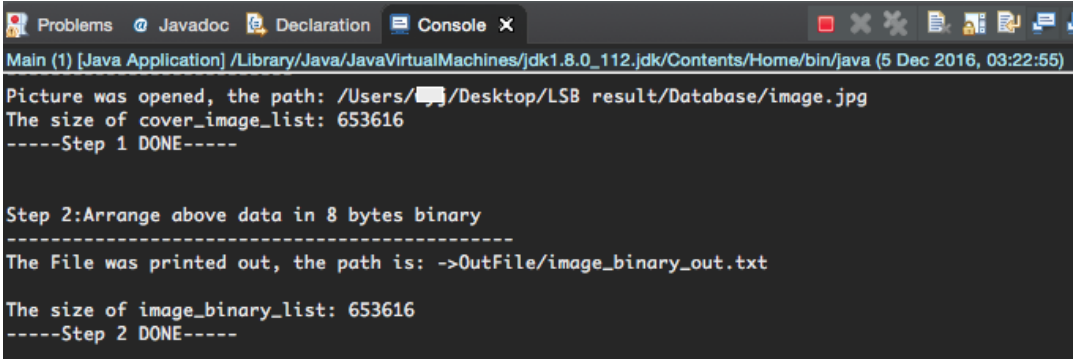
The *FileProcess* package contains two java classes: **ReadFile** and **WriteFile**, the responsibility of which are file processing.

- The *ReadFile* class is implemented for reading in the cover image and the confidential (hidden) data.
- The *WriteFile* class focuses on saving the stego key in a user-defined directory.

The *MainPage* package is the main function of this tool lies, and presented in the **Main** class.

The *SteganProcess* package, which is responsible for the procedure of data embedding, consists of four java classes: ***ArrangeData***, ***Exchange***, ***GeneratImage*** and ***ProcessFlow***.

- In the *ArrangeData* class, the decimal numbers are converted to binary. In order to collect the data precisely during extraction manipulation, it should be noticed that the length of the converted binary must be equal to 8 bits in the embedding procedure. Otherwise, the program will fill 0s automatically in the front of each binary number until this length requirement is met.
- In the *Exchange* class, the *n* bit(s) LSB in the cover image will be replaced by the corresponding *n* bits in the secret message. The value of *n* is pre-set by the users. The binary representation of the stego image will be calculated based on the user input variables, and stored in an *ArrayList*. In addition, this class also contains two types of replacement methods, one is in the horizontal or vertical direction, while the other is in random positions.
- In the *GeneratImage* class, the stego image will be generated from the newly generated *ArrayList*.
- The *ProcessFlow* class controls the work flow of the complete data embedding process. The status of each step will be printed on the system console (as shown in Figure 42), to make it convenient for the users to keep track of the system progress.



```
Problems @ Javadoc Declaration Console X
Main (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_112.jdk/Contents/Home/bin/java (5 Dec 2016, 03:22:55)
Picture was opened, the path: /Users/.../Desktop/LSB result/Database/image.jpg
The size of cover_image_list: 653616
----Step 1 DONE----

Step 2:Arrange above data in 8 bytes binary
-----
The File was printed out, the path is: ->OutFile/image_binary_out.txt
The size of image_binary_list: 653616
----Step 2 DONE----
```

Figure 42: The details of process track in system console

In the package of *UIDesign*, two classes - ***ButtonFrame*** and ***MyFrame*** are developed to build the GUI for this tool.

- In the *ButtonFrame* class, the specific buttons are placed in the bottom of the GUI, and aligned using the *GridLayout*.
- In the *MyFrame* class, the integral layout of this application is designed by using the method of *BorderLayout*. The screenshot of the final application GUI is shown in Figure 43:

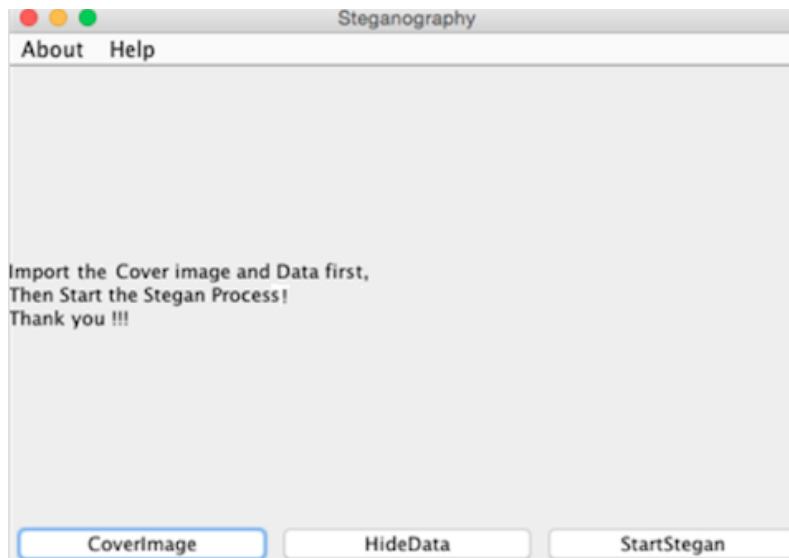


Figure 43: The GUI layout in Steganography Tool

Combining the above descriptions, this tool contains 9 java classes in total, as shown in Figure 44 below:

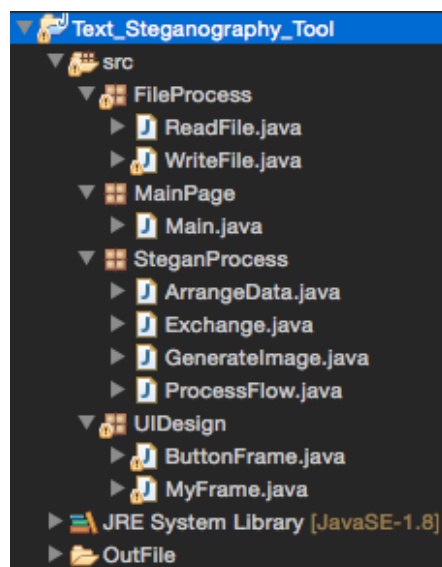


Figure 44: Details of the developed classes and packages in the steganography tool

### **Procedure Description**

The work flow of the data embedding process can be divided into eight steps, there are:

- **Step 1: Import cover image.**

The user can select one cover image from the local directory. Three cover image formats are supported in this program - JPEG, BMP and PNG. The (R, G, B) value (0 to 255) of each pixel of the imported image are stored in an ArrayList named "*cover\_image\_list*".

- **Step 2: Convert decimal presentation of the image into binary form.**

Each item in the `cover_image_list` will then be converted from decimal into binary. After the conversion, each item in the list will be padded with extra 0s in the left side until its length reaches 8. The converted result will be saved into an ArrayList name "`image_binary_list`". As shown in Figure 45:

```
for(int i=0; i<list.size(); i++){
    String data = Integer.toBinaryString(Integer.valueOf(list.get(i)));
    StringBuffer sb = new StringBuffer(data);
    for(int j=0; j<8-data.length(); j++){
        sb.insert(0, "0");
    }
    arrange_list.add(sb.toString());
}
```

Figure 45: The code explanation, convert decimal to string, fill the extra 0s until the length meets 8

- **Step 3: Import confidential data.**

The secret data will be imported in this step (only TXT format is allowed in this tool). In order to extract the data precisely in the steganalysis process, a "line break" symbol must be added to each end of the rows of the secret data (shown in Figure 46). All information can be saved in decimal form and stored into an ArrayList called "`data_list`".

```
while((data=br.readLine())!=null){
    fileData.add(data);
    fileData.add("\r\n");
}
```

Figure 46: The code explanation, write an extra line break during reading the secret message.

- **Step 4: Convert the formatted secret data into binary form.**

The data will firstly be split into single characters, and then converted from string to binary form. The result of this conversion will be stored in an ArrayList named "`data_binary_list`". Again, extra 0s are inserted in front of each item (pixel) until the length of the binary value meets 8, in Figure 47 below:

```

for(int i=0; i<list.size(); i++){
    String data = list.get(i);
    char[] strChar = data.toCharArray();

    for(int j=0; j<strChar.length; j++){
        String binary = Integer.toBinaryString(strChar[j]);
        StringBuffer sb = new StringBuffer(binary);

        for(int k=0; k<8-binary.length(); k++){
            sb.insert(0, "0");
        }
        arrangelist.add(sb.toString());
    }
}

```

Figure 47: The code explanation, convert the secret message to 8-bit binary format.

- **Step 5: Split the data binary list into a single character**

In this step, in order to execute the data replacement operation more conveniently during the embedding process, the generated ArrayList of binary strings will be split into single digits and stored in the ArrayList called "*single\_data\_binary\_list*".

- **Step 6: Execute LSB algorithm**

In this step, the replacing procedure is implemented. According to the requirements of the user, the number of replacing bits  $n$  can be selected from the range of [1,8]. The format of the generated stego image can be set to PNG, BMP or JPEG, although the JPEG format is not a good option for concealing data by using LSB algorithm.

During the replacing process, the quantity of pixels to be replaced will be calculated, as shown in Figure 48:

```

index = insertList.size() % key;

if(index == 0){
    rows = insertList.size() / key;
}else{
    rows = insertList.size() / key + 1;
}

```

Figure 48: The code explanation, the equation of quantity of expected replacing pixels

where, the variables "*key*" and "*insertList*" represent the number of replace bit " $n$ " and "*single\_data\_binary\_list*" separately. The variable "*rows*" is calculated to determine the quantity of pixels to be replaced from the cover image (the quantity of RGB in the cover image).

According to the value of "*rows*", the corresponding data will be extracted from "*image\_binary\_list*". And then the execute  $n$  (key) bit replacement operation is applied with the data list "*single\_data\_binary\_list*" to recombine a new pixel, as shown in Figure 49 below:

```

//put those char into charList
if(i==rows-1 && index!=0){
    for(int j=0; j<charData.length-index; j++){
        charList.add(charData[j]);
    }
}
}else{
    for(int j=0; j<charData.length-key; j++){
        charList.add(charData[j]);
    }
}

//exchange the new bits
if(i==rows-1 && index!=0){
    for(int j=0; j<index;j++){
        charList.add(insertList.get(j));
    }
}
}else{
    for(int j=0; j<key; j++){
        charList.add(insertList.get(j));
    }
}

```

Figure 49: The code explanation, the procedure of replace the bits and generate a new pixel

Finally, these stored data elements will be re-combined from the character form to the 8-bit binary form, and returned to “ProcessFlow”.

In addition, in this class, two functions can be chosen; there are “changeXY” and “changeRandom”. If the data is to be embedded in either the horizontal or vertical direction, the function of “changeXY” will be called. Instead, if the data is to be embedded randomly, the second function “changeRandom” will be executed.

The random procedure is almost the same as the function of “changeXY”. According to the value of variable “rows”, the relative quantity of cover image pixels will be extracted by using the random method before implementing the replacement with the corresponding data. Also in order to record the position of these random pixels, an ArrayList named “random\_index” is used to store them and print to a “random\_index.txt” file, as shown in Figure 50:

```

int num;
Random ran = new Random();
boolean[] bool = new boolean[img_list.size()];
for(int i = 0; i<rows; i++){
    do{
        num = ran.nextInt(img_list.size());
    }while(bool[num]);
    bool[num] = true;
    random_index.add(num);
}
Collections.sort(random_index);
String ran_key_path = "OutFile/random_index.txt";
print.WriteData(random_index, ran_key_path);

```

Figure 50: The code explanation, the method of obtaining the random position in the cover image.

#### - Step 7: Generate stego image

In this step, the colour of the stego image will be combined and stored into an ArrayList named “myColor” according to the new pixel information from the above step, as shown in Figure 51:

```

int colorSize = decimalList.size()/3;

for(int i=0; i<colorSize; i++){
    int index = i*3;
    red = decimalList.get(index);
    green = decimalList.get(index+1);
    blue = decimalList.get(index+2);
    color = new Color(red, green, blue);
    myColor.add(color);
}

```

Figure 51: The code explanation, the procedure of re-combine a new pixel

Subsequently, the stego image can be drawn from the “myColor”, also the image format can be chosen from the one of BMP, JPEG and PNG, as shown in Figure 52:

```

//set color for image
int indexColor = 0;
for(int y=0; y<height; y++){
    for(int x =0; x<width; x++){
        int rgb = myColor.get(indexColor).getRGB();
        img.setRGB(x, y, rgb);
        indexColor++;
    }
}

String[] ss = path.split("\\.");
String types = ss[1];
File out = new File(path);
try {

    ImageIO.write(img, types, out);
}

```

Figure 52: The code explanation, the procedure of repaint a stego image from the new pixel matrix.

Moreover, the stego image can be displayed in a separate window, as shown in Figure 53:

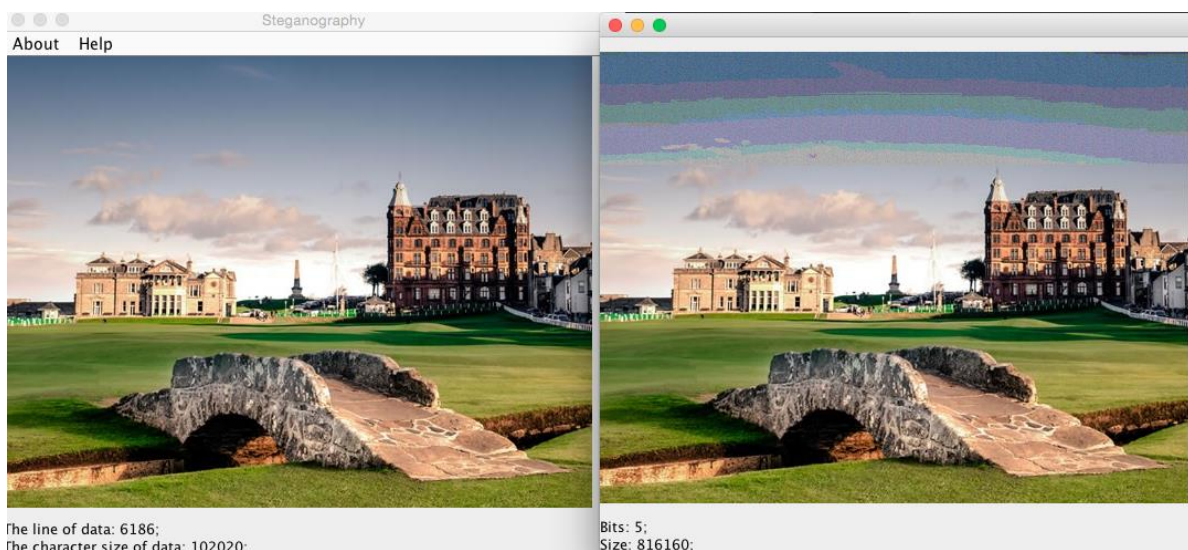


Figure 53: The visual comparison between the cover image and the LSB 5 stego image, in StAbaridge.png

- **Step 8: Release the stego key**

Lastly, the stego key will be printed to a user-defined path. It contains two parameters; value of  $n$  and the character *length* of the embedded data, where,  $n$  denotes the number of bits of replaced data in each pixel.

#### 4.1.2. Text Steganalysis Tool

This program consists of four packages, “*AnalysisProcess*”, “*FileProcess*”, “*MainPage*” and “*UIDesign*”. Besides the package “*AnalysisProcess*”, the functionalities of other packages are the same as the above *Text\_Steganography\_Tool*. Also the GUI is the same, as shown in Figure 54:

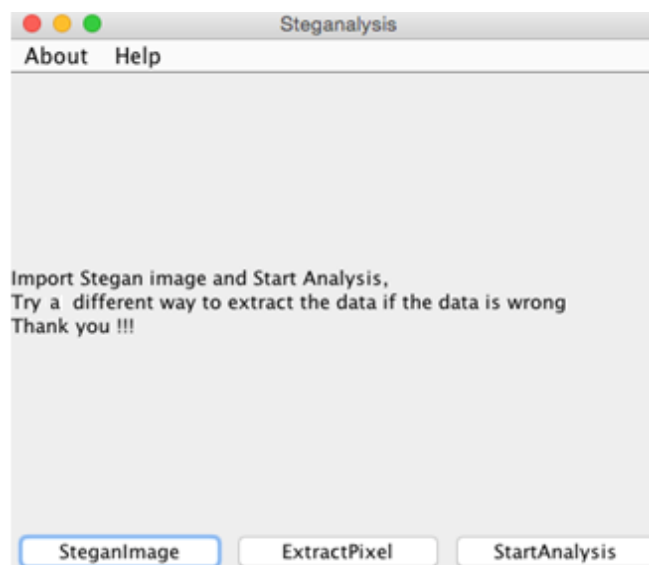


Figure 54: The GUI layout in Steganalysis Tool

The package of “*AnalysisProcess*” is the core operation for extracting the secret data from the stego image and it contains four different class files, ***CollectData***, ***CombineStr***, ***ProcessFlow*** and ***TransformData***.

- In the *CollectData* class, the extraction of embedded data is executed according to the gathered pixel from the stego image and the detail of the *key*. The final data is stored in 8-bit binary form.
- In the *CombineStr* class, the data will be re-combined from character to 8-bit binary form.
- In the *ProcessFlow* class, the complete extracting data procedure is implemented. As with the *ProcessFlow* in *Text\_Steganography\_Tool*, the status of each step will be printed out on the system console (as shown in Figure 55), to make it convenient for the users to keep track of the system progress:



```

Main (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/jav
Loading Stegan Image
Extracting Pixel
Extract ways: row by row
Step 1: load new image in Binary
-----
Save image data into image_list !
----- Step 1 DONE -----

```

Figure 55: The details of process track in system console

- In the *TransformData* class, the extracted data will be transformed from the 8-bit binary form to the human readable plaintext.

During the extraction process, the work flow can be divided into five steps and the specific procedure is described as follows:

- **Step1: import stego image and extract pixel data**

The stego image will be imported, the file format can be selected in a range of [PNG, BMP, JPEG], and all of the RGB information is gathered into an ArrayList named "*image\_list*" in 8-bit binary form. According to the previous stego image generation method, two functions will be considered to describe the horizontal direction or vertical direction of data manipulation.

- **Step2: analyse the key and extract corresponding information**

The information of stego key will be read, the replaced bit *n* and the character *length* of embedding data will be extracted and stored, which are referred to the variables "*KeyOfIndex*" and "*KeyOfSize*" separately in this case.

- **Step3: extract the replaced pixels according to their quantity.**

In this step, according to the details of the stego key, the quantity of replaced pixels can be determined, and these pixels will be stored in an ArrayList named "*changed\_data\_list*" in the 8-bit binary form, as shown in Figure 56,

```

if(step3.equals("X or Y")){
    if(index == 0){
        rows = KeyOfSize / KeyOfIndex;
    }else{
        rows = KeyOfSize/ KeyOfIndex + 1;
    }

    for(int i=0; i<rows;i++){
        changed_data_list.add(image_list.get(i));
    }
}

```

Figure 56: The code explanation, the equation of quantity of replaced pixels

However, for the random embedding, the extraction process will more complicated. The system needs the sequence information of randomly embedded pixels. Hence, one more randomness pixel information file is required, which is generated while embedding data by using the randomness method. The detail of these pixels will be gathered to an ArrayList named “*random\_key\_list*”. Subsequently, the replaced pixels will be extracted and stored to “*changed\_data\_list*” in an 8-bit binary form according to these random sequences, as shown in Figure 57:

```
random_key_list = read.readData(random_key_path);  
  
for(int i =0; i<random_key_list.size(); i++){  
    changed_data_list.add(image_list.get(Integer.valueOf(random_key_list.get(i))));  
}
```

Figure 57: The code explanation, the method of identify the randomly hidden position

- **Step 4: extract the specific embedded data**

The specific embedded data can be extracted in character form according to the detail of replaced pixels “*changed\_data\_list*” and value of replaced bits  $n$ . All of these characters will be re-combined into a new ArrayList in 8-bit binary form.

- **Step 5: transform the embedded data to plaintext.**

According to the information from step 4, the final embedded data can be transformed to the human readable plaintext, and printed out to “*extract\_out.txt*” file, which is located in the “OutFile” file in this program.

## 4.2. Experimental Procedure

As mentioned from the Methodology section, the entire experimental procedure is divided into three phases:

- Phase 1, the  $n$  bit LSB algorithm will be implemented with different integers in a range of [1 ... 8], and the cover image can be selected from the image database. The specific stego image result will be selectively displayed.
- Phase 2, identify the potential threshold (of human visual detection) by analysing the distribution of the measurement units (see below).
- Phase 3, according to the threshold result from Phase 2, the stego image, will be selected from the range of  $n$  that is under the threshold, to verify the applicability and reliability of two proposed solutions.

In this section, the experimental procedure is further discussed below. Prior to implementing the experiment, the corresponding experimental materials are described, and specific equations of measurement units are introduced for the analysis.

#### 4.2.1. Experimental materials and measurement units

##### **Cover Images Database**

In total, there are eight images to be investigated according to their different background colours and image resolution in the experiments. Lena and ModrianTree are typical images frequently used in image processing and image steganography examples. The other images are selected randomly from the website of the research university.

(NB: The large resolution images are filled in red)

	Image type	Size	Resolution	Bit depth
Lena.bmp	Bitmap	768 KB	512 * 512	24
ModrianTree.bmp	Bitmap	3.97 MB	1447 * 959	24
Tiger.bmp	Bitmap	225 KB	320 * 240	24
Cathedral.jpg	JPEG	14.9 KB	253 * 450	24
Edinburgh.jpg	JPEG	68.2 KB	615 * 410	24
Lena.png	PNG	602 KB	512 * 512	32
Modrian.png	PNG	18.1 KB	1024 * 768	32
StABridge.png	PNG	399 KB	534 * 408	32

Table 2: The details of experimental images

##### **Confidential Data**

There are three different sizes of text files (.txt) to be used as the confidential data. The sizes are 20 bytes, 42 KB and 119 KB respectively, as shown in Figure 58:

 dataLarge.txt	12 July 2016 11:48	119 KB	text
 dataSmall.txt	12 July 2016 11:50	42 KB	text
 dataTest.txt	4 May 2016 01:22	20 bytes	text

Figure 58: The details of experimental secret files

##### **Measurement Units**

Three measurement units are used to identify the threshold as well as verify the viability of the proposed solutions. There are *PTR*, *PSNR* and *MSE*.

