SYSTEM STEGANALYSIS: IMPLEMENTATION VULNERABILITIES AND SIDE-CHANNEL ATTACKS AGAINST DIGITAL STEGANOGRAPHY SYSTEMS

a thesis submitted to
The University of Kent
in the subject of computer science
for the degree
of doctor of philosophy.

By
Thomas Sloan
2018

Abstract

Steganography is the process of hiding information in plain sight, it is a technology that can be used to hide data and facilitate secret communications. Steganography is commonly seen in the digital domain where the pervasive nature of media content (image, audio, video) provides an ideal avenue for hiding secret information. In recent years, video steganography has shown to be a highly suitable alternative to image and audio steganography due to its potential advantages (capacity, flexibility, popularity). An increased interest towards research in video steganography has led to the development of video stego-systems that are now available to the public. Many of these stego-systems have not yet been subjected to analysis or evaluation, and their capabilities for performing secure, practical, and effective video steganography are unknown.

This thesis presents a comprehensive analysis of the state-of-the-art in practical video steganography. Video-based stego-systems are identified and examined using steganalytic techniques (system steganalysis) to determine the security practices of relevant stego-systems. The research in this thesis is conducted through a series of case studies that aim to provide novel insights in the field of steganalysis and its capabilities towards practical video steganography.

The results of this work demonstrate the impact of system attacks over the practical state-of-the-art in video steganography. Through this research, it is evident that video-based stego-systems are highly vulnerable and fail to follow many of the well-understood security practices in the field. Consequently, it is possible to confidently detect each stego-system with a high rate of accuracy. As a result of this research, it is clear that current work in practical video steganography demonstrates a failure to address key principles and best practices in the field. Continued efforts to address this will provide safe and secure steganographic technologies.

Acknowledgements

First and foremost, I would like to express my gratitude and thanks to my supervisor, Professor Julio Hernandez-Castro. Throughout my Ph.D. Julio has always been there to offer support, insight and encouragement. Without his guidance and knowledge, this work would not have been possible.

A special thanks must go to my parents Andrew and Alison. For all of the challenges and hardships that I have encountered, they have been there to give their full support with unending patience. Words cannot express how grateful and fortunate I am to have them as parents and role models.

I would also like to thank Alejandro Cervantes and Pedro Isasi from University Carlos III of Madrid, for their collaboration and advice on automated system steganalysis.

Finally, I would like to thank the supporting staff within the School of Computing, University of Kent. They have made a considerable effort to create an excellent Ph.D. programme and ensure that all candidates have a high level of support. I am grateful to have been enrolled in this programme.

List of Publications

The following list of publications are of special relevance to this thesis.

1. **Sloan, T**. and Hernandez-Castro, J., 2015. Steganalysis of Openpuff through Atomic Concatenation of MP4 Flags. Digital Investigation, 13, pp.15-21 ***Chapter 5***.

2. **Sloan, T**. and Hernandez-Castro, J., 2015. Forensic analysis of video steganography tools. PeerJ Computer Science, 1, p.e7 ***Chapter 4***.

3. Cervantes, A. **Sloan, T**. Hernandez-Castro, J. and Isasi, P., 2018. System Steganalysis with Automatic Fingerprint Extraction. PLoS one, 13, 4, p. e0195737 ***Chapter 6***.

4. **Sloan, T**. and Hernandez-Castro, J., 2018. Dismantling OpenPuff PDF Steganography. Digital Investigation ***Chapter 5***.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditional approaches to cyber-security and data protection have a heavy reliance on encryption. Cryptographic protocols and systems are implemented worldwide to keep information and communications secure. This is a result of the digital era and the consumption of digital content.

In many cases, cryptography provides the ideal solution when a user is mainly concerned about privacy and the confidentiality of their information. However, cryptography still faces challenges. Secured communications can still be at risk of attracting unwarranted intervention. An encrypted message, although indecipherable, may be tampered if it has been intercepted. In some cases, the messages can be analysed to attempt decryption, or the communication channel may be blocked to cause disruption between parties. Finally, the message can be modified or destroyed through a sanitization process. These concerns demonstrate a need for an alternative method of communication, a subliminal channel in which communicating parties would be safe from intrusion and intervention. This was originally conceptualised and formulated by Gustavus Simmons in the well known Prisoner's Problem [138]. The aim was to emphasise the need to explore alternative subliminal channels for secret communication.

In this formulation, Alice and Bob are locked in separate cells. The two prisoners plan to escape confinement even as their communication is monitored and controlled by the prison's warden Eve. If Eve suspects that Alice and Bob are exchanging messages secretly, any communication is blocked and the plan will fail.

Three scenarios can be described to demonstrate approaches towards creating and using a subliminal channel.

Scenario 1 can be called "The plaintext problem" (Figure 1). Alice and Bob use a typical method of communication where information is being exchanged between the two prisoners in clear, plain-text. Eve, who has access to the communication channel can easily observe the messages and see that unauthorised information is being exchanged. With control of the channel, Eve can block, modify or sanitise the messages, causing any plan formulated between Bob and Alice to fail.

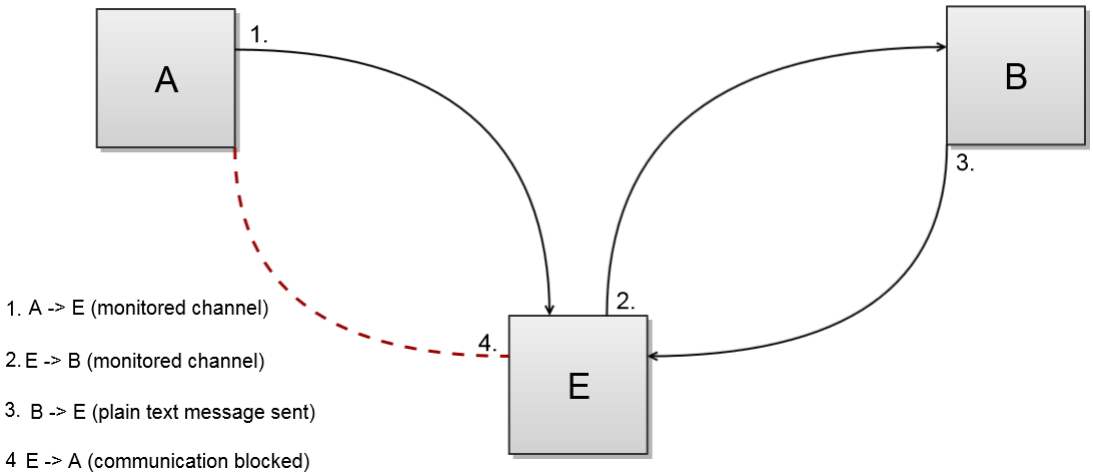

Figure 1: Scenario 1 - The plaintext problem

Scenario 2 can be named "The cryptographic addition" (Figure 2). In this scenario, Alice and Bob attempt the use of cryptography over plaintext. The two prisoners encrypt a message to make it indecipherable so that the warden Eve cannot read the contents of any exchanged messages. This method of communication although potentially secure in terms of confidentiality, draws attention to the unreadable content of each message. Eve believes this to be suspicious and she will again block the prisoner's channel of communication.

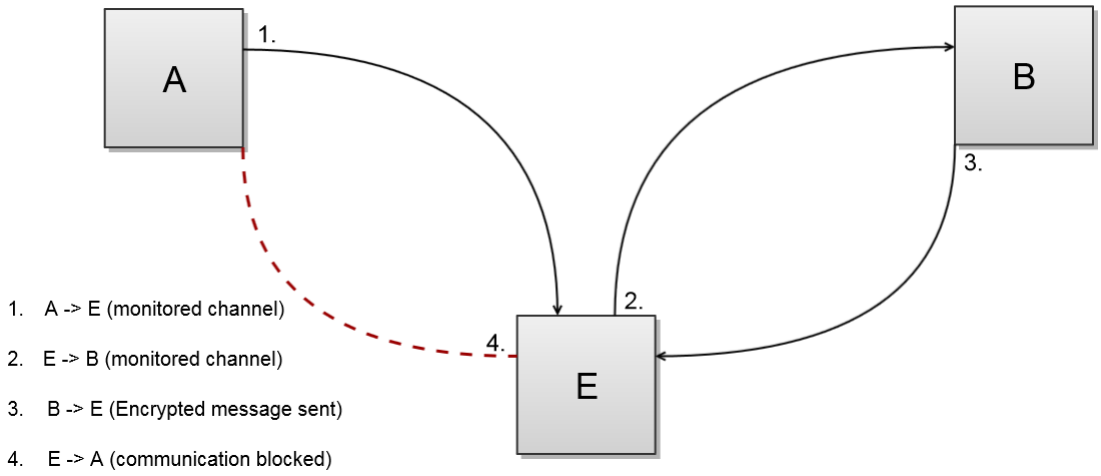

Figure 2: Scenario 2 - The cryptographic extension

Scenario 3 is the "steganographic solution" (Figure 3). Alice and Bob create a subliminal method for exchanging messages, one that is hidden within the original channel of communication. Through this, the two prisoners can hide their messages in plain sight and keep any plans to escape hidden from the warden Eve. Without knowledge of this subliminal channel, communications are allowed through.

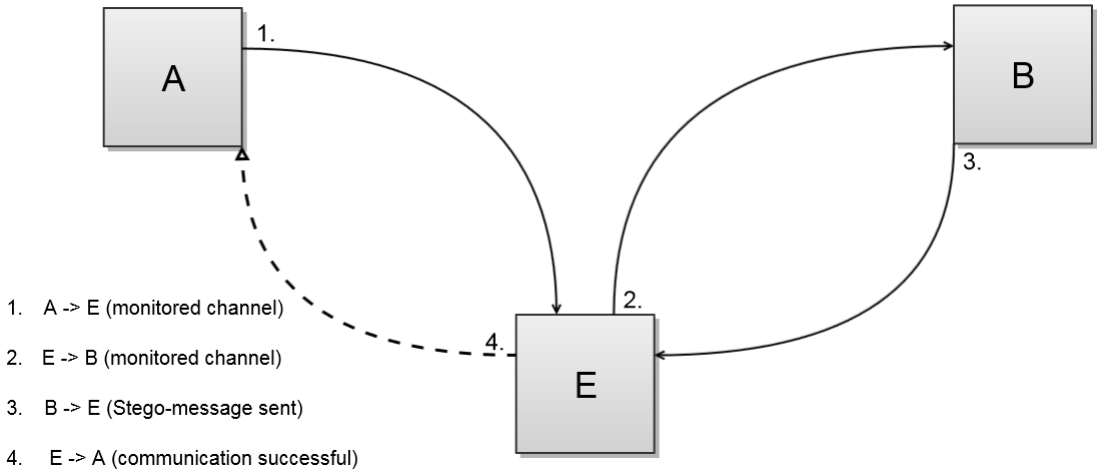

Figure 3: Scenario 3 - The steganographic solution

Formally this is defined as steganography. It is the process of hiding information in plain sight while concealing any evidence of itself. Steganography has become particularly popular in the digital domain due to the diverse nature of digital

content, and the widespread usage of the Internet. Messages can be hidden through a variety of file formats and easily exchanged. Detection of steganography is left to the field of steganalysis. This is the pursuit of identifying and proving the existence of steganography. Typically, a steganalytic attack is based upon one of two main approaches. The first falls under the traditional paradigm for steganography, where the approach is modelled as a data hiding and detection problem. Through this, the attack is interested in how an embedding algorithm impacts a digital signal. The second approach extends this paradigm to consider steganography as a coding challenge. This is described as a system attack and through this, a diverse range of steganalytic attacks can be constructed throughout an entire given object. In this case, the attack may not be related exclusively to the embedding algorithm.

Statistical steganalysis can at times reliably detect the presence of steganography. However, these techniques face multiple challenges in a practical setting. Statistical methods are sophisticated and typically incorporate machine learning and feature extraction techniques. When applicable, system attacks can markedly simplify the process of steganalysis. Implementation vulnerabilities can be far easier to identify as they are often predictable and consistent. This is because high-level features used for extraction will usually be an unintentional by-product of the implementation. In some cases, it could be that a stego-system has not considered its impact on a stego-object outside of the digital signal. This effectively creates a new avenue for steganalysis. These vulnerabilities quickly degrade a stego-system into a cryptographic application and defeat the fundamental purpose of steganography.

Steganography can be used either for malicious or legitimate purposes. Typically, stego-systems are created and intended for those who are unable to communicate safely. This could include journalists or those who are under oppressive, non-democratic regimes or whistle-blowers. These are the target audience for many modern steganography tools such as Pixelknot [120]. However, the unintended use of steganography by criminals often motivates the need for advancements in steganalysis.

Terrorists have long been suspected of using steganography to conceal all evidence of their communications. A report by the United Nations recognises the use of steganography tools such as Camouflage to hide information [110] for terrorist

purposes. The report also identifies the FARC organisation and showcases their use of steganography and similar anti-forensic techniques. Many other known cases give support to this concern. In May 2011, an al-Qaeda operative, Masqood Lodin, was detained by German police and found to be in possession of a concealed memory stick containing a video. Embedded within this video using steganographic techniques were 141 documents including reports on past al-Qaeda operations and plans for future attacks [60]. Another high-profile case involved a Russian spy ring based in the USA. This group used steganography to embed messages within image files and upload them to websites to avoid detection from the U.S. government [146]. It has also been reported that ISIS supporters are using the Telegram channel to share steganography tools [111].

The use of steganography by cyber-criminals to distribute malware and control compromised machines is another concerning use of the technology. A case presented by researchers from Proofpoint had highlighted the use of malvertising campaigns. In October 2015 criminal groups used malicious code embedded within advertising banners to target and infect users [121]. Due to the secretive nature of steganography, data hiding techniques provides an ideal method for data exfiltration. Sensitive information can be leaked from secure locations if reliable detection systems are not in place [100]. Finally, steganography has been used to support the distribution of illicit images, where it has been employed to exchange and store such content over seemingly innocuous media [153].

Steganography has shown a decreasing interest over the past decade, as shown in Figure 4. Trends in user searches for “Steganography” appear to be more or less consistent in recent years. In contrast to this, interest and searches for its counterpart “Steganalysis” show significantly less interest.

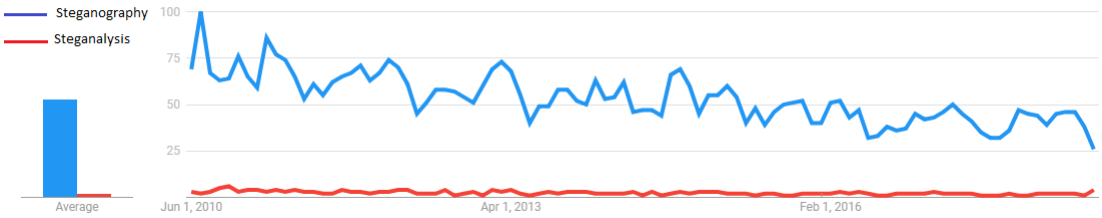

Figure 4: Trends in searches for "steganography" and "steganalysis" according to Google Trends as of 27/05/2018 - Trends are relative to the highest point

Regional trends show an interesting variability. As shown in Figure 5, interest in steganography is far from consistent globally, and once again, regional results show far more interest in steganography as opposed to steganalysis.

Figure 5: Regional interest in steganography relative to the highest point (Google Trends as of 27/05/2018)

As evidenced by these figures, and by cases of real-world criminal use for steganography, more attention is needed in steganalysis. Steganalytic techniques require continuous development to keep up with the advancements in steganography.

1.1 Research Problem

From a practical standpoint, there are many challenges with the current state of steganalysis. Many experts agree that state-of-the-art steganalysis cannot be used effectively in the real-world [88]. This problem comes from the difficulty of modelling statistical solutions into a practical environment, where the parameters of

theoretical implementations would struggle with the highly variable aspects of the real-world. This is especially relevant when considering the practical applications of a steganalytic method, where time allocation, resources, and expert-knowledge become a major constraint for forensic practitioners. In addition to this, many of the attacks presented in the academic literature often assume the steganalyst will have a vast pre-existing knowledge of the embedding algorithm used to hide secret information. In a practical case, without access to a stego-system, this is unlikely and other methods must be considered for successful steganalysis.

System-attacks provide an avenue for practical steganalysis, especially as a method to combat the emergence of modern video stego-systems targeted to criminal or terrorist use. System steganalysis is a particularly useful method to examine the broader impact of the stego-system over a stego-object, to quickly identify the method of embedding and construct a targeted attack. These techniques can provide an ideal approach for forensic analysts and investigators. However, even though these attacks can simplify the process of steganalysis, this avenue of research still remains largely unexplored with too many unanswered questions. The scope of system steganalysis has not yet been extensively evaluated, and the applications for system attacks are not fully understood. This is especially true with regards to video steganography, a field that is rapidly advancing in recent years. A global methodology for system attacks does not exist, and little research has been conducted to explore why these techniques can be so effective. Because of this, system attacks are at risk of being overlooked in favour of other well-known approaches, even in cases where this approach might provide better results.

Academic research in steganalysis has a significant weighting towards the statistical methodology. These attacks provide an interesting mathematical solution to the complex problems brought forward by modern advancements in steganography. However, this can lead to the development of stego-systems that lack consideration towards their implementation vulnerabilities, as little effort is directed towards examining robust security coding. This was evident with the implementation of the JPEG compressor for the F5 algorithm [46]. As a result, system attacks are a highly viable method of steganalysis, but their capabilities and limitations over new and emerging steganographic technologies are not fully understood. Given these problems, this thesis considers the following research questions.

1. How are system attacks best used within the scope of steganalysis? 2. Can system attacks be successfully applied to achieve practical results over video steganography?

Research in system steganalysis presents several challenges. Firstly, the focus of this research is to examine the practical applications for system steganalysis. Because of this, the methodology must be heavily focused towards a series of case studies that examine existing stego-systems. The range of embedding algorithms across all stego-systems must have unique embedding properties to suitably address the different applications of system steganalysis (Weak encoding/implementation, poor practice for pre- and post-processing, and secret key/cryptographic key vulnerabilities). In addition, the primary focus of this research surrounds the application of system attacks to video steganography stego-systems. Therefore, it is important that a diverse range of video steganography tools are identified to address the primary focus of the thesis.

1.2 Research Approach

The work presented in this thesis proposes a methodology for system steganalysis. This is achieved by examining the state-of-the-art in practical video steganography to demonstrate the applications for system attacks. Furthermore, by using practical case studies, this approach highlights the motivation for system steganalysis and emphasises why this method of steganography can be successful in realistic settings.

The primary scope of this research surrounds the practicality of steganalysis with regards to implementation vulnerabilities of a stego-system. To evaluate this work, system attacks are constructed and tested over existing stego-systems. The research on this thesis is conducted primarily through a series of case studies to examine and evaluate the practical scope of system attacks over video steganography. The approach to this research consists of several stages.

First, system attacks are used to attack pre- and post-processing requirements

of a stego-system (double compression). This approach demonstrates that stego-systems constructed to target subtle, temporal features of a given object can unintentionally modify auxiliary components. This leaves the stego-system vulnerable to steganalysis.

Second, system steganalysis is used to detect implementation vulnerabilities surrounding the weak encoding and encryption features of a stego-system. This approach shows practical examples of poor practice in steganographic security that allow highly accurate methods of detection.

Third, weak embedding algorithms for stego-systems are subjected to system steganalysis. This research demonstrates that even in cases where coding problems are considered, the impact is compromised at the expense of a strong embedding algorithm. From this, it is possible to use system attacks to detect the presence of steganography.

Finally, these approaches are implemented into a framework for automated blind system steganalysis. This method is capable of detecting steganography for each case study presented throughout this thesis and can be applied as a practical steganalytic attack to new stego-systems.

1.3 Thesis Contributions

The main aim of this thesis is to determine whether system attacks can be used to detect video steganography systems and whether these attacks can achieve practical results. From this, there are several contributions, which are outlined in the following.

1. A series of novel system attacks are presented in Chapters 3, 4, and 5. These steganalytic attacks are not only new but also valuable from a practical point of view.

2. This work examines the possible applications of system steganalysis on video stego-systems. The aim of this contribution is to determine if practical results can be achieved and if they simplify the process of steganalysis.

3. A framework is presented for the autonomous detection of stego-systems.

Although the focus of this is the detection of practical video steganography, it is also applicable to many other types of digital steganography (image, audio, documents).

4. The gap between theoretical and practical models for video steganography is examined. This contribution aims to explore and identify the association between modern stego-systems and system attacks.

1.4 Structure of Thesis

The remainder of this thesis is structured as follows. Chapter 2 provides a literature review that outlines the scope of the current state of steganography and the outlying research problem. The literature review discusses the topics of steganography, steganalysis and video steganography. The literature review outlines academic research and well-known methods for steganography over digital media. This section provides a discussion of the state-of-the-art in image and video steganography and steganalysis.

Chapter 3 presents the experimental and research methodology carried out in this thesis. This chapter discusses the experimental decisions made throughout this work and aims to justify and explain why components of the experimental procedure had been chosen. In addition, this chapter discusses the formal outputs from this thesis and examines the research impact of the work.

Chapter 4 examines the association between stego-systems and system attacks with regards to implementation vulnerabilities across several different stego-system. The research in this chapter focuses on the impact of poor encoding, encryption, embedding, and pre- and post processing to evaluate the scope of system attacks.

Chapter 5 examines a stego-system that itself has demonstrated consideration towards some of the challenges of practical steganography. This is performed as a means to reduce its risk of vulnerability. However, this is achieved by compromising the embedding algorithm and using a method that is vulnerable to steganalysis. The research in this chapter is evaluated through a case study, presenting a novel detection scheme to identify the method of steganography and the stego-system.

Chapter 6 presents a blind autonomous framework using a trained classifier to extract signatures through system steganalytic methods. This framework is applied to state-of-the-art steganography tools across multiple formats to demonstrate the capability of system steganalysis and the scope of system vulnerabilities.

Chapter 7 evaluates the work of this thesis. This chapter examines the initial research questions of the thesis and discusses whether they have been suitably addressed by the research conducted throughout the thesis. Furthermore, this chapter examines the experimental results of previous chapters to identify formal outputs and determine their success.

Chapter 8 concludes this thesis by evaluating the research that has been conducted and examining the contributions made. Finally, future avenues of research are discussed.

Chapter 2

Literature Review

2.1 Introduction

Data hiding is an anti-forensic technique that is particularly suitable to the digital domain. Sensitive information can be susceptible to a wide variety of internal and external attacks that target intellectual property, critical resources, or personal data. In typical cases, these attacks are quickly detected [12] and resulting damage can be mitigated by patching vulnerabilities and informing those affected. However, subtle attacks can cause significant damage by remaining undetected over longer periods of time [174]. An emerging trend is seeing anti-forensic and specifically data hiding techniques being employed to secretly exfiltrate sensitive information or hide unlawful data or communications. The art and science of hiding information in such a way is called steganography.

Steganography is a field within the domain of information hiding. This is an area of research that exists to support data protection and information security. The goal of steganography is to hide information in plain sight, often by creating a covert channel within a digital medium that can be recovered at a convenient time. This is achieved through embedding techniques that modify components of digital data. Figure 6 illustrates the broader scope of information hiding and some of its active fields of research.

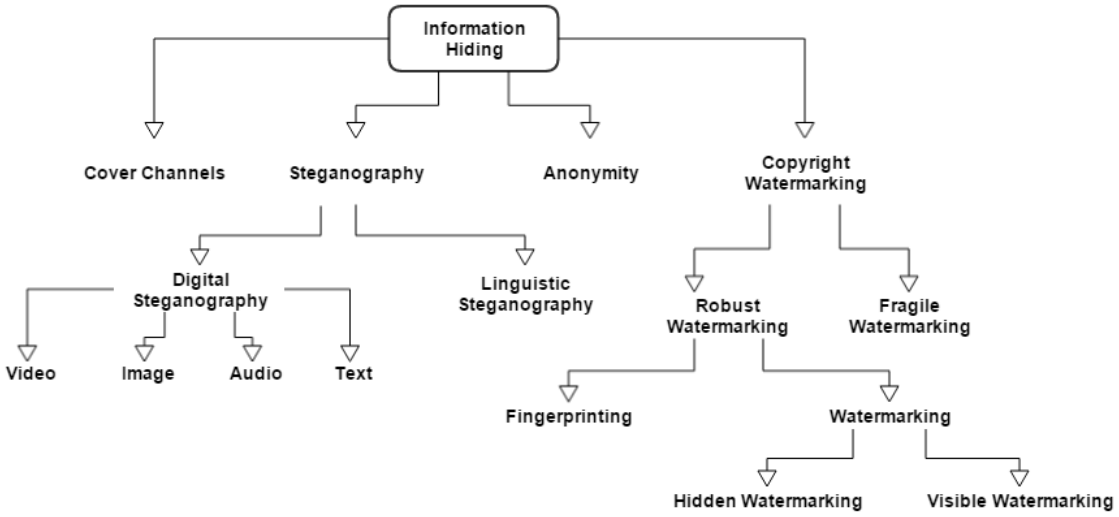

Figure 6: Classification of information hiding concepts based on [118]

Steganography is often seen as a two-sided or dual-use technology. On one side, the steganographic approach exists to support those who may be in an environment unaccommodating of free speech. States where censorship is heavily enforced, or where channels of communication are actively monitored are common cases where steganography can be legitimately used. This is particularly relevant to journalists and whistle-blowers. Under such circumstances, classic models of communication would likely fail, as monitored channels can detect interesting or uncommon communications. This would include cryptographically secure exchanges, as even the suspicion of undesirable communication would be enough to warrant intervention in many scenarios. However, when given access to a secure steganographic scheme, a user can communicate safely without any concern of detection.

The other side to steganography shows potential for malicious use. Data hiding techniques are used to facilitate and exchange illicit content, spread malware, steal sensitive data [114], or communicate terrorist activities [60, 110]. The challenges brought forward by this use of steganography have led to the need of new and accurate detection methods.

Steganalysis is the counterpart to steganography. This is an area of research in which its pursuit is to identify and detect steganography. Similarly, steganalysis has a familiar two-sided use. On one side, it can be used to identify the presence of secret terrorist communications and possibly extract any hidden criminal

messages. However, malicious use of steganalysis may result in identifying and capturing those who communicate secretly to avoid intervention from dangerous third parties under censorship or in oppressive states. This could lead to journalists or whistleblowers being placed in unnecessarily dangerous situations that can be avoided. These challenges have resulted in a technological arms race between steganography and steganalysis, where the former is in pursuit of the ideal steganographic scheme and similarly, the latter endeavours to develop novel steganalytic approaches. Consequently, this technological arms race has become an important part of cyber-warfare in the 21st century [150].

Steganography has attracted attention over recent years from both academic and industrial sectors, especially within the larger domain of information hiding. Watermarking techniques are used prominently for data protection and copyright enforcement. The role of steganography and information hiding techniques is particularly important at a time where cryptography is facing uncertainty regarding the implementation of backdoor exploits. This is particularly relevant in states where governments may have a direct involvement over their countries encryption standards [132]. Such activity undermines the fundamental objectives of cryptographic schemes, and points to a need for alternative secure solutions.

The use of steganography can provide an ideal solution in these circumstances. The exchange of sensitive information can avoid suspicion by concealing the channel of communication. However, various controls need to exist to maintain assurances against the abuse of steganographic systems. These can range from administrative procedures (security policies and protocols) to technical solutions (e.g., file sanitizers, detection systems). In a digital environment, it is crucial to understand that there is no clear end to this arms race as it will be a continuously evolving pursuit. For as long as there is a need to keep information and communication secret, new tools and techniques will be created to hide and detect secret data.

The research in this thesis focuses on steganalysis by presenting a methodology and framework for system steganalysis. Specifically, the goal is to evaluate an approach towards system steganalysis by using system attacks to provide insights into practical, usable approaches for steganalysis. This research supports analysts and investigators to handle the steganographic challenges they may often encounter. The research in this thesis has a particular focus towards video steganography

and video-based stego-systems. One challenge of modern steganalysis with regards to a steganalytic investigation surrounds time and resources available, as well as the practicality of any proposed steganalytic solution. The diversity in which digital signals can be modified means that a steganographic system can consider many separate digital features to perform steganography. This makes steganalysis particularly important in media files such as video. Video formats are a pervasive media type that offer multiple possibilities for embedding (e.g., frames, audio, metadata). In steganalysis, the attacker must have a well-rounded understanding of the embedding algorithm before a secure steganalytic method can be constructed. System attacks in steganalysis provide an ideal solution to not only determine the embedding techniques used, but also detect the presence of steganography where traditional statistical methods assume a pre-existing knowledge of the steganographic scheme.

In this chapter, an extended literature review explores the topics of steganography and steganalysis. It is structured as follows. Section 2.2 provides a core overview of steganography. Common embedding techniques and models within academia and industry are reviewed alongside emerging techniques and steganographic concepts that are of particular interest within the field. Section 2.3 discusses the steganalytic counterpart to the field, and the techniques used for identifying the presence of steganography, and how this has affected both fields. Section 2.4 provides an overview of the current state of the art for video steganography which, is of significance throughout this thesis. This section will review how embedding algorithms and steganographic solutions have evolved and where they fit in, academically and practically. Section 2.5 covers video steganalysis and reviews the detection techniques used to attack video based stego-systems. Lastly, in Section 2.6, a summary of the literature review is provided.

2.2 Steganography

Steganography is the process of hiding information in plain sight. By doing so, secret communications are concealed in such a way that third party intruders would be unaware of such practices [76]. This method facilitates secret communication by creating a secure channel in which two parties can exchange information safely.

Digital communication is now a fundamental component of the modern world and everyday life. From this technological advancement, privacy is an important feature that is expected from its users, and for many this is possible, but is not always the case. For some, unwarranted or unknown intervention could be in action. In many cases, steganography mitigates potential consequences that might arise from the use of open or insecure communication channels. Figure 7 demonstrates steganography in use over the JPEG format. The image on the left is clear of any form of steganography, while the image on the right has been modified by the OpenPuff steganography tool, with a 6KB of pseudo-random data embedded.

Figure 7: Steganography in use (left is clean, right is modified)

Digital steganography has a single goal, that is, to conceal any evidence of hidden data or its transmission. Imperceptibility is the primary feature of steganography in which a modified file (stego-object) must be indistinguishable from the original unmodified file (cover-object). The challenge is to create a steganographic system that can accommodate a desired capacity while maintaining security against steganalysis. The development of a secure steganographic system is a continually evolving process. Over time, traditional concepts and practices in the field adapt and what may once have been a highly secure scheme, will sometimes be easily detectable by advancements in steganalysis. In light of this, it is important for any user of steganography to consider how a stego-system will be influenced by four key steganography principals: The cover-object, the selection rule, the embedding operation, and the embedding efficiency [54].

The cover-object is the object used to hide secret information. The selection

of a cover-object and its properties will have a significant impact on the development of a steganography scheme and the overall steganographic security. A poor choice of cover-object will likely produce suboptimal results, specifically, this could lead to a low rate of embedding efficiency and produce a highly detectable stego-object. Regarding digital steganography, the choice of cover-object is diverse. Typically, a cover-object can be any form of digital representation that contains redundancy to hide secret information [124]. In the context of steganography, redundancy can be defined as any component of a signal that maintains minimal visual and statistical deviation between cover-object and stego-object. A highly diverse range of files and formats can be used for steganography. The most popular being media files such as image, audio, and video.

The selection rule is the principle used to identify components of the cover-object that will be modified during the embedding process. The steganography algorithm must consider the importance of each component in a cover-object to minimize the impact of the embedding process. The selection rule will determine the potential for capacity based on how it identifies digital redundancy within a cover-object.

The embedding operation determines how the signal component of a cover-object will be modified to incorporate the secret information. This is different from the selection rule, which identifies the component of the cover-object to be modified. Specifically, the embedding operation determines how the selection rule affects the cover-object. For example, this could be a randomized or sequential process.

Embedding efficiency calculates the average number of message-bits introduced into the cover-object for each embedding change made. The purpose of this is to determine the amount of information embedded within a file against the number of changes required to successfully embed the message [171]. A higher embedding efficiency increases steganographic security as it reduces the likelihood of detection. Ideally, a secure steganographic scheme will offer higher embedding efficiency, where a stego-object can accommodate a high capacity while maintaining the statistical properties of the cover-object. Modern steganographic schemes are capable of achieving this by implementing cover-codes to improve embedding efficiency [169].

In addition to these four principles, the design of a good steganographic system should demonstrate consideration towards steganalysis. A secure steganographic scheme should aim to satisfy the criteria of imperceptibility, embedding capacity, robustness, and tamper-resistance with regards to steganalytic methods [124]. The first three are often considered to be the pillars of steganography, as shown in Figure 8.

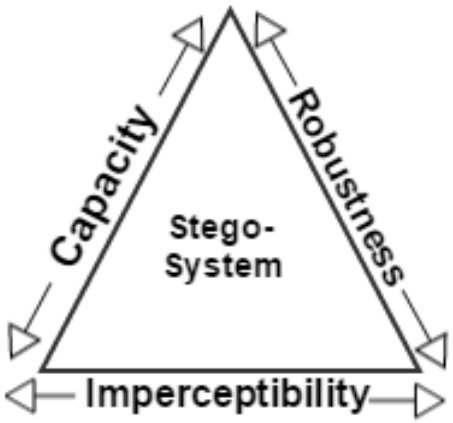

Figure 8: The three characteristics of steganography

Imperceptibility is the potential for a stego-object to appear indistinguishable from the cover-object. In any case an observable deviation is identified, the presence of a hidden message can be proven and the application of steganography fails. This is the main feature of good steganographic systems, and frequently one has to compromise the others to maximise this.

Embedding capacity relates to the amount of data that can be embedded within a file. This has a direct trade-off with imperceptibility. It is ideal for a steganographic system to hide a large amount of information; however, if embedding capacity exceeds the threshold for imperceptibility, identifiable artifacts will be introduced.

Robustness is the ability for an embedded message to accommodate unintentional third-party changes to the stego-object. A common example of this could be compression of a file. File hosting sites often compress a file to their particular standard for storage purposes. This can effectively be a sanitization process for

many steganography schemes, where the hidden message is removed upon compression. Again, this can also be a trade-off with the other two properties.

Tamper-resistance relates to an attackers' ability to modify a message once it has been embedded within a carrier. Similarly to file sanitization, steganographic schemes are not often developed with robustness in mind and can be fragile to small changes. This principle is often given the least consideration when developing a steganographic scheme. The security of a stego-system relies fundamentally on imperceptibility. With regards to this, an attacker should not be aware of the presence of steganography to warrant targeted tamper-resistance.

