ADVANCED STEGANOGRAPHIC ALGORITHMS AND ARCHITECTURES

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul Ştiinţa Calculatoarelor
de către

Ing. Septimiu Fabian Mare

Conducător ştiinţific:	prof.univ.dr.ing. Mircea Vlăduţiu
Referenţi ştiinţifici:	prof.univ.dr.ing. Mircea Stelian Petrescu
	prof.univ.dr.ing. Daniela Elena Popescu
	prof.univ.dr.ing. Horia Ciocârlie

Ziua susţinerii tezei: 23.11.2012

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare al Universităţii „Politehnica” din Timişoara.

Mulţumiri deosebite se cuvin conducătorului de doctorat Prof. Univ. Dr. Ing. Mircea Vlăduţiu pentru timpul preţios acordat şi pentru faptul că mi-a deschis calea către cercetare.

Prezenta teză, reprezintă rezultatul mai multor ani de cercetare în domeniul procesării imaginilor. Alegerea steganografiei ca temă de cercetare a originat dintr-o simplă descoperire făcută de mine în 2006 pe vremea când programam un utilitar pentru extragerea culorii de sub cursorul de pe ecran. Descompunerea și conversia tripletului RGB în diverse reprezentări de culoare necesare pentru aplicația pe care tocmai o dezvoltam mi-a stârnit interesul în modul în care pixelii unei imagini sunt reprezentați și relația dintre reprezentarea binară și efectul vizual al culorii. Am constatat imediat că schimbările fine asupra tripletului RGB nu produc modificări vizibile asupra culorii, diferența de nuanță fiind percepută doar la modificari mai mari. Drept urmare, am dezvoltat rapid o aplicație prin intermediul căreia puteam specifica modul în care modific fiecare pixel al unei imagini de test, iar rezultatul era afișat în timp real pe ecran. Întrucat modificările fine asupra tripletului RGB aplicat pe un singur pixel nu păreau să afecteze deloc percepția mea asupra culorii, m-am gândit că modificarile vor fi mai evidente dacă sunt aplicate asupra unei imagini de mari dimensiuni și cu o complexitate ridicată a culorii și a acoperirii de culori (număr de culori efectiv reprezentate din totalul posibil). În ciuda tuturor celor presupuse, modificările de eliminare a ultimilor 1-2 biți de culoare nu au produs aproape nici o schimbare aparentă asupra percepției mele.

Din acest moment mi-am dat seama că descoperisem ceva nou, nu toate culorile reprezentabile sunt percepute de ochiul uman, acestea avand deseori un overhead de informații. Documentându-mă, am descoperit că multe din formatele de imagini care prezintă compresie se folosesc de acest aspect pentru a elimina overhead-ul sau pentru a reduce paleta de culori doar la cele care sunt efectiv reprezentate. Interesul meu totuși a fost îndreptat asupra acelor formate de imagini care prezintă atât compresie fără pierdere de informații cât și prezervarea overhead-ului. Cei 1-2 biți care îi puteam elimina din fiecare componentă a tripletului RGB m-a făcut să privesc imaginea ca un spațiu de stocare în care primii 6 biți stocau culoarea, iar ultimii 2 stocau o informație ascunsă sub aparențe.

Cu această descoperire m-am adresat imediat domnului Prof. Univ. Dr. Ing. Mircea Vlăduţiu, care, încântat de cele prezentate m-a ajutat sa transform această joacă într-un proiect amplu de cercetare finalizat cu prezenta teză de doctorat.

Timişoara, Noiembrie 2012 Mare Septimiu Fabian

Tuturor celor care m-au susţinut

Mare, Septimiu Fabian

Advanced Steganographic algorithms and architectures

Teze de doctorat ale UPT, Seria 10, Nr. 40, Editura Politehnica, 2012,

120 pagini, 39 figuri, 12 tabele.

ISSN: 1842-7707

ISBN: 978-606-554-543-4

Cuvinte cheie: steganografie, aproximare LSB, ascundere de informatii, algoritm genetic, algoritm adaptiv

Rezumat: prezenta teză de doctorat se axează pe cercetarea în domeniul steganografiei digitale introducând totodată o serie de algoritmi capabili de a stoca informaţia într-o manieră ingenioasă care reduce probabilitatea de detecţie a steganogramei menţinând tabela de culori a imaginii originale sub control. Scopul acestei lucrări, pe langa introducerea de noi algoritmi de steganografie, este de a prezenta totodată un nou mod de abordare a problemelor care apar în urma degrădarii imaginii ca urmare a procesului de injectare de informaţii. Gradul ridicat de abstractizare şi modularizare al algoritmilor serveşte viitoarelor cercetări in domeniu drept model de lucru şi de abordare.

CONTENTS

1 Introduction 13
1.1 Motivation 16
1.2 How this work came to be 17
1.3 Thesis aims 18
1.4 Thesis outline 19
2 Steganographic background 23
2.1 Information Hiding: Digital Steganography 24
2.1.1 Classification of information hiding techniques 24
2.1.2 Types of carriers 25
2.1.3 Steganography vs. Cryptography 26
2.1.4 Steganography vs. Watermarking 27
2.1.5 Reverse engineering: Steganalysis 27
2.2 Digital Images 28
2.2.1 The RGB representation 28
2.2.2 The RGBA representation 32
2.2.3 Image compression 33
2.2.4 Pixel matrix 34
2.2.5 Color Histogram 36
2.3 Digital Image Steganography 37
2.3.1 Images as data carriers 37
2.3.2 Lossless or lossy coded images 38
2.3.3 The paradigm of steganography 39
2.3.4 LSB steganography 40
3 Related work 43
3.1 Steganalysis detection overload algorithms 44
3.2 Steganalysis avoiding algorithms 45
3.2.1 Preprocessing the image 46
3.2.2 Choosing embedding regions 46
3.2.3 Calculating total reliable storage space 46
3.2.4 Degrading the image prior to embedding and noise elimination 46
3.2.5 Controlled insertion 47

3.2.6 Processing the image while embedding 47
3.2.7 Error correction during and after embedding 47
3.2.8 Postprocessing the image 47
3.2.9 Controlled insertion using histogram preservation 47
3.3 LSB Matching steganography 48
4 Advancing Steganography 49
4.1 Natural images 50
4.2 Data deduplication 52
4.3 The ideal cover image 53
4.4 Embracing the origin 53
5 Nexim One Algorithm 57
5.1 Steganographic Encoder 58
5.1.1 Image segmentation 60
5.1.2 Solution generation 62
5.1.3 Solution evaluation 64
5.1.4 Solution embedding 65
5.2 Steganographic Decoder 67
6 Nexim One Extended Algorithm 69
7 Nexim Two Algorithm 75
7.1 Image segmentation 76
7.2 Solution generation, evaluation and optimization 78
7.2.1 Level 1 Address Generation 80
7.2.2 Level 2 Address Generation 80
7.3 Solution Enhancement 80
7.4 Solution Embedding 81
8 Nexim Two Extended Algorithm 83
8.1 Image segmentation 84
8.2 Solution generation 85
8.2.1 Level 1 Address Generation 85
8.2.2 Level 2 Address Generation 85
8.3 Solution embedding 86
9 The Nexim SCS Infrastructure 89
9.1 Sending 90
9.1.1 Encryption key generation 91
9.1.2 1st AES key half encryption 92

9.1.3 Data encryption *93*
9.1.4 Steganographic embedding *93*
9.2 Transmitting 93
9.3 Receiving 94
9.3.1 Steganographic extraction *94*
9.3.2 Decryption key recovery *94*
9.3.3 Data decryption *95*
10 Experimental results and evaluation 97
10.1 The experimental setup 97
10.1.1 Basic tests *97*
10.1.2 Advanced tests *98*
10.2 Evaluation approach 99
10.3 Experiments 100
10.3.1 Basic tests for Nexim One *100*
10.3.2 Basic test for Nexim One Extended *101*
10.3.3 Basic test over Nexim Two *102*
10.3.4 Basic test over Nexim Two Extended *103*
10.3.5 Advanced tests on all Nexim algorithms *104*
10.4 Overall result evaluation 106
11 Conclusions and perspectives 107
11.1 Thesis impact and contributions 107
11.2 Supporting publications 109
11.3 Future directions 110
11.3.1 Algorithm Refinements *110*
11.3.2 Security Infrastructure Refinements *110*
12 Bibliography 111

LIST OF FIGURES

Figure 2.1 – Timeline of the evolution of hidden data carrier [22] 23
Figure 2.2 – Steganographic carrier classification (extended from [23][24]) 24
Figure 2.3 – Carrier classification (extended from [25]) 25
Figure 2.4 – Modern digital security 26
Figure 2.5 - The RGB color representation model 29
Figure 2.6 - The RGB lightness variation 29
Figure 2.7 - Non-colors (grayscale) using RGB triplet 30
Figure 2.8 - RGB Color cube 31
Figure 2.9 – Image with alpha channel = 127 (50% opacity) 32
Figure 2.10 - Image with alpha channel = y (0% to 100% opacity) 33
Figure 2.11 - Most commonly used image formats 34
Figure 2.12 - Matrix - array transformation 35
Figure 2.13 - Fine granularity vs. coarse granularity (read buffer speed) 36
Figure 2.14 – The steganographic paradigm illustrated 39
Figure 2.15 – 1 LSB-based steganographic algorithm example 41
Figure 4.1 – Light perception experiment 51
Figure 5.1 – Encoder workflow 59
Figure 5.2 - Example segmentation (bpp = 2, tileMultiplier = 0, tableMultiplier = 3) 61
Figure 5.3 – Solution generator 63
Figure 5.4 - Example on a 4x4 pixel image 65
Figure 5.5 - Example data distribution using the jump table 66
Figure 5.6 – Decoder workflow 67
Figure 5.7 - Example jump table and data extraction 68
Figure 6.1 - The improved solution generator 70
Figure 6.2 - Enhancement through control bit 72
Figure 7.1 - Initial vs. new workflow 76
Figure 7.2 – Image segmentation 77
Figure 7.3 - The solution generator, evaluator and optimizer 79
Figure 7.4 - New addressing method 81
Figure 8.1 - Adaptive capacity 86
Figure 9.1 - Abstract representation of the communication system 89
Figure 9.2 - Detailed representation of the communication system 90
Figure 9.3 - Encryption key generator 91
Figure 9.4 - Encryption process and key appending sequence 92
Figure 9.5 - Extracting the secret data and the 1st Half AES Key 93
Figure 10.1 – The four standard images used for basic tests 98
Figure 10.2 – The four standard images used for basic tests 104
Figure 10.3 – The four standard images used for basic tests 105
Figure 10.4 – General comparison of output quality 106

LIST OF TABLES

Table 2.1 - Digital color representation..... 31
Table 5.1 – Example tiling for 2 bpp..... 61
Table 7.1 - Channel order..... 78
Table 10.1 – Capacity for the four test images 100
Table 10.2 – Optimal parameter test results 100
Table 10.3 – Performance comparison (Nexim One) [dB] 101
Table 10.4 – Gain comparison (Nexim One) [dB] 101
Table 10.5 – Performance comparison (Nexim One Extended) [dB] 102
Table 10.6 – Gain comparison (Nexim One Extended) [dB] 102
Table 10.7 – Performance comparison (Nexim Two) [dB] 102
Table 10.8 – Performance comparison (Nexim Two Extended) [dB] 103
Table 10.9 – Gain comparison (Nexim Two Extended) [dB] 103

Published papers and impact

This thesis is supported by the following published papers:

- **S.F. Mare**, M. Vladutiu, L. Prodan, "Decreasing Change Impact Using Smart LSB Pixel Mapping and Data Rearrangement", Proceedings CIT 2011 "The 11th IEEE International Conference on Computer and Information Technology", Paphos, Cyprus, August 2011, pp. 269-276, ISBN: 978-1-4577-0383-6 (IEEE Computer Society rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "HDR based steganographic algorithm", Proceedings SIITME 2011 "IEEE 17th International Symposium for Design and Technology of Electronics Packages", Timisoara, Romania, October 2011, pp. 333-338, ISBN: 978-1-4577-1276-0 (IEEE rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "Secret data communication system using steganography, AES and RSA", Proceedings SIITME 2011 "IEEE 17th International Symposium for Design and Technology of Electronics Packages", Timisoara, Romania, October 2011, pp. 339-344, ISBN: 978-1-4577-1276-0 (IEEE rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "High capacity steganographic algorithm based on payload adaptation and optimization", Proceedings SACI 2012 "IEEE 7th International Symposium on Applied Computational Intelligence and Informatics", Timisoara, Romania, May 2012, pp. 87-92, ISBN: 978-1-4673-1013-0 (IEEE, Australian Research Council list class C rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Advanced Steganographic Algorithm Using Payload Adaptation and Graceful Degradation", Proceedings ICIE 2012 "International Conference on Information Engineering", LNIT "Lecture Notes in Information Technology Vol.25" (IERI Press), Singapore, Singapore, June 2012, pp. 121-127, ISBN: 978-1-61275-024-8 (Ei Compendex, Cambridge Scientific Abstracts, Google Scholar, IEE, ISI rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Algorithm Concepts and Ways of Building More Robust, Stronger and Stealthier Steganographic Algorithms", Ph.D. Report 1, "Politehnica" University of Timisoara, January 2012, pp. 1-57

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Algorithm Concepts and Ways of Building More Robust, Stronger and Stealthier Steganographic Algorithms", Ph.D. Report 2, "Politehnica" University of Timisoara, July 2012, pp. 1-84

CHAPTER 1

Introduction

> "Steganography is the dark cousin of cryptography, the use of codes. While cryptography provides privacy, steganography is intended to provide secrecy." - Bryan Clair [1]

The Internet as a global network for intercommunication and data exchange has been ever since its creation one of the biggest and most profitable inventions of the digital age. Internet started as a closed network of super-computers (mainframes) with the main purpose being able to access data remotely at any time [2]. The outcome of its evolution over the last three decades ever since it got released to the public has not been considered when it was first introduced, what started small quickly evolved into a huge global network of inter-connected and inter-dependent devices. Since the introduction of the TCP/IP protocol suite in 1982 the Internet opened new ways to integrate almost every traditional communication aspect within, making it the number one global communication system. As outlined by Coffman and Odlyzko in [3] the initial network design was overwhelmed by the actual traffic explosion registered since the beginning of the 21st century. Comparing the traffic registered on the NSFNet backbone in 1994 where the monthly bandwidth was 15 TB/month with the crash registered by the end of the year 2000 where the traffic exceeded 238 EB (Exabytes), the registered growth in bandwidth/year came up to an outstanding 1500%, doubling every three months in size. Although it was first designed to be used exclusively by computers, the times have quickly shifted towards other devices that are able to access and communicate over the Internet. Today the Internet is not limited only to computers, but also to all sorts of personal devices such as smartphones and tablets, we live in a world where almost every aspect of communication has been integrated, if not completely adopted, by the Internet. The Internet has revolutionized and automated the ways in which we communicate, learn, exchange and access data, everything we do today weather willingly or unwillingly is done through Internet.

Computational systems have also been continuously evolving, we now own devices that fit in our hands (smartphones, tablets) that are more powerful and can store more data than a full-sized computer system from the last decade. The majority of our modern day devices are capable of connecting to the Internet making them a part of this ever-growing network [4]. We can also easily identify a global trend in using smaller and smaller devices with continuously increasing processing power [5]. Until the late 90s the only devices capable of connecting to the Internet where PCs, but this has changed since then because the majority of today's modern electronic devices are basically computers in one way or another. In

a recent article written by Cynthia Harvey [6] the smartphones and tablets are considered the new PC. Being more accessible and capable for most basic communication tasks, weather used for email, browsing the web, chatting, this type of devices have been evolving up to the point in which they now represent the majority of Internet capable devices. In [6] a comparative analysis is made between the Internet explosion in the year 2000 facilitated by the number of computer connected and today's growing market of small Internet ready devices. Still limited by bandwidth and wireless speeds, the rate in which the mobile world grows has pushed wireless research to a new level, experts estimating the same internet bottleneck by the end of 2018 if no proper precautions are taken by ISPs.

Aside from the endless list of benefits the technologic breakthroughs of the last three decades offer, comes a big risk in the form of data security. The growth of the Internet can be considered to be chaotic and somewhat uncontrolled, and since there is a raising trend of connecting and adding more devices on the Internet, privacy in our day is a very precious resource, hard to achieve and maintain [6]. The security risk we expose our devices to is widely underestimated, many considering most of the security breaches harmless. Very little attention has been paid to the need of enhanced security [7]; most companies and authorities rely on the guarantee that the security standards today are sufficient for the next few years. Enhancing security systems is only a central interest point in government or private companies, but even those have continuously proven weak against the constantly evolving attack and exploit systems [8].

Since the beginning of the digital age security systems have been developed as part of the research domain entitled Cryptography. According to Ross Andersen [9], cryptography is the intersection between security engineering and mathematics. In the early days security was an aspect that concerned more companies than private individuals, but the focus has shifted ever since. Researchers have been struggling to keep the pace with the evolution of processing power, since the strength in a cryptographic algorithms resides seldom in the higher complexity of reverse engineering that cannot be achieved with today's technology. A security system that involves cryptography in order to encode and protect data is considered secure as long as the present processing power cannot be used to easily reverse engineer the protected data in order to extract the actual information encoded within. In other words cryptography offers a limited liability for the degree of security and privacy that it offers. What is secure today may not remain secure in the near future. The cryptographic standards today basically split into three types of algorithms: symmetric [10], asymmetric [11] and one-way cryptography [12] (hashing algorithms).

In parallel to cryptography, cryptanalysis research is trying to identify weak spots that can be exploited in the nearest future with the aid of more processing power. This cat and mouse game between engineering and reverse engineering cryptographic standards is maintaining an equilibrium between the algorithm complexity and processing power needed to solve the mathematical complexity. Usually the complexity is chosen in a way that facilitates high speeds for encoding and decoding data when the proper perquisites are used (the keys are known) and in the same time, reverse engineering would need to solve a complexity x times higher than the statistical forecast of computational power in the near future. This may seem enough for the time being, but most cryptographic standards in our days are proven weaker and weaker as the technology evolves. Cryptographic research, although intensive proves itself weak against recent advances in security research. Since the demand of security surpasses by far the slow pace in which cryptography

evolves, researchers have been focusing on introducing security infrastructures that combine the best and strongest mix of algorithms in order to secure data. Systems like the PKCS [11] currently at revision 2.1 are widely used in most secure communication today.

These security systems use a mixture of strong military strength algorithms in a manner that sandboxes the encryption process, offering a variety of applications and implementation possibilities. Given that this standard utilizes a variety of ciphers and hashing algorithms, it becomes vulnerable if one of its components are broken or proven weak. The recent successful hacks on the popular MD5 algorithm has weaken the entire chain of security protocols that relied on it [13]. In a recent paper published in 2005 by Wang and Yu, the authors offer a relatively simple solution to exploit hashing in almost every algorithm existent (general mathematical exploit). This research issued an alarming state regarding internet security as hashing algorithms are intensively used to store passwords, fingerprint digital information or the signing of documents as described by the public key cryptography standard [11].

Another contemporary security threat does not imply breaking cryptographic cyphers, but listening to the connection between two entities in a process also known as man-in-the-middle attack ([13][14][15]). The secure communication between two entities can be broken if a third party listener is involved. Usually we refer to these listeners as hackers trying to intercept the data exchanged - this is often done without the entities even knowing their presence. If the entities are communicating without any sort of security system involved the intercepting party can immediately make use of the intercepted data. If the communicating entities use a secure channel, the data becomes harder if not impossible to extract. This has been considered for many years a foolproof system.

Many consider data interception a smaller threat when cryptography is involved for securing the packages sent between the communicating entities since modern security standards offer end-to-end encryption [16]. Only the communicating parties are able to generate, encrypt and decrypt the information. The usage of one-time passwords and key exchange mechanisms make the man in the middle type of attack harder if not impossible [17]. Still, identifying the usage of cryptography over secure a communication channel can be easily pointed out because of the RAW format in which the data travels. Without the necessary keys the data may seem gibberish and without any logic at first sight. Once the encoded data has been intercepted it is a matter of time until it can get reverse engineered, usually if today's processing power or cryptanalysis advances do not facilitate reverse engineering the cipher text, it can still be saved for later analysis [15]. Even if today's processing power does not aid in the extraction, one cannot guarantee what the future will bring. The privacy of the secured data is also time related. If the data is only important from a small time perspective, future breaching and extraction of the intercepted data does not pose a risk. If on the other hand the validity of the secured data can be used from a larger time perspective, interception is as dangerous as it can be [18]. Today, many cryptographic cyphers and hashing algorithms have gone extinct due to their weakness against hardware/software, mathematical and statistical attacks.

In the digital age security is almost a synonym with cryptography. Analyzed apart, security implies more than just transcoding the sensitive information into an unreadable RAW state without the proper perquisites (e.g. keys). In the most recent years a new trend has been developed that covers the aspects that cryptography alone can't: hiding the trace and existence of a secured communication even taking

place [19]. This technique may not be new, but its integration with digital systems is: steganography [20]. Steganography is the art and science of communicating in a way that only the parties implied in the communication process know of the existence of the data. Usually it implies the usage of decoy information within which the actual secret information is stored. From the digital point of view, it implies using a format that was not intended to store other information apart from its designed one. For example, one can communicate secret messages disguised within plain text, images, video or audio streams called cover formats. In this way, an intercepting party would receive the decoy information (with the secret information contained within) but would be unable to identify something other what the appearance offers. The choice of a decoy format is the first and most important step to take in order to achieve a desired degree of stealth [19][20]. One of the most commonly used and transmitted media types on the Internet are images, making them the number one decoy (cover) formats used by many researchers in the steganography research field. The joint usage of cryptography and steganography leads to a truly secure communication system that covers many aspects of the ideal communication system.

1.1 Motivation

When it comes to securely transmit sensitive information, there are several ways in which this can be accomplished. With security protocols and cryptographic techniques growing stronger and more complex every day it is almost impossible to break a secure communication chain. From an attacker's perspective though, decoding the secret information is not always the main target. In our days breaking a secure communication chain in order to extract and decode the data is not the attacker's main intention, but to be able to trick the legitimate participants into passing their data through a rogue gateway in a technique that is called man in the middle or phishing. This technique has been proven very effective in numerous attacks. The legitimate participants are connected to a third party gateway that acts like an intermediate between the participants in a secure communication chain. The key exchange process needed for temporary session encryption passwords is done by the rogue gateway, which communicates to both parties as if it wasn't even there. The legitimate participants have the impression that their connection is direct, but instead they are passing information to the gateway, encrypted using the keys provided by the third party intercepting platform, therefor exposing the secret information without even knowing. The described scenario is just a small example from the numerous existing exploiting techniques attackers use in our days.

By analyzing the above scenario a little bit we can identify that the actual starting point of such an attack originates by identifying the communication channel. Cryptography represents a technique that transcodes (encodes) the sensitive information into an unreadable and gibberish state called ciphertext that can only be reassembled (decoded) using the proper keys. Given that there is a man in the middle in a communication attack, the attacker would listen to the communication of the legitimate parties and filter out only those packets that present some sort of interest. Usually it is hard to track and filter these packets without knowing where or for what to look for. In the event the legitimate parties switch from unsecured plain data to encrypted data, the attacker could easily spot out these information because the marker is represented by the ciphertext itself. Cryptography only places the sensitive information in a secure "vault" that is the ciphertext. Unlike real life, data

can be copied and multiplied, implying that our ciphertext can have multiple destinations out of which not all must be legitimate. The degree of security is influenced by multiple variables that define a cryptographic algorithm: the strength of the cryptographic algorithm used the length of the keys, the size of the actual ciphertext. The effects of a data interception usually are not immediately benefactor for the attacker. Usually it takes a long time until the sensitive information can be extracted, facilitated by numerous advances in processing power and cryptanalysis. If the secret information transmitted is valid only for a short period, breaching the security of an interception in the near future does not present a threat. If on the other hand the secret information has a validity that lasts for years, interception and storage for later analysis and deconstruction presents a serious security breach.

In modern days usually these types of information are transmitted through private or internal networks that are completely isolated from the Internet. This may prove effective in most of the situations but does not cover the event in which there is that 1% time in which the information must leave the secure private network and be transmitted through the Internet. Once the data leaves the secure network it can be susceptible to multiple types of attacks and interception attempts. Cryptography in essence is not a guarantee for security as the cryptographic standards of today usually have a relatively reduced lifecycle in which they are considered secure.

It may not be immediately obvious for security researchers but there are several ways in which security in these situations can be enforced, cryptography alone is not always the best solution. Steganography fills this gap by including the sensitive data either encrypted or in its original state within a carrier format (hidden inside other digital formats) that, in the event of which it gets intercepted, would not arouse any suspicion that a covert communication is even taking place. Steganography does not represent an invention of the digital age; it has been used long before the digital revolution. Basically steganography implies the usage of conventional formats used as a data carrier for the secret information. The main aim of steganography is to hide the communication channel itself, leaving the third party attacker monitoring a communication that seems normal at first sight; the only ones knowing where to look for the secret data would be the communicating parties.

Choosing steganography as a research domain represents a very challenging and interesting task, as it represents a technique unknown for the majority researchers, even in cryptography. Many researchers see this technology as a threat because of the numerous ways in which this can be used for espionage purposes, omitting the actual benefits this technology brings when used in combination with modern security protocols and cryptography.

1.2 How this work came to be

The current thesis started out as a personal project derived from a discovery I made when playing with the pixel table of a normal RGB image back in 2006. The intention was to program a simple tool that extracts the color under the cursor and then converts it to various color representations such as YUV, YCbCr, HSL and HSV. Internally the tool captured screenshots of the entire working area and then extracted a 100 by 100 pixel region around the current position of the mouse cursor. The color information was stored as a BMP formatted pixel table. In addition to the color under the mouse cursor I had to build a set of 4 color histograms for the squared pixel region to analyze color distribution so R, G, B and RGB combined

histograms here built. In order to build a histogram for each channel, the pixel needed to be deconstructed into its primary components that captured my attention so I did a little bit more research in this domain. I then studied the relationship between the binary representations of the three-color components and the resulting pixel color. The fascination for the bond between visual and digital led me into building a program where I would take a normal image and play with its pixel table. I studied filters, the logic behind them and how they affect the image and our perception upon it. At some point I wanted in a purely unintentional attempt to tamper the LSBs of every color component and discovered that finely altering the digital value of the color components does not affect my perception on the resulting image. It was then clear that the image format I used in my testing was designed to store more color variations than my average human eye can distinguish between. What started out as a play derived from the current work quickly turned into a growing curiosity on why things happen the way they do. Since the unintentional discovery that the image format presented overhead information that was insensitive to change from a visual perspective, I shifted my focus to the LSBs that I've seen as potential storage bits.

The experiment quickly shifted and I built a small application that would include a PDF file inside the LSBs of every pixel and output an image. By comparing the original with the tampered image I discovered that they seem absolutely identical from the visual perspective, although binary they were different.

This discovery represented the propulsion for every research I've done ever since then. As my research continued I stumbled upon the domain called steganography, which was not immediately identified since I first believed I invented something completely unique. In the coming years I studied different steganographic algorithms that worked under various image formats and pixel representation standards. I also started working on reverse engineering these algorithms in a process known as steganalysis. Unlike cryptography, in steganography the main threat is detection, so I tried to analyze the faults presented by other steganography techniques. Over the coming years several algorithms have been developed that covered a variety of aspects.

In parallel to this research a second research has been conducted in which I studied cryptographic algorithms in order to statistically find weak spots that could lead to exploits in the near future. The idea came to somehow bind the two projects in order to achieve maximum security. Cryptography would then become an underlying component inside our steganographic algorithms and together the results where both secured and hidden in plain sight.

Since 2009 when the current PhD research has officially started I've introduced a series of very strong algorithms to be used and implemented either in modern security infrastructure systems or to serve as a basis for future research for others in the steganography field.

1.3 Thesis aims

Ever since my initial discovery, I've been constantly building steganographic algorithms and improving upon them to achieve a certain level of design perfection and performance. Since this domain is widely unexplored, many researchers have been introducing algorithms that vary in quality, storage size, resistance to attacks and detection. I've identified a lack of standardization and metrics needed to

quantify the validity of a solution; it is hard to compare one steganographic solution to another. This is one of the main reasons I've been trying to build our own set of quality metrics and benchmarks in order to place our discoveries in relationship with others.

The current thesis introduces the finest and most sophisticated algorithms I've developed throughout the years. I focused on building algorithms that don't compromise neither the quality of the resulting image nor the storage space for the secret data in my own unique and innovative ways. The algorithms are designed in a very modular way so that they serve both as complete stand-alone solutions and abstract models for future improvements and analysis. The introduction of my algorithms and the presentation at various conferences was well received by the others researchers, some of my solutions where used by others in their own work. One of my personal observations made at almost every presentation I held was the difficulty with which the majority of researchers understand this technology and its benefits. Most researchers that deal with data security and cryptography have a hard time understanding the problem that the interception of the ciphertext presents. Other than that it is not immediately obvious for most what the relationship between cryptography and steganography is all about. This is why, in addition to new steganographic algorithms I also introduced a new secure communication infrastructure that uses cryptography both symmetric and asymmetric together with steganography in order to achieve the maximum level of information security between two entities that are communicating through an unsecure communication channel such as the Internet.