PTR denotes the “Pixel Transformed Rate”, it represents the replaced (embedded) rate after implementing the LSB algorithm in the cover image. The specific equation derived and presented by this thesis is shown below:

$$PTR = \frac{Bit\_of\_message}{n * Bit\_of\_img} = \frac{A\_char * 8}{M * N * BBP * n}$$

Where  $n$  denotes  $n$  bit LSB algorithm,  $M$  and  $N$  denote the width and height of this image respectively,  $bpp$  denotes the Bit Per Pixel, which gives the resolution of the image. For example, in a colour image, each pixel has 24 bits consisting of three colour channels, Red, Green and Blue.  $A\_char$  denotes the number of characters in the secret data file (including the space and other characters or symbols).

As mentioned in the above Methodology sections,  $PSNR$  and  $MSE$  are the measurement units that represent the correlation between the original image and the current image. The quality of image can be indicated by the value of  $PSNR$  directly as they are proportional. Instead, the  $MSE$  is inversely proportional as it indicates dissimilarity of images. The equations are shown in the formula below [65] [66]:

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

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

Both equations can be implemented in MATLAB directly and the specific process is as below:

- Import both images; original cover image and generated stego image, as shown in Figure 59:

```
>> cover_image = imread('H:\Documents\Database\Exp Result\SmallMessage\PNG\StAPNG50 Result\sta.png');
>> stego_image = imread('H:\Documents\Database\Exp Result\SmallMessage\PNG\StAPNG50 Result\LSB1\LSB1.png');
```

Figure 59: The methods of reading the cover image and the stego image in MatLab

- Call the corresponding functions directly, as shown in Figure 60:

```
>> PSNR = psnr(cover_image, stego_image)

PSNR =

    54.2641
```

Figure 60: The PSNR result between the cover image and the stego image

### Experiment Presentation Summary

Cover	Hidden Data	Algorithm	Embedding Method	Measurement Units	Tools
From image set.	dataLarge.txt, dataSmall.txt, dataTest.txt.	LSB 1 ~ 8	Vertical, Randomness	PTR, MSE, PSNR.	Developed Tools, MatLab

In the experiments, the relevant manipulations are implemented by the developed tools, whether in the embedding procedure or in the extraction procedure. The measurement units PSNR and MSE are computed by MATLAB directly through its built-in functions.

#### 4.2.2. Phase 1: The Implementation of the LSB Replacement Algorithm

The implementation was based on the size of confidential data, the algorithm was chosen from LSB 1 to LSB 8 in vertical way to implement the data replacement, and the cover images are selected from the cover image database. In this sub-section, the results of newly generated stego images were displayed for human visual detection (comparison). Also, in order to compare the visual difference between the vertical embedding and random embedding, the relevant randomness replacement manipulations were performed with a large size data file, and the result of newly generated stego images were displayed in the end of this sub-section.

*(NB: In this sub-section, only the results of LSB 1, LSB 3 and LSB 5 will be shown for brevity, and the rest of the image results can be found in Appendix B)*

**In dataTest.txt file, (20 bytes)**

*Lena.bmp*



*Figure 61: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with very short size data file.*

*MondrianTree.bmp*



*Figure 62: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.bmp, with very short size data file.*

*Tiger.bmp*



*Figure 63: The stego image results in LSB 1, LSB 3 and LSB 5, in Tiger.bmp, with very short size data file.*

*Cathedral.jpg*



*Figure 64: The stego image results in LSB 1, LSB 3 and LSB 5, in Cathedral.jpg, with very short size data file.*

*Edinburgh.jpg*



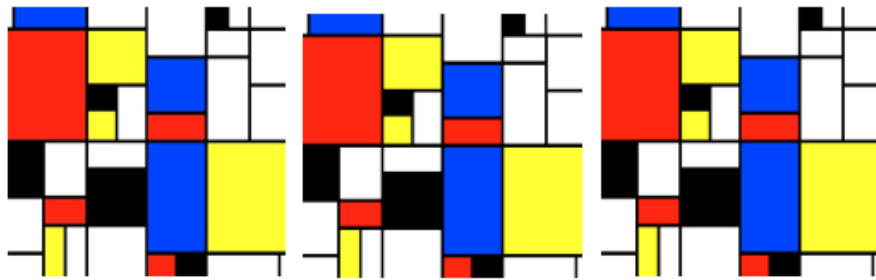
*Figure 65: The stego image results in LSB 1, LSB 3 and LSB 5, in Edinburgh.jpg, with very short size data file.*

*Lena.png*



*Figure 66: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with very short size data file.*

*Mondrian.png*



*Figure 67: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.png, with very short size data file.*

*StABridge.png*



*Figure 68: The stego image results in LSB 1, LSB 3 and LSB 5, in StABridge.png, with very short size data file.*

From the above experimental images (Figures 61 - 68), the variation that is caused by the LSB algorithm is not appreciable, as the embedding data is quite small, only 20 bytes. By inspecting the experimental result of LSB 8 on StABridge.png closely, at the top left hand corner of the image, some small bold "black point pixels" can be viewed. This is caused by the injection of the secret data, as shown in Figure 69 below:



*Figure 69: The position of hidden (very short size) message file in the cover image, StABridge.png*



**In dataSmall.txt file, (42 KB)**

*Lena.bmp*



*Figure 70: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with a small size data file.*

*MondrianTree.bmp*



*Figure 71: The stego image results in LSB 1, LSB 3 and LSB 5, in MondrianTree.bmp, with a small size data file.*

*Tiger.bmp*



*Figure 72: The stego image results in LSB 1, LSB 3 and LSB 5, in Tiger.bmp, with a small size data file.*

*Cathedral.jpg*



*Figure 73: The stego image results in LSB 1, LSB 3 and LSB 5, in Cathedral.jpg, with a small size data file.*

*(NB: An appreciable variation can be viewed from the left side of image, particularly in the experimental image result of LSB 5 in Figure 73.)*

*Edinburgh.jpg*



*Figure 74: The stego image results in LSB 1, LSB 3 and LSB 5, in Edinburgh.jpg, with a small size data file.*

*Lena.png*



*Figure 75: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with a small size data file.*

*Mondrian.png*

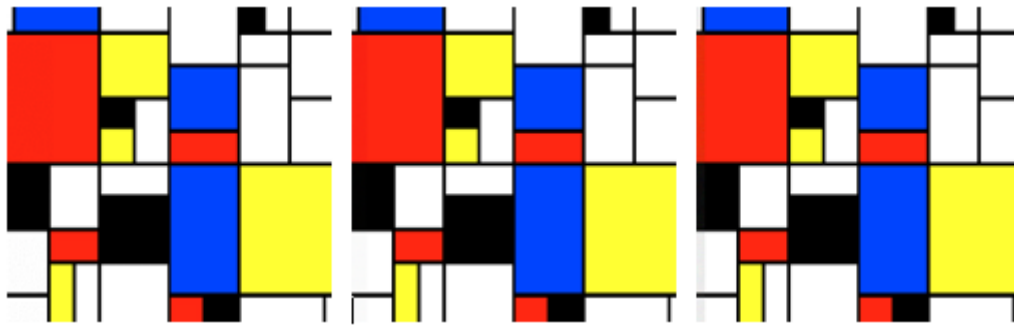


Figure 76: The stego image results in LSB 1, LSB 3 and LSB 5, in *Mondrian.png*, with a small size data file.

*StABridge.png*



Figure 77: The stego image results in LSB 1, LSB 3 and LSB 5, in *StAaBridge.png*, with a small size data file.

*(NB: Some variations can be viewed from the left side of image, particularly in the experimental image result of LSB 5 in Figure 77.)*

From the above experimental images, compared with the result of embedding a “testing” file, the variation can be distinguished easily. By inspecting the left side of the image, the affected areas, which are caused by the injection of embedding data, have been pixelated/distorted with the increase in replaced bits in the LSB algorithm. This can be seen on the top left corner of Figure 73 above in the right hand side image. Particularly, the visual result is obvious in the bright (background) images, such as “*StABridge.png*”. In the result of LSB 3, this area becomes “smooth”; in the result of LSB 5, this area is extremely visible; in the result of LSB 8, this area is substituted by “Black” colour, as show in Figures 78 & 79 below:



Figure 78: The distortion areas of the stego image in LSB 3



Figure 79: The distortion areas comparison in LSB 3, LSB 5 and LSB 8

**In dataLarge.txt file, (119 KB)**

*Lena.bmp*



Figure 80: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.bmp, with a large size data file.

*MondrianTree.bmp*



*Figure 81: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.bmp, with a large size data file.*

*Tiger.bmp*



*Figure 82: The stego image results in LSB 3 and LSB 5, in Tiger.bmp, with a large size data file. LSB 1 is not enough for concealing the secret message*

*Cathedral.jpg*



*Figure 83: The stego image results in LSB 3 and LSB 5, in Cathedral.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file.*

*Edinburgh.jpg*



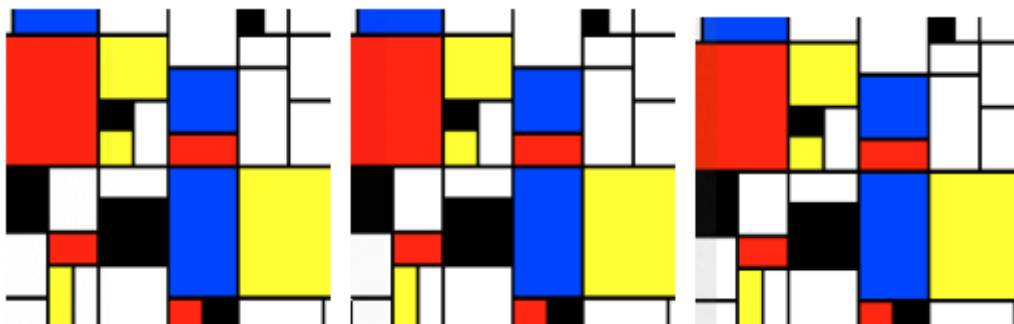
*Figure 84: The stego image results in LSB 3 and LSB 5, in Edinburgh.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file.*

*Lena.png*



*Figure 85: The stego image results in LSB 1, LSB 3 and LSB 5, in Lena.png, with a large size data file.*

*Mondrian.png*



*Figure 86: The stego image results in LSB 1, LSB 3 and LSB 5, in Mondrian.png, with a large size data file.*

*StABridge.png*



*Figure 87: The stego image results in LSB 3 and LSB 5, in StABridge.jpg, with a large size data. LSB 1 is not sufficient to conceal the secret message file.*

From the above experimental images, the area of variation is intensively distributed in the stego image. There is a significant variation as the size of the embedding file is up to 119 KB. Again, the appreciable difference can be viewed from the left side of the image. The visual result is extremely obvious in the bright (background) colour images. Particularly, in the higher replaced bit LSB algorithm.

***In Randomness (with large size data), (119 KB)***

Unlike the vertical replacement, the replaced position was not picked one by one from the beginning of the image (top left corner). The system will compute the amount of required pixels according to the specific algorithm and the length of confidential data, then the position of these pixels will be chosen randomly from the cover image. The corresponding replacement manipulations are executed in these collected pixels.

*Lena.bmp*



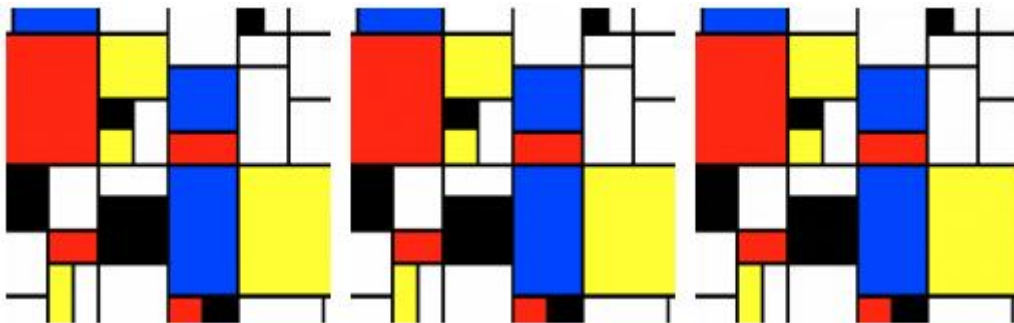
*Figure 88: The stego image results in LSB1, LSB 3 and LSB 5, in Lena.bmp, with a large size data file with a random embedding algorithm.*

*Cathedral.jpg*



*Figure 89: The stego image results in LSB 3 and LSB 5, in Cathedral.jpg, with a large size data in randomly embedded. LSB 1 is not sufficient to conceal the secret message file.*

*Mondrian.png*



*Figure 90: The stego image results in LSB1, LSB 3 and LSB 5, in Mondrian.png, with a large size file with a random embedding algorithm.*

*StABridge.png*



*Figure 91: The stego image results in LSB 3 and LSB 5, in StABridge.png, with a large size data file with a random embedding algorithm. LSB 1 is not sufficient to conceal the secret message file.*



From the above experimental images, unlike the horizontal or vertical embedded result, the visual difference is relatively reduced as the area of variation is averagely distributed in the stego image. As shown in Figure 92 & 93 below, the visual result is compared in StABridge.png with the algorithm LSB5 between the horizontal embedding and randomness embedding.

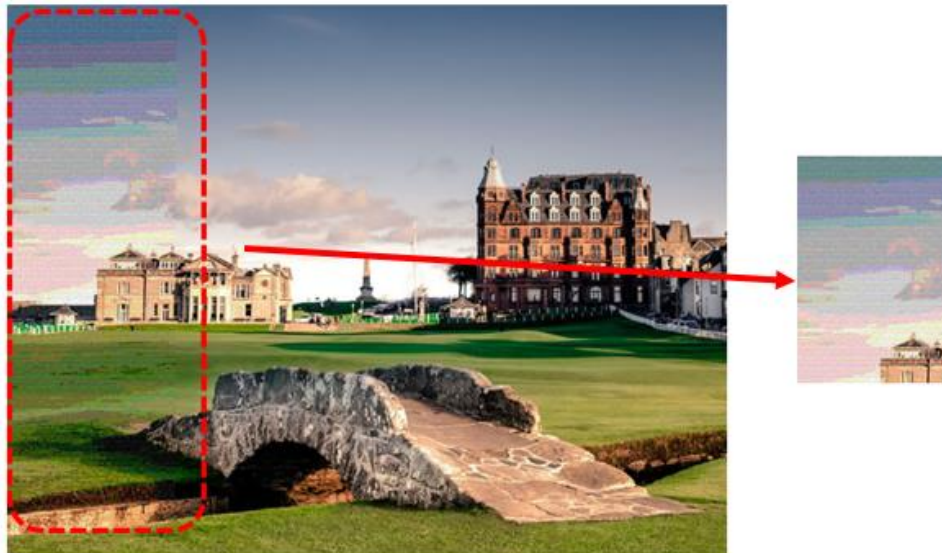


Figure 92: The stego image visual result in LSB 5, with a large size of data in vertically embedded.



Figure 93: The stego image visual result in LSB 5, with a large size of data file with a random embedding algorithm .

### 4.2.3. Phase 2: Determine the potential visual Threshold

As the size of the embedding data increases the affected areas of the image become more visually evident. These areas have become distorted, with the application of replaced bits during the embedding process, especially with the embedding of large data files. Thus, the stego algorithm can be seen as a procedure that injects noise into the cover image [51].

The quality of a stego image is inversely proportional to the volume of injected noise. By looking for the existing distortion, the stego image that is injected and the associated noise can be distinguished from the cover image. Fyffe [64] presented good literature references to describe and verify the human visual based perception for steganography as well as a discussion on stego images, LSB algorithms and detection via distortion rates, especially when the replaced bits exceed more than LSB3. However, this result cannot be seen as the identification of the LSB algorithm threshold in the steganographic technique. The identification result of distortion can be affected by various factors, including the colour intensity of image background and the distribution of embedded data, as mentioned in the above sub-section. Therefore, the value of the detection threshold should be identified by combining more factors, such as the relevant image measurement units, rather than the boundary of imperceptible distortion (noise) within human visual perception only.

As shown in the Tables 3 – 5 below, the Pixel Transfers Rates (*PTR*) value was calculated based on the LSB algorithm under the different conditions, such as the replaced bits variable and different secret data files. The value of *PTR* undergoes a significant decrease when there is an increase of replaced bits *n*. Therefore, the measurement unit *PTR* is proportional to the size of embedded data, but it is inversely proportional to the replaced bit *n*.

PTR for Images with Different LSB replacements for a very short secret message (20 bytes)								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lena.bmp	0.02%	0.01%	0.006%	0.005%	0.004%	0.003%	0.0029%	0.0025%
ModrianTree.bmp	0.0038%	0.0019%	0.0013%	0.0009%	0.0008%	0.0006%	0.0005%	0.0004%
Tiger.bmp	0.069%	0.034%	0.023%	0.017%	0.014%	0.012%	0.009%	0.008%
Cathedral.jpg	0.047%	0.023%	0.016%	0.012%	0.01%	0.008%	0.007%	0.006%
Edinburgh.jpg	0.021%	0.011%	0.007%	0.005%	0.004%	0.0035%	0.003%	0.0026%
Lena.png	0.02%	0.01%	0.006%	0.005%	0.004%	0.003%	0.0029%	0.0025%
StABridge.png	0.024%	0.012%	0.008%	0.006%	0.005%	0.004%	0.0035%	0.003%

Table 3: The result of PTR with different LSB replace bits, for a short size data file.

PTR for Images with Different LSB replacements for a small size message (42 KB)								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	40.61%	20.30%	13.54%	10.15%	8.12%	6.77%	5.80%	5.08%
ModrianTree.bmp	7.67%	3.84%	2.56%	1.92%	1.53%	1.28%	1.09%	0.96%
Tiger.bmp		69.30%	46.20%	34.65%	27.72%	23.10%	19.80%	17.32%
Cathedral.jpg	93.49%	46.75%	31.16%	23.37%	18.70%	15.58%	13.36%	11.69%
Edinburgh.jpg	42.21%	21.11%	14.07%	10.55%	8.44%	7.04%	6.03%	5.28%
Lena.png	40.61%	20.30%	13.54%	10.15%	8.12%	6.77%	5.80%	5.08%
StABridge.png	48.86%	24.43%	16.29%	12.21%	9.77%	8.14%	6.98%	6.10%

Table 4: The result of PTR with different LSB replace bits, for a small size data file.

PTR for Images with Different LSB replacements for a large size message (119 KB)								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	97.49%	48.74%	32.50%	24.37%	19.50%	16.25%	13.93%	12.19%
ModrianTree.bmp	18.42%	9.21%	6.14%	4.60%	3.68%	3.07%	2.63%	2.30%
Tiger.bmp				83.19%	66.55%	55.46%	47.54%	41.59%
Cathedral.jpg			74.83%	56.12%	44.89%	37.41%	32.07%	28.06%
Edinburgh.jpg		50.68%	33.78%	25.34%	20.28%	16.89%	14.48%	12.67%
Lena.png	97.49%	48.74%	32.50%	24.37%	19.50%	16.25%	13.93%	12.19%
StABridge.png		58.65%	39.10%	29.32%	23.46%	19.55%	16.76%	14.66%

Table 5: The result of PTR with different LSB replace bit, for a large size data file.

Next, as shown in Table 6 – 8 below, the *PSNR* is presented with different LSB algorithms, which have been implemented with the very short data file (20 bytes), small size data file (42 KB) and large size data file (119KB) respectively. Normally, the *PSNR* represents the quality of image after injection of the noise. Here, this value denotes the quality of the stego image, and it shows a regular decrease through increasing replaced bits  $n$ .

Images in Different LSB algorithm with a short size data (20 bytes), PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	87.8966	84.1428	80.565	74.8188	70.7913	64.3026	56.635	54.4044
ModrianTree.bmp	95.5743	92.15	86.843	81.5995	77.8044	71.644	65.8269	64.6837
Tiger.bmp	83.1823	78.6807	74.7833	68.8821	64.1296	56.5398	55.5495	55.455
Cathedral.jpg	84.7147	80.0609	77.7599	70.1228	65.6962	63.2659	53.3547	57.9495
Edinburgh.jpg	87.9423	83.8223	79.0867	75.6801	70.2106	65.0496	59.5871	58.0296
Lena.png	87.8966	84.1428	80.565	74.8188	70.7913	64.3026	56.635	54.4044
Modrian.png	92.0814	86.3932	81.0704	74.9505	70.9872	65.3621	59.0501	53.8722
StABridge.png	87.5334	83.4737	79.3998	74.9614	69.9242	64.7897	55.2974	59.3559

Table 6: The result of PSNR with different LSB replace bit, for a short size data file.

Images in Different LSB algorithm with a Small size data (42 KB), PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	55.0502	50.5015	46.7445	41.7662	37.0966	31.3968	27.1656	23.1781
ModrianTree.bmp	62.3016	57.7675	54.0163	49.2764	44.2707	38.0667	34.0729	31.6379
Tiger.bmp		45.1958	41.4377	36.5983	31.4848	25.8304	21.5112	22.3038
Cathedral.jpg	51.4481	46.9013	43.1292	38.4403	33.3533	27.5782	22.5752	19.9057
Edinburgh.jpg	54.8853	50.3406	46.5321	41.6851	36.6524	31.1613	26.0346	22.0725
Lena.png	55.0502	50.5015	46.7445	41.7662	37.0966	31.3968	27.1656	23.1781
Modrian.png	59.774	54.0478	49.3926	44.5531	39.4495	33.8858	28.639	22.7656
StABridge.png	54.2641	49.7087	45.9368	41.2664	36.2349	30.5299	25.6643	20.2394

Table 7: The result of PSNR with different LSB replace bit, for a small size data file.