2.2.1 The Process of Steganography

Embedding schemes differ greatly due to the diverse range of steganographic redundancy across digital content. Typically, the functionality of an embedding algorithm will be determined by the selection of the cover-object. This can vary based on different cover-objects and the form of redundancy identified. In some cases, an embedding scheme can be applied across different cover-object types. This has been demonstrated by Luo et al. [102], who proposed an edge-adaptive scheme for image that can be extended to video. Alternative methods of steganography, however, may be limited to a specific cover-object. This is often seen when the embedding algorithm modifies features that are specific or unique to the cover-object (e.g., metadata components, compression features, etc). Considering the diverse range of potential for steganography, the embedding operation follows a typical process across most stego-systems, outlined in Figure 9.

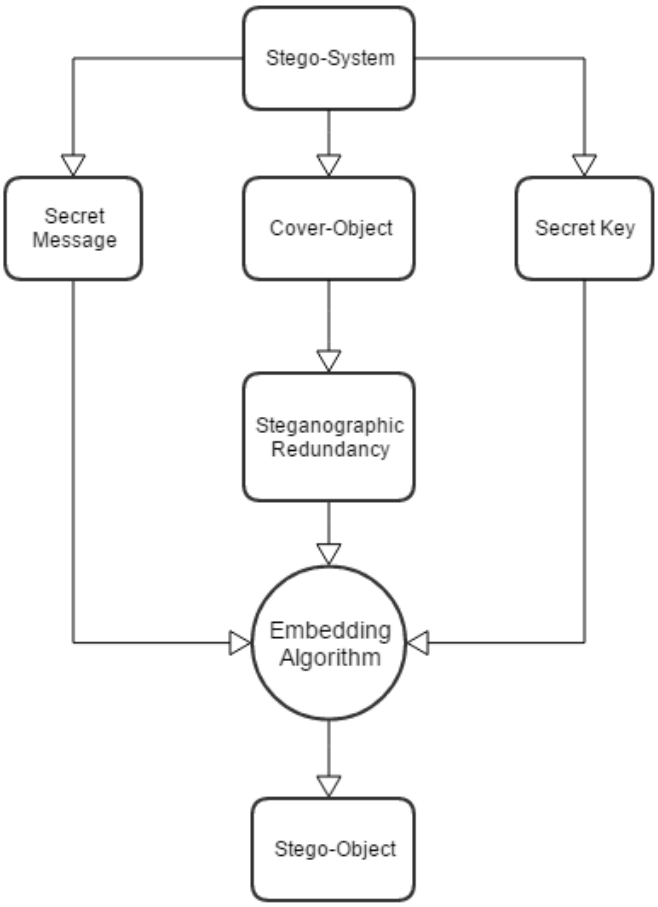

Figure 9: The process of steganography

Four components are required to carry out steganography: the steganographic scheme, a cover-object, secret data, and a secret key to encrypt hidden contents. The inclusion of encryption is seen as a best practice for steganography and an added layer of security. This key is usually agreed upon by all parties involved in the communication and would be shared privately beforehand.

The embedding scheme follows a predetermined operation, outlined by the stego-system. The selection process will identify components within a cover-object that contain steganographic redundancy (cover-bits). The cover-bits are replaced by the secret message and any additional padding that may be necessary as determined by the embedding operation. The final result is a stego-object containing a secret message that should be indistinguishable from the cover-object prior to the

steganographic process.

Traditional embedding methods use one of two main selection criteria for steganography. These are sequential and random embedding. The former was popular in early steganography schemes, where embedding locations are selected sequentially (e.g., one after another in a particular order). This embedding principle had fundamental vulnerabilities, as it was simple to trace a selection path and find a hidden message [26]. The latter method randomizes the bits through a seemingly arbitrary permutation. This offers a more secure solution, as the embedded bits are challenging to detect and the embedding path will appear inconsistent.

2.2.2 The Warden

In steganography and steganalysis, it is often considered that there are three separate entities. The first two are Alice and Bob, who are attempting to communicate secretly and securely. The third is Eve, the warden who acts as an eavesdropper in the communication channel. The role that Eve plays in the process of steganography and steganalysis varies depending on each scenario, which can be summarised as follows:

- Active Warden - In this role, Eve has an active role in steganalysis with a goal to prevent hidden communication, or at least impede it. It can be assumed that Eve has total control over the communication channel and can choose to block it at any point. Furthermore, Eve is allowed to modify the message that is being sent. This introduces additional challenges to the process of steganography. Notably that any new steganography method constructed must be capable of surviving an active modification.

- Passive Warden - In this role, Eve will not interfere with any messages sent. A passive warden has a read-only presence, in which their goal is to correctly identify the use of steganography.

These scenarios are constructed as adversary models. This is similarly seen in cryptography and is used to create a set of definitions that illustrate the goals and limitations of a steganalyst. This is an important feature of both steganography

and steganalysis, as it is impossible to achieve meaningful security goals when the warden is omnipotent [18].

2.2.3 Spatial Domain Steganography

Steganography is commonly performed in digital media, where it can be applied to a diverse range of applications. Spatial domain steganography remains a popular method, whereby images are represented as a matrix of intensity values. It is the individual values of the spatial domain that are commonly modified. This is because small changes to intensity values generally cause hardly recognisable visual differences to the stego-object as a whole. The early basis for this type of steganography was to exploit the limitations of the human visual system (HVS).

2.2.3.1 LSB Substitution

LSB (Least Significant Bit) substitution is considered the most well-known method of embedding in digital steganography. It relies on the principle that the least significant bit of any byte is frequently indistinguishable from random noise and can therefore be replaced by bits from a secret message. For example, Figure 10 shows the full representation of a greyscale image (on the left) paired with its representation in the LSB bit-plane. The LSB bit-plane appears random and should therefore accommodate bit replacement with an encrypted message. Advancements in steganalysis, however, revealed this naive assumption to be completely untrue, leading to steganography methods exploiting this false concept [15].

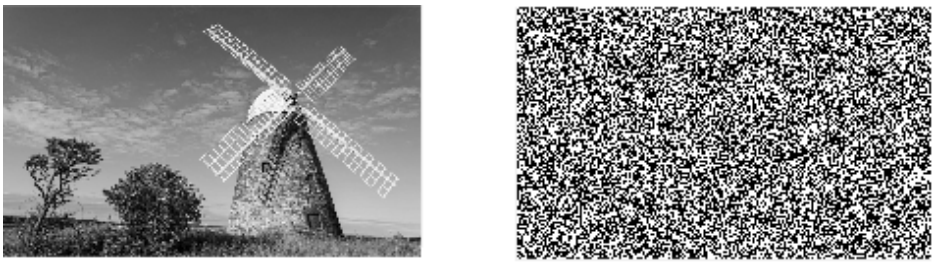

Figure 10: Comparison of greyscale image and LSB bit plane, image from [29]

The early work on image steganography was by Bender et al. [16], Adelson [2], and those working in spread-spectrum steganography [38], [139], [31]. Initially,

these researchers based their schemes on exploiting the limitations of the Human Visual System (HVS) and its sensitivity to contrast versus spatial frequency.

In the naive applications of LSB steganography, there was one qualifying characteristic, that the change of the LSB of a byte would result in a stego-object being visually indistinguishable from the cover-object. The concept of LSB is still widely applied in modern steganography schemes [35, 41, 106], However, these methods are typically supported by complex operations and cover codes to produce a secure scheme with improved embedding efficiency, that is significantly harder to detect [45].

One of the first known implementations for digital steganography was a spatial domain technique proposed by Kurak and Mchugh [92]. This method embedded secret data within 4-LSB components and was the first case of steganography exploiting detailed knowledge of the HVS. Typically, spatial techniques modify bits within an image's pixel values, replacing them directly with the secret message bits [19]. The concept of bit replacement would be that, for each pixel in an image, it can be assigned a corresponding integer value, based on the intensity of a colour that is displayed by the pixel. Once converted into a binary stream, individual bits can be extrapolated to determine whether they are suitable for substitution [129]. For example, functions can be included to prevent even values from being decreased and odd values from being increased. Figure 11 demonstrates the process for LSB bit substitution steganography.

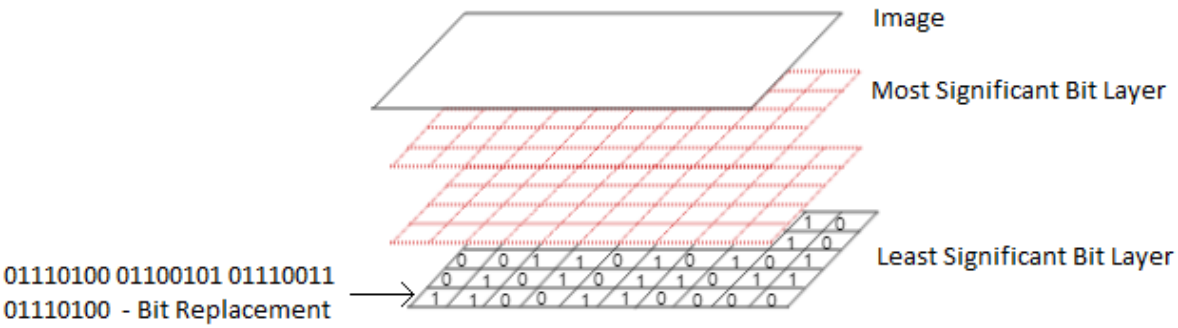

Figure 11: Breakdown of bit-planes in an image

This concept formed the initial works in LSB steganography. However, naive

LSB schemes are highly susceptible to Chi-square tests and statistical steganalysis [144].

LSB encoding within 24-bit colour images was the most popular form of early spatial domain steganography. The embedding technique was used across many software products [76, 77]. Capacity became the focus of these techniques, as 24-bit images allowed the user to store up to 3-bits for each modified pixel. The early work in spatial domain steganography for colour based images was broken by the efforts of Fridrich et al. [55]. To detect LSB steganography, Fridrich and her team considered a calculation of the ratio between close colour pairs and all colour pairs in an image.

Their idea was that for an image with no secret messages, the number of close pairs of colours relative to the number of all possible pairs of colours will be smaller than for an image that has a message already embedded in it. To determine the ratio threshold, their team observed that further modifying a stego-object with a new secret message will not modify the ratio in any significant manner. Whereas if the image does not contain a secret message, then embedding will drastically change the ratio. This knowledge can be used to determine the presence of LSB colour encoded steganography and develop a detection algorithm. A test was used where a message was randomly distributed throughout the LSB of selected pixels. The calculation is given to the new stego-object and if the two ratios are almost identical, LSB steganography is suspected.

2.2.3.2 LSB Matching (+1 Method)

LSB matching is a method of LSB steganography first proposed by Sharp [136]. The concept was that instead of sequentially and predictably substituting bits of target LSBs within an image, an adaptive scheme would be constructed to be selective of the features present in any particular cover-object. If the message bit fails to match the LSB of the cover-image, then a '1' value is either added or subtracted from the value of the target pixel. To ensure that the scheme for AoSo (Add or Subtract One) would be invertible, an algorithm would be implemented to stop pixel values from being modified outside of a predetermined range. While simple to implement, it is surprisingly and significantly more challenging to detect

than basic LSB substitution. A similar method was proposed by Mielikainen [105] who proposed that instead of using a random function to determine the use of the $+1$ or -1 principle, a selection feature can be introduced to set a binary function of two cover pixels to the desired value. A sample of this is shown in Algorithm 1.

Algorithm 1 Sample algorithm for improved LSB matching by [105]

Input: Pixel pair from cover image x_i, x_{i+1} & two secret message bits m_i, m_{i+1}
Output: Corresponding stego-image pixel pair y_i, y_{i+1}
if $m_i = \text{LSB}\ (x_i)$
  if $m_{i+1} \neq f(x_i, x_{i+1})$
    $y_{i+1} = x_{i+1} \pm 1$
  else
    $y_{i+1} = x_{i+1}$
  end
  $y_i = x_i$
else
  if $m_{i+1} = f(x_i - 1, x_{i+1})$
    $y_i = x_i - 1$
  else
    $y_i = x_i + 1$
  end
  $y_{i+1} = x_{i+1}$
end

This improved method of LSB matching maintained the same embedding efficiency as the original but provided greater resistance to known attacks, such as the one proposed by Westfeld [156]. Westfeld's attack relied upon a count of arbitrary neighbour colours introduced by the embedding algorithm for LSB matching. However, this means that the detection method was only applicable to colour images and would not work for a grayscale representation [84]. A stronger method for greyscale images was presented by Andrew Ker [85].

2.2.3.3 Prediction Error

Prediction error was another method of spatial domain steganography that gained popularity. The scheme assumed a fairly intuitive principle within images in that; for a stego-object to maintain visual quality and be indistinguishable from the cover-object, secret data should be hidden within complex areas of an image. To determine local complexity, pixel prediction error can be used where the larger the prediction error, the more obvious the local fluctuation.

Using this method, data can be hidden within prediction errors. These are pairs of pixels isolated where each corresponding neighbour pixel is used to predict the current pixel value. This calculation is assumed to be the prediction error where data can be hidden within the difference values. Suppose the following of two-group neighbour pixels from a partitioned image where p_i and $p_{i_{+}1}$ are the pixels used. Their difference value would be.

$$d_i = p_{i_{+}1} - p_i \tag{1}$$

Where $0 \leq | d_i | \leq 255$ [95]. The size of d_i denotes the complexity of the data and determines its suitability for steganography. This allows for any embedding distortion to be distributed evenly and to maintain the features of the cover-object.

The selection process has a fundamental role in any steganography scheme. A robust criteria for selection defines the security offered by the embedding algorithm. Many of the discussed embedding schemes are based upon Pseudo-Random-Number-Generation (PRNG) to make the embedding path less predictable. This often means that any embedded message, given enough length will be spread across the entire stego-object. Typically, consideration towards embedding location often reduces visual distortion and increases imperceptibility. This concept was presented by Hemalatha [69] who proposed a technique called Hiding Behind Corners (HBC) to target specific regions for LSB embedding. The effect of this is demonstrated in Figure 12. The presence of black pixels in (b-g) describe where corresponding pixel values have been modified by steganography. This demonstrates the selection process for HBC.

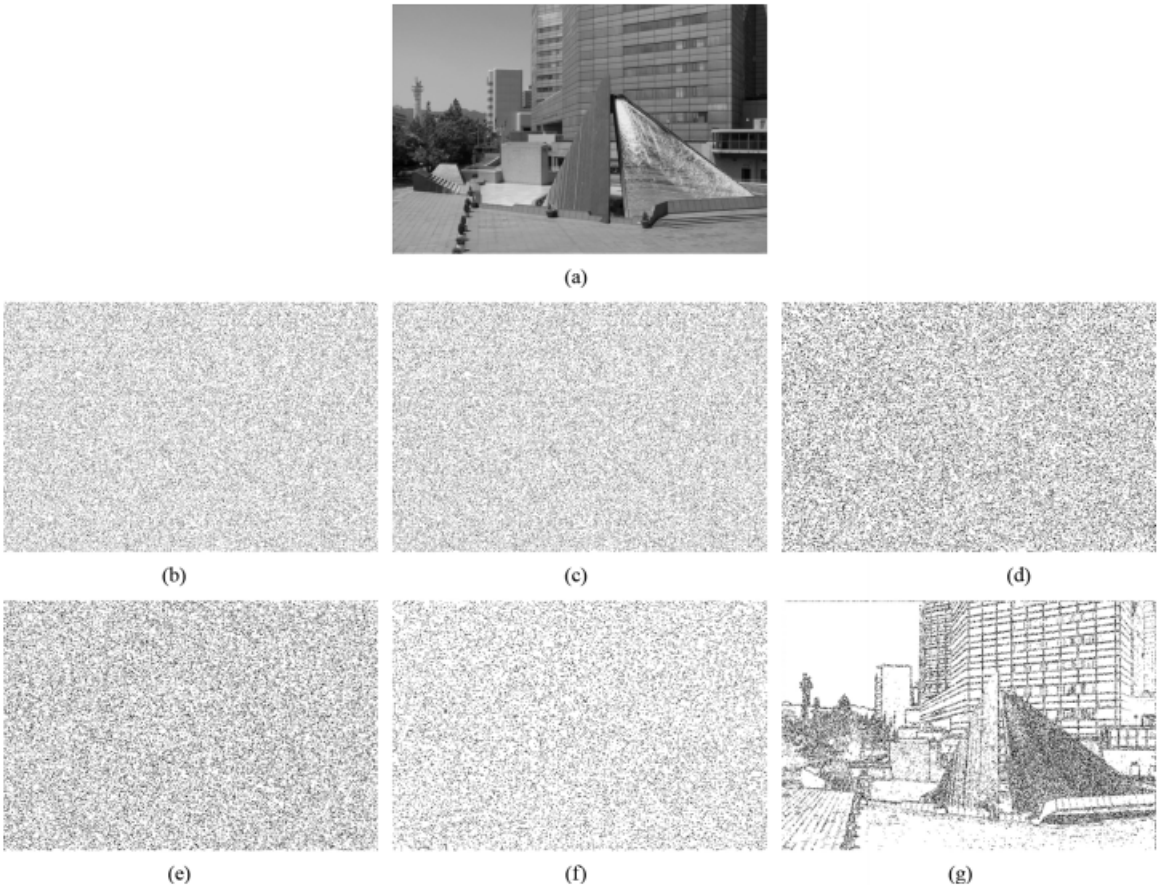

Figure 12: Impact of HBC steganography vs traditional LSB, from [102]

Many of the early methods for spatial domain steganography were quickly outperformed by academic advancements in transform domain steganography. However, avenues of LSB steganography are still actively pursued and recent advancements led to complex spatial domain systems. Particularly of note are those using content adaptive steganography such as HUGO (Highly Undetectable SteGO).

2.2.3.4 Content Adaptive Steganography

Traditional schemes for steganography, particularly with LSB, are often consistent in their selection process. Even when using a pseudo-random scheme, features of the embedding process can be predictable. Content adaptive techniques instead alter the selection process based on the content of the cover-object. For this,

embedding locations are often made in complex regions where content noise masks any steganographic noise being introduced.

2.2.3.5 Highly Undetectable SteGO (HUGO)

Advancements in steganography and steganalysis have also been motivated by competitions. The "Break Our Steganographic System" (BOSS) competition launched in 2010 proposed one of the most advanced image steganography tools to date and challenged researchers to break it [13]. This competition was inspired by similar watermarking competitions in 2005 and 2007 titled BOWS and BOWS-2 [119]. Contestants of BOSS were provided datasets, a 9074 pair of clean and modified image files, with the embedding algorithm for training. The goal was to use steganalytic techniques over a sample of 1000 images and determine which contained steganography featuring the "Highly Undetectable SteGO" (HUGO) system. The competition featured an international pool of competitors, each with varying degrees of success. The most noteworthy contributions were achieved by Fridrich et al., with an accuracy of 80.3%, Gul & Kurugollu (76.8%) and A. Westfeld (67%) [44].

HuGO Steganography worked in the spatial domain via LSB in grayscale images. The primary feature of the embedding operation was the application of the minimum-embedding-impact principle; where for any given message that is to be embedded, the distortion between cover-object and stego-object is minimal. The function of the scheme can be isolated into two parts, the image model and the coder. The image model is tasked with generating a space in which the distance between two points leads to a suitable distortion function, this is in accordance with the minimum-embedding-principle. The distortion function is then applied by the encoding, using content-adaptive functionality. This will determine which features of the cover-object can be modified during embedding [13]. The advanced security behind the HUGO system is based on its ability to preserve a high-dimensional representation of cover-objects [57]. The content-adaptive scheme used by HUGO provides excellent statistical resistance by limiting modifications to cover-regions that are complex to model. The edge adaptive embedding operation is shown in Figure 13, where +1 embedding changes are shown through white pixels and -1

embedding is shown through black pixels.

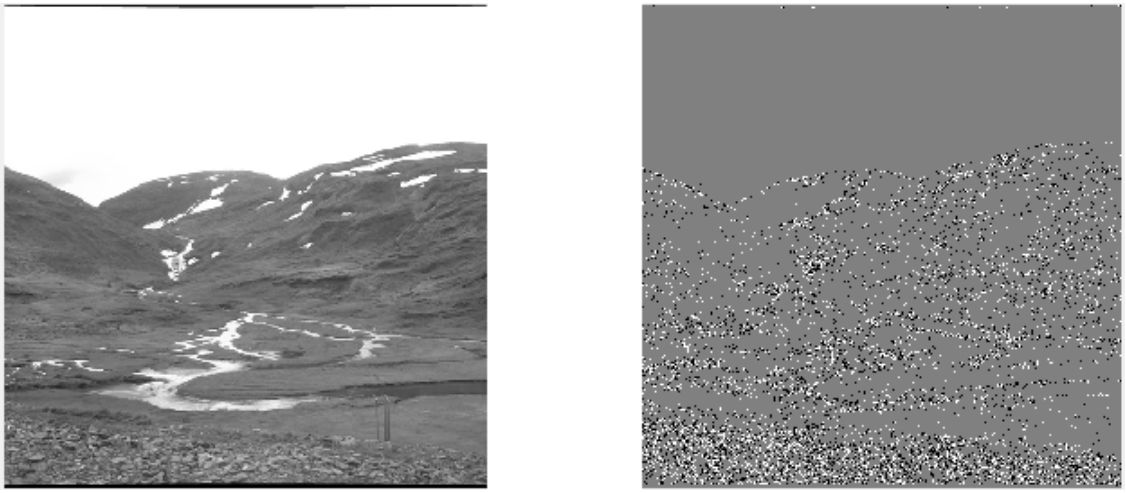

Figure 13: Cover-object (left) and encoded image (right) shows +1 embedding changes on white pixels and -1 embedding on black pixels

2.2.4 Transform Domain Steganography

Transform domain techniques provide a solution to many of the challenges in spatial domain steganography. By transitioning an image into the frequency/transform domain, embedding methods demonstrate increased capacity and robustness to signal processing attacks [150]. This happens because the embedding techniques in transform domain steganography would spread changes across the entire image. Well-known methods for image-based transform domain steganography, include DCT (Discrete Cosine Transform), and DWT (Discrete Wavelet Transform).

2.2.4.1 DCT Steganography

For each colour component in the JPEG image, the compression scheme uses a discrete cosine transform to convert an image block of 8 x 8 blocks through a matrix into 64 DCT spectral sub-bands using a quantization process. The processed result provides a sum of varying magnitudes and frequencies that scale according to the image's visual quality. The significance of this is that for a typical image, most of the visually significant information will be stored in a small number of the coefficients resulting from the DCT operation. To perform this operation, the

DCT coefficients $F(u, v)$ of each 8 x 8 block (where u and v are the frequency coordinates) are calculated as follows based on the algorithm by Provos [124].

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)v\pi}{16}\right] \quad (2)$$

Afterwards, the DCT operation quantizes coefficients in a 64-element quantization table for $Q(u, v)$ using

$$F^Q(u,v) = \left[\frac{F(u,v)}{Q(u,v)}\right] \quad (3)$$

From this, the least significant bits of quantized DCT coefficients are used as steganographic redundancy for message embedding. The resulting change for each coefficient is shown in Figure 14.

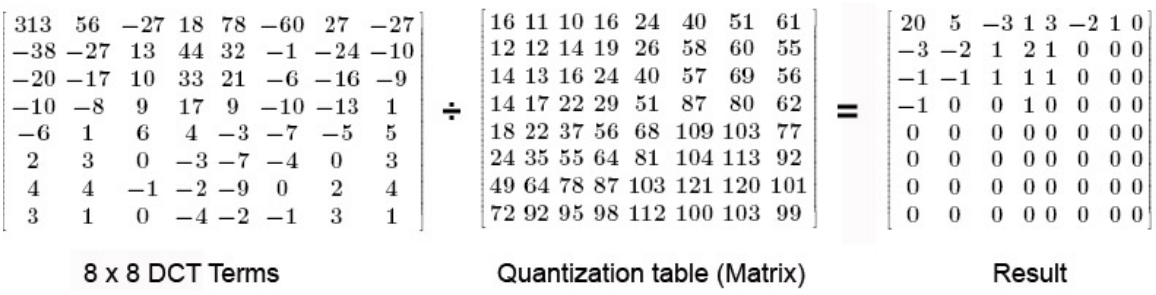

Figure 14: DCT quantization process, from [145]

For image steganography, particularly over the JPEG format, a common technique is to identify the least significant bit components of the quantized DCT coefficients as redundant data within the image, to be used to embed a secret message, using similar bit-replacement techniques to LSB steganography. One of the first publicly available schemes to carry out steganography in the transform domain was JSteg, developed by Derek Upham [149]. This used a sequential embedding operation to replace the least significant bit of DCT coefficients with the secret message. However, it has several notable security issues. Firstly, there was no secret shared key, this means that any user with access to the steganographic scheme could extract the hidden message. The second vulnerability was first presented by Andreas Westfeld and Andreas Pfitzmann. They found that sequential

operations used for embedding caused distortions that are detectable by steganalysis [158]. They observed that the embedding of high entropy data would cause deviations to a histogram that could be easily predicted, for example, in that uniform embedding to distribute message bits throughout would reduce frequency differences between adjacent images.

An improvement for the predictability problem of sequential embedding was proposed by Niels Provos in a system called OutGuess 0.1 [122]. This stego-system used a pseudo-random number generator to select the DCT coefficients at random for embedding. This method provided resistance to the statistical chi-squared attack. However, it was eventually broken by Fridrich et al [58]. Fridrich had shown that macroscopic features of the steganographic file predictably change with the length of the embedded message. At this point, they were able to successfully determine the parameters of any quantity by calculating an approximation to the cover image.

Andreas Westfeld carried out research on DCT schemes and proposed three separate systems F3, F4 and F5 [155]. The function of F3 was to decrease the coefficient's absolute values in case the LSB did not match. This improvement over Jsteg means that zero coefficients would not be selected by the embedding operation, except when matching the corresponding bits from the secret message. This provided resistance to the standard Chi-square test, but was statistically predictable due to the exclusive reduction of zeros. F4 advanced this by fixing the vulnerabilities of F3. This was achieved by mapping negative coefficients to the inverted steganographic value. F4 however, used a technique called Continuous Embedding where the process of embedding a secret message would result in changes at the start and end of the file. This made the scheme easily detectable and hence susceptible to steganalysis.

The final release, F5 attempted to maximise embedding efficiency through matrix encoding and adapting the mechanism from F4 to shuffle all coefficients through a permutation. This would provide a simpler estimation for expected file capacity. F5 was seen as the first implementation of matrix encoding using a technique conceptualised by Ron Crandall [32]. These methods for using cover codes to maximise embedding efficiency are still pursued in modern research [98, 104, 108]

Matrix encoding relies on the assumption that for a standard stego-object where message bits are uniformly distributed, approximately half of the embedded message will cause changes to the carrier file. This provides a base-line embedding efficiency of 2-bits-per-change. However, with many early schemes this could be lower when accounting for shrinkage. This would occur when the changes made during embedding would not account specifically for the hidden message itself. These changes could consist of parity information or compression components. F4 is an example of this where the embedding efficiency was 1.5 bits per change, due to the problem of continuous embedding.

2.2.4.2 DWT Steganography

DWT (Discrete Wavelet Transform) steganography is an alternative method of embedding within images the transform domain. A well-known method for this is the Haar transform [1]. For any given image, there are two stages to this method of steganography. Firstly, image pixels are scanned horizontally. Neighbouring pixels are calculated through addition and subtraction operations and the sum (S) is stored on the left while the difference (D) is stored on the right. This is repeated for every row of pixels, where the sum values on the left are the low frequency band, and the difference values on the right are the high frequency band. This stage is shown in Figure 15.

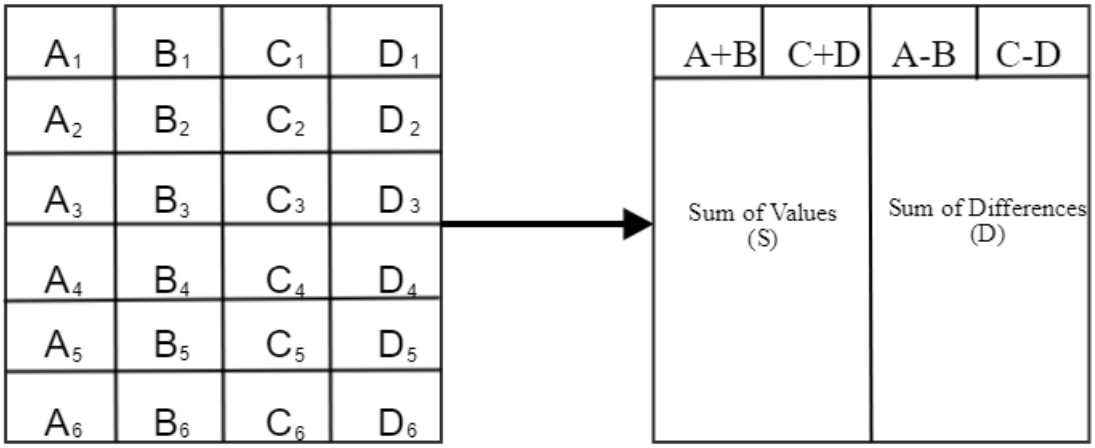

Figure 15: First stage of DWT transformation

Pixels are then scanned vertically and the same operation is performed on neighbouring pixels. In this step, the sum of the values for each column is stored at the top, and the difference is stored at the bottom. From this, 4 sub-bands are created, each called LL, HL, LH, and HH (as shown in Figure 16) where LL represents the low frequency band.

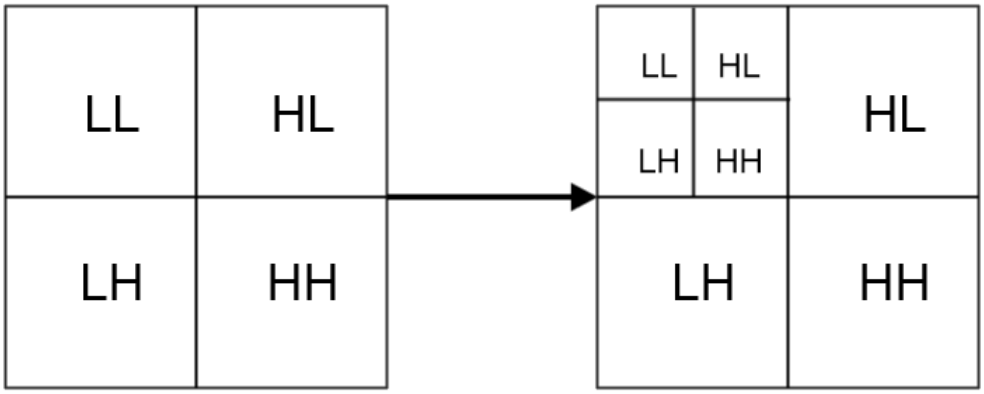

Figure 16: Second stage of DWT transformation

The low frequency sub-band contains the most significant features of the image file. Therefore, embedding is recommended in this region to preserve the secret message from signal processing modifications.

2.2.5 Social Relevance and Impact of Steganography

The practical applications of steganography have made it a useful technology for security and privacy. This can, however, support a small minority of its users to avoid detection while planning or committing crime or terrorist activities. It should be highlighted that the vast majority of steganography users will have legitimate reasons (i.e. journalists, civil rights and democracy activists under threat of imprisonment, etc.) to employ it. However, it is the criminal use of steganography which often gains widespread exposure through the media. Recent cases in point are the use of image steganography by a network of Russian spies in the US [134], video steganography usage by an Al-Qaeda operative in Germany [60], or recent strands of malware employing steganography for communication using command and control servers [165].

In recent years, video steganography has become an increasingly popular technique for data exfiltration. One example is the case of an undisclosed Fortune 500 company that was hit by this type of attack [114]. The use of video steganography allowed for large amounts of sensitive data to be exfiltrated from the company network after the breach, bypassing all the Data Leakage Prevention (DLP) tools and Intrusion Detection Systems (IDS) put in place. The impact that steganography has on DLP systems has been subjected to some academic research [4], but still remains largely unexplored.

The use of steganography in the real-world has often been a subject of discussion, especially surrounding the distribution of steganographic content throughout the internet as a means to support terrorist communications. Shortly after the 9/11 attacks in 2001, US media discussed the prospect of this, and it wasn't until an article published in *USA Today* that academics became involved [83]. The article claimed that Al-Qaeda used websites such as eBay to distribute messages through bid images. An investigation of this claim was carried out by Niels Provos and Peter Honeyman [124]. They attempted to not only approach these allegations, but also question how steganalytic methods, particularly statistical attacks, can be used in a real-world setting. A detection framework was created to download JPEG images from the Internet and use their proposed attacks to identify the presence of steganography. It utilised StegDetect to detect content hidden in JPEG images across popular tools (JSteg, JPHide, and Outguess 0.13b). This was paired with a dictionary attack against suspect images in an attempt to extract steganographic content.

Throughout the course of the investigation, over two million images were downloaded from eBay auctions, to which StegDetect indicated 17,000 could potentially have steganographic content. However, the dictionary attacks gave no positive results, for which, the authors offered four possible explanations.

1. Steganography users chose passwords not susceptible to dictionary attacks.
2. Steganographic content may be hidden within other sources.
3. Steganographic systems examined throughout the investigation are not used by actual users.

4. Messages are too small for detection.

Although Provos and Honeyman's work did not identify evidence of steganography, recent studies reveal how modern steganography is being used maliciously. A study conducted by McAfee in 2017 [133] has shown a significant rise in the use of steganography for the purposes of spreading malware. The impact of malicious steganography demonstrates a need for further research in steganalysis, and the development of reliable detection methods.

2.3 Steganalysis

The field of steganography has established a wide variety of methods for hiding secret messages. By embedding data in digital content, communicating parties can securely exchange messages through a stego-object that is for all purposes indistinguishable from the cover-object. However, the process of steganography is invasive and can leave detectable traces within a stego-object. Techniques used to detect the presence of steganography fall under the scope of steganalysis. This is the pursuit of identifying and proving the existence of steganography. This is different from cryptanalysis, where the goal is to identify the key and read the contents of an encrypted message.

Steganalysis is considered a decision problem in that a steganalyst must decide between one of two statements with confidence for any given object. Firstly, the steganalyst can assert that "the given-object contains content hidden by steganography". Secondly, the steganalyst can assert that "the given object contains no hidden data". This statement is considered successful only when the confidence of a steganalytic assertion is significantly higher than that of a random guess. However, the result is still prone to two types of error. The first is the likelihood that a clean object is misclassified as a stego-object. This is known as the probability for false positives. The second is the likelihood that steganalytic scheme fails to correctly classify a stego-object. This is called false negatives. Every steganalytic scheme should aim to minimise the rate of false positives while maintaining a high rate of detection accuracy.