1.4 Thesis outline

Steganography represents a very complex research domain, facilitated by the multitude of carriers available and also by the endless ways in which stealth can be achieved. The current thesis is structured in 11 chapters. In *Chapter 1* I present a brief introduction into the evolution of computing systems and the increasing demand for security as our world tends to be more interactive and interconnected. Also, in this chapter I present the starting point of the current work and what drove me into researching this domain and turning it into a personal passion in algorithm design. The thesis introduces one of the finest and most powerful algorithms I've developed since I discovered steganography and also presents an in-depth analysis of the factors that influence a truly powerful steganographic algorithm.

In almost all my presentations held at international conferences I noticed a constant confusion and difficulty on understanding this technology and its strong relationship with cryptography and overall digital security. Given this, *Chapter 2* covers all aspects related to steganography from ancient methods up until the adoption of traditional steganography in the digital area. In digital steganography the only limitation is the imagination of the algorithm designer, since the digital world opens endless possibilities in which one can securely and stealthy transmit secret sensitive information. The widely unexplored domain of steganography lacks in standards and classifications, each researcher introduces new overviews over this research domain. In 2.1.1 I took the liberty to introduce a better and well-structured classification of steganography as well as the carrier types one can use to hide data. For many, the three building blocks of modern digital security, cryptography, steganography and watermarking, are not clear, this is one of the

main reasons this chapter also covers a comparative analysis of the three domains. Also, this chapter introduces the basics of digital imaging and image-based steganography since this is the main scope of the current thesis: introducing new and enhanced image steganography algorithms.

In *Chapter 3* I present an overview of the current state of the art in digital image steganography as well as an analysis of the current algorithms. This chapter also highlights the flaws that make many of the algorithms vulnerable to reverse engineering: steganalysis. At the end of *Chapter 3* I present a new emerging trend in algorithm design that shifts from the traditional methods of enhancing the quality and capacity of the carrier after the data has been embedded to a more cover-oriented model that controls the degradation more accurately.

In order to avoid the limitations that other algorithms face, it is important to change the starting point of conceiving a new steganographic strategy. *Chapter 4* presents a new direction I wish to introduce with the current work, in which I analyze and set the building blocks of an ideal model. By avoiding the same mistakes other algorithm designers make and by changing the design perspective one can successfully build even stronger algorithms with a dramatically increased resistance to reverse engineering and steganalysis filters.

Chapter 5 introduces one of the first algorithms I designed that try to cover as much as possible the ideal model presented in *Chapter 4.* The algorithm itself is capable of achieving a higher resistance to detection algorithms than most of the strong algorithms known up to this day. The algorithm presented serves both as an implementation solution with notable results, but also as an abstract model for other implementations and derivations. With the introduction of Nexim One I wish to set a standard in LSB matching algorithm design. The modularity in which the entire workflow was designed facilitates improvements in almost all stages of the embedding and enhancement process. In my extended research and after intense testing I found several isolated cases in which the algorithm was unable to output the desired quality. This is why in *Chapter 6* I present an extension to the original design that eliminates the errors I've found whilst testing.

In an attempt to improve the storage capacity of the initial design, I've come up with a new algorithm that follows the guidelines of the original Nexim One, but increases the storage capacity of the original design, maintaining or even improving the resulting quality. Therefor in *Chapter 7* I introduce Nexim Two, which follows the general design of the original Nexim One but changes the weakest links in the entire process. *Chapter 8* presents an extension to Nexim Two, which focuses mainly on the dynamic part, which the original algorithm lacks. Nexim Two Extended therefor represents my strongest algorithm so far that surpasses not only my other designs, but also one of the best LSB matching algorithms up to this day.

Besides introducing several new steganographic algorithms, the current paper also introduces a steganography-based secure communication infrastructure (*Chapter 9)* that combines the power of both symmetrical and asymmetrical cryptography with information hiding technologies (steganography). The infrastructure represents an abstract concept in which the data that is communicated between two parties travels both secure and hidden in plain sight, inside steganographic carriers. The introduction of this system aims to present better alternatives to the existing cryptographic infrastructures that have been continuously proven weak against new emerging types of attacks and spoofing methods.

Chapter 10 presents the experimental setup that I used in order to differentiate and highlight the performance of my algorithm against other state of

the art algorithms. This chapter presents the experimental setup used, the metrics and the results of my designs. In addition the chapter also presents the performance gain in stress tests and prolonged execution times.

The final chapter, *Chapter 11,* concludes the current thesis by highlighting once again the performance of the new designs as well as the original contributions introduced by this work. Also, I conclude my current thesis by providing future research directions in the field of LSB steganography.

CHAPTER 2

Steganographic background

> "Any sufficiently advanced technology is indistinguishable from magic." - Arthur C. Clark

> "The most beautiful thing we can experience is the mysterious. It is the source of all true art and all science." - Albert Einstein

Steganography represents art and science of hiding information in a manner than no other person except the two communicating in this way knows of the communication is even taking place [21]. The term steganography originates from ancient Greek meaning "concealed/protected writing" and has been used for over 2500 years in various forms and applications. Basically steganography is a term used to describe any sort of communication, where the secret message is hidden within an object that is not apparent to an outside observer. Ancient steganography usages have been found in a variety of forms, mostly used in military, political or espionage. Steganography basically is a form of ancient privacy methods that combines the ingenuity of writing secrets with the art of hiding them [21].

The ancient cryptographic cyphers basically are one and the same as steganography, since the ciphertext (codex) needed to be stored securely in one form or another. Cipher texts inside ancient books basically are just a few of the most commonly discovered forms of steganography. *Figure 2.1* is a graphical timeline illustration on how hidden data carriers have evolved over time, depending on recent discoveries as described by the authors in [22].

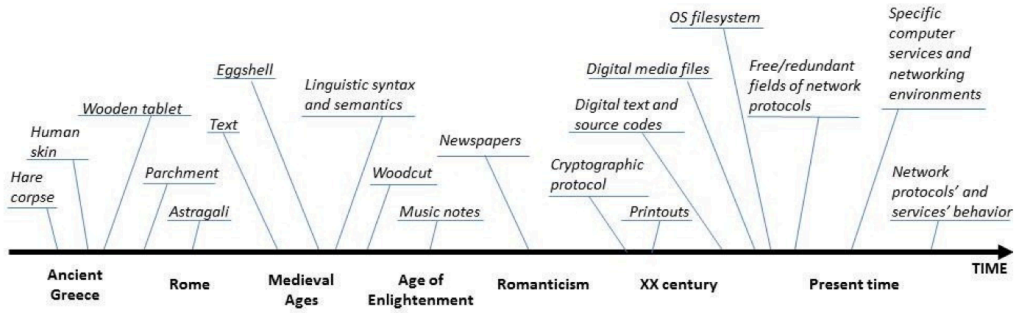

Figure 2.1 – Timeline of the evolution of hidden data carrier [22]

2.1 Information Hiding: Digital Steganography

Digital steganography represents the science of writing hidden messages (text, data, etc.) inside of other digital content, possible by the joining of the old methodologies with the digital world [20][21][22]. The old physical related embedding techniques used in ancient steganography pose themselves limited in comparison with the new possibilities of the digital world. In a binary world with huge quantities of data, there is no end in sight when it comes to choosing the right carrier or the right method. Just like the definition states, steganography represents the intersection of science and the art of hiding: the science in this case is the engineering of algorithms and the art relies in finding the perfect secret data carrier.

2.1.1 Classification of information hiding techniques

Although steganography is a very large and mostly unexplored research domain, there is a lack of classification; almost every researcher classifies steganography different, weather in context with cryptography or as part of a larger research domain called information hiding. Probably the most adequate classification I've found so far is the one by P. Pfitzman [23] and P. Petricolas [24], where they classify steganography as being a member of the information-hiding domain, together with watermarking, covert communication channels and anonymity methods. I consider this categorizing most adequate because it covers both digital and traditional methods unified in one dependency tree. Being a member of a large research field called information hiding, steganography covers the security aspects that cryptography alone can't: hiding the very existence of data. *Figure 2.2* illustrates the categorization of the image hiding research domain.

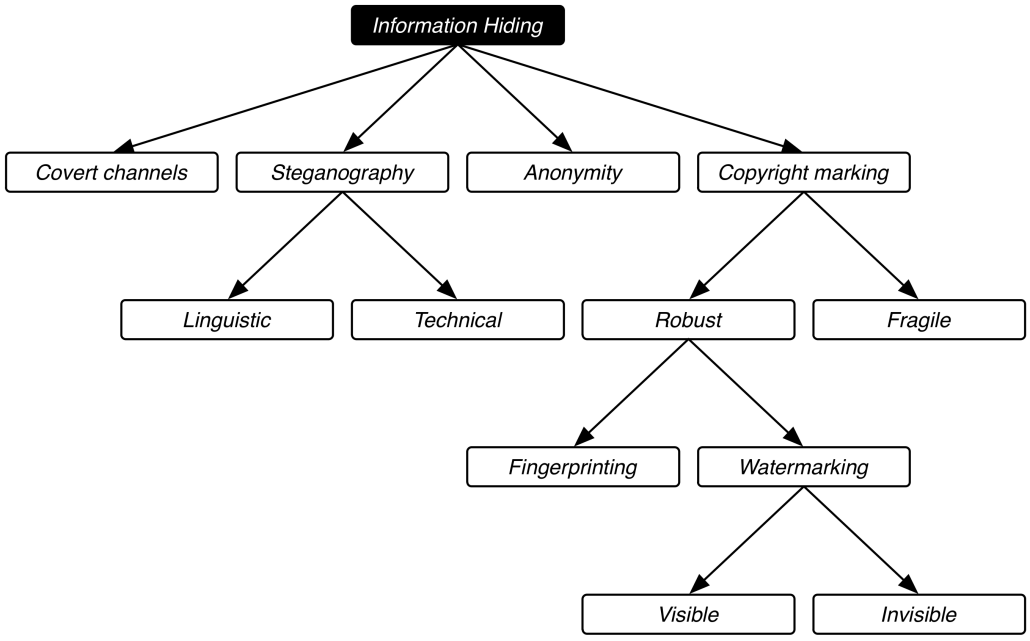

Figure 2.2 – Steganographic carrier classification (extended from [23][24])

2.1.2 Types of carriers

In digital terms steganography implies using digital formats for the storage of secret data. The choice of format varies depending on the scope of the intended communication, but typically the widely used categories of digital carriers are the following:

- Images
- Video
- Audio
- Archives
- Executables and libraries
- Documents
- Ciphertext

The above list represents only a few of the most used formats, but steganography is not limited to them. Basically steganography can be applied on any format that has overhead data that can be manipulated without affecting the visual, statistical or logical integrity of the original carrier [25]. *Figure 2.3* illustrates a detailed classification of the most common carrier types used in digital steganography.

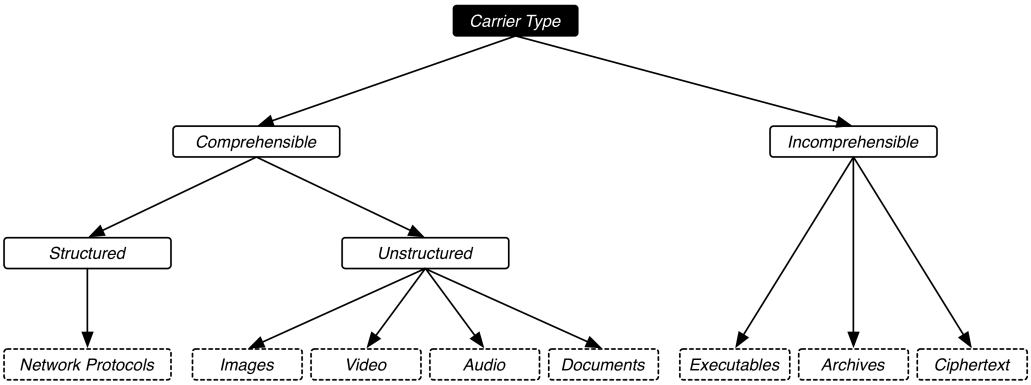

Figure 2.3 – Carrier classification (extended from [25])

As these digital formats where not designed to hold any secret data it is normal that a new research direction has been opened in which the main focus is to find new ways to store the secret data without affecting the perceptive integrity of the digital carrier. The perceptive integrity of a digital format represents the way in which a human or a computerized analysis system perceives the digital information; weather if it's visual (images or video) or audio (music), steganography must handle

and add the subtle change to the original form, in a way that doesn't raise suspicion. Suspicion is raised in the event that the information carrier is altered up to the point where it resembles more the data within than its original representation. Nevertheless, the logical scattering of information in the carrier must be kept under control in order to avoid digital detection.

2.1.3 Steganography vs. Cryptography

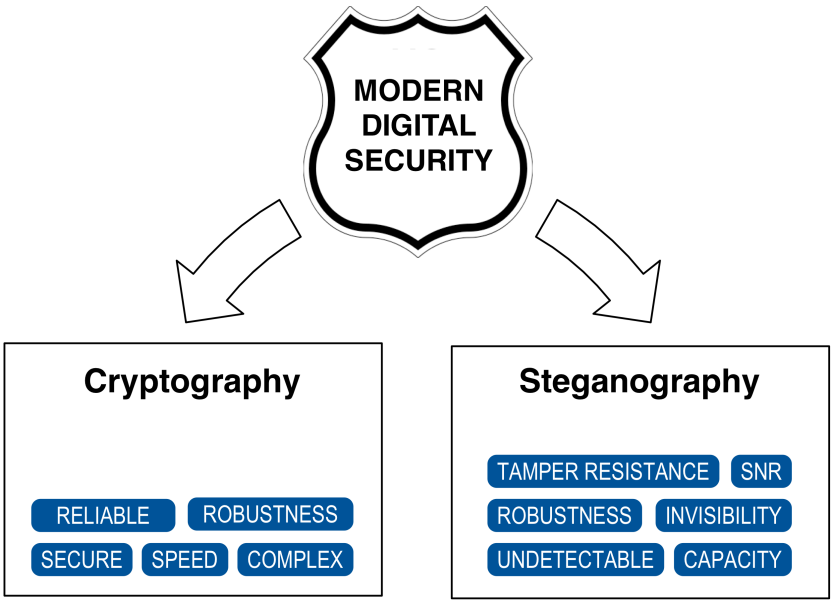

Figure 2.4 – Modern digital security

Cryptography in general is synonym with encryption, which represents the technique of processing information until it becomes unreadable without the proper tools, perquisites or variables. In digital terms, encryption resembles the intersection of mathematics, computer science and engineering to build algorithms capable of encrypting data into unreadable (but logic) state called ciphertext [26]. The information can only be re-assembled/extracted (decrypted) using the adequate password(s). Although cryptography can assure a high level of security, the secured data is still exposed and can be intercepted by a third party attacker or listener. The problem appears in situations in which the data cannot be fixed located, but it must travel through unsecure channels such as the Internet. [27][28] Although the data travels in a safe mode represented as a ciphertext, it can be still intercepted and reverse engineering can be performed onto it. In one word, the main problem of cryptography is: data exposure. Data exposure is not always an immediate threat. In order for the data to become vulnerable an attacker must first intercept it. Depending on the strength of the encryption algorithm used, the data can either be extracted using known attacks leading to an immediate breach of security or it can be stored for future reverse engineering when the encryption standard has become vulnerable due to recent studies and cryptanalysis.

Steganography also represents a type of encryption but instead of outputting an incomprehensible and identifiable format such as the cryptographic ciphertext, it outputs a digital, comprehensible format such as an image, video, audio or any other type of digital information that has redundant data to begin with. Its main purpose is to resolve the problem that cryptography alone can't: data exposure. Steganography covers more the hiding of data than the security of it [29]. Basically, steganography is security through obscurity or based on the hypothesis "what can't be proven does not exist". If a steganographic algorithm is undetectable, the data also becomes secure even without usage of encryption. The steganographic data can travel through unsecured channels without raising suspicion, in opposition with cryptography, where the secured data is spotted in an instance because of its ciphertext representation [29][30].

Steganography is a complementary technology that, if used together with cryptography or even archiving, can achieve a level of security higher than any stand-alone security system offers. Combining these three systems offers a truly secure and robust communication channel suited for our modern day security requirements.

2.1.4 Steganography vs. Watermarking

Although it is often considered a separate research direction [31], watermarking and steganography are directly related. Both technologies serve for information hiding, the main difference relies in the scope and usage. In steganography there is usually no relation between the cover medium and the secret data stored within, the cover serves solely for storage and hiding purposes. Watermarking on the other hand is directly related to the cover medium, as its main purpose is to store copyright or important information about the medium in a tamper-secure way.

Where as steganography offers capacity at the cost of losing resistance to modification, watermarking offers a negligible amount of capacity but it enforces tamper resistance and removing techniques. In invisible watermarking the data is stored in multiple regions and with an improved reconstruction mechanism that handles the attempts to remove the mark from the medium. Usually watermarking techniques are mostly used in digital rights management systems (DRMs) in order to protect the authenticity and origin of the medium [32].

Steganography focuses on hiding a large amount of information. In an ideal case a steganographic algorithm is capable of preserving the secret data even when the cover medium surpasses modifications. The more resistant this type of algorithm is to the changes, the more the steganographic algorithm turns into a watermarking solution. The principles of watermarking are rarely used when conceiving a steganographic algorithm, as they are very hard to realize and usually decrease the overall storage space that steganography deals with.

2.1.5 Reverse engineering: Steganalysis

The reverse process of steganography is called steganalysis and it deals with estimating the chances of hidden data existence [32][33]. Because all carrier mediums are formed out of data that does not necessarily follow a certain rule (images are a hazardous distribution of colors, sounds are a hazard of tones) the steganalysis algorithms must try to differentiate between a natural hazardous distribution and a controlled hazardous distribution. Digital steganography is a complex research domain in all aspects. With each digital format used as carrier for

the secret data, arise problematics that concern the various types of implementations available.

The biggest problem in digital steganography is not the reverse engineering of the method itself with the purpose of extracting the secret data, but detecting that there is secret data present. Every algorithm is as strong as its resistance to advanced detection algorithms (steganalysis) [31][32]. Regardless of the type of carrier used, the original form normally represents a natural hazard of signal variations (the signal can be either color variations when using images or videos, or sound intensities when dealing with audio types). The data we usually hide inside other carriers represent logical information that added to the original alters the overall result by introducing noise channels. Since a normal high quality digital format stores more information than we can perceive, this can go undetected up to a certain point. However, computerized detection is the main problem when developing and testing the strength of a steganographic algorithm and is one of the key benchmarks to use in order to validate it.

Since there are a variety of techniques for hiding data in a steganographic manner, the complexity of the algorithm can be much higher than in the case of cryptography. Depending on the carrier, a steganographic analysis must start with the identification of method used to hide the data [33]. Based on these premises, steganalysis basically elaborates a set of tests and measurements on the carrier and issues a statistical analysis over the chance of secret data presence within. Since steganalysis can also be applied to normal digital formats that are unaltered through a steganographic process, the analysis method must issue a metric that is able to distinguish between normal and steganographic formats, either by issuing a suspicion rate in percentage or a quality metric.

Steganalysis research has two major purposes: on the one hand researchers are trying to detect steganographic use for illegitimate purposes, on the other hand it serves as a benchmark for newly developed steganographic techniques. The techniques of detecting the secret communication channel represents a more elaborated research domain than steganography itself, algorithm designs must take multiple types of attacks and filtering techniques into consideration in order to avoid them. Since the main purpose of steganography is avoiding detection, this has proven not to be a simple task to achieve.

2.2 Digital Images

A digital image format is a file in which the picture is represented in a way in which its components are stored as pixels. The pixel is the smallest measurement unit in digital imaging, and it is normally arranged in a two dimensional grid (matrix). The word pixel is derived from two words: pix meaning pictures and el meaning element, so basically a pixel is an image element. The pixel may be the smallest unit in digital imaging, but it has its own intern representation.

2.2.1 The RGB representation

The RGB model is a pixel color representation in which the color is given by a combination of the base colors plus light. There are three base colors, also known as color channels:

- Red (R)
- Green (G)
- Blue (B)

The RGB model is a device dependent color representation model because not every device will display the same color. RGB, also known as the RGB triplet , is used by most digital imaging systems because it is a RAW image format, meaning that it preserves the color information. The RGB triplet can be represented in multiple forms such as:

- Numeric
- Geometric
- Digital

The *numeric representation* indicates the quantity of each color channel that will be used to form the final color. The quantity of each color is defined as a triple value, in which each color channel has a value interval that starts from 0 (meaning absolute obscurity), and ends in a custom defined maximum (meaning absolute intensity).

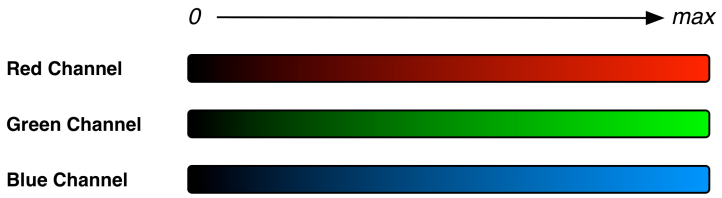

Figure 2.5 - The RGB color representation model

It can be observed in *figure 2.5* that the colors vary from dark (absolute obscurity) to pure (absolute intensity). The only missing part that would complete the color spectrum is the *lightness* of these channels. Lightness in digital imaging is obtained by double or triple equal variations of the colors, meaning that if two or three of the RGB channels remain equal as shown in *figure 2.6*, the colors begin to fade, creating the luminosity effect.

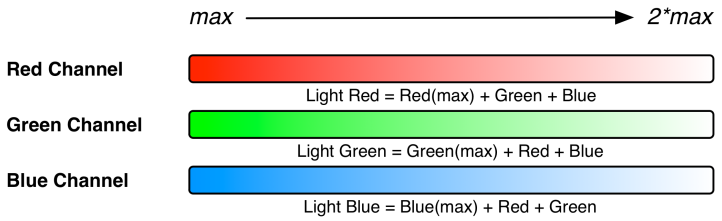

Figure 2.6 - The RGB lightness variation

The RGB standard is not applicable in real life color theories because combining other colors cannot reproduce the non-colors, black and white. The explanation why this works in digital imaging is the following:

- Digital screens on which the colors are displayed are black (obscure)
- The three colors are reproduced by colored components (for example LED's or CRT dotted matrix) that can achieve a maximum lightness that defines the actual color. This means that if the three-color components have the maximum amount of lightness, the perceived color is white.

Each color has two intensities: none and pure. Every variation between these two boundaries is a variation from dark color tones. The variation of light color tones is obtained with the help of the other two color components. If the color value remains the same for each of the three colors, the result will be a black and white variation. This is possible because, the colors auto-negate each other if they have the same intensity. If one or more of these values differ, it will result in a color. In *figure 2.7* we can see that equally distributed values for RGB generate the non-colors (black and white are considered non-colors).

Figure 2.7 - Non-colors (grayscale) using RGB triplet

The *numeric representation* of the RGB triplet is in fact a two-dimensional matrix. The Y-axis represents luminosity (varying from dark to light) and the X-axis represents all possible color combinations between the red, green and blue channel.

Another approach in representing the RGB triple is the *geometrical representation.* Because of the RGB structure, we can consider each component of the triplet as being a coordinate in a 3D space. The three axes are replaced by the values of red, green and blue channels. The RGB triplet, viewed as Cartesian coordinates in a Euclidian space determine a cube where the origin of the axis represent the absolute black point, while its diagonal opposite determine the absolute white point of the graphic. Each corner of the cube that is situated on one of the axis represents a pure color.

The *geometrical representation* of the RGB triplet is very laborious to be used in digital imaging, but it was designed because it is the easiest way to determine the similarity between two colors. This is often used in algorithms that do digital processing/analysis on the image. Given two colors, the similarity between them is equal to the distance between the two points in the color cube. The color palette of the *geometrical representation* is hard to observe because of the third coordinate and depends on what side we observe the cube. In *figure 2.8* the geometrical cube is observed from the absolute white point side.

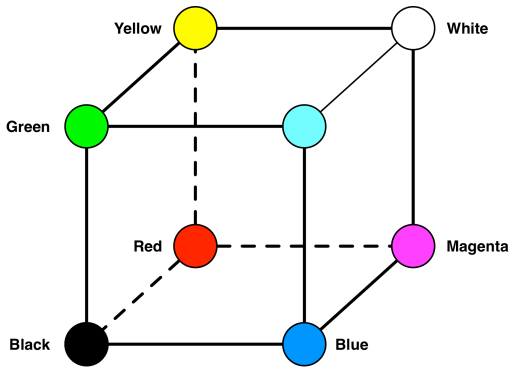

Figure 2.8 - RGB Color cube

The *digital representation* is strictly related to the *numeric representation* because of its design simplicity. Like in the numeric representation the intensity of each color can be represented in multiple forms of data structures as shown in *table 2.1*.

Intensity representation		RGB triplet interval	Example
Arithmetic		Floating point interval	(1.0, 0.5, 0.0)
Percentage		0..100%	(100%, 50%, 0%)
Digital	8 bit	0..255	(255, 127, 0)
	16 bit	0..65535	(65535, 32767, 0)

Table 2.1 - Digital color representation

Digital imaging uses the RGB model differently, based on the size of the color palette. The size of the color palette depends on the level of color detail needed for the specific image and therefore, it can be divided into:

- Monochrome
- Grayscale
- High Color
- True Color
- Deep Color

The *monochrome* color palette is the simplest and smallest color palette, it contains two values: black and white. In digital terms, in the monochrome color depth, each pixel represents one byte, where 0 stands for black and 1 stands for white. In this representation, one image byte can store 8 pixels.

The *grayscale* color palette is used for representing high quality, colorless black and white images. Each pixel represents one byte of information with a value range situated between 0 and 255.

The *high color* palette is the first palette that displays color variations, each channel having 5-6 bits. This means that each pixel is represented on two bytes. There are two high color modes:

- 15-bit high color (32768 colors)
- 16-bit high color (65536 colors)

In the 16-bit high color representation one of the three channels gets an extra bit, although the difference between 15 and 16 bit high color is almost inexistent for the human perception.

The most commonly used color palette is the *true color* palette. Every pixel is 3 bytes long (24 bits), one byte corresponding to each color channel. Every channel has 256 color variations, totalizing 16,7 million colors (for all three channels). The true color is used in digital imaging, photography and video recording as it can store all the colors the human eye can perceive.

In real life there are more than 16,7 million colors, so there is need to store more color information in certain circumstances. Therefore, the *deep color* comes with 30, 36 and 48 bits per pixel (10, 12 and 16 bits per color channel), sufficiently enough to store 281,5 trillion colors.

2.2.2 The RGBA representation

The RGB model is sufficient to store color information, but since the first graphical editing tools came out, one big problem was the overlapping of two images. This has not been a problem since the beginning, but as the graphical user interfaces evolved, so did the need to combine two colors of different layers into a new resulting color.

The RGBA model offers the solution to this problem by completing the RGB model with "color information" called the *alpha channel.* The alpha channel is just like every usual pixel color, with the difference that it provides transparency instead of color information. We can observe certain similarity to the high color RGB model. Each pixel in the RGBA model has its own transparency; this means that certain pixels can have different transparency values. *Figure 2.9* illustrates two images overlapping, out of which, the second has alpha transparency of 127 for all pixels. The resulting image is a combination between the two original images.

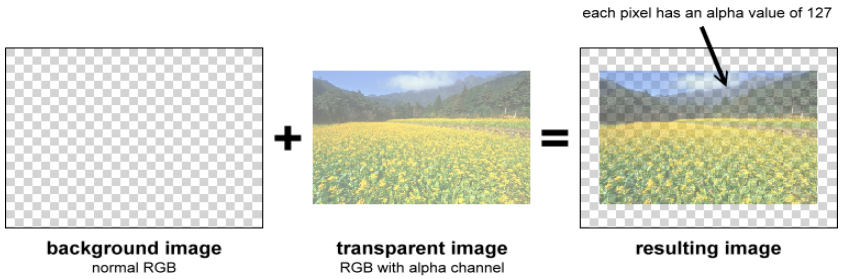

Figure 2.9 – Image with alpha channel = 127 (50% opacity)

The alpha channel is used mostly to make an image blend in with the background in cases in which the background itself is not fixed defined (variable). *Figure 2.10* illustrates the fade-out effect of an image if the alpha channel equals the y image coordinate. The effect resulting is that the top of the image presents maximum opacity whereas the base of the image has maximum transparency (invisible).