Images in Different LSB algorithm with a Large size data (100 KB), PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	51.2448	46.7001	42.9225	37.9999	33.3183	27.6487	22.239	17.1854
ModrianTree.bmp	58.491	53.9527	50.203	45.4608	40.4743	34.4072	30.1762	27.8316
Tiger.bmp				32.8255	27.7445	22.2494	17.7332	16.4262
Cathedral.jpg			39.3308	34.6137	29.5077	23.5822	18.648	15.8614
Edinburgh.jpg		46.5208	46.6918	37.8154	32.8547	27.3779	22.2979	18.5529
Lena.png	51.2448	46.7001	42.9225	37.9999	33.3183	27.6487	22.239	17.1854
Modrian.png	55.879	50.2433	45.6369	40.8125	35.6901	30.1468	24.9079	19.2195
StABridge.png		45.8939	42.119	37.4525	32.3639	26.7355	21.7479	16.2661

Table 8: The result of PSNR with different LSB replace bit, for a large size data file.

According to the details of the above Tables 6 – 8, the corresponding *PSNR* distribution charts can be generated, as shown in Figure 94 – 96 below. Appreciably, a linear decrease can be viewed with the increasing replaced bits. Also the value of *PSNR* depends on the size of injected noise (data).

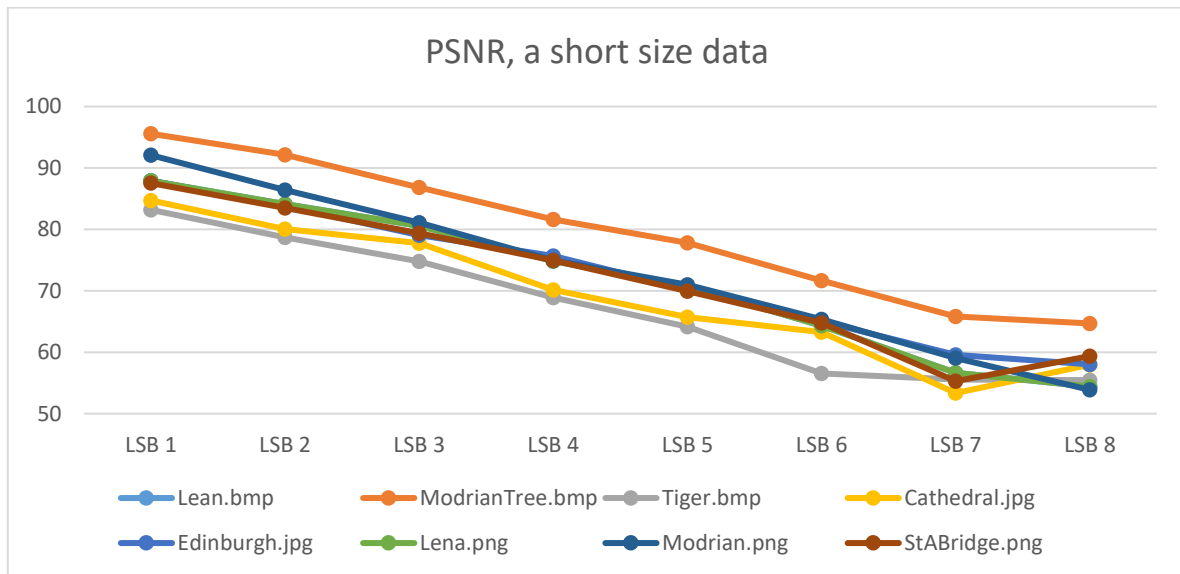


Figure 94: The *PSNR* distribution of experimental images, embedded a very short size data file. (20 bytes)

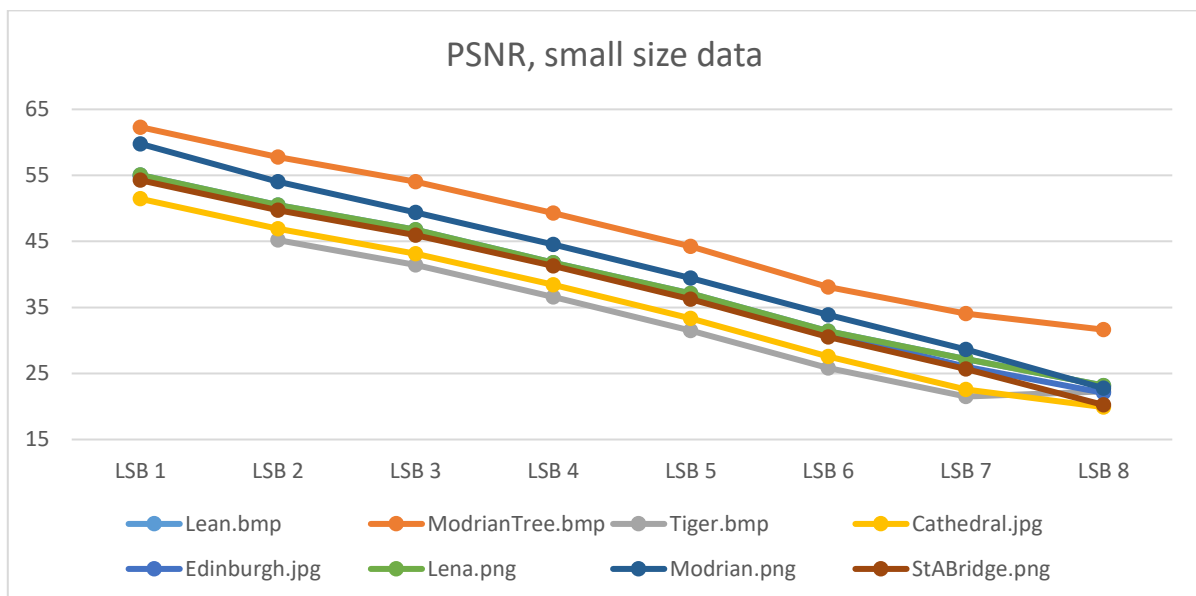


Figure 95: The *PSNR* distribution of experimental images, embedded a small size data file. (42 KB)

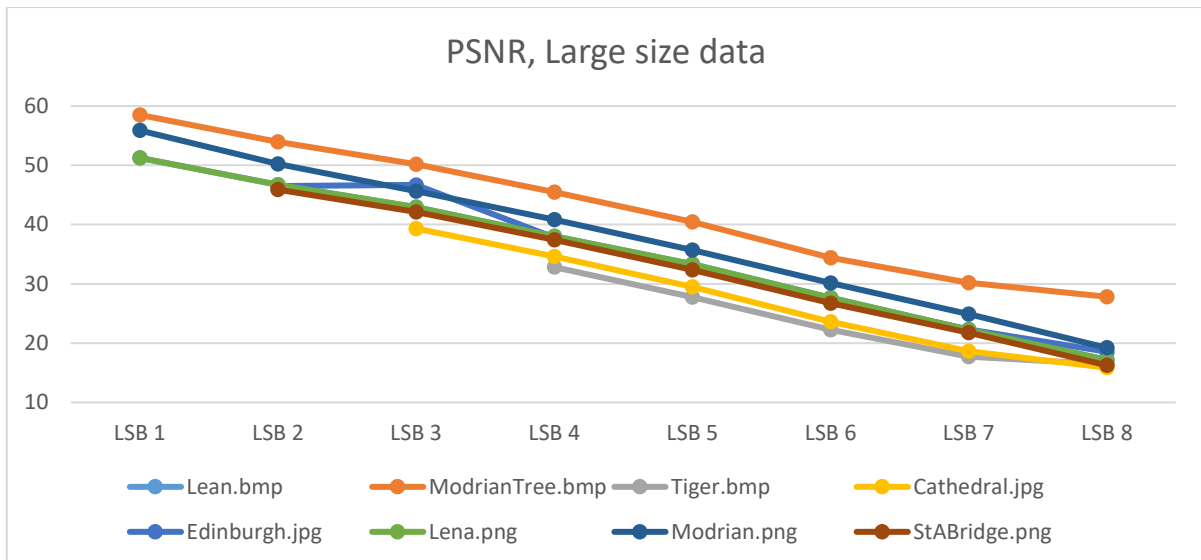


Figure 96: The PSNR distribution of experimental images, embedded a large size data file. (119 KB)

However, it is difficult to identify the threshold of an LSB replacement algorithm from the distribution of *PSNR* due to its linear change. Although, these charts demonstrate that the quality of image has been reduced towards the higher LSB end with the increase of replaced bits  $n$ , a significant difference is hard to identify. Thus, the second measurement unit was implemented to identify this threshold. As shown in Tables 9 – 11 below, the *MSE* is presented with different LSB algorithms, which were implemented with a very short data (20 bytes), a small size data (42 KB) and a large size of data (119 KB) respectively. The *MSE* represents the difference between the cover image and the stego image and is inversely proportional to the replaced bits  $n$ .

Images in Different LSB algorithm with very short size message (20 bytes), MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	0.0001	0.0003	0.0006	0.0021	0.0054	0.0241	0.1411	0.2359
ModrianTree.bmp	1.80E-05	3.96E-05	1.35E-04	4.50E-04	0.0011	4.50E-03	1.70E-02	0.0221
Tiger.bmp	3.13E-04	8.81E-04	0.0022	0.0084	0.0251	0.1442	0.1812	0.1852
Cathedral.jpg	2.20E-04	6.41E-04	0.0011	0.0063	0.0175	0.0307	0.3003	0.1043
Edinburgh.jpg	1.04E-04	2.70E-04	8.02E-04	0.0018	0.0062	0.0203	0.0715	0.1024
Lena.png	1.06E-04	2.51E-04	5.71E-04	0.0021	0.0054	0.0241	0.1411	0.2359
Modrian.png	4.03E-05	1.49E-04	5.08E-04	0.0021	0.0052	0.0189	0.0809	0.2666
StABridge.png	1.15E-04	2.92E-04	7.47E-04	0.0021	0.0066	0.0216	0.192	0.0754

Table 9: The result of MSE with different LSB replace bit, for a short size data file.

Images in Different LSB algorithm with Small size message (42 KB), MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	0.2033	0.5793	1.376	4.3297	12.6888	47.1416	124.8889	312.8001
ModrianTree.bmp	0.0383	0.1087	0.2579	0.7681	2.4323	10.1488	25.4561	44.5957
Tiger.bmp		1.9656	4.67	14.2313	46.1955	169.8421	459.1533	382.5622
Cathedral.jpg	0.4659	1.3272	3.1635	9.3122	30.0436	113.568	359.385	664.5241
Edinburgh.jpg	0.2111	0.6012	1.445	4.4114	14.0552	49.7685	162.0401	403.4895
Lena.png	0.2033	0.5793	1.376	4.3297	12.6888	47.1416	124.8889	312.8001
Modrian.png	0.0685	0.256	0.7479	2.2791	7.3812	26.5764	88.9567	343.9684
StABridge.png	0.2436	0.6954	1.6573	4.8578	15.4735	57.5563	176.46	615.3696

Table 10: The result of MSE with different LSB replace bit, for a small size data file.

Image in Different LSB algorithm with Large size message (119 KB), MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
Lean.bmp	0.4882	1.3902	3.3177	10.306	30.2869	111.7404	388.3093	1.24E+03
ModrianTree.bmp	0.092	0.2617	0.6206	1.8493	5.8298	23.5699	62.44	107.1327
Tiger.bmp				33.9258	109.3019	387.3826	1.10E+03	1.48E+03
Cathedral.jpg			7.5857	22.4754	72.8299	285.0088	887.7255	1.69E+03
Edinburgh.jpg		1.4488	3.4986	10.7534	33.6985	118.9304	383.0841	907.3879
Lena.png	0.4882	1.3902	3.3177	10.306	30.2869	111.7404	388.3093	1.24E+03
Modrian.png	0.168	0.6148	1.7758	5.393	17.5417	62.8633	210.0325	7.78E+02
StABridge.png		1.6737	3.9919	11.6903	37.7299	137.8901	434.8029	1.54E+03

Table 11: The result of MSE with different LSB replace bit, for a large size data file.

According to the details of the above Tables 9 – 11, the corresponding MSE distribution charts are generated, as shown in Figure 97 – 99. However, this distribution is not similar to the PSNR's; the change is not regular and can be divided into three phase:

- Phase 1, from LSB 1 to LSB 3, the distribution is horizontal across the X axis.
- Phase 2, from LSB 3 to LSB 4, there is a negligible rise.
- Phase 3, after the LSB 4, there is a significant increase.

(NB: The distribution of LSB 8 may be affected by image background intensity.)

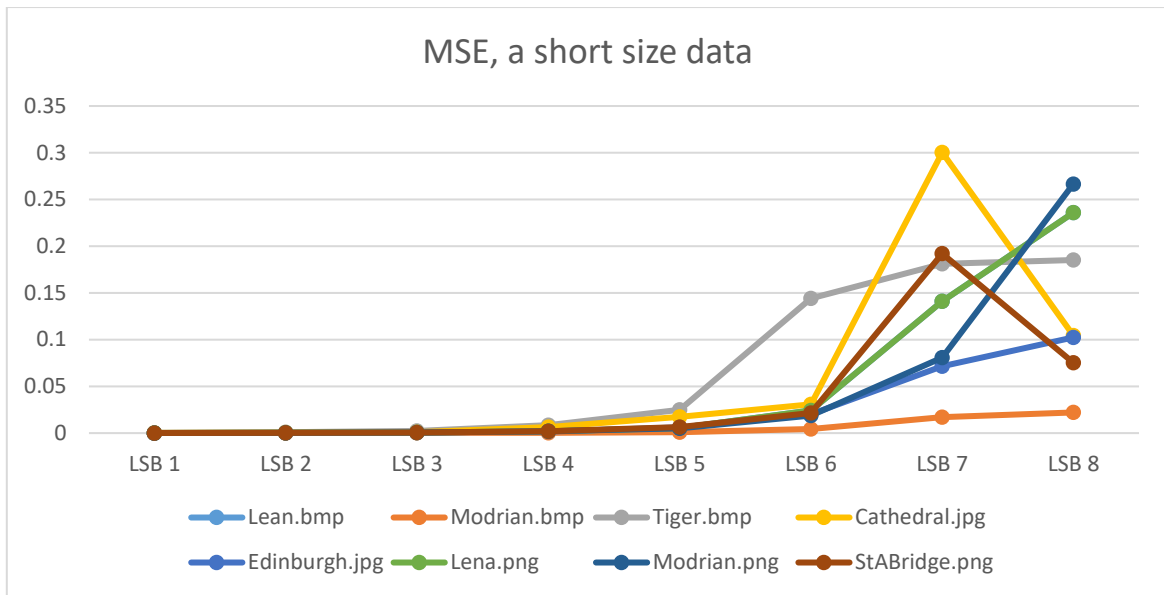


Figure 97: The MSE distribution of experimental images, embedded a very short size data file. (20 bytes)

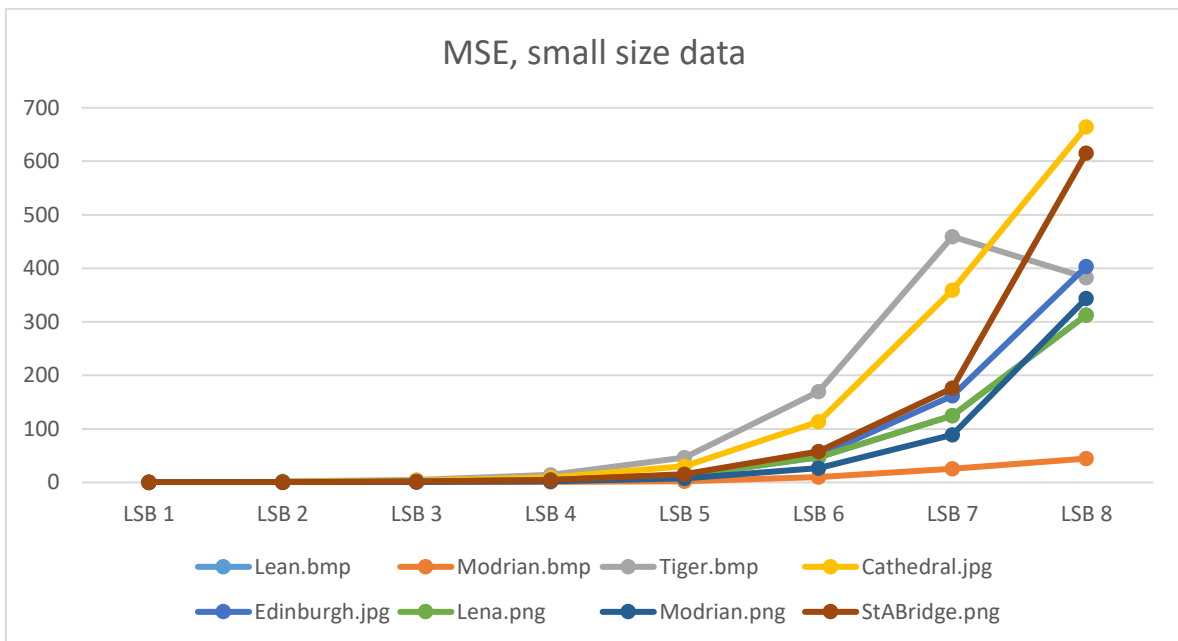


Figure 98: The MSE distribution of experimental images, embedded a small size data file. (42 KB)



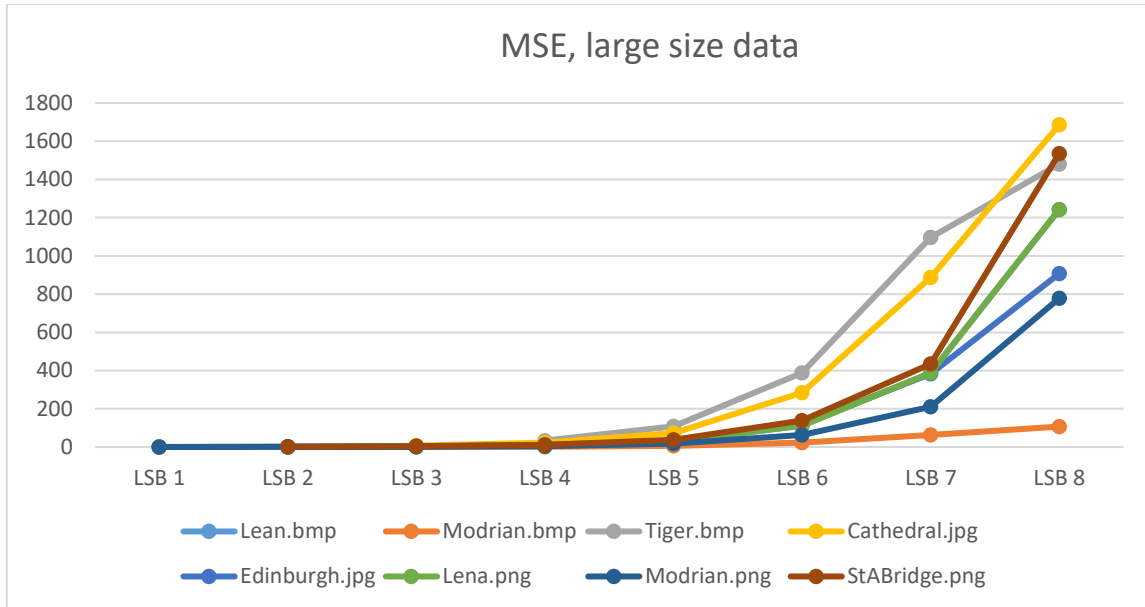


Figure 99: The MSE distribution of experimental images, embedded a large size data file. (119 KB)

Therefore, according to the above distribution charts, the PSNR chart presents the reduction of image quality with the increase of replaced bits  $n$ , which means the stego image is becoming more distorted. In the MSE charts, due to its irregular distribution, the difference value in LSB 3 and LSB 4 can be seen as the turning point in the overall profiles. Combined with the result a thesis investigating human perception of LSB steganography[64], the threshold of the LSB replacement algorithm can be identified by relevant measurement units as well as by human perception, near to LSB 3.

(NB: There are some distribution results for the random position data embedding, for the relevant details please check the section, Appendix C).

#### 4.2.4. Phase 3: Verify the applicability and reliability of solutions

##### Experiment 1:

Cover Image	Secret Data	Algorithm
StABridge.png	dataTest.txt (20 bytes)	LSB1

As shown in Figure 100 below, a visual comparison is made between the original PNG image (left) and stego LSB-1 PNG image after embedding data (right).