Steganography and steganalysis are continually evolving fields. A diverse selection of cover material brings considerable challenges to the efforts of steganalysis. Many formats of natural media offer low embedding distortion to make stego-objects appear indistinguishable from the original cover [164]. For this reason, digital steganography provides an ideal means to facilitate secret communication. Despite these challenges, steganalysis is an actively pursued field of research. Methods used for steganalysis are highly varied with each often being tailored to fulfil the particular needs for a steganalytic investigation. It is often considered that in steganalysis, there are two approaches to the detection of steganography.

- **Blind Steganalysis**: The goal of blind steganalysis is to detect any steganographic technique regardless of the scheme used. The steganalyst has no pre-existing knowledge of how an embedding algorithm functions. A blind attack can be approached by constructing a trained classifier to extract feature sets between cover-object and stego-object [65]. Blind attacks cannot deal with the full representation of a stego-object, but instead will focus on low-dimensional feature space. Although feature extraction is considered a well suited approached for blind steganalysis, it is often that methods only manage a small number of features or a single domain of the stego-object. This can provide low detection accuracy. However, methods do exist that manage blind steganalysis across a wider variety of features and multi-domain instances [116]

- **Targeted Steganalysis**: It is assumed that the steganalyst will have pre-existing knowledge of a steganographic scheme before any attack is carried out. A steganalytic method is constructed against that particular scheme to detect the presence of steganography [58]. Typically the scheme used will not be applicable to other stego-systems. The pre-existing knowledge of the stego-system simplifies the challenge of steganalysis, but can only be used when this information is available, which is not always practical. Similar to blind steganalysis, these attacks cannot work over the full representation of a stego-object. However, a targeted attack should require fewer features to be extracted, as they are constructed from pre-existing knowledge of the embedding mechanism.

Although the single goal of steganalysis is to detect the presence of steganography, different attacks can be constructed to achieve this. Applications for steganalysis are defined by the approach used. System and statistical attacks are considered to be two of the major types of steganalytic attack [46].

2.3.1 Steganalytic Attacks

The success of any steganalytic intervention relies on the information that is available to an attacker. For any scenario, different approaches can be used based on any pre-existing knowledge of the cover-object, stego-object, stego-system, secret message, or secret key. Attempts have been made to classify many of the attacks that exist within steganalysis [112, 46]. These attacks and methods can be described as follows:

2.3.1.1 System Attacks

System attacks are one of two major types of steganalytic attack. Steganalysis in this instance has a particular focus on the stego-system to identify implementation vulnerabilities or obtain side-channel information. This information is used to construct detection methods for that particular stego-system. System attacks may target third-party features of the stego-system or exploit the presence of weak cryptographic features and encoders. Typical approaches for system steganalysis can include the following:

1. Secret Key Attacks: A secret key used to hide a message with steganography should be sufficiently strong so that it cannot be attacked. A PRNG implementation should be strong enough to mitigate attacks that could break the stego-system and even provide the attacker with readable information.

2. Magic-prefix - This can be a common issue with steganographic schemes that embed extra information unrelated to the message itself. Compression artifacts are a typical example of this, as they are common practice in the construction of a steganography scheme. These artifacts may introduce predictable changes that can be detected by observing known prefix locations, such as the header or footer of a file.

3. Signature steganalysis: Steganographic systems that feature implementation vulnerabilities are often the result of a weak encoder. This can result in the presence of consistent signatures that are used to detect steganography. The advantage of this, is that signatures like this, are often simple to use as the basis for a classifier while offering a very high rate of accuracy (in many cases 100% due to the length of these signatures). This approach relies on the existence of signatures that are often introduced unintentionally, and do not exist in every stego-system; however, the occurrence of this can be seen across a wide number of stego-systems [141]. In cases where signatures exist but the strings are not consistent or long enough, feature extraction can be used to predict a deviation from the cover-object to obtain a pseudo-signature.

4. Reverse engineering: This can often be a valuable method in a system approach where an attacker wants to identify vulnerabilities directly within the steganographic system as opposed to a stego-object. This can lead to exposing cryptographic vulnerabilities, embedding flaws or system-based weaknesses that might leave consistent artifacts within a stego-object.

Implementation vulnerabilities are often an unintentional by-product of the encoding process by a stego-system. As a result, the developer may be unaware of such problems, or little effort may be given to correct them. If these vulnerabilities can be directly associated with the steganographic process, they provide an ideal avenue for steganalysis. These methods not only markedly simplify an investigation, but make it a highly practical choice over alternative methods of steganalysis.

System attacks use techniques that can be highly efficient when compared to alternative methods of steganalysis. For example, statistical attacks are often interested in a particular signal component of the stego-object and how it is modified by the embedding algorithm. This requires or assumes pre-existing knowledge of the embedding operation and is not practical in every case. The techniques used by system steganalysis often incorporate the entire stego-object. This extra information can provide valuable early insights into the presence of steganography.

Once the functionality of a steganography scheme has been determined and implementation vulnerabilities have been detected, a steganalytic approach can be

constructed. From a practical standpoint, system attacks are ideal. Many of the techniques have a low demand for computational resources and detection methods constructed from signatures are simple to implement. Accuracy for signature attacks are often high, particularly in cases when long, consistent fingerprints are present. This is favourable in cases where a significantly large number of datasets exist for analysis.

The advantages of system steganalysis have been discussed by Fridrich with regards to the steganalysis of the F5 algorithm, developed by Westfeld [155]. Shortly after the development of F5, many researchers focused their efforts towards statistical methods to detect the F5 scheme. These attacks were met with varying degrees of accuracy, some as low as 75% [51]). Niels Provos, however, pointed out that the compression component of the F5's embedding algorithm left a consistent signature within the headers of modified JPEG files that read "JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech". A signature of this length can be used to detect the presence of F5 steganography with a significantly higher rate of success. Within academic literature, system attacks have little surrounding literature.

2.3.1.2 Statistical Attacks

Statistical Steganalysis is recognised as the second major steganalytic attack. Statistical attacks observe deviations between the statistical properties of a stego-object and a cover-object by training a classifier to extract features in low dimensional space. When used against specific stego-systems, these attacks can be highly accurate. However, this can be resource intensive as feature space increases and statistical methods will not trace the presence of steganography to a particular stego-system.

A wide variety of attacks exist within the field of statistical steganalysis to analyse stego-objects for the presence of steganography. Regarding images and video, commonly used methods can include, image processing, first order statistics (histogram), or second order statistics (pixel or feature extraction and correlation) [27].

Working in low dimensional space is a major feature of statistical steganalysis.

A full representation of any digital media is vastly complex and will provide an overwhelming number of redundant features to a steganalyst. To overcome this, one can assume that it is best to work in a low dimensional feature space that is appropriate to the embedding algorithm, for example, in JPEG images, a statistical attack should work with the analysis of DCT coefficients in the transform domain as opposed to any spatial representation of the stego-object [75].

It is a general rule of thumb in steganalysis, that to successfully break a stego-system, an attacker must be able to predict the presence of steganography at a percentage higher than that of a random guess (50%). This has lead to steganalyic schemes being proposed that offer average detection rates accommodating for a high number of false positives. In a practical setting where storage devices can host thousands or even millions of files, examining a large number of suspicious files that may in fact be false positives may be too time consuming to successfully and practically implement.

2.3.1.3 Peak Signal to Noise Ratio

Peak Signal to Noise Ratio (PSNR) is often used as a feature of evaluation in steganography. It is a technique used to compute a peak-signal-to-noise ratio in decibels between two images and is used as a test for quality measurement. It is often used by authors developing steganographic schemes to demonstrate the quality degradation of their scheme between a stego-object and cover-object. A common scenario for this test is one where an attacker realises the existence of steganography within a file. In any case where a high PSNR is presented, it should be challenging for that attacker to identify the hidden information. A high PSNR ratio makes the difference between stego-object and cover-object almost indistinguishable [1]. It is often calculated as follows (defined by the mean squared error):

$$PSNR = 10.log_{10}\frac{MAX_1^2}{MSE} \tag{4}$$

Where MAX_1^2 is the maximum possible pixel value of the image. Typically value ranges for PSNR within compressed video and images will be between 30 and 50 dB for 8-bit representations and between 60 and 80 dB for 16-bit representations [154].

2.3.1.4 Chi-Square Attack

One early assumption in LSB steganography was that the values corresponding to LSB pixels would be completely random. This led to the belief that these bits could be replaced as they have minimal impact over the stego-object. This idea led to the development of tools for LSB steganography [103, 67]. Although the distribution of bits in the LSB plane appear random (as can be seen in Figure 17), these bits do in fact correlate to the full representation of the image. Through a series of visual attacks, Westfeld and Pfitzmann proved this to be true [158].

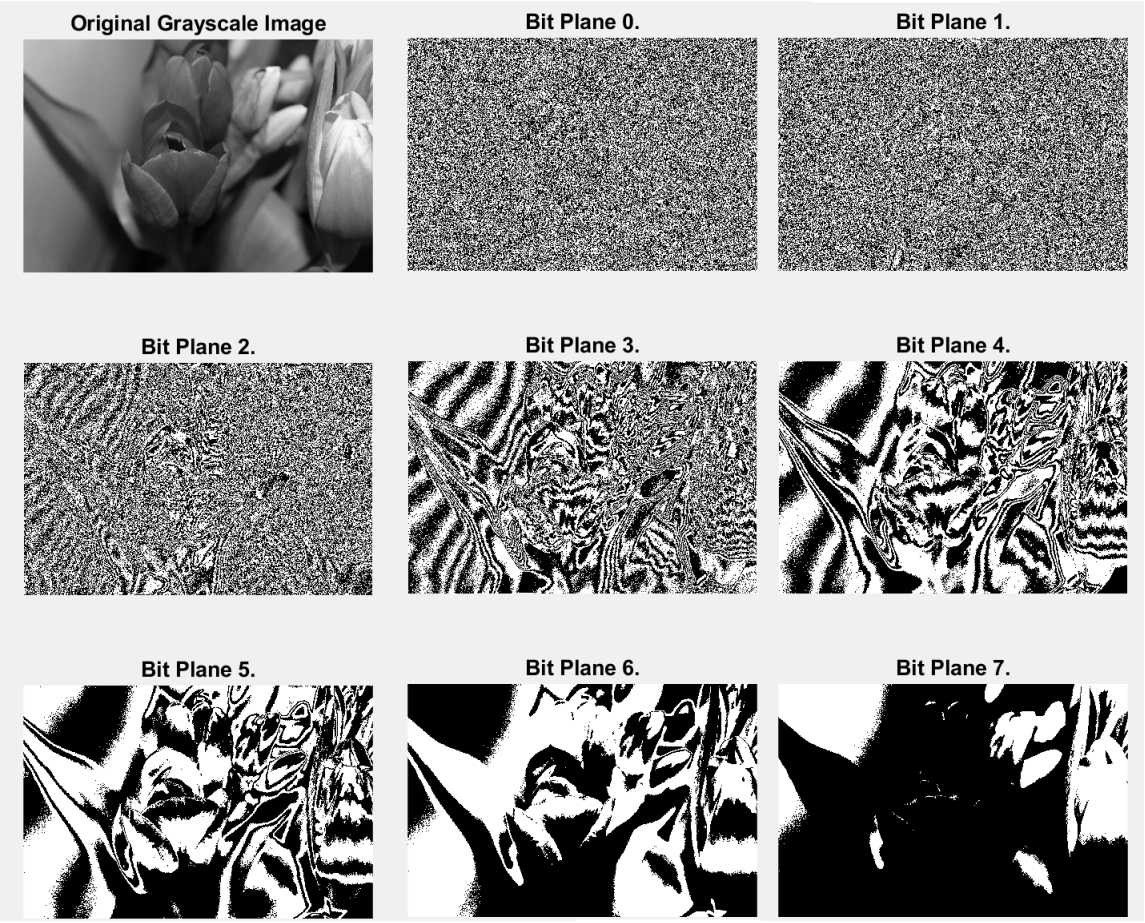

Figure 17: Representation of images in different bit planes (LSB to MSB)

One of the first statistical attacks to be successful against embedding in the transform domain was the Chi-square attack. The Chi-square attack works as

follows: For any transform domain scheme that equally distributes message bits throughout an image, a pair of values are created that have equal frequencies. The Chi-square attack compares the expected frequency distribution with that of the observed stego-object. This could be observed through a histogram as shown in Figure 18.

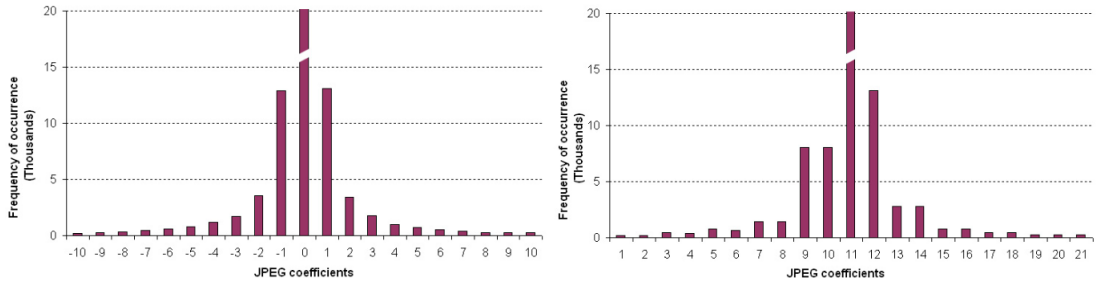

Figure 18: DCT coefficients of before (left) and after (right) embedded by EZStego [91]

The Chi-square method has shown to be effective in the transform domain against tools such as EZStego, JSteg and Steganos. This is because the Chi-square attack is based on analysing the higher-order statistics of the stego-object. However, it cannot detect steganographic methods that work in low dimensional feature space (OutGuess, F5).

2.3.1.5 RS-Analysis

RS-Analysis is a steganalytic method developed by Fridrich et al. to detect LSB and provide accurate message estimation [49]. At the time, this method was considered state-of-the-art in spatial domain steganalysis. This technique divides the processed image into small equally sized blocks and applies two mapping functions. The first, F_1 for $0 \leftrightarrow 1, 2 \leftrightarrow 3$, and $254 \leftrightarrow 255$ corresponds to the flipping of the LSB of a pixel. The second function maps pixel changes in the opposite direction to F_1, i.e., $F_{-1} : -1 \leftrightarrow 0, 1 \leftrightarrow 2, ..., 255 \leftrightarrow 256$. The mapping function F_1 is applied to each part of a block to track the ratio of blocks in which the magnitude of fluctuations increases (R_m) and the ratio of blocks with a decreasing fluctuation of magnitudes (S_m). Typically $R_m + S_m \leq 1$. Similarly, F_{-1} is applied to a part of each block to generate the parameters R_{-m} and S_{-m}. If the tested image contains

no secret data, then F_1 and F_{-1} should equally increase the magnitudes of fluctuation. In any case where LSB steganography is carried out R_m and S_m become more linear whereas a wider deviation appears for R_{-m} and S_{-m} as a result of the LSB operation and the mapping functions counteracting one another.

2.3.2 Forensic Steganalysis

Suspicion of the existence of steganography can arise in many ways. This could be from the presence of a steganography tool on a computer, or the transfer of files in a suspicious manner (Internet or storage). One objective will be to identify files that could be considered suspicious, in that a steganalyst suspects the presence of steganography within. The second objective will be to examine and understand the embedding algorithm that is used to hide secret messages. If an investigator is capable of doing this, then a steganalytic scheme can be developed to detect the presence of steganography and the stego-system is considered broken. This is the classic model for steganalysis and once the presence of hidden information is proven, little more is required. However, this approach can be extended to incorporate further steganalytic tasks such as identifying the stego-key and extracting any hidden contents to a readable state. This falls within the sub-domain of forensic steganalysis [56]. Many of the techniques within this field of steganalysis belong to digital forensic analysis and can feature many other investigative tasks. These can be summarised as follows:

1. Use of forensic tools and techniques to detect the presence of steganographic systems: This can involve the use of specially configured forensic tools that monitor filesystems for specific contents. These tools can be adapted to look for well known stego-systems. This however relies on the pre-existing knowledge of a certain tool.

2. Identification and examination of steganographic hosts - This will typically involve searching for suspicious systems that may be harbouring steganographic tools or contents, such as computers, storage devices, or servers/webpages.

3. Development of automated tools for the detection of steganography: One particular challenge facing forensic steganalytic investigations, will be time constraints. Providing automated and highly accurate solutions to this can significantly benefit these investigations in identifying steganographic systems and the presence of steganography. This is often facilitated by the use of system steganalysis.

4. Profiling of steganographic and steganalytic software: A significant foundation of pre-existing knowledge regarding steganography and steganalytic tools will be a critical component of forensic steganalysis investigations. It is particularly useful in any case where examination time can be significantly reduced through access to necessary pre-existing resources and documentation regarding known steganographic schemes.

5. Observation of steganographic networking: The pervasive nature of media files throughout the Internet make them ideal candidates for steganography. It can be important to observe how steganography is applied to these files and transferred throughout the Internet by networks of individuals. One subsection of this is the restriction of steganographic networking by file sanitization. Primarily to remove any potential for steganography to occur through file compression, thereby removing sensitive hidden contents.

6. Identification of steganographic key: One underlying principle of steganographic security is that the key should always be kept hidden. Once found, the scheme is no longer secure. This can be a task of forensic steganalysis to obtain or predict the key through various methods such as brute force attacks.

7. Extraction of hidden content: This often relies on a precursor of identifying a secret key for a stego-system. However, particularly vulnerable schemes can be broken without the presence of a key. Embedded content is often hidden in an encrypted state, so this task will require the efforts of cryptanalysis.

Under this model for forensic steganalysis, a scope is defined that allows forensic analysis tasks to be applicable towards steganalysis for a broader investigation.

This domain can also allow researchers to present a broader range of steganalytic approaches. Research carried out by Trivedi and Chandramouli presented a forensic steganalysis technique to estimate a secret key within sequential steganography [148]. The authors' goal was to propose a technique applicable to any type of sequential message embedding, however the primary focus of their work was carried out across spread spectrum embedding. This assumed that the key could be determined through the beginning and the end of the subsequence modulated during embedding of the secret message. Similarly, Fridrich et al built on the work of Trivedi and Chandramouli [56] to be applicable to JPEG schemes particularly using LSB and +1 embedding within the spatial domain.

Early work in spatial domain steganography led to the conceptualisation of these steganalytic techniques that would expand the single goal of steganalysis. Schemes were presented to perform forensic techniques that could be applied to steganalysis. This included estimating the length of secret messages and determining the existence of a stego-key. These techniques were both presented by Fridrich [56, 48]. In the former case, this scheme offered a competing approach to similar (at the time) state-of-the-art techniques, outperforming Sample Pair-Analysis (SPA) and giving comparable results to RS-analysis (RSA).

Work carried out by Niels Provos and Peter Honeyman looked into the development of a tool (StegDetect/Break) paired with a webcrawler to detect the presence of steganography throughout the internet [123]. Although much of their work is primarily focused towards purely statistical steganalysis, many aspects of their investigation fall under a forensic theme. Firstly, there being a target host site the authors wanted to investigate for the presence of steganography. Secondly, brute force attacks were used in an attempt to gain access to embedded contents upon finding suspicious files. Ultimately, they developed automated tools for the detection of steganography. However, no content was found throughout their investigation.

2.3.3 Information Adaptive Approaches to Steganalysis

As a detection problem, steganalysis can be approached through a variety of hypothesis-testing scenarios. The information available to an attacker during an investigation can markedly impact the steganalytic process (time, resources, method of attack required). These scenarios consider the information that is available to an attacker or warden during an investigation [37].

- Known-carrier attack: This attack relies on the existence of a pair of objects, the stego-object and the cover-object. It is assumed that there is no access to the steganographic scheme to train a classifier. If only a single object pair are given, this limits the potential for statistical steganalysis. In this situation, system attacks are favourable, as they can provide quick insights into the functionality of a particular scheme.

- Known-message attack: This is an unusual scenario where only the hidden message is known. If a stego-system is following recommended practices and is encrypting data before embedding, then the significance of knowing the message can be greatly diminished.

- Chosen-message attack: Both the secret message and the embedding algorithm used to hide the message are available to an attacker. This can provide a useful sample for testing and comparison.

- Fully blind attack: This is perhaps the most challenging scenario for steganalysis. An attacker only has information regarding the stego-object. They will have no access to a cover-object or stego-system. This limits the potential for many statistical and system attacks, and a steganalytic will have to work by identifying arbitrary features of the stego-object.

- Chosen-steganography attack: The stego-object and the embedding algorithm are known.

- Known-steganography attack: In this attack, all components related to steganographic activity are typically known, this includes the stego-object, the embedding algorithm, and the cover-object.

2.3.4 Steganalysis as a Testbed for Steganography

Although the primary goal of steganalysis is to prove the existence of steganography, it is important for an author to develop a robust steganographic scheme. One step in achieving this is by using steganalytic techniques to test and demonstrate the security and robustness of proposed steganographic systems. An embedding algorithm should not only be evaluated by the visual imperceptibility it offers. A secure, complex scheme should also demonstrate resistance to steganalysis as means to go beyond basic steganography as the primary goal is to now demonstrate resistance to steganalysis. This concept can be captured through five well known techniques discussed by Sadek et al. [127]. Their discussion, though aimed primarily towards video, is certainly applicable to other digital forms of steganography and steganalysis. It illustrates the following ideas:

- Perceptual transparency: This is the fundamental concept of steganography upon which the academic field is built. Once the existence of hidden information is reliably discovered, the steganographic scheme is considered broken. Any system that poorly implements the embedding algorithm can often be directly visible (e.g., a flicker in a video, excessive noise in a picture, or arbitrary audible noise in an audio file). Steganalysis in this case is simple and can usually be achieved through visual analysis.

- Histogram type analysis: Histogram techniques are well known, especially amongst traditional DCT-based schemes where histogram distortion caused by manipulation of coefficients could be utilised for steganalysis [51]. Formally, histogram attacks are used to observe changes in an image or frame from a statistical perspective. These generally include not only simple measures such as the average, the mean, the maximum and minimum, the variance, entropy, chi-square etc., but also higher order ones such as skewness (asymmetry of the curve) and kurtosis (sharpness of the peak).

- Robustness against compression: Testing a modified carrier file's susceptibility to compression is an important factor. How it deals with various compression algorithms will determine how easily the stego-file can be distributed and through which channels.

- Bit error rate: This is a calculation linking the stego-content that has been embedded and the information that can be successfully extracted. In cases where data is likely to be lost (due to compression), additional message bits can be embedded to ensure that the secret message maintains its integrity.
- Robustness against manipulation: Signal processing changes, whether accidental or intentional, can impact any contents hidden within a file. Determining the resistance to these types of attacks is important in many settings.

The steganalytic techniques discussed above are not only useful when attempting to identify the presence of steganography, but are often used to validate the strength and robustness of a given embedding algorithm. This is an important practice when conceptualising new schemes to ensure that it will offer basic resistance to steganalysis.

2.3.5 Steganalysis in the Real-World

The majority of research in steganography and steganalysis is conducted in what is often defined as a laboratory environment. In this setting, any scheme presented can be constructed under a set of parameters that define the experiment. This is often needed because the real-world is highly variable and difficult to model. Because of this, there is a lack in research endeavours that evaluate the practices of steganography and steganalysis outside of a laboratory environment. However, several stego-systems and steganalytic tools have been evaluated by researchers. Work by Khalind et al. [89], looked at the false positive rate of the StegDetect steganalysis tool developed by Niels Provos. Their research helped determine the most efficient methods for use of the tool to minimise the risk of false positive occurrences.

Similarly, research by Ker and Pevny [86] evaluated the "universal pooled steganalyzer" presented in [87]. Their research aimed to determine that the presented method could be applied to a number of steganography embedding algorithms. In addition, their work attempted to consider the problems surrounding a payload spread across multiple file types. The emphasis of this research was that it would focus on practical applications and could be tested over real-world data.

2.4 Video Steganography

Video content is ubiquitous, it is a format that has been given multiple platforms for widespread distribution through the Internet. Media sharing and hosting websites have transformed the way in which users access digital content to make it more accessible. The pervasive nature of video means that these files can be easily shared and distributed. This provides an ideal avenue for steganography as secret messages can be embedded in digital video and exchanged with rarely any reason to justify suspicion.

The large embedding capacity available to video is a highly desirable component for steganography. By accommodating larger messages, video steganography can reduce the risk of detection as modifications can be made over a larger spectrum of redundancy. This has become one of the major advantages associated with video steganography when compared to alternative forms of media.

Video files offer a high amount of flexibility for those constructing a steganographic scheme. Digital video combines spatial and temporal dimensions to produce a three-dimensional array. This provides potential for steganography from the spatial domain by borrowing techniques from image steganography, and in the temporal domain, where compression features are often used [71, 135]. With regards to video steganography, research is more commonly carried out in the temporal domain. Any steganographic redundancy identified in this domain provides an ideal basis for a steganography scheme. This is largely related to compression components of various video formats such as MPEG-2, MP4, AVI and FLV. Figure 19 shows the common methods of embedding for video steganography.

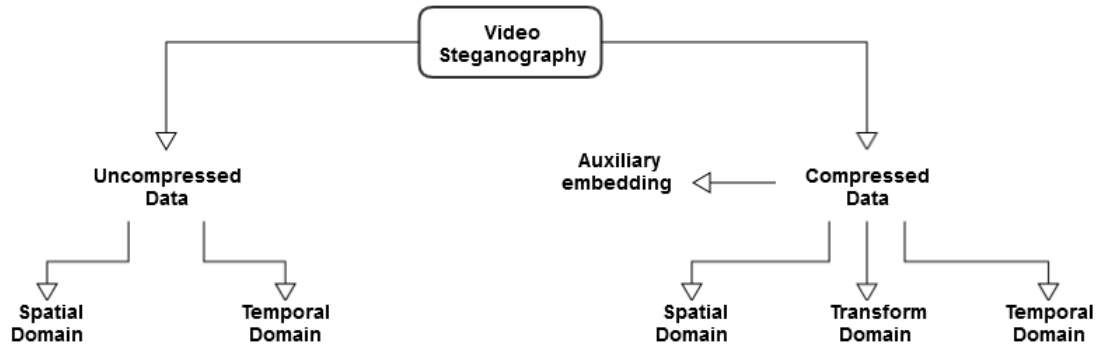

Figure 19: Overview of techniques available for video steganography

As a form of digital representation, video streams contain a high degree temporal redundancy. This is commonly seen with macroblocks, which are groups of pixels. Adjacent frames can share identical or highly similar macroblocks between them, which for the purposes of compression should be minimised [128]. Managing and reducing temporal redundancy is a task handled by the compression algorithms of video formats to keep each file at a practical size.

It is generally considered that steganography schemes for video function in either compressed or uncompressed data [68]. Although research is actively conducted in both, there are significant challenges for the latter. Uncompressed data that contains secret information may be compressed at a later stage. Therefore, a steganography scheme must be robust enough to survive potentially significant changes. Because of this, it is far more common to see video steganography more actively pursued within compressed video formats. In this state, the content is already compressed, so it is unlikely to be exposed to severe changes in the future. However, it is important to ensure that if lossy compression does occur, then these features must keep the embedded message, as any secret message bits stored within the lossy features will be sanitised. The early research into this problem was presented by Chae in 1999 [24].

As seen with image steganography and particularly with video, many of the early and modern schemes are influenced by conceptualisations of watermarking algorithms [30]. These techniques are often adapted for steganography but rely

less on robustness, instead focusing on imperceptibility and capacity.

2.4.1 Steganography in Uncompressed Data

Uncompressed files have a large amount of redundancy that is lost during future compression to save storage space. Because of this, steganography in uncompressed video data has the advantage of high capacity. However, it is uncommon to find video in this state, as stored and shared offers few practical benefits.

Attempts have been made to advance the development of steganography in uncompressed video. Xu and Ping [161] presented a method in which a secret message is embedded into low-frequency DWT coefficients. However, this method lacks robustness and is sanitised by compression techniques.

Block error correction schemes for steganography have been examined over uncompressed video. The Hamming-code technique in particular, which is a well-known method for error detection, has been tested extensively over image steganography [11]. This technique has been extended to video where frames are isolated and treated individually with an embedding algorithm that uses the Hamming code technique to perform video steganography. In this method, the cover-object is pre-coded with the addition of adding extra data so that the minimum amount of redundancy is achieved. This extra data is called a codeword which will consist of length n bits. The codeword contains parity information to accommodate a message that will be embedded of length $(n - k)$ bits where k is the predicted message. The advantage of this technique is in its ability to detect and correct a single bit error of added parity. A proof of concept of this technique over uncompressed video was presented by Mstafa and Elleithy [108]. It showed that the Hamming code over video provides a high PSNR ratio of 51 dBs, when maintaining a relatively small payload (16-Kbits per frame). However, the authors had not considered the robustness of their technique and did not test if it would survive compression.

2.4.2 Spatial and Transform Domain Techniques in Video

For image steganography, spatial domain techniques are common as they are considered simple to implement. However, they are often vulnerable to cover modification and statistical steganalysis. Transform domain techniques improve on this by offering robustness and some level of resistance to statistical steganalysis. With the increased popularity for video steganography, researchers have extended spatial and transform domain methods to video. MSU StegoVideo [39] is a stego-system for spatial domain steganography over video. This tool embeds data in the lower bit-planes of individual frames to perform spread-spectrum steganography in AVI video files. Besides the steganalytic vulnerabilities, one challenge for spatial domain steganography in video is to maintain a desirable capacity. These schemes often have to encode the message with error-correcting code in multiple locations to resist potential video compression. This additional measure can significantly reduce the capacity available for any given message to be hidden.

The major advancements in transform domain embedding come from the efforts conducted in image steganography. In recent years, these methods have been extended to video. Transform domain techniques, particularly DCT-based steganography has been applied to video with the MPEG-2 format. Early video steganography schemes conceptualised this by embedding secret messages using a DCT-based approach [24]. To perform this, the secret message and target frames from the cover-object are transformed into 8x8 macroblocks. Coefficients from the secret message are quantized and encoded then embedded into the corresponding coefficients from the cover-frame. This method of embedding still faces the challenges of video processing, where embedded data is lossy and susceptible to file modification. However, if each frame is treated individually, there is potential for higher capacity for an embedded message.

2.4.3 Pixel-Value Differencing

Tri-Way Pixel-Value Differencing (TPVD) is an embedding technique adapted from the original image steganography method called Pixel-Value Differencing (PVD) [159]. For the image scheme, the embedding operation segments a file into non-overlapping pairs of neighbouring pixels. A calculation is given to the

prediction error between them and is used to hide the secret message. This provides relatively high capacity while maintaining imperceptibility. For video, this scheme uses the underlying concept that locally complex regions will yield the highest PSNR. Smooth areas provide small pixel-value differences, whereas locally complex regions provide a wider deviation between neighbouring pixels and are optimal for data embedding. TPVD [137] is a video steganography method that fragments and classifies vertical, horizontal and diagonal directions. The main advancement of TVPD over PVD is that PVD isolates two-pairs of neighbouring pixels along a one-directional edge. TPVD extracts four-pairs of neighbour pixels to provide a wider array of prediction error. Figure 20 demonstrates how four-pair pixels are represented.

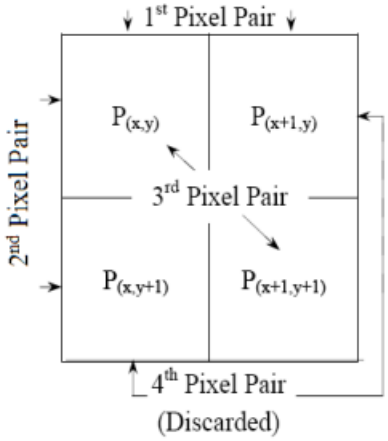

Figure 20: 4-pair neighbouring pixels, from [137]

This scheme offers high capacity, an improvement over the original PVD scheme by embedding data in a wider array of image features. The algorithm for TPVD specifically targeted frames with maximum scene change to exploit any temporal redundancy.

2.5 Motion Estimation Schemes

Capacity is one of the three desirable features of any successful steganography scheme. Video files are a popular media format for this due to the high embedding

capacity offered by most schemes. Motion estimation and vector modification schemes provide this due to the significantly large number of vectors available per-frame. Motion vectors are a key component in video compression to reduce temporal redundancy between successive frames [70]. The Motion Vectors (MV) are bi-dimensional pointers used to represent the position of a macroblock in one frame based on the position of the macroblock in another frame, as shown in Figure 21. Steganography by MV modification is highly desirable as it is resistant to video processing techniques and provides a robust solution.

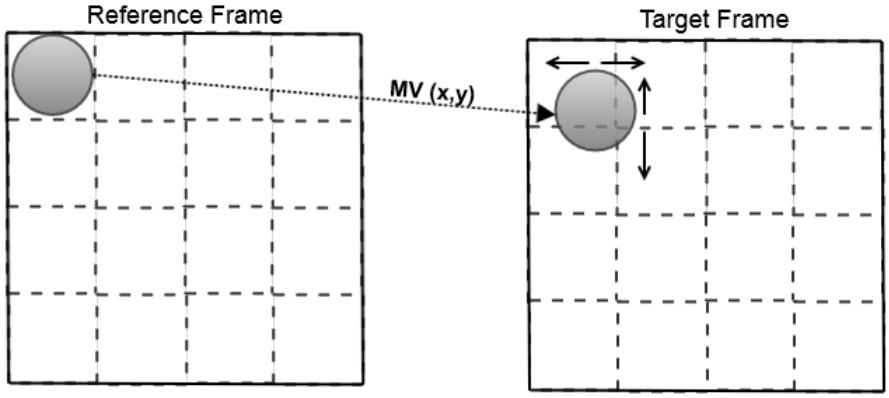

Figure 21: Motion vector representation

MV steganography is a technique observed in the temporal domain of video files. MVs are a component of video compression that describe movement between frames as a method of identifying and reducing redundancy. For this, each frame is typically divided into 16 x 16 MacroBlocks (MBs), for which, each MB is taken and the reference frame is analysed for a matching MB to pair it to in the target frame. The position estimation between the reference and target frame for an MB is described by a MV. These compression features are used for steganography by embedding secret message bits in the LSB of MV components.