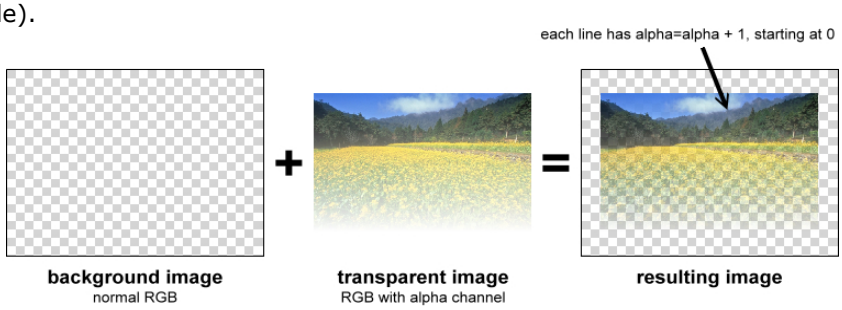

Figure 2.10 - Image with alpha channel = y (0% to 100% opacity)

2.2.3 Image compression

Image compression represents a fundamental aspect when it comes to shrink the color table's size. There are two major types of compression algorithms that deal especially with color information:

- Lossless
- Lossy

Lossless compression algorithms decompress the original data 1:1 without any loss of information. Typically these techniques are used to reduce the size of the image, but in the same time retain the possibility of accessing the original color information with no loss. Formats like BMP, PNG and JPEG HD use lossless algorithms to preserve the integrity of the original pixel table. Lossless image compression is a class of compression algorithms that allow the exact reconstruction of the original data (in this case color information) from the compressed data. This is useful to preserve the color information whilst keeping the size under reasonable boundaries. The compression algorithms use pattern detection techniques and/or arithmetic coding to achieve the best compression possible. The compression size depends mostly on the pixel color information, but also on the dictionary size, specified in percentage. The dictionary size represents how many compression iterations the algorithm will execute before the data is completely compressed. The more iterations it executes, the smaller the compressed data can become (that's not always true) but in the same time, the decompression speed decreases.

Lossy compression algorithms on the other hand, try to eliminate the overhead color information based on the fact that the human eye cannot distinguish all possible color variations a typical pixel can store. Formats like JPEG use a set of transformation tables to dramatically reduce the size of the image, but maintaining the visual integrity of the original image. The decompression process approximates the original pixel combinations, but it is unable to output a 1:1 resemblance to the original even when the compression quality is set to 100%. Lossy image

compression algorithms only allow an approximation-based reconstruction of the original data. The decompressed data is different from the original one (it depends on the quality percentage used when compressing) but closely enough to be useful in some kind. Lossy compression is used when the color preservation doesn't matter to be taken into consideration, and the size is very important. The compression ratio compared to lossless compression is way better, but it comes with a certain cost: quality. *Figure 2.11* illustrates a few of the most popular image standards and their compression capabilities.

Image format		Compression type		
Extension	Name	Lossless	Lossy	None
ABO	Adaptive Binary Optimization	✓	×	×
BMP	Bitmap	✓	×	✓
CPC	Cartesian Perceptual Compression	×	✓	×
GIF	Graphics Interchange Format	✓	×	×
HAM	Hold And Modify	×	✓	×
JBIG	JPEG BIG	✓	✓	×
JPEG	JPEG	×	✓	×
JPEG XR	HD Photo	✓	×	×
JPEG-2000	JPEG 2000	✓	✓	×
PGF	Progressive Graphics File	×	✓	×
PNG	Portable Network Graphics	✓	×	✓
TIFF	Tagged Image File Format	✓	✓	×

Figure 2.11 - Most commonly used image formats

In terms of steganography, *lossless image compression* is the only way to hide data and in the same time achieve high steganographic capacity.

2.2.4 Pixel matrix

Digital images are shown as a matrix of pixels. In reality, the pixels are arranged as a stream of bytes that form an array. The digital image has three components:

- Image information header
- Color pallet (optional)
- Pixel stream

The *image information header* contains information like width, height, bits per pixel (bpp), etc. The *color pallet* is an optional image component, and it is used for image formats in which the colors are limited to a reduced set of colors. This is used in order to reduce the channel representation bits to a smaller number, in order to decrease the image size.

The pixel stream is the part of the image where the color information is stored. The image pixels are displayed as a matrix, although they are stored as an array. In order to address the pixels as a matrix certain formulas are used:

$$Pixel_{matrix}(line, col) = Pixel_{array}(line * width + col)$$

where

$$line = 0 \,..\, height - 1$$

$$col = 0 \,..\, width - 1$$

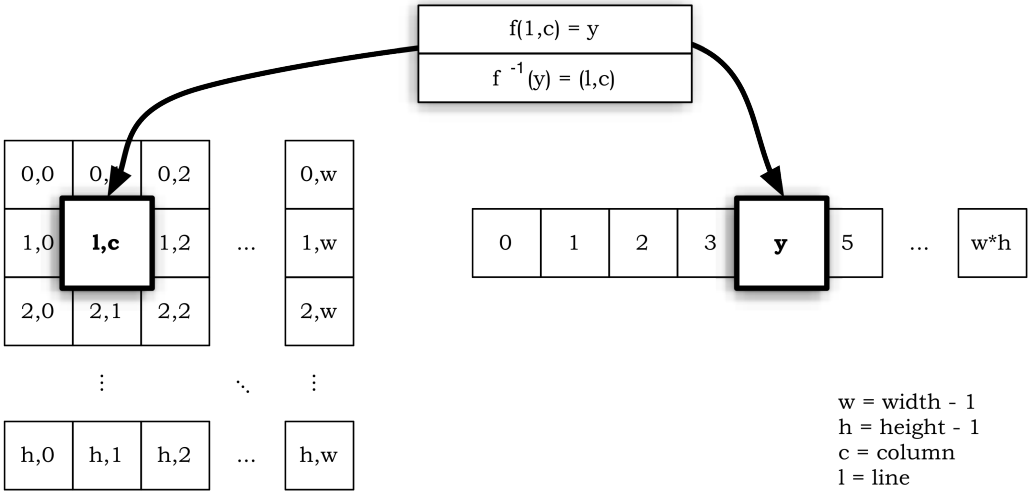

Figure 2.12 - Matrix - array transformation

As one can observe in *figure 2.12,* there is a simple formula for calculating the coordinates from the matrix to the array and vice versa. The pixel-by-pixel addressing is a very time and resource consuming process; this is why normally the image is red line-by-line using a *scanline* function. Scanline uses a buffer of size $width$ to read the exact amount of information corresponding to a line of image. In this way, data is red from the file into the destination matrix (usually situated in RAM for better access timings).

For a performance evaluation, I use the notation τ, representing the time it needed to fetch a pixel. Fetching the entire pixel information of the image would require $speed = width \cdot height \cdot \tau$, which is a very high delay in accessing speed. This is caused by the computational intensity of the seek timings (used for every pixel – fine granularity) and it mostly depends on the hardware speed and implementation. Therefore, alternatives like bigger buffers are taken into consideration. The buffers used to read data are then up scaled to read more information (pixels) in one single read cycle (coarse granularity). This process is also known in digital image processing as *scanline* reading. Instead of reading the information one pixel at a time, we read an entire line of length=width. In scanline the read time is dramatically reduced to $speed = height * \tau'$, where $\tau \leq \tau' << width * \tau$.

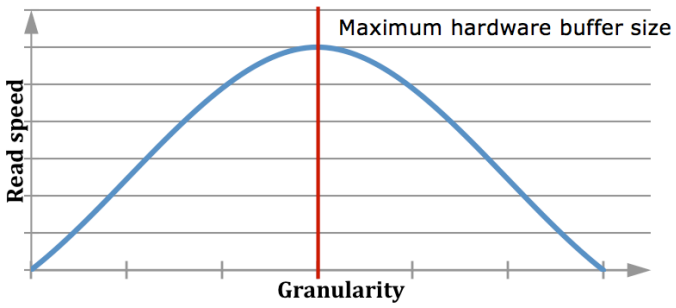

Figure 2.13 - Fine granularity vs. coarse granularity (read buffer speed)

In *figure 2.13* one can observe that there is a dependency between the read buffer size and the read speed. Depending on the hardware implementation, there is a maximum buffer size that can be read in one read cycle. The absolute speed is obtained by scaling the scanline buffer to the maximum size given by the hardware capability. This is difficult to realize because normally such data is hardware dependent, so the only alternative is to determine the maximum buffer size with progressive benchmarks. Once the pixel information is stored in RAM, the access to individual pixel color channels is more easily realized (better access speeds).

2.2.5 Color Histogram

A *histogram* by definition is a frequency table that counts the occurrences of specific values. In digital imaging, a *color histogram* is a tonal distribution table that counts the individual values of each color channels. A color histogram is used normally for image processing, as it contains valuable information over the color distribution. Most image effects and correction algorithms are performed on the color histogram.

Normally, the color information inside the pixels of a digital image can be represented into three base histograms (Red, Green and Blue histogram), but there are additional histograms that can be realized to represent color information (RGB median, Color, Luminosity, Saturation, Median, etc.).

The X-axis represents the number of colors of the specific channel (for example in a 24 bpp image, the maximum color number is 256), and the Y-axis represents the frequency (the number of occurrences of the specific color value).

The color histogram can be considered as a fingerprint for the image, as it counts its color components. Besides the digital image processing, steganalysis techniques intensively use histograms to identify steganographic images.

If we take the RGBA model, the four basic histograms (Red, Blue, Green and Alpha histogram) are generated using the following code:

```
1   ZeroArray(RedHist)
2   ZeroArray(GreenHist)
3   ZeroArray(BlueHist)
4   ZeroArray(AlphaHist)
5   for y = 1 to image.height do
6     for x = 1 to image.width do
7       begin
```

```
8         Increment(RedHist[pixels[y,x].red])
9         Increment(GreenHist[pixels[y,x].green])
10        Increment(BlueHist[pixels[y,x].blue])
11        Increment(AlphaHist[pixels[y,x].alpha])
12      end
```

The first step is to reset the peak values of each color to zero (*lines 1-4*), then the image (*lines 5-6* - in this case we use a traditional matrix scanning). For each pixel we increment the histogram peak with the index given by the color component's value (*lines 8-11*). At the end of the code sequence we get an array with the color distribution table (frequency) that we can either use for digital processing or for representation. Usually histograms are used for image processing as they can be analyzed to identify faults in image representation.

2.3 Digital Image Steganography

In digital steganography, probably the most elaborated and intensely researched domain is represented by image-based steganography. Images facilitate advanced embedding methods due to the variety of formats and digital color representation standards. Image steganography represents the intersection of traditional steganography with digital image processing techniques. The main reason why this specific domain presents more interest to researchers worldwide is because most steganographic techniques are used to transport a very small amount of secret data that is normally big enough to fit in a normal average-sized image. Other than that, the size of the transmitted carrier blinds in with the rest of normal harmless image sharing procedures, making the cover-up for the secret communication channel the ideal storage and hiding place.

2.3.1 Images as data carriers

Depending on the application of steganography in security systems, the choice of what carrier type to use is a central task. For my research I've focused on working with images because of multiple factors such as:

- Widely spread throughout the internet
- Reduced size
- Overhead information
- High compression

Images, after text, represent one of the most widely used formats throughout the Internet. The majority of websites, email attachments and documents contain images, making them the perfect carrier in almost every aspect. Unlike audio or video streams, images offer limited storage space for the secret data meaning that they are unsuitable for live communication systems that implies a constant transmission stream such as audio or video. However, if we take into consideration a recent study [34] based on the file exchanges conducted in the last

three years through either file sharing systems or emails, it is immediately obvious that the majority of formats exchanged by most people imply digital documents and images. The study was conducted on emails exchanged solely in the Gmail email system and showed that out of 10 million emails sent, 12% have attachments alongside text. The majority of emails with attachments are sized under 250 KB (75%) and contain mostly images (29% - PNG, JPG, BMP) and documents (25% - PDF, DOC), making the image format the most utilized type of attachment throughout the internet, beside text. Since by definition steganography implies hiding in plain sight, images make the perfect data carrier.

The majority of images are small in size, making them easy to transmit over the Internet. This is another aspect for which images are more suitable over other data carriers. Also, unlike other formats, the pixel table of a normal high-quality image presents more overhead data than other formats that are compressed using lossy compression. Except for formats like JPEG, high quality image formats often use lossy compression, meaning that the decompression recovers the original color information 1:1 without any loss.

In most cases the sensitive information one wishes to secure against a third party listener or interceptor, is represented either by a document, message or image. Given the size of this average data, images offer a decent amount of storage space and are perfect for this purpose.

2.3.2 Lossless or lossy coded images

Just like every other media format, images can use multiple compression standards. As outlined in a previous chapter, compression algorithms are divided into two major categories lossless and lossy. Comparing these two techniques highlights that the fundamental starting point of steganography is represented by the overhead of digital information that is not perceived. The main difference between these two compression techniques is the size of the resulting compressed image. If the color quality is not a big issue, lossy compression techniques is definitely the best choice since it can compress the image up to a very small fraction of the original one. Lossless compressions offer a decent amount of size reduction, but compared to the lossly encoded images are still substantial in size. Image steganography is split into these two directions. The reduction of overhead color information in lossy encoded images eliminates the majority of storage space one has for the secret data, making this format suitable only for very small files or short text messages. If one wishes to send a document, lossless encoded images are the wisest choice.

The two major image formats used today are either JPEG (lossy encoded) or PNG (lossless encoded). A 100% quality encoded JPEG is comparable to PNG in image size making PNG one of the widely used formats for image steganography. In my research I've focused entirely on lossless encoded formats such as PNG for a multitude of reasons out of which I can mention:

- High storage capacity
- Easy to manipulate
- Small size when compressed
- Higher processing speeds

2.3.3 *The paradigm of steganography*

The first fundamental problem when facing with steganography is within its own definition: steganography must meet both quality and capacity requirements. Given the fact that we use an existing format, the highest quality form of information is represented by the original data carrier. Once we tamper with its integrity by including the secret information within the original carrier we slowly decrease the quality in order to make room for the secret data. When taking into consideration that we use a carrier that was not designed to have secret data embedded onto it, steganography must keep a controlled balance between these two variables. The higher the resulting image quality the lower the embedding capacity, and vice-versa, represents the main problem and the central breakpoint. The struggle to keep image quality and embedding capacity under acceptable levels implies the acceptance of certain compromises. These however, are not easy to achieve as balancing these two offers an output that is not satisfactory in both directions.

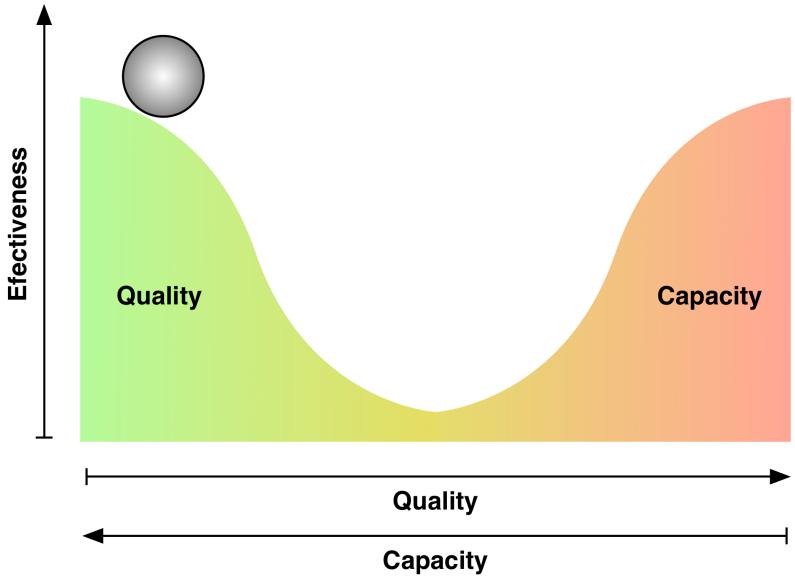

Figure 2.14 – The steganographic paradigm illustrated

Quality and embedding capacity represent the two main constraints of every algorithm, it is hard to meet both at the same time as shown in *Figure 2.14*. The effectiveness of a steganographic algorithm is either on the side of quality, case in which we consider an algorithm to be effective if it retains more quality of the original carrier, or on the side of capacity where as an algorithm is considered to be effective if it can offer an increased storage space for secret data. The hypothetical ideal case represents the algorithm that is considered effective if both increased quality and capacity requirements are met. This is however near to impossible to achieve given that they can almost never meet in real life situations.

Alternative solutions have been researched in the last decade, but most fail to achieve a decent amount of quality and capacity as it always implies a

compromise. The inter-dependency of quality and capacity is near to impossible to break due to the limitation of using the pixel table of the original carrier image.

2.3.4 LSB steganography

One of the most commonly used methods in lossless image steganography is represented by the LSB method. A typical high color digital image is constructed based on an array of pixels. Each pixel stores the color as a combination of the three computerized primary color channels:

- Red (R)
- Green (G)
- Blue (B)

A typical 24-bit high color image stores each of the primary color channels on one byte, totalizing over 16 million possible colors when viewed as a whole. Although from a digital perspective these colors are unique, from a visual perspective (human vision) many of these colors are redundant and repetitive. The average human eye can only distinguish between 2 million out of the total colors possible. If we analyze the relationship between the three color component from a binary perspective we see that the last bit of every color component affects the color spectrum that goes unnoticed if tampered with, from a human perspective:

$$SizeOf(RGB) = 2^8 * 2^8 * 2^8 = 16.777.216 \quad (2.1)$$

If we where to take this relationship and ignore the color combinations introduced by the last bit of each color component we would obtain:

$$SizeOf(RGB) = 2^7 * 2^7 * 2^7 = 2.097.152 \quad (2.2)$$

This discovery led to the conclusion that, if not needed, the last bit of every color channel, totalizing 3 bits/pixel can be reused in order to store other data within. The image is viewed as an unformatted disk where the last bits (LSBs) become volatile and serve for storage and the first bits (MSBs) remain nonvolatile in order to preserve the original aspect and cover the actual secret data within.

Figure 2.15 best illustrates a typical 1 LSB based steganographic algorithm. The technique of hiding within the last bits of each pixel carries the name of 1 bit LSB steganography. From the storage point of view if we where to take a typical 1024x768 image as an example we would see that it contains 768432 pixels. These pixels, multiplied my 3 give the total amount of color channels the image is represented upon, 2359296. Since the above example is using 1 LSB steganography as reference, this number is equivalent to the total number of bits available for storage. In this case, the image would be able to store 294912 bytes (288 KB), which in most cases would be sufficient for a small document or even another image to be included within the original carrier. The storage space in the outlined example represents 12,5% of the total image's size, which represents the major downside of this technique: the need of substantially big images. Although this technique may

seem pretty simple and straightforward research on this basis has not ended here. In order for this steganographic technique to be effective in real life situations and usage, the size of the carrier image must also be relatively small in order to avoid detection.

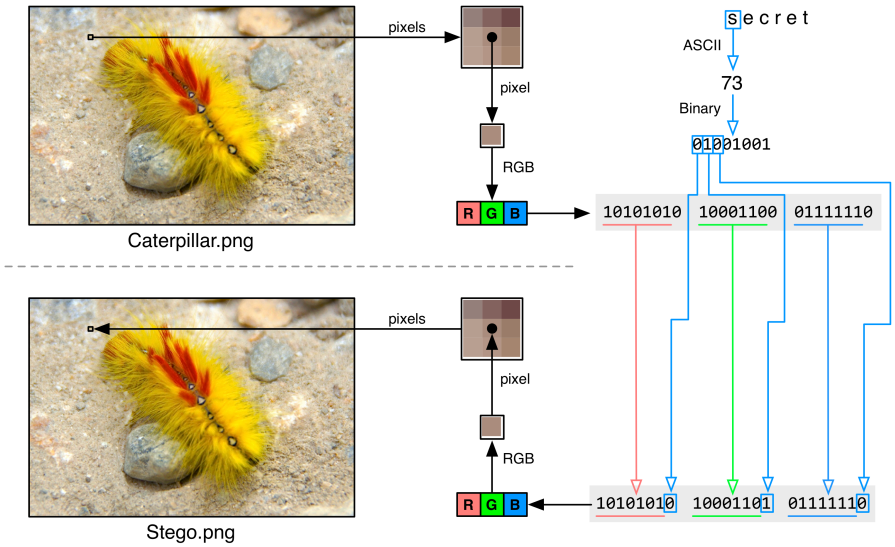

Figure 2.15 – 1 LSB-based steganographic algorithm example

This is one of the main reason researchers have been trying to go beyond the 1 LSB limitation and use more bits for data storage. The major problem that arises with these new techniques derives from the fact that tampering the other bits directly affects the color spectrum of the image that can be humanly perceived and identified.

The construction of an algorithm that alters more information, therefore utilizing smaller image to achieve the same storage space, requires more complex image processing and normalization algorithms in order to reestablish the original look and feel of the carrier image. This has been proven not to be an easy task, as one can easily meet technology's limitation.

The computerized representation of color described as an RGB triplet is not directly related to human vision and perception. Although it may seem as if the three color components red, green and blue influence the resulting color in an equal manner, in reality colors behave different depending on the image context and surrounding color pallet.

CHAPTER 3

Related work

> "We should be taught not to wait for inspiration to start a thing. Action always generates inspiration. Inspiration seldom generates action." – Frank Tibolt

> "Not everything that can be counted counts, and not everything that counts can be counted" – Albert Einstein

When it comes to build a high capacity steganographic algorithm one of the most commonly used methods is represented by the LSB technique. Based on the fact that a normal natural image presents many overhead data that store unperceivable color variations one can embed the secret data within the noisy regions, replacing random white noise with computerized white noise. Since the choice of carriers for this type of image steganography is mostly limited to using lossless algorithms, the size of the carrier image often presents the main problem of such an implementation. The outputted steganographic image is normally substantial in size directly related to the size of the secret data we wish to embed. In most cases, if we wish to embed data such as text, the lossless format does not present a problem. If on the other hand we wish to secretly embed data such as images, documents and other type of digital file formats, the need for increased storage space often requires a bigger image to hold the data within. The classical 1 LSB method offers a decent amount of space but fails to offer the adequate storage space for larger files. If we take a 24-bit image where each of the primary color channels R, G and B are represented on 8 bits, by embedding 1 secret data bit in each of these three components would mean that the image offers 12.5% embedding capacity of it's total pixel table size. In other words, to store secret data using this method would require an image that is 800% bigger than the size of the secret data we wish to embed. If we take a typical PDF file as a secret data, the average size would be around 100-400 KB depending on the content; the 1 LSB method would need an image that is 800-3600 KB in size which adds much information that would render the entire steganographic process useless in terms of stealth because of the high quantities of cover data needed to hide the actual secret information.

Recent advances in steganography go beyond the point of 1 LSB steganography, by utilizing more color LSBs in order to embed the secret information. By crossing this barrier, researchers also face big problems that arise by altering the more sensible color variations. The field of high capacity steganographic algorithms represents a very vast and complex research domain

that is directly related to image processing techniques, as one must restore the original look and feel of the original carrier image after coarsely embedding high quantities of information within the more sensible color regions. In addition, the problem of enhancing the image through advanced processing and normalization filters, the higher the embedding capacity a typical algorithm offers he higher the chance of detection using steganalysis techniques becomes.

The need for more storage space pushed steganographic research into a new level, creating in the same time a new complex research domain. Steganalysis research also evolved with the new trend, pushing the limits and constraints needed to take into consideration when conceiving a new steganographic algorithm. Just like the embedding algorithms themselves, steganalysis algorithms rely on a set of metrics that estimate statistically the chance of secret data presence. The steganalysis algorithms are split into multiple categories based on the metrics and methodologies they use to reliably estimate the chance of secret data existence inside the carrier analyzed. One of the most commonly used detection algorithms use a set of evaluators such as:

- Quality evaluators
- Embedded capacity calculators
- Histogram anomalies detectors
- Logical or PRPG induced noise detectors
- LSB scattering detectors
- Color anomalies detectors and evaluators
- Peak-to-Signal Noise Ratio evaluators

Trying to avoid detection by respecting all the commonly used detection algorithms can be a very hard task to achieve given their complexity and metric scales in which they quantify the result. There are two emerging strategies that researchers commonly use: avoiding as much as possible detection at the cost of losing storage space or adapting the image result in a way that issues the highest suspicion rate when passed through an analysis stage.

3.1 Steganalysis detection overload algorithms

The main starting point of steganography is to issue a data carrier that travels undetected through these types of detection filters. Given this fact, algorithms that overload these detection strategies to issue maximum suspicion rate fail to fulfill the main purpose of steganography as the idea that a constant alert in detection systems may let all carriers that were processed using this type of steganography pass unnoticed. In reality they form a separate type of images that are always suspicious of containing secret information within when compared to normal harmless images that are sent without the intention of using them as data

carriers. In a recent work presented by Shreelekshmi in [35] a new technique of cover preprocessing based steganographic algorithm has been presented that trick the most reliable steganalysis methods into estimating a high embedding ratio of almost 100%, in order to issue a false positive. In reality this technique may be effective in the event it does not get used very often in these types of embedding strategies, but as the algorithm becomes more known and utilized, the constant false positive the steganalysis detection filter issues can be threated as the exact opposite, well distinguishing the technique among others.

Since a typical natural image contains much noise in the LSBs of each pixel, many researchers have tried to build algorithms that insert random noise in order to trick a few of the most commonly used steganalysis techniques into issuing false positives or very high chances of secret data existence. Normally a steganalysis algorithm that issues the chance of secret data existence has a scale defined by a upper and a lower boundary, but the reliability of the estimated result is higher in a region called the confidence space. Usually overload steganographic algorithms insert much noise in order to keep the steganalysis results out of the confidence space, making their estimation inaccurate in theory.

In reality however the things look slightly different, since a constant set of carriers that pass these detection stages issuing a result out of the confidence space, becomes suspicious by definition and can be easily filtered.

3.2 Steganalysis avoiding algorithms

Probably the most intensively researched type of steganographic algorithms is represented by the steganalysis avoiding algorithms. These algorithms take the steganalysis metrics into consideration when deciding where and how to embed the secret data. There are multiple ways in which these tasks can be accomplished such as:

- Preprocessing the image
- Choosing embedding regions
- Calculating total reliable storage space
- Degrading the image prior to embedding
- Noise elimination
- Controlled insertion
- Processing the image while embedding
- Error correction during and after embedding
- Postprocessing the image
- Controlled insertion using histogram preservation

This is just a small list of the main categories of algorithm design guidelines many researchers follow in order to achieve maximum performance in terms of stealth and resistance to detection. The main problem that I've found throughout my research is that most algorithms tend to avoid detection only for a very limited amount of steganalysis techniques.

3.2.1 Preprocessing the image

A very vast category of steganographic algorithms relies on preprocessing the cover image prior to embedding. This stage usually smoothens the image and eliminates the noise prior to embedding secret data. These types of algorithms basically eliminate the natural noise of the image and transform it into a computer-enhanced image. The embedding process is basically a normalization process that introduces the initially stripped noise. However, the output is not always the one that we wish for, since the logical noise introduced can alter portions of the image up to the point where the steganographic image looses its original aspect after embedding.

3.2.2 Choosing embedding regions

The technique of choosing the embedding regions is proven more effective than preprocessing based algorithms since it initially evaluates the image from a steganalysis point of view in order to find the most noisy regions of the image or those regions that are blind to most filtering algorithms. Depending on the cover image chosen, this type of steganographic algorithms have a high failure rate when it comes to capacity since the strength of the method comes at the cost of losing valuable embedding space. In some cases, where the image color distribution is perfect, such algorithms fail to find suitable embedding regions, or find the entire image as suitable for embedding. These extremities, although isolated, don't offer the adequate amount of success guarantee that is to be expected from a powerful steganographic algorithm.

3.2.3 Calculating total reliable storage space

Exactly like embedding region detection algorithms, these algorithms calculate the reliable storage space before the embedding process can begin. Just as in the case of region detectors, these algorithms offer a limited storage space, as they tend to estimate the capacity using pessimistic approaches in order to guarantee the stealth level. By comparing this type of algorithms with the original 1 LSB algorithm, in may cases the reliable storage space estimated is even smaller that the initial design, rendering the algorithms useless for practical usage.

3.2.4 Degrading the image prior to embedding and noise elimination

Many algorithms use the LSB [56] insertion method after an intitial cover preparation stage. This stage is mainly used in order to rearrange the initial original color information up to the point where it would become suspicious for a detection filter. The embedding stage is then used as a normalization stage where the initially degraded cover is altered tending to resemble the original. In reality however these algorithms do not offer any type of guarantee that the theory is always found within the real life examples. Without testing the effect of the resulting steganographic

image, this type of algorithms prove weak even against a statistical evaluation of the method, making them vulnerable to most common types of attacks.

3.2.5 Controlled insertion

Controlled LSB insertion algorithms prove to be more effective than other types of steganographic algorithms because they alter the original information in a controlled way with a set of quality metrics as guideline. These algorithms try to eliminate the anomalies that are normally introduced, in the incipient phase, without the need of future corrections to be applied. These algorithms are very powerful to steganalysis attacks but become less effective if we want to increase the storage space, as their correction stage is limited to a narrow set of reusable data.

3.2.6 Processing the image while embedding

Just like in controlled insertion steganography, many algorithms rely on image enhancement filters during the embedding process. Depending on the filter used, once a limited set of pixels get altered after inserting the secret data onto them, these algorithms then apply a set of enhancement filters in order to restore the normal color trend line. Although effective in most cases, these algorithms also suffer from their limited possibility to enhance the image when reusing to much color information for embedding purposes.

3.2.7 Error correction during and after embedding

The error correcting algorithms use additional color information bits to restore the original look and feel of a specific image region immediately after the secret data bits have been inserted. These types of algorithms are able to offer an increased embedding capacity compared to other real-time enhancement algorithms. They're breakpoint however resembles in their vulnerability against signal difference attacks or even histogram attacks since the usage of more sensitive information for color correction usually alters the image coarsely up to the point where, in some cases, the abnormalities introduced are observable by a human analysis.