Figure 100: The visual comparison between the original image and the stego image in LSB 1

The colour frequency histograms between the two images above were compared. The distribution, in this example, was almost identical as the size of embedded data was negligible, as shown in Figure 101 below:

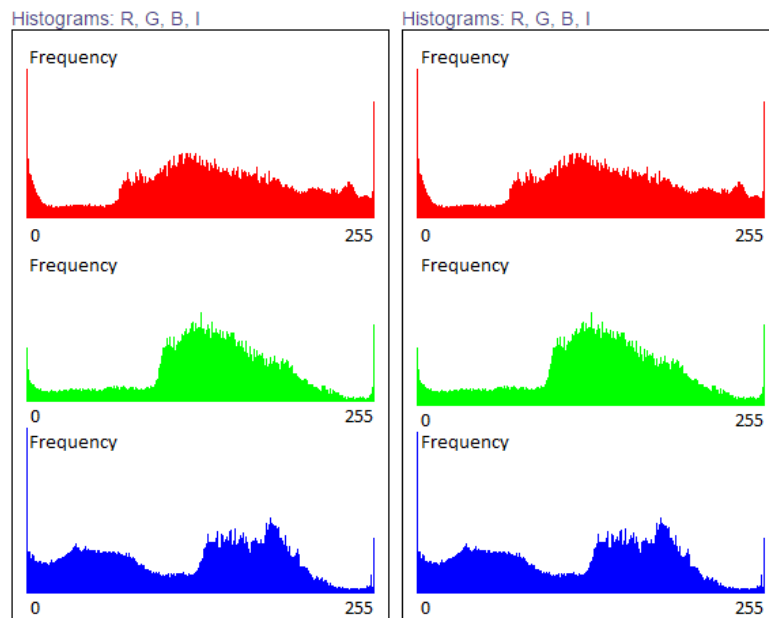


Figure 101: The histogram comparison between the cover image and stego image in LSB 1

Next, embedded data was extracted from the above stego image according to its **Key** information (LSB 1), and the secret data can be read accurately, as shown in Figure 102 below:

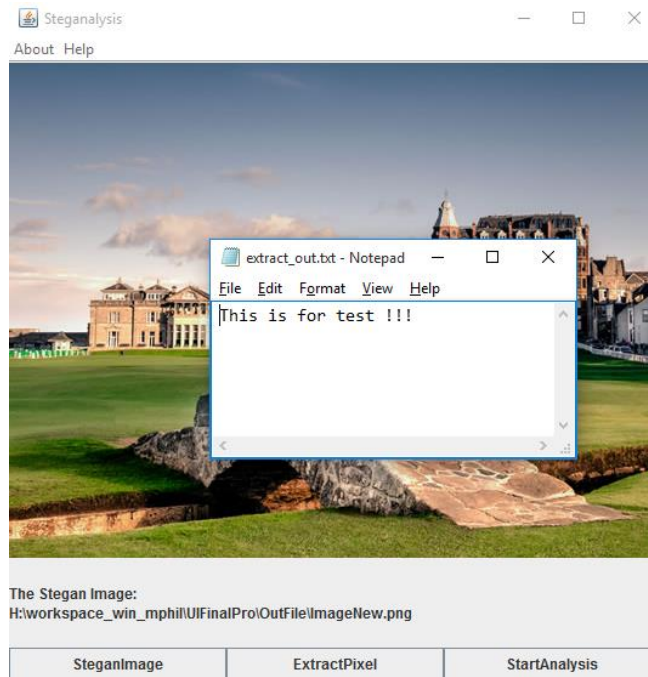


Figure 102: The extracted result from the stego image by using the stego key

### ***Solution 1: Presentation Domain Transformation (PDT)***

Converting the stego image to a frequency domain, which is from a PNG file format to a JPEG. The visual and histogram chart between the original stego image (PNG in left) and the newly converted stego image (JPEG in right) are compared to establish differences and variations, as shown in Figures 103 & 104 below. Although under visual comparison the images appear identical, the colour frequency distribution has been changed under the conversion as seen in the histogram comparison and is noted by the red circles in the chart.



Figure 103: The visual comparison between the Original stego image and the **Transformed** stego image

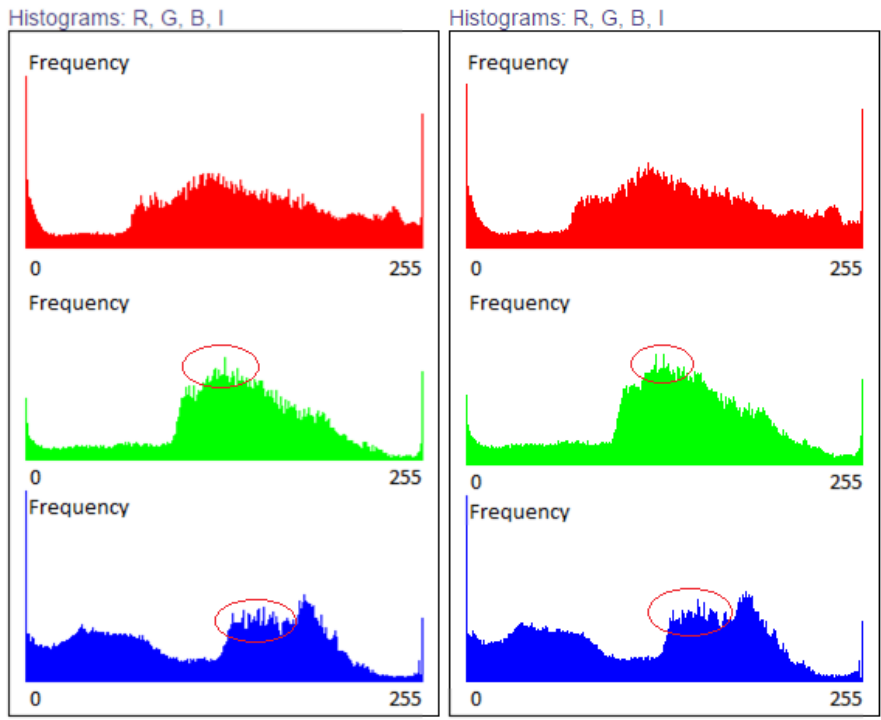


Figure 104: The histogram comparison between the Original stego image and the **Transformed** stego image

The same key (LSB 1) is used to extract the embedded data from the newly generated image (.jpg), presented in the frequency domain, as shown in the Figure 105 below. Saving the image in JPEG format can destroy the embedded message due to the manipulation of the JPEG (frequency domain) compression.

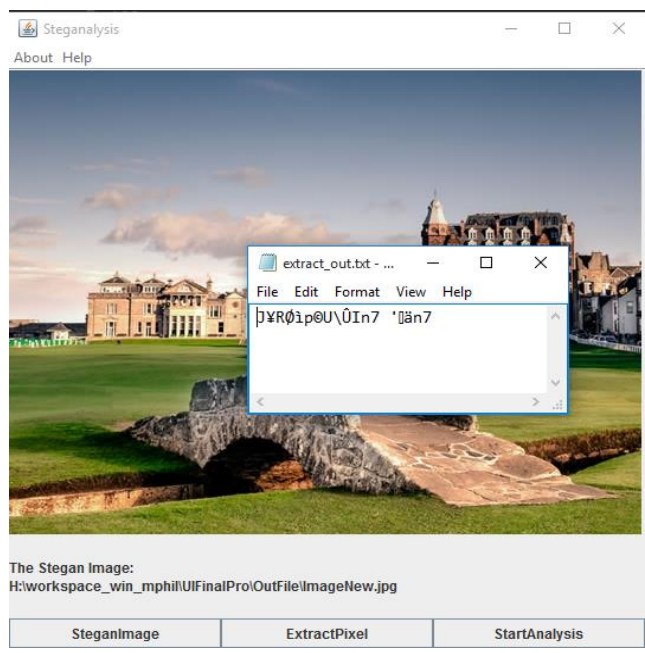


Figure 105: The extracted result from the Transformed stego image by using the same stego key

In addition, due to the lossy compression, saving this image back to the spatial domain (e.g. BMP, PNG) will not recover the message, as shown in Figure 106:

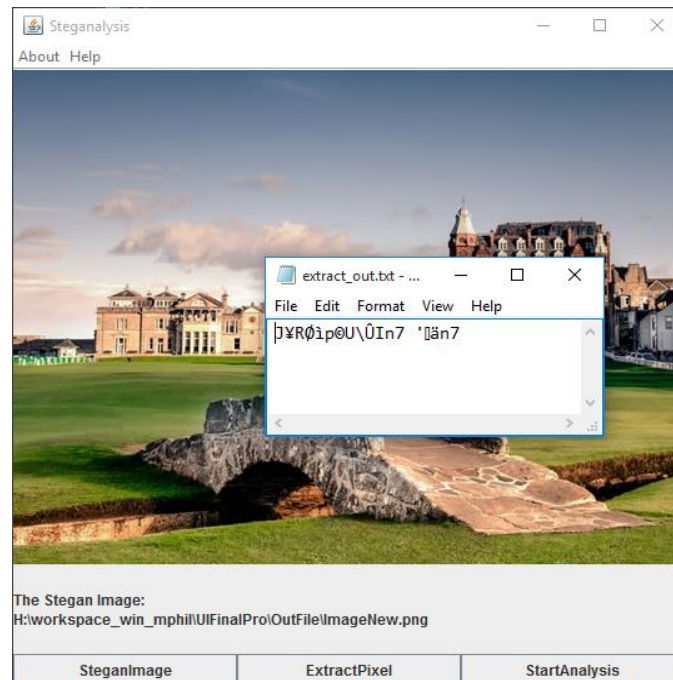


Figure 106: The extracted result after transform back to the previous image format.

### Solution 2: Chrominance Modification Algorithm (CMA)

The modified stego image can be generated from the original stego image due to the CMA algorithm. This newly generated image has a negligible difference in colour intensity as its value is affected through changing the value of chrominance. As shown in Figures 107 & 108 below, the comparison of both the visual and histogram chart between the original stego image (left) and the newly modified stego image (right) is established to look for the differences and variations. As with the PDT solution, although under visual comparison the images appear identical, the colour frequency distribution has been changed under the conversion and is noted by the red rectangles in the blue frequency chart of the histograms in Figure 108.



Figure 107: The visual comparison between the Original stego image and the **Modified** stego image

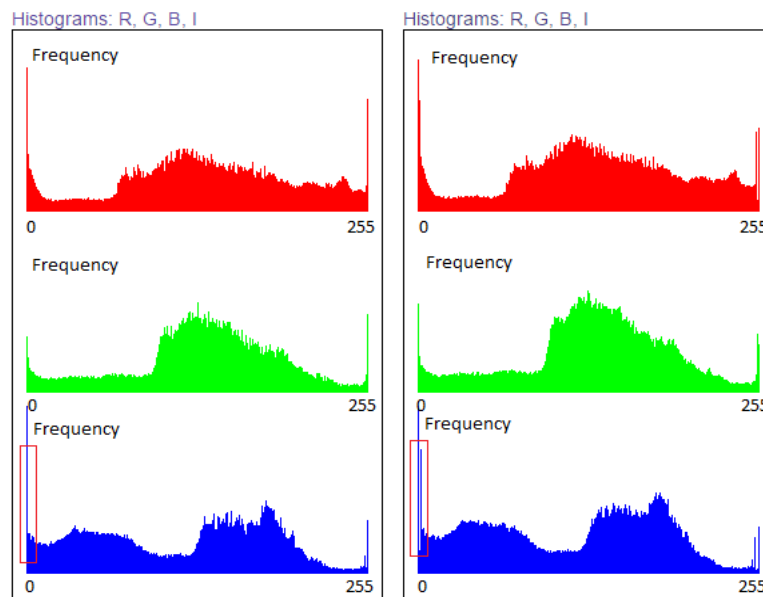


Figure 108: The histogram comparison between the Original stego image and the **Modified** stego image

By using the same key (LSB 1) to extract the embedded data from the newly modified stego image, the data is unreadable, as shown in the Figure 109 below.

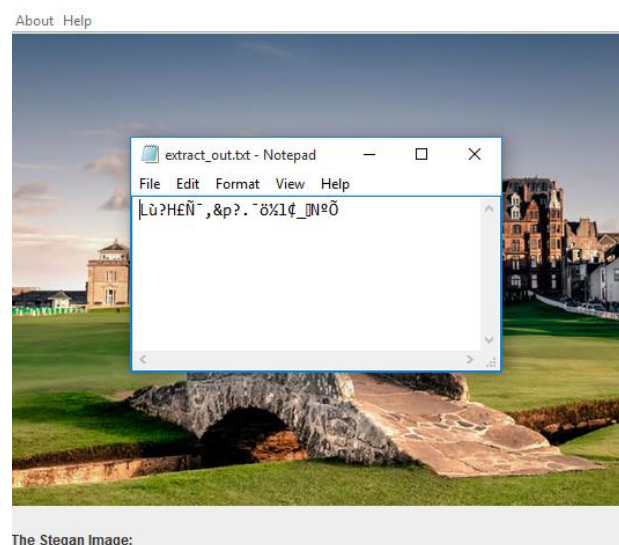


Figure 109: The extracted result from the Modified stego image by using the same stego key

In addition, the secret data after this modification is still unrecoverable due to the randomness in the algorithm modification during execution.

**Experiment 2:**

Cover Image	Secret Data	Algorithm
StABridge.png	dataLarge.txt (119 KB)	LSB 2

Firstly, the comparisons between the visual and histogram are established, as seen in Figures 110 & 111. Results show that the visuals are the same, due to the replacement of the lowest bits that cannot significantly affect the quality of image. The differences, however, can be easily detected by using histogram detection as the hidden data, in this case, has a large volume.



Figure 110: The visual comparison between the cover image and the stego image in LSB 2

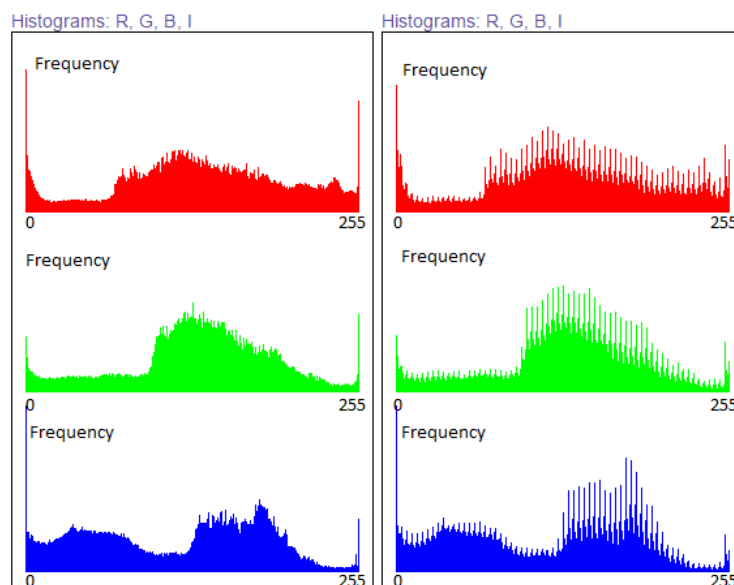


Figure 111: The histogram comparison between the cover image and the stego image in LSB 2

### Solution 1: PDT

Convert the stego image format from PNG to JPEG. Although the images under visual comparison appear identical, the histogram has a significant difference in the blue colour channel as the converted JPEG image has a less ragged profile, as shown in Figure 112.

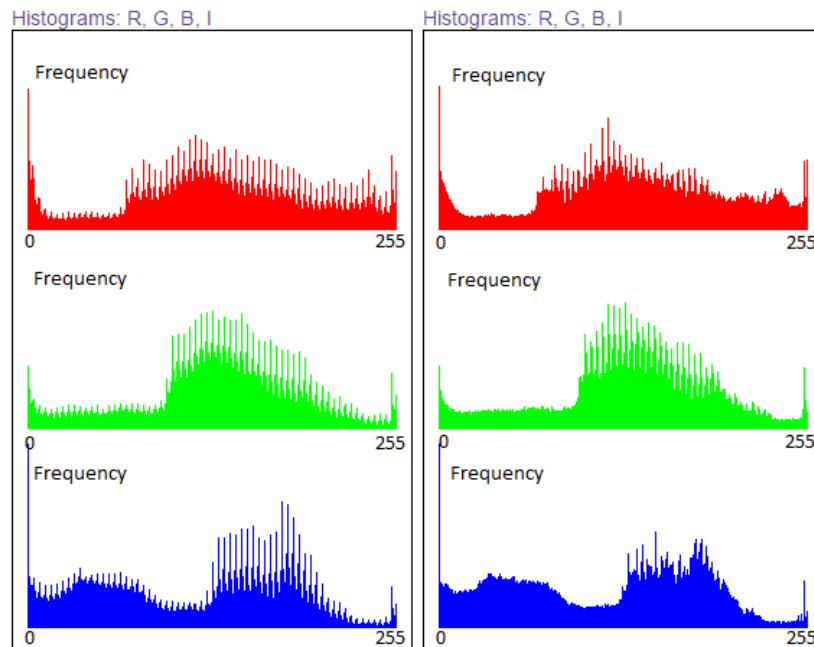


Figure 112: The histogram comparison between the Original stego image and the **Transformed** stego image

Next, comparing the extracted result between these two images in the Figure 113 below. The results indicated that the extracted data was still not readable after converting back to the previous image format.

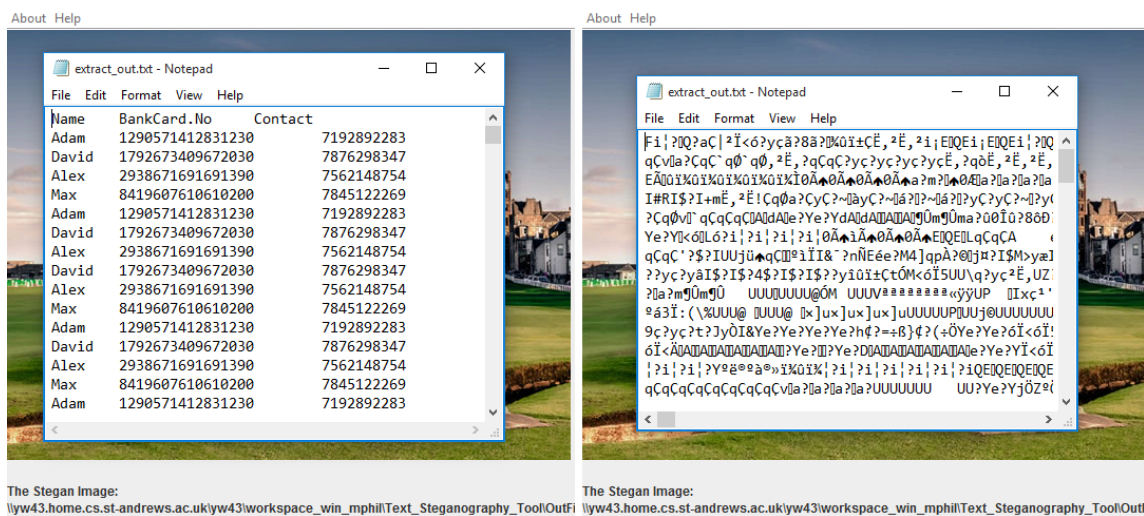


Figure 113: The comparison of extracted result between the Original stego image and the **Transformed** stego image by using the same stego key



**Solution 2: CMA**

As shown in Figure 114, the comparison between the stego image and modified stego image can be established after the execution of the CMA algorithm. Although the difference cannot be distinguished from the visual detection, it can be found in the colour frequency histogram comparison, see Figure 115.

*(NB: This histogram difference is appreciably lower than the converted stego image in solution1.)*



Figure 114: The visual comparison between the Original stego image and the **Modified** stego image

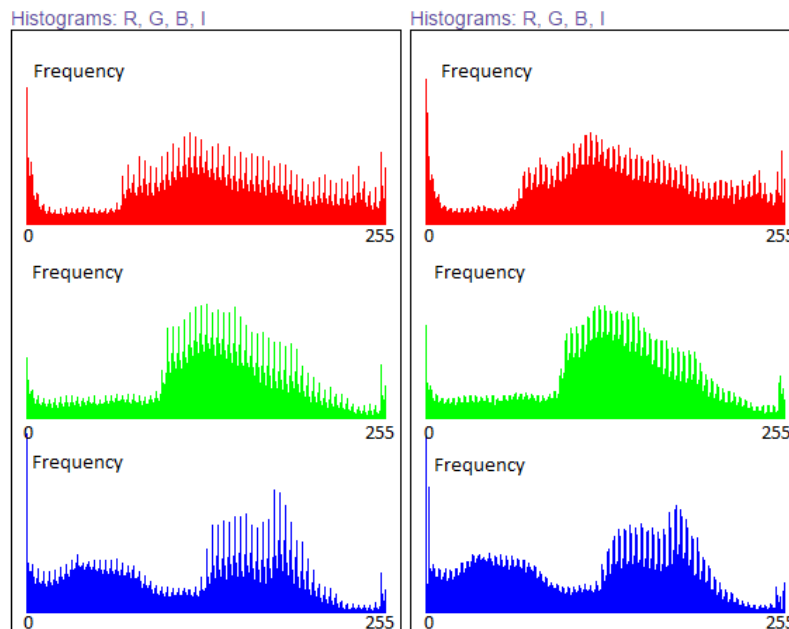


Figure 115: The histogram comparison between the Original stego image and the **Modified** stego image

The data extracted result is still unreadable due to the change of colour intensity, as shown in Figure 116. Also, this data is unrecoverable because of the randomness in the CMA algorithm.

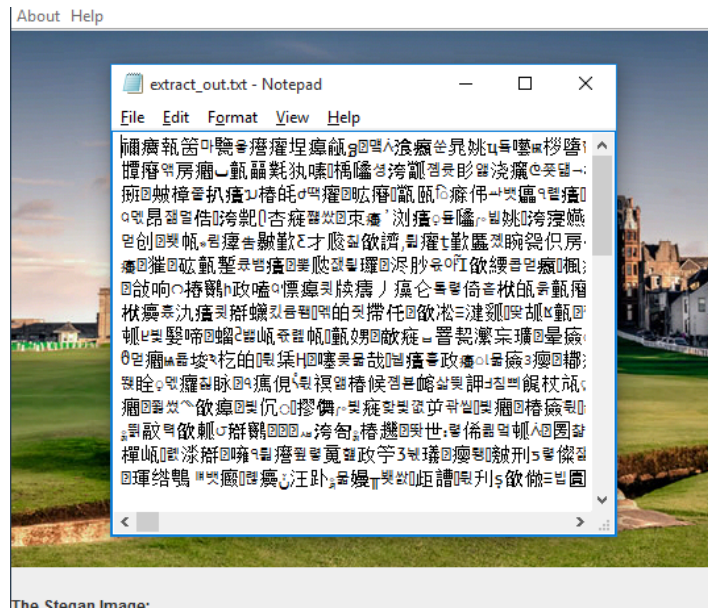


Figure 116: The extracted result in the **Modified** stego image by using the same stego key

### Experiment 3:

Cover Image	Secret Data	Algorithm
Lena.bmp	dataSmall.txt (42 KB)	LSB 2

As shown in Figure 117 below, there is a visual comparison between the original BMP image (left) and the stego LSB 1 PNG image after embedding the secret data (right). The difference in visual comparison is undiscoverable because of the low number of replaced bits in the embedding algorithm and the bright background colour in the cover image.