A common compression technique for spatial redundancy in video frames is to limit the amount of data represented in each frame. The MPEG standard is a common example of this, where motion estimation is achieved by splitting frames into three categories; I-frames, P-frames, and B-frames, shown in Figure 22. Intra-frames (I-frame) is the key frame of the three, it is completely self-referential,

meaning that it only relies upon the information stored within itself. Prediction-frames (P-frames) reference previous I-frame or P-frames to identify redundant data. This leads to the generation of forward-only MVs. Bi-directional frames (B-frames) result from an encoder that is generated using previous and/or future reference frames resulting in bi-directional MVs. The resulting MVs are adaptive and any particular values they hold are dependant on how complex the given scene of motion will be.

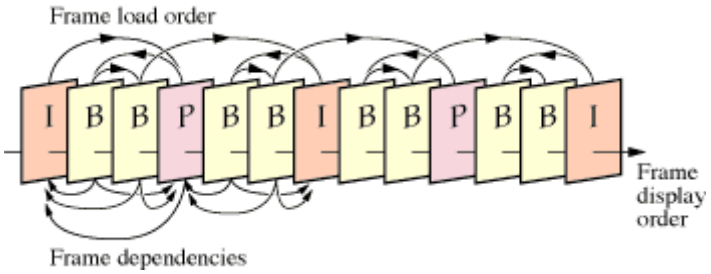

Figure 22: Illustration of frames in MPEG compression, from [66]

Early work in motion estimation steganography was influenced by research in watermarking [82, 168]. The underlying principle is to embed data in the least significant bits of a candidate MV's horizontal or vertical components, where the magnitude is above a predefined threshold. The original research experimented to identify the ideal threshold value. It was discovered that lower thresholds would accommodate a higher capacity but introduce distortions, therefore a careful approach should be taken to achieve high capacity without affecting the object's integrity. Embedding data within high threshold magnitude motion vectors would mean that data is typically embedded within noisy areas to conceal any distortions. In 2006, this method of embedding was adapted for the purposes of steganography by Xu et al [162]. This was achieved in parallel to similar work carried out by Fang and Chang [43] who instead conceptualised a method to utilise the phase angles of a motion vector. Both researchers followed the same principle, that modifying MVs with larger magnitudes would introduce fewer distortions.

A typical scheme for MV steganography can be modelled as a function that is inserting additive noise into the horizontal and vertical components of the first motion vector for target frames. This can be described similarly to the equation

formulated by Wang et al [151].

$$\begin{cases} SV_{k,l}^{h} = V_{k,l}^{h} + N_{k,l}^{h} \\ SV_{k,l}^{v} = V_{k,l}^{v} + N_{k,l}^{v} \end{cases} \tag{5}$$

The example shown in eq.(5), describes a standard formulation of MV steganography as additive noise $N_{k,l}^{h}, N_{k,l}^{h}$ into the horizontal $SV_{k,l}^{h}$ and vertical $SV_{k,l}^{v}$ components of a MV [151]. $k = 1, 2, ...N$, and $l = 1, 2, ..., L_k$, where N describes the frame number and L_k describes the number of MVs in the kth frame.

Additional protocols can be added to manage and control any potential distortion, or determine when a MV is a suitable candidate. For an improved MV-based scheme, the embedding algorithm integrates a control mechanism for MV steganography to embed within specific X or Y-components of a target motion vector of the kth frame. This could be a feature controlled by the user or predetermined by the stego-system. This can be formulated as shown in eq. (6).

$$\begin{cases} SV_{k,l}^{h} = V_{k,l}^{h} + C_{k,l}^{h} N_{k,l}^{h} \\ SV_{k,l}^{v} = V_{k,l}^{v} + C_{k,l}^{v} N_{k,l}^{v} \end{cases} \tag{6}$$

Where $k = 1, 2, ...N$, and $l = 1, 2, ..., L_k$, where N describes the frame number and L_k is the first MV of the kth frame. The additional control adds $C_{k,l}^{h}$ and $C_{k,l}^{v}$ that are either user or system controlled features to determine whether the horizontal or vertical components of the MV as used.

This results in a stego-object being visually indistinguishable from the cover-object, as shown in Figure 23, where data is embedded using control features for the first x or y MV in each target frame.

Figure 23: Cover-object and stego-object result from the Steganosaurus MV tool [126]

It was later realised that a selection rule based solely on MV magnitude would not be capable of ensuring minimum prediction error. In 2011, Aly presented a selection rule in which MVs associated with large prediction errors are selected to accommodate this [7], thereby improving steganographic security.

An alternative scheme for MV steganography was proposed, that was inspired by Fridrich et al's model for Pertubed Quantization (PQ) [53]. The adapted scheme for video aimed to apply the concept to the temporal domain of video, by embedding message bits at the same time that motion estimation is determined. In addition, only suboptimal MVs were selected for embedding. As no additional modifications are required to the stego-object, the compressed data appears to be naturally generated. This would preserve the statistical qualities of modified MVs and increase steganographic security [23]. To test the steganalytic resistance of this scheme, Cao et al. subject their algorithm to a previously presented steganalysis scheme, tailored specifically for MV-based steganography [167], it had shown to outperform competing MV schemes.

2.5.1 Challenges for Video Steganography

Typical video files are an amalgam of three data streams upon which, generate a functional video. These are frames, audio, and metadata. Each feature provides a unique avenue for steganography. When compartmentalized into these three components, stego-systems can adopt features from pre-existing stego-systems. For steganography, this presents an interesting range of possibilities. The high temporal redundancy offers image steganography techniques a unique advantage where data can be split among many frames as opposed to a whole message directly in a

single image. This increases the imperceptibility between cover-object and stego-object by keeping PSNR quality high with a generally indistinguishable solution. Similarly, the audio components of a video file would imply that audio steganography schemes are applicable. Finally, the metadata used to describe the structure of a video file has been used for steganography [141]. However, video steganography schemes can also be subject to a wider variety of attacks and limitations to the embedding operation. Therefore, one would need to confidently understand how any embedded data might be effected by compression, change of frame rate, format switching, and modification of frames during video processing.

The evolution of video steganography has followed a similar theme to that of more traditional approaches. Modern schemes identify redundancies within various components of a file, this could be within individual frames, audio data, or metadata. Over time, schemes evolve in complexity by focusing on resistance to steganalysis. Embedding algorithms are extended to use cover codes or optimise the impact of an embedding algorithm to present a more steganographically secure scheme. Ultimately, one should aim to achieve ideal embedding efficiency without compromising the integrity of the carrier file and diminishing imperceptibility.

2.6 Video Steganalysis

Intuitively, the field of steganalysis must be advanced for any domain that its counterpart can be applied to. The emergence of video steganography has inevitably led to steganalysis of video. The pursuit of video steganography is justified by the many advantages it offers. Capacity and flexibility for steganographic schemes are shared in parallel to the popularity of video files as a form of media. As a solution, video steganography provides a wide variety of embedding algorithms that can be customised to any particular component of a video file that contains steganographic redundancy. When compared to traditional steganalytic methods, video steganalysis will have to adopt a flexible approach. For a blind attack, it has to be assumed that a steganographic scheme can hide secret messages within any aspect of the video file. Generally, this means that an attacker would need to account for the possibility of spatial and frequency-based image schemes, temporal compression schemes, audio schemes and metadata/auxillary schemes.

This section provides an overview of existing work in the field of video steganalysis. Although relatively new, much of the steganalytic focus has been directed towards motion estimation schemes, particularly MV steganography.

2.6.1 Motion Estimation Steganalysis

Schemes for MV steganography were first presented in 2006 [162]. However, it wasn't until 2008 that the first steganalytic scheme capable of detecting MV steganography was presented. Work by Zhang et al [167] detected the first MV schemes that had been constructed as additive noise to the horizontal and vertical components of MV. Although successful, the accuracy of this scheme is dependant entirely on the video used. Positive detection varied between 87.6% and 100% while negative detection varied between 76.8% and 100%. These changes in accuracy were based heavily on the existence of frames with complex motion. Locally complex regions were difficult for the detection scheme to attack.

Su et al [147], implemented a similar steganalytic scheme to analyse the early ME watermarking concepts [82, 34, 168]. Their steganalytic scheme functioned by calculating the difference between two adjacent MVs and extracting features directly from the statistics of those differences. It performed similarly to the work of Zhang that found the success and performance of such a scheme relied heavily on the distribution and properties of particular MVs. This created a challenge in the field to find a more consistent method for ME/MV-based steganalysis.

In 2012, Cao et al, attempted to improve upon existing MV-based steganalysis by using a reversion technique that decompresses the stego-object into a spatial state [22]. The data is then compressed back into its initial format with no further embedding involved. This revealed that the altered MVs would revert to their prior values. At this point, features can be extracted and any observation of MV-reversion would indicate the presence of steganography. Their results had shown to outperform other existing techniques at the time. However, the scheme relies heavily on the compression algorithm for MPEG being used in both stages of initial compression and recompression. This would mean that practical use is limited and any deviation from the exact embedding mechanism could heavily influence their results.

MV schemes are generally constructed as targeted attacks. This means each steganalytic scheme will work only on a particular MV-scheme. If the steganographic scheme is advanced, the performance of a targeted attack will drop or fail completely. Attempts have been made towards universal steganalysis of MVs that would be applicable to a wider array of embedding schemes. This was the culmination of work carried out by Wang et al [151], who presented the Add-or-Subtract-One (AoSO) scheme for video steganography. Their work demonstrated that MVs extracted from a cover video will be least locally optimal according to the SAD (Sum of Absolute Difference) values. This slight influence that modified MVs imposed on the SAD, can be used to calculate the optimal SAD values from a localized region of MVs to extract features accordingly.

2.6.2 Intra-Prediction Mode (IPM) Steganography and Steganalysis

A large number of steganography methods embed secret information during the encoding process. As a result, these methods have high computational complexity and can be difficult to model in real-world parameters. Especially when these methods are slow and ineffective in a practical environment.

Research in steganography by Intra-Prediction Modes has been presented by Hu et al. [72], and by Yang et al. [163] as an improvement to Hu's method. The motivation of their research was to develop an embedding algorithm that resolves many of practical challenges for video steganography. The method presented by Hu aimed to perform steganography over H.264 objects, by targeting the components of I-frame intra-prediction. These features are part of the H.264 coding standards and were introduced to reduce spatial redundancy and improve compression efficiency.

This method of steganography is performed by embedding a single secret message bit into each qualified 4x4 luma macroblock. To do so, the embedding algorithm would exploit 4x4 Intra-Prediction Modes (I4PM) and hide secret message data by modulating the intra-prediction modes of qualified intra 44 luminance

blocks. Because of the high number of macroblocks for a given I-frame, IPM algorithms demonstrate low computational complexity and offer considerable steganographic capacity.

Research by Zhao et al. [172], attempted steganalysis of IPM steganography. Their research presented a low-dimensional method to detect non-optimal (modified) IPMs, under the assumption that H.264 encoding generates optimal IPMs modulated by specific encoding control algorithms. Based on this assumption, if spatially compressed IPMs are decompressed, then stego-modified IPMs would likely revert to their prior optimal value when subjected to a encoding algorithm.

2.7 Conclusion

In summary, this chapter has presented an overview into the background and state-of-the-art surrounding steganography and steganalysis. There has been a focus on topics that are relevant to this thesis, in addition to a comprehensive overview of many well-known and influential techniques. This Chapter has illustrated how researchers in steganography and steganalysis have advanced the field to its current state-of-the-art and identified areas that require further exploration.

One of the most underlying principles of steganography is that it is inherently advanced through the practices of steganalysis. This has been observed throughout the literature review, as new advancements in steganography are often developed to combat novel steganalytic methods and provide an optimal scheme for hiding secret information. These methods often aim to achieve high capacity while maintaining a low rate of detection against strong steganalytic attacks, this is a complex task, as a compromise is often needed between these two features or the steganographic method will risk being detected.

The application for steganography in the real-world are discussed to highlight unexplored avenues of research in steganography and steganalysis. From this, it is clear that system-based attacks in steganalysis are a relatively under-researched area and advancements in this domain will help develop the field, particularly for practical steganography and steganalysis. The two major types of steganalytic attack (statistical and system attacks) are discussed to show the differences between them and illustrate where they can be effectively applied. Further discussion is

given towards forensic steganalysis, a method developed to extend the scope of steganalytic investigation to include the extraction of a secret message.

Avenues of research for video steganography and steganalysis have been discussed at length. It has been shown that there is a particular focus given to MV embedding as a form of steganography. This method is certainly interesting for steganography, but its practical applications have not yet been fully explored and evaluated. This is addressed further in the following chapter (Chapter 3).

Chapter 3

Methodology

3.1 Introduction

Research in steganography and steganalysis has advanced rapidly in the past two decades [174]. However, there are still major gaps in the field that have led to stego-systems being a security concern for its users.

Much of the existing literature in steganography models the field as a data hiding and detection problem. Because of this, statistical methodologies have a justified but at times overshadowing impact on the field. These methods can be effectively applied in what is often defined as a laboratory environment [88], but faces challenges in real-world scenarios. When applied to a practical stego-system, the impact of steganography schemes over a cover-object is not simply exclusive to the embedded message, as such, a wide array of features must be considered. This lack of consideration is what gives potential to system attacks.

In laboratory environments, commonly accepted frameworks exist to define conditions for steganographic experimentation [152, 20, 17]. These remove the ambiguities and some of the difficulties surrounding the real-world problems that might impose additional mathematical challenges. In these environments, parameters can be adjusted so that the requirements of a steganography scheme can be more easily applied. However, steganography and steganalysis are real-world problems and if improperly implemented, will lead to damaging consequences due to their practical nature. For example, detection accuracy of a steganalytic method

can vary greatly depending on the scheme and cover. In some cases, detection methods provide accuracy as low as 51.30% over known stego-systems [93], which is still, however, considered a successful attack. In a practical setting, particularly where millions of files may require analysis, detection accuracy becomes more pressing and the effectiveness of the rule for successful steganalysis (>50% positive detection rate) can become diminished. In addition, if a stego-system is vulnerable to multiple steganalytic attacks, performance should be the determining factor for the optimal attack.

Furthermore, a focus on steganalysis as a detection problem has led to the development of stego-systems that devote considerable effort to provide resistance to statistical attacks. This is seen, for example, with OpenPuff [113] where the stego-system provides resistance to known statistical tests (Chi Square, Entropy, Monte Carlo, Compression, Mean Value). While this is an advantageous feature for a stego-system, it frequently comes with a lack of consideration for system vulnerabilities. This can lead to the development and even popularisation of many stego-systems that might be vulnerable to system attacks. From this, it is important that the research and experimental methodology followed through this thesis adopts a robust framework to ensure that any formal outputs are of high quality.

This chapter presents and discusses the experimental methodology used in subsequent chapters (Chapter 4, Chapter 5, and Chapter 6). The rest of this chapter is as follows: Section 3.2 discusses the testing and evaluation framework for this thesis. Section 3.3 provides an overview and discussion of the dataset selection and generation procedures to justify the primary choices for constructing the necessary datasets. Section 3.4 presents the experimental methodology that is followed in this thesis, in this section, emphasis is given to explore each step of the experimental process. Finally, a brief summary of the chapter is discussed in Section 3.5.

3.2 Constructing a Testing and Evaluation Framework

This thesis examines the practicality of system attacks, and a series of high-level steganalytic schemes are proposed and constructed to demonstrate the scope and limitations of system steganalysis. To evaluate each scheme, a framework for testing should be carefully defined and used. This will ensure that testing does not introduce any bias and does not lead to false conclusions. The test framework can be adequately described through three components as follows:

Component 1 Datasets must be produced for testing and validation. For this, the method of dataset creation must be consistent, comprehensive, and reproducible. Each dataset element is a file pair consisting of a cover-object and a stego-object. These are used to determine detection accuracy and false positive rate of any proposed steganalytic schemes. For each dataset, secret messages and cryptographic keys are randomly generated. This mitigates the risk of flagged signatures being linked to weak cryptographic implementations and ensures that traceable steganalytic features are generated from the system vulnerabilities.

Component 2 Accuracy testing must be comprehensive. To ensure confidence in the accuracy of a steganalytic scheme, enough datasets (minimum of 100) must be constructed to determine with a certain degree of assurance that a steganalytic scheme is detecting the presence of steganography with suitable accuracy. This is in contrast to the standards set for typical steganalysis, in which a steganalytic scheme can be deemed successful provided it detects steganography at a rate higher than that of a random guess (50%). To prove that no bias is introduced in these tests, at least 50% of files used for the generation of datasets should be randomly obtained. Detection accuracy is calculated as a frequency of occurrence based on the number of instances in which a stego-object is correctly classified.

Component 3 Steganalytic schemes derived from high-level features are susceptible to false positives (as with many other steganalytic methods). Comprehensive testing should be carried out for any steganalytic scheme presented. Tests generated for false positives should be performed over a significant number of files (minimum 1000). To ensure that no bias is introduced, datasets generated from

files will consist of at least 50% randomly obtained files. False positives are calculated as a frequency of occurrence based on the number of instances in which clean object is misclassified as a stego-object.

3.3 Dataset Selection and Generation

The data used in the experimental phases of this research consist exclusively of digital media content. This includes the use of images (JPEG), document (PDF), audio (WAV), and video (MP4, MPEG, FLV, and AVI) to support the testing and evaluation of the constructed methods and to help answer the proposed research questions presented in Chapter 1. The content of each dataset component is relevant to the requirements for each associated experiment. Typically, stego-systems offer a variety of embedding operations across different file formats. To provide a comprehensive analysis, it is important to ensure that a significantly large and diverse collection of relevant file types are available for testing. As such, appropriate file formats were identified and used as a filtering option when constructing the Datasets.

To ensure that the experimental methodology is robust, steps have been taken to ensure that no bias is introduced in any part of the experimental procedure. From a scientific perspective, it is important to ensure that specific data objects are not being picked with the intention of giving 'expected' results. The most suitable way to address this concern is to randomly obtain a dataset that is gathered without influence from the experimental participants. A pseudo-randomised webcrawler using a shuffle function provides an ideal means to obtain a video dataset unaffected by selection bias. However, video files can be computationally expensive in resources and a limit must be imposed to ensure that experiments are conducted within the projects allocated time-frame. To address this, the webcrawler was given a full week to pseudo-randomly download videos from the Internet and successfully obtained 5,000 video files. This can be labelled the DMOZ dataset as the DMOZ archive was used to psuedo-randomly generate a list of URLs.

The second challenge of dataset generation through this methodology relates directly to the pseudo-randomly downloaded data. It is important to question how the author of any experiment conducted this way would be certain that any

downloaded contents are not already affected by some form of steganography? The presence of steganography in the downloaded dataset has the potential to impact the experiments and any given results. To address this, a second dataset is generated by our own trusted source. This is called the Kent Digital Media Archive and is discussed in further detail in Appendix D. The videos generated from this dataset are used primarily for tests associated with detection accuracy and its purpose is to ensure that tests are not performed over data that might be unknowingly modified by a steganographic algorithm. The Kent Digital Media Archive currently hosts over 2,000 videos. However, when testing detection accuracy, only 100 stego-objects are created for each experiment. This is largely a limitation based on resource constraints as the generation of stego-objects can be computationally demanding and time-consuming. The presence of both datasets will ensure a robust unbiased methodology.

3.4 Experimental Methodology

The experimental methodology for this thesis has been designed to support the use of case studies. Each case study reflects a steganalytic investigation and analysis of a practical stego-system with focus given to the capabilities and feature of video steganography tools. The purpose of each case study is to address the research questions that were proposed in Chapter 1; specifically, to answer "How are system attacks best used within the scope of steganalysis?" and "Can system attacks be successfully applied to achieve practical results over video steganography?". There are several key steps to the experimental methodology that has been proposed.

The first step involves the identification of video steganography tools. Each tool must available to the public and demonstrate the capability to perform video steganography. Many video stego-systems are available as open-source or commercial products and it is important that the full scope of video stego-systems are identified to ensure a comprehensive and accurate representation of video steganography in the real world.

The second step involves data collections and building of the required datasets. Relevant file types are identified based on the features of each stego-system and

a comprehensive sample of files are obtained. The full collection is managed between two datasets, the first comprises a significant sample of pseudo-randomly downloaded files from the Internet (DMOZ), which are to be used for false positive testing. The second dataset uses the KDMA database to test detection accuracy of any proposed attacks. The DMOZ dataset is intended to represent a sample of files that might be observed in the real world.

The third step involves an analysis of each stego-system using the constructed datasets to examine the functionality of each stego-system that had been identified in step 1. Stego-systems are grouped according to their functionality and system steganalysis is carried out to identify vulnerabilities that can lead to detecting each steganography tool. The steganalytic investigations are performed as case studies to demonstrate the real world impact of each tool.

Any steganalytic attacks discovered from the associated case studies are evaluated on their effectiveness through an experimental testing phase. In this step, detection accuracy and false positive rates are observed using the relevant datasets (KDMA and DMOZ) to determine the strength of any presented detection methods.

Figure 24 (below) illustrates the process for the experimental methodology used in Chapters 4, 5, and 6.

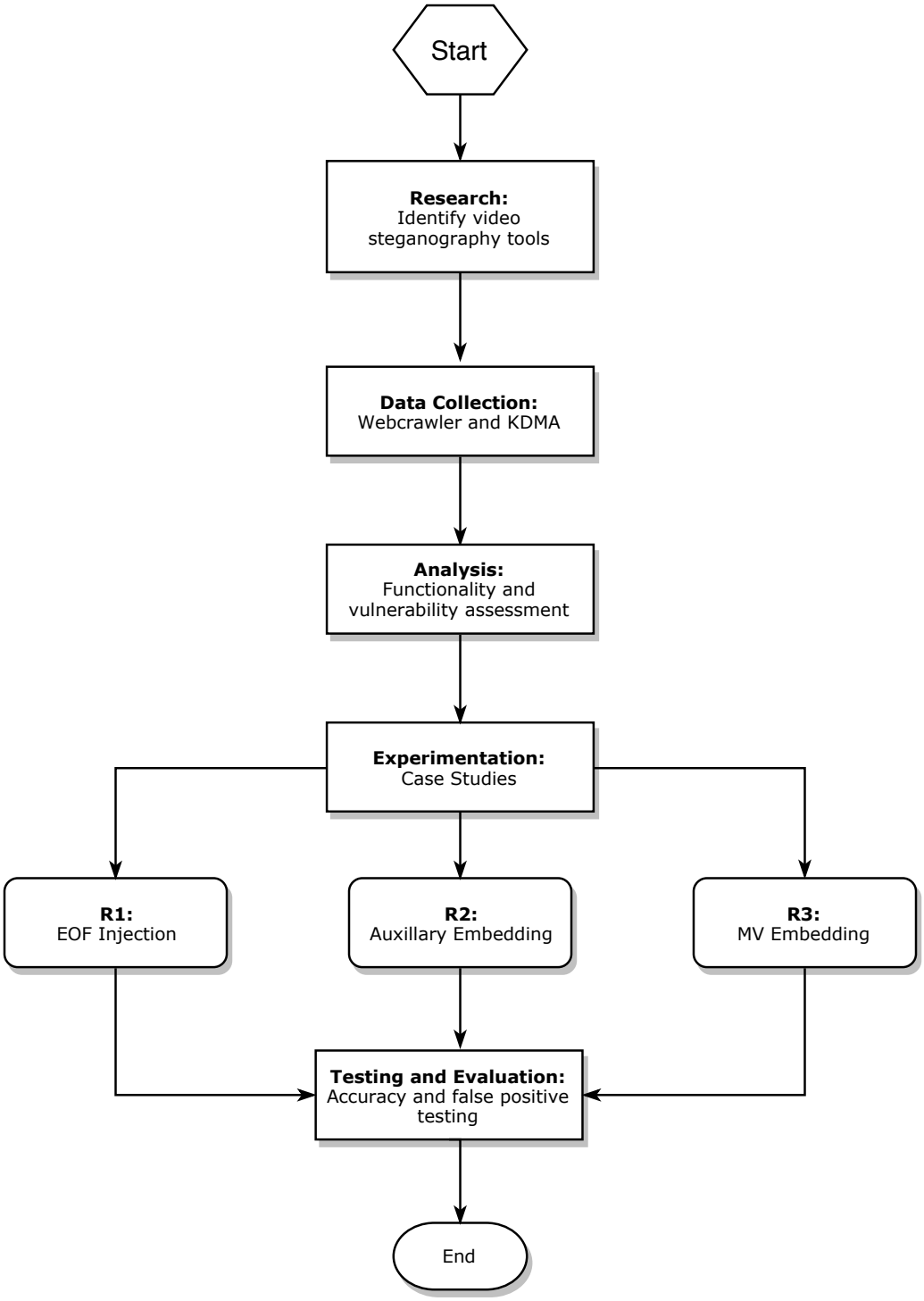

Figure 24: Flowchart illustrating the experimental methodology for this research

As shown in Figure 24, R1, R2, and R3 illustrate research steps that produced formal outputs. These three components reflect case studies that examine and evaluate the state-of-the-art in video steganography and system steganalysis. These steps intend to produce original research by presenting novel attacks against well-known steganography systems. In each case, the identified stego-systems have not yet been subjected to the efforts of steganalysis in academic literature, therefore the research conducted and presented through each case study can be considered novel.

3.5 Summary

This chapter has presented and discussed the experimental and research methodology decisions in this thesis. This research endeavour is supported by a series of case studies that evaluate the scope of application for system steganalysis and determine the capabilities of system attacks to video steganography. The methodology has been discussed to provide clarity and justification for each step of the experimental procedure followed. Two primary datasets exist to support the research, the first of which comprises a large sample of pseudo-randomly downloaded data from the Internet, and the second dataset is constructed from our own trusted source of student generated images. The use of both datasets will provide a robust methodology and ensure that selection bias is avoided while giving certainty that data used for accuracy tests are trusted.

The experimental process followed in this thesis is largely driven by an analysis of dataset generated components. Each component can be considered an artifact created for steganographic testing. This will typically consist of a pair of objects (cover-object and stego-object) for an associated stego-system. From this a sufficiently large sample should be generated to provide confidence when observing for consistency. This will form the initial foundation for the steganalysis testing to identify relevant deviations between object pairs.

Chapter 4

Attacks against Video Stego-Systems

4.1 Introduction

The rapid advancement of digital steganography has left a wide gap between academic research and its practical applications. This was observed early on by Andreas Westfeld in 2006 [157], who demonstrated that image-based stego-systems feature a range of implementation vulnerabilities based on weak cryptographic implementations and encoding functions. The increasing interest in video steganography can be related to many of the apparent advantages offered by video media. This has led to the development of video steganography tools that may encounter similar vulnerabilities to those previously seen in image steganography tools. These video stego-systems exist as free and commercial products offering a steganography solution to protect user data and facilitate secure communications. Analysis of these video steganography tools can help close the gap between current academic advancements and the existing solutions on offer. To examine practical stego-systems, system attacks are used to evaluate existing video steganography tools that have emerged to become popular and widely available.

In this chapter, an investigation is conducted into the scope and application of system attacks over video. This investigation is carried out by identifying a range of video steganography tools which are grouped and analysed according to

their embedding capabilities. The first case study examines a series of tools that perform steganography by end-of-file injection. The second case study examines a single stego-system using motion vector steganography.

Video-based stego-systems appear to adopt one of two main approaches to performing steganography. In one instance, the stego-system can adapt an embedding algorithm from state-of-the-art academic literature, however, this can present a range of challenges, especially with video-based media. Alternatively, a stego-system can implement a novel method of steganography, one that won't impact a given object in a known way. Redundancy in digital representation can be identified throughout many of the components of a media file, and this is especially prevalent in video. The presence of multiple encoding streams (video, audio, and metadata) will ensure that video media files can accommodate a wide variety of embedding methods. Because of this, practical video stego-systems have shown a particular predisposition for embedding in high-level file features. This chapter examines these video steganography tools and discusses the implementation vulnerabilities that make each stego-system susceptible to steganalysis.

The contributions of this chapter are the following:

- A series of video stego-systems are identified and subjected to steganalysis via system attacks. Each video steganography tool is publicly available as either an open-source solution or as a commercial product. It is shown that there are fundamental similarities among many of the video stego-systems available.

- The research in this chapter demonstrates coding challenges in video stego-systems that leave them susceptible to system attacks. The focus in this chapter is given to vulnerabilities surrounding weak encoding, poor cryptographic implementation, and weaknesses in pre- and post-processing over video.

- This work demonstrates that each stego-system can be broken by a series of system attacks and, in every case, it is possible to detect the presence of steganography with high accuracy.

The rest of this chapter is presented as follows. Section 4.2 provides a brief summary of existing literature that is relevant to this Chapter. Section 4.3 Presents the stego-systems that have been used as part of the two case studies in the chapter. This section continues to discuss the impact that stego-systems can impose over relevant image/video files through their invasive embedding methods. Section 4.4 presents the first case study against video steganography tools that share a common theme in their functionality. This section continues to demonstrate how each tool can be broken by the efforts of steganalysis and provides a series of experiments to evaluate the effectiveness of the proposed attacks. Section 4.5 presents the second case study against a single stego-system that uses MV-embedding. In this section, the impact of the embedding algorithm is discussed from a practical approach and a proposed attack is evaluated to demonstrate the effectiveness of system attacks over MV embedding. Finally the chapter conclusions are discussed in Section 4.6.

4.2 Background

Although system attacks are considered a known method for detecting the presence of steganography, research into these techniques is often overlooked in favour of statistical steganalysis. As a result, only a relatively small amount of literature exists on the field. A popular example of the usefulness of system attacks refers to the early steganalysis of the F5 steganographic scheme created by Andreas Westfeld [155]. The embedding scheme focused on modifying the DCT coefficients of a JPEG picture with a matrix encoding operation, to improve embedding efficiency. At the time, most researchers were focused on attacking the impact of the embedding operation using techniques typical of statistical steganalysis [50]. However, Niels Provos pointed out that the JPEG compression algorithm consistently imprinted a comment into the header of the JPEG file that could be used as a very reliable detector for the presence of F5 steganography [46]. Work carried out by Johnson et al. has also examined tools that leave digital fingerprints upon modified carrier files. In these cases, it is possible to use systems attacks to detect the presence of steganographic schemes [76, 78, 79].

In 2006 Andreas Westfeld carried out a study to confirm the idea that many

end-user tools suffer from weaknesses in the encryption and encoding components of a stego-system [157]. The research looked at a small number of image steganography tools to conclude that implementation vulnerabilities existed and steganalytic attacks could be constructed. However, this study was limited to image steganography tools and did not cover video steganography at all.

Finally, research by Ker et al. [88] has looked into the problems brought forth by a lack of practical steganalysis. Their work aims to address the concept of bringing steganography and steganalysis out of the laboratory and into the real-world. Their research is very relevant to this thesis and it explains the problems that lead to the generation of system vulnerabilities, but they note that there is only a small amount research on video steganalysis.

4.3 Stego-Systems

This chapter analyses seven video steganography tools: StegoStick, OurSecret, Masker, OmniHide Pro, BDV DataHider, Max File Encryption, and Steganosaurus. This comprises a significant proportion of the small number of video stego-systems available to the public. For each stego-system, system vulnerabilities are identified based on their steganographic strategy and it is shown how different attacks can be constructed leading to a reliable detection method. The analysis is split among two different case studies based on the functionality of each stego-system.

4.3.1 Impact of a Stego-System

This section discusses the impact of a stego-system, with a focus given to the pre- and post-processing requirements of an embedding algorithm (e.g., double compression).

The gap between theoretical and practical steganography exists largely due to the challenges of designing a secure stego-system capable of fulfilling the parameters of a practical environment. This is because the real-world is highly variable and can be difficult to model consistently. When a laboratory-based embedding algorithm is developed into a practical stego-system, a steganographic model must be considered as a coding problem in addition to a data hiding and detection

problem. This avenue of research has not yet been fully explored, especially with regards to video steganography and steganalysis.

In general, the strength of a steganography scheme is influenced heavily by the choice of cover-object. If the cover is capable of concealing the secret message, then a steganography scheme can be used with confidence. However, if the cover fails to provide enough redundancy, the scheme is doomed to fail. In digital steganography, it is frequently the cover properties that determine the effectiveness of the process. An embedding algorithm that is capable of distributing data throughout the cover should be harder to detect than one which embeds all data in a single area. However, to practically achieve this, many steganography methods have pre- and post-processing requirements for hiding a secret message. These can be key identifiers for the presence of steganography. This is in depth discussed in Section 3.5.

System attacks identify implementation vulnerabilities in the stego-system and use side-channel information to prove the existence of steganography. In many cases, stego-systems are too invasive and leave detectable traces in the stego-object that can be detected by system-attacks. By observing the impact of the stego-system, the warden can determine the best possible approach for their attack. If a signature is identified, it can be used as a component of a steganalytic attack and trace the use of steganography to detect the stego-system. Detectable features in these cases, are not exclusive to the embedding process. Through this, the features may not relate specifically to the secret message, but will often be an important part of the steganography process. This is often seen to be a product of compression taking place and introducing predictable changes to the header or footer of a file, as shown in Figure 25.

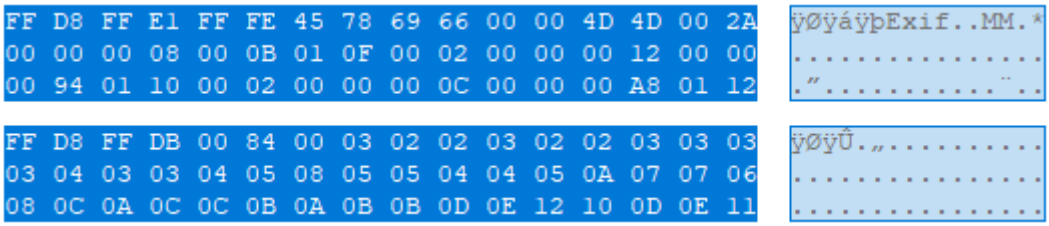

Figure 25: Pixelknot v1.01 - Cover-object header (top) and stego-object header (bottom)

In this case, the header changes are linked to the use of publicly available compression libraries. For example, the Pixelknot stego-system is based on the F5 embedding algorithm. This method works in the transform domain to substitute the LSB of target DCT coefficients. The resulting stego-object deviates subtly from the cover-object, as the compression algorithm has an impact on many of the object's features. This includes the size of the stego-object, as can be seen in Figure 26. This figure shows files sizes of files obtained from a variety of sources (random Internet downloads, KDMA archive, and the Dresden Image Database) when compared between cover-object and stego-object. The size of Pixelknot stego-objects are consistent and provide strong insights into the impact of the Pixelknot stego-system.