3.2.8 Postprocessing the image

Algorithms that rely on postprocessing prove to be more effective against most steganalysis detectors as they have the full overview on the degree of change the original cover sustained after embedding. Using differential filters and having both original on modified image, these algorithms try to restore the original color information or histogram distribution of colors according to the model, which is the original input image. Depending on the quantities of secret data embedded onto the original image, the postprocessing stage can be rendered useless in some cases in which the changes would imply heavy processing and enhancement that would affect the sensitive portion of the image which is represented by the secret data itself. In these cases the filters aren't even applied leaving the algorithm with no choice but to output a highly degraded steganographic image.

3.2.9 Controlled insertion using histogram preservation

Since many steganalysis algorithms rely on histogram evaluation, this type of steganographic algorithms try to control the insertion by keeping the histogram of

the modified image in balance with the original. In this way the image maintains its original look throughout the embedding process without the need of additional filtering and image enhancement stages. The histogram attack is one of the simplest types of steganalysis detection algorithms, but it has been proven repeated times that some natural images have a non-uniform color histogram, leading this type of attack to issue a false positive on a natural image with no secret data contained within.

3.3 LSB Matching steganography

In most recent years, researchers have been concentrating on a new way to avoid most detection algorithms but in the same time retain more of the original cover medium's color information. This technique is known as LSB matching steganography and it's based on the fact that the image we use as cover already contains many of the binary combinations that can be found within the secret data we wish to embed, just not in the right order. As opposed to traditional steganographic methods, this new trend tries to focus on the cover image preservation aspect rather than coarsely modifying it in order to store secret data. These algorithms use more computational power during the embedding process, but have the advantage of not requiring postprocessing on the resulting image. Regardless of the approach, the secret data will degrade the image; therefore the main purpose is to minimize the degradation, and at the same time keep various quality metrics such as peak noise to signal ratio (PSNR) within acceptable intervals. Higher PSNR values improve the overall immunity against statistical analysis. By minimizing color degradation the hazardous distribution of colors is preserved, thus making secret data harder to detect.

Chang [26] proposed in 2009 a hybrid method that combines two methods: the optimal LSB substitution [39] and the optimal pixel adjustment [38]. The hybrid combines a remapping algorithm with a pixel value adjustment technique. The optimal LSB (OLSB) [39] method rearranges the secret data according to a bijective mapping function. The image is scanned sequentially into an analysis matrix and a set of solutions is generated. The best solution is chosen based on a cost function that selects the proper combination of secret data bits that produces the minimum amount of distortion after embedding. The optimal pixel adjustment process (OPAP) [38] minimizes the difference between the original LSBs and the secret data. Given a number of k bits that are used for storage, the method uses another bit to control the difference. For example, if we use the last 3 bits of each pixel, we assume the original pixel value is 00001000 and the secret data bits that needed to be embedded are a sequence of 111. The simple substitution would transform the original pixel in 0001111 that would mean an error of +7, which is an unacceptable change. Instead, we change the leading bit, transforming the original pixel's value in 00000111, which means an error of -1, which is acceptable.

In [37] a new optimization strategy has been presented that uses genetic algorithms to find the best mapping function between cover image and secret data, therefor decreasing the overall degradation. This approach however, is computational intensive and cannot be completed in polynomial time when the number of iterations is big. Also there is no guarantee that the technique generates better results when the number of iterations is small or in acceptable timing intervals.

CHAPTER 4

Advancing Steganography

> "Besides the noble art of getting things done, there is the noble art of leaving thing undone. The wisdom of life consists in elimination of non-essentials." – Lin Yutang

> "Digital imaging has untied our hands with regards to technical limitations. We no longer have to be arbiters of technology; we get to participate in the interpretation of technology into creative content." – John Dykstra

Since my research started, I've been constantly searching for new ways to construct and build steganographic algorithms. Throughout my research over the years I've discovered that traditional LSB steganography is faulty by concept. The coarse data replacement followed by computational intensive filtering and image enhancement stages were definitely not the right path, as my research, just like the research of many others in the field, often led to a dead ends. The multitude of steganographic algorithms developed covers a large set of problems and aspects, but rarely in a large basis. Some problems get solved; some are left unsolved, dividing the entire research field in local optimums of solutions to chose from. The quality-capacity constraint as well as the steganographic paradigm of effectiveness is usually a hard task to resolve when taking the technologic limitation into account.

In LSB steganography, once we go beyond the 1 LSB method we alter the color space that can be identified even by human vision. Since the average human eye is limited in distinguishing between 2 million color combinations, going even further with the storage space to 2-4 LSBs/color channel modifies the visual spectrum up to the point where it renders the stealth level of the steganographic algorithm useless. Unlike watermarking, high capacity steganography is not resistant to changes; even the slightest change to the image can destroy the secret data contained within. This is one of the biggest problems when trying to optimize the resulting steganographic image using postprocessing filters. Most of these filters were not designed to preserve digital combinations to a binary level, instead they enhance the perceptive integrity of the output.

For my research work I've set my goal high: trying to resolve the problem by shifting away from the normal workflow. One essential thing I learned throughout my research is that if I try to optimize the image after the secret data embedding took place, I position my research more in the field of image processing than in steganography, were as the main idea behind steganography relies in the ingenuity with which one finds ways of storing data with a minimum of effort. Since my work

is based on LSB steganography I found out that instead of ignoring the LSBs we want to reuse and replace, the best way to avoid major quality loss is to embrace the existing combinations. Although these LSBs influence a small amount of color information, they still are the building stone of the original unchanged image, forming the maximum quality possible.

4.1 Natural images

Digital images can be divided into two main categories: natural and computer generated images. Computer generated images are usually represented by a logical color distribution and trending shapes. Natural images represent images taken using a digital camera or any other digitalization device (e.g. scanner). In real life, light is never uniform and constant, but rather random and noisy. As humans, we perceive light as a constant and uniforms signal, but in reality things look slightly different. Just like any other physical factor, light is basically a signal that varies in intensity, shape, size and duration. The reason why we see light as being static rather than dynamic has a multitude of causes:

- Light frequency
- Limited color spectrum
- Approximation mechanisms
- Light in context

Light frequency is one of the key factors why human vision is faulty. We see the surroundings at a very low frequency compared to the actual light frequency. Depending on the light source, we normally perceive an overlapping set of frames that added together form an average amount of light we perceive. The stacking process of multiple frames makes the light variations and transitions smooth, as we only measure the average amount of light accumulated between a large amount of unconsciously perceived frames. If we were to capture an image with a high speed device or camera, we can see that light varies in intensity over time, but at a very high speed, going almost every time unobserved by our faulty eye.

Other than light itself and the limited color spectrum we perceive is also one of the reasons light may seem constant for us. If we assume that light intensity is constant, colors outside of our visual spectrum may look exactly like other colors contained within the visual spectrum, making them identical. In reality we are talking about either about small color variations near enough to look identical, or even variations outside of the visual spectrum that correspond to others. Visual decomposition implies a two-layered process. In one process we perceive light as a signal with a minimum and maximum intensity. Colors represent a mask that influences this intensity, transposing the linear 2D signal into a 3D signal.

Human vision also acts like a digitalization algorithm. As above stated, fine-grained colors look the same, meaning that we see these colors by approximating them to the nearest perceivable color. This does not necessarily apply to all, for example painters, photographers and artists can have a more sensitive vision that is

capable of reducing the approximation, therefor visualizing a increased set of perceived color variations.

The last and probably most important cause why we see light constant is based on the light in context principle. Different mixtures of colors can make the appearance of the whole seem like a completely different color. A white shaded circle placed over a black background seems less intensive than the same white circle placed over an intense yellow as can be observed in *Figure 4.1*. The explanation behind this process is that the intensely colored background, overwhelms the human eye, stressing its perception of the actual object within. The distracting background makes it impossible to focus on the shape, making the light coming out of it seem more intense. Also, there is a big similarity between the two colors, making the circle's borders hard do distinguish, the eye shifts from yellow to white and vice-versa, losing focus on both figures.

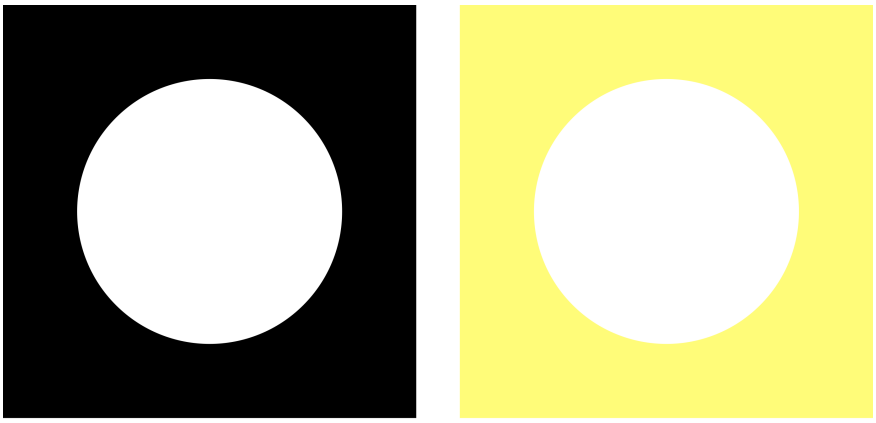

Figure 4.1 – Light perception experiment

On the other hand, the left image with the black background relaxes the eye, making the shape border clearly visible. Although both sides use the same white circle, the effects are pretty impressive, illustrating one of the multiple faults in our human vision.

Digital vision on the other hand is more sensible to these aspects. If we were to take a set of burst shots of the same subject and in ideal conditions, none of the shots would be equal or even be nearly comparable from a binary point of view since the image sensor used to capture the images also registers light intensity variations in the form of white noise. White noise in images is a phenomenon directly associated to the fact that light is just another set of particles that travel in time with a certain speed and distance from other light particles (photons). The moment in which the camera begins to register light only captures a limited set of photons that hit the sensor during the time the camera registers them. The burst set of pictures may seem identical for the human eye, but analyzed in their binary form we can see that there is always change between every shot taken. Although this white noise is registered, there is a constant balance between the noisy pixels; where in one frame the LSBs would indicate a brighter color and light intensity, in

other frames it would indicate a darker color. Put in context as groups of colors, the visual effect looks identical between the frames.

From this observation I came to the conclusion that a natural image is basically an image originating from a real light source, captured by a sensor that transcodes physical light into digital format. The image captured, viewed from the LSB perspective represents a hazardous but balanced distribution of colors and light intensities.

One must distinguish between colors and light intensities. Although in computerized vision each color has a light intensity, for human vision the light intensity for the smallest color representation unit (typically the pixel) is almost never perceived. Instead, we perceive light over a larger portion, formed by a group of pixels. This also represents one of the building blocks for lossy image compression such as JPEG where lightness covers an area of 64 pixels grouped as an array of 8x8. The 64:1 reduction basically covers light over the specific region, leaving the actual colors being represented as deviation deltas from a central average color variation.

The reason I insisted on natural images is because of their high importance to steganography. Computer generated images are usually unsuited for steganographic purposes because they do not contain noise, which is crucial for my purpose, as a steganographic algorithm designer. Embedding secret data onto a computer generated image them would introduce noise, making them vulnerable to detection. Natural images on the other hand present much noise (if they have not been processed and enhanced), making the noisy areas the perfect insertion spot for secret data.

4.2 Data deduplication

Data deduplication is probably the number one storage resource saver. This technique has been introduced mainly for the use inside server storage systems and databases, where multiple clients have access to the resources, but it is not limited to them. When multiple parties share the same storage space, it is unnecessary to have multiple copies of the same information; instead one could divide the information into smaller manageable chucks (blocks) and search for a corresponding component already stored. If such a component is found, instead of storing it again we can point the retrieval system to the right area with the help of pointers. If the data chunk is not found it can be stored either as a whole or as a delta change from the nearest similar block (if possible). Once two parties access and modify the same block, the first one would lock the resource and modify it, where as the second one would create a new storage area for its modified block. In this way we can store data reliable and by using less storage space.

The technique of deduplicating information only becomes effective if a larger amount of data is already present. When starting from scrap, the system evolves with every new data stored, constantly optimizing its space. Although hard to imagine, the effectiveness increases when adding and storing more and more data. This is one of the main reasons these systems are usually initialized by random data, in order to speed up the entire optimization process. Data deduplication can be associated with evolutionary algorithms: the higher the breed the better the chance of evolving the entire system.

In my research I found deduplication techniques very useful if used together with steganography. If we strip the LSBs we tend to reuse for storage space, we get a messy storage space that already consists in a multitude of binary combinations, crucial for a deduplication system. By analyzing the secret data at a binary level we can identify that many of the combinations can already be found within the white noise of the image (the extracted LSBs), depending on the granularity with which we handle the binary blocks (how many bits we group).

4.3 The ideal cover image

Steganography proves itself effective if it can surpass detection filters. Unlike cryptography where breaking the cipher would facilitate the exposing of the secured information within, steganography's biggest threat is steganalysis detection. The variety of algorithms and techniques used to store and map data onto the carrier image present a reverse engineering complexity higher than most cryptographic ciphers. Since the main purpose of a steganographic algorithm is to hide the trace of the communication channel, one must meet several precautions in order to surpass analysis gateways.

For steganography to be more effective, the choice of carrier is crucial. If the same carrier is used for repetitive embedding stages, one can easily expose the differences by analyzing a wider set of intercepted images. Ideally the carrier should be an image that is nowhere to be found in its original form, usually this would imply taking an image and immediately using it for embedding.

4.4 Embracing the origin

The majority of steganographic algorithms presented start with the premise that the noisy LSBs of each color component are less important. Although this may be a correct assumption in 1 LSB steganography, it is faulty when coarsely modifying more binary LSBs. The embedding of secret data bits in on a wider space inside the pixel table over lapses with important color channel bits that can alter either the color to a completely new and perceivable nuance, or by affecting the brightness of the pixel. This may not have an immediate effect on perception because of the reduced size of the pixel, but in context, groups of coarsely modified pixels have a strong visual impact and are easily spotted.

In my research I've tried to avoid this problem, instead of ignoring the binary combinations that the cover image offers, I see them as a database of already existent information. In an ideal case, the secret data viewed as binary combinations of bits are identical to the LSBs of the carrier image. This case wouldn't even require the use of an algorithm to embed the secret data, as it is already present. As good as it may sound this is almost impossible to achieve since the incidence of this to happen decreases with the size of the secret data we wish to embed. As my research went further I focused on finding ways that are tangible to the ideal model.

Instead of focusing on changing the LSBs I focused on modeling and manipulating the secret data in a manner that if embedded produces the smallest degree of change on the original carrier. The idea is to reorder the secret data so

that the algorithm achieves the maximum similarity between the original bit combinations as given by the image's white noise and the secret data bits. This technique is also known as LSB Matching steganography and represents a new trend in algorithm design. The steganographic algorithm morphs into an advanced binary analysis tool that tries to correlate the secret data bits with the already contained color channel LSBs. By doing so, the entire process analyzes different combinations of data rearrangement techniques using quality and logical metrics to study the effects the secret data would produce to the original image in the event of embedding. This type of approach outputs a variety of embedding strategies in an iterative manner, limited by several factors such as:

- **Running time** – Since the LSB Matching algorithms are repetitive processes; the notion of best solution is represented by local optimums. The longer we let the algorithm do analysis tasks the better the solution can become but this does not represent a guarantee. Depending on the scope we use steganography for, the running time of the algorithm can be placed under constrains, considering the best resulting solution as being the one that can be found given a limited timeframe. This constraint can prove effective even when dealing with differently sized cover images, as the algorithm must end its execution at a certain point in time.

- **Iteration counts** – Exactly like the running time constraint, iteration count limiter can be introduced in order to limit the maximum number of attempts to remap the secret data. This constraint however can be less effective than the time constraint since the number of iterations needed to reach the desired quality can vary depending on the environment (cover image, secret data). This metric however can be used to benchmark the effectiveness of the evaluator. If we try to improve the evaluator that adapts the secret data onto the carrier image, using a limited number of iterations we can reliably compare the evolution of embedding strategies.

- **Quality metrics** – Probably one of the best limitations, the quality constraint stops the algorithm cycle when the secret data is remapped in a manner that produces the smallest amount of change on the original carrier. This limitation assures that the output image is within the desired quality limits, in order to enforce the steganographic image's result against various types of steganalysis and detection filters. The only downside of this method appears in cases in which the quality demand is higher than the optimization capacity of the algorithm, leaving the algorithm run endlessly.

- **Recurrence counters** – Directly related to the quality metric limitation, recurrence counters identify repetitive solutions that have the same desired quality. This method groups the best solutions within a group of local optimums. If a specific amount of solutions remain within the same quality limits, the method assumes that the algorithm's possibilities capped and terminates its execution in order to avoid unnecessary processing time. The quality metric and recurrence counter

limitations are best used together, issuing an algorithm termination routine fitted for all non-benchmarking tests.

- **Steganalysis metrics** – Probably the most intensive limitation the steganographic algorithm's runtime can be limited by steganalysis detection filters. This ensures that the algorithm will stop searching for solutions if the best solutions found so far can surpass the embedded detection filter. This limitation is most effective when we know what steganalysis methods are used in order to detect our secure and secret communication. However, the downside of this limitation relies in the increased complexity of the resulting algorithm. Also, if the algorithm fails to find embedding solutions that pass the analysis stage, the process never terminates.

- **Full recombination coverage** – In LSB matching, in order to avoid recurrent and repetitive solutions, they can be indexed in a hash table, only the best solution would be stored in its entirety in order to avoid high memory usage. If the secret data and cover image is reduced in size, there is a small possibility to cover all data combinations possible, case in which the algorithm stops because it covered the entire solution space. Usually both, the secret data and cover image, are largely enough for this never to happen within acceptable timeframes.

CHAPTER 5

Nexim One Algorithm

> "There is a single light of science, and to brighten it anywhere is to brighten it everywhere." – Isaac Asimov

With Nexim One I propose an alternative hybrid, which is based on the OPAP method, but with a new matching algorithm. My method is similar to other LSB matching approaches [26][40] but uses flexible parameterization to best fit the purpose of reducing the image degradation in terms of PSNR values. For testing purposes the algorithm operates with grayscale 8bit images, but can also be applied on color-rich images.

Particular to this algorithm is the employment of a jump table as a data rearrangement map. The jump table represents a collection of addresses that scatters the secret data onto the image in a manner that reduces the error of bit replacement. The error is given by the difference between the original and modified value defined by:

$$error = (original - modified)^2 \quad (5.1)$$

These addresses point out image regions called segments. The algorithm builds these segments according to its initial configuration. The jump table is stored in the image along with the secret data at the cost of shrinking the capacity by a negligible amount. In contrast to other similar algorithms that employ image segmentation and the usage of jump tables [40] my method does not offer any constraints regarding the number and size of each segment.

Once the embedding process starts, the algorithm does a series of generation-evaluation iterations to get the best possible result (the generation time or number of iterations is only limited by the initial parameters). As opposed to normal genetic algorithm based steganographic ciphers, the current method has two optimization and evaluation stages:

- Solution generation
- Solution evaluation

The solution generation stage uses a local fitness function that selects a solution based on the bit replacement error it produces (the smaller the error the better the solution). By controlling the solution generation the algorithm obtains

solutions that offer a minimum amount of steganographic performance (small error values). This approach is different from traditional genetic algorithms where the initial solutions are randomly chosen without a selection criterion. In this way we reduce the faulty combinations and provide optimal initial solutions. These solutions are analyzed using a global fitness function (solution evaluator) that studies the effect of embedding the data using the solution generated (the jump table). The jump table indicates the order in which the data is scattered onto a set of image blocks. By scattering the data, the algorithm minimizes both the logical data distribution and the change error, because the jumps are always made to blocks in which the error is at its minimum.

Following the data trail along the image blocks, the evaluator calculates the cascading effect of embedding and measures it using Peak Signal-to-Noise Ratio (PSNR) as the base metric. Using this metric, the evaluator detects and tries to eliminate weak spots, that is, jumps that produce minimum error for the first element and higher for overall repetitions. Because the evaluation stage also attempts to optimize the solution, it also represents a second-generation stage (or more commonly a fitting stage). If no better solution can be found after this process and if not limited by initial parameter constraints, the first solution generation stage gets reactivated and the entire process is repeated. Each new complete generation-evaluation pair is compared to prior results in order to pick the best one as final solution (in terms of jump table combination). The final solution is then used to embed the data using traditional LSB replacement methods onto the image.

This algorithm approach is basically a pre-optimization stage that rearranges the data so that traditional methods produce small deviance errors while embedding. In the next paragraphs I will provide more details about the generation, evaluation and embedding processes.

5.1 Steganographic Encoder

Using an abstract representation, *Figure 5.1* illustrates the general workflow of the algorithm. The algorithm is structured into four stages:

A. Image segmentation

B. Solution generation

C. Solution evaluation

D. Solution embedding

In the first stage, *image segmentation*, the image is divided into a set of blocks (usually equally sized). These blocks are streamed together with the secret data to a jump table generator present in stage 2, *the solution generation.* The solution generation process builds a new jump table, in a sequential manner (element by element) until the table is full. The process also applies a minimal set of metrics to assure the quality and validity of the new solution found. Because this is a recursive algorithm, once a jump table combination has been found, the algorithm saves its state in order to resume the generation process if needed. In other words, the generator issues a large number of solutions sorted descending using internal solution evaluators. The jump table generated during this stage is passed to the *solution evaluator* present in step 3.

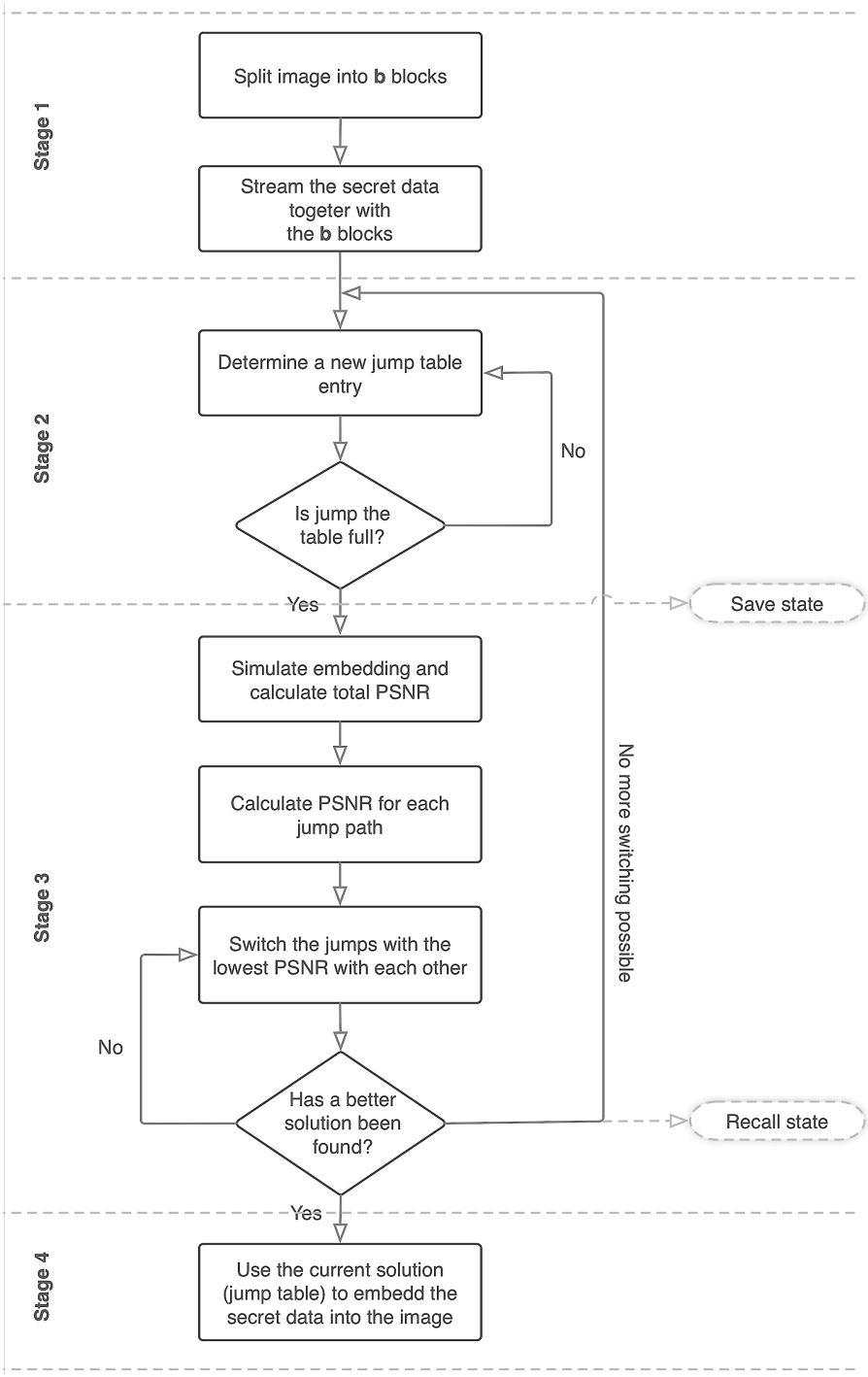

Figure 5.1 – Encoder workflow

First the evaluator simulates embedding using the jump table and calculates a PSNR value for the solution. This value is taken as a referenced best solution. The algorithm also generates a PSNR value for each table element and identifies the weak spots in the table. After these metrics have been computed, the algorithm takes pairs of two faulty jumps and performs a switch between them. The switching consumes a negligible amount of time and the evaluator tests the effect of this process. If a better PSNR has been obtained during this process, then the reference best solution is replaced with the current one. If no other switching is possible or if no better solution can be found, the algorithm recalls the generator's recursive stack and asks for a new solution.

The algorithm stops depending on the configuration parameters, if either a certain number of solutions have been generated (stage 2) or the steganographic image quality exceeds a specified dB limit (stage 3). Stage 4, *solution embedding,* takes care of the embedding process. The jump table is initially embedded into the image; then, using the jumps, the secret data is scattered among the image blocks until there is no more secret data to embed.

Let us go into more details for the 4 stages of the encoder and offer an overview of the entire embedding process.

5.1.1 Image segmentation

In the image segmentation stage, the cover image *C* (with the internal pixel matrix denoted as cij) is divided into b blocks defined by:

$$b = 2^{bpp+tileMultiplier} \quad (5.2)$$

- bpp - Represents the amount of volatile bits used for embedding
- $tileMultiplier \geq 0$ - Represents an extension of the initial minimum segmentation blocks that can lead to better solutions by offering a higher block number and implicitly a better solution space.

These blocks represent regions of the image (tiles). Depending on the number of bits per pixel we intent to reuse, it was necessary to allocate a number of blocks equal or greater than the possible combinations of the bpp pixels. If we chose for example a bpp of 2, then we have four possible bit combinations: 00, 01, 10, and 11. In this case the minimum amount of tiles would be 4. This constraint was introduced because we assume that with a tiling equal or greater to the combinations possible we raise the chances of having in each stage a minimum of 1 block containing a specific combination so that the embedding will produce no change at all. Please note that the above statement is just assumed, for better results it is recommended to keep the tiling at a higher number than the combinations possible to raise the diversity of concurrent combinations.

A solution in this implementation is a new *jump table* combination. The jump has a size described by:

$$tableSize = b * tableMultiplier \quad (5.3)$$

- $tableMultiplier \geq 1$ – The $tableMultiplier$ indicates the size multiplier of the jump table. The more jumps the table can store, the better the result, but at the same time the lower the steganographic capacity. Each table entry represents a binary address, a jump to a specific image tile.