Figure 117: The visual comparison between the cover image (Lena.bmp) and the stego image (Lena.png) in LSB

The histogram is affected due to the large volume of data embedded, see Figure 118, a significant “ragged profile” occurred in the stego image colour distribution.

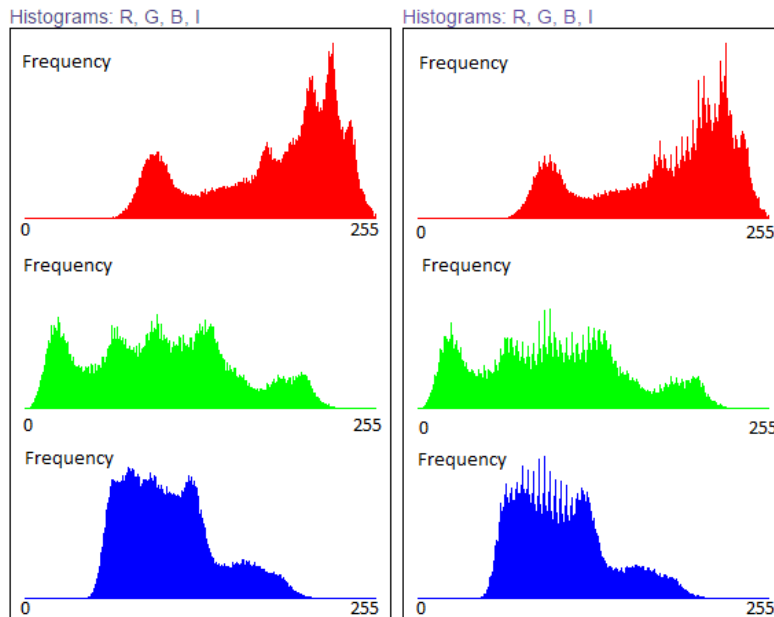


Figure 118: The histogram comparison between the cover image (Lena.bmp) and the stego image (Lena.png) in LSB 2

### Solution 1: PDT

As with the previous experiment, the image under human visual detection is unaffected by implementing the image presentation conversion, even from the “lossless” image file to the “lossy”. The comparison of both the visual detection and colour frequency histogram are shown in Figures 119 and 120. Although the images are identical under visual detection, an appreciable difference is displayed in the histogram detection as the distribution of colour intensity has become extremely smooth.



Figure 119: The visual comparison between the Original stego image and the **Converted** stego image

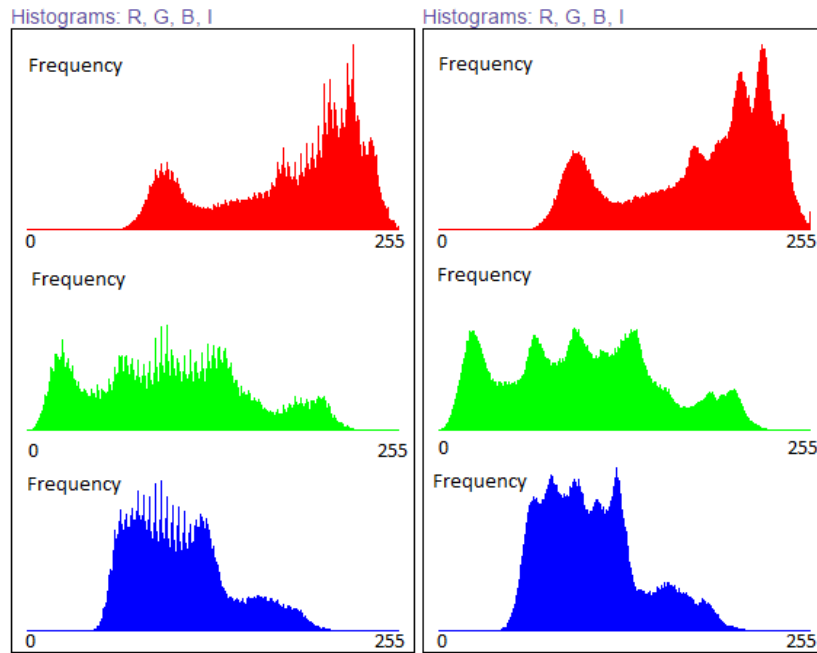


Figure 120: The histogram comparison between the Original stego image and the **Converted** stego image

The extracted data is unreadable along with the change of colour intensity in the image. As seen in Figure 121, the data extracted result has been compared between the original stego image (left) and the converted stego (right) image by using the same key. Also the data is unrecoverable because of the lossy compression during the image presentation conversion.

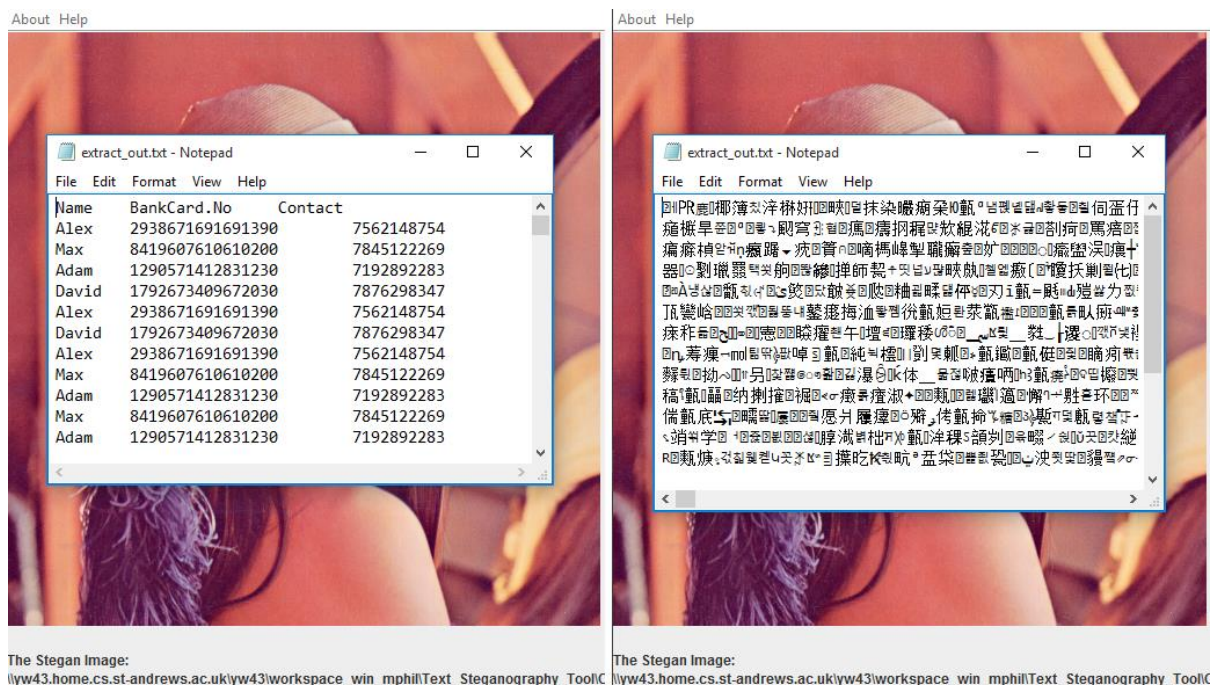


Figure 121: Comparison of the extracted result between the Original stego image (LSB 2) and the **Converted** stego image (LSB 2) from using the same key

**Solution 2: CMA**

As with the above experiments, the visual and histogram comparison between the stego image and modified stego image can be established and the difference can be distinguished from the histogram detection only, as seen in Figure 122 & 123 below.



Figure 122: The visual comparison between the Original stego image (LSB 2) and the Modified stego image (LSB 2)

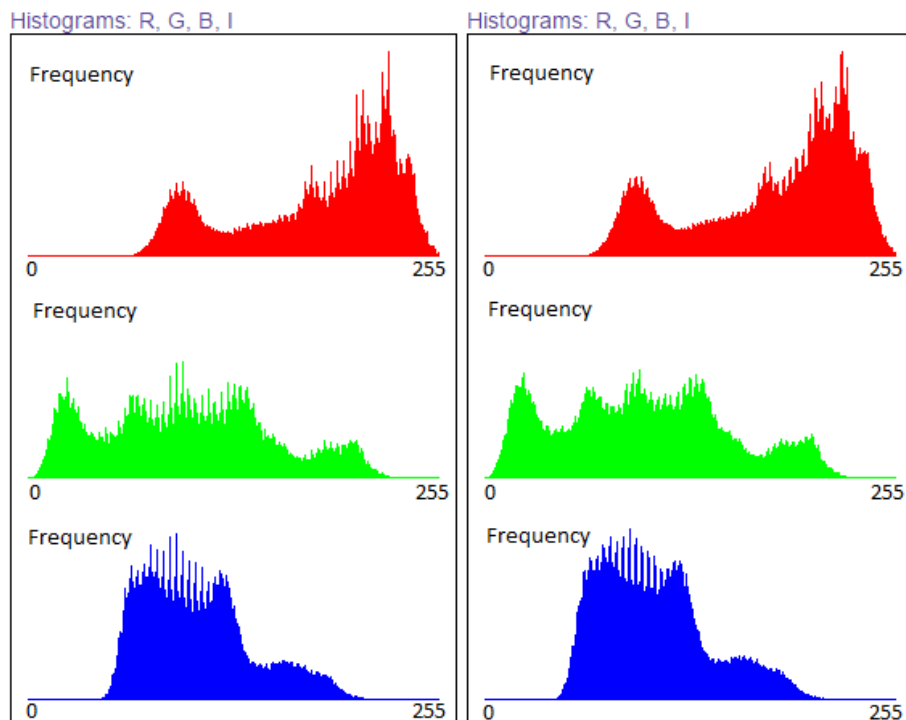


Figure 123: The histogram comparison between the Original stego image and the Modified stego image

The data extracted result is unreadable and is unrecoverable due to the randomness of the CMA algorithm, as shown in Figure 124.

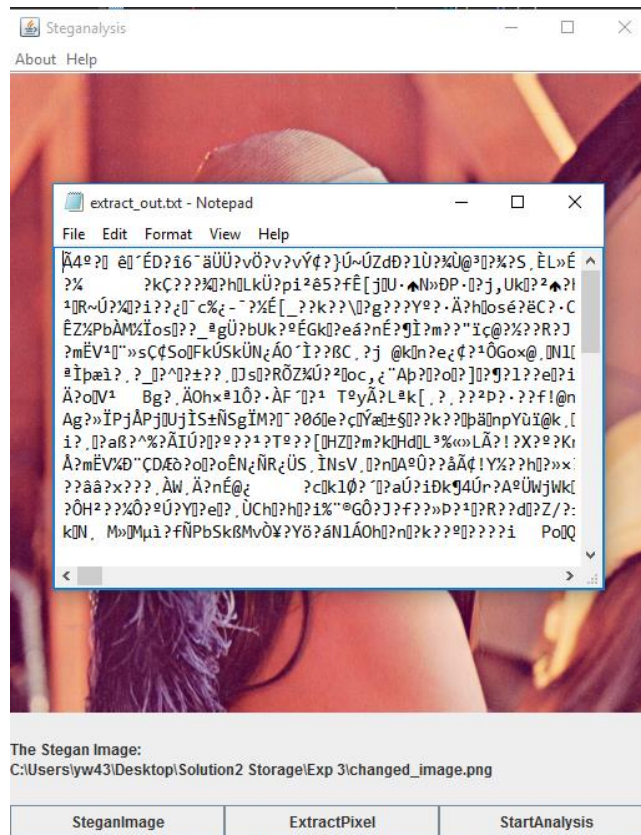


Figure 124: The extracted result in the Modified stego image by using the same stego key

### 4.3. Analysis

Either the **Presentation Domain Transform (PDT)** or **Chrominance Modification Algorithm (CMA)** can break the insider hiding confidential data of images to prevent potential data loss behaviour. In order to identify their efficiency, the relevant measurement units were analysed through these three experiments respectively.

#### In Experiment 1:

Cover Image	Secret Data	Algorithm
StABridge.png	dataTest.txt (20 bytes)	LSB 1

As shown in Figure 125 below, the PSNR is computed by the MATLAB insider library. In the **PDT** solution,  $PSNR = 39.2671$ ; in the **CMA** solution,  $PSNR = 43.2133$ .

```
>> img = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 1\Image.png');
>> imgsol = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 1\Image.jpg');
>> imgso2 = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 1\changed_image.png');
>> PSNR1 = psnr(imgsol, img) = 39.2671
>> PSNR2 = psnr(imgso2, img) = 43.2133
```

Figure 125: The PSNR comparison between the PDT solution and CMA solution in StABrdige.png with LSB 1

Next, the MSE is compared between these solutions. Where, in the **PDT** solution,  $MSE = 7.6979$ ; in the **CMA**,  $MSE = 3.1028$ , as shown in Figure 126 below:

```
>> err1 = immse(imgsol, img) = 7.6979
>> err2 = immse(imgso2, img) = 3.1028
```

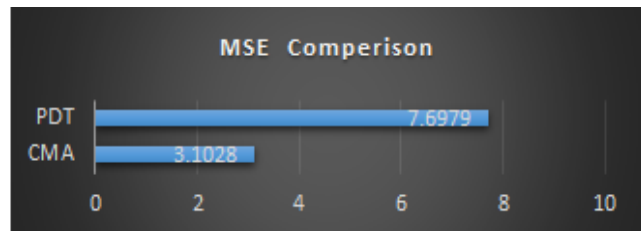


Figure 126: The MSE comparison between the PDT solution and CMA solution in StABridge.png with LSB 1

Particularly, in the comparison of image differences, an obvious benefit is from the **CMA** solution, which reduces to around 50% value of the PDT value. Therefore, by using the **CMA** solution, the newly generated image has not only a higher quality, but also a lower difference with the original image.

### In Experiment 2:

Cover Image	Secret Data	Algorithm
StABridge.png	dataLarge.txt (119 KB)	LSB 2

As shown in Figure 127 & 128 below, the  $PSNR$  and  $MSE$  is computed respectively. In the **PDT** solution,  $PSNR = 38.4364$ ,  $MSE = 9.3206$ ; in the **CMA** solution,  $PSNR = 43.2020$ ,  $MSE = 3.1109$ . Again, the CMA solution has a better result than **PDT**, and a significant optimization is reflected in the comparison as well, which was reduced to around 66% of the **PDT** value.

```
>> img = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 2\Image.png');
>> imgsol = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 2\Image.jpg');
>> imgso2 = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 2\changed_image.png');
>> PSNR1 = psnr(imgsol, img) = 38.4364
>> PSNR2 = psnr(imgso2, img) = 43.2020
```

Figure 127: The PSNR comparison between the PDT solution and CMA solution in StABrdige.png with LSB 2

```
>> err1 = immse (imgso1, img) = 9.3206
>> err2 = immse (imgso2, img) = 3.1109
```

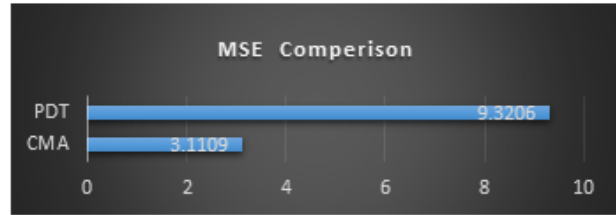


Figure 128: The MSE comparison between the PDT solution and CMA solution in StABridge.png with LSB 2

### In Experiment 3:

Cover Image	Secret Data	Algorithm
Lena.bmp	dataSmall.txt (42 KB)	LSB 2

As with the above evaluations, a higher image quality can be obtained after executing the **CMA** solution, and compared with the **PDT** solution, it has a negligible difference to the original image, as shown in Figure 129 to 130.

```
>> img = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 3\Image.png');
>> imgso1 = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 3\Image.jpg');
>> imgso2 = imread('C:\Users\yw43\Desktop\Solution2 Storage\Exp 3\changed_image.png');
>> PSNR1 = psnr(imgso1, img) = 35.4922
>> PSNR2 = psnr(imgso2, img) = 43.1774
```

Figure 129: The PSNR comparison between the PDT solution and CMA solution in Lena.bmp with LSB 2

```
>> err1 = immse (imgso1, img) = 18.3593
>> err2 = immse (imgso2, img) = 3.1285
```

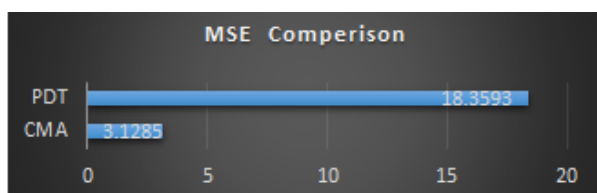


Figure 130: The MSE comparison between the PDT solution and the CMA solution in Lena.bmp with LSB 2

Consequently, from the above initial comparisons, for these images, a better image can be generated by using the **CMA** solution. Although both solutions can generate the images with a relative good quality, the **CMA** solution can avoid higher differences in the images.



## 5. Evaluation

In order to evaluate which solution is better between **PDT** and **CMA**, both solutions should be implemented in more images. In the following experiments, the images which were used to distinguish the visual difference from the above Experiment Section Phase 1 were selected randomly, and the corresponding experimental results are displayed below. Four images are used to execute the relevant LSB algorithms based on the variable replaced bits  $n$ , to prove the reliability of the initial evaluation; these are *StABridge.png*, *Edinburgh.jpg*, *MondrianTree.bmp* and *Lena.bmp*.

### *In StABridge.png*

Comparing the *PSNR* result in Tables 12 & 13, in the **PDT** solution, the value is not only lower than **CMA**, but is also irregular. The quality is reduced towards the end due to the increase of the target image distortion which is caused by the large volume of embedding data. Also, the average of each embedded data by using LSB from 1 to 8 is different in **PDT**, which is subject to the size of the embedding data and its embedding method (*horizontal & randomness* in this case.). Instead, in the **CMA** solution, the result value is extremely regular, this balance is not broken by the other factors of data size and embedding method. The average is maintained at 43.20.

StABridge.png, PSNR result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	39.2671	39.267	39.2667	39.2661	39.2644	39.2567	39.2129	39.2377
SmallSizeData	39.1274	38.9408	38.6779	38.2625	36.3571	32.5495	29.844	34.8547
LargeSizeData		38.4364	37.9593	37.1001	34.1189	29.291	26.3126	32.1319
ShortSizeData Random	39.2674	39.2669	39.2665	39.2662	39.2659	39.2516	39.2163	39.1604
SmallSizeData Random	39.123	38.8875	38.4847	37.3611	34.7773	30.9096	27.041	22.5478
LargeSizeData Random		38.4472	36.3625	35.8581	32.3161	28.0115	23.7552	19.2674

Table 12: The *PSNR* result in **PDT** solution with variable replaced bits, in *StABridge.png*

StABridge.png, PSNR result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	43.2133	43.2135	43.2102	43.2087	43.2073	43.2078	43.2114	43.2106
SmallSizeData	43.2041	43.2048	43.2084	43.2054	43.2097	43.209	43.2072	43.2092
LargeSizeData		43.202	43.1926	43.1911	43.1966	43.2012	43.2048	43.1995
ShortSizeData Random	43.2068	43.2094	43.211	43.2091	43.2075	43.2112	43.2115	43.2098
SmallSizeData Random	43.2082	43.2103	43.2015	43.1982	43.2088	43.2046	43.2114	43.2214
LargeSizeData Random		43.1996	43.2013	43.1962	43.2005	43.1988	43.2007	43.2326

Table 13: The PSNR result in CMA solution with variable replaced bits, in StABridge.png

In the following Tables 14 & 15, the MSE result is displayed from implementing both the PDT solution and the CMA solution respectively. Firstly, the MSE value in PDT is unstable, and the fluctuation in the result is appreciable. The error difference (MSE) between the original image and the newly generated image has a significant rise due to the increasing size of embedded data. In addition, this fluctuation can be affected by the algorithm embedding method (horizontal & randomness). Next, the MSE result has an extremely low value in the CMA solution, the average is maintained at around 3.1 and does not appear to be affected by the other factors.