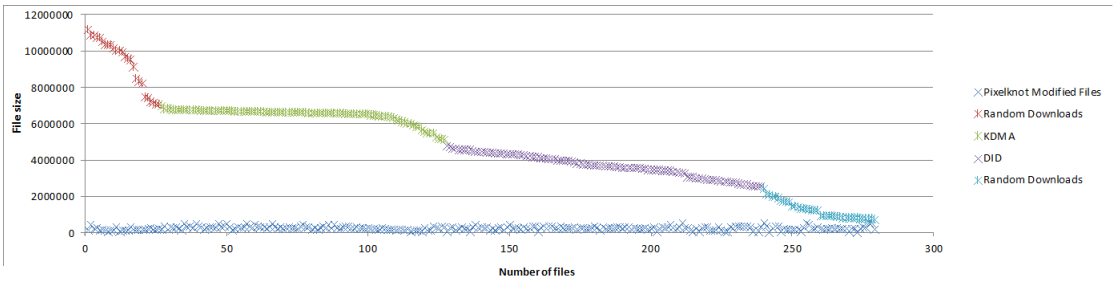

Figure 26: Pixelknot v1.01 - Changes to file size between cover-objects and stego-objects

There are many other ways in which the overly invasive nature of real-world stego-systems reduce these tools to simple cryptographic implementations. Stego-systems sometimes use a predictable string of bytes (a signature) to let the extraction algorithm know where the secret data is hidden. If identified by an attacker, it can immediately be used as part of a system attack to detect the presence of steganography for that specific tool. This is discussed further in Chapter 4.

The use of weak keys can also be a problem for stego-systems. In a typical stego-system there are two types of keys. Steganographic keys and cryptographic keys. A steganographic key controls the embedding and extraction process, while a cryptographic key encrypts the secret message before embedding. When improperly implemented, these can become obvious identifiers for the presence of steganography. In the worst case, both keys can be guessed or retrieved to provide a plaintext version of the secret message. To mitigate this, the stego-system should

ensure that keys are not susceptible to brute-force attacks [52] and that the PRNG is providing sufficiently random data.

4.4 Case Study: EOF Injection Steganography

This first case study provides an evaluation and steganalysis of a series of video steganography tools. All tools discussed in this case study are similar in their functionality and will be used to determine if system attacks can be extended to video steganography and to evaluate the current practical state-of-the-art.

4.4.1 Embedding Algorithms

An exhaustive search has revealed nine video steganography tools available to the general public. This includes the six named above and in addition, MSU StegoVideo, OpenPuff (discussed in chapter 5), and Steganosaurus (discussed in chapter 3). From this, it is observed that four distinct embedding algorithms are in use, namely EOF Injection by the six EOF tools listed above, metadata/auxilliary embedding by OpenPuff steganography, DCT by MSU StegoVideo, and motion vector by Steganosaurus. The limitations of the DCT implementation of MSU StegoVideo has been examined at length by [160, 99, 28]. This chapter focuses on the implementation of EOF injection by the six noted video steganography tools. This is certainly considered a weak form of steganography and the impact of this method is discussed alongside an identification of concrete system vulnerabilities for each stego-system.

4.4.2 Data Injection Tools

Six video steganography tools have been found to use EOF injection techniques (OurSecret, OmniHide Pro, Masker, StegoStick, BDV DataHider, and Max File Encryption). This is a method where secret message data is simply appended to the end of a given object after the final byte of data from the cover. For these six tools, this is performed over video files as shown in Figure 27. This is considered to be one of the simplest and weakest methods of steganography, which provides an

ideal example to illustrate the gap between literature and practical stego-systems. Many of these tools demonstrate poor practices in steganography, which make them highly susceptible to system and statistical attacks.

Figure 27: EOF data injection in FLV file

The choice to inject all secret data sequentially in a single file area is a very weak method of steganography. Statistically, this is easy to identify as the increased entropy is obvious. This can be illustrated fairly easily through the binary visualisation shown in Figure 28. The image on the left shows a visualisation of a cover-object with no secret data embedded. The image on the right shows a visualisation of a stego-object modified by EOF injection (within the marked area). This would happen in most stego-systems based on EOF injection.

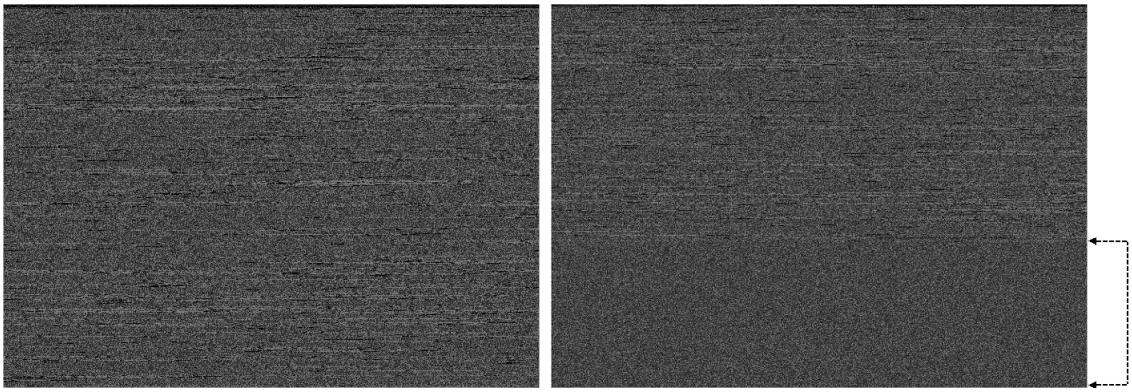

Figure 28: EOF injection shown through binary visualisation with a cover-object (left) and stego-object (right)

The high entropy of encrypted data, paired with sequentially clustered embedding, makes its presence easily identifiable. However, to a casual observer, this may not be immediately apparent. This method of steganography can be subjected to a series of system attacks as the overarching impact of EOF stego-systems can be high. As data is simply injected, no bit substitution or replacement techniques generally take place. Because of this, file size will always increase as should entropy, if strong encryption is used.

When looking at each stego-system individually, a series of unique system-vulnerabilities can be identified and isolated for each, relating to weak implementations and encodings.

To provide an overview of each stego-system, Table 4 shows the details for each tool analysed in this chapter [1]. Based on an observation of downloads across multiple sites, it can be confidently determined that many of these tools are still downloaded on a weekly basis, showing that these stego-systems are currently in use.

[1]Many of the tools listed in Table 4 do not show the number of downloads on their host site or any other centralised form. These figures have been gathered from a number of downloading sites and added. It can be safely assumed that the actual number of downloads is significantly higher that the figures shown here.

Toolkit	Price / Downloads	Resource location
OurSecret v2.5.5 (12/06/2012)	Free / 247,700+	http://www.securekit.net/oursecret.htm
OmniHide Pro v1.0 (16/08/2011)	$8 / 24,000+	http://omnihide.com/
Masker v7.5	€59 / 16,272+	http://www.softpuls.com/masker/
StegoStick Beta (16/06/2008)	Free / 9098+	http://stegostick.sourceforge.net/
Max File Encryption v2.0 (19/07/2013)	Free / 3982+	http://www.softeza.com/fileencryption/
BDV Data Hider v3.2 (01/06/2010)	$14.99 / N/A	http://www.bdvnotepad.com/products/bdv-datahider/

Table 1: Summary of EOF injection tools and downloads taken as of 15/08/2018

4.4.3 System Steganalysis

In the following, implementation vulnerabilities and weak security practices are discussed. This research aims to demonstrate examples of implementation vulnerabilities and weak encoding in video stego-systems that make system attacks viable and effective.

4.4.4 OurSecret

Formerly called "Steganography", OurSecret provides the capability for video steganography on a wide variety of video formats. This tool is currently available as version 2.5.5, and offers an option for data protection by the use of an encryption key as well as a steganographic key. As a freely available product, the OurSecret tool imposes no restrictions or limitations on use or functionality. Research carried out by Adonis in 2007 identified a weakness within previous versions of this tool (affecting v.1.7.1 and 1.8) that allowed an attacker to replace the user password with their own to extract embedded contents. However, this vulnerability as discussed was limited to JPEG images only [3].

Similar to the vulnerability identified by Adonis in 2007, it is still possible to extract the full embedded contents from a video file modified by OurSecret. In the absence of a password, the embedding algorithm will use a fixed 16-byte string. As this 16-byte hexadecimal string is used in place of a user-provided password, an attacker can simply copy this value into any OurSecret encoded files using a hex editor and bypass the user-provided password with this null-password string. This simple modification will deceive the tool into recognising a carrier file as having no

password requirements. As a result, the attacker is then able to access and extract the full hidden contents. This can be considered a Null-Password Vulnerability.[2]

Figure 29 illustrates the Null-Password exploit in practice. These are the final 26 bytes of a file modified by OurSecret. The highlighted bytes are the Null-Password replacing the original 16 bytes of a file that were injected by OurSecret. This simple change grants the attacker full access to the hidden contents with no need for the secret key.

```
00 00 10 1A 00 00 6C 3C 39 6C 30 6B 6C 31 30 6E
38 38 6A 3A 38 3C 00 9B 12 00
```

Figure 29: Null-password replacing any previous user password

Weak stego-systems often implement their own signatures as identifiers for the presence of steganography. This won't be apparent to a casual observer, but can be easily identified by an attacker looking to break the stego-system. OurSecret provides a consistent 40-byte string, as shown in Table 5. This is a sequence of hexadecimal characters that occurs within all OurSecret modified video files, across all formats and encoding options. This can be classified as a valid signature to identify the existence of any contents hidden with OurSecret. This signature appears directly after the final byte of data. The presence of signatures provide multiple practical advantages for steganalysis. If their length is sufficiently long, detection accuracy will be high enough to mitigate any concern for false positives. In addition to this, if the string is completely unique to the stego-system by being part of the encoding process as opposed to being a common feature (compression/double compression), then the signature should be suitable for tracing the presence of steganography back to the relevant stego-system. For the purposes of a digital forensic investigation, this would be very valuable information.

[2]This vulnerability is an improvement to the JPEG exploit as referenced on CVE-2007-0163, where this weakness was not mentioned.

9E	97	BA	2A	00	80	88	C9	A3	70	97	5B	A2	E4	99	B8	C1	78	72	0F
88	DD	DC	34	2B	4E	7D	31	7F	B5	E8	70	39	A8	B8	42	75	68	71	91

Table 2: OurSecret signature

The OurSecret stego-system is a highly suitable example to illustrate the gap between steganography in the laboratory and steganography in real-world scenarios. It is completely oblivious to many security recommendations, as evidenced by the use of weak cryptography and encoding. Because of this, a system attack based on the identified signature can be constructed to detect the presence of OurSecret steganography.

4.4.5 OmniHide Pro

OmniHide Pro is a commercial product capable of embedding data within image, audio, video and other media. Released in 2010 for the Windows operating system, OmniHide offers a Trial and a Pro version, the former providing only limited functionality. OmniHide Pro costs $8, and will provide the user with all the tools' features. In the following, only the the Pro version is subjected to system steganalysis.

Figure 30 shows a typical stego-object through an hexadecimal editor. It reveals the same embedding method of OurSecret, also employing EOF data injection. Reviewing the offset where the unmodified file would have ended, it is apparent that the OmniHide Pro tool also suffers from several vulnerabilities that can compromise the security of a users' data. OmniHide embeds information within videos by appending the data directly to the end of the file. The first sequence of bytes are followed by a string of white space characters. These are reserved bytes to be replaced by longer file names that are embedded into the carrier.

```
EC 30 E1 8C 5F EC BF EC  C3 AE 7C D8 9B FF FF FD  FA A6 B8 7A 05 D5 7F 55  6F 60 8A FA F3 B8 2A 53
54 31 30 6D 62 2E 74 78  74 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
```

Figure 30: First 11 bytes of embedded content

Figure 31 shows that it is possible to identify the name of the text document embedded within the video. For the purposes of these tests, a single file was embedded called "ATest.txt". This converted ASCII is the first string of identifiable information. Using a hex viewer, this data follows the offset in which the unmodified file would have ended. The presence of a file name that is so easily identifiable within a carrier file raises many security concerns. For example, if an attacker has access to the users' computer, it may be possible to identify the initial file still present in a plain text format. It is best practice for steganography tools to fully encrypt a file before embedding. This provides an additional layer of security and will make it more challenging for any attackers to identify the existence and location of hidden contents, to then recover them. OmniHide Pro clearly lacks this fundamental feature.

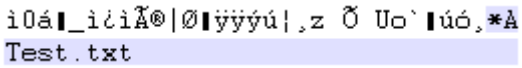

Figure 31: Hex string of embedded data, converted to ASCII

Further examination of OmniHide Pro revealed other characteristics of the embedding algorithm that can be considered as security weaknesses from a steganalytic perspective. As the file size of the carrier increases proportionally to the embedded content, it is apparent that no compression is taking place during the embedding process. This is again against recommended best practice, as file compression should be used to reduce the size and redundancy of embedded contents. Figure 32 captures the end of file characteristics of data injected by OmniHide Pro (approximately 19MB in this case). This reflects the size of the carrier file prior to data injection. A subtraction between these two values will allow any attacker to quickly and accurately calculate the size of the injected hidden contents.

Figure 32: File size of the unmodified carrier

This inclusion of file size is an important characteristic, as extraction functions of a EOF stego-system use this information to determine where to begin looking

for hidden data. However, other stego-systems may hide this information so that it is not as easily detectable as the case for OmniHide Pro.

4.4.6 Masker

Masker is a commercial application that claims to offer secure video steganography. The latest stable format, version 7.5, was released in 2009 and can be accessed as either an evaluation version for a limited period of time, or alternatively, can be purchased for €59. Masker provides the option of several different encryption algorithms to secure user data, this includes: Blowfish, CASTS, DES, Serpent-256, AES-256, TripleDES and TwoFish. Masker does not work over MP4 files, but is capable of processing the following file formats: AVI, MOV, MPG, MPEG, ASF, MPA and MPE. Upon examining the functionality of this tool, there are several concerning weaknesses with the stego-system. It is apparent that Masker shares many vulnerabilities with the previously discussed tools. Analysing an AVI file with a small amount of embedded content, it is apparent that Masker also employs EOF data injection.

Figure 33 illustrates the EOF injection technique adopted by Masker. The first 10 bytes in the image are the final part of the source file. The highlighted bytes reflect the start of Masker's embedding. Using a hex viewer, these EOF injection algorithms are trivial to detect and exploit.

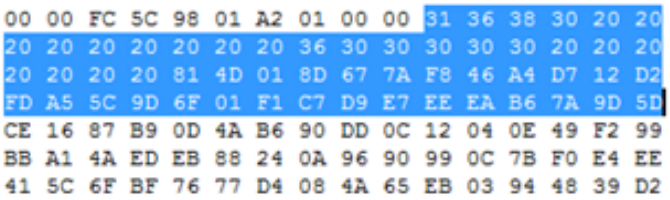

Figure 33: Masker embedding

Similarly to OurSecret, it is possible to identify a distinctive sequence of bytes that can uniquely characterise the embedding algorithm. This is consistent across the varying embedding options, irrespective of the encryption algorithm used or the choice of secret key. This string can be considered to be a signature for the Masker steganography tool and, due to its unusual length, it will ensure a high

detection rate.

The signature identified and shown in Table 6 shows a 73-byte string that is consistently embedded in stego-objects modified by Masker steganography. The consistency of injected data leaves this stego-system susceptible to the construction of simple steganalytic schemes. These attacks break the tool from a steganalytic perspective. Similar to a vulnerability within OmniHide Pro, it is possible to identify bytes that reflect the size of embedded contents.

9E 8D DB 50 73 C2 BF 65 B8 1E 03 AE D5 62 C6 CC 71 9A B9 B9 48 49 D2 64 EE
49 B7 63 7B EF EB A1 01 04 40 7B F0 ED 2D 86 47 E1 5F FC 7C 41 C9 8B C9 02
CA 1C 07 21 EE 8D 32 66 47 7C 8F E1 4E E8 AE 66 AB 32 E3 D2 F9 A1 0E

Table 3: 73-byte Masker signature

Having identified the existence of a signature, it is possible to create a simple script to detect the presence of steganography. An automated script would be easily capable of detecting a signature with a length of 73 bytes to a high degree of accuracy. From an academic perspective, this type of vulnerability completely breaks the tool's capability for secure steganography. In Section 3.3, further discussion is given in the approach towards signature steganalysis based on the strings that have been identified.

4.4.7 Max File Encryption

Max File Encryption is a cryptographic tool with video steganography capabilities. The current version is 2.0, released on 07/2013. Analysis of Max File Encryption using the aforementioned test framework has provided similar discouraging results and illustrates many of the same mistakes. By employing EOF injection, it displays a concerning number of weaknesses. Upon inspection on a hex editor, it is apparent that the carrier file contains key components of the program, such as the libraries and public key tokens used, and other artifacts. Not only this can lead to a large number of signatures, it is also possible to identify key parts of the program source code through this.

Figure 34 identifies an embedding characteristic identical to that of OmniHide

Pro. This tool injects information illustrating the file size of the unmodified carrier. This provides an attacker with the opportunity to estimate the size of embedded content. Provided an attacker has access to the source computer, it may be possible to identify the hidden file, generally the only one matching that exact size.

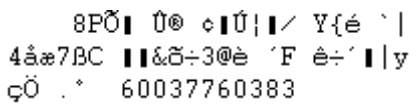

Figure 34: File size of embedded contents

Max File Encryption appears to be another generic EOF injection tool containing a number of already familiar vulnerabilities. The presence of these vulnerabilities negate any security features offered by the tool.

4.4.8 StegoStick

StegoStick is a free open-source steganography tool that claims to have the capability to hide any file type into any other file type. This concept, although appealing, becomes significantly less tempting upon the realisation that it employs EOF data injection for video. The current release is a beta version from 08/2013. At present, the examination of the tool has not revealed any form of signature that can be used. The only recoverable information that could be extracted from the stego-system identifies the file types embedded into the stego-object.

Figure 35 shows an ASCII representation of hex data captured from a StegoStick carrier file. Varying file types were embedded in video files such as PDF and TXT. These were identifiable during analysis and had been present in every test. The ability to extract embedded file types can provide small insights such as what file types to look for on a suspect computer. Although this tool uses EOF injection, which is again a very poor form of steganography, StegoStick does apparently not have the same number of security issues found in the previously analysed tools.

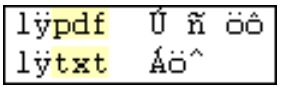

Figure 35: StegoStick embeds the file type of hidden data

Even though signature steganalysis may not be possible in this case, system attacks are still possible over StegoStick. This is shown in Section 4.4.

4.4.9 BDV DataHider

BDV DataHider is a commercial steganography application developed by Bedavlad Software. The most recent release is 3.2, available from 06/2010. This tool can be purchased for $14.99. Similar to the other listed video steganography tools, BDV DataHider employs EOF injection techniques. Although the developers express high confidence in the quality of the tool and state "There is nothing more powerful on the market when you want to transfer unnoticeably (sic) some files over Internet or on a flash drive" [142]. However, it is apparent that the embedding algorithm features very similar weaknesses to those previously exposed. A major identifiable vulnerability is provided by the tools' interface.

Under the model of system steganalysis, an attacker can obtain valuable side-channel information about the BDV DataHider stego-system. The tool has a catch that will inform the user if data can be embedded or has been embedded within a video file. This is shown in Figure 36.

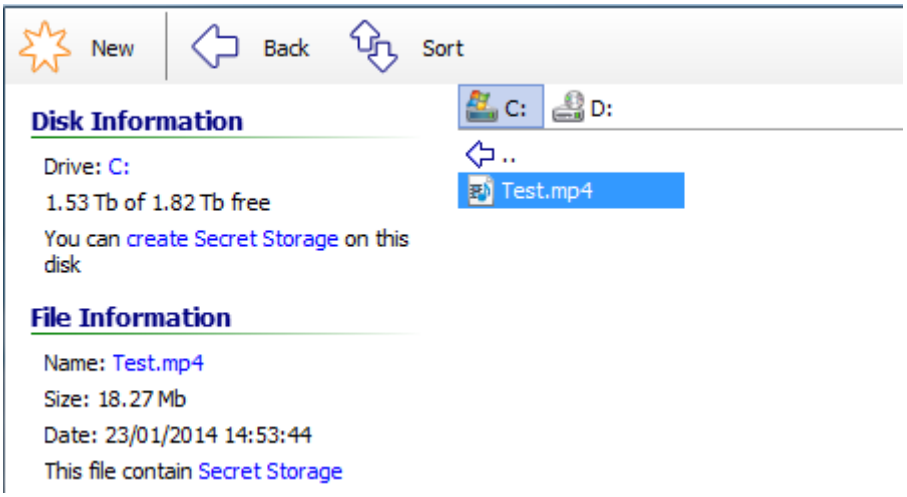

Figure 36: Data Hider provides its own detection system.

This information illustrates that a signature exists within stego-objects that the stego-system is looking for to determine whether or not embedding and or

extraction of a message is possible. This information is represented through the following sequence of bytes (as shown in Table 7) inside a stego-object modified by BDV DataHider.

FB EA D8 81 25 0F F9

Table 4: Signature for BDV DataHider

To determine that these bytes are related to the tool's detection function, they can simply be removed from a stego-object that contains BDV steganography. The stego-system will once again inform the user that "the file can be used to create secret storage".

4.4.10 Constructing a System-Attack

The practical applications for system attacks are wide-ranging, providing efficient methods to detect the presence of steganography. Analysis of each tool in this chapter has shown that system attacks have practical applications to video steganography. However, it is important to consider if these methods can achieve useful results as a method for real-world steganalysis.

For each stego-system discussed in this chapter, it is possible to obtain side-channel information that can be constructed into a system attack. These stego-systems demonstrate the use of not only a weak method of steganography, but also weak encoding. The method of steganography used in each case is simple to detect, but the presence of a weak system implementation provides an attacker with multiple avenues of steganalysis (detection of EOF, detection of secret key, detection of secret message). The traditional model for steganography must be extended to consider the method as a coding problem and therefore, a whole avenue of security parameters and practices should be included.

System steganalysis has identified and proven the functionality of each stego-system. Furthermore, the analysis has exposed a number of implementation, encoding and key related vulnerabilities across each stego-system. For each case presented, an attacker can construct efficient and highly accurate attacks to detect discovered signatures within stego-objects. Proof-of-concept scripts for each

attack is shown in Appendix A.3 - A.5.

4.4.11 Testing

A testing framework for system steganalysis was presented in Chapter 3. For each stego-system discussed in this chapter, system attacks were constructed and evaluated through a proof-of-concept detection attack. For each video steganography tool, 100 stego-objects were produced from the KDMA dataset to test the accuracy of signature-based system-attacks. The results of these tests are shown in Table 8. False positive tests for each stego-system were carried out over a significantly larger sample size of 5,000 randomly downloaded files.

Table 5: Accuracy and false positive tests for signature steganalysis

Steganography Scheme	Dataset Size	Detection Accuracy	FP	Avg Runtime
OurSecret	100	100%	0%	1.95s
OmniHide Pro	100	100%	0%	2.35s
Masker	100	100%	0%	3.21s
BDV DataHider	100	100%	0%	3.6s
MaxFileEncryption	100	100%	0%	1.76s

The results for signature steganalysis show high accuracy paired with a zero false positive rate. In most cases, these promising results are based on the significant length associated to each signature. From a practical standpoint, there is low risk of identifying these signatures within natural digital media making them ideal for the purposes of steganalysis against many of the existing video stego-systems.

4.4.12 Generalising EOF Injection Attacks

There are two main advantages when using signature detection schemes for steganalysis. Firstly, the method is highly accurate; a sufficiently long and entropic sequence of bytes will ensure that the signature is unique and that it is highly unlikely to appear in natural digital media. This also leads to the second advantage. Signature detection methods constructed from a system attack will trace the

use of steganography to a specific stego-system via targeted steganalysis. As evident from the findings in this chapter, there are many EOF-based stego-systems and system tracing provides an additional practical advantage in a steganalytic investigation.

One of the challenges surrounding signature steganalysis of video relates to processing requirements for the system attack. Video files are large and the method presented to detect signatures would not be ideal in a practical environment (over potentially millions of files). Therefore, faster methods can be constructed as a first stage attack against EOF systems. A generalised attack against EOF injection would reduce the sample size of suspicious files significantly. Therefore a generalised attack can be constructed to reduce the initial sample size and provide a broad spectrum for detecting all EOF injection stego-systems. This method is constructed to detect the presence of arbitrary atoms, as shown in Figure 37. In this case, the final atom of the MP4 tree represented encrypted, high entropy data that MP4 parsers cannot read.

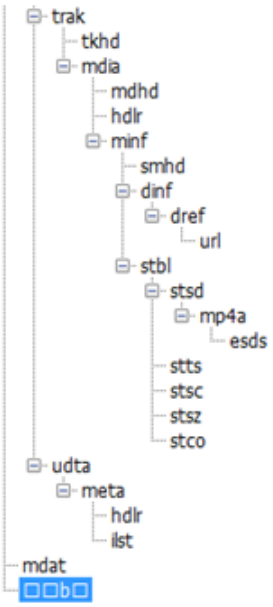

Figure 37: Arbitrary atom shows use of EOF injection

The presence of this arbitrary atom is identified for every EOF-based stego-system. As such, a proof-of-concept (shown in Appendix A.6) has been constructed

to detect EOF steganography across all relevant stego-systems with the exception of Masker [3]. There are two main advantages to using this attack. Firstly, this provides an ideal method to develop an underlying suspicion of steganography. If detected, then more accurate methods (such as signature attacks) can be used. Secondly, this is significantly faster to perform when compared with signature attacks. This provides an ideal opportunity to reduce a sample size of potential stego-objects.

Tests were performed to determine detection accuracy using 100 stego-objects from the KDMA dataset. False positive rate of the generalised attack was tested over the DMOZ dataset of 5000 clean files for each stego-system. The results are shown in Table 6.

Steganography Scheme	Data Size (Steg/Clean)	Detection Accuracy	False Positive	Avg Runtime
OurSecret	100/5000	100%	0%	0.01s
OmniHide Pro	100/5000	100%	0%	0.01s
BDV DataHider	100/5000	100%	0%	0.01s
MaxFileEncryption	100/5000	100%	0%	0.01s

Table 6: Tests performed to analyse performance of generalised attack

As a first stage steganalytic attack, the generalised method provides high accuracy and a zero false positive rate. This limitation of this method is that it cannot provide system tracing at which point signature-based system attacks can be implemented.

4.5 Case Study: Motion Vector Steganography

This second case study presents a steganalysis and evaluation of the video steganography tool "Steganosaurus". The goal of this case study is to contribute to the comprehensive evaluation of the state-of-the-art in practical video steganography and determine if system attacks can be successfully applied. Relevant background to this work is first discussed.

[3]Masker performs EOF injection over AVI files, to detect this a different method would need to be constructed

Steganosaurus [126], is a video steganography tool created by James Ridgway and released in 2013. The stego-system uses motion vector embedding in the X or Y coordinates of the first macroblock of a frame. This is described as temporal domain steganography in which an embedding algorithm is using the temporal redundancy of video compression to modify target motion vectors. The large number of motion vectors available per-frame, alongside the relatively small amount of changes introduced will ensure that a stego-object is visually indistinguishable from the cover-object. The Steganosaurus stego-system is shown in Figure 38.

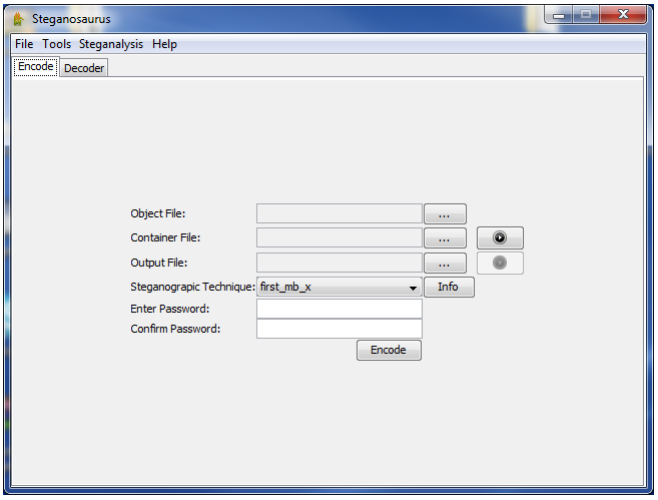

Figure 38: Interface of Steganosaurus v1.0

4.5.1 Impact of Pre- and Post-Processing requirements

Any embedding operation for motion vector steganography should avoid embedding secret message bits directly within a compressed file, because the resulting changes would introduce noticeable distortions and corrupt the target motion vectors. To resolve this, a transcoding step is introduced during embedding. In this step, a file is compressed again and new motion vectors are created to embed secret message bits. Visually, the transcoding solution that is the stego-object, should be indistinguishable from the cover-object. However, compression will likely introduce changes that are ideal for steganalysis. This happens mainly because focus has only been directed to maintaining imperceptibility with the use of a transcoder.

The transcoder for Steganosaurus was built using functions of the “avcodec”

and "avformat" libraries of FFMPEG. This rewrites the structure of an MP4 file as shown in Figure 39. One key observable feature resulting from the structural modification is the re-positioning or insertion of the "free" atom. An atom can be considered a self-contained data component of the MP4 file structure. In cases where atoms (such as MDAT) are added, modified or removed, the tree is also changed. As a result, lengths of atoms may need to be recalculated. In this case, with the "MDAT" atom, further adjustments are made and this is supported by the "free" atom which minimises the need for unnecessary adjustments [143]. An arbitrary shift in the stego-object can be a clear indicator at a high-level of the presence of steganography.

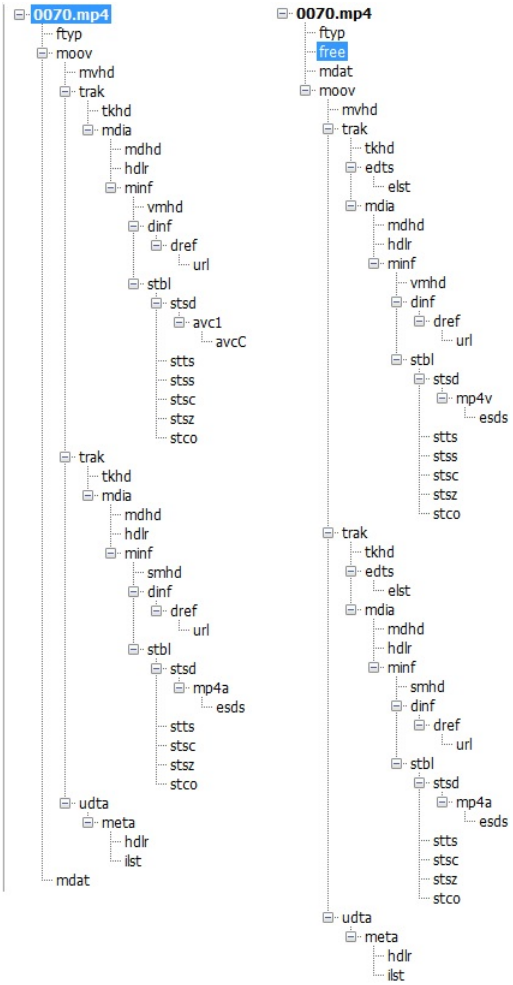

Figure 39: Unmodified file MP4 (left) and MP4 carrier produced by Steganosaurus (right)

A steganalytic scheme can be constructed to detect arbitrary ordering of the MP4 file structure and, in particular, the modifications caused by the transcoding component of the Steganosaurus stego-system. This is shown below and a sample script can be seen in Appendix. A.1. The purpose of this steganalytic scheme is to identify a particular file structure that is almost never encountered in the typical MP4 file format. Although this can be used to develop an underlying suspicion of the use of Steganosaurus steganography, it should be used alongside other steganalytic techniques to reduce any potential for false positive occurring from natural use of "avcodec" and "avformat" libraries.

```
for file in /TestSets/*.mp4
do
mp4file --dump ‘‘${file}’’> /tmp/mp4dumpfile
sed ’6,7!d’ /tmp/mp4dumpfile >/tmp/tmp1

for line in $(< /tmp/tmp1); do
echo $line | sed ’s/.*(\(.*\))/\1/’>>/tmp/tmp2
done


if egrep -q "free|mdat" /tmp/tmp2; then
echo -e \ "${file}" "Atom Test - Arbitrary file strucuture found"
else
echo -e \ "${file}" "Atom Test - No Steganosaurus Steganography found"
fi
```

Listing 4.1: Sample script to detect changes to the MP4 structure by Steganosaurus

The structural deviation between cover-object and stego-object is only one of the changes made during the transcoding process by the Steganosaurus tool. The use of the function *"avformat_write_header"* configures a file header for the stego-object. The header describes the stream configuration for a video file and metadata details regarding format and codecs. This description appears as a consistent sequence of bytes as shown in Figure 40. For a stego-object with a

header produced by Steganosaurus, the header format will be consistent with the *"avformat_write_header"* function and produce predictable byte sequences.

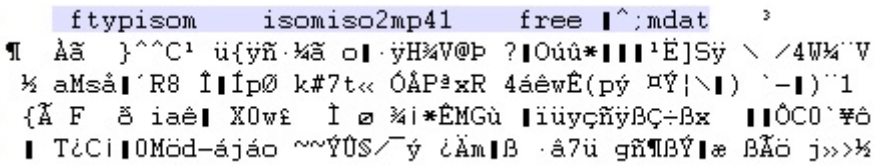

Figure 40: ASCII representation of header signature with codec use

Known as a magic prefix, this is a consistent sequence of bytes that appear as a result of the transcoder creating a unique header for stego-objects. In this case, this digital imprint appears as an ISO codec fingerprint in the header and can be used as another predictable feature for system steganalysis. The transcoding process provides a unique signature that can be implemented into a simple script to markedly simplify the process of steganalysis. The sequence of bytes produced are shown in Table 1. [4].

1C 66 74 79 70 69 73 6F 6D 00 00 02 00 69 73 6F 06
D6 97 36 F3 26 D7 03 43 10 00 00 00 86 67 26 56 05

Table 7: Header fingerprint left by Steganosaurus

Used within a script, this signature creates a system attack that provides a high level of accuracy due to its significant length. The sample code (as shown below) uses signature steganalysis to identify the presence of Steganosaurus steganography. The full script can be seen in Appendix. A1 & A.2.