For example, if we use an image of $512x512$ with a bpp of 1, tile multiplier index of 1, table multiplier index of 6, we get a $b = 2^{1+1} = 4$ image segments (tiles) and a jump table size of $tableSize = 4 * 6 = 24$. *Table 5.1* lists the pixel regions of the tiles:

Tile	Binary	Range Y	Range X	Pixels/Tile	Capacity/Tile
1	00	0..255	0..255	65536	8192 bytes
2	01	0..255	256..511	65536	8192 bytes
3	10	256..511	0..255	65536	8192 bytes
4	11	256..511	256..511	65536	8192 bytes

Table 5.1 – Example tiling for 2 bpp

In this case, each image tile reserves space for 6 elements. A table entry would take 2 bits for each of the jump table entry leading to a total physical storage space for the jump table of $2bits * 24elements = 48\ bits$ out of which each image tile stores $12\ bits$ (1,5 *bytes).*

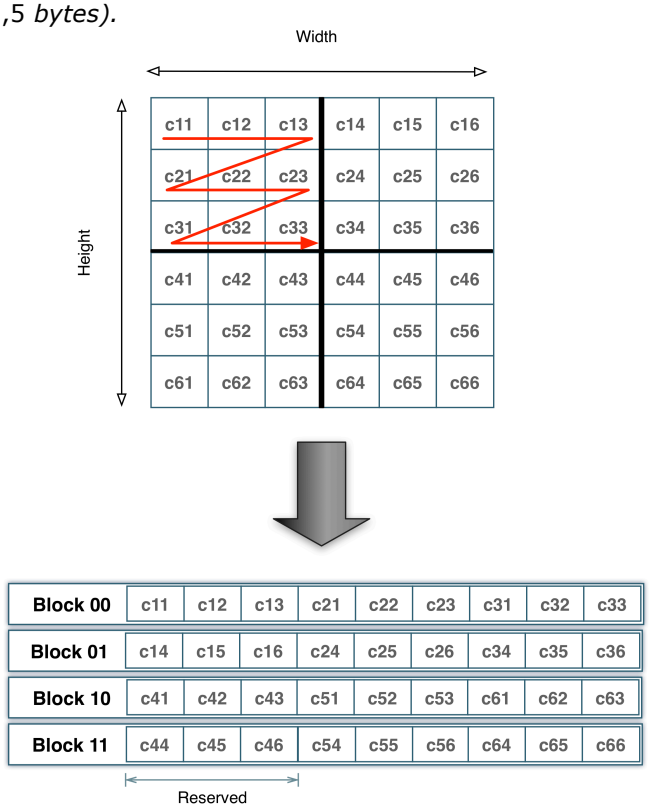

Figure 5.2 - Example segmentation (bpp = 2, tileMultiplier = 0, tableMultiplier = 3)

The segmentation stage builds up *b* streams (noted $S[x]$ where $0 \leq x < b$), where each stream contains the pixels corresponding to the associated image tile. Each of the *b* streams works like a separate evaluation entity, being a record set composed of pixel information and states. *Figure 5.2* illustrates the segmentation process of the image into b = 2^2 = 4 blocks. Each stream receives the pixel range of each block. This can either be done physically by building the streams in the memory or virtually, where each stream knows the address of every element.

Internally, the stream record consists on multiple components:

- **Pixel table** – stores all pixel values of a specific image tile represented in a continuous vector
- **Capacity left** - indicates if the stream has any embedding capacity left
- **PSNR** - measures the deviation between the original signal (cover image C) and the resulting image (stego image) for the specific image tile handled
- **Initial induced error function** – measures the error between the original information stored within the pixel table (the currently available pixel table combination) and the currently available secret data sequence that needs to be embedded. This function is used mainly in the solution generation stage and will be further detailed there.
- **Cascading error function** – evaluates an overall PSNR for each table element, by sequentially analyzing differences between the original and modified string of data (least significant bit groups). This function is called each time a solution has been generated and it is mainly used to measure the deviation as a result of embedding the secret data in the order described by the jump table. As opposed to the initial induced error function, this function compares all changes made, not only the current combination available. This function will be further detailed in the solution evaluation stage.

5.1.2 Solution generation

The solution generation stage is a recursive method that generates solutions in form of jump table combinations. In the solution generation stage the secret data *D* is streamed to an evaluation block.

The algorithm iterates one group of secret data bits at the time and passes them to a decision block (solution generator in *Figure 5.3)*. Here, each of the *b* streams is provided with the current combination as an input for the *initial induced error function.* This function computes the embedding error:

$$error = (D[D.p] - S\,x[Sx.p])^2 \quad (5.4)$$

The error function takes the current group of secret bits ($D\,[D.p]$, $D.p$ = the current index position inside the secret data stream) and the currently available

combination given by the original image's pixels ($Sx\,[Sx.p]$, x = active stream – each stream is accessed sequentially with each iteration, $Sx.p$ = current index position inside the pixel stream of each tile). Each of the b streams responds with the error value. These values are stored in an array, which is then sorted ascending. The sorting makes sure that the first choice is always the one that produces the minimum amount of degradation. If the recursive algorithm returns for further solutions (the initial algorithm parameterization is configured to find more than one solution), the next solution will be picked from the list and the algorithm evaluates each new solution found in comparison with the best solution yet.

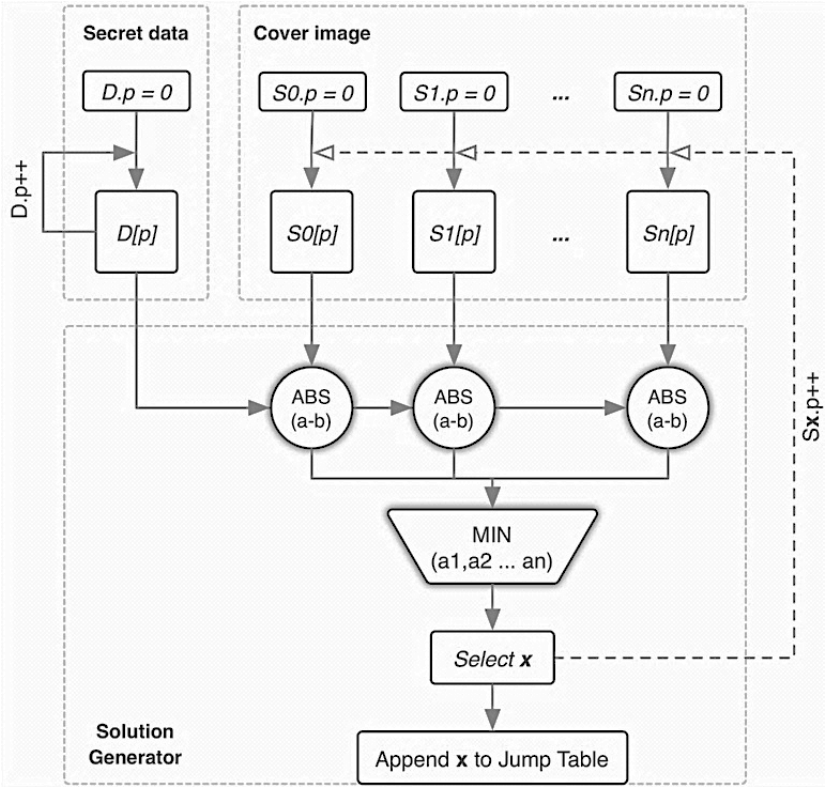

Figure 5.3 – Solution generator

The solution generation stage stops when the jump table is full and passes the solution to the *solution evaluator*. This stage uses a number of iterations equal to the jump table size counted in elements. For the example from Section A, the generation would iterate and sequence 24 secret data bit groups, after which the jump table gets full and the solution is passed to the analysis stage.

The first generated solution is optimal in the near-sighted field because of the *initial induced error function*. Near-sighted field means that the evaluation function operates on a limited field of view and does not study the effects on the entire embedding process. The optimal solution here is not necessarily the best overall; the evaluation stage is used to test the effects on the entire storage space of the cover image, thus providing a far-sighted approach.

5.1.3 Solution evaluation

Because of the multitude of possible solutions, the generation function is implemented in a near-sighted manner, making the evaluation stage a crucial analysis point. In the example from Section 5.1.1 Image segmentation, the total capacity of the 512x512 pixel image at 1bpp would be 262144 bits, out of which 48 bits are reserved for the storage of the jump table. In other words, the jump table occupies 0.01832% of the total image capacity. We consider this as a tiny amount of storage compared to the total capacity. The first solution, as it leaves the generation stage, offers a guarantee that a minimum of 0.01832% of the secret data bits produces almost no change in the original image. This stage analyzes the repetitive nature of the embedding that follows the jump table generation and decides if the solution is valid or not.

The solution evaluation follows the jumps presented in the jump table in a circular sequence. The algorithm cycles through the table jumps until there is no secret data left to be embedded. During a cycle the algorithm follows the addresses and jumps to the indicated blocks. The first generated solution is assumed to be the local best solution found. Each new solution is compared to the local best, which gets updated if a better solution is found.

As opposed to the solution generation stage, where the *initial induced error function* analyzes a near-sighted set of data, the evaluation stage uses a far-sighted evaluation function called the *cascading error function.*

For each of the *b* blocks, the evaluation stage counts the jumps to each block (noted $Jb\,[x]$). This count specifies the distance between embedded repetitions inside each block. For the example in Section 5.1.1, we have the following jump table solution:

(2,1,3,1,3,2,2,0,2,1,1,0,0,0,0,3,3,3,3,0,3,3,1,2)

For this example, the algorithm would count the occurrence of every jump to a specific block tile as shown in the list below:

- $Jb[0] = 6$
- $Jb[1] = 5$
- $Jb[2] = 5$
- $Jb[3] = 8$

If we take $Jb\,[0]$ as a reference, its count 6 indicates that each 6'th jump would point to the indicated block, in this case 0. In other words, the algorithm streams the secret data onto the cover image and computes for each jump the PSNR value and builds up a secondary list of table jump evaluations (one PSNR value for each element of the jump table).

The lowest PSNR values are extracted from the list in pairs of two and switched with another. This process is repeated until a certain number of permutations have been made or until no better solution has been found. The best solution found at this stage is compared in terms of overall image PSNR (cover vs. stego image) to the normal LSB substitution method's PSNR value. If the solution found is better, the algorithm can stop and consider its current jump table

combination as the best yet. The algorithm stops depending on the recursivity limitation if a certain number of solutions have been found or if the newly generated solution meets a certain degree of quality (external output evaluation function that measures the deviance between the original and steganographic image).

If the algorithm has been configured to try multiple recursive iterations, then stage 2 gets reactivated and a new solution will be found. The more generation-evaluation iterations the algorithm makes the better the results can get (this is not always a guarantee); at the same time the overall processing time and the demand of computational power are increased.

Nevertheless, up to a certain point, several iterations can be made without having a significant impact to the processing speed and the experimental results show for a majority of cases a better PSNR value than other LSB matching algorithms.

Because of its design, this method can be considered a genetic algorithm where stage 2 would generate a new population (solution), and stage 3 would be responsible for selection, breeding and reproduction.

5.1.4 Solution embedding

The algorithm cycles the solution generation and evaluation pair until a certain number of iterations has been reached or a desired PSNR target has been met. Once the generation and analysis stage have completed this stage is responsible to actually build the output steganographic image.

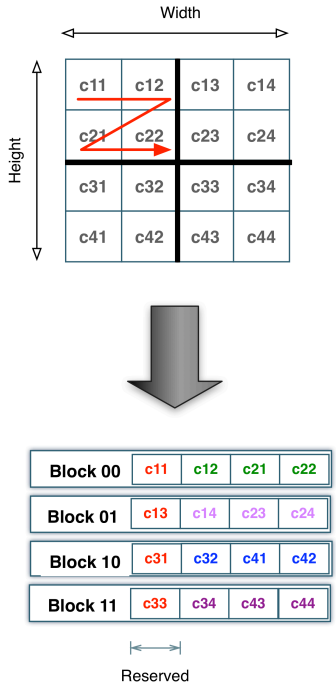

Figure 5.4 - Example on a 4x4 pixel image

First, the algorithm embeds the jump table into the *b* image tiles, after which it embeds the secret data in the order dictated by the jump table.

If during the embedding of secret data one of the *b* tiles gets full, the algorithm then skips this block tile and embeds the data in the first block indicated by the next jump. In other words, if the remaining capacity of a certain block reaches 0, the future jumps to this specific block will be invalidated and the data will be embedded in the block specified by the jump elements that are still valid inside the jump table.

The example presented in *figure 5.4* illustrates a sample 4x4 pixel image with:

- $bpp = 2$
- $tileMultiplier = 0$
- $tableMultiplier = 0$

The jump table 00,10,11,01 has been generated and data needs to be embedded following these jumps. The encoder partitions the image into 4 tiles and creates 4 handling streams, one for each tile. Each of these streams will have 8 bits of total capacity, out of which 2 bits are reserved for the jump table. This leaves 6 bits capacity/tile, 24 bits capacity/image for the secret data.

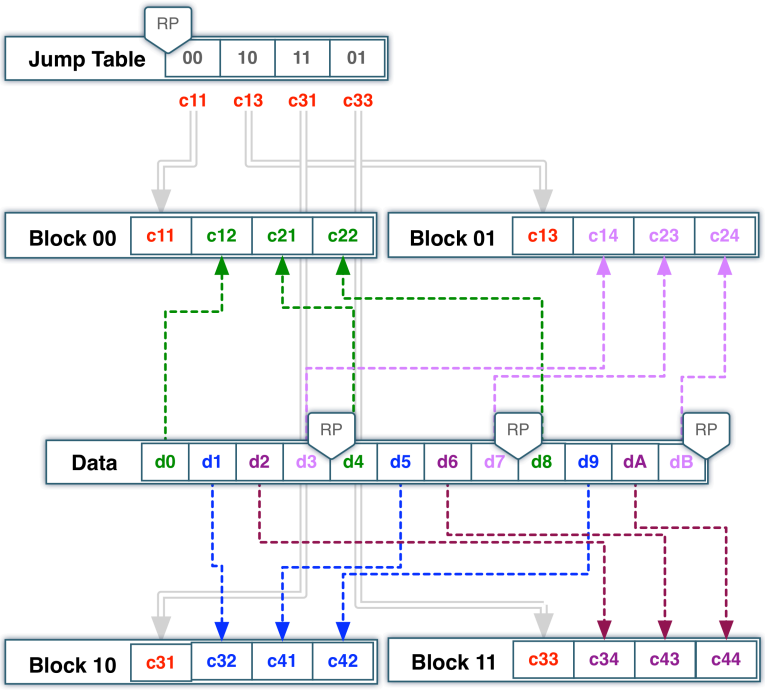

Figure 5.5 - Example data distribution using the jump table

Figure 5.5 shows how the data would be embedded into the stream. The algorithm would operate on the 4 streams:

- Stream 1: **Block 00**
- Stream 2: **Block 01**
- Stream 3: **Block 10**
- Stream 4: **Block 11**

On the first position of each of the 4 streams comes one element from the jump table (marked with red in the graph), after which, following the jumps and reordering dictated by the jump table, the secret data would be sequentially streamed to block 00,10,11,01 (in this order). After all, secret data has been stored within the streams; the streams get united, forming the steganographic image.

If the primary target is performance, then physical implementation for the streams is required, whereas a focus on memory usage may rely on virtual streams as simple pointer lists. Depending on the implementation, the stream concept does not necessarily have to be physically implemented, a simple pointer list (virtual stream) could also work, but since an image is a matrix of pixels, one could lose performance on one hand, but use less memory because the data doesn't get duplicated. For this algorithm design, because of performance issues, I used physically implemented streams.

5.2 Steganographic Decoder

The steganographic decoder is the reversed encoding stage. The decoder uses the steganographic image as main data input and the number of segments *b* and table multiplier as the single parameter input.

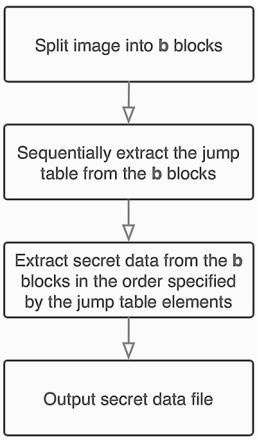

Figure 5.6 – Decoder workflow

Using these parameters, the decoder first splits the image into *b* blocks (*Figure 5.6)* of data. The splitting can either be implemented physically by assigning a number of b streams to hold the data of each image segment, or virtually by mapping a pointer array to the image's pixels. Either way the result is the same, the only difference is the speed or the computational complexity. From each of the generated streams (or arrays, depending on the implementation) the jump table is extracted by sequentially reading the streams. Once the jump table has been extracted, the secret data stream is built by extracting the data from the streams in the order specified by the table jumps.

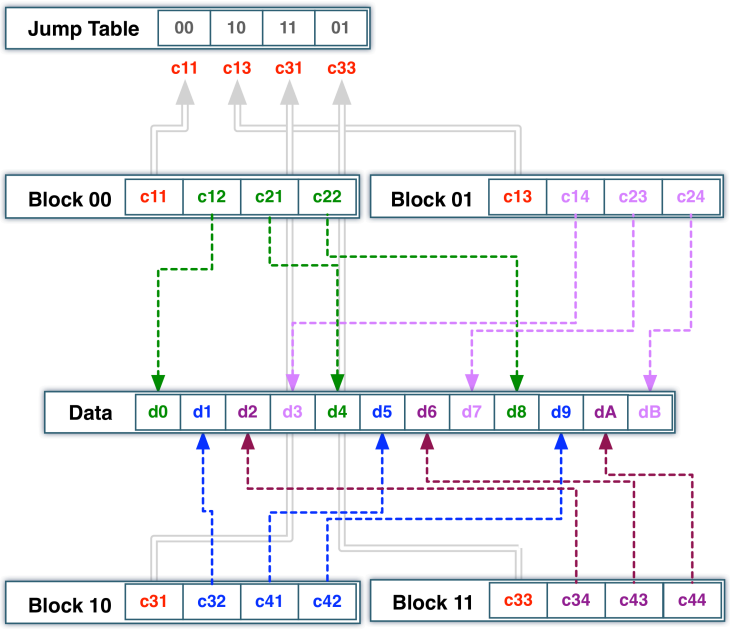

Figure 5.7 - Example jump table and data extraction

The algorithm stops requesting data from a specific stream if the stream has reached EOF (or if we use arrays, until the index has reached the array size). If all streams reach their end, there is no more data to extract and the encoding process completes and saves the output.

In opposition to the encoding process, the decoding process is a straight-forward fixed algorithm recursion.

Using the same example as in *figure 5.4*, *figure 5.7* illustrates how the decoder works. After initially splitting the image into 4 blocks, the decoder sequentially extracts the jump table by visiting block 00,01,10,11 (in this order). After the jump table is complete, the data is extracted by visiting block 00,10,11,01 (in this order) and the data stream is completed.

Regardless of the implementation method used, physically (streams) or virtually (pointer arrays) the algorithm offers high extraction speed that finishes in polynomial time.

CHAPTER 6

Nexim One Extended Algorithm

> "The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom." – Isaac Asimov

Extending my previous work, I try to enhance the overall quality of the resulting image and also avoid the brightness spot problem.

The image segmentation stage remains the same as in the original design. First the cover image used for secret data storage is split into *b* blocks, depending on the embedding capacity chosen using the relation:

$$b = 2bpp + tileMultiplier \quad (6.1)$$

The second stage of the algorithm has been modified in order to enhance and correct the previous design. The initial optimization strategy used in the generator tried to minimize the change relating only to the original image. When replacing the LSBs of the original image we have three possible deviations on each color channel (RGB):

- **No deviation**, when the bits to be embedded are the same as the original;
- **Gain deviation**, when the individual color channel's value gets increased;
- **Loss deviation**, when the individual color channel's value gets decreased.

Because the prior method did not take these into consideration, in some test cases where larger portions of the image had boundary combination of RGB triplets (near to maximum or minimum brightness) changing these RGB values in the wrong direction saturated the color, leading to visual identifiable spots either through their color or brightness abnormally. With the use of virtual high dynamic-range images (HDR) I try to avoid this problem.

Figure 6.1 illustrates the new proposed solution generator. Instead of fixating to reduce the color loss only in relation to the original color table, I now set two color boundaries: upper and lower. The original image is considered as being a HDR image. Based on this fact, the current design treats the original image as

having two components: an underexposed (decreased brightness) and an overexposed (increased brightness) component. Both components represet virtual low-dynamic range (LDR) images that added up form the original HDR image. The splitting into two LDR images assures the upper and lower acceptable color change boundaries.

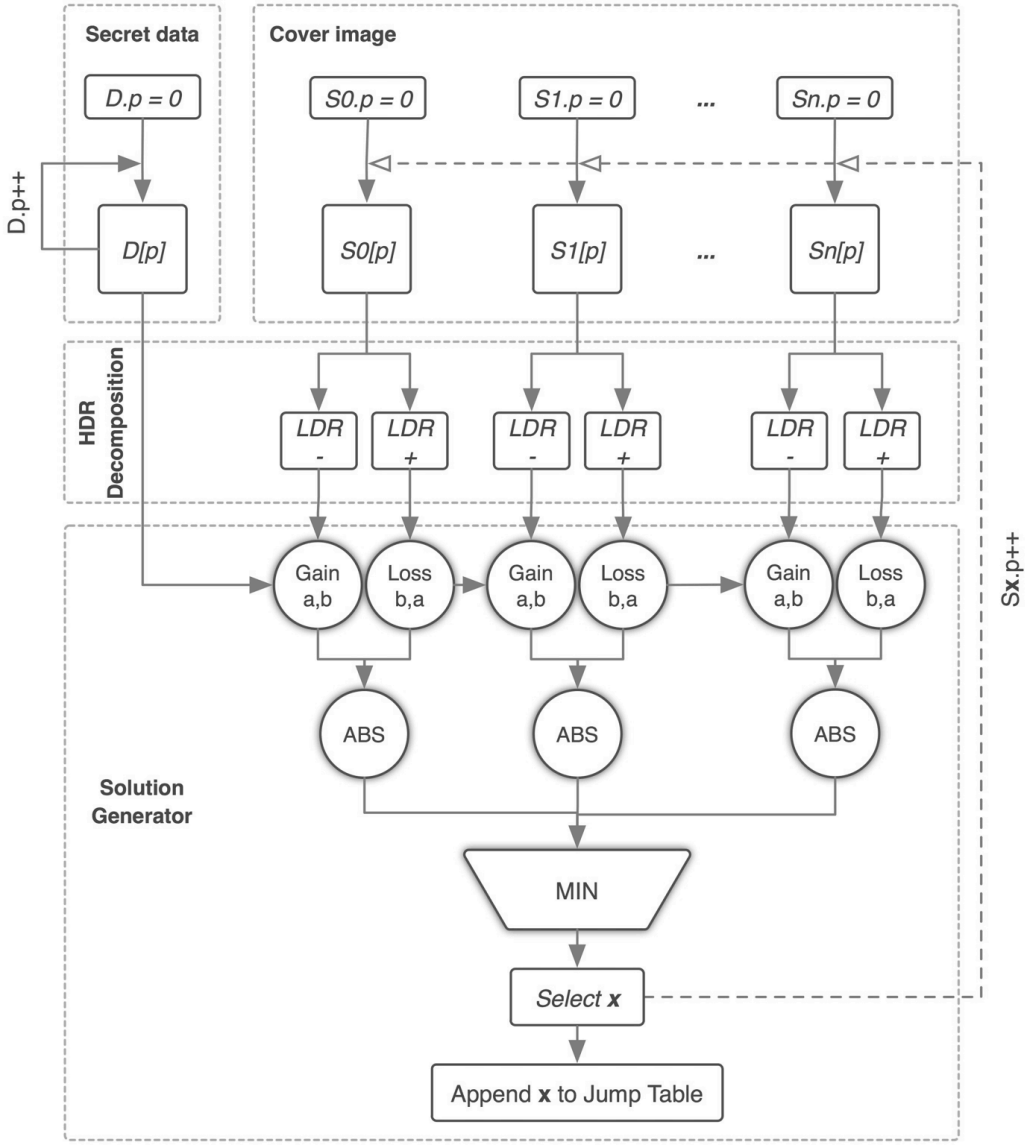

Figure 6.1 - The improved solution generator

As in my previous design, the current available bit combination from the secret data stream ($D\,[p]$) is streamed along with the currently available bit combination of each image segment ($Sx\,[p]$). $Sx\,[p]$ is processed through an additional HDR decomposition stage and split into two values:

- **LDR-** (brightness decreased or underexposed)
- **LDR+** (brightness increased or overexposed)

The amount of brightness loss/gain is calculated using $brightness = 2bpp + 1$. It is important to note that, given an embedding rate (volatile bits per pixel), the possible combinations on *bpp* pixels cannot exceed 2^{bpp+1} in any direction, regardless if we register gain or loss.

The input bit combination $D[p]$ is tested against both LDR+ and LDR- components using the *gain* or *loss* function. These two functions essentially subtract the new bit combination from the original one. In both cases we limit the change only in one direction (positive or negative). To better understand this process we will take a simple example. We have the following RGB triplet defined by:

- R = 111111**00** (252 decimal)
- G = 111111**11** (255 decimal)
- B = 110001**01** (197 decimal)

If we try to embed the following data bit combinations **11 00 01** we can observe a color abnormally. The simple replacement of these combinations over the original ones will produce a new RGB triplet defined by:

- R = 111111**11** (+3 gain)
- G = 111111**00** (-3 loss)
- B = 110001**01** (no change)

This is considered as being one of the problematic cases in our previous design, because it did not take into consideration whether the change implies loss or gain. Because the original G component was at its maximum, the only direction in which we can alter is by subtraction. When compared with the R channel whose value gets increased, the overall effect on the pixel (RGB triplet) would be a saturated color, distinguishable by the human eye. Note that these are however isolated cases. Using the new method, and by generating the two LDR+ and LDR- we can easily identify the boundaries before even trying to embed. If component RGB limits are identified when building the LDR+ and LDR- the system accepts only those combinations that affect brightness, not saturation, meaning that the change of all color components must be in the same direction. Bit combinations that exceed these boundaries are immediately filtered using the MIN function, and only the solutions within the limits, nearest to the original colors are picked.

This processing step only affects the boundary values, all other colors in-between ignore this combination, meaning that the algorithm then functions exactly as described in the old design (*Chapter 6*). The processing is based on min-max principles. Initially we filter out the highest gain/loss, meaning that we filter out the

worst case scenario, immediately flowed by picking out the best solution that produces the minimum changes in relationship with the boundaries and the original color information. By introducing this degree of flexibility, our new design mainly focuses on avoiding problems, before they even occur.

The solution evaluation and enhancement step however, does not always maintain the generated solution because it tests out the jump table's influence on the rest of the unoptimized space. If this stage is able to improve the overall effectiveness in terms of color preservation, the algorithm can out rule the previous local optimum in favor of a general optimum, introducing possible errors. Whit this in mind, we also introduced a post-processing method that corrects the error introduced by the solution evaluator.

As in our previous design, the solution generation and solution evaluation stages repeatedly iterate building better overall solutions, until certain limitations have been reached:

- Iteration limit;
- Timeout limit;
- Desired performance/quality in terms of PSNR has been obtained;
- Manual termination of the two stages;

Once a solution is found, the fourth stage (solution embedding) embeds the data according to the best solution found so far (jump table combination). The algorithm first embeds the jump table sequentially in the available image segments, followed by the secret data, in the order indicated by the jump table. Until this point, the algorithm built a steganographic image that resembles the original as much as possible.

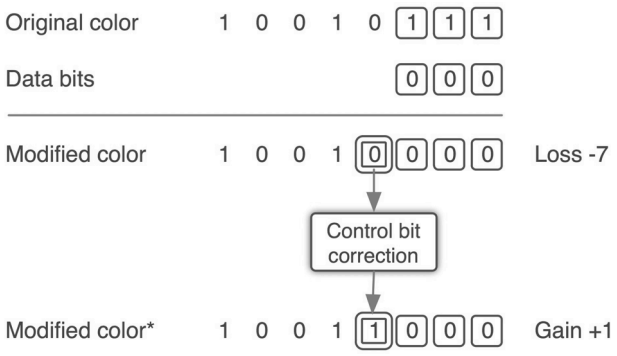

Figure 6.2 - Enhancement through control bit

I introduced a further fifth post-processing stage that handles the omitted errors. *Figure 6.2* illustrates one of the possible errors when dealing with 3 bpp embedding ratio. The original color combination bits are altered using the secret data bits producing a high degradation. The original combination of bits is 111 (as

marked in the figure) and the solution (jump table) embeds a secret data bit combination of 000. The impact on the color channel (either R, G or B) is very high, producing a loss of -7, which represents the worst-case scenario in image degradation. Using the post-processing step I analyze the original cover image and the resulting steganographic image in comparison and identify these cases. Regardless of the number of bits we reuse for secret data embedding (1-4 volatile bits/color channel), we use the last high order bit of the non-volatile color information as a control bit as shown in *Figure 6.2*.

In the example above, this error is easily corrected by changing the control bit, in which case we turn the loss of -7 into a gain of +1, which is acceptable. This newly introduced stage does not modify any of the embedded data bits, it just adjusts the control bits accordingly, increasing in the same time the PSNR between the original and steganographic image. In the event that these errors occur when dealing with border values (minimum or maximum) where we cannot successfully use the control bits to change the alteration rate, the algorithm uses the neighboring channels to normalize the color. The chessboard principle is applied to sharpen the resulting color by raising the saturation level, meaning that the erroneous color channel remains intact, while the neighboring channels brightness value gets increases/decreased depending on the case.

With these changes added to my original design I built an even stronger algorithm capable of offering better quality than the older model and in the same time eliminating minor faults.

CHAPTER 7

Nexim Two Algorithm

> "A subtle thought that is in error may yet give inquiry that can establish truths of great value." – Isaac Asimov

I propose an alternative LSB matching algorithm with a more complex solution generator. This algorithm has been designed to work with high color RGB images (24 bpp and above) with an embedding ration of 3:8, meaning that 9 LSBs will be used for storage for each image pixel (3 bits of Red, Green and Blue channel). These constraints have been introduced since the original design has been proven more effective in this particular constellation.

The algorithm is built in a repetitive manner, generating better embedding strategies with each additional running loop. The embedding strategy consists in a jump table that indicates the exact order in which the data is embedded in order to produce the minimal amount of change. Since the original image already has bit combinations in the last 3 LSBs of each color channel, the data must be embedded in a manner that reduces the difference (delta) between the original and inserted bits. The smaller the delta, the greater the resemblance to the original color variation of the image, the stronger the algorithm gets. In a hypothetical ideal case, the secret data (payload) is identical on a binary level with the original color combinations. In this case the image would already contain the secret data without the need of embedding. This case however is impossible, but the jump table tries to reach this goal by choosing strategies that minimize the change effect of applying payload bits.