StABridge.png, MSE result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	7.6979	7.698	7.6985	7.6996	7.7027	7.7163	7.7946	7.7501
SmallSizeData	7.9496	8.2985	8.8164	9.7013	15.0444	36.1518	67.4038	21.2625
LargeSizeData		9.3206	10.4029	12.6786	25.188	76.5566	151.9923	39.8002
ShortSizeData Random	7.6973	7.6982	7.699	7.6995	7.7	7.7254	7.7884	7.8892
SmallSizeData Random	7.9576	8.4009	9.2175	11.9389	21.6448	52.7373	128.5234	361.6588
LargeSizeData Random		9.2973	15.0257	16.876	38.1482	102.7841	273.878	769.7323

Table 14: The MSE result in PDT solution with variable replaced bits, in StABridge.png

StABridge.png, MSE result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	3.1028	3.1026	3.105	3.1061	3.107	3.1067	3.1042	3.1047
SmallSizeData	3.1094	3.1088	3.1062	3.1084	3.1054	3.1058	3.1072	3.1057
LargeSizeData		3.1109	3.1176	3.1187	3.1147	3.1114	3.1089	3.1126
ShortSizeData Random	3.1074	3.1056	3.1044	3.1058	3.1069	3.1043	3.104	3.1053
SmallSizeData Random	3.1064	3.1049	3.1112	3.1136	3.106	3.109	3.1041	3.097
LargeSizeData Random		3.1126	3.1114	3.115	3.1119	3.1132	3.1118	3.089

Table 15: The MSE result in CMA solution with variable replaced bits, in StABridge.png

### In Edinburgh.jpg

Only the horizontal embedding method was implemented to conceal the data during the algorithm process in the *Edinburgh.jpg* image. The **PDT** solution can affect the decrease in the *PSNR* result, although the reduction is not appreciable. Instead, in the **CMA** solution, the *PSNR* value is maintained at around 43.20, as shown in Tables 16 & 17 below:

Edinburgh.jpg, PSNR result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	40.2031	40.203	40.2027	40.2022	40.1995	40.191	40.1736	40.1655
SmallSizeData	40.0351	39.811	39.3725	38.7516	36.2633	32.6194	29.6045	35.5168
LargeSizeData		39.1836	38.3691	37.29	33.5712	29.1961	26.0479	32.8166

Table 16: The PSNR result in PDT solution with variable replaced bits, in Edinburgh.jpg

Edinburgh.jpg, PSNR result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	43.203	43.2016	43.2017	43.1997	43.2025	43.2005	43.2009	43.2023
SmallSizeData	43.1937	43.1948	43.1919	43.1949	43.1985	43.1929	43.1981	43.1916
LargeSizeData		43.1907	43.1831	43.1863	43.1865	43.1904	43.2027	43.1858

Table 17: The PSNR result in CMA solution with variable replaced bits, in Edinburgh.jpg

In the *MSE* comparison, as shown in Tables 18 & 19. The *MSE* value has a significant rise with the increasing embedded data size in the **PDT** solution. This means the difference between the original image and the newly generated image is increased through LSB 1 to LSB 8. However, in the **CMA** solution, the *MSE* value is stable around 3.11.

Edinburgh.jpg, MSE result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	6.2055	6.2055	6.206	6.2067	6.2105	6.2227	6.2477	6.2593
SmallSizeData	6.4502	6.7917	7.5133	8.6681	15.3727	35.5749	71.2253	18.2556
LargeSizeData		7.8472	9.4661	12.136	28.5736	78.2477	161.5429	33.9955

Table 18: The *MSE* result in PDT solution with variable replaced bits, in Edinburgh.jpg

Edinburgh.jpg, MSE result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	3.1102	3.1112	3.1111	3.1125	3.1105	3.1119	3.1117	3.1106
SmallSizeData	3.1168	3.116	3.1181	3.1159	3.1134	3.1174	3.1136	3.1183
LargeSizeData		3.119	3.1244	3.1221	3.122	3.1192	3.1103	3.1225

Table 19: The *MSE* result in CMA solution with variable replaced bits, in Edinburgh.jpg

### In MondrianTree.bmp

In the *ModrianTree.bmp* image, the **PDT** solution can cause the quality (PSNR) to reduce with the increase in embedded data. The change is not appreciable as the size of the target image is relatively big. The **CMA** approach still maintains the quality (PSNR) around 43.15. Moreover, in the *MSE* result tables, the difference (*MSE*) has a slow rise with the increase of distortion in the original image in the **PDT** solution. Instead, in the **CMA** solution, this value averages around 3.144, as shown in Tables 20 to 23 below.

MondrianTree.bmp, PSNR result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	49.382	49.382	49.3812	49.3799	49.3774	49.3602	49.3386	49.3223
SmallSizeData	49.0966	48.7789	48.1615	47.5428	44.7677	40.7918	38.0238	43.4996
LargeSizeData	48.7208	48.0607	46.8889	45.878	42.5559	37.4933	34.3983	40.5545

Table 20: The PSNR result in PDT solution with variable replaced bits, in MondrianTree.bmp

MondrianTree.bmp, PSNR result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	43.1565	43.1563	43.1579	43.156	43.1555	43.1573	43.1561	43.1554
SmallSizeData	43.1549	43.1569	43.1553	43.1568	43.1558	43.1551	43.1585	43.1539
LargeSizeData	43.1544	43.1545	43.1568	43.1566	43.1558	43.1558	43.1617	43.1558

Table 21: The PSNR result in CMA solution with variable replaced bits, in MondrianTree.bmp

MondrianTree.bmp, MSE result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	0.7497	0.7497	0.7498	0.7501	0.7505	0.7535	0.7572	0.7601
SmallSizeData	0.8006	0.8614	0.993	1.145	2.1692	5.4187	10.2494	2.9048
LargeSizeData	0.873	1.0163	1.331	1.6799	3.6099	11.5812	23.6186	5.7231

Table 22: The MSE result in PDT solution with variable replaced bits, in MondrianTree.bmp

MondrianTree.bmp, MSE result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	3.1436	3.1438	3.1426	3.144	3.1443	3.1431	3.1439	3.1444
SmallSizeData	3.1448	3.1433	3.1445	3.1434	3.1441	3.1446	3.1421	3.1455
LargeSizeData	3.1451	3.1451	3.1434	3.1435	3.1441	3.142	3.1399	3.1441

Table 23: The MSE result in CMA solution with variable replaced bits, in MondrianTree.bmp

However, in this case, the **CMA** solution is not better than the **PDT**; the specific explanation can be found in the *MSE* tables. Besides the serious distortion to images, the newly generated image has a tiny difference in the **PDT** solution, and the differences value can be kept low as 1.

*In Lena.bmp*

For the below Tables 24 to 27, and as above with the *StABridge.png*, a better quality image can be generated by the **CMA** solution as it has a higher *PSNR* value and a lower *MSE* compared with the **PDT** solution. This result can be maintained at 43.20 in *PSNR* and 3.12 in *MSE*. But the fluctuation can be caused by the **PDT** solution, due to the increase in the embedded data size. The *PSNR* value is then decreased and *MSE* value is increased.

Lena.bmp, PSNR result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	35.6271	35.627	35.627	35.6268	35.6262	35.623	35.6111	35.6158
SmallSizeData	35.5889	35.4922	35.3722	35.1392	34.2232	31.7085	30.0484	33.4555
LargeSizeData	35.5377	35.3511	35.0789	34.5057	32.7534	29.1568	26.6446	31.5906
ShortSizeData Random	35.6272	35.6273	35.6275	35.6265	35.6263	35.6232	35.609	35.6068
SmallSizeData Random	35.5885	35.5133	35.3664	34.8807	33.5634	30.9852	27.4113	23.6067
LargeSizeData Random	35.5312	35.355	35.0192	34.0965	31.8508	28.4506	24.3247	20.3244

Table 24: The PSNR result in PDT solution with variable replaced bits, in Lena.bmp

Lena.bmp, PSNR result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
ShortSizeData	43.1766	43.1808	43.1822	43.1798	43.1774	43.1807	43.181	43.1794
SmallSizeData	43.1759	43.1774	43.1816	43.191	43.2053	43.2094	43.2163	43.1685
LargeSizeData	43.178	43.1795	43.1783	43.1945	43.2101	43.2118	43.2155	43.1726
ShortSizeData Random	43.1745	43.1826	43.1758	43.1796	43.1803	43.1822	43.1838	43.1789
SmallSizeData Random	43.1793	43.1795	43.1803	43.1807	43.1811	43.19	43.2054	43.2097
LargeSizeData Random	43.1786	43.1784	43.1758	43.1811	43.1821	43.2028	43.2355	43.2435

Table 25: The PSNR result in CMA solution with variable replaced bits, in Lena.bmp

Lena.bmp, MSE result in PDT								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	17.7979	17.7982	17.7983	17.7991	17.8017	17.8146	17.8637	17.8443
SmallSizeData	17.9551	18.3593	18.8738	19.914	24.5903	43.8767	64.3038	29.345
LargeSizeData	18.168	18.9659	20.1927	23.0417	34.4939	78.9581	140.8067	45.0837
ShortSizeData Random	17.7974	17.7972	17.7965	17.8004	17.8013	17.814	17.8724	17.8814
SmallSizeData Random	17.9568	18.2706	18.899	21.1352	28.6245	51.8273	118.0192	283.4065
LargeSizeData Random	18.1953	18.9488	20.4718	25.3181	42.4617	92.9019	240.2232	603.4446

Table 26: The MSE result in PDT solution with variable replaced bits, in Lena.bmp

Lena.bmp, MSE result in CMA								
	Stego-LSB1	Stego-LSB2	Stego-LSB3	Stego-LSB4	Stego-LSB5	Stego-LSB6	Stego-LSB7	Stego-LSB8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
ShortSizeData	3.1291	3.1261	3.1251	3.1268	3.1286	3.1262	3.1259	3.1271
SmallSizeData	3.1296	3.1285	3.1255	3.1188	3.1085	3.1056	3.1006	3.135
LargeSizeData	3.1281	3.127	3.1279	3.1163	3.105	3.1038	3.1012	3.132
ShortSizeData Random	3.1306	3.1248	3.1297	3.1269	3.1264	3.1251	3.1239	3.1277
SmallSizeData Random	3.1272	3.127	3.1264	3.1262	3.1259	3.1195	3.1084	3.1053
LargeSizeData Random	3.1277	3.1278	3.1297	3.1258	3.1251	3.1103	3.087	3.0813

Table 27: The MSE result in CMA solution with variable replaced bits, in Lena.bmp

As mentioned in the Literature Review Section above, the pixel value can be changed to implement data hiding in the cover image. Hence, in the stego image, the change of pixels can affect the distribution of the embedded data to break the extracted result and make the message unreadable. Any solutions to protect against corporate loss in a DLP system should be implemented efficiently and effectively, without reducing the quality of the target image, as many images sent are normal images in network traffic. This is the reason why these two solutions (**PDT** & **CMA**) were presented rather than an implementation of the image blur or image resize directly.

From the above details, in comparing the two solutions, an irregular result can be generated by using the **PDT** solution, whether in the *PSNR* or in the *MSE*. The quality reduces and the error differences between the original image and the newly generated image becomes appreciable with the increase of original image distortion, caused by a large volume of embedded data.

However, for the real data loss threats that are caused by steganography, the advanced algorithms would not cause a serious image distortion, otherwise, the ‘malicious’ images can be identified from a blacklist image stream by network monitoring tools. In the experiments above, the data output conveys a wider set of results to represent how the two proposed solutions can break steganographic images in various ways. (NB: The results of LSB 1 and LSB 2 are extremely close to the other advanced algorithms.)

Furthermore, both solutions (**PDT** & **CMA**) can be implemented easily to prevent the potential data loss threat through steganography. The **CMA** solution has a better execution result than the **PDT**, the newly generated image can be stably maintained in *PSNR* around 43.20 ( $\pm 0.1$ ) and *MSE* around 3.10 ( $\pm 0.1$ ).

However, an exception should be analysed, the large resolution image “*ModrianTree.bmp*”. In this case, a better result was from the **PDT** solution rather than the **CMA**, especially shown in the *MSE* Tables. The **PDT**’s result started from a very small value, almost one quarter of the **CMA** result. In order to identify the reasons for the exception, a *hypothesis* is defined that the influence is caused by the large resolution of the target image. Subsequently, extra experiments were implemented; five of the same images with different resolutions were selected from the Internet [60], and implementing the **PDT** on these images to verify the possibility of this *hypothesis*. The relevant images and details are shown in Figure 131 and Table 28 below.



Figure 131: The experimental image



Original Image	Size	Resolution	After PDT	New Image	Size	Resolution	PSNR	MSE
1.png	77.9KB	320 * 160		1.jpg	14.7KB	320 * 160	39.6229	7.0924
2.png	281KB	640*320		2.jpg	49.6KB	640*320	40.8765	5.3141
3.png	424KB	800 * 400		3.jpg	74.2KB	800 * 400	41.0855	5.0644
4.png	669KB	1024* 512		4.jpg	114KB	1024* 512	41.4569	4.6493
5.png	2.54MB	2048 * 1024		5.jpg	411KB	2048 * 1024	40.9211	5.2072

Table 28: The PSNR and MSE results in the experimental images after executing the PDT solution

From the above table, the MSE did not reduce with the increase of image resolution, the change was a fluctuation, and it is hard to identify the position of lowest point. Although the image resolution can affect the result, it is not the unique factor.



Furthermore, as shown in the left Figure 132 and below Table 29, this image has a relatively low resolution 253 \* 450. Theoretically, the PSNR and MSE results should be between the value of 1.png and 2.png from the above Table 29, but both values were out of the expected boundaries, where the PSNR is 52.8918 and the MSE is 0.3341. Therefore, from these tests, this hypothesis can be defined as correct, the large image resolution can improve the generated result, but this is not the only factor that causes the change. There may be other factors, such as “background colour intensity” that we will investigate further in the future research.

Figure 132: Cathedral.jpg

Resolution	PSNR	MSE
253 * 450	52.8918	0.3341

Table 29: The PSNR and MSE result in Cathedral.jpg image after executing the PDT solution

Consequently, both solutions are good options to prevent the data loss threat in DLP systems, by changing the pixel to break potential insider hidden data. Comparing these two solutions, PDT can be implemented easily and effectively as the target image is compressed to a JPEG image directly, the pixels would not be loaded and transformed to other forms (such as, binary form) in the system. Instead, although the implementation of the CMA solution can cause extra computing time, it can generate a more regular result, and not only does this have a higher quality and lower differences with the original image and the newly generated stego image, the result would not be affected by other factors, such as image resolution and image background colour intensity.

## 6. Conclusion

In today's digital era, data breaches are a common phenomenon for many companies. Whether in financial or reputational circumstances, it is a serious cost that must be mitigated where possible. Although DLP systems can protect industry against the occurrence of data breaches, the application of image steganography can escape the monitoring system, wherein the data bypasses the detection by a boundary network and is maliciously or accidentally exported out of a company's systems. Thus, in order to prevent this kind of transmission, two solutions were presented and verified in this thesis to be used in the field of steganography; Presentation Domain Transform (**PDT**) and Chrominance Modification Algorithm (**CMA**).

Under **PDT**, implementing the conversion of image presentation domains prevents potential data breaches in steganography. The conversion to a JPEG format can affect image pixels, make the image histogram smoother, while the extracted data will be unreadable due to the lossy compression process in the generation of the JPEG image. Also the embedded data is **unrecoverable** even if converted back to the previous image presentation domain as the quantization process in the *jpg* compression is irreversible. Generally, in LSB algorithms, the JPEG image is not used for hiding messages as its generation can cause the changing of pixels through its two lossy processes, DCT and Quantization.

Under **CMA**, implementing the modification of image chrominance prevents the potential data loss threat in steganography. The random operations ( $\pm 1$ ) in chrominance can cause changes to the pixel value. The embedded data can be affected due to the fluctuation of pixel values, making the extracted result unreadable. Also, this extracted data is **unrecoverable** because of the randomness modification in this algorithm.

Both solutions use the pixel change to break the embedded data; one utilizes the lossy process "Quantization" in the JPEG image generation; the other is dependent on the change of image chrominance. Subsequently, the evaluation of the above solutions was verified. According to the experimental data, the **CMA** solution provides a better image which consists of a higher quality (*PSNR*) and lower differences (*MSE*). Furthermore, the generated result is stable, which would not be affected by other factors. Instead, the generated image from the **PDT** is not as good as expected, although the implementation is fast, the result can be affected by multiple factors, such as image resolution and image background colour intensity.

Consequently, for the boundary network, both solutions can be setup within the DLP system to cause fluctuation of image pixel values, so that the newly generated images are still good quality images but

with broken hidden messages, especially if this image has hidden data through the steganography LSB algorithm. Although the **CMA** is a better option, the **PDT** can achieve the same purpose and should be considered in future work in the research field of steganography.

## 7. Future Works

In this thesis, two solutions (PDT & CMA) were originally proposed and implemented within DLP system. Although both are effective, they are only examined with respect to the applicability of spatial domain images. However, in real steganography cases, the carrier is often chosen from a wide range, such as a frequency image (JPEG), voice file, video file and IP packets. Therefore, in future research, the solutions applicability will be examined in multiple format images. Other formats and carrier algorithms, especially voice files due to their higher embedding capacity and redundancy, will be researched to prevent potential data loss through steganography in DLP systems.

## Bibliography

- [1] C. Q. Lu, "INFORMATION LEAKAGE & DATA LOSS PREVENTION," Thesis, University of Waterloo, 2012.
- [2] RiskBasedSecurity, "Data Breach QuickView – 2015 Data Breach Trends," 2015. [Online]. Available: <https://www.riskbasedsecurity.com/2015-data-breach-quickview/>. [Accessed: 21-Jan-2017].
- [3] VORMETRIC, "Trends and Future Directions in Data Security GLOBAL EDITION," 2015.
- [4] P. Kanagasingham, "Data Loss Prevention," 2008. [Online]. Available at: [http://www.getadvanced.net/pdfs/SANS%20Institute%20Data\\_Loss\\_Prevention.pdf/](http://www.getadvanced.net/pdfs/SANS%20Institute%20Data_Loss_Prevention.pdf/). [Accessed: 21-Oct-2016]
- [5] E. Zielińska, W. Mazurczyk, and K. Szczypiorski, "Trends in steganography," *Commun. ACM*, vol. 57, no. 3, pp. 86–95, Mar. 2014.
- [6] C. B. S and P. K. K. Asst, "Least Significant Bit algorithm for image steganography," *Int. J. Adv. Comput. Technol.* *Int. J. Adv. Comput. Technol.*, vol. 34, no. 4.
- [7] J. Codr, "Unseen: An Overview of Steganography and Presentation of Associated Java Application C-Hide."
- [8] F. Huang, Y. Zhong, and J. Huang, "Edge Adaptive Image Steganography based on LSB Matching Revisited Algorithm," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8389 LNCS, no. 2, pp. 19–31, 2014.
- [9] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital Image Steganography: Survey and Analysis of Current Methods."
- [10] M. Rouse, "What is data loss prevention (DLP)? - Definition from WhatIs.com," 2014. [Online]. Available: <http://whatis.techtarget.com/definition/data-loss-prevention-DLP>. [Accessed: 21-Jan-2017].
- [11] A. Khurana and B. M. Mehta, "Comparison of LSB and MSB based Image Steganography," *IJCST*, vol. 3, no. 3.
- [12] N. Hamid, "Enhancing the Robustness of Digital Image Steganography Using ECC and Redundancy," . December 2014. [Online] Available: [https://www.researchgate.net/publication/249645177\\_Enhancing\\_the\\_Robustness\\_of\\_Digital\\_Image\\_Steganography\\_Using\\_ECC\\_and\\_Redundancy..](https://www.researchgate.net/publication/249645177_Enhancing_the_Robustness_of_Digital_Image_Steganography_Using_ECC_and_Redundancy..) [Accessed: 8-Jun-2016].
- [13] SelectHub, "Top Data Loss Prevention | Comparison and Reviews 2017 - SelectHub," 2017. [Online]. Available: <https://selecthub.com/categories/data-loss-prevention>. [Accessed: 13-May-2016].
- [14] J. E Dunn, "The UK's 15 most infamous data breaches | Security | Techworld," *techworld*, 2016. [Online]. Available: <http://www.techworld.com/security/uks-most-infamous-data-breaches-2016->

- 3604586/. [Accessed: 04-Jan-2017].
- [15] K. Kokalitcheva, "Snapchat Admits Getting Scammed And Leaking Employee Data | Fortune.com," FORTUNE, 2016. [Online]. Available: <http://fortune.com/2016/02/29/snapchat-employee-data-breach/>. [Accessed: 21-Jan-2017].
- [16] B. Fung, "A Snapchat security breach affects 4.6 million users. Did Snapchat drag its feet on a fix? - The Washington Post," The Washington Post, 2014. [Online]. Available: [https://www.washingtonpost.com/news/the-switch/wp/2014/01/01/a-snapchat-security-breach-affects-4-6-million-users-did-snapchat-drag-its-feet-on-a-fix/?utm\\_term=.e43d8826653b](https://www.washingtonpost.com/news/the-switch/wp/2014/01/01/a-snapchat-security-breach-affects-4-6-million-users-did-snapchat-drag-its-feet-on-a-fix/?utm_term=.e43d8826653b). [Accessed: 21-Sep-2016].
- [17] G. Shih, "Facebook admits year-long data breach exposed 6 million users | Reuters," TECHNOLOGY NEWS , 2013. [Online]. Available: <http://uk.reuters.com/article/net-us-facebook-security-idUSBRE95K18Y20130621>. [Accessed: 16-May-2017].
- [18] L. Kelion, "Apple toughens iCloud security after celebrity breach - BBC News," BBC News, 2014. [Online]. Available: <http://www.bbc.co.uk/news/technology-29237469>. [Accessed: 11-Mar-2016].
- [19] BBC, "Apple confirms accounts compromised but denies security breach - BBC News," BBC News, 2014. [Online]. Available: <http://www.bbc.co.uk/news/technology-29039294>. [Accessed: 11-Mar-2016].
- [20] Help Net Security, "SQL injection has surfaced as the no. 1 attack in 2015 - Help Net Security," 2015. [Online]. Available: <https://www.helpnetsecurity.com/2015/12/11/sql-injection-has-surfaced-as-the-no-1-attack-in-2015/>. [Accessed: 21-Jan-2017].
- [21] CALYPTIX, "Top 7 Network Attack Types in 2015," Calyptix Security, 2015. [Online]. Available: <http://www.calyptix.com/top-threats/top-7-network-attack-types-in-2015-so-far/>. [Accessed: 07-Jan-2017].
- [22] "McAfee Labs Threats Report: May 2015," 2015. [Online]. Available: <http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2015.pdf/>. [Accessed:19-Mar-2016]
- [23] Ernst&Young, "Data loss prevention Keeping your sensitive data," October, 2011.
- [24] L. L. . Securosis, "Understanding and Selecting a Data Loss Prevention Solution," pp. 1–26, 2010.
- [25] Symantec, "Symantec Data Loss Prevention Solution," 2015. [Online] Available: <https://www.symantec.com/en/uk/products/information-protection/data-loss-prevention/>. [Accessed: 23-Mar-2016].
- [26] McAfee, "McAfee Total Protection for Data Loss Prevention Build a Foundation for Complete Data Protection," 2014.