[4]ISO details available at http://www.ftyps.com/

```
for file in /TestSets/*.mp4
do
xxd -p "${file}" > /tmp/tmp1
tail -c 40 < /tmp/tmp1 | tr -d '\n' > /tmp/tmp2
if grep -q -c
    "1C6674797069736F6D0000020069736F06D69736F326D703431000000008667265605"
    /tmp/tmp2; then
echo -e \ "${file}" "Signature Test - Steganosaurus content detected"
else
echo -e \ "${file}" "Signature Test - No Steganosaurus Steganography
    found"
fi
```

Listing 4.2: Sample script to detect signatures introduced by Steganosaurus

Additional steps can be taken to reduce the potential of false positives occurring from the use of the "avformat_write_header" function. The signature detection scheme can be concatenated with other steganalytic features. For instance, a concatenation of atom detection alongside the signature script would significantly reduce the likelihood of false positives while increasing detection accuracy. The sequence is shown in Figure 41. In this, atom analysis is carried out as a first stage featur, that will filter out a large selection of candidate files and reduce the total runtime for the script. The second stage of the script is run once the presence of steganography is suspected. A signature detection script will run to detect consistent byte patterns typical with the use of the *"avcodec"* function. Runtime of this process is often slower than atom detection, so it advisable to run it in the second stage after filtering has been carried out and a small sample of potential stego-files is available.

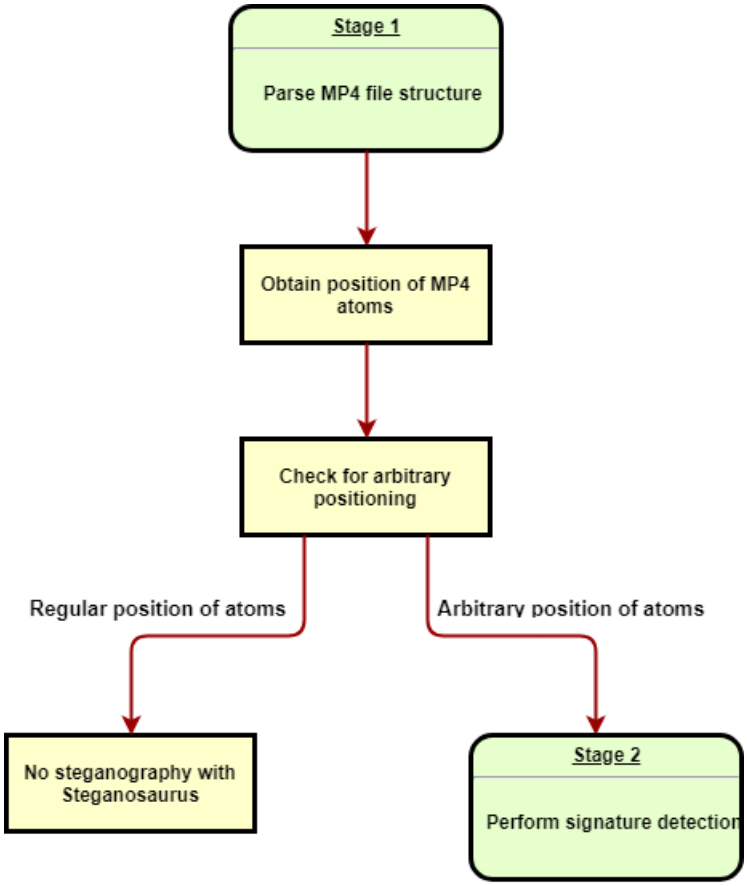

Figure 41: Illustration of the proposed sequence for concatenated script

4.5.2 Testing

The steganalytic schemes constructed to detect Steganosaurus steganography were tested and evaluated following the test framework described in Section 3.4. Each scheme is tested independently and then concatenated to provide a juxtaposition. Tests are carried out in two stages. Firstly, accuracy tests are run over 100 MP4 files from the KDMA dataset, of which the full 100 are modified. The intention of this test is to determine if the stego-objects can be correctly identified and compute a rate at which each scheme can accurately perform positive detection. The second test is performed to determine a false positive rate. This is conducted over a significantly large number of files (5,000) that are clean all of steganography.

These files were randomly downloaded from a URL list that was generated pseudo-randomly from the DMOZ directory [9]. The results are shown in Table 8.

Table 8: Tests performed for detection accuracy and false positive rate

Steganalysis Scheme	Detection Accuracy	False Positive Rate	Average runtime
Atom Detection	100%	10.9%	0.06s
Signature Detection	100%	0%	3.28s
Script Concatenation	100%	0%	0.60s

The tests performed for the steganalytic scheme for Steganosaurus demonstrate that system attacks provide an ideal alternative route for steganalysis. High level features extracted from a stego-object can be linked to the use of steganography and trace that use to the related stego-system. In addition to this, tests were carried out for detection of this steganographic scheme amongst other well-known stego-systems. This is shown in Table 9.

Table 9: Tests performed to demonstrate tool tracing

Steganography Scheme	Dataset Size	Positive Detection	False Positive
Steganosaurus	100	100%	0%
OpenPuff	100	0%	0%
OurSecret	100	0%	0%
OmniHide Pro	100	0%	0%
BDV DataHider	100	0%	0%
StegoStick	100	0%	0%

Datasets were created for known video steganography tools that use the MP4 file format, including: OpenPuff, OurSecret, OmniHide Pro, BDV DataHider, and StegoStick. The steganalytic scheme provided 100% accuracy and had shown to detect Steganosaurus steganography exclusively. This ensures that the use of steganography can be directly and accurately traced to its corresponding stego-system. The high accuracy and low false positive rate of these tests demonstrate the effectiveness of system attacks over MV-embedding, ignoring the direct impact of the embedding algorithm and instead opting to attack pre- and post-processing requirements of the embedding operation.

4.6 Conclusion

This chapter has presented a comprehensive study of the practical state-of-the-art in video steganography. In total, seven video steganography tools were identified and broken by a series of system attacks. The analysis was split into two separate case studies based on the functionality of the tools and experiments were conducted to evaluate the effectiveness of each attack. Results of the analysis had shown that many of the publicly available video steganography tools use a weak form of steganography. In other cases where more effective embedding methods are used, the steganographic security of the stego-object can still be compromised. A summary of findings is shown in Table 10.

Toolkit	**Embedding Algorithm**	**Vulnerabilities Identified**
OurSecret	Data injection - EOF	40-byte signature, full retrievable contents with null password
OmniHide Pro	Data injection - EOF	Recover hidden filename and size, found pseudo-signature
Masker	Data injection - EOF	73-byte signature, hidden file size can be determined
StegoStick	Data injection - EOF	Hidden file type recoverable
Max File Encryption	Data injection - EOF	Multiple signatures
BDV Data Hider	Data injection - EOF	14-byte signature
Steganosaurus	Motion Vector Embedding	atom and signature detection

Table 10: Summary of the vulnerabilities found

This chapter has shown that video steganography in its practical state has concerning technical and methodological issues. In many cases, the embedding algorithms use a weak form of steganography that is simple to detect and the resulting process of steganalysis can be straightforward. In addition, it is clear that video steganography is largely considered a data hiding and detection problem; however, there appears to be a lack of awareness support video steganography as a coding problem. As a consequence to this, developers of video steganography tools have failed to consider the invasive impact that embedding algorithms can impose in the real world.

The following points can be considered as contributions of this chapter:

- Contribution 1: Evaluated the state-of-the-art in practical video steganography.

- Contribution 2: The scope of system attacks have been extended from image steganography to video steganography to demonstrate that the applications for system steganalysis are wide-reaching and effective
- Contribution 3: Work in this chapter has presented a series of novel system-attacks against popular methods for practical video steganography.

This research has identified an exhaustive list of video steganography tools that are available to the public. An analysis of each tool has exposed the gap between academic literature and the real world. From this, it is clear that many of the tools share similar features and functionality which as a consequence, increases the risk associated to their use. System attacks have been previously explored in academic literature over image steganography and until now had not been discussed with respect to video steganography. This contribution provides support to the advantages of system attacks to show that their scope of application goes beyond image and perhaps into other unexplored avenues of steganographic research. The high rate of detection accuracy shown in the experimental results provides a foundation of evidence to the practical benefits of system attacks. However, continued research into system steganalysis over stronger steganographic systems will help understand the scope and limitations of this research. In the next chapter (Chapter 5), a state-of-the-art stego-system is identified and examined to further test the scope of system attacks.

Chapter 5

Steganalysis of the OpenPuff Stego-System

5.1 Introduction

This chapter presents two attacks against the OpenPuff stego-system. The first is an attack targeting its MP4 embedding algorithm for video steganography. The second attack focuses on the OpenPuff embedding algorithm for PDF steganography. OpenPuff is an interesting stego-system from a steganalytic perspective, as it has achieved praise from many sources [113] and boasts a diverse range of embedding capabilities, a feature that is unusual. Typically, a stego-system should provide a single embedding method and focus attention on a single scheme to ensure it satisfies all security requirements. In other cases where a stego-system provides embedding options across multiple file types, typically a simple method of steganography is used for each format, this is a bad practice as it generally simplifies the process of steganalysis. This is a feature that is often seen with EOF injection steganography tools. However, OpenPuff offers unique embedding approaches across multiple file formats. In addition, a few of these methods appear unique to OpenPuff, and have not been identified elsewhere. This effort by OpenPuff demonstrates an attempt to bridge the gap between academic research and the practical applications of steganography and has probably contributed to its success.

In previous chapters of this thesis, system attacks have been applied to video stego-systems, the results so far have shown that video-based stego-systems feature a variety of implementation vulnerabilities, many of these can be used as possible avenues of steganalysis. In addition, where a steganalytic attack can be constructed, it is often seen to give high accuracy with a low rate of false positives. In this chapter, the scope of system-steganalysis is extended to the OpenPuff stego-system. This tool is considered by its author to be highly suitable for the purposes of steganography.

The rest of this chapter is organised as follows. Section 5.2 provides an overview of the OpenPuff stego-system. This section discusses the features of the tool and the cryptographic implementation it employs. Section 5.3 is split into two main subsections. 5.3.1 presents the first system attack against the MP4 features of OpenPuff video steganography. This subsection discusses the impact of the embedding algorithm over stego-objects and identifies traceable features that can be used for the purposes of steganalysis. This is built into an attack as a proof-of-concept, and evaluated. Subsection 5.3.2 presents the second attack against the PDF features of OpenPuff document-based steganography. This subsection similarly discusses the impact of the embedding algorithm over stego-objects and traceable features are used in a proof-of-concept attack to evaluate the proposed method. Finally, section 5.4 provides a summary of the contributions and the results obtained.

5.2 OpenPuff

OpenPuff is a tool capable of performing steganography over a large number of different file formats, concretely (as listed on the tool's website) sixteen, covering media such as images (bmp, jpx, pcx, png, and tga formats), audio (aiff, mp3, NEXT/SUN, wav), video (3gp, mp4 mpg, wob) and Flash/Adobe formats (flv, swf, and pdf).

OpenPuff (currently in version 4.0) is marketed as "Yet **not** another Steganography Software" and is a free semi-open-source tool, which uses the libObfuscate v.2 library, developed by the same author, to perform its cryptography related tasks. The libObfuscate library implements a number of encryption algorithms

(including AES, Anubis, Camellia, CAST-256, Clefia, Frog, Hierocrypt3, Idea-NXT, MARS, RC6, Safer+, SC200, Serpent, Speed, Twofish, and Unicorn-A). This large selection of sixteen block ciphers is employed to realise the concept of multi-cryptography, first introduced by its author, where the cipher used to encrypt data is picked pseudo-randomly. It is important to note that, as shown in the architectural design in Figure 42, these block ciphers are used in the insecure ECB mode, strongly against common practice. It may well be pertinent to add that most of these block ciphers are considered to be very weak or, in some cases, totally broken. LibObfuscate also implements four hash functions, all producing a 512-bit digest. These are Grostl, Keccak (the new SHA-3 standard), SHA-2 and Skein. OpenPuff also offers a cryptographically secure pseudo-random number generator (CSPRNG) based on AES but not following any known PRNG standard. It is perhaps relevant to stress that, although libObfuscate's source code is available, OpenPuff is in itself closed source, so exact details regarding the functionality of the steganography algorithms are unknown. This is also against best practice and is considered a security problem.

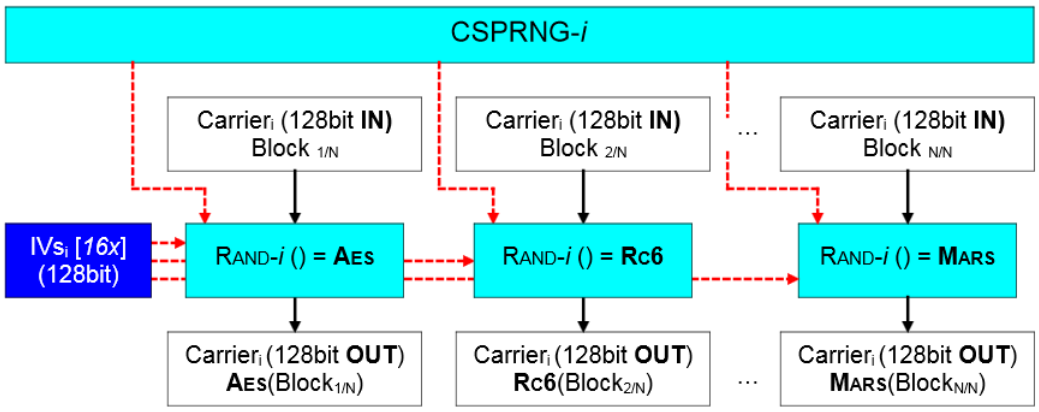

Figure 42: OpenPuff cryptographic architecture, from [113]

Although details are not available on the tool's website, third party sites reveal large download numbers (these figures are as of 27/09/2017), as shown in Table 11. It is reasonable to expect the total numbers of downloads to be significantly higher than those found and shown below.

Website	Downloads in Last Year	Total Downloads
http://download.cnet.com/OpenPuff/3000-2092_4-75450743.html	2038	11,405
http://www.pcworld.com/product/1252470/openpuff.html	N/A	1420
http://www.softpedia.com/get/Authoring-tools/Authoring-Related/Puff.shtml	808	59,611
http://www.downloadcrew.co.uk/article/27852-openpuff	462	2,725
http://www.freewarefiles.com/OpenPuff_program_58719.html	694	6,334
http://www.majorgeeks.com/files/details/openpuff.html	1,229	10,827
http://www.winpenpack.com/en/download.php?view.913	311	3,471
http://www.baixaki.com.br/download/openpuff.htm	494	3,571
http://www.brothersoft.com/openpuff-34008.html	13	38,960
http://openpuff.en.uptodown.com/	686	2,200
Total	**6,735**	**140,224**

Table 11: Approximate download figures for OpenPuff as of 15/08/2018

OpenPuff limits the amount of data that can be embedded within a cover-object. This ensures that the resulting stego-object does not raise suspicion and will introduce minimal statistical deviations. The author of OpenPuff has tested the tool for resistance to statistical steganalysis based on a suite of CSPRNG tests using Ent [113]. Table 12 shows the following results on 64 KB, 128 KB and 256 MB samples:

Test	Result
Bit entropy	>7.9999xx / 8.000000
Chi square	20% <deviation <80%
Mean value	127.4x / 127.5
Monte Carlo	error <0.01%
Serial correlation	<0.0001
Compression test	0%

Table 12: Results from tests using Ent Suite [113]

OpenPuff's resistance to statistical steganalysis attempts to provide increased security when hiding data. However, the data whitening process requires a large number of carrier bytes for data injection/substitution, and this results in fewer bytes being reserved for the user's data.

5.3 Attacks on the OpenPuff Stego-System

In this chapter, two separate components of the OpenPuff stego-system are subjected to steganalysis via system attacks. The aim of this is to provide further insight into the practical applications of system attacks and system steganalysis. Firstly, steganalysis of OpenPuff MP4 steganography is performed to extend the scope of system attacks and determine if practical results can be achieved against this stego-system. Secondly, the PDF steganography feature of OpenPuff is subjected to system attacks to determine if the vulnerabilities associated with OpenPuff are isolated to video, or if they exist across multiple embedding features in OpenPuff.

5.3.1 Steganalysis of OpenPuff MP4 Steganography

In this subsection, a system attack is presented over the MP4 component of the OpenPuff stego-system. OpenPuff is a multi-feature steganography tool that provides several methods for embedding secret data across file types. MP4 steganography was selected as an ideal carrier to attack as this file type is commonly used throughout the Internet and has become a ubiquitous form of video content.

5.3.1.1 MP4 Data Structure

The embedding algorithm for OpenPuff in part manipulates technical metadata and video metadata in a video file. The technical component used to describe metadata in MP4 files is known as Atoms or Boxes [166]. An atom in an MP4 file is a feature that describes metadata for a given object. This includes components relating to: video frames, audio frames, interleaving AV data, captioning data, chapter index, title, poster data, user data, and technical metadata [94]. The structure of each atom is derived from a type that is identified as a four-byte printable character [61], such as: ftyp, moov, or mvhd. The MP4 file is based directly upon the MPEG-4 part 12: ISO structure, which derives from the QuickTime file format [170]. Through this, atoms are represented as a data hierarchy of parent and child atoms describing the structure and content of a video file.

Figure 43 shows the structure for MP4 atoms. The first “trak” atom (left) is

a media "trak" while the second is called the hint "trak" (right). This hierarchy derives from type "moov" which is placed as a top-level atom [61]. The moov atom acts as an index of the video data and is the component of an MP4 required for video playback. It is this structure of the MP4 file that is modified by the OpenPuff embedding algorithm.

As previously discussed, the video stego-systems that were observed in Chapter 4, hide secret information by injecting data at the end of a video file. OpenPuff is a novel stego-system from what has been previously studied, as it performs steganography by embedding data within MP4 files by modifying their metadata and atomic structure. This embedding approach, paired with a robust security implementation provides resistance to classical statistical steganalysis [113]. However, it is still possible to detect the presence of OpenPuff steganography.

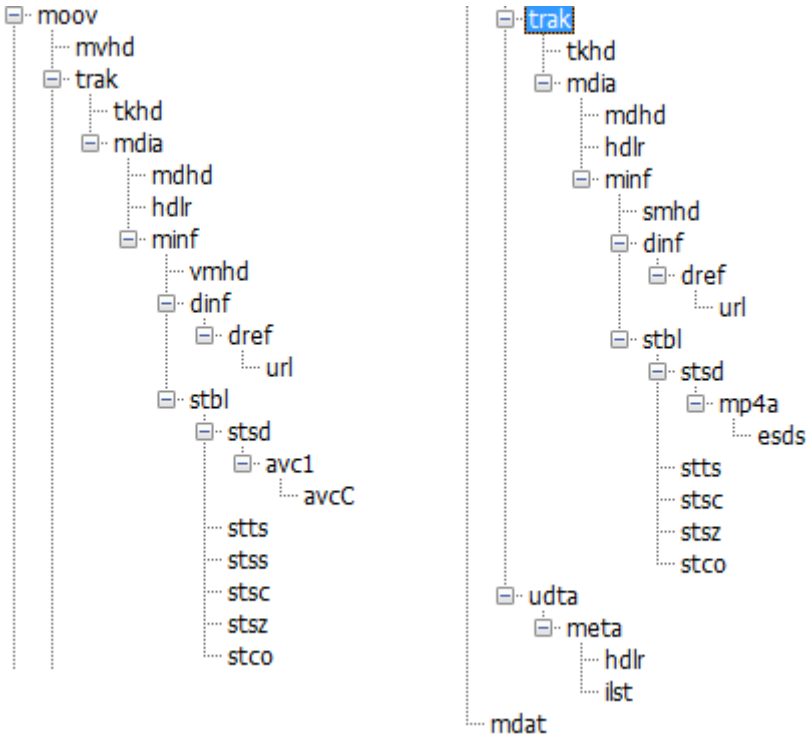

Figure 43: Atomic structure of MP4 video file

5.3.1.2 Impact of the OpenPuff MP4 Embedding Algorithm

Figure 44, shows the atomic structure and metadata output of an MP4 video file modified by OpenPuff. If compared to an unmodified file, the major features (atom size and offset) for each atom remain identical. OpenPuff does not perform modifications directly to these features of the atomic structure, as this would corrupt the playback quality of a stego-object. Instead, OpenPuff modifies semi-redundant atom components that remain unmodified in pre- and post-processing phases.

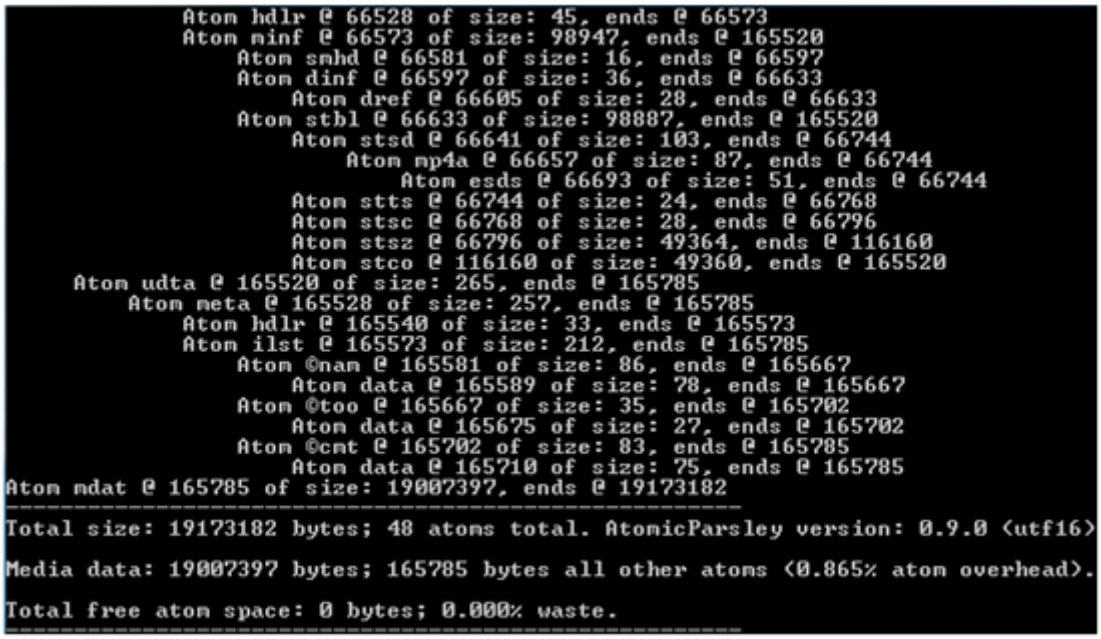

Figure 44: Data size of corresponding MP4 atoms

The features modified by the OpenPuff stego-system are called flags. Flags are reserved space, considered to be a semantic component of the MP4 atom. For OpenPuff these features appear to be consistently modified in any given stego-object. Flags are components of an atom that are preserved in a null state for future use. The OpenPuff stego-system uses this form of digital redundancy to embed secret information. However, this is a feature that once identified, leaves the OpenPuff stego-system vulnerable to steganalysis.

As shown in Figure 45, the flag field of the target STTS (Time to Sample) atom has been modified by the OpenPuff embedding algorithm. The STTS atom typically contains data concerning the duration of a given media sample. This reveals that embedding method uses injection techniques as opposed to substitution over target features. OpenPuff's modification to MP4 atoms is a consistent feature of the embedding algorithm and will occur regardless of cover size, embedded data size/type, and key usage.

Decoding Time to Sample Box	
Start offset	540 (0X0000021C)
Box size	24 (0X00000018)
Box type	stts (0X73747473)
Version	0 (0X00000000)
Flags	0 (0X00000000)

Decoding Time to Sample Box	
Start offset	540 (0X0000021C)
Box size	24 (0X00000018)
Box type	stts (0X73747473)
Version	0 (0X00000000)
Flags	139266 (0X00022002)

Figure 45: MP4 atoms with their respective flag values, cover-object (left) and stego-object (right)

Modifications to the reserved space will occur consistently across a number of atoms in an MP4 file. However, the data embedded into each flag is inconsistent as it is based off pseudo-random encrypted data, implying that it is not possible to construct a signature from any single atom. Following the hierarchy of the MP4 atoms, modifications to the flag value occur from the type **mvhd** and child atoms of type **moov**. Table 13 provides a sample of modifications to a single MP4 file across 24 atoms (other file samples may consist of more or fewer atoms).

Atom (Handler)	**Value (Pre-Stego)**	**Value (Post-Stego)**
MVHD (parent atom)	Null	8240 (0x00002030)
MDHD (video)	Null	16384 (0x00004000)
HDLR (video)	Null	9570320 (0x00920810)
DREF (video)	Null	2097474 (0x00200142)
STSD (video)	Null	2105600 (0x00202100)
STSS (video)	Null	3146304 (0x00300240)
STTS (video)	Null	12583425 (0x00C00201)
CTTS (video)	Null	2129922 (0x00208002)
STSC (video)	Null	4352 (0x00001100)
STSZ (video)	Null	49216 (0x0000C040)
STCO (video)	Null	671744 (0x000A4000)
ELST (video)	Null	1538 (0x00000602)
MDHD (audio)	Null	131650 (0x00020242)
HDLR (audio)	Null	48 (0x00000030)
SMHD (audio)	Null	147474 (0x00024012)
DREF (audio)	Null	8708 (0x00002204)
STSD (audio)	Null	589836 (0x0009000C)
STTS (audio)	Null	262288 (0x00040090)
STSC (audio)	Null	8388800 (0x008000C0)
STSZ (audio)	Null	2105600 (0x00202100)
STCO (audio)	Null	1179666 (0x00120012)
ELST (audio)	Null	4194576 (0x00400110)
META (audio)	Null	10752 (0x00002A00)
HDLR (audio - sub meta)	Null	532545 (0x00082041)

Table 13: Sample of differences between cover-object and stego-object flag values

It should be noted that several flags are reserved in the MP4 file structure to provide metadata configurations and compatibility tests [10]. The reserved flags are unmodified by the OpenPuff embedding algorithm, as arbitrary changes could corrupt the video. Samples of these reserved flags are given in Table 14.

Atom	Value
TKHD (video)	15 (0x0000000F)
TKHD (audio)	15 (0x0000000F)
VMHD (video)	1 (0x00000001)
URL (video)	1 (0x00000001)
URL (audio)	1 (0x00000001)
ESDS (audio)	0 (0x00000000)

Table 14: Reserved flags that will not be modified by the OpenPuff embedding algorithm for MP4

As evident from these findings, a steganalytic method can be constructed to detect the presence of modified flags within an MP4 file. The next section discusses the experimental setup for constructing a system attack against the OpenPuff stego-system.

5.3.1.3 Constructing a Distinguisher

The detection of OpenPuff MP4 steganography is a binary decision for an attacker. In one instance, flag values are null and for each case, the relative weight of suspicion is reduced. In the second instance, flags have an arbitrary value and the relative weight of suspicion increases cumulatively for each encountered non-null flag. Therefore, an efficient method of attack would be to concatenate the result of positive and null values from the flag space of each atom. From this, it is possible to determine with high confidence whether or not a file should be considered a stego-object.

A simple script can be developed as a proof-of-concept for this system attack (as shown in the Appendix A7). The objective of this script is to detect OpenPuff MP4 steganography with a high rate of accuracy and do so in an automated manner so that it would be practical over a significantly large dataset. The detection method would need to incorporate the capability for MP4 file parsing so that the full metadata structure can be extracted and for atom flags to be isolated.

This method would simply check flag values to determine "ifnull" while skipping important flags that have been reserved by the compression algorithm. For each instance "ifnull" is found to be modified, the result can be concatenated to calculate a relative confidence threshold. For each additional result of identifying modified flags, suspicion of OpenPuff MP4 steganography is increased. The threshold used to determine a stego-object in this environment is evaluated in two separate cases of >50% and >90% flags modified in a given object. These are tested in over two separate datasets labelled Dataset 1 and Dataset 2.

To evaluate this proof-of-concept, the script is given the same testing environment originally discussed in Chapter 3. This includes the use of pseudo-randomly generated data embedded in each MP4 file, with pseudo-randomly generated passwords that are unique to each stego-object.

5.3.1.4 Testing and Results

The following section presents two configurations for the testing and evaluation of the presented proof-of-concept. This set-up consists of two datasets used to evaluate detection accuracy and false positive rates. Data is embedded for each bit rate possible for OpenPuff steganography (12%, 14%, 17%, 20%, 25%, 33%, 50%). Datasets are taken from two sources to test detection accuracy for each detection threshold, the first dataset consists of 700 videos pseudo-randomly downloaded from the Internet. The second dataset consists of 700 videos taken from our own self-created collection, the Kent Digital Media Archive (KDMA). Table 15 shows results for Dataset 1.

Embedding Ratio	Dataset size	Detection Accuracy	Run time
.12	100	100%	3.67s
.14	100	100%	3.68s
.17	100	100%	3.56s
.20	100	100%	3.84s
.25	100	100%	3.84s
.33	100	100%	3.85s
.50	100	100%	3.87s

Table 15: Detection accuracy for Dataset 1 > 50% flags modified

As shown in Table 15, detection accuracy gives a perfect score for each embedding ratio tested. Table 16 shows identical results for Dataset 2, which has a higher threshold for identifying OpenPuff stego-objects.

Embedding Ratio	Dataset size	Detection Accuracy	Run time
.12	100	100%	6.13s
.14	100	100%	6.20s
.17	100	100%	6.10s
.20	100	100%	6.28s
.25	100	100%	6.16s
.33	100	100%	6.16s
.50	100	100%	6.22s

Table 16: Detection accuracy for Dataset 2 > 90% flags modified

The difference in run time for the two sets of tests is justified in that the dataset sourced from the KDMA database consists of significantly larger MP4 files than those randomly downloaded from the Internet.

Finally, Table 17 shows false positive (FP) rates for each dataset consisting of 2,500 MP4 files and were tested against varying thresholds for atom modification. From these tests zero false positives were identified.

Dataset	Dataset size	FP rate
1 (>50%)	2,500	0.0
2 (>90%)	2,500	0.0

Table 17: False positive rate for Dataset 1 and Dataset 2

Flag modification appears to be a feature unique to OpenPuff MP4 steganography. It has not been identified as a method of embedding for any other practical application or stego-system. Furthermore, the majority of flags are not modified by any compression features for MP4, suggesting that it is highly unlikely to observe modified flags in any MP4 files clean of steganography.

This section has extended the scope of system attacks over video and has determined that OpenPuff's embedding algorithm, although novel, illustrates another case study for the gap between theoretical and practical steganography. The system attack constructed to detect OpenPuff MP4 steganography has given a high rate of detection accuracy to demonstrate that it can be applied as a highly practical avenue of steganalysis.

In the next section (subsection 5.3.2), the attack against OpenPuff is extended to the PDF embedding algorithm. The aim of this is to determine whether the implementation vulnerabilities for OpenPuff MP4 steganography are an isolated case, or whether they can be observed across multiple features of the tool.

5.3.2 Steganalysis of OpenPuff PDF Steganography

In this subsection, a system attack is presented over the PDF component of the OpenPuff stego-system. PDF documents, although frequently used, are an unexplored area for steganography. The contributions of this research are twofold. Firstly, additional steganalysis of OpenPuff should provide further insights into the implementation vulnerabilities surrounding the tool. Secondly, this work aims to provide a practical steganalytic attack via system steganalysis to detect OpenPuff PDF steganography.

5.3.2.1 Related Work

Known steganalytic results against OpenPuff are surprisingly scarce. The audio component of OpenPuff was examined by [97], who were able to successfully detect OpenPuff steganography as implemented on an older version of the tool(v3.10). The authors proposed the StegAD scheme to detect audio steganography in cloud services, using an enhanced version of the RS algorithm originally proposed Fridrich et al. [47].

The use of PDF steganography has been explored both inside and outside academic literature. Several embedding algorithms have been proposed by researchers, while a number of tools exist for public and commercial use. In current literature, algorithms can either target the PDF format directly, or exploit a form of ASCII steganography, which can then be converted from one text-type format to another. Rafat & Sher [125] propose an ASCII based steganography algorithm to allow conversion between MS Word and PDF files without losing any of the embedded content.

Alizadeh et al. [5] examine a number of existing techniques for PDF steganography including word/character embedding and the manipulation of incremental updates. However, these are not robust and cannot be used outside of the PDF environment. The authors also examine similar but more efficient embedding algorithms such as steganography via hidden PDF components and the manipulation of margins and TJ operators. The authors continue the work of Zhong et al. [173] to develop two similar algorithms. The first has slightly better security but lower capacity and the second offers higher capacity but worse security.

Furthermore, tools such as StegoStick, DeEgger Embedder, and wbStego are examples of what is available to the general public to perform steganography over PDF files.

On the other hand, there are only a handful of published steganalytic attacks against PDF in the existing literature. One such example is a detection method for word shift steganography in PDF, by Lingjun et al. [96]. The authors propose a blind steganalytic attack against steganography through inter-word space length, through the analysis of the statistical properties of spacing.

5.3.2.2 The PDF Data Structure

The contents of this section and the description of the PDF data structure are largely based on [90]. The Portable Document Format, known widely as PDF, is a document format published originally by Adobe as a proprietary model in 1993. It was not until 2008 that it was published as an open standard, as ISO 32000-1:2008. The functionality of a PDF file is largely determined by a series of objects such as dictionaries, arrays, streams and other values (character sets, operators, etc.) that act as metadata to describe the file. The PDF syntax is best understood by thinking of it in four parts, as described below:

- Objects. A PDF document is a data structure composed by a small set of basic types of data objects.
- File structure. The PDF file structure determines how objects are stored in a PDF file, how they are accessed, and how they are updated. This structure is independent of the semantics of the objects.
- Document structure. The PDF document structure specifies how the basic object property types are used to represent components of a PDF document: pages, fonts, annotations, and so forth.
- Content streams. A PDF content stream contains a sequence of instructions describing the appearance of a page or other graphical entities. These instructions, while also represented as object, are conceptually distinct from the basic objects that represent the document structure.