Particular to this design, I've changed the core evaluation metric inside the solution generator to a more human vision oriented one. Although an increased Peak-to-Signal-Noise Ratio (PSNR) value indicates a higher quality conservation level between the original and modified image from a computerized perspective, the brightness spots identified in some cases in the original design are easily identified using human vision. This observation pushed me into finding a better evaluation metric that satisfies not only computerized analysis but also human-vision-based analysis by using alternative color space representations for the evaluation process. The human eye is known to be very good at identifying light and color variations over a larger pixel area than distinguishing between light spikes inside a smaller area. This led my research into using YCbCr as alternative color representation for evaluation purposes. The most important component of this color representation is the Y channel that represents the luminance of a color value (the amount of light as perceived by a human eye). The Y channel is defined as described below:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (7.1)$$

The luminance channel quantizes the light in relationship with its perceivable spectrum, with the green channel being the most important color component leading to brightness change. The generation stage choses those solutions that produce the minimal Y-channel PSNR over smaller portions of the image. The algorithm uses a series of 4 separate stages, as did its predecessor in *Chapter 5*, but in a modified manner, as observed in *Figure 7.1*.

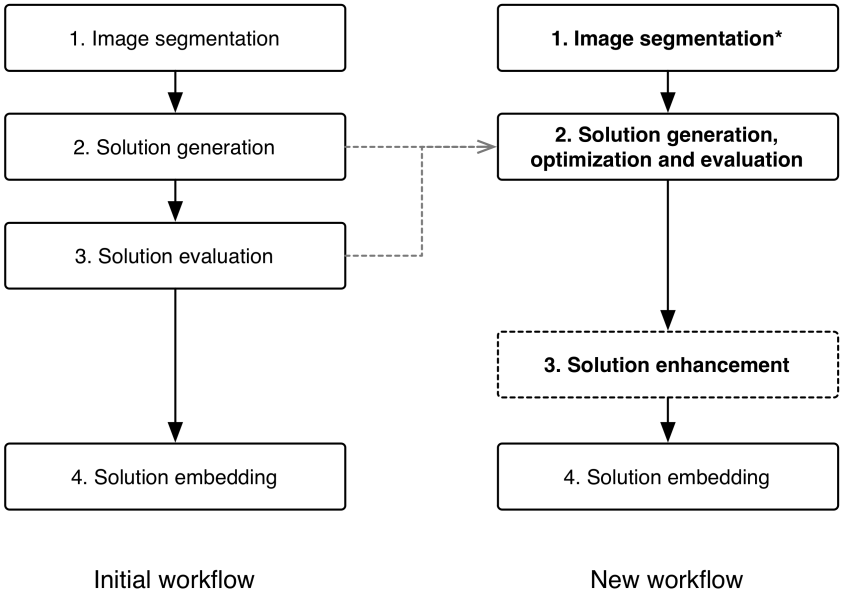

Figure 7.1 - Initial vs. new workflow

In the next paragraphs I will provide more information on the entire algorithm workflow ass it passes the four stages.

7.1 Image segmentation

In order to successfully remap the secret data into a less destructive manner over the original color information, the algorithm must be able to have multiple embedding regions and chose the region where the data produces the minimum amount of change. Therefore, the image segmentation represents a crucial step in the entire process as it offers the necessary amount of binary combinations to choose from. The image is cut into smaller portions that can be analyzed and handled more easily in a two-step process, as shown in *Figure 7.2,* which involves tiling and partitioning.

First the cover image C (with an internal pixel matrix denoted as c_{ij}) is divided into 4 equal tiles (denoted Sx, where $x = 1..4$). For faster processing the pixel values corresponding to each of these tiles are stored within four separate data vectors (Sx), that handle each tile separately. Tiles are then cut into smaller portions in a process called partitioning. Just like in JPEG compression, the tiles are split into 8x8 blocks (64 pixels), which represent a relatively small visual group of pixels. Each tile consists in blocks of 8x8 pixels denoted $Sx.B\,[y]$ (where x represents the tile address and y represents the block number. In *Figure 7.2,* an example image is shown having a size of 512x512 pixels. The tiling process would cut the image into four equal tiles, each sized at 256x256 pixels. The partitioning step would split each tile into 8x8 block, in this case building up 1024 blocks/tile.

Figure 7.2 – Image segmentation

As opposed to the original design where the data was inserted one pixel at a time, this algorithm uses a 64-pixel buffer, for performance and optimization purposes, as later described in the paper.

This stage of the algorithm is intended to split the image into multiple pixel streams to choose from when trying to find the best embedding path. Since the algorithm uses 3 bpp, the maximum number of available combinations is 8: *000, 001, 010, 011, 100, 101, 110* and *111.* The tiling process covers a maximum of 4 combinations, which is less than the theoretical minimum needed in order to statistically assure a balanced distribution of data. For this reason I introduced a third virtual direction that we call *layer ordering address* that basically tells the order in which the data is inserted within a single pixel. Due to the fact that a color pixel is represented as a RGB triplet, the data can be inserted using different ordering of the color channels. Out of the total number of combinations possible (6) I only chose 3, as shown in *Table 7.1*.

Channel order	Binary Address
RGB	00
GBR	01
BRG	10
Reserved	11

Table 7.1 - Channel order

Using this additional addressing, the algorithm has 12 embedding combination to chose from each step of the generation process.

7.2 Solution generation, evaluation and optimization

This stage is a recursive method that generates the best embedding strategy in form of jump table entries. A table entry in the solution generator stage represents a two-value address that consists in:

- Level 1 Address (channel reordering address)
- Level 2 Address (tile address)

The four tiles *S1..S4* and the secret data *D* represent the main inputs of this stage, as shown in *Figure 7.3*. Each tile internally stores the current active pixel block $Sx.bi$ (active Block Index). The algorithm starts by reserving a certain number of blocks where the jump table will be stored. The number of blocks is always a multiple of 4 in order to assure a balanced distribution among the four image tiles. The minimum number of blocks required by this algorithm is 4, leaving room for 576 table entries. For the image in *Figure 7.2* it would mean that by reusing 6,82% of the image's total storage capacity we can optimize up to 54,71% of the entire storage capacity, representing an important improvement over the original design. The secret data is streamed 72 bytes (576 bits) at a time:

$$576\ bits = 3\ channels \cdot \left(64\ groups \cdot 3\frac{bits}{pixels}\right) \quad (7.2)$$

As opposed to the initial design where the data is streamed and analyzed one bit group at a time, this algorithm uses a larger buffer to increase the overall effectiveness of each choice over al larger area turning the once near-sighted generator (*Chapter 5*) into a far-sighted one in this design. There are two direct advantages of this approach:

- The algorithm optimizes more information because it tests the effect onto multiple cascading pixels
- Dramatically reduces the size of the jump table

In the original design optimizing 64 pixels would require 192 table entries with an address length of 3 bits each, totaling 576 bits for the storage of the jump table alone. By using a 4 bit address in this case the same effect is achieved, meaning that the algorithm optimizes that same area as it's predecessor by utilizing 99,3% less storage space for the jump table.

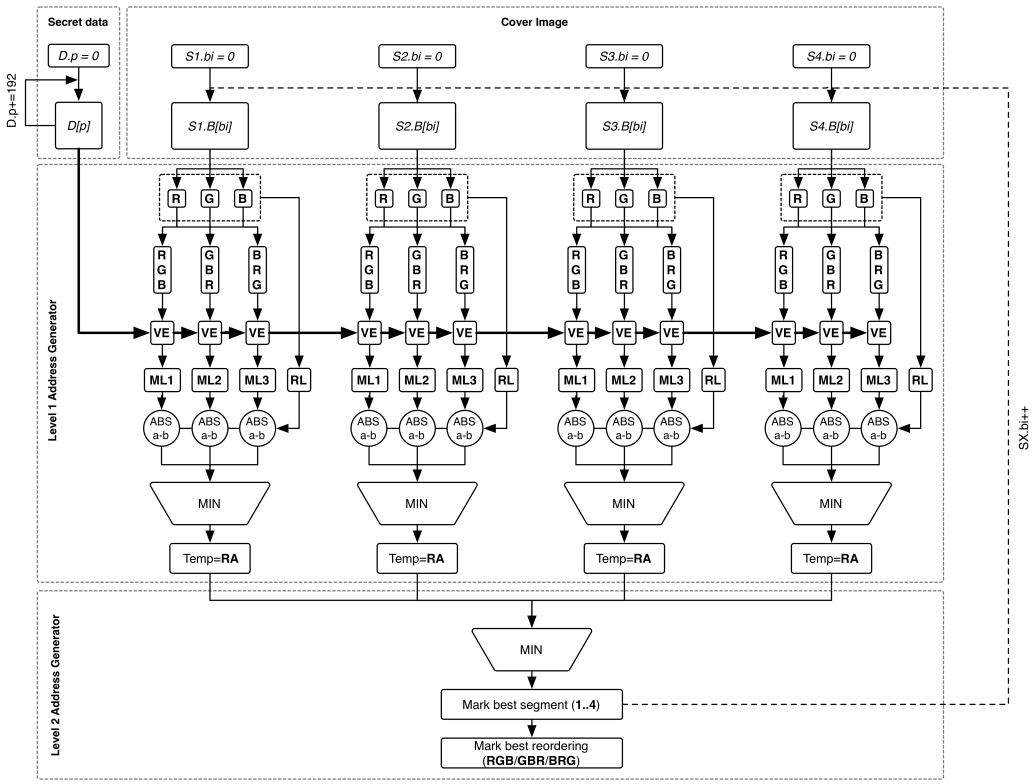

Figure 7.3 - The solution generator, evaluator and optimizer

These changes not only improve the general effectiveness of the entire algorithm, but also render the evaluation stage from the previous design useless. The current algorithm is capable of finding better solutions easier and more rapidly and the jump table gets negligibly small in size.

7.2.1 Level 1 Address Generation

The four image tiles stream the currently active block ($Sx.bi$) are divided into individual the color channels R, G and B. Based on these channels, for each active block of every tile, the algorithm determines the lightness component Y as *reference lightness* (RL). The RL is used to determine the alteration delta after the data has been embedded. The three-color channels are ordered as indicated above in RGB, GBR and BRG using three parallel reordering streams. Together with the secret data the algorithm *virtually embeds* (VE) the secret data onto the currently available block ($Sx.bi$) of each image tile (Sx). The embedding is simulated in order to determine the *modified lightness* (ML) of the resulting block in case that specific block is chosen for embedding. This part of the evaluation takes place on each tile and for each reordering sequence, totalizing 12 possible choices. Each tile picks out the reordering sequence that produced the minimum loss of luminosity in forms of PSNR (the greater the PSNR the higher the overall quality), reducing the 12 choices to just 4. The best reordering address (RA) is saved temporarily for the next step of the process in form of a *Level 1 Address*.

7.2.2 Level 2 Address Generation

In the second part of the process the algorithm picks out the tile that has the smallest degradation of lightness Y. Once this block is determined from the 4 remaining choices only one is picked and marked as being a solution, building the *Level 2 Address.* Together with the RA corresponding to the currently picked image tile, the Level 1 and Level 2 addresses are joined and added as a jump table entry.

The tile that produces the minimum amount of change will change its state to the next available free block ($Sx.bi++$), while all other tiles remain unchanged. Also, the data stream positions itself to the next 72 Bytes of data to be inserted.

The solution generator stage iterates for new solutions limited only by execution or iteration time limits, constantly searching for better solutions. One complete solution is considered as being obtained once the jump table has been filled. Since the size of the jump table has been reduced by 99,3% since the original design, we can use more table entries, allowing the algorithm optimize even larger portions of the image and therefore issuing better solutions than its predecessors (Chapter 5 & 6).

7.3 Solution Enhancement

In this design, with the introduction of a more advanced solution generator, optimizer and evaluator, there is no need for a separate evaluation step. The enhancement step was introduced more as a guideline for future implementations, but has not been explicitly built into the testing algorithm used for the current paper. As later shown in the experimental results, the algorithm is capable of increasing the overall PSNR of the resulting image up to 3 dB more than it's

predecessors without the need of any additional solution enhancement steps (in *Chapter 5*) or post-processing methods (in *Chapter 6*).

7.4 Solution Embedding

Once the algorithm has reached this stage, the optimization process is finished and the best solution in form of a full jump table has been determined. The algorithm then uses the reserved blocks of each image tile to first embed the jump table for data extraction purposes. The data is embedded by embedding the data sequentially in groups of 72 Bytes in the order indicated by the jump table. The evaluator extracts a 4-bit address from the jump table representing a full addressing sequence. The first two bits represent the image *tile address*. Once selected the algorithm reads the current state and identifies the currently available image block and selects it. Based on this block and using the *layer ordering address* the block's pixels are red in the right order (RGB, GBR or BRG) as shown in *Figure 7.4.*

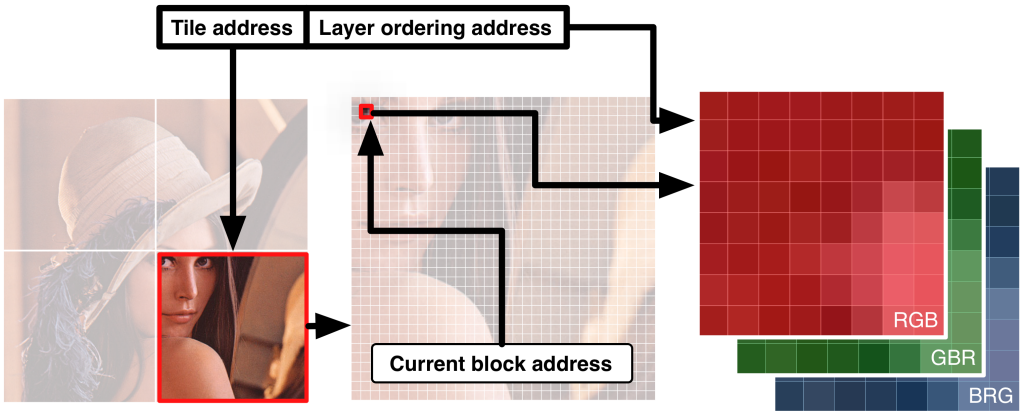

Figure 7.4 - New addressing method

The solution embedding stage cycles several times throughout the jump table, consuming the secret data bits. Once there are no data bits left to embed the algorithm finishes and outputs a steganographic image.

The process is simple and requires only the image segmentation and reverse solution embedding stage. First the extraction algorithm must rebuild the four image tiles and the image blocks accordingly. After this step the jump table is extracted from the reserved blocks (assuming the size of the jump table is known – the only parameter that needs to be transmitted), after which, following the exact sequence illustrated in *Figure 7.4* the last 3 bits of each color channel are red and put together to reassemble the original secret information.

CHAPTER 8

Nexim Two Extended Algorithm

> "The test of courage comes when we are in the minority. The test of tolerance comes when we are in the majority." – Ralph W. Sockman

The Nexim Two extended algorithm represents an enhancement to the original Nexim Two algorithm, offering an adaptive storage space that changes based on the size of the secret data we wish to embed.

The algorithm functions similar to a genetic algorithm, generating solutions in an iterative process and evaluating them. A solution issued by this algorithm is represented by a jump table combination. This jump table dictates the order in which to embed the data so that when embedded that payload produces the smallest amount of change to the original image. The jump addresses that form the jump table represent image region addresses where to store the secret data in a manner that alters the original color information by the smallest delta. The table itself is stored inside the image by using a small amount of the total available capacity for embedding. For better understanding, this algorithm can be separated into three steps: image segmentation, solution generation and solution embedding. As opposed to the original design, the solution generator stage also evaluates and quantifies the validity of the solution by measuring the delta difference between the original and modified version of the image in terms of peak-to-signal noise ratio (PSNR) applied on the RGB triplet. During the tests on the original design I identified rare cases in which a group of coarsely modified pixels where found within a small region of the image. Although overall the PSNR value indicated a small global change, the small region with saturated colors was easily identifiable using the human eye. These faults were generated because the RGB PSNR metric does not take into consideration the amount of light a color produces (luminance), but rater a logical average between the R, G and B channels. This observation led me to choose alternative color representations in order to quantify the color changes suffered after embedding the secret data.

The human eye is known to be very good at identifying light and color variations over a larger pixel area than distinguishing between light spikes inside a smaller area. This fact led my research into using YCbCr as an alternative color representation as it measures the luminance Y as being the amount of light as perceived by the human eye. As this representation is used only for measurement purposes the only channel adequate for PSNR measurements is the luminance channel. The quotients used for the Y channel are described below:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (8.1)$$

The luminance channel quantizes the light in relationship with its perceivable spectrum, with the green channel being the most important color component, leading to the most significant changes and variations if altered coarsely. The generator stage choses those solutions that produce the minimum Y-channel PSNR, therefor producing the minimum amount of lightness loss.

The current chapter also introduces a new variable embedding capacity, fitting the capacity to the storage needs. In the following paragraphs I will detail each step of the steganographic process as it passes the three stages: image segmentation, solution generation and embedding.

8.1 Image segmentation

In order to find alternative mapping strategies for embedding the secret data, the algorithm must have a series of pixel combinations to chose from. The image segmentation represents the preemptive preparation step, needed by the solution generator in order to choose the image regions that produce the smallest degradation after data embedding. First the cover image C (with an internal pixel matrix denoted as c_{ij}) is divided into 4 equal tiles (denoted Sx, where $x = 1..4$). For faster processing the pixel values corresponding to each of these tiles are stored within four separate data vectors (Sx) that handle each tile separately. Tiles are then cut into smaller portions in a process called partitioning. Just like in JPEG compression, the tiles are split into 8x8 blocks (64 pixels), which represent a relatively small visual group of pixels. Each tile consists in blocks of 8x8 pixels denoted $Sx.B\,[y]$ (where x represents the tile address and y represents the block number. In *figure 7.2*, an example image is shown having a size of 512x512 pixels. The tiling process would cut the image into four equal tiles, each sized at 256x256 pixels. The partitioning step would split each tile into 8x8 block, in this case building up 1024 blocks/tile.

As opposed to the original design where the data was inserted in one pixel at a time, this algorithm uses a 64-pixel buffer, for performance and optimization purposes, as later described in the paper.

This stage of the algorithm is intended to split the image into multiple pixel streams to choose from when trying to find the best embedding path. The tiling process covers a maximum of 4 available combinations. In addition to that I also introduced a third virtual direction that I call layer-ordering address that basically tells the order in which the data is inserted within a single pixel. Due to the fact that a color pixel is represented as a RGB triplet, the data can be inserted using different ordering of the color channels. Out of the total number of combinations possible (6) I only chose 3: RGB, GBR and BRG. With the use of this additional addressing the algorithm now has 12 embedding combinations to choose from throughout the generation process. A complete jump address is mapped on 4 bits, out of which the first two bits (MSBs) represent the tile address and the last two (LSBs) represent the layer ordering address.

8.2 Solution generation

This stage is a recursive method that generates the best embedding strategy in form of jump table entries. A table entry in the solution generator stage represents a two-value address that consists in a Level 1 Address (channel reordering address) and a Level 2 Address (tile address). The four tiles $S1..S4$ and the secret data D represent the main inputs of this stage, as shown in *Figure 7.3*. Each tile internally stores the current active pixel block $Sx.bi$ (active Block Index). The algorithm starts by reserving a certain number of blocks where the jump table will be stored. The number of blocks is always a multiple of 4 in order to assure a balanced distribution among the four image tiles. As opposed to the initial design where the data is streamed and analyzed one bit group at a time, this algorithm uses a larger buffer to increase the overall effectiveness of each choice over a larger area turning the once near-sighted generator into a far-sighted one in this design. There are two direct advantages of this approach: the algorithm optimizes more information because it tests the effect onto multiple cascading pixels and dramatically reduces the size of the jump table. In the original design optimizing 64 pixels with an embedding strength of 3 bits/pixel would require 192 table entries with an address length of 3 bits each, totaling 576 bits for the storage of the jump table alone. By using a 4 bit address in this case the same effect is achieved, meaning that the algorithm optimizes that same area as it's predecessor by utilizing 99,3% less storage space for the jump table. These changes not only improve the general effectiveness of the entire algorithm, but also render the evaluation stage from the previous design useless. The current algorithm is capable of finding better solutions easier and more rapidly and the jump table gets negligibly small in size.

8.2.1 Level 1 Address Generation

The four image tiles that stream the currently active block ($Sx.bi$) are divided into the individual color channels R, G and B. Based on these channels, for each active block of every tile, the algorithm determines the lightness component Y as reference lightness (RL). The RL is used to determine the alteration delta after the data has been embedded. The three-color channels are ordered as indicated above in RGB, GBR and BRG using three parallel reordering streams. Together with the secret data the algorithm virtually embeds (VE) the secret data onto the currently available block ($Sx.bi$) of each image tile (Sx). The embedding is simulated in order to determine the modified lightness (ML) of the resulting block in case that specific block is chosen for embedding. This part of the evaluation takes place on each tile and for each reordering sequence, totalizing 12 possible choices. Each tile picks out the reordering sequence that produced the minimum loss of luminosity in forms of PSNR (the greater the PSNR the higher the overall quality), reducing the 12 choices to just 4. The best reordering address (RA) is saved temporarily for the next step of the process in form of a Level 1 Address.

8.2.2 Level 2 Address Generation

In the second part of the process the algorithm picks out the tile that has the smallest degradation of lightness Y. Once this block is determined from the 4 remaining choices only one is picked and marked as being a solution, building the Level 2 Address. Together with the RA corresponding to the currently picked image tile, the Level 1 and Level 2 addresses are joined and added as a jump table entry.

The tile that produces the minimum amount of change will change its state to the next available free block ($Sx.bi++$), while all other tiles remain unchanged. Also, the data stream positions itself to the next 64 bits of data to be inserted.

The solution generator stage iterates for new solutions limited only by execution or iteration time limits, constantly searching for better solutions. One complete solution is considered as being obtained once the jump table has been filled. Since the size of the jump table has been reduced by 99,3% since the original design, we can use more table entries, allowing the algorithm optimize even larger portions of the image and therefore issuing better solutions than its predecessor.

8.3 Solution embedding

In this design, as opposed to the original approach, the algorithm does not finish once reaching the embedding stage. Depending on the size of the secret data one wishes to embed, the algorithm degrades the image step-wise and in a controlled matter. As shown in *Figure 8.1* the algorithm calculates the capacity per pixel depth (DCap) which is described as being the storage capacity of the image in 1 bit/color channel embedding and is described as being $width * height * 3$, where 3 represents the number of color channels (in this case R, G and B). Depending on the size of the secret data, the algorithm choses how many pixel LSBs to reuse. Based on the size of the secret data one wishes to embed, the algorithm can run a maximum of 4 generation-embedding cycles, one cycle corresponding to each layer. In *Figure 8.1* a 4-layer depth embedding is used, meaning that 4 LSBs of the cover image get replaced with the secret data bits.

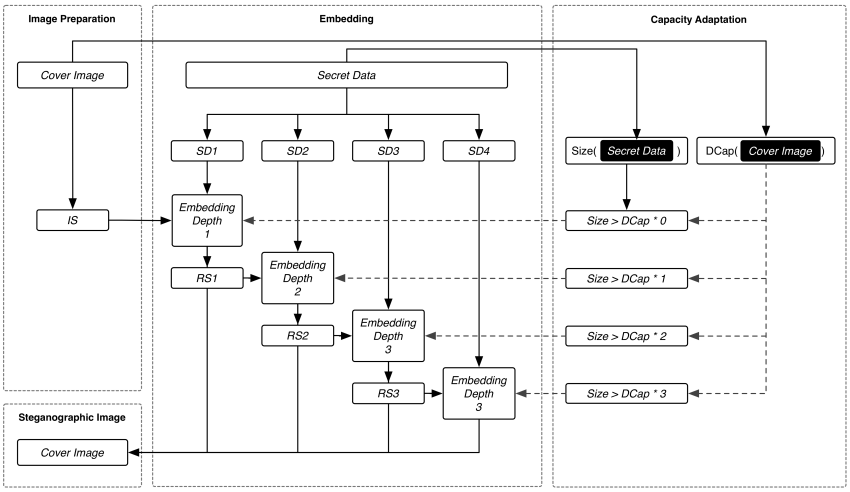

Figure 8.1 - Adaptive capacity

This stage starts by dividing the secret data into 4 parts (SD1..4), each sized at a maximum buffer of size DCap (the maximum amount of data a layer depth can store). The solution generator is called for embedding depth 1 and using the initial segmentation (IS) as prior described in this paper. Once the generator has

found the best embedding strategy the jump table generated gets stored inside the image along with the secret data SD1. The same process repeats itself for each additional storage depth. Once one layer has been filled, the algorithm calculates the PSNR for each image block and reorders (RS) the blocks inside an image tile from the mostly degraded to the least degraded. In this way the next algorithm iteration will prioritize the optimization of the coarsely degraded blocks. The algorithm finishes once the entire secret data is embedded along with its corresponding jump table.

CHAPTER 9

The Nexim SCS Infrastructure

In this chapter I propose an alternative communication system that uses cryptography (symmetric and asymmetric) and steganography to secure and hide the secret data, prior to sending it through insecure communication channels.

The design uses two encryption algorithms: AES and RSA. AES is used in order to obtain high encryption speeds in combination with long encryption passphrases. RSA is used to introduce one-way communication and even higher security by encoding the passphrase used in AES. As in the Public Key Infrastructure system (PKI) the joint usage of RSA and AES builds a strong encryption system characterized not only by high encryption speeds, but also by even greater overall security.

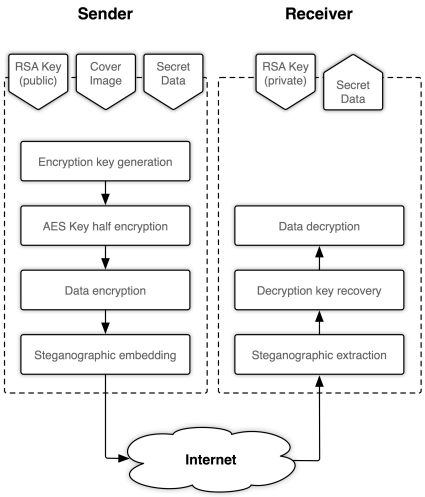

Figure 9.1 - Abstract representation of the communication system

The basic communication process used in this design can be defined as having three components: sending, transmitting and receiving, as shown in an abstract representation in *Figure 9.1*. The communication initiating party (sender) uses three inputs to prepare the data for transport: the secret data he wishes to transmit, a cover image that will store the data and the receiving party's public RSA key. The receiving party uses its private key and the transmitted steganographic image as its system inputs and extracts the secret data. For a better understanding

on how the system works from start to finish I will detail each step of the communication process in the following paragraphs.

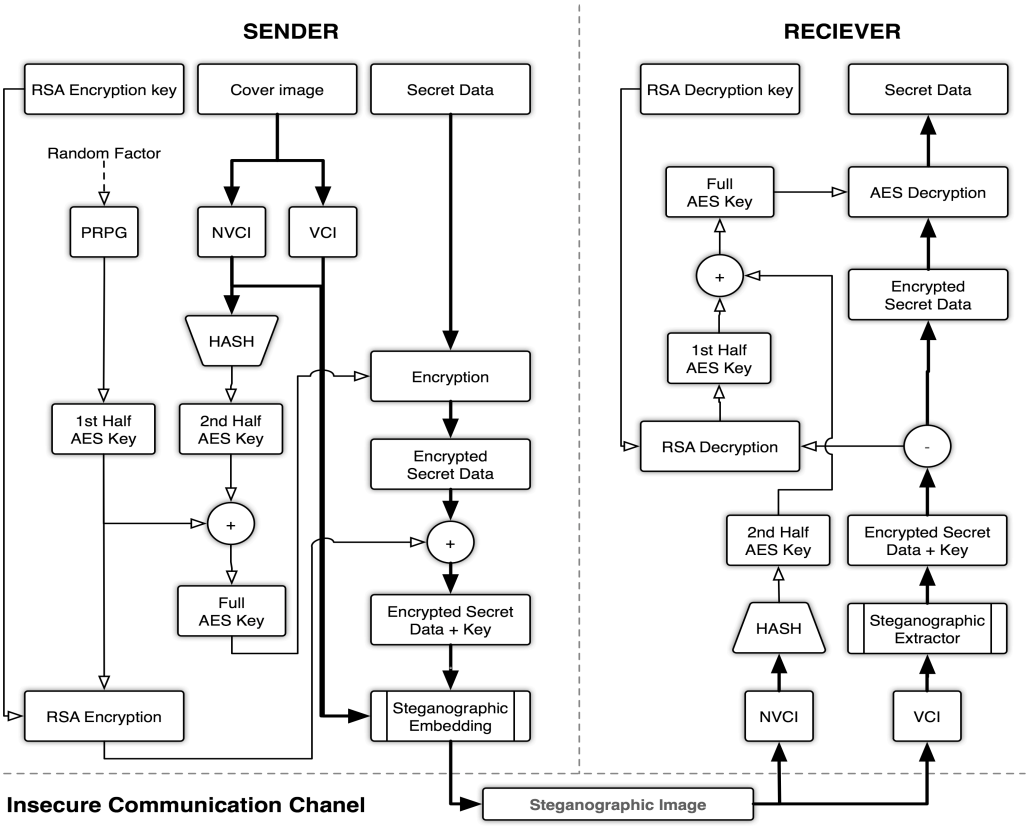

Figure 9.2 - Detailed representation of the communication system

9.1 Sending

In the sending stage of the communication process the secret data gets prepared to enter the transmission through an insecure communication channel (internet). The sending party selects the secret data to be sent along with a carrier that will store it (cover image). *Figure 9.2* illustrates the entire communication process. The carrier image is handled as two separate color information streams: the volatile and non-volatile part. The *Non-Volatile Color Information (NVCI)* stream is the part of the image that gets preserved throughout the entire steganographic embedding process, represented by the high-order bits of each color channel (RGB). The *Volatile Color Information (VCI)* stream represents the least significant bits (LSBs) of each pixel that get altered (replaced) with the secret data bits.