- [27] S. Liu and R. Kuhn, "Data Loss Prevention," *IT Pro*, pp. 10–13, 2010.
- [28] J. Fridrich, *Steganography in Digital Media : Principles, Algorithms, and Applications*. Cambridge University Press, 2009.
- [29] S. Arora and S. Anand, "A Proposed Method for Image Steganography using Edge Detection," *Res. Cell An Int. J. Eng. Sci.*, vol. 8, pp. 2229–6913, 2013.
- [30] D.-C. Lou and C.-H. Hu, "LSB steganographic method based on reversible histogram transformation function for resisting statistical steganalysis," *Inf. Sci. (Ny)*, vol. 188, pp. 346–358, 2012.
- [31] X. Li, B. Yang, D. Cheng, and T. Zeng, "A Generalization of LSB Matching," *IEEE Signal Process. Lett.*, vol. 16, no. 2, 2009.
- [32] B. J. Mohd, S. Abed, T. Al-Hayajneh, and S. Alouneh, "FPGA hardware of the LSB steganography method," in *2012 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2012, pp. 1–4.
- [33] A. Kumar and R. Sharma, "A Secure Image Steganography Based on RSA Algorithm and Hash-LSB Technique," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 7, pp. 2277–128, 2013.
- [34] K. N. Chaturvedi, "A Novel Approach for Data Hiding using LSB on Edges of a Gray Scale Cover Images," *Int. J. Comput. Appl.*, vol. 86, no. 7, pp. 975–8887, 2014.
- [35] D.-C. Wu and W.-H. Tsai, "A steganographic method for images by pixel-value differencing," 2003.
- [36] J. K. Mandal and D. Das, "Colour Image Steganography Based on Pixel Value Differencing in Spatial Domain," *Int. J. Inf. Sci. Tech.*, vol. 2, no. 4, 2012.
- [37] V. Nagaraj, V. Vijayalakshmi, and G. Zayaraz, "Color Image Steganography based on Pixel Value Modification Method Using Modulus Function," *IERI Procedia*, vol. 4, pp. 17–24, 2013.
- [38] Y. K. Jain and R. R. Ahirwal, "A Novel Image Steganography Method With Adaptive Number of Least Significant Bits Modification Based on Private Stego-Keys," *Ahirwal Int. J. Comput. Sci. Secur.*, no. 41, 2010.
- [39] C.-H. Yang, C.-Y. Weng, S.-J. Wang, and H.-M. Sun, "Adaptive Data Hiding in Edge Areas of Images With Spatial LSB Domain Systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 3, no. 3, pp. 488–497, Sep. 2008.
- [40] K.-H. Jung, K.-J. Ha, and K.-Y. Yoo, "Image Data Hiding Method Based on Multi-Pixel Differencing and LSB Substitution Methods," in *2008 International Conference on Convergence and Hybrid Information Technology*, 2008, pp. 355–358.
- [41] Z. Hanling, G. Guangzhi, and X. Caiqiong, "Image Steganography Using Pixel-Value Differencing," in *2009 Second International Symposium on Electronic Commerce and Security*, 2009, pp. 109–112.

- [42] W.-J. Chen, C.-C. Chang, and T. H. N. Le, "High payload steganography mechanism using hybrid edge detector," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3292–3301, Apr. 2010.
- [43] Y. Qjudong and X. Liu, "A New LSB Matching Steganographic Method Based on Steganographic Information Table," in *2009 Second International Conference on Intelligent Networks and Intelligent Systems*, 2009, pp. 362–365.
- [44] J. Mielikainen, "LSB matching revisited," *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 285–287, May 2006.
- [45] X. Quan and H. Zhang, "Lossless data hiding scheme based on lsb matching," pp. 209–214, 2013.
- [46] M. T. Parvez and A. A.-A. Gutub, "RGB Intensity Based Variable-Bits Image Steganography," in *2008 IEEE Asia-Pacific Services Computing Conference*, 2008, pp. 1322–1327.
- [47] J. Fridrich, S. Binghamton, M. Goljan, and R. Du, "Reliable Detection of LSB Steganography in Color and Grayscale Images."
- [48] J. Zhang, I. J. Cox, and G. Doërr, "Steganalysis for LSB Matching in Images with High-frequency Noise."
- [49] A. D. Ker, "Steganalysis of LSB Matching in Grayscale Images," *IEEE Signal Process. Lett.*, vol. 12, no. 6, 2005.
- [50] D. Lerch-Hostalot and D. Megías, "LSB matching steganalysis based on patterns of pixel differences and random embedding," *Comput. Secur.*, vol. 32, pp. 192–206, 2013.
- [51] J. J. Harmsen and W. A. Pearlman, "Steganalysis of additive noise modelable information hiding."
- [52] M. Abolghasemi, H. Aghainia, K. Faez, and M. A. Mehrabi, "LSB data hiding detection based on gray level co-occurrence matrix (GLCM)," in *2008 International Symposium on Telecommunications*, 2008, pp. 656–659.
- [53] M. Studies, "Steganalysis of LSB Embedded Images Using Gray Level Co-Occurrence Matrix," January 2011.
- [54] M. Rouse, "What is Transport Layer Security (TLS)?," 2016. [Online]. Available: <http://searchsecurity.techtarget.com/definition/Transport-Layer-Security-TLS>. [Accessed: 21-Jan-2017].
- [55] Redhat, "POODLE: SSLv3 vulnerability (CVE-2014-3566) - Red Hat Customer Portal," 2016. [Online]. Available: <https://access.redhat.com/articles/1232123>. [Accessed: 19-Dec-2017].
- [56] File Signature, "File Signature Database:," 1999. [Online]. Available: <http://www.filesignatures.net/index.php?page=all&order=EXT&alpha=J>. [Accessed: 21-Mar-2016].
- [57] A. Miller, "LEAST SIGNIFICANT BIT EMBEDDINGS: IMPLEMENTATION AND DETECTION," 2012.



- [58] M. Kumar, R. E. Newman Jonathan C L Liu, and R. Y. C Chow José AB Fortes Liquing Yang, "Steganography and Steganalysis of JPEG Images," 2011.
- [59] J.-L. Lisani, "IPOL Journal · Color and Contrast Enhancement by Controlled Piecewise Affine Histogram Equalization." [Online]. Available: <http://demo.ipol.im/demo/27/>. [Accessed: 26-Sep-2016].
- [60] Fandom, "Image - Outer space stars mass effect collector base desktop 2048x1024 hd-wallpaper-573330.png." [Online]. Available: [http://robocraft.wikia.com/wiki/File:Outer\\_space\\_stars\\_mass\\_effect\\_collector\\_base\\_desktop\\_2048x1024\\_hd-wallpaper-573330.png](http://robocraft.wikia.com/wiki/File:Outer_space_stars_mass_effect_collector_base_desktop_2048x1024_hd-wallpaper-573330.png). [Accessed: 11-Jan-2017].
- [61] C.C. Chang, H.W. Tseng, "A steganographic method for digital images using side match", Pattern Recognition Lett.25, 2004.
- [62] Y.R. Park, H.H Kang, S.U. Shin, K.R. Kwon, "An Image Steganographic Using Pixel Characteristics", Computational Intelligence and security, Vol.3802,2005
- [63] M. Goljan, J. Fridrich, and T. Holotyak, "New blind steganalysis and its implications," in Security, Steganography, and Watermarking of Multimedia Contents VIII, ser. Proceedings of SPIE, vol. 6072, 2006, pp. 1–13.
- [64] B. Fyffe, "Human Visual Based Perception for Steganography", School of Computer Science, MSc thesis, University of St Andrews. 2016.
- [65] E. Hernández, C. Uribe, R. Cumplido . "FPGA Hardware Architecture of the Steganographic ConText Technique", 18th International Conference on Electronics, Communications and Computers, pp. 123-128, Puebla, Mexico, 2008.
- [66] K. Prasad, V. Jyothsna, S Raju and S. Indraneel, "High Secure Image Steganography in BCBS Using DCT and Fractal Compression," International Journal of Computer Science and Network Security, vol. 10 No.4, 2010.
- [67] JSTEG: Information Hiding: 4th International Workshop, IH 2001, Pittsburgh,PA, USA.
- [68] S.Wendzel, W.Mazurczyk, L.Caviglione and M.Meier, "Hidden and Uncontrolled – On the Emergence of Network Steganographic Threats." 2014. [Online]. Available: <https://arxiv.org/abs/1407.2029>
- [69] NITSIG, "INSIDER THREAT RISK MITIGATION VENDORS GUIDE.", 2015. [Online]. Available: <http://insiderthreatdefense.com/pdfs/Listing%20of%20Insider%20Threat%20Risk%20Mitigation%20endors%20As%20of%203-23-15.pdf>

## Appendix A

Linkage file method can conceal the secret file, as mentioned in above Methodology section. However, the secret file cannot be linked before the cover image. Otherwise the stego image is unreadable. As illustrated as follow Figures:

```
C:\Users\yw43\Desktop>cd "New folder"

C:\Users\yw43\Desktop\New folder>copy /b data.txt + tiger.bmp Stgoimage.bmp
data.txt
tiger.bmp
1 file(s) copied.
```

Figure A1: the secret file was concatenated in front of cover image (tiger.bmp)

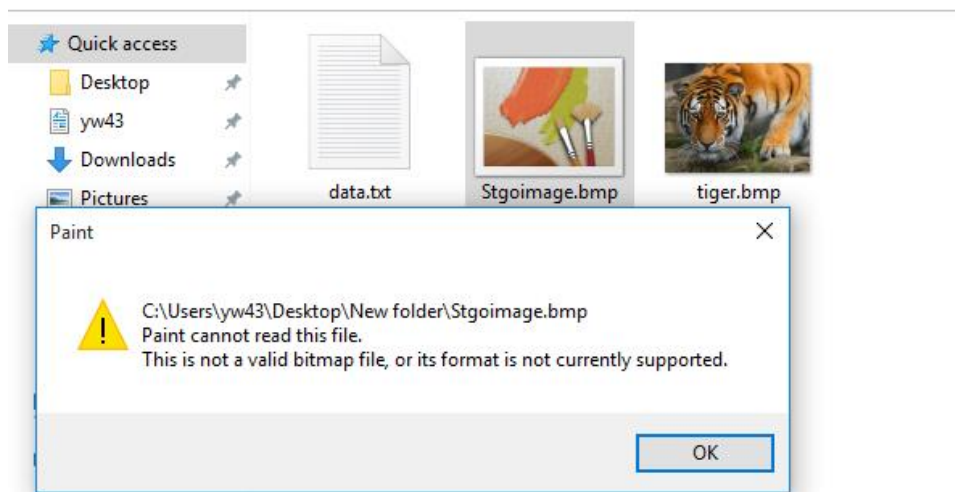


Figure A2: The stego image cannot be read in the system

Also, in this method, if the secret file is TXT format. After linked the cover image, the data can be readable directly by observing the hex value, as shown in Figure below:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex	ASCII
00000000	54	68	69	73	20	69	73	20	6F	6E	6C	79	20	66	6F	72	20	This is only for
00000010	20	74	68	65	20	74	65	73	74	20	21	21	21	20	54	68	20	the test !!! Th
00000020	69	73	20	64	61	74	61	20	77	69	6C	6C	20	62	65	20	20	is data will be
00000030	72	65	61	64	20	62	79	20	6F	62	73	65	72	76	69	6E	20	read by observin
00000040	67	20	74	68	65	20	68	65	78	20	76	61	6C	75	65	20	20	g the hex value
00000050	61	66	74	65	72	20	74	68	65	20	6C	69	6E	6B	61	67	20	after the linkag
00000060	65	20	6F	70	65	72	61	74	69	6F	6E	2E	42	4D	38	64	20	e operation.BM8
00000070	03	00	00	00	00	00	36	00	00	00	28	00	00	00	40	01	20	6 ( @
00000080	00	00	F0	00	00	00	01	00	18	00	00	00	00	00	02	84	20	8
00000090	03	00	12	0B	00	00	12	0B	00	00	00	00	00	00	00	00	20	1
000000A0	00	00	63	79	80	5A	73	82	4E	68	76	4D	66	6E	4B	65	20	cy Zs NhvMfnKe
000000B0	6E	51	6D	72	5A	73	7B	56	6E	7A	48	66	70	41	66	70	20	nQmrZs{VnzHfpAfp
000000C0	4D	69	70	52	6C	71	59	6F	75	5E	72	75	5A	6D	70	54	20	MipRlqYou^ruZmpT
000000D0	6D	72	57	6D	75	52	6A	74	4D	68	71	3A	53	5F	32	4F	20	mrWmuRitMha·S^?Q

Figure A3: Human readable data can be found by observing the hex value, if the secret file is TXT format

## Appendix B

**Embedded a short size data file:**

*Lena.bmp:*



*Figure B1: Stego images from LSB 1 to LSB 8, with a short size data file.*

*MondrianTree.bmp*



*Figure B2: Stego images from LSB 1 to LSB 8, with a short size data file.*

*Tiger.bmp:*



*Figure B3: Stego images from LSB 1 to LSB 8, with a short size data file.*

*Cathedral.jpg*



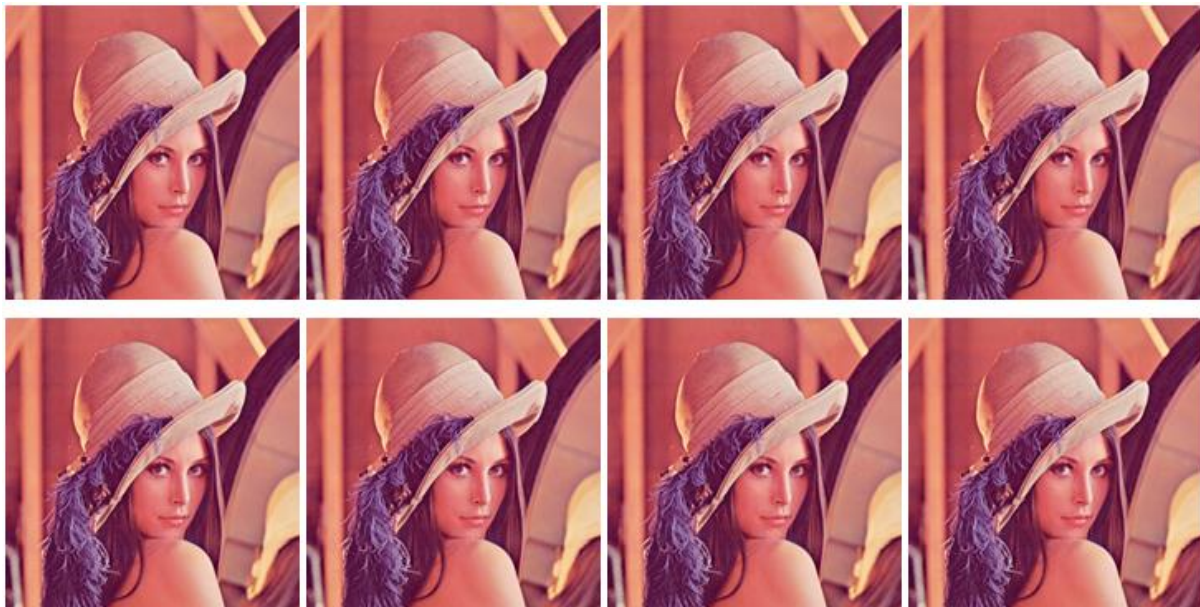
*Figure B4: Stego images from LSB 1 to LSB 8, with a short size data file.*

*Edinburgh.jpg*



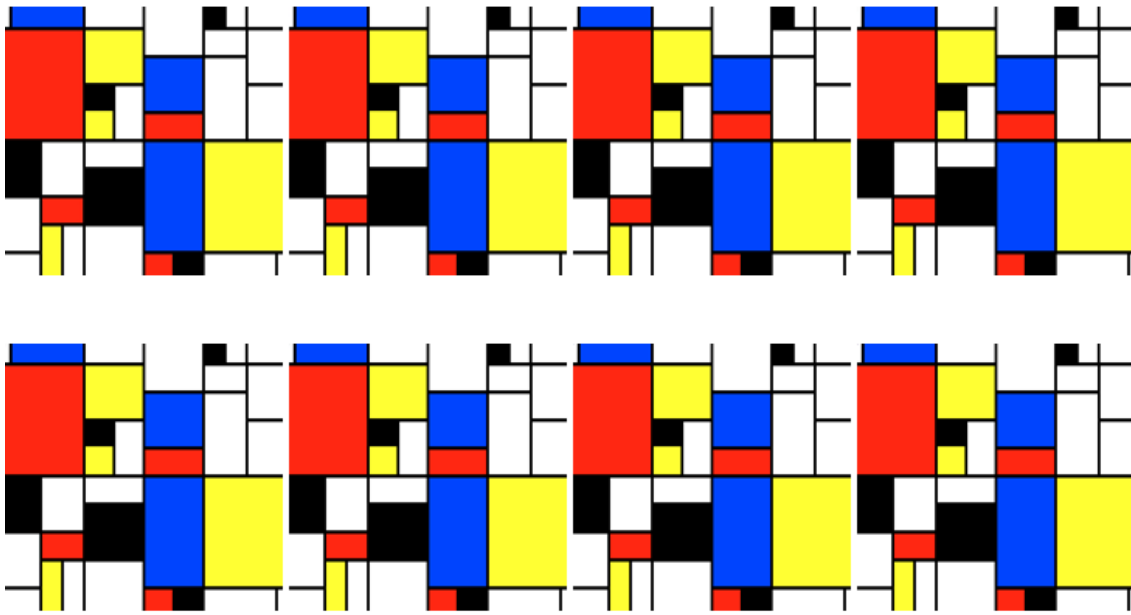
*Figure B5: Stego images from LSB 1 to LSB 8, with a short size data file.*

*Lena.png*



*Figure B6: Stego images from LSB 1 to LSB 8, with a short size data file.*

*Mondrian.png*



*Figure B7: Stego images from LSB 1 to LSB 8, with a short size data file.*

*StABridge.png*



*Figure B8: Stego images from LSB 1 to LSB 8, with a short size data file.*

**Embedded a small size data file, (42 KB)**

*Lena.bmp*



*Figure B9: Stego images from LSB 1 to LSB 8, with a small size data file.*

*MondrianTree.bmp*



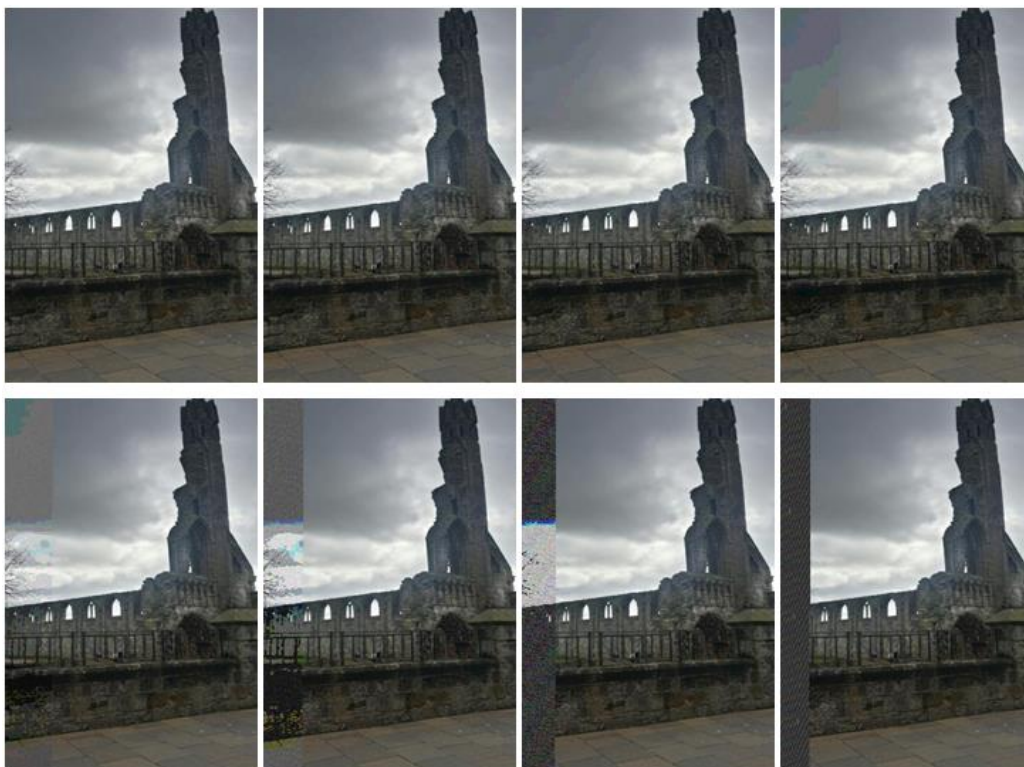
*Figure B10: Stego images from LSB 1 to LSB 8, with a small size data file.*

*Tiger.bmp*



*Figure B11: Stego images from LSB 2 to LSB 8, with a small size data file. LSB 1 is not enough to conceal the secret data*

*Cathedral.jpg*



*Figure B12: Stego images from LSB 1 to LSB 8, with a small size data file.*



Edinburgh.jpg



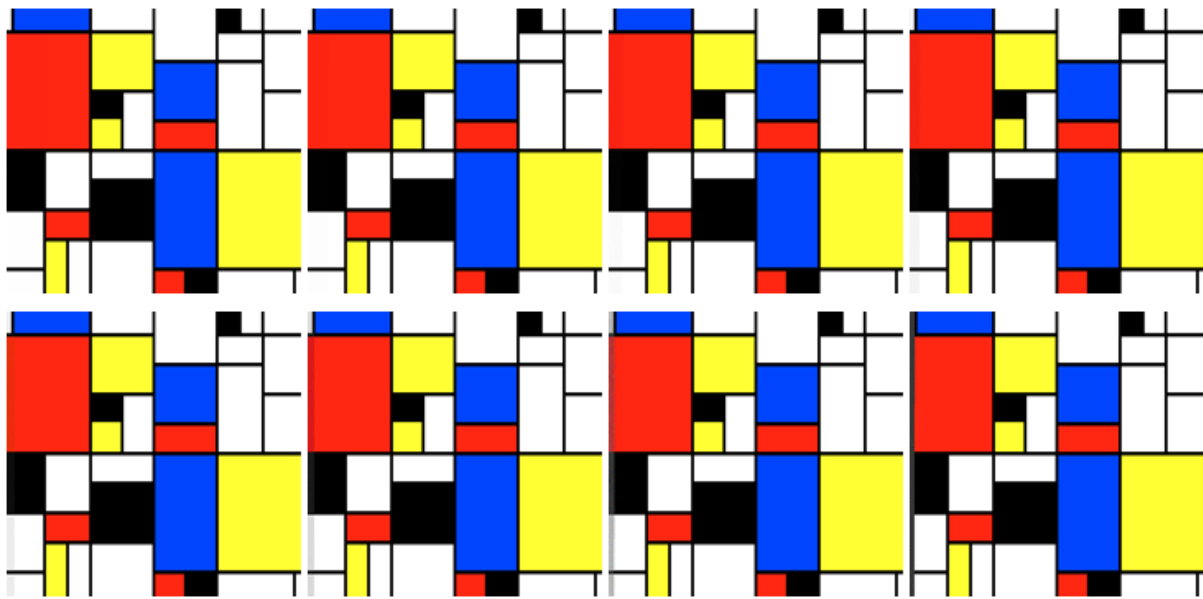
Figure B13: Stego images from LSB 1 to LSB 8, with a small size data file.