In [90], a stream object is defined as a sequence of bytes. Furthermore, a stream may be of unlimited length, whereas a string shall be subject to an implementation limit. For this reason, objects with potentially large amounts of data, such as images and page descriptions, shall be represented as streams. A stream consists of a dictionary followed by zero or more bytes bracketed between the keywords **stream** (followed by newline) and **endstream**. The keyword **stream** that follows the stream dictionary is followed by an end-of-line marker consisting of either a CARRIAGE RETURN and a LINE FEED, or just a LINE FEED and not by a CARRIAGE RETURN alone. The sequence of bytes that make up a stream

lie between the end-of-line marker following the **stream** keyword and the **endstream** keyword; the stream dictionary specifies the exact number of bytes. There should be an end-of-line marker after the data and before **endstream**; this marker shall not be included in the stream length. There shall not be any extra bytes, other than white-space between **endstream** and **endobj**. Figure 46 provides illustrative examples of the PDF format. Both objects and streams play an important role in OpenPuff steganography over PDF files.

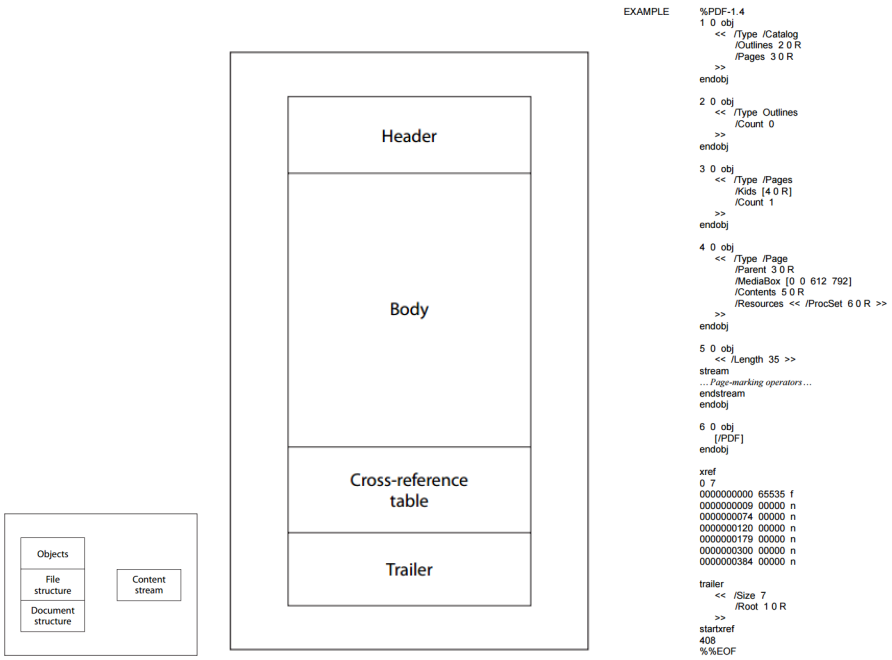

Figure 46: Components (left) and structure (centre) of a PDF file. The code to the right is that of a minimalist PDF example, all from [73]

5.3.2.3 Impact of the OpenPuff PDF Embedding Algorithm

OpenPuff claims that data is pre-encrypted with multi-cryptography before being embedded into a file. The message is then scrambled (shuffled with random indexes), whitened (mixed with a high amount of noise, taken from an independent CSPRNG seeded with hardware entropy), and encoded (with adaptive non-linear

encoding, that takes original carrier bits as input). OpenPuff PDF steganography embeds data through the modification of end-of-line (EOL) markers, called "white-space characters". In particular, the carriage return (CR) and line feed (LF) characters which are used to denote newlines in the metadata of a file. These are represented by 0x0A and 0x0D in hex, which for the purposes of steganalysis, can be used as a key identifier for the detection of OpenPuff PDF steganography.

As shown in Figure 47, the CR and LF characters are flipped once modified. Embedding modifications by OpenPuff should be undetectable by forensic and metadata parsers. This emphasises that the OpenPuff embedding algorithm requires a more focused analysis.

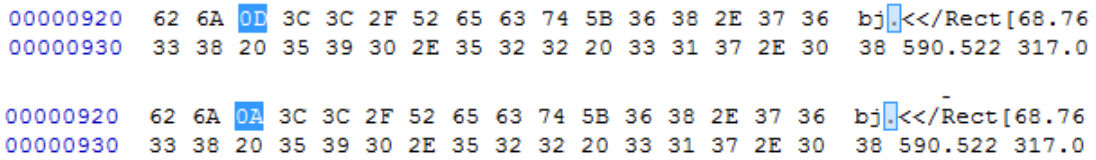

Figure 47: Hex analysis of EOL components as modified by OpenPuff

When the original file employs exclusively the 0x0A character, OpenPuff operates in exactly the opposite fashion. Again, it is the coexistence of both 0x0D and 0x0A markers that reveal hidden contents, because the tool never flips all available values, as it has a maximum capacity of 50%. Because of this, a method can be constructed to perform a system attack against the OpenPuff PDF steganography method and extract CR + LF characters.

5.3.2.4 Testing and Results

To evaluate this system attack against OpenPuff PDF, a proof-of-concept has been created (as shown in the Appendix A8) to detect this method of steganography and examine how effective it can be in a practical environment. To test the accuracy of the attack, two separate experiments are performed. The first uses a small dataset and non-random data embedding. The second comprises a significantly larger dataset and pseudo-random embedded data.

In the first round of testing, a total of 3,000 PDF files were downloaded from the Internet. These were accessed through a web-crawler that traversed three

separate locations. The first was a random selection of URLs generated from the DMOZ archive [9]. The second dataset of PDF files was downloaded from the Archive.org website and the third set of PDF files were obtained from Google Scholar. These were tested in three batches of size 1,000 candidate PDF files which were run through the OpenPuff embedding algorithm and successful carriers were embedded with a small text file that contained a single letter "a". This resulted in a total of 50 successful candidate carrier files that had information embedded via OpenPuff. In addition to this, false positive rates for the distinguisher were also tested. Tables 18 and 19 provide an overview of these tests.

Datasets	**Number of files**	**Processed by OpenPuff (%)**	**Detection Rate**
Dataset 1	1000	10 (1.0%)	100%
Dataset 2	1000	13 (1.3%)	100%
Dataset 3	1000	27 (2.7%)	100%

Table 18: Detection accuracy for first series of tests

Number of files	**False Positives / Negatives**	**Accuracy**
3000 (FP test)	13 / 1	99.6%

Table 19: False positive results from first series of tests

The selection process for OpenPuff appears to be highly restrictive, based on the small number of successfully modified files (approximately 1.6%). Due to the exclusive modification of white-space characters, a PDF file can only be used if the file contains enough carrier bytes.

The second round of testing included a significantly larger dataset. These were downloaded in three separate instances from Archive.org, using the search phrases "PDF", "Computing", and "Medical". The new datasets comprised a total of 13,000 PDF files. From this, 135 were eligible candidates for OpenPuff steganography.

For each eligible PDF file, OpenPuff embedded random data at the maximum capacity offered by the tool (50%). The distinguisher demonstrated results similar to those shown for the first series of tests with again 100% accuracy for each stego-object analysed (see Table 20).

Datasets	Number of files	Processed by OpenPuff (%)	Detection Rate
Dataset 1	5000	56 (1.1%)	100%
Dataset 2	5000	67 (1.3%)	100%
Dataset 3	3000	12 (0.4%)	100%

Table 20: Detection accuracy for second series of tests

False positives produced similar results to those shown on the first series of tests. A false positive rate of 0.5% should ensure that investigative time spent addressing any flagged objects is minimal (see Table 21). This method of false positive testing covers a diverse range of PDF files. In this case, the files have been obtained randomly from different creators and sources and in some cases do not specifically follow official standards.

Number of files	False Positives / Negatives	Accuracy
13000 (FP test)	68 / 0	99.5%

Table 21: False positive results from second series of tests

The detection component of the system attack can be extended to include an estimator. This feature determines the maximum possible number of bytes that could be embedded within each file. The following formula estimates the maximum possible embedded size by using least squares [101].

```
maxhiddensize=0.2204*occurf11+0.2117*occurf21+0.2115*occurf31-194.6563
```

From a practical standpoint, this feature is particularly useful as it provides any practitioner with support to perform forensic steganalysis of stego-object (extraction of the secret message). The estimator was tested against a series of files to show the accuracy for each. Table 22 shows a sample of 10 files with the given maximum capacity for each PDF file and the scheme's estimation. Each file is from a self-created dataset using the presented proof-of-concept (Appendix A8).

File	Maximum Capacity (bytes)	Estimation (bytes)	Error (%)
Test File 1	80	90.64	+13.3
Test File 2	112	110.74	-1.1
Test File 3	64	54.25	-15.2
Test File 4	544	545.19	+0.2
Test File 5	224	218.34	-2.5
Test File 6	864	858.92	-0.5
Test File 7	80	84.01	+5.0
Test File 8	1232	1219.72	-0.9
Test File 9	80	76.62	-4.2
Test File 10	416	412.68	-0.7

Table 22: Estimator results

The attack shown in this section demonstrates a method for the detection of the OpenPuff PDF embedding algorithm. This was evaluated by a proof-of-concept and tested against a total of 16,000 PDF files, which showed a high rate of detection accuracy and a low rate of false positives. This attack paired with the estimator adds credibility to system attacks as a practical method of steganalysis.

5.4 Conclusion

In this chapter, two separate system-attacks have been presented against the OpenPuff stego-system. The first is an attack against the MP4 embedding algorithm that detects arbitrary modifications to MP4 flags. The second attack is against the OpenPuff PDF embedding algorithm, which identifies arbitrary changes to stream data to determine the presence of steganography.

While it is clear that OpenPuff gives consideration to the coding problems associated with real-world steganography, the security improvements have been implemented at the expense of a weak embedding algorithm. The stego-system has not demonstrated weaknesses surrounding poor implementation and encoding, but to avoid these challenges, OpenPuff has used a method of steganography that can be detected with a steganalytic method that offers high accuracy and is optimal

as a practical attack.

The results of both experiments present an unprecedented vulnerability in the OpenPuff stego-system. This research has demonstrated that the vulnerabilities associated with the OpenPuff stego-system are not isolated to any single embedding algorithm and creates the suspicion that other embedding features of OpenPuff will be highly vulnerable to similar steganalytic efforts. This presents an avenue of future work to explore whether multi-functional stego-systems are typically weaker than a counterpart that focuses exclusively on a single embedding feature.

In addition, this chapter has shown that the presented attacks extend the scope of system steganalysis to video and PDF and can achieve practical results. Each method gives support towards the practical applications of system attacks and demonstrates a successful endeavour to bridge the gap between steganalysis in the laboratory and steganalysis in the real-world.

There are several novel contributions from this chapter that can be discussed.

- An attack against the MP4 component of the OpenPuff embedding algorithm. The presented method has shown an optimal rate of detection accuracy and efficiency and can extend to any other steganographic tools (existing or emerging) that might adopt similar practices in the embedding operation.

- A second novel attack has been presented against the PDF component of the OpenPuff embedding algorithm. The presented method provides a steganalytic resource to the community from the proof-of-concept experimentation. This gives support 0to a domain that otherwise has a limited scope of existing research.

- System attacks have proven to be an effective method of breaking the OpenPuff stego-system over MP4 and PDF as demonstrated through this chapter's case studies. This research not only supports the applicability of system attacks to video and PDF, but also exposes the coding challenges surrounding video steganography.

Finally, the results of this chapter have shown that there is potential for future work, as OpenPuff appears to be a highly popular stego-system that is still in

use. It is important to understand how the other embedding features of OpenPuff function and determine whether they can be broken by system attacks.

Chapter 6

Autonomous Framework for System Attacks

Introduction

There has been a rapid growth of interest and progress in steganography over the past two decades. New avenues of application have been explored and tools are developed to support an increasingly large community of users. Video stego-systems have emerged as a modern paradigm that provide an ideal method of communication. This is in part due to the pervasive nature of video content, but also the many specific advantages that relate to video steganography.

From a practical approach, video cover-objects provide a multiple paths for steganographic embedding. Video files are an amalgam of data streams (frame, audio, auxiliary) and different methods of steganography are applicable to all of them. However, many existing stego-systems avoid any mathematical and coding challenges and use only very simple methods of steganography. This is evident from previous chapters, where auxiliary channels are employed to avoid dealing with the security considerations for temporal/transform domain embedding. In addition, the impact of these methods over video will generally not be as apparent to a casual observer when compared to other forms of steganography. New stego-systems are continually emerging but there is a lack in the development of blind steganalytic system-attacks capable of detecting a range of implementation

vulnerabilities across many different stego-systems.

Previous chapters of this thesis have explored practical methods for system steganalysis with a particular focus given to video (Chapters, 3, 4, and 5). This chapter presents the culmination of this research, and presents a framework to evaluate implementation vulnerabilities that can form the basis of practical system attacks.

In this chapter, a framework is evaluated for automatic system steganalysis, based on the construction of a classifier to detect implementation vulnerabilities introduced by stego-systems. This classifier operates on attributes that are extracted from the training data using pattern-matching. The choice of classifier is Naive Bayes classifier as it provides flexibility over a simple signature-matching system.

The framework has been applied successfully to a wide variety of schemes and across different formats, because it has been designed to be as general as possible. Hence, it makes very few assumptions on the underlying nature of the stenography tool or cover media being used.

The resulting framework has several improvements on previous work in this field. First, it decouples attribute extraction from classification. Also, this work presents a generalisation phase that is able to improve results, provided that training media objects are representative of real-world media objects. Finally, the trained classifier can operate even when some of the obtained fingerprints for a given steganographic tool are not present in observed media, as the constructed model may combine several different findings for a given tool.

The rest of the chapter is structured as follows. Section 6.1 provides a discussion of related work in the field and gives an overview of Bayesian classification. Section 6.2 applies this framework to a series of datasets created with several steganography tools. In this section, the method is discussed and the performance of approach is evaluated to provide a general discussion of its limitations. A conclusion is given in Section 6.3, where final thoughts and insights surrounding the contributions of this work are discussed. [1]

[1] **The research in this chapter was conducted as a collaborative effort with University Carlos III of Madrid. I would like to thank Dr Alejandro Cervantes and Professor Pedro Isasi, who developed the framework.**

6.1 Related work

6.1.1 System steganalysis

System attacks rely on unintentional implementation vulnerabilities that are often relatively independent of the steganographic scheme. This has been briefly discussed before. Previous research [141, 140] used system-type attacks to examine and evaluate the current state of steganography tools, with a focus on video-based applications. This work found that many of the most accessible and popular tools share similar embedding schemes and were highly susceptible to system attacks.

Chen et al. have discussed the concept of file structure steganography and its application to images. Their work has shown how implementation vulnerabilities can be used to reliably find signatures [25]. They noted that one particular advantage of this type of approach is that it is not affected by techniques typically employed to increase resistance against statistical steganalysis. Cantrell examined file-structure based steganography with a focus on digital documents [21]. This work further emphasised the need for steganalytic systems that exploit the very common vulnerabilities arising from a poor handling of metadata during the steganographic process.

System attacks have already been successful in the field of steganalysis. In [14], the authors propose an attack that is aimed to detect the use of steganographic software independently of the actual media used. The process simply compares the average values of bits at fixed positions of both an unmodified object and the corresponding stego-object when subject to the steganographic tool. Results showed that many tools were introducing anomalies that could be easily used to detect stego-objects. However it must be noted that if the anomaly is not present at the exact same position in all files, this method can't detect it. Also, objects are only labelled as stego-objects if all the regularities are present, due to the problem of false positives.

Another work [115] extends this idea to the search of binary or hexadecimal patterns by proposing a recognizing automaton that can also find regularities that are not restricted to specific positions in the analysed objects. This method was used for steganography on image files, and different policies were defined for the

header, content, and tail parts of those formats.

The implementation of sequence extraction presented in this chapter is an extension of this type of approach. However, it is more powerful in several ways. First, this method uses a formal definition of a pattern by using a specific grammar. This makes clear the type of regularities that the system is able to detect. A more complex grammar might be used with increased detection capabilities. Similarly to the most recent work [115], the process of pattern extraction and verification is not restricted to specific positions in the file. However, in this case, assumptions are not required for a given object that is analysed. This allows the presented method to work with a broad range of different file formats. Third, phases for pattern extraction and classification are separated. This is done so both that components can be substituted if required. Finally, a probabilistic classifier is generated that is more complex than simple signature detection. This is performed to complete a task using a combination of patterns that are not necessarily present in all given objects. In the discussion of the results, cases are identified and discussed where classical signatures were found. This method is tested over several stego-systems not covered in previous works and presents multiple new results.

6.1.2 Overview of Bayesian Classification

Bayesian Probabilistic Classification is a broad but well-established research area with algorithms currently deployed in many different real-world scenarios and fields of research. A general definition for all such systems would state that each relies on the construction of a Bayesian Network [117], that takes as evidence a feature set extracted from the data; the feature set can be queried to make a prediction for the class to which data belongs. Prior knowledge of the problem can be used to determine the structure of the network, and known data is used to learn the network parameters, the structure and/or the Conditional Probability Tables (CPTs) required to execute queries. In addition, some approaches employ Markov Chains to reflect the temporal dimension. A good introduction to Bayesian Probabilistic Classification can be found in [63].

Similar strategies have recently been used in a computer security context. For instance, in [81] the authors construct a Network Intrusion Detection System

(NDIS) that uses byte patterns as attack signatures for generating alerts. This Bayesian Network can even act dynamically (providing stateful detection) [36]. However, learning the parameters for the Bayesian Network can be a very complex process. Also, there are circumstances in which there is not *a priori* knowledge of the best structure for the network. For these reasons, simpler approaches are often preferred. For example, in [6, 109, 74] the authors employed a Naive Bayes Classifier [42], which is equivalent to a simple Bayesian Network with no inter-feature dependencies.

For this work, the steganalysis detection problem can be formulated as a binary classification problem. In this case, cover-objects are defined as belonging to $Class_0$ (or negatives) and stego-objects belong to $Class_1$ (or positives). Each object (X_i) is represented by a set of up to N attributes ($X_i = \{A_{i1}, A_{i2}, \cdots, A_{iN}\}$).

The principle of Bayesian classification is to estimate the posterior class probability of an observation, X_i, using the empirical probabilities obtained from a previous sample of the domain. For any class k, $Pr(Class_k|X_i)$ can be estimated from $Pr(X_i|Class_k)$ using the Bayes Theorem

$$Pr(Class_k|X_i) = \alpha Pr(Class_k)Pr(X_i|Class_k) \tag{7}$$

where $\alpha = 1/P(X_i)$ is independent of the class.

In the general case, obtaining $Pr(X_i|Class_k)$ is not a trivial task, because the distributions of features may have multiple dependencies and correlations. However, previous experience in this domain suggests that this needs not be the case for the problem being addressed. Also, a simpler model can be more accommodating to analyse a broad class of steganographic schemes, as features are not the same for all schemes. For these reasons this work uses the simplest approach to Bayesian classification, the well known Naive (or Simple) Bayes classifier [42].

This approach is based on a strong assumption, which is the conditional independence of the distribution of values for each attribute (Eq. 8). This kind of model is simple enough to train, and has proved to be reliable even in scenarios where it is well known that there is at least a partial violation of its theoretical assumption [40].

$$Pr(X_i|Class_k) = \prod_{j\in\{1..N\}} Pr(A_{ij}|Class_k) \tag{8}$$

Where $Pr(A_{ij}|Class_k)$ is an abbreviated notation for the probability of observing the particular value A_{ij} as the $j-$th attribute for any object of class Class$_k$.

The most probable class is assigned to an unknown observation (Eq. 9). If classes are balanced then $\alpha Pr(Class_k)$ is also constant, and only in this case the class can be selected by using the one producing a maximum product of likelihoods. This is called the Maximum Likelihood Rule.

For this particular domain, the values for each feature are obtained by finding the matches of a set of patterns in the file extracted from training data (shown later in this chapter). The resulting N attributes are used then to predict the class of the file using Eq. 9.

$$Class(X_i) = \arg\max_{k\in\{0,1\}} \prod_{p\in Model} Pr(Class_k) \times Pr_p(A_{ip}|Class_k) \tag{9}$$

Where each factor $Pr_p(A_{ip}|Class_k)$ is the probability of obtaining the value A_{ip} in files of class $Class_k$, when the pattern p is applied to a file X_i.

Let us assume that the possible values obtained when matching pattern p on the available objects are $v_{pt} \in \{v_{p1}, v_{p2}, \cdots\}$. Then, the task of model construction becomes equivalent to obtaining the probability distributions $Pr_p(v_{pt}|Class_k)$, for each specific value v_{pt}. This can be done by replacing the actual distribution by the empirically estimated $Pr^*_p(v_{pt}|Class_k)$, calculated as a frequency. First, each pattern is matched against both a set Set_0 of cover-objects and a set Set_1 of stego-objects. Then, for each value v_{pt} thus obtained, calculate its frequency using Eq. 10.

$$Pr^*_p(v_{pt}|Class_k) = \frac{Count(v_{pt})\ \forall objects \in Set_k}{|Set_k|} \tag{10}$$

These constitute the empirical Conditional Probability Tables (CPTs) of the constructed Naive Bayes Model.

The previous concepts are used in Steps 3 and 4 of the procedure, Model Generation and Model Refining. An overview of the framework is shown in Appendix B.

6.2 Experimentation

In this section the framework is applied, as described in the previous section, to datasets created with 11 different steganography schemes. Results are then generated to evaluate the success of the presented approach. For eight of these schemes, results are obtained that can be considered either important or promising, while for the remaining three, the presented approach was not successful. A discussion of the results on these latter approaches is given as future work.

To begin, a description of the data generation process is provided. Data generation is critical for creating a robust experimental setting, because an incorrect generation process would lead to the detection of phantom vulnerabilities, i.e. introduced by the datasets (such as the detection of repeated passwords or predictable file names, etc.), instead of real vulnerabilities of the steganography schemes themselves. Then, a discussion is given on how the values of some parameters that are of relevance to this framework are decided. After that, the global results over all datasets are presented, followed by a detailed description of the actual solutions found, and how to use them to automate detection for each steganography scheme.

6.2.1 Data generation

Traditional models in steganalysis often rely on knowing everything about the system, with the exception of a secret key that is shared between the two parties. This is known as Kerckhoff's Principle [62]. This presented steganalytic approach, instead, can be considered as a blind attack with a known cover [65]. The framework does not assume any pre-existing knowledge on how each stego-system fundamentally operates. A classifier is trained that works by extracting features from both cover and stego-objects, to build a distinguisher that does not rely on any preliminary knowledge or assumption.

To ensure that the process of dataset generation did not introduce any bias into the solution, passwords and embedded contents were kept unique to each object. For any given stego-system, 100 pairs of stego-objects and cover-objects are generated from our own repositories (such as the Kent Digital Media Archive),

and from external souces such as archive.org. The passwords were created randomly, using the service at http://randomkeygen.com, using different password tables as sources (256-bit WEP, 156-bit WEP, 128-bit WEP, 64-bit WEP, "Decent Passwords", etc.).

Contents to be embedded were created randomly thanks to the service at https://www.random.org/bytes/. Content size varied between 1-byte and 10KB. This process thus ensures that each message and key was unique inside a given dataset. Table 23 lists the actual tools used for embedding.

Name	Version
F5	1.0
OpenPuff	4.0
OurSecret	2.5.5
OmniHide Pro	1.0
Masker	7.5
DeepSound	2.0
Pixelknot	1.0.0

Table 23: Steganography tools used in the experiments.

It can be argued against the generality of the use of only 100 pairs of objects. In this sense the framework has been limited by the size of the datasets for practical reasons, as the whole process of collecting, encoding data, etc. had to be repeated for each of the analysed tools. Let's suppose that one identifies a given sequence of bytes as consistently found in all stego-objects for a given tool. Later, when testing on a different collection of objects, that sequence may not match any object if it included some bytes that are linked to a specific encoding option that was not used in the collection. If this system detected additional sequences for this tool, then this may not affect the result, but in a general case, this is an obvious problem. There is no easy way to go around this problem besides training the model with larger and more diverse collections of objects, which is why the results are presented as a methodology or framework. In this sense, results are reported that may be general but are likely open to improvement.

6.2.2 Experimental Process

For each dataset, the experimental procedure was performed as described in Algorithm 2, and N-fold cross validation was used. This method splits the available data in N disjoint subsets called folds, and then performs N experiments. On each experiment, one fold is reserved to evaluate the model (validation set) and the rest is used for training. The procedure generates N models as a solution. Results for the N experiments are averaged, to provide a final overall result for the procedure. For each experiments, $N = 10$ is used as this is a commonly used technique in relevant academic literature [8].

As a measure of accuracy, the classification success rate is used, that is, the average number for the sum of true positives (stego-objects correctly classified) and true negatives (cover objects correctly classified). This measure is calculated by testing the trained model on a validation set that was randomly extracted from the generated data set and was not used for neither attribute extraction nor model generation.

Algorithm 2 Experimental procedure for N-fold cross-validation experiments with constant length of patterns ($N = 10$ for the experiments)

Sample data into N arrays TRN, TST, VAL
for $fold = 1$ to N **do**
  $cnstlen \leftarrow 10$
  while $success < Threshold$ **do**
    Patts $\leftarrow \emptyset$
    for *cover*,*stego* $\in$ TRN **do**
      Patts $\leftarrow$ Patts $\cup$ $extract(cover, stego, cnstlen)$
    end for
    Patts $\leftarrow$ Patts $\cup$ $generalize$(Pats)
    Model $\leftarrow \emptyset$
    for *pattern* $\in$ Patts **do**
      for *cover*,*stego* $\in$ TRN **do**
        Model $\leftarrow$ Model $\cup$ $getCPT_0(pattern, cover)$
        Model $\leftarrow$ Model $\cup$ $getCPT_1(pattern, stego)$
      end for
    end for
    Model $\leftarrow$ $refine$(Model)
    $success \leftarrow test$(Model, TST)
    $cnstlen \leftarrow cnstlen - 1$
  end while
  Models[$fold$] $\leftarrow$ Model
  Results[$fold$] $\leftarrow$ $test$(Model, VAL)
end for
$result \leftarrow average$(Results)
return $result$, Models

This procedure calls several functions:

- *extract* is the function that searches for patterns.
- *generalize* is the function that creates the generalized patterns.

- $getCPT_i$ is the function that calculates the conditional probability tables for each class i.
- $refine$ performs Model Refining.
- *test* classifies a set of labelled objects, and obtains a result in terms of success rate.

The Pattern Extraction phase requires two parameters: the length of the pattern constant part (*cnstlen*), both for Flag Replacement patterns and for EOF code injection patterns. For Flag Replacement another parameter is required, *varlen*, the maximum length of the variable part. The actual length of the variable part (*varlen*) was allowed to take any values between 1 and 5 for all the experiments.

The value of *cnstlen* is important, because longer patterns are less likely to produce false positives. However, searching only for long patterns reduces the chance of finding consistent but shorter signatures. In order to set the best value for this parameter, a maximum value was first used and a candidate model was generated. Then, this model was tested on a separate set of objects (test set) that were not used for training. If the success rate obtained on this test set did not reach a certain threshold, the training process is refined, decreasing the value of *cnstlen* in fixed steps. Otherwise, the training process stops and the model is considered final.

Also, in the Pattern Extraction phase, the search can explore all or only a part of each file. Preliminary experimentation found that it was enough to search 1 KB of data at the start of the file (or at the start of the injected code, generally towards the end of the file) to attain good results. Thus, these two sections are explored for each of the files in the given pair. This limit is optional, and can be increased depending on the available computational resources. Also if specific knowledge of file structure for a format is available, search can be directed to the specific sections of the file that are known to be likely to be modified as a side result of the embedding process (headers, etc.). For the subsequent phases, model generation and validation, the whole file was always considered: that is, a search is performed for these patterns on the whole length of the file.

For the clustering mechanism used for pattern generalisation, a distance threshold (Th) has to be defined. This is the maximum percentage of bytes that may

differ in patterns that belong to the same cluster (intra-cluster distance). The value of $Th = 0.20$ is set after some experimentation, as it tended to lead to good results.

Table 24 summarizes the parameters used in the experiments.

cnstlen	*varlen*	Explored	Th
$10 \rightarrow 2$ (steps of -1)	$\{1, 2, 3, 4, 5\}$	1 KB	0.20

Table 24: Parameters used in the Experimental Procedure.

6.2.3 Experimental results

Results and new findings are shown in this section, corresponding to three different steganographic tools that had no previous known attacks. Results are also included over a number of other tools with published weaknesses and for each case due citation is given.

The experiments were performed using 10-fold cross-validation. Results are the average success rate for the 10 folds. In Table 25, the success rate is provided as attained in the experiments that can be considered successful, and for which further analysis is performed in section " Discussion on the solutions". In these cases a success rate of practically 100% was obtained for most of the data sets, which means that vulnerabilities can be automatically identified that are related to weak coding of the steganographic tools.

Conversely, Table 26 shows that the presented method was not able to obtain satisfactory results in all the analysed steganography schemes, due to several reasons, for the OpenPuff tool. Partial results are commented in depth in " Conclusion".

Even when this automatic process can be useful as an attack, true insight on the nature of the vulnerability requires further explanation of the results. In some cases, models are obtained that are capable of detecting stego-objects, in other cases, the classifier is identifying a regularity in the cover-objects. It is possible that a single regularity is found that is able to perform classification, while in others, several different regularities were found and had to be combined in order to detect the use of steganography. Therefore, in order to be certain that the

Tool	Method	Format	Success Rate	Refs.
Verification of previous results				
F5	F5 Scheme	JPEG	100%	[51]
OpenPuff	Flag Replac.	MP4	100%	[141]
OurSecret	EOF Data Inj.	MP4	100%	[140]
OmniHide Pro	EOF Data Inj.	MP4	100%	[140]
Masker	EOF Data Inj.	AVI	100%	[140]
—New results—				
OpenPuff	**Flag Replac.**	**FLV**	**100%**	
DeepSound	**Unknown**	**WAV**	**99.5%**	
Pixelknot	**F5 Scheme**	**JPEG**	**100%**	

Table 25: Average Success Rate for successful experiments.

Tool	Format	Matched files	Success Rate	Comments
OpenPuff	MPEG	80% in extended dataset	99.5%	False negatives Base data set not general
OpenPuff	MP3	47.5% of our data set	100%	Results not general enough
OpenPuff	WAV	0%	No patterns found	

Table 26: Average Success Rate for unsuccessful experiments.

resulting solutions are truly useful, the results have to be analysed case by case. Thus, a detailed description of a representative solution is included for each of the analysed steganography schemes in the following section.

6.2.4 Discussion on the solutions

This section shows and discusses the relevant parts of the model identified for the data sets for which satisfactory results were obtained. Not all the patterns in those models are listed, as the training process generates many redundant patterns. Instead, the "best" patterns in each model are provided. This section is included in order to provide useful sequences for detection of stego-objects that allows in some cases to avoid the use of a full Bayes classifier, and instead requires only a simple "signature detection" mechanism.

Best patterns are selected manually using one of the following criteria: either they match the greatest number of files in the corresponding data set (that is,

the *generality* of the pattern), or they have a high success rate over the files they actually match. In tables 27, 28, 29, 30, 31, 32, 33, and 34, two values are provided: "matches" means the percentage of objects of the proper type where the pattern was found, while "success" is the classification success rate measured only on those matched objects.

6.2.4.1 OpenPuff FLV

Though it was possible to detect many patterns that were useful to classify some of the files, no pattern alone matched all files and provided a 100% success rate. In Table 27, it is shown that patterns 1 and 2 are some of the patterns that were found more commonly. These two patterns achieve a 100% success rate if combined in a single model such as the Naive Bayes model. That is why results of 100% for this tool are reported in Table 25.

Pattern 1 is related with the default null values located in the FLV header, to describe streamID. Research into FLV steganography through metadata tags has been discussed before in the academic literature, showing that stego-systems can embed data into metadata tags without corrupting the file whilst maintaining a good degree of imperceptibility [33, 107].

Pattern		Matches	Distribution	Success
1	`0000 0012 0001 3600 0000 ???? ???? 0200 0a6f 6e4d 6574 6144`	79.0%	Cover: `0000 0000` Stego: ¬ `0000 0000`	100%
2	`0141 0900 0039 0000 0000 ???? ???? 0000 0000 014d 001f ffe1`	68.0%	Cover: `0000 0017` Stego: ¬ `0000 0017`	100%
Both patterns 1 or 2		100.0%		100%

Table 27: Best patterns found for the OpenPuff FLV tool.

A method that ensures detection of stego-objects over the FLV scheme would use both patterns, 1 and 2, as in the following:

- If the sequence `0000 0012 0001 3600 0000 ???? ???? 0200 0a6f 6e4d`

`6574 6144` is found and any value in the indicated positions is **not** `0000 0000`; *OR*

- If the sequence `0141 0900 0039 0000 0000 ???? ???? 0000 0000 014d 001f ffe1` is found and the value of those positions is **not** `0000 0017`, then the object can be classified as a stego-object.

This method covers 100% of the files and achieves a 100% success rate in distinguishing cover from stego-objects with this tool. Thus, it can be considered a new steganalytic result over this tool and format.

6.2.4.2 OpenPuff MP4

In this case, it was possible to find a single pattern that matched all files and provided 100% success. Auxiliary patterns were also identified that have shown to be successful. These results are shown in Table 28.

Pattern	Matches	Distribution	Success

`6600 0000 1c64 7265`
`6600 ???? ???? 0000`
`0100 0000 0c75 726c`
	100%	Cover:
		`0000 0000`
		Stego:
		¬ `0000 0000`
			100%

`6961 0000 0020`
`6d64 6864`
`???? ????`
	100%	Cover:
		`0000 0000`
		Stego:
		¬ `0000 0000`
			99%

Table 28: Best patterns found for the OpenPuff MP4 tool.

These results are consistent with the findings published in [141] for the OpenPuff MP4 steganography scheme.

Using the patterns that has a 100% success rate would yield the following rule for quick classification: All stego-objects include the pattern `6600 0000 1c64 7265 6600 ???? ???? 0000 0100 0000 0c75 726c` where the value of the variable part is not `0000 0000`.

6.2.4.3 DeepSound WAV

The tool DeepSound proved to be more of a challenge. For DeepSound, patterns extracted straight away from a pair of cover and stego-objects were never found in any different pairs. When pairs of stego- and cover-objects were analysed, patterns were found that had shown to never be consistent over more than one pair.

However, in this case good results were obtained by introducing the generalization process, as all the patterns that were found in separated pairs of objects could be automatically combined in the extended pattern shown in Table 29. Bytes that had different values across the data in that pattern were replaced by the general sign "*" and are not used to extract the attribute value. This model was able to achieve perfect accuracy results in the experiments.