The system uses AES as primary encryption algorithm using 256 bit keys (the strongest alternative). Throughout the preparation process, four stages can be

distinguished: *encryption key generation, 1st AES Key half encryption, data encryption* and *steganographic embedding.*

9.1.1 Encryption key generation

In this system we make use of three keys: encryption key, public and private key. The encryption key is used to actually encode the secret data using the AES cipher. A strong typical 256-bit AES key can be seen in the following example:

Full AES key:

```
AF A7 06 A8 A7 2D 64 94 59 E7 5A 3C 89 2B 95 8C
B5 72 ED 5D 7E 38 9D 5F 8A 3C 52 53 41 AF 46 EB
```

The encryption key (256 bits) is generated with every new communication sequence, and consists in two parts: computated and random both having a 128-bit size, as shown in *Figure 9.3*. The random part (1st Half AES Key) of the key is generated using a pseudo-random pattern generator (PRPG) seeded with either variable or constant data. If the sender does not wish to use an additional password for the communication, the initial seed of the PRPG can use some variable system information data (mouse movement, time, clock cycle, RAM content, etc.). However, if the sender wishes to add further security, a password can be used as the initial seed for the PRPG. In this case both communicating parties must have the exact password in order to encrypt/decrypt the data. Because the sending party generates the random part of the encryption key internally, it is necessary to transmit it in an encrypted form, inside the data carrier.

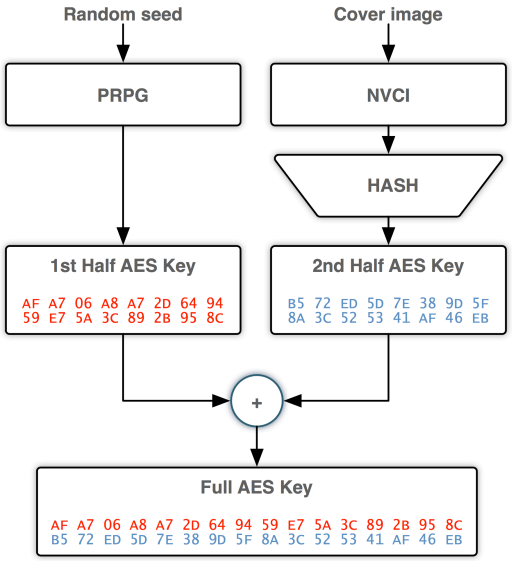

Figure 9.3 - Encryption key generator

The computed part of the encryption key (2nd Half AES Key) is calculated based on the non-volatile color information (NVCI). The high order bits of each color

channel (RGB in a color image) are hashed using a strong algorithm (for example SHA512, MD5). Only a portion of the hash value will be used in the 2nd Half AES Key. This component of the encryption key assures the authenticity of the transmission: if the carrier image gets replaced, the new image's NVCI will have a different hash than the original image, leading to a different encryption /decryption key and therefore, the original data cannot be extracted. This part of the encryption key is never stored; it must be generated on the fly in order to certify the authenticity of the transmission. The omission of this encryption key component from the transmission process is intentional, and it increases the reliability of the system against fault injection, faulty transmission or cover image changing.

The two parts are merged together into the main encryption key (Full AES Key). The full AES key is represented on 256 bits, offering highly secure encryption (NIST standard for military strength security).

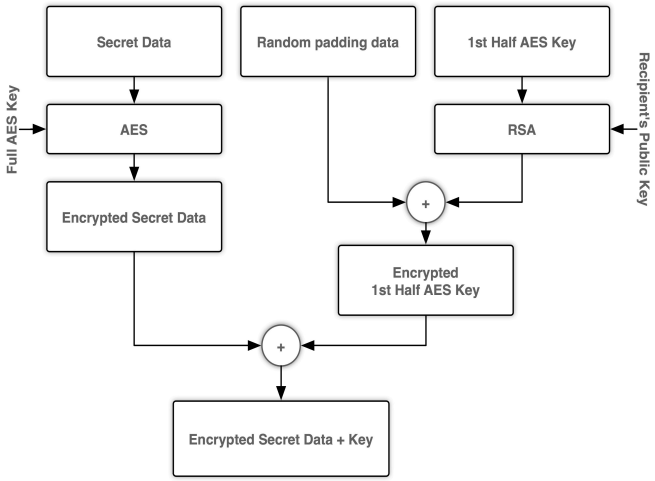

Figure 9.4 - Encryption process and key appending sequence

9.1.2 1st AES key half encryption

The random part of the encryption key is the one that gets transmitted over the insecure channel; therefore I use RSA as a strong asymmetric encryption algorithm. Using the recipient's public key, the random part of the encryption key gets encrypted and later appended to the raw encrypted secret data. By encrypting a part of the AES key using RSA, I add asymmetrical encryption to the process, limiting the possibilities of decrypting the data. Only the sender holds the right private key used to extract the key and decipher its secret data.

Full AES key with padding bytes (shown in green):

```
11 9A EF 3A 3F D0 78 2C 33 1B 51 BE 1A D0 20 27
05 71 5C 8F 19 D8 48 30 F6 54 D2 69 CC DE F9 FC
AF A7 06 A8 A7 2D 64 94 59 E7 5A 3C 89 2B 95 8C
B5 72 ED 5D 7E 38 9D 5F 8A 3C 52 53 41 AF 46 EB
```

In order to avoid reverse engineering using brute force attacks, given that the AES key is fixed in size, the system also injects random padding data at the beginning of the key (as shown in the example above). Only the recipient can successfully decode this key and ignore the junk data. The RSA encrypted key portion (encrypted 1st half AES Key) occupies 512 bits, after the 256 bits of random padding data have been added.

9.1.3 Data encryption

In this stage the system uses the AES algorithm to encrypt the secret data using the previously generated encryption key (Full AES Key). At the end of the raw encrypted data obtained in this step of the process, we add the previously encrypted 1st half AES key (512 bits), as shown in *Figure 9.4,* hereby forming the RAW data stream.

9.1.4 Steganographic embedding

Using my *smart LSB pixel mapping and data rearrangement* method, the encrypted secret data is embedded into the cover image. The output of the steganographic algorithm represents our data vault and consists in a plain modified image (cover image) with secret data bits embedded into it. This output is the one that leaves the sender system and travels through the insecure channel to its destination (recipient). Due to the high resistance of the steganographic algorithm to steganalysis-driven attacks, the detection rate for the data vault is very low, indistinguishable among other plain images without secret data embedded.

9.2 Transmitting

The steganographic image can securely travel through insecure communication channels regardless of the communication protocol used (email, stream, file sharing, P2P networks, etc.). The hidden content is hard to differentiate and identify, because images are among the most commonly transmitted media types on the Internet, making them a perfect cover for secret data communication.

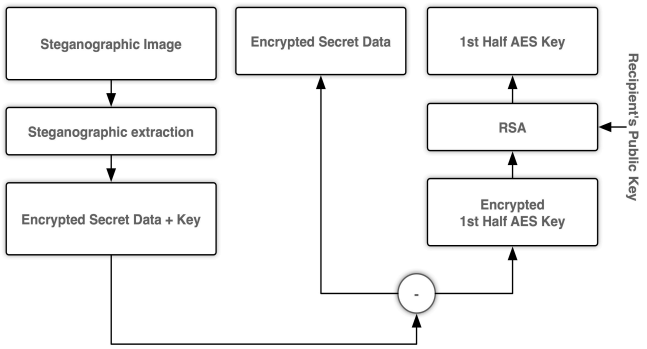

Figure 9.5 - Extracting the secret data and the 1st Half AES Key

Normally, during the process of transmitting data it is possible that the steganographic image travels through a series of filtering and detection checkpoints. When trying to intercept a steganography-aided transmission, the filters usually try to distinguish natural images from digitally altered and processed images. A natural

image represents an unprocessed, camera sensor captured image with high color representation accuracy and even chaotic noise distribution; whereas a digitally altered image introduces logic in the color representation (e.g. reducing the color palette, optimizing gradient transitions, sharpening contour edges, etc.). Depending on the degree of logic introduced by image enhancement and processing algorithms, steganalysis filters issue a suspicion rate measured in percentage on how high the chance is that the image contains secret data.

In the event that the transmitted data gets intercepted, it is hard to analyze if the image has data embedded onto it because of two main reasons: the NVCI part of the image has been left intact by the sender (as in almost all LSB-based steganographic algorithms) and because of the data bit optimization and rearrangement technique used in [1] the VCI part of the image (the part that actually stores the secret data) resembles the original image's VCI to a higher degree, leaving detection algorithms into issuing very low suspicion rates. In other words, steganalysis algorithms do not raise suspicion, because the steganographic image is identified as being a natural image. In addition to that, if the communication between the two parties does not always imply sending secret data, but instead also sending normal natural images, it becomes nearly impossible to intercept and successfully detect the exact image, out of the many sent images, that stores secret data inside.

9.3 Receiving

The receiving process is the reverse sending process. In this stage of the communication process, the steganographic image enters the receiver's processing zone. The receiving party uses the received data along with it's private key in order to extract the secret data, by reversing the steps needed to build the carrier image (steganographic image). The image is again handled as two separate color information streams: NVCI and VCI. The VCI part of the image is used to extract the data in its encrypted form. The NVCI part of the image is used to generate the 2nd half AES key exactly like in the sending process.

9.3.1 Steganographic extraction

The system processes the VCI part of the steganographic image in order to extract the secret data. Using the algorithm [1] the raw joint secret data and the encrypted portion of the AES key get extracted from the image, as shown in *Figure 9.5*. The extracted data is split into two streams: the encrypted secret data stream and the encrypted 1st Half AES Key stream.

9.3.2 Decryption key recovery

Using the RSA algorithm together with the receiving party's private key, the encrypted half of the AES key gets extracted. The previously added random padding bits are ignored and the system recovers the 1st Half AES Key.

The second half of the AES Key is calculated based on the NVCI part of the image. If the transmitted image was not altered in any way, using the same hash algorithm used by the sender, the system can successfully determine the 2nd Half AES Key. This step is crucial for authenticity verification, as the slightest change to the carrier image (cover image) makes the AES key unrecoverable, and therefore jeopardizing the integrity and authenticity of the transmission. The decryption key was intentionally designed as having two components. The first part of the key (the random generated part) is encoded using asymmetric cryptography in order to

introduce authentication, only the intended recipient can decipher it correctly. The second part of the key was designed to relate the cover image to the encryption process, offering security against unwanted changes. If the steganographic image has not been received intact, the system is unable to extract the data and signalizes a breach of security.

Once the two AES key halves have been successfully obtained/extracted, they are merged and the full AES Key is recovered.

9.3.3 Data decryption

The last step in the communication process represents the decryption of the secret data. Once the AES key has been retrieved, the secret data can be successfully decrypted and the receiving party receives the intended secret data, ending a successful communication.

CHAPTER 10

Experimental results and evaluation

> "A fact is a simple statement that everyone believes. It is innocent, unless found guilty. A hypothesis is a novel suggestion that no one wants to believe. It is guilty, until found effective." – Edward Teller

In order to prove the effectiveness of the algorithms presented in this thesis I conducted several tests. The tests presented in this chapter cover a large variation of quality and performance indicators that helped differentiating my algorithms from others in the field. The algorithms I've designed feature an advanced modularity that serves both as an abstract model for future improvements but also for advanced benchmarking because of the multitude of testing points that can be measured during the algorithm's execution. Because of the high complexity in design, I've given much attention to optimizing every component of the algorithms in order to achieve maximum performance in both quality and execution type. The current chapter presents the experimental setup and results in comparison with other state of the art algorithms.

10.1 The experimental setup

The experimental setup is divided into two parts: basic and advanced tests. The basic test outlines the performance of the original algorithms presented in comparison with the basic LSB steganography method, but also against one of the most powerful state of the art algorithms to date: Optimal LSB (OLSB) [39] and Optimal Pixel Adjustment Process (OPAP) [40]. In order to validate the tests I've used the same testing environment, as did the authors of the above-mentioned algorithms. The advanced tests represent a more elaborated and performance oriented set of tests that compare the algorithms introduced in the current thesis, over a series of variables that influence the outcome of the algorithm.

10.1.1 Basic tests

The basic test was conducted on the standard set of images: Lena, Baboon, Jet and Peppers (as shown in *Figure 10.1*) sized at 512x512 pixels. These four images have become a standard in steganographic benchmarking, as it is easier to differentiate and highlight the advantages of a new algorithm over another by using

a common starting point, which is the cover image. The basic test is performed on the images in *Figure 10.1* using 100 randomly generated secret data streams, and an average PSNR over all 100 tests result is calculated. This result represent the actual comparison metric that is used in order to differentiate my own designs against one of the most powerful algorithms to date: OPAP and OLSB.

(a) Lena

(b) Baboon

(c) Jet

(d) Peppers

Figure 10.1 – The four standard images used for basic tests

10.1.2 Advanced tests

In my extended test I've used 100 full HD images (1920x1080 pixels) with a randomly generated payload data (occupies 90% of the total steganographic capacity) and with jump table size that does not exceed 10% of the total steganographic capacity the carrier medium offers. The tests have been conducted over a longer timespan (8 hours of execution time), allowing the algorithms to find the best solution. The experiment also highlights the evolution of the algorithm over time in terms of best solution a general tendency.

10.2 Evaluation approach

In LSB matching steganography one of the most commonly used methods for output quality measurement is peak signal-to-noise ratio (PSNR). Given that a steganographic algorithm takes a normal image and transforms it in the process of embedding the secret data within, the algorithms outputs a modified image. In the ideal case the outputted image resembles the original from a visual and digital perspective. Since the secret data introduced by the algorithms is basically noise, the similarity of the images is calculated using the mean squared error (MSE) formula:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$where$$

$$m = image\ height$$

$$n = image\ width$$

$$I(x,y)\ and\ K(x,y) = \ image\ pixel\ matrix$$

MSE quantifies the changes suffered due to the embedding process, counting the alterations suffered according the degree of deviance between the original and modified value. Using this function finely modified pixels have a smaller weight than coarsely modified ones. The usage of MSE is limited to image sets of the same size (number of effective pixels), making it unsuitable as a general metric that can be applied independent of the size of the image.

The general metric for noise between two signals, in this case the original and modified pixels of the image, is measured in dB. The PSNR is directly related to the MSE function, the difference being that PSNR measures the noise in dB, without taking the effective number of pixels an image uses into consideration. PSNR is defined as being:

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

The MAX_I represent the absolute binary value a pixel can store. In the case of a high-color pallet where a pixel is stored on 24 bits, each channel has a maximum combination of 255, being represented on 8 bits.

PSNR applied between two identical images is infinite, representing an exception in the above formula. This metric is mostly used in lossless compression algorithms as it quantifies the alteration suffered by the compressed image in relation with the original. PSNR values between $30\ dB$ and $50\ dB$ are considered high quality approximations of the original, where the higher the PSNR the more the modified resembles the original. A typical high quality JPEG image obtained by compressing an image using the quality factor $Q = 90\%$ usually issues a PSNR value of over $40\ dB$.

10.3 Experiments

10.3.1 Basic tests for Nexim One

The testing process for Nexim one has been structured in two stages. Since Nexim one was constructed in a modular way with various parameters to tweak, in the first stage I determined the optimum parameters for the embedding process to later use them in a comparative analysis against other algorithms in the second stage of my experiment.

Each test conducted was made using 4 different volatile bits/pixel values (LSBs): 1,2,3,4 bits/pixel. The maximum steganographic capacity in each of the four cases is presented in *Table 10.1*. Since all images have the same size in pixels and color depth, each image has the same total capacity, depending on the number of volatile bits used.

LSBs	1 LSB	2 LSB	3 LSB	4 LSB
Capacity	262144	524288	786432	1048576

Table 10.1 – Capacity for the four test images

For each of the four embedding depths above, four random streams where generated using a PRNG (Pseudo Random number Generator), represented in a binary form with values between 00-FF (hex). In order to assure a less time consuming algorithm execute the tile multiplier was limited according to the embedding depth:

- 0 to 7 for 1 bpp
- 0 to 6 for 2 bpp
- 0 to 5 for 3 bpp
- 0 to 4 for 4 bpp

For each of the above combinations, different jump table multipliers have been tested: 0,2,4,8,16,31,64 and 128. The jump table multiplier was adjusted so that the storage of the table itself would require at most 5% of the total available steganographic capacity. The algorithm was configured to iterate 50 times (per embedding process) in order to find the best solution. The best results of this test can be observed in *Table 10.2.*

LSBs	Tiling multiplier	Tile number	Jump table multiplier	**PSNR**
1	2	8	32	**55.9436**
2	3	32	16	**48.2192**
3	4	128	16	**42.5083**
4	4	256	8	**36.7142**

Table 10.2 – Optimal parameter test results

An essential aspect outlined by this first tests is that increasing the number of LSBs leads the algorithm requiring more tiles to chose from in order to provide

better results. A larger tile number also raises the number of elements in the jump table, thus requiring a smaller jump table size multiplier.

The most important part of this test was the second stage which used the parameterization defined above to test the algorithm's performance against other state of the art algorithms (the most powerful to date): OPAP and OLSB.

As opposed to all other techniques, this algorithm also embeds the table into the secret data. For comparison reasons, therefore the table was appended without any additional enhancements done to the resulting image.

Table 10.3 illustrates the results in terms of PSNR of the various combinations. *Table 10.4* presents the gain obtained using Nexim One in relationship with the PSNR values obtained with the other 3 methods.

LSBs	SLSB	OLSB	OPAP	**Nexim One**
1	51.1335	51.4125	53.3104	**55.9436**
2	44.0167	44.3650	46.1571	**48.2192**
3	37.9402	38.2194	40.2401	**42.5083**
4	31.6843	31.9907	34.0843	**36.7142**

Table 10.3 – Performance comparison (Nexim One) [dB]

LSBs	SLSB	OLSB	OPAP
1	4.8101	4.5311	2.6332
2	4.2025	3.8542	2.0621
3	4.5681	4.2889	2.2682
4	5.0299	4.7235	2.6299

Table 10.4 – Gain comparison (Nexim One) [dB]

As one can observe, Nexim One has an overall advantage over OLSB of +2 dB in PSNR. The output of this experiment highlights the performance of this method, which succeeded in preserving more original information with the penalty of losing a small amount of steganographic capacity. The maximum capacity loss registered was 5% of the total embedding capacity, which represents a small penalty in favor of overall quality of the outputted steganographic image.

10.3.2 Basic test for Nexim One Extended

The Nexim One algorithm did not require any pre or post processing stages in order to enhance the quality of the output image. This however had one drawback; in some isolated cases the algorithm produced "brightness spots" on the carrier image, meaning an accumulation of modified pixels that presented either higher or lower brightness than the rest of the image. These spots did not affect the overall PSNR and logical distribution of the data, but failed due to the inconsistency with the human vision analysis. Nexim One Extended represents a relative small addition to the original design that focuses entirely in eliminating the bright spots but also their initial cause of appearance. The tests were conducted using the same randomly generated data streams presented in the previous chapter. The results are shown in *Table 10.5.*

LSBs	SLSB	OLSB	OPAP	Nexim One	**Nexim One Extended**
1	51.1335	51.4125	53.3104	*55.9436*	**56.1653**
2	44.0167	44.3650	46.1571	*48.2192*	**49.4301**
3	37.9402	38.2194	40.2401	*42.5083*	**42.9735**
4	31.6843	31.9907	34.0843	*36.7142*	**36.8062**

Table 10.5 – Performance comparison (Nexim One Extended) [dB]

As one can observe in *Table 10.6*, the additions introduced by Nexim One Extended changes successfully raise the overall quality of the steganographic image, retaining more color information than the original Smart LSB algorithm.

LSBs	SLSB	OLSB	OPAP	**Nexim One**
1	5.3180	4.7528	2.8549	**0.2217**
2	5.4134	5.0651	3.2730	**1.2109**
3	5.0333	4.7541	2.7334	**0.4652**
4	5.1219	4.8155	2.7219	**0.0920**

Table 10.6 – Gain comparison (Nexim One Extended) [dB]

If we take a look at the gain of the new method (*table 10.6*) over the references used, we can see that the effectiveness of the new changes in the original algorithm design offer more quality when using 2 volatile bits. From the quality point of view, the improvements finely tune the original Smart LSB algorithm in terms of PSNR.

With the inclusion of a better solution generator, this design initially picks better solutions than the original algorithm, reducing therefor the time needed for obtaining equal or better solutions. Throughout the testing process we registered a speed increase of 1.7X ±0.3 compared with the original design (Nexim One).

The steganographic images successfully passed several steganalysis filters (RS stego, Weighted Masks and RD steganalysis) issuing no alarm at all, situating the average suspicion rate in normal boundaries (7-21%). In other words, using the new improvements this algorithm successfully hides the secret data in a manner that preserves enough of the original color information to pass detection filters, therefor fulfilling the main purpose of the algorithm.

10.3.3 Basic test over Nexim Two

The Nexim Two algorithm is the first algorithm that eliminates the weakest modules in the original design. The combined solution generator with the solution evaluator, as well as the new fixed tiling and region-based PSNR analysis facilitated a reduced execution complexity and at the same time surpassing the previously introduced Nexim One and it's extension, as observed in *Table 10.7.*

SLSB	OLSB	OPAP	Nexim One	Nexim One Extended	**Nexim Two**
37.9402	38.2194	40.2401	42.5083	42.9735	**44.3701**

Table 10.7 – Performance comparison (Nexim Two) [dB]

The new and improved algorithm registered higher average PSNR values in all of the four testing images, with a gain of +1.39 dB over Nexim One and +1.86 dB over Nexim One Extended. The gain in quality translates into higher color conservation in the embedding process, even when the payload remains the same

throughout all algorithms. The only limitation of this method in comparison with it's predecessors is that it only deals with an embedding depth of 3 LSBs/color component.

10.3.4 Basic test over Nexim Two Extended

The last introduced algorithm, Nexim Two Extended, represents an extension to Nexim Two, that introduces a novel adaptive capacity that choses the embedding depth needed according to the size of the secret data one wishes to embed. In comparison with the other variations of Nexim One and Nexim Two, this algorithm is fully automated, eliminating many of the variables that were otherwise manually set or using benchmarks in order to determine the optimum set of parameters. Based on the principles on Nexim Two, the additions I made to this algorithm eliminates the initial 3 LSB constraint, but retaining in the same time the improved solution generator and evaluator and the higher processing speed due to the pixel grouping process. The tests have been conducted on the same set of secret data streams as in all other cases to have a valid departing metric between the algorithms.

LSBs	SLSB	OLSB	OPAP	Nexim One	Nexim One Extended	Nexim Two	**Nexim Two Extended**
1	51.1335	51.4125	53.3104	*55.9436*	56.1653	-	**56.1503**
2	44.0167	44.3650	46.1571	*48.2192*	49.4301	-	**49.1654**
3	37.9402	38.2194	40.2401	*42.5083*	42.9735	44.3701	**44.4951**
4	31.6843	31.9907	34.0843	*36.7142*	36.8062	-	**37.0864**

Table 10.8 – Performance comparison (Nexim Two Extended) [dB]

As can be observed in *Table 10.8* the Nexim Two Extended algorithm issues similar PSNR values in the case of 1 and 2 LSBs, but successfully surpasses all other algorithms when it comes to preserve the color information while coarsely modifying 3-4 LSBs/color components. *Table 10.9* illustrates the gain over the other algorithms.

LSBs	SLSB	OLSB	OPAP	Nexim One	Nexim One Extended	**Nexim Two**
1	5.0168	4.7378	2.8399	0.2067	-0.0150	**-**
2	5.1487	4.8004	3.0083	0.9462	-0.2647	**-**
3	6.5549	6.2757	4.2550	1.9868	1.5216	**0.1250**
4	5.4021	5.0957	3.0021	0.3722	0.2802	**-**

Table 10.9 – Gain comparison (Nexim Two Extended) [dB]

The table above outlines that the algorithm is less effective than its predecessors for finely modified images (1-2 LSB), but improves the quality even more in the case of 3 LSB. The algorithm loses an average of 0.3 dB on fine modified images, but when compared to the benefits that come from the shrined jump table and high speed processing, this is by far the strongest algorithm I've introduced.

10.3.5 Advanced tests on all Nexim algorithms

The two algorithms and their extensions have been tested over a longer time span in order to highlight their behavior over time:

- Time it takes to find the best solution
- Completed iteration cycles

The basic tests conducted have shown that all 4 algorithms are most effective when the embedding depth it 3 LSBs/color component, which places the algorithms in the domain of high capacity steganographic algorithms using LSB matching techniques. The current experiment has been conducted over a timespan of 8 hours for each of the four algorithms, totalizing 32 hours of execution time for all Nexim series algorithms. For the ease in writing the four algorithms presented will be denoted NX1 (Nexim One), NX1e (Nexim One Extended), NX2 (Nexim Two) and NX2e (Nexim Two Extended).

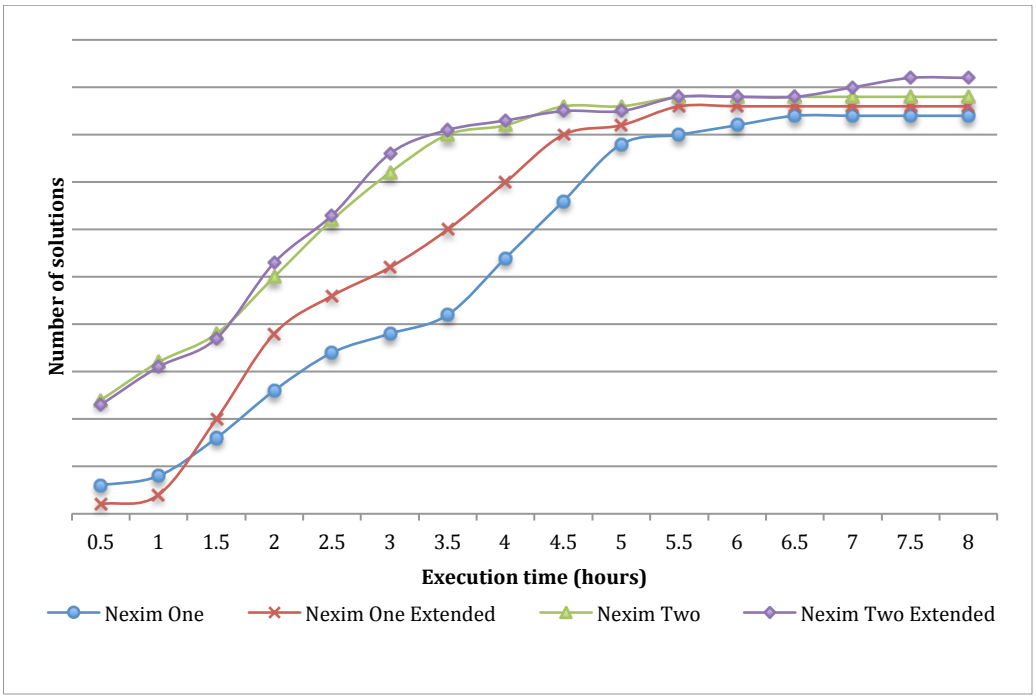

Figure 10.2 – The four standard images used for basic tests

Figure 10.2 outlines the relationship between the execution time and quality of the best solution found so far. All algorithms have a linear tendency of finding better solutions in the first 5 hours of execution. However, NX1 and NX1e, because of their weaker solution generator and evaluator issue lower quality results in the first 4 hours, starting with a very high alteration rate compared to NX2 and NX2e which find better solutions within the first half hour of execution time.

The nonlinear evolution of the solutions is a direct effect of the many embedding strategies that can be found. Another run of the same experimental setup but using different random streams can register small changes, but the general tendency of solution quality over the 8-hour execution time remains similar to the current experiment's results.