Lena.png



Figure B14: Stego images from LSB 1 to LSB 8, with a small size data file.

*Mondrian.png*



*Figure B15: Stego images from LSB 1 to LSB 8, with a small size data file.*

*StABridge.png*



*Figure B16: Stego images from LSB 1 to LSB 8, with a small size data file.*

**Embedded a large size data file, (119 KB)**

*Lena.bmp*



*Figure B17: Stego images from LSB 1 to LSB 8, with a large size data file.*

*MondrianTree.bmp*



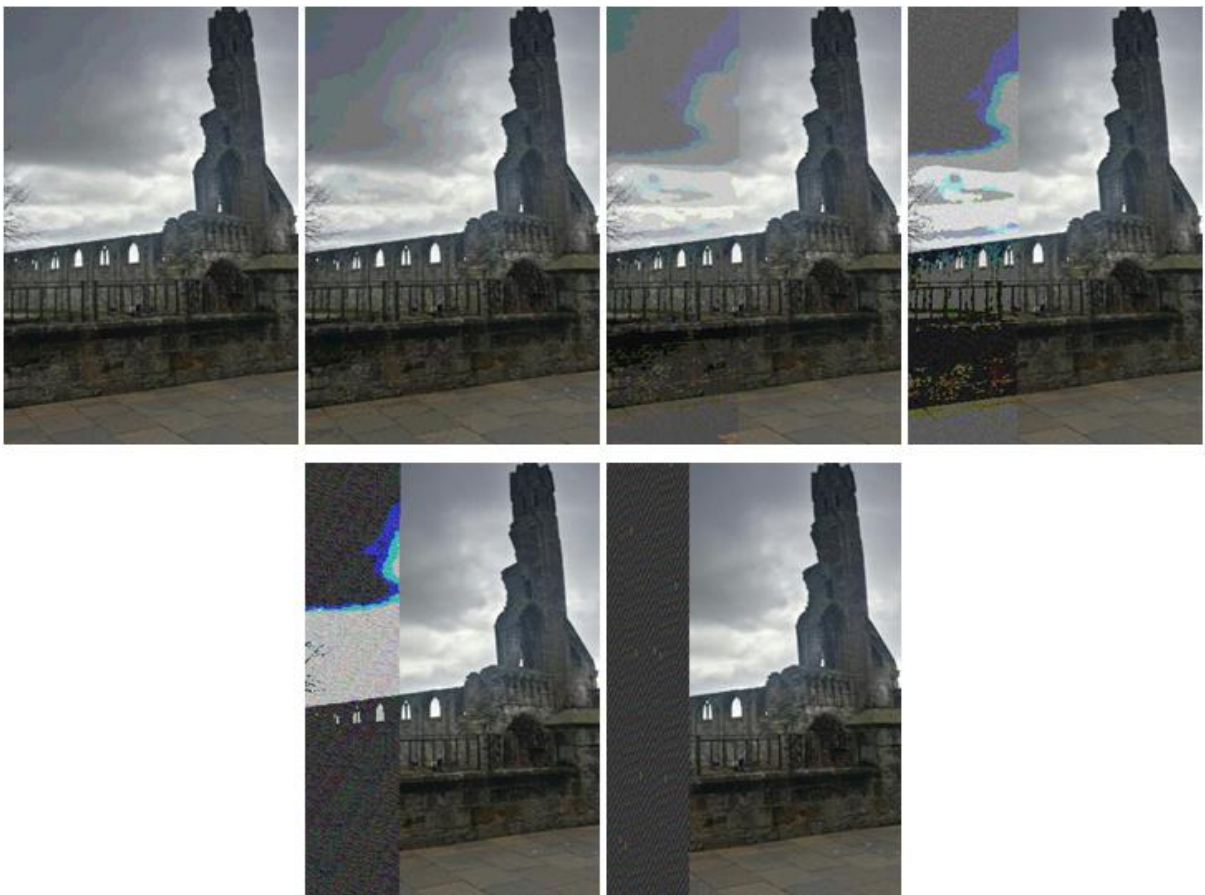
*Figure B18: Stego images from LSB 1 to LSB 8, with a large size data file.*

*Tiger.bmp*



*Figure B19: Stego images from LSB 4 to LSB 8, with a large size data file. LSB 1, LSB 2 and LSB 3 are not enough to conceal the secret data file.*

*Cathedral.jpg*



*Figure B20: Stego images from LSB 3 to LSB 8, with a large size data file. LSB 1 and LSB 2 are not enough to conceal the secret data file.*

Edinburgh.jpg

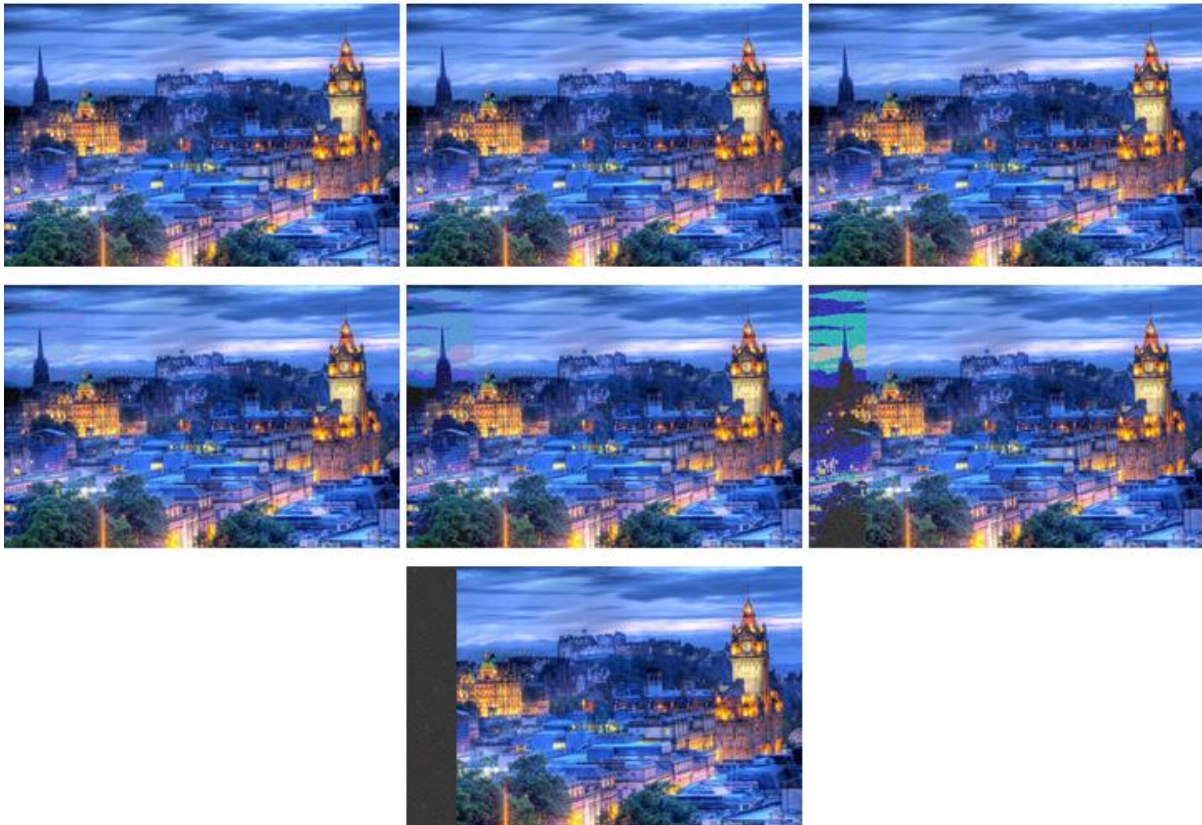


Figure B21: Stego images from LSB 2 to LSB 8, with a large size data file. LSB 1 is not enough to conceal the secret data file.

Lena.png

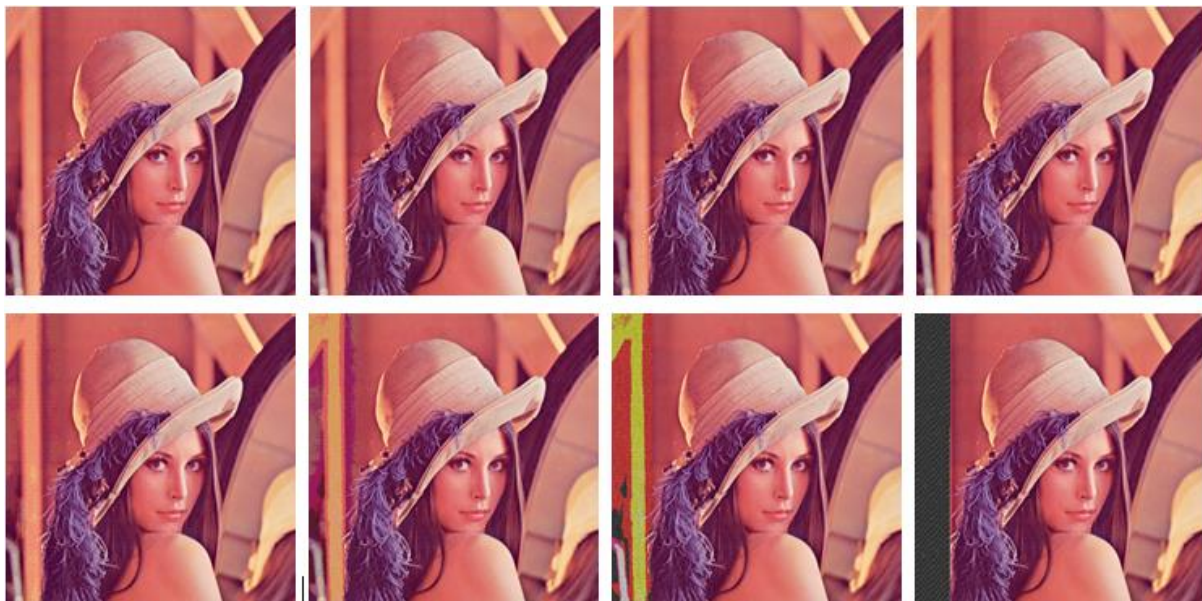


Figure B22: Stego images from LSB 1 to LSB 8, with a large size data file.

Mondrian.png

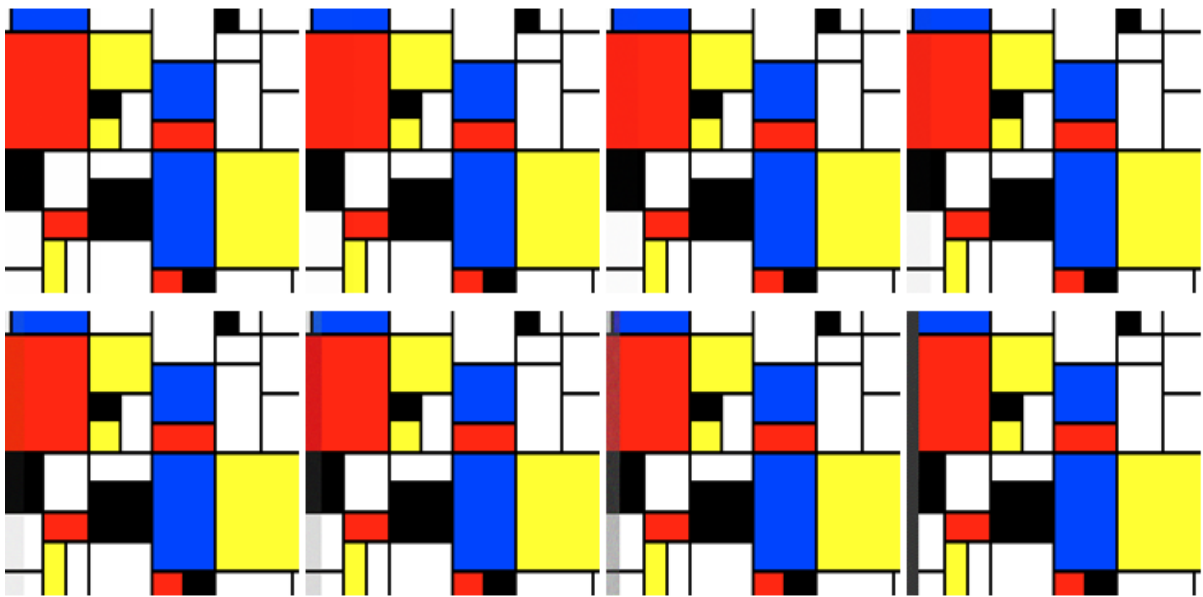


Figure B23: Stego images from LSB 1 to LSB 8, with a large size data file.

StABridge.png



Figure B24: Stego images from LSB 2 to LSB 8, with a large size data file. LSB 1 is not enough to conceal the secret data file.

**Embedded a large size data file., with random positions in the cover image, (119 KB)**

*Lena.bmp*



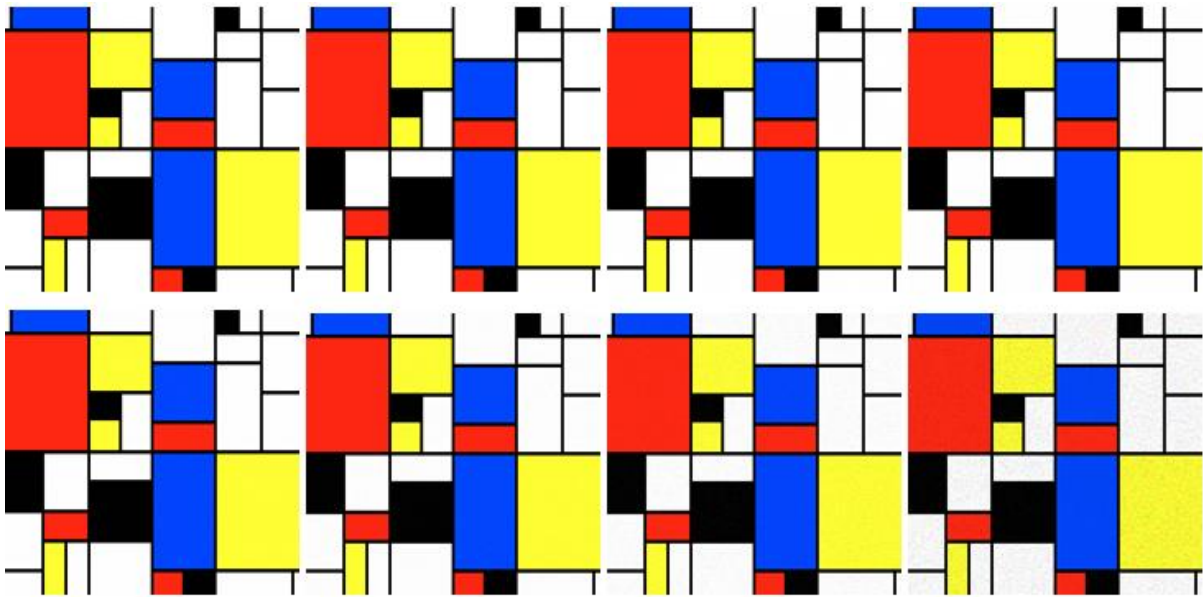
Figure B25: Stego images from LSB 1 to LSB 8, with a large size data file.

*Cathedral.jpg*



Figure B26: Stego images from LSB 2 to LSB 8, with a large size data file. LSB 1 and LSB 2 are not enough to conceal the secret data file.

*Mondrian.png*



*Figure B27: Stego images from LSB 3 to LSB 8, with a large size data file.*

*StABridge.png*



*Figure B28: Stego images from LSB 2 to LSB 8, with a large size data file. LSB 1 is not enough to conceal the secret data file.*



## Appendix C

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
1	PSNR	From LSB 1 to LSB 8	A short size	Randomly

Images in Different LSB algorithm with a <b>Short</b> size data <i>file</i> , PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
Lean.bmp	87.9493	84.5106	79.9778	75.8229	70.2931	65.4612	58.1199	57.6972
Cathedral.jpg	84.9528	80.3904	75.968	71.7981	66.7573	61.9526	55.0434	53.1763
Modrian.png	92.0359	87.2048	81.9155	77.1063	72.4719	67.3686	60.4812	54.9635
StABridge.png	86.9898	83.2955	78.175	74.125	69.468	63.4469	57.7722	54.5356

Table C1: PSNR result in images with variable LSB algorithms, randomly embedded a short size data *file*.

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
2	MSE	From LSB 1 to LSB 8	A short size	Randomly

Images in Different LSB algorithm with a <b>Short</b> size data, MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
Lean.bmp	1.04E-04	2.30E-04	6.54E-04	0.0017	0.0061	0.0185	0.1003	0.1105
Cathedral.jpg	2.08E-04	5.94E-04	0.0016	0.0043	0.0137	0.0415	0.2036	0.3129
Modrian.png	4.07E-05	1.24E-04	4.18E-04	0.0013	0.0037	0.0119	0.0582	0.2074
StABridge.png	1.30E-04	3.04E-04	9.90E-04	0.0025	0.0073	0.0294	0.1086	0.2288

Table C2: MSE result in images with variable LSB algorithms, randomly embedded a short size data.

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
3	PSNR	From LSB 1 to LSB 8	A small size	Randomly

Images in Different LSB algorithm with a <b>Small</b> size data <i>file</i> , PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
Lean.bmp	55.0319	50.5128	46.7616	41.8304	36.786	31.491	26.2654	21.4347
Cathedral.jpg	51.4417	46.8993	43.1348	38.3781	33.2206	27.7671	23.3568	19.5186
Modrian.png	59.6583	53.8652	49.4396	43.8598	39.0328	33.3797	28.3396	22.0956
StABridge.png	54.253	49.7129	45.9108	41.1638	36.021	30.4338	25.4765	20.4336

Table C3: PSNR result in images with variable LSB algorithms, randomly embedded a small size data *file*.

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
4	MSE	From LSB 1 to LSB 8	A small size	Randomly

Images in Different LSB algorithm with a <b>Small</b> size data file, MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
Lean.bmp	0.2041	0.5778	1.3706	4.2662	13.6295	46.1297	153.6544	467.3168
Cathedral.jpg	0.4666	1.3278	3.1594	9.4465	30.9754	108.7356	300.1922	726.4662
Modrian.png	0.0703	0.267	0.7398	2.6736	8.1246	29.8611	95.306	401.3491
StABridge.png	0.2442	0.6947	1.6672	4.9739	16.2547	58.8433	184.2584	588.4631

Table C4: MSE result in images with variable LSB algorithms, randomly embedded a small size data file.

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
5	PSNR	From LSB 1 to LSB 8	A large size	Randomly

Images in Different LSB algorithm with a <b>Large</b> size data file, PSNR								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
Lean.bmp	51.2429	46.7175	42.9293	38.0505	32.9881	27.6771	22.4928	17.6361
Cathedral.jpg			39.3437	34.5689	29.4275	23.9593	19.581	15.7201
Modrian.png	55.8605	50.0709	45.6369	40.0756	35.2311	29.5645	24.5452	18.3155
StABridge.png		45.9027	42.1065	37.3702	32.2217	26.617	21.6426	16.6691

Table C5: PSNR result in images with variable LSB algorithms, randomly embedded a large size data file.

No	Measurement Unit	Algorithm	Embedded Data Size	Embedded Method
6	MSE	From LSB 1 to LSB 8	A large size	Randomly

Images in Different LSB algorithm with a <b>Large</b> size data file, MSE								
	LSB 1	LSB 2	LSB 3	LSB 4	LSB 5	LSB 6	LSB 7	LSB 8
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
Lean.bmp	0.4884	1.3846	3.3124	10.1867	32.6792	111.0113	366.2703	1.12E+03
Cathedral.jpg			7.5632	22.7087	74.1874	261.3094	716.117	1.74E+03
Modrian.png	0.1687	0.6397	1.7758	6.3903	19.4971	71.884	228.3293	958.3587
StABridge.png		1.6704	4.0034	11.914	38.9866	141.7042	445.4726	1.40E+03

Table C7: MSE result in images with variable LSB algorithms, randomly embedded a large size data file.