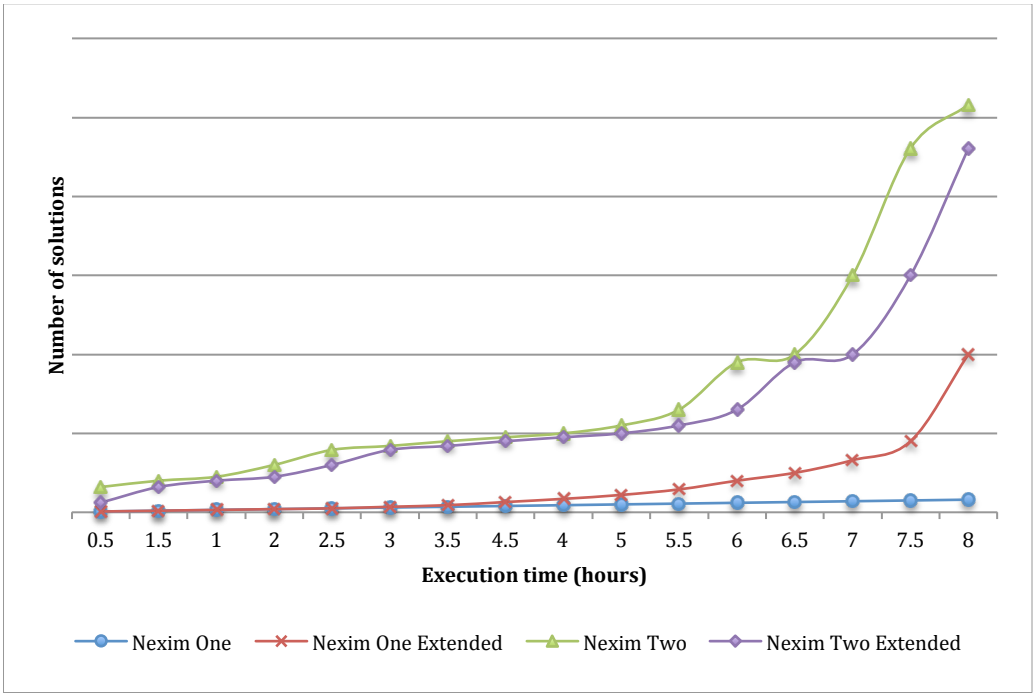

Figure 10.3 – The four standard images used for basic tests

The iteration count evolution over the 8-hour period is shown in *Figure 10.3*. NX1, because of its weak solution evaluator has a constant and linear growing iteration count. Even if no better solutions are found, this algorithm continues to test even worse solutions. NX1e on the other hand, with its HDR filtering techniques can filter bad solutions, increasing the overall iteration counts. Still, NX1e due to its computational intensity can only filter the best solutions after a certain breed of solutions have been generated.

NX2 and NX2e facilitate the highest iteration count, being able to filter bad solutions in the incipient phase, before the evaluator quantifies the entire solution. Solutions that would imply a lower PSNR than the best found so far are immediately marked as bad and the algorithm breaks the execution of the current iteration moving to the next one. The experiments have shown that both NX1 and NX1e have a solution drop rate of over 93% whereas the enhanced NX2 and NX2e have a drop rate of 61%, meaning that the algorithm drops less solutions and is able to find constant evolving solutions over a smaller timespan.

10.4 Overall result evaluation

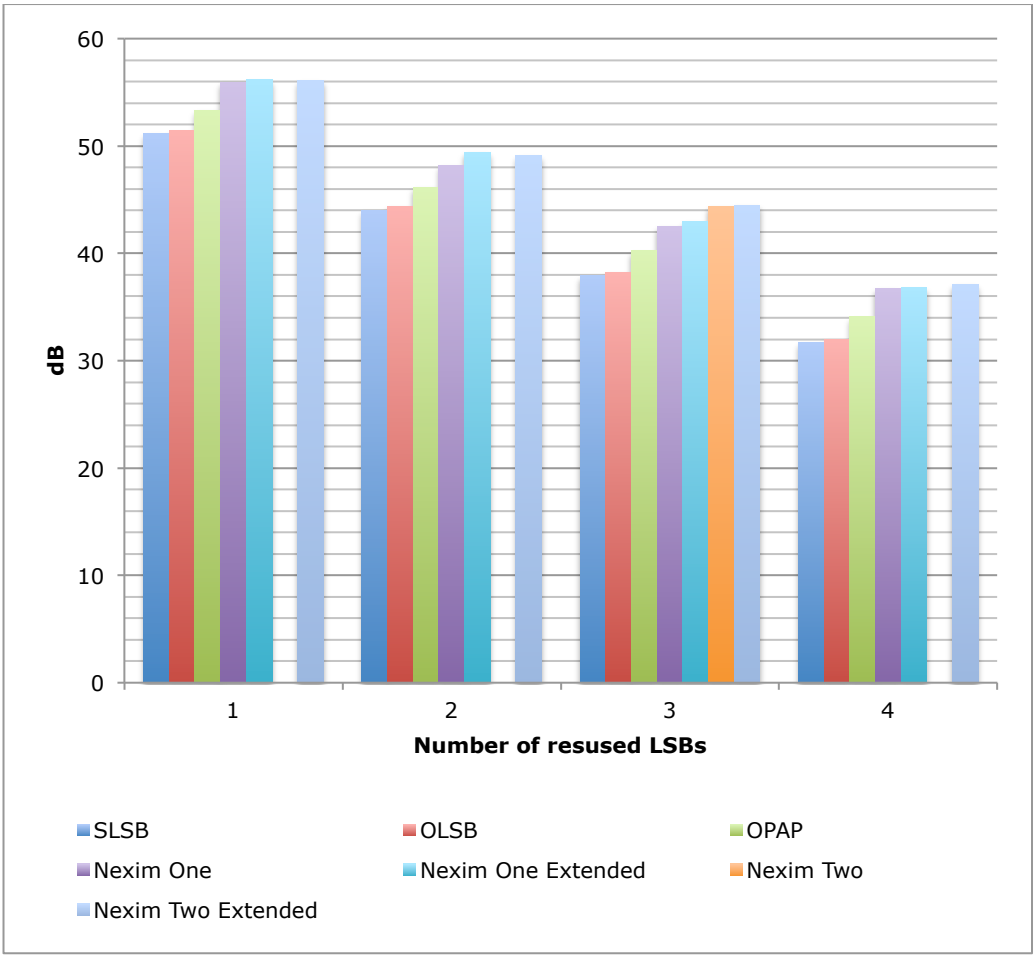

Figure 10.4 – General comparison of output quality

The experiments conducted and their results have shown the incremental quality updates each of the Nexim algorithm series brings. Whereas Nexim One was strong the initial design that proves strong against other state of the art algorithms like OPAP and OLSB, the later extensions and redesign introduced in the Nexim algorithms series have proven to be more effective in both overall resulting quality, processing speed, resource consumption and even increased resistance to steganalysis attacks. To conclude the experimental results, this thesis introduces a new set of powerful LSB matching algorithms that fulfill the both quality and capacity requirements to a degree that has not been achieved so far. The design of the four algorithms facilitates future improvements and tweaks in the entire steganographic process, serving both as standalone implementations and design guideline for future additions.

CHAPTER 11

Conclusions and perspectives

> "Imagination is not only the uniquely human capacity to envision that which is not, and therefore the fount of all invention and innovation. In its arguably most transformative and revelatory capacity, it is the power to that enables us to empathize with humans whose experiences we have never shared." – J. K. Rowling

The continuous evolution of computing systems and global networks pushes the digital world to a completely new level. We tend to be more inter-connected each day, computers are almost everywhere in our lives. Almost every aspect of modern communication is a synonym with Internet. Along with the tremendous benefits technology offers, comes an even bigger risk of privacy. Cryptography alone is not enough, since the present security standards rely on algorithms that were developed over 2 decades ago, at the time when breaking or solving the high complexity mathematical formulas on which ciphers relay seemed a task impossible to achieve in the next millennium. However, statistics have shown the opposite: the evolution of both processing power and hardware implementation at even smaller scales has not been foreseen as possible.

The current thesis follows the new trend of data through obscurity: steganography. With this work I address an important aspect of digital security that, although still under development, is an emerging technology that is adopted each day in many situations where cryptography alone is insufficient.

The vastly unexplored domain of steganography lacks of standardization or certain rules, many researchers that are fascinated by this technologic breakthrough in data security fail to see the applications that it can support.

11.1 Thesis impact and contributions

With the current thesis I wish to complete this research domain with the results of over 5 years of work in the steganographic domain. Throughout these years I've developed several algorithms that vary in scope, performance and design, out of which I've picked 4 of the strongest and best suited algorithms as my contribution to the field of digital image steganography.

The current thesis introduces 4 LSB matching algorithms: Nexim One, Nexim One Extended, Nexim Two and Nexim Two Extended, as well as an advanced

security infrastructure used for secure digital communication using steganography and cryptography in an original way.

The four algorithms I've introduced in this thesis serve for two purposes: a powerful stand-alone steganographic algorithm and a general design template for future improvements and changes. During my research I've studied several algorithms and analyzed their structure, performance and design. This observation led me into building an abstract representation of a steganographic algorithm, guideline that I also used in the process of designing the four algorithms presented.

The Nexim series was introduced at several international conferences and was very well received by the audience, the first algorithm introduced Nexim One already has been analyzed and used as a reference by recent steganography researchers.

In the process of designing the four algorithms I've also given much attention to the physical implementation of the algorithms and their optimization. Therefor I've developed the algorithms as both stand-alone solutions, but also a benchmarking tool to test their performance and better analyze their weaknesses, faults and obsolete internal components.

Given these facts, the main contributions of this thesis are:

- Nexim One and Nexim One Extended – two new steganographic hybrids that are capable of preserving the original image better than other algorithms.

- Nexim Two – representing the most powerful and fast algorithms so far, even compared with the original two designs.

- Nexim Two Extended – representing a big improvement to Nexim Two, adding the flexibility of variable storage space to the already high quality output after the embedding process took place.

- Nexim SCS (Secure Communication System) – the abstract communication system that hides the already cryptographically secured information inside harmless carriers such as image, in order to prepare the sensitive information to travel unnoticed on the internet in the case of an interception or man-in-the-middle attack. The new communication model features increased reliability because it covers key aspects in terms of data security:

    - Unidirectional encryption system (the sender can encode, but only the receiver can decode)

    - Authenticity verification for both cover image and secret data (several checkpoints have been intentionally added so that even the smallest change in the steganographic packet breaks the seal, signalizing an unauthorized manipulation)

    - Chained cryptographic systems (AES + RSA)

    - Unidentifiable communication stream (the image containing secret data cannot be distinguished among other normal images)

- The joining of cryptography and steganography builds up a near perfect communication system, presenting higher reliability than stand-alone cryptographic methods. By combining the security level offered by cryptographic cyphers (data encoding) with the stealth introduced by steganographic encoders (data hiding), it is nearly impossible to breach this system

11.2 Supporting publications

The current thesis is supported by the following list of papers published at international conferences:

- **S.F. Mare**, M. Vladutiu, L. Prodan, "Decreasing Change Impact Using Smart LSB Pixel Mapping and Data Rearrangement", Proceedings CIT 2011 "The 11th IEEE International Conference on Computer and Information Technology", Paphos, Cyprus, August 2011, pp. 269-276, ISBN: 978-1-4577-0383-6 (IEEE Computer Society rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "HDR based steganographic algorithm", Proceedings SIITME 2011 "IEEE 17th International Symposium for Design and Technology of Electronics Packages", Timisoara, Romania, October 2011, pp. 333-338, ISBN: 978-1-4577-1276-0 (IEEE rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "Secret data communication system using steganography, AES and RSA", Proceedings SIITME 2011 "IEEE 17th International Symposium for Design and Technology of Electronics Packages", Timisoara, Romania, October 2011, pp. 339-344, ISBN: 978-1-4577-1276-0 (IEEE rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, "High capacity steganographic algorithm based on payload adaptation and optimization", Proceedings SACI 2012 "IEEE 7th International Symposium on Applied Computational Intelligence and Informatics", Timisoara, Romania, May 2012, pp. 87-92, ISBN: 978-1-4673-1013-0 (IEEE, Australian Research Council list class C rank).

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Advanced Steganographic Algorithm Using Payload Adaptation and Graceful Degradation", Proceedings ICIE 2012 "International Conference on Information Engineering", LNIT "Lecture Notes in Information Technology Vol.25" (IERI Press), Singapore, Singapore, June 2012, pp. 121-127, ISBN: 978-1-61275-024-8 (Ei Compendex, Cambridge Scientific Abstracts, Google Scholar, IEE, ISI rank).

The current thesis was also defended using two PhD reports sustained at my local university:

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Algorithm Concepts and Ways of Building More Robust, Stronger and Stealthier Steganographic Algorithms", Ph.D. Report 1, "Politehnica" University of Timisoara, January 2012, pp. 1-57

- **S.F. Mare**, M. Vladutiu, L. Prodan, F. Opritoiu, "Algorithm Concepts and Ways of Building More Robust, Stronger and Stealthier Steganographic Algorithms", Ph.D. Report 2, "Politehnica" University of Timisoara, July 2012, pp. 1-84

11.3 Future directions

Although this thesis deals with many aspects related to digital image steganography and the ways in which we can achieve both quality and storage capacity, there is still room for future improvements and questions worth future investigation.

11.3.1 Algorithm Refinements

- Improving the matching algorithm – this represents the key point in LSB matching steganography. Because I've focused into creating algorithms that do not rely at all on pre or post processing image enhancement filters, one could improve the quality of my current designs with the help with some advanced filtering and image enhancement stages.

- Parallelized implementation – represents one of the most adequate benchmarking solutions, because one could test the current algorithms over a wider set of inputs and analyze the output faster and get results and quality metrics that hopefully can identify weak points until the algorithm gets adopted in some situations.

- Improved reliability – even if this is a hard task to achieve in the case of steganography, the reliability of the algorithms can be enforced with the insertion of error correction algorithms.

- Data compression algorithms – built into the steganographic process with a cross-linked dictionary could shrink the secret data size, therefor requiring less coarsely modified color bits.

11.3.2 Security Infrastructure Refinements

- Improving the communication infrastructure with the combination with digital certificate standards in order to enforce the authenticity an uniqueness of a transmission.

- Hardware implementation of the entire system for both cryptographic and steganographic, facilitating rapid encoding and decoding.

- Large scale simulation of the effectiveness of the infrastructure and the insertion of steganographic detectors and filters.

Bibliography

[1] B. Clair, "Steganography: How to Send a Secret Message", Strange Horizons, October 2001

[2] L. Kleinrock, "Personal History/Bibliography: the Birth of the Internet", University of California Articles, 2009

[3] K.G. Coffman, A.M. Odlyzko, "Growth of the Internet", AT&T Labs – Research, pp. 1-44, July 2001

[4] "Mobile Broadband Explosion", Rysavy Research/4G Americas, September 2011

[5] "The Mobile Broadband Evolution: 3GPP Release 10 and Beyond – HSPA+, SAE/LTE, and LTE Advanced", Rysavy Research/4G Americas, February 2011

[6] C. Harvey, "The Smartphone is the New PC", Processor.com Featured Article, pp. 29, June 2011

[7] P.G. Neumann, "Computer Insecurity", SRI International, pp. 50-54, 2003

[8] P.G. Neumann, "Computer-Related Risks", ACM Press/Addison-Wesley, 1994

[9] R. Andersen, "Security Engineering: A Guide to Building Dependable Distributed Systems", University of Cambridge Computer Laboratory Paper, Chapter 5 – Cryptography, pp. 73-114

[10] J. Daemen, V. Rijmen, „The Design of Rijndael: AES – The Advanced Encryption Standard", Springer, 2002

[11] R. Rivest, A. Shamir, L. Adelman, „PKCS #1 v2.1: RSA Cryptography Standard", RSA Laboratories, 2002

[12] I. Mironov, "Hash functions: Theory, attacks, and applications", Microsoft Research, November 2005

[13] U. Meyer, S. Wetzel, "A Man-in-the-Middle Attack on UMTS", The Fifth International Conference on Web Information Systems Engineering, pp. 90-97 Brisbane, Australia, November 2004

[14] R. Saltzman, A. Sharabani, "Active Man in the Middle Attacks: A security advisory", IBM Rational Application Security Group, February 2009

[15] P. Burkholder, "SSL Man-in-the-Middle Attacks", InfoSec – Information Security Reading Room, 2002

[16] N. Asokan, V. Niemi, K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", Nokia Research Center, pp. 1-15, Finland, 2002

[17] T. Shorter, "A "real-life" Man-in-the-Middle Attack on SSL", RSA Conference 2005, San Francisco, California, 2005

[18] C. Hepner, E.Zmijewski, "Defending against BGP Man-In-The-Middle Attacks", Black Hat DC 2009, Arlington, Virginia, February 2009

[19] W. Bender, D. Gruhl, N. Morimoto and A. Lu, "Techniques for data hiding", IBM Systems Journal, Vol. 35, No. 3&4, pp. 313-336, 1996

[20] E. Cole, R. D. Krutz, "Hiding in Plain Sight: Steganography and the Art of Covert Communication", Wiley Publishing Inc., 2003

[21] J. C. Judge, "Steganography: Past, Present, Future", InfoSec – Information Security Reading Room, 2001

[22] E. Zielinska, W. Mazurczyk, K. Szczypiorski, "The advent of Steganography in Computing Environments", Article from Cornell University Library, February 2012

[23] B. Pfitzmann, "Information hiding terminology", Information hiding: first international workshop, Cambridge, England, May 1996

[24] F. A. P. Petricolas, R.J. Anerson, M. G. Kuhn, "Information Hiding – A survey", Proceedings of the IEEE: special issue on protection of multimedia contend, July 1999

[25] G. Fisk, M. Fisk, C. Papadopoulos, N. Joshua, "Eliminating Steganography in Internet Traffic with Active Wardens", 5th International Workshop on Information Hiding, pp. 18-35, Noordwijkerhout, The Netherlands, October 2002

[26] C. -C. Chang and H. –W. Tseng, "Data Hiding in Images by Hybrid LSB Substitution", Third International Conference on Multimedia and Ubiquitous Engineering, pp. 360-363, 2009

[27] A. J. Raphael, V. Sundaram, "Cryptography and Steganography – A Survey", International Journal of Computer Technology and Applications Vol. 2, pp.626-630, May-June 2011

[28] M. R. N. Torkaman, N. S. Kazazi, A. Rouddini, "Innovative Approach to Improve Hybrid Cryptography by Using DNA Steganography", Second International Journal on New Computer Architectures and Their Applications, pp.225-236, 2012

[29] P. Vitthal, B. Rajkumar, P. Archana, "A Novel Security Scheme for Secret Data using Cryptography and Steganography", Computer Network and Information Security, pp.36-42, 2012

[30] M. K. Goel, N. Jain, "A Novel Visual Cryptographic Steganography Technique", International Journal of Computer, Electronics & Electrical Engineering, pp.39-43, 2012

[31] P. Reddy, S. Kumar, "Steganalysis Techniques: A Comparative Study", University of New Orleans Theses and Dissertations, 2007

[32] N. F. Johnson, S. Jajodia, "Exploring Steganography: Seeing the Unseen". IEEE Computer, pp. 26-34, February 1998,

[33] S. Lyu and H. Farid, "Detecting Hidden Messages Using Higher-Order Statistics and Support Vector Machines", 5th International Workshop on Information Hiding, Noordwijkerhout, The Netherlands, 2002

[34] B. Morency, "Interesting facts about email attachements", Google DokDok Statistics, September 2010

[35] R. Shreelekshmi, M. Wilscy, C.E.V. Madhavan, "Cover Image Preprocessing for More Reliable LSB Replacement", ICSAP 2010: International Conference on Signal Acquisition and Processing, February 2010, pp.153-156

[36] R. –Z. Wang, C. –F. Lin and J. –C. Lin, "Hiding data in images by optimal moderately significant-bit replacement", IEE Electronic Letters, Vol. 36, pp.2069-2070, 2000

[37] R. –Z. Wang, C. –F. Lin and J. –C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm", Pattern Recognition, Vol. 34, pp. 671-683, 2001

[38] C. –K. Chan and L. M. Cheng, "Improved hiding data in images by optimal moderately significant-bit replacement", IEE Electronic Letters, Vol. 37, pp.1017-1018, 2001

[39] C. –C. Chang, J. –Y. Hsiao and C. –S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy", Pattern Recognition, Vol. 36, pp. 1583-1595, 2003

[40] R. Ji, H. Yao, S. Liu and L. Wang, "Genetic Algorithm Based Optimal Block Mapping Method for LSB Substitution", IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 215-218, 2006

[41] N. Hopper, L. Ahn, J. Langford, "Provably Secure Steganography", IEEE Transactions on Computers, pp.662-676, May 2009

[42] W. Fraczek, W. Mazurczyk, K. Szczypiorski, "Stream Control Transmission Protocol Steganography", IEEE 2010 International Conference on Multimedia Information Networking and Security, pp. 829-834, November 2010

[43] M. N. Miyatake,H. P. Meana,S. T. Maya, "An Image Steganography Systems Based on BPCS and IWT", IEEE 16th International Conference on Electronics, Communications and Computers, pp.51-59, March 2006

[44] Y. Yang, X. Niu, N. Jiang, J. Wang, "Symmetric Steganography Secure Against Chosen Message and Original Cover Attacks", IEEE First International Conference on Innovative Computing, Information and Control, pp.661-664, September 2006

[45] E. Cole, R. D. Krutz, "Hiding in Plain Sight: Steganography and the Art of Covert Communication", Wiley Publishing Inc., 2003

[46] A. Westfeld, A. Pfitzmann, „Attacks on Steganographic System", In proceedings of Information Hiding, 3rd Workshop in information Hiding, Springer-Verlag, pp.61-76, 1999

[47] H. Tian, K. Zhou, Y. Huang, D. Feng, J. Liu, „A Covert Communication Model Based on Least Significant Bits Steganography in Voice over IP", IEEE The 9th International Conference for Young Computer Scientists, pp. 647-652, November 2008

[48] Y. Huang, B. Xiao, H. Xiao, „Implementation of Covert Communication Based on Steganography", IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1512-1515, August 2008

[49] A. Castiglione, U. Fiore, F. Palmieri, „E-mail-based Covert Channels for Asynchronous Message Steganography", IEEE 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 503-508, July 2011

[50] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel", CRYPTO, 1983, pp. 51–67.

[51] R. Rivest, A. Shamir, L. Adelman, „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, 1978, pp. 120-126

[52] S. D. Rane and G. Sapiro: "Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression standard for compression of high resolution elevation data". IEEE Transacations on Geoscience and Remote Sensing, 2298-2306, 2001

[53] M. Stone: "Representing Colors as Three Numbers". IEEE Computer Graphics and Applications, 2005

[54] M. Goesele: "New Acquisition Techniques for Real Objects and Light Sources in Computer Graphics", ISBN 3833414898, 2004.

[55] A. Cheddad, J. Condell, K. Curran, P. Kevitt: "Biometric Inspired Digital Image Steganography". 15th Annual IEEE International Congference and Workshop on the Engineering of Computer Based Systems, 159-168, 2008

[56] Y. Sun, F. Liu: "Selecting Cover for Image Steganography by Correlation Coefficient". Second International Workshop on Education Technology and Computer Science, 159-162, 2010

[57] R. Shreelekshmi, M. Wilscy, C.E.Madhvan: "Cover Image Preprocessing for More Reliable LSB Replacement Steganography". Internetional Conference on Signal Acquisition and Processing, 153-156, 2010

[58] L. Lu, X. Li, M. Qi, J. Li, J. Kong: "Lossless and Content Based Hidden Transmission for Biometric Verification". Second International Symposium on Intelligent Information Technology Application. 462-466, 2008

[59] X. Luo, D. Wang, P. Wang, and F. Liu, "A review on blind detection for image steganography". Signal Processing, 2138-2157, 2008

[60] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes., Information Hiding, 6th International Workshop, volume 3200 of Lecture Notes in Computer Science, 67–81, 2005

[61] Y. Q. Shi, C. Chen, and W. Chen, "A Markov process based approach to effective attacking JPEG steganography". 8th Information Hiding Workshop, 249-264, 2006.

[62] T. Pevny and J. Fridrich, "Merging Markov and DCT features for multi-class JPEG steganalysis", Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents, 2005

[63] H. Farid, "Detecting hidden messages using higher-order statistical models". International Conference on Image Processing, Rochester, New York, 2002

[64] J. Lin, X. Wang, X. Zhong, "Reduction of Markov Extended Features in JPEG Image Steganalysis". 2nd International Congress on Image and Signal Processing, 2009 IX,volume 6505, pp.28-40, San Jose, CA, January 29 – February 1, 2007

[65] T. Pevny and J. Fridrich, "Multi-class blind steganalysis for JPEG images". SPIE, (San Jose,CA), 2006

[66] X. Hongping, X. Jianhui, "Steganalysis method based on svm and statistic model in contourlet domain". SPIE - The International Society for Optical Engineering, 2007

[67] C. Guangxi, L. Zhensheng, W. Daoshun, "Steganalysis SVM algorithm based on adjacent pixel and texture correlations". Qinghua Daxue Xuebao, pp.1233-1236, 2009

[68] S. Amari, S. Wu, "Improving support vector machine classifiers by modifying kernel function". Journal of Neural Networks, No.12, pp.783–789, 1999

[69] S. Wu, S. Amari, "Conformal Transformation of Kernel Functions: A Data-Dependent Way to Improve Support Vector Machine Classifiers", Neural Processing Letters, No.15, pp.59-67, 2002

[70] S. Amari, "Information Geometry and Its Applications: Convex Function and Dually Flat Manifold". LIX Fall Colloquium on Emerging Trends in Visual Computing, pp.75-102, 2009

[71] An. Wensen, S. Yanguang, "An information-geometrical approach to constructing kernel in support vector regression machines". First International Conference on Natural Computation, ICNC, pp.546-553, 2005

[72] S. Ziwen, H. Maomao, Guan Chao, "Steganalysis Based on Co-occurrence Matrix of Differential Image". International Conference on Intelligent Information Hiding and Multiedia Signal Processing, pp.1097-1100, 2008

[73] W. T. Stephen, "Fractal and wavelet image compression techqniques", SPIE Publication, pp.155-156, 1999

[74] J. Nichols, T. Babty, "A model-based self-adaptive approach to image processing", 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04), pp.456-461, 2004

[75] S. J. Osher, A. Sethian, "Front propagation with curvature dependent speed: Algorithms ased on hamilton-jacobi formulations", Journal of Computational Physics vol. 79, pp. 12–49, 1988

[76] Y. Bai, J. Hu, Y. Luo, "Self-adaptive Blind Super-Resolution Image Reconstruction", 3rd International Congress on Image and Signal Processing (CISP2010), pp. 1208-1212, 2010

[77] J. Palis, W. de Melo, "Geometric theory of dynamical systems", SpringerVerlag, New York, 1982

[78] G.P. Penney, J. Weese, J.A. J.A. Little, P. Desmedt, D.L.O Hill, D.J. Hawkes, "A comparison of similarity measures for use in 2-D-3-D medical image registration, IEEE Transactions on Medical Imaging, pp.586–595, 1998

[79] E. Pichon, A. Tannenbaum, R. Kikinis, "Statistically based flow for image segmentation", Medical Imaging Analysis, pp.267–274, 2004

[80] J.P.W Pluim, J.B.A. Maintz, M.A. Viergever, "Mutual-information-based registration ofmedical images: a survey", IEEE Transactions on Medical Imaging pp.986–1004, 2003

[81] J. Bolz, I. Farmer, E. Grinspun, P. Schroder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid", SIGGRAPH, 2003

[82] J. Kruger, R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms", SIGGRAPH, 2003

[83] L. Li, B. Luo, Q. Li, X. Fang, "A Color Image Steganography Method bt Multiple Embedding Strategy Based on Sobel Operator", International Conference on Multimedia Information, Networking and Security, pp.118-121, 2009

[84] Z. Zhang, G. Zhao, "Butterworth filter and Sobel edge detection image", International Conference on Multimedia Technology, pp.254-256, 2011

[85] L.-J. Kau, C.-S. Chen, "A low complexity dual mode edge detector", Visual Communications and Image processing, pp.1-4, 2011

[86] "Parametric effects in color-difference evaluation", Bureau Central de la CIE, 101, 1993

[87] P. Colantoni, A. Tremeau, "3D Visualization of color data to analyze color images", Proceedings of PICS Conference, pp 500-505, 2003

[88] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST", Proceedings of the IEEE international Test Conference 2001 pp.868-877, 2001

[89] F. Brglez, D. Bryan, K. Kozminski, „Combinational Profiles of Sequential Benchmark Circuits", Proceeding of the International Symposium of Circuits and Systems, pp.1929-1934, 1989

[90] R. Yang, G. Welch, "Fast Image Segmentation and Smoothing Using Commodity Graphics Hardware", Journal of graphics tools - special issue on "Hardware Accelerated Rendering Techniques", 2003

[91] M. Arnold, S. Wolthusen, M. Schmucker, Techniques and Applications of Digital Watermarking and Content Protection", Artech House, 2003

[92] D. Huang, T. Yeo, "Robust and Inaudible Multi-echo Audio Watermarking," Proceedings of the IEEE Pacific-Rim Conference on Multimedia, pp.615 – 622, 2002

[93] I. Avcibas, N. Memon, B. Sankur, "Image Steganalysis with Binary Similarity Measures," Proceedings of the IEEE International Conference on Image Processing, pp. 645 – 648, 2002

[94] I. Cox, J. Kilian, F. Leighton, T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," IEEE Transactions on Image Processing, pp. 1673 – 1687, 1997