There were, however, two cover-objects and one stego-object whose values for this feature were different from the rest. This posed a problem for the cross-validation experiments, that did not reach 100% success for the folds where those files were not used for training, thus the resulting value of 99.5% in Table 25. This problem was the result of the limited number of objects used for training (see section " Data generation"). The reason for reporting 99.5% accuracy was thus related to fact that a small number of objects of these types were available. It may be related to characteristics of the cover object, but further analysis is pending to provide insight of the singularity of these objects.

Table 29 describes one of the models that used those objects for training, where the aforementioned problematic values were included.

From these results, a simple method can be described for the detection of steganography with DeepSound. In every stego-object the pattern `1000 6461 7461 bc** **** **** **** 0500 0300 0400 0300 0400` can be found (the values of the "*" positions should not be taken into account). Note that because two different attribute values for the stego-objects are given, for this detector, part of the value can be replaced by the wildcard character (*). This may be useful to avoid any bias that might be introduced in the data generation procedure. This pattern matched all the available stego-objects and none of the cover-objects, and is thus far the first steganalytic result reported on DeepSound.

Pattern	Matches	Values	Prob.	Success
1000 6461	100%	Cover:		100%
7461 bc**		0000 0000 0000	0.96	
**** ????		0000 0000 0000		
???? ????		0000		
???? ????		feff feff 0200	0.02	
???? ????		0200 0400 0400		
		0000		
		0000 0000 0400	0.02	
		0040 0030 0030		
		0200		
		Stego:		
		0400 0400 0500	0.98	
		0300 0400 0300		
		0400		
		f4ff f4ff 0500	0.02	
		0300 0400 0300		
		0400		

Table 29: Best patterns found for the DeepSound WAV tool Probability values are listed behind each of the alternative values for a feature.

6.2.4.4 F5 JPEG

For this tool, a pattern has been identified that matches every file pair and achieves a 100% success rate in classification. This pattern is shown in Table 30. Cover-files have random values after the first byte of the file, while stego files have a constant string as shown in the Table.

Pattern	Matches	Values	Success
ffd8 ????	100%	Cover: ffe1	100%
???? ???? ????		¬ 0010 4a46 4946	
???? ???? ????		0001 0000 0001	
		Stego: ffe0	
		0010 4a46 4946	
		0001 0000 0001	

Table 30: Best patterns found for the F5 JPEG tool

The signature found by this system was ffd8 ffe0 0010 4a46 4946 0001

`0000 0001`, which is the start of a constant block of data inserted by the F5 algorithm implementation. This block then contains a known and relatively long signature that translates to the ASCII string "JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech." and was previously reported in [51]. If the extraction tool is run with the aim of searching for patterns with a longer variable part, it manages to extract the complete long signature, further showing the viability of the presented approach. This signature was not found in cover objects, where the block starts with the value `ffd8 ffe1` and then continues with different, unrelated values.

It is important to note that this result indicates a problem with the data generation process. The header `ffd8 ffe1` corresponds to cover-objects in EXIF format, but the tool generates JFIF format, where that header changes to `ffd8 ffe0`. Because it didn't include files of JFIF format in the cover-files, the method includes the value `ffd8 ffe1` as constant part of this signature. If the data had included cover-files of both types (EXIF and JFIF) files for output, the model would have included two patterns, each one matching one of the two versions of the header.

6.2.4.5 Pixelknot JPEG

Pixelknot is an popular steganography application for Android devices, available at the Google Play store, which is part of *The Guardian Project*, a group of developers that make open-source secure mobile apps for everyone but particularly tailored for Journalists, Human Rights Activists and citizens in repressive regimes. Their web is at https://guardianproject.info. This issue has been responsibly disclosed to their team before publication.

For Pixelknot, which had no previous reported weaknesses, it was possible to detect the exact same vulnerability than for F5. The signature found can also be used to classify files with a 100% success rate (see Table 31). This is not entirely surprising, as the developers recognise their tool is based on F5. What is interesting is that the long pattern present in F5 is not present in Pixelknot, but despite this slight improvement in security, other identical patterns are present and identified by the framework.

Pattern	Matches	Values	Success

`ffd8 ????`
`????  ????  ????`
`????  ????  ????`
	100%	Cover: `ffe1`
		$\neg$ `0010 4a46 4946`
		`0001 0100 0001`
		Stego: `ffe0`
		`0010 4a46 4946`
		`0001 0100 0001`
			100%

Table 31: Best patterns found for the Pixelknot JPEG tool

6.2.4.6 OurSecret MP4, Omnihide MP4 and Masker AVI

The approach has been applied to three algorithms known to use EOF data injection. Models were then generated that were able to achieve 100% accuracy in each case.

The framework was able to extract several constant patterns for these files. In all cases patterns matched all the stego-files and never matched any of the cover-files. This is, of course, the best case scenario for the steganalyst.

In Tables 32, 33 and 34, as attributes are not given, you can list the probability of matching cover-objects or stego-objects.

- For OurSecret, it was possible to find a signature that had previously been found (manually) by previous work on this steganographic scheme [140]. In Table 32, the first 10-byte patterns detected are listed, which can be joined to form the full signature `9e97 ba2a 0080 88c9 a370 975b a2e4 99b8 c178 720f 88dd dc34 2b4e 7d31 7fb5 e870 39a8 b842 7568 7191`.

- For OmniHide, a sequence of bytes was detected with the value `2020` in stego-objects that was not present in any cover-object. Although this sequence can be used to classify the data with 100% success rate, as a signature it is problematic as it is relatively low entropy, so has the potential to lead to many false positives. It simply encodes a series of blank spaces to accommodate file names, whose hexadecimal code is `0x20`. a test was conducted for false positives to determine the chance of this signature appearing naturally in MP4 files. These results can be seen in Section 6.3.5.

	Matches	
Pattern	Cover	Stego
99b8 c178 720f 88dd dc34 2b4e 7d31 7fb5 e870 39a8	0%	100%
c178 720f 88dd dc34 2b4e 7d31 7fb5 e870 39a8 b842	0%	100%
720f 88dd dc34 2b4e 7d31 7fb5 e870 39a8 b842 7568	0%	100%
Plus the rest of the patterns included in the 40-octet signature		

Table 32: Best patterns found for OurSecret MP4 tool

	Matches	
Pattern	Cover	Stego
2020 2020 2020 2020 2020 2020 2020 2020 2020 2020	0%	100%

Table 33: Best patterns found for OmniHide Pro MP4 tool

- For Masker, the signature 3030202020202020 was identified. This signature was consistently present in the initial $1KB$ of the stego-object. This pattern can be used for classification, though it can also be seen as a weak detector due to its low entropy. It can be expected that it might lead to an large percentage of false positives. So it is important to continue pursuing the identification of different signatures. This was successful and a longer signature was found that appears in a variable position in each of the stego-objects. This finding is consistent with previous results in [141]. The full 73-octet string that can be used as signature is:

  9e8d db50 73c2 bf65 b81e 03ae d562 c6cc 719a b9b9 4849 d264 ee49 b763 7bef eba1 0104 407b f0ed 2d86 47e1 5ffc 7c41 c98b c902 ca1c 0721 ee8d 3266 477c 8fe1 4ee8 ae66 ab32 e3d2 f9a1 0e.

  In Table 34 the three first patterns found are shown that are part of that signature.

	Matches	
Pattern	Cover	Stego
`9a9e 8ddb 5073 c2bf 65b8 1e03 aed5 62c6 cc71 9ab9`	0%	100%
`8ddb 5073 c2bf 65b8 1e03 aed5 62c6 cc71 9ab9 b948`	0%	100%
`5073 c2bf 65b8 1e03 aed5 62c6 cc71 9ab9 b948 49d2`	0%	100%
Plus the rest of the patterns included in the 72-byte signature		

Table 34: Best patterns found for Masker AVI tool

6.2.5 False Positives Testing

For each format and tool that has presented a signature, a large number of files obtained from various sources in which steganographic content was not expected or assumed. This is to determine whether the proposed signatures are likely to occur naturally inside cover files. Table 35 shows the signatures used for each tests.

Tool	Number of Tests	Signature
F5 JPEG	2000	ffd8 ffe0 0010 4a46 4946 0001 0000 0001
OpenPuff FLV	1000	$\neq$ 0000 0000 and 0000 0017
OurSecret MP4	2000	See Table XIII
OmniHide Pro MP4	2000	2020 2020 2020 2020 2020
Masker AVI	1000	3030 2020 2020 2020
OpenPuff MP4	2000	Atom detection
OpenPuff MPEG	1000	Arbitrary values
Pixelknot JPEG	2000	ffe0 0010 4a46 4946 4946 0001 0100 0001
DeepSound WAV	1000	0500 0300 0400 0300 0400

Table 35: Test signatures for the false positive analysis

Table 36, shows the results for false positives tests across known signatures. For each test a new batch of cover files were used. JPEG images were downloaded from the Dresden Image Database [64]. Videos were tested across MP4,

AVI, FLV and MPEG formats. These files were obtained from YouTube (25% of files), Archive.org (50% of files) through an automatically generated query, and our own KDMA archive (25% of files). Audio WAV files were obtained from WAV-source.com.

Tool	Method	Format	False Positive Rate
F5	F5 Scheme	JPEG	0%
OpenPuff	Flag Replac.	MP4	0%
OurSecret	EOF Data Inj.	MP4	0%
OmniHide Pro	EOF Data Inj.	MP4	0%
Masker	EOF Data Inj.	AVI	0%
OpenPuff	Metadata	FLV	0.4%
OpenPuff	Metadata	MPEG	0%
Pixelknot	F5 Scheme	JPEG	0%
DeepSound	Unknown	WAV	0%

Table 36: Tests for false positives across known signatures

The false positives for Masker only consider the shorter signature. If the full 73-byte signature is used, there is a rate of zero false positives. For OpenPuff FLV, the 0.4% rate for false positives comes from corrupted FLV files downloaded from Archive.org.

6.2.6 Limitations of the framework

Steganographic schemes have been identified where the presented results were considered unsatisfactory. However, it was still possible to find some regularities that can be exploited in order to improve the proposed framework.

- For OpenPuff MPEG, it was possible to find patterns capable of classifying all files with great accuracy, achieving a success rate of 99.5%. This is due to the model generated in one of the folds being unable to classify just one of the test files. This file was correctly classified with the rest of the models generated in cross-validation.

  However, these results did not generalise well. The patterns detected depend, as concluded after further analysis, on particular details of the compression mechanism of the MPEG file. When tested on additional objects taken from

a different source, 20% of those objects were not matched by the model. For that reason, one might consider that the work on this steganographic scheme is not concluded yet.

In Table 37, a model with a single pattern is shown that is able to classify the remaining 80% of the provided objects correctly. Considering only these "vulnerable" objects, cover-objects always matched the pattern and had one of three different specific values that were never present in the stego-objects. Note that 25% of the "vulnerable" stego-objects were not matched by the pattern (the value *False* is returned in this case), but this situation was never found in cover-objects.

Pattern	Matches	Distribution	Prob.	Success
`01ba 4400 0400`	100%	Clean:		100%
`0401 43** ????`		`2bf8 21ff`	0.33	
`0000 01bb 000c`		`3bf8 21ff`	0.18	
`**** **** ????`		`ebf8 21ff`	0.49	
`c0c0 20e0 e0e6`	75%	Stego:		
`0000 01c0`		None of the above	0.75	
		`False`	0.25	

Table 37: Best patterns found for the OpenPuff MPEG tool, frequencies listed calculated over the 80% *of the objects that were vulnerable to the attack.*

These results can be jointly used to design a method to detect files with hidden info embedded with this scheme. Note that in this case, a signature that simply matches any stego-object is not provided. Instead the method should search the file for the string `01ba 4400 0400 0401 43** ???? 0000 01bb 000c **** **** ???? c0c0 20e0 e0e6 0000 01c0` (where the `*` can be any value). If the value extracted for this pattern is either `2bf8 21ff`, `3bf8 21ff` or `ebf8 21ff`, the MPEG is clean. If the value is different, or the pattern does not match the object, it is a stego-object produced with OpenPuff.

- For OpenPuff MP3 files, patterns were found that provided 100% accuracy but did not match all the available objects. The detected regularity was

only found in 47% of the files. This result is not good enough for us, because perfect (or near perfect) accuracy was required, as discussed in the Introduction.

For roughly half of the object pairs in the dataset, a constant string was detected that was consistently found in cover-objects but was never found in stego-objects. This string (`0000 3483 8000 004c 414d 4533 2e39`) can be used to definitely classify an object as clean, if present.

An in-depth analysis of the objects matched by this pattern showed that the steganography scheme under study systematically inserted a block of data inside the stego-object. It was detected that, for the cases where this signature is found, the cover-object had a section of data that started with `414d 4533 2e39 3155 5555` or `414d 4533 2e39 3255 5555` and was filled up to the length of 256 bytes with a series of `5555` constant byte values. The steganographic process removed the filling bytes and inserted a block of 256 bytes just after the four initial bytes (`0000 3483 8000 004c`), ending it with the rest of the signature (`414d 4533 2e39`) which was shifted exactly 256 bytes away from its original position.

Even if the framework was unable to break this steganography scheme, it provided us with useful insights as to how this flawed stego-system fundamentally operates.

- For OpenPuff WAV, the pattern extraction mechanism was unable to find any consistent patterns. Thus, it was unable to successfully perform classification. This may mean that either the given definition of pattern was insufficient to describe the kind of modifications performed by the steganography scheme, or that this scheme has no vulnerabilities that can be detected by a system attack, thus being a good candidate for a targeted statistical attack.

6.2.7 Scope and limitations of approach

Experimental results shown that the approach to steganalysis presented in this chapter was able to detect steganography via signature detection across a range

of different stego-systems and embedding algorithms. This was achieved without any prior assumptions to which kind of scheme would be used and demonstrates an effective blind attack. Although the pattern extraction mechanism could be extended using a more powerful grammar for pattern definition, the current version was enough to provide confident results over several stego-systems. However, there are stego-systems without this type of vulnerability and therefore, these methods are complementary to the statistical attacks developed for each media type. Moreover, findings by this method are specific to a given stego-system and its release. The effectiveness may change as the tool is advanced over time.

In this experimental setting, there were some practical limitations due to the computational resources available. For instance, if modifications are introduced into unexamined object areas, it would not be possible to locate them. Generally it would be optimal to limit the search to small sections both at the beginning and the end of the files. This practical limitation can be overcome by either increasing computational effort dedicated to pattern extraction, or by implementing a more efficient search strategy. Other paths can be explored in the future.

Another practical limitation is due to the fact that the framework operates at the byte-level. Certain software might only introduce bit-level changes that might not be easily recognised by the software.

Though the Naive Bayes Classifier has performed well in this setting, it has obvious limitations from a theoretical perspective if features do not show the property of conditional independence. However, domain knowledge suggests that, because the changes are largely unintentional, this assumption works for at least some of the features detected by this framework. This limitation can be overcome by learning these dependencies from the training data and constructing a more complex Bayes Network when generating the model.

Data Generation is a tricky matter as it has been shown in this Chapter. If a vulnerability is not traced back to the tool code (which is not generally available), it may happen that the vulnerability found is linked to certain characteristics of the cover file. Universality of the results, that is, whether the findings are applicable to every object generated using a given tool could be improved by increasing the number and diversity of files in the data set. This would provide the chosen method with representative training examples for a wider range of possibilities. However,

even in the case that generated data does not cover all cases (for instance, when a given signature is not present in 100% of the examples of a given tool), partial success is useful to trace the vulnerability in code.

6.3 Conclusion

This chapter has presented and tested a framework for the autonomous detection of steganography. The constructed method has shown to be particularly effective as a supporting framework for system steganalysis as the detection capabilities have proven to be optimal against weak encoding, and vulnerabilities typically associated to pre- and post-processing modifications. The use of an autonomous framework can provide useful insights into a steganalytic investigation and in cases where it may not provide optimal detection accuracy can instead support the efforts of steganalysis by identifying artifacts that may be of relevance to an investigator.

This research emphasises a key point in the field of steganography: Steganalysis should be extended beyond the scope of the embedding algorithm. A practical model for steganography must consider the impact of steganography on the whole stego-object. Any traceable features, be that from the scheme itself or from an implementation component[2], can compromise the security of the stego-system.

The vulnerabilities automatically identified by the framework were consistent with the results shown in previous chapters, in addition, this method can successfully perform system steganalysis against schemes for which no vulnerabilities had been discovered. The proposed approach can work reliably even with a limited number of training cases and it is flexible enough to scale with the computational resources available. This allows for a framework that can increase accuracy as needed based on the parameters of the practical model.

As evident by the obtained results, the presented framework can provide useful steganalytic insights, even in cases where it cannot fully break a stego-system.

[2]It has been observed empirically that many libraries for image processing leave traces of their use in the resulting images. This is, in many cases, due to old copyright messages that are no longer relevant but can still severely impair the security of the stego-system, as seen with F5 and Pixelknot.

This was shown to be useful as a reverse engineering feature. In cases where it could not consistently identify signatures, the framework was capable of identifying the exact parts of a file that were modified by the stego-system. This provides us with a fundamental understanding of how a given stego-system operates. This can be a good starting point in a practical investigation, as it provides valuable early insights that may lead to more significant findings.

There are several contributions from the research in this chapter that can be summarised through the following points:

- This work has evaluated and supported the work in previous chapters by obtaining identical results through an autonomous system
- Several embedding schemes for stego-systems have been broken by the framework for which there previously existed no method of detection
- The research in this chapter continues to support the extended scope of application for system steganalysis to video steganography by breaking the FLV feature of OpenPuff

Chapter 7

Evaluation

Steganalysis as a counterpart of steganography demonstrates a sense of duality. Formally, its primary purpose is the process of detecting the presence of steganography and proving its existence. However, steganalysis also advances the state-of-the-art in steganography by identifying weaknesses in the methodology and application of steganographic schemes. From this, steganalysis strengthens and advances the next generation of steganographic research. This impacts not only lab-based environments, but also steganography in the real world provided that there is sufficient awareness. This thesis has set out to address the latter and research the practical applications for system attacks over video steganography. In doing so, the results of this work contribute to the field of steganalysis by successfully applying system attacks to achieve practical results over video steganography and by identifying many of the existing concerns and challenges within the relevant practical state-of-the-art.

To effectively evaluate this research, it is important to provide a baseline set of standards for practical steganography that can be used to contrast the experimental and methodological results. The following are considered the four main principles for practical steganography [46].

1. The stego-system must preserve a model of the cover-object in that the resulting stego-object should appear indistinguishable from the cover-object

2. The embedding method should reflect some form of naturally occurring process so as to not introduce obvious, predictable, or arbitrary modifications

3. The stego-system should be capable of resisting known steganalytic attacks

4. The impact of embedding should be minimised to ensure that the stego-object

All of the stego-systems examined in this thesis can be broken by the efforts of system steganalysis. It can be shown that it each case where a stego-system can be detected with a high rate of accuracy, that the steganography tool fails to address at least one of the four principles for practical steganography. In chapter 4, the first case study analysing EOF injection tools demonstrates a complete disregard for at least three of the standards (2, 3, and 4) and covers a significant majority of the publicly available resources in video steganography. The second case study focusing on MV-embedding, presents a tool that makes great effort to address these four principles but fails at the third and is vulnerable to signature steganalysis as a result of the coding problem in practical steganography.

In Chapter 5, the OpenPuff tool is analysed to demonstrate the practical applications for axillary embedding. The presented system attack successfully detects OpenPuff steganography over MP4 and PDF with a high rate of accuracy (100%), once again reflecting a failure to suitably address the four principles. In this case, the detection capabilities of the attack presented in Chapter 5 specifically exploits the second principle. The OpenPuff embedding algorithm for MP4 does not reflect any naturally occurring process by embedding secret data in null atoms of flags. This appears arbitrary when considering MP4 compression standards and leads to a highly invasive and detectable steganographic implementation. As such, system attacks prove to be a highly capable detection method over this type of steganography.

System attacks as a viable method of steganalysis are highly dependant on how these four principles are addressed by stego-systems. A Lack of consideration towards addressing these points will provide a means to use system attacks and break the associated stego-system. As such, awareness and following well-established best practices in the field appears to be one of the major challenges in the field of practical video steganography.

These four principles for practical steganography have a direct impact over the experimental results obtained in this thesis. The highly capable system attacks

presented in earlier Chapters rely heavily on attacking stego-systems that fail to suitably address each point; however, these stego-systems are fully representative of the current state-of-the-art in practical video steganography and reflect many of the present and future challenges in the field, thus strengthening the position of system attacks in the domain of video steganalysis. If a stego-system can suitably address the four principles for practical steganography then the impact and effectiveness of system steganalysis will be restricted.

The research in this thesis has endeavoured to answer the two following research questions that were first presented and discussed in Chapter 1.

1. How are system attacks best used within the scope of steganalysis?
2. Can system attacks be successfully applied to achieve practical results over video steganography?

From this, the practical state-of-the-art has been examined to identify an exhaustive list of video stego-systems that are publicly available. To address the first research question, system attacks were used as a method of steganalysis to break the stego-systems listed in Chapters 4, 5, & 6. It has been shown that they are an ideal method to attack steganographic systems in the real world and can be useful in evaluating practical steganographic tools and techniques. It is also evident from this research that system attacks are best used when a stego-system fails to address any of the four key principles for practical steganography. In this case, system attacks are likely to provide optimal results and demonstrate a high rate of detection accuracy.

To address the second question, novel detection methods were constructed under the scope of system attacks. These detection methods offered a high rate of accuracy with a low rate of false positives. Video steganography was the primary focus for system attacks in this thesis and it was that in each case, each stego-system could be broken by the practices of system attacks using a representative sample of practical video steganography.

The results of this work have produced formal outputs and novel contributions by addressing the two research questions. To suitably answer each question, novel detection methods have been produced that did not previously form part

of existing relevant academic literature. These detection methods present highly effective techniques to detect steganography across a wide-range of stego-systems. Furthermore, the presence of an autonomous framework for system steganalysis can provide useful steganalytic insights for both existing and new/emerging stego-systems.

7.1 Evaluating Experimental Results

To effectively evaluate the findings of this thesis and the associated experimental results it is important to determine what is considered successful in the scope of steganalysis. As mentioned in earlier chapters, a steganography scheme is typically considered broken if a detection method is capable of determining the presence of steganography at a rate higher than that of a random guess (50%). This is a well recognised concept in academic literature and if this research is to follow this rule of thumb, the novel detection methods presented in this thesis are successful and can be used with a high degree of confidence.

However, this concept should not be deemed sufficient for successful steganalysis in the real world as it can lead to detection methods that are considered successful but yield a significantly high rate of false positives. In contrast to this, it was proposed in Chapter 3 that system attacks should offer a significantly higher rate of detection accuracy. This proved to be true for many of the detection algorithms presented in the experimental sections of this thesis (OpenPuff, OmniHide, OurSecret, BDV Datahider, etc.). The high rate of detection accuracy associated to the use of system attacks occur for several reasons.

Firstly, it is clear that many practical video stego-systems are using weak methods of steganography that can be simple to detect and a far behind the academic state-of-the-art. Secondly, the process of steganography in the real-world is highly invasive and can leave many detectable traces that are both predictable and simple to use as traceable fingerprints. Thirdly, system attacks are often overlooked in academic literature in favour of alternative methods, this lack of awareness can strengthen the scope of system attacks as new and emerging stego-systems may repeat the same mistakes.

7.1.1 Successful Implementations of System Attacks

The experimental results in Chapters 4 and 5 provided a foundation of evidence that illustrates the potential for system attacks over video. Steganalytic methods presented in each chapter demonstrated 100% detection accuracy Furthermore, the use an autonomous framework in Chapter 6 supported these findings by achieving comparable, and in many cases identical results to those observed in previous chapters. The scope of system-based vulnerabilities among stego-systems has been extended by this framework as it has achieved successful results over stego-systems and steganographic scheme not yet subjected to steganalysis (OpenPuff FLV, MPEG, DeepSound).

Table 38 provides a summary of successful system-attacks against stego-systems. In each case, detection accuracy is 100% or near (99.5%) demonstrating that these attacks can be applied to achieve practical results over a wide-range of steganography tools. The use of an autonomous framework to identify signatures for steganal-

Tool	Method	Format	Success Rate
OurSecret	EOF Data Inj.	MP4	100%
OmniHide Pro	EOF Data Inj.	MP4	100%
Masker	EOF Data Inj.	AVI	100%
BDV Data Hider	EOF Data Inj.	MP4	100%
Max File Encryption	EOF Data Inj.	MP4	100%
Steganosaurus	MV	MP4	100%
OpenPuff	Flag Replac.	MP4	100%
OpenPuff	Metadata	FLV	100%
OpenPuff	Metadata	MPEG	99.5%
OpenPuff	Metadata	PDF	100%
F5	F5 Scheme	JPEG	100%
Pixelknot	F5 Scheme	JPEG	100%
DeepSound	Unknown	WAV	99.5%

Table 38: Average Success Rate for all successful experiments.

ysis provides an ideal solution for digital forensic investigators. Signature-based detection scripts are simple to implement and have demonstrated highly practical results. In this sense, the autonomous framework can provide the foundation for building a signature database capable of detection a comprehensive and diverse

range of stego-systems. Although the primary goal was to test this over video steganography, the results have shown that this is capable of performing well over alternative data formats such as image and audio.

False positive rates provide another means to evaluate the efficiency of a constructed attack. Results that incorrectly classify cover-objects as containing a secret message will add to investigative time and increase resource demand. Because of this, it is ideal for a constructed attack (especially when considering practical methods) to maintain a low rate of false positives. Table 39 presents the false positive rates across all experiments conducted in Chapters 4, 5, and 6.

Tool	Method	Format	False Positive Rate
OurSecret	EOF Data Inj.	MP4	0%
OmniHide Pro	EOF Data Inj.	MP4	0%
Masker	EOF Data Inj.	AVI	0%
BDV Data Hider	EOF Inj.	MP4	0%
Max File Encryption	EOF Inj.	MP4	0%
Steganosaurus	MV	MP4	0%
OpenPuff	Flag Replac.	MP4	0%
OpenPuff	Metadata	FLV	0.4%
OpenPuff	Metadata	MPEG	0%
OpenPuff	Metadata	PDF	0.5%
F5	F5 Scheme	JPEG	0%
Pixelknot	F5 Scheme	JPEG	0%
DeepSound	Unknown	WAV	0%

Table 39: Tests for false positives across all experiments

The experimental results have shown that the system attacks presented in earlier chapters provide a low rate of false positives, supporting their suitability for use as viable methods of detection. Across 84.6% of the stego-systems analysed, signatures were high entropy which provided a zero rate of false positives during testing. The remaining cases reflect methods where the construction of a fingerprint might be through the concatenation of smaller signatures or patterns. As such, entropy may be lower and would be associated with a higher false positive rate.

7.1.2 Limitations

Although system attacks have shown to be highly capable of successfully performing steganalysis in the real world, there are obvious limitations in their application. As demonstrated through the attacks on OpenPuff MPEG, an autonomous framework acquired strong results; however, these did not generalise well as patterns depended on specific compression features. In any environment where there could be variants in pre- and post-processing, it is unlikely that the success rate will be consistent and in some cases, might be unsuccessful. Because of this, system attacks can be limited in their scope to stego-systems that present highly predictable patterns. Resource management should also be taken into consideration when constructing system attacks. Particularly when identifying signatures, it is understood that many of the fingerprints identified will typically be present in predictable features of a given object. Because of this, an attack can limit the number of usable features to maintain efficiency. However, if exploring an unknown stego-system, pre-existing assumptions could lead to a steganalyst to miss key features that might offer optimal results for detection. In cases, this can be overcome by increasing the available computational resources or by implementing more efficient search strategies.

Data generation has also been a challenge in the scope of this research. To ensure that any results obtained are robust, the experimental methodology and data generation phases must be unbiased. System attacks have shown to rely heavily on signatures and are often capable of being traced back to an associated stego-system (if sufficiently entropic). However, it may occur that a given-object already contains the signature. If the data generation is sufficiently representative, this insight will demonstrate that the presented attack is not strong enough and should be improved. The challenge in this sense exists when the data is not sufficiently representative, from this potential signatures may infrequently match features generated by pre- and post-processing of a given object.

Chapter 8

Conclusions and Future Work

This thesis has evaluated the scope and limitations of system attacks, with a focus on video steganography and steganalysis. The results of this research have shown that system attacks provide a practical application for steganalysis capable of achieving practical and accurate results. This work contributes to an area of steganalysis that has remained largely unexplored in academic literature.

It is important to acknowledge that steganography in the real-world should be modelled as a coding problem as well as a data hiding and detection problem. A lack of awareness into this paradigm has led to the development of many stego-systems with serious implementation vulnerabilities. In each case, these vulnerabilities markedly simplify the process of steganalysis and can provide an attack with strong signatures that offer a high rate of detection accuracy. Throughout many of the case studies presented, it was evident that weak encoding methods can lead to object-wide steganalytic attacks. A steganographic system must undoubtedly adopt strong security principles and practices to ensure that any user is safe. However, many of the stego-systems analysed throughout this thesis should now considered unsafe and unfit for use, and if this trend continues, the applicability for system attacks will continue to grow.

Video stego-systems have shown to be particularly vulnerable to system steganalysis. System attacks can be highly accurate for detecting the presence of steganography and tracing its use to any particular stego-system or tool. The work in this thesis has presented a series of novel system attacks against video

stego-systems, as case studies, to demonstrate the scope and application for system attacks as a method of practical steganalysis. The work throughout this thesis has provided several contributions that are outlined below.

- Firstly, the applications for system attacks are evaluated over video steganography and its scope has found to be wide-reaching. This work has shown that these methods can be applied to achieve practical results over video steganography and fulfil the requirement for practical steganalysis in the real-world.

- Novel steganalytic methods are constructed to detect popular and well-known video stego-systems. Each steganalytic attack presented is capable of detecting the presence of steganography with a high rate of accuracy and a low rate of false positives. This is often linked to highly entropic signatures identified for each stego-system. Furthermore, many of the attacks presented are capable of tracing steganography to its corresponding stego-system. This is a useful feature, especially for forensic practitioners and investigators.

- Insight is given to better understand the gap between steganography in the laboratory and steganography in the real-world, with a particular focus on video driven steganography. The scope of this endeavour involved using a series of case-studies to demonstrate how steganographic implementations ignore best practice and, by doing so, endanger its users. In light of this, system attacks become a desirable method for steganalysis, where it would be far more practical to detect implementation vulnerabilities of the stego-system as opposed to alternative methods.

8.1 Challenges for System Steganalysis

System attacks have shown to be an effective method of steganalysis against practical stego-systems. However, it cannot be claimed that as a method of steganalysis it is viable for each and every scenario.

The paradigm for practical steganography suggests that the highly variable nature of the real-world makes it challenging to incorporate steganalytic concepts

formulated in a laboratory environment. In cases where this happens (as seen throughout this thesis), a stego-system is defeated by what is initially deemed to be a requirement for the embedding algorithm to function (i.e., converting a file to the frequency domain for DCT steganography). This leads to the creation of steganalytic attacks against features seemingly unrelated to the steganographic method. From this, implementation vulnerabilities are largely identified by the impact of pre- and post-processing changes made by the stego-system, these changes are often performed to prepare a file for secret message embedding, but the process ultimately modifies key features in a predictable fashion. Raising awareness to fully understand this problem could impact the scope of system steganalysis.

It was noted in Chapter 3 that when predictable changes occur, it can be due to the use of publicly available libraries being used in a privacy aware environment. In these cases, the libraries are not developed with security and privacy in mind, especially for the purposes of steganography. A secure vetting of reliable libraries for steganography tools would be a highly recommended and valuable addition to the development of practical stego-systems. This would also reduce potential applications for system attacks. However, it is apparent that for as long as steganography in the real-world is to be considered a coding problem, there will always be potential for system attacks.

8.2 Future Work

Steganography and steganalysis are continuously evolving fields of research. Advancements in steganography often inspire the development of new stego-systems and from this, there will always be a need for a diverse range of research in steganalysis. Going forward, there are many different avenues to consider for future work. The steganalytic attacks presented in this thesis are being consolidated into a framework that is being used to determine the presence of steganography throughout the Internet under the EPSRC project EP/N024192/1 (SEEK).

The work in this thesis has explored the applications for system attacks over video media, an area that was largely unexplored. However, there is significant potential for continued research in system steganalysis. It is important to raise awareness into the impact of system attacks and improve many of the poor security

practices that occur in practical steganography. For this reason, it would be desirable to explore alternative methods of steganography that show growing interest and complexity. Ideal avenues of research can include: document, executable files, game-based, and network (TCP/IP)-based methods.

Chapter 6 presented a framework to automate signature detection based on implementation vulnerabilities of stego-systems. As noted in the detailed discussion of each scheme, the framework is able to obtain a series of patterns that can be joined in a successful model for classification. However, a specific method for selecting the optimal patterns in the model has not been implemented. This step is not straightforward, as every pattern can be characterized by a set of possibly conflicting properties: for instance, as previously mentioned, the number of matches and the success rate in matched files. Other features may be of relevance for further applications: patterns with low entropy will be more prone to false positives; long patterns (that is, patterns with long constant parts) generally are less subject to false positives but have lower generalisation abilities. These aspects can be evaluated using a multi-criteria approach that can be used for model refining or even during the model construction step, to ensure that the resulting classifier uses the best possible combination of patterns. This is another possible avenue for future work.

Appendices

Appendix A

Proof-of-Concept Scripts

A.1 Steganosaurus - Atom Detection

```
#!/bin/bash
for file in /home/ts424/Desktop/SeekSets/*
do
mp4file --dump "${file}" > /tmp/mp4dumpfile
sed '6,7!d' /tmp/mp4dumpfile >/tmp/tmp1

for line in $(< /tmp/tmp1); do
echo $line | sed 's/.*(\(.*\))/\1/'>>/tmp/tmp2
done

if egrep -q "free|mdat" /tmp/tmp2; then
source /home/ts424/Desktop/Seek/Steganosaurus/SaurusSig.sh
else
echo -e \ "${file}" "Atom Test - No Steganography found" >>
    /home/user/Results/Negatives;
fi

done
```

A.2 Steganosaurus - Signature Detection

```
#!/bin/bash

xxd -p "${file}" > /tmp/tmp1
tail -c 40 < /tmp/tmp1 | tr -d '\n' > /tmp/tmp2
if grep -q -c "0000004c61766635342e35392e313036" /tmp/tmp2; then
echo -e \ "${file}" "Signature & Atom Test - Steganosaurus content
    detected" >> /home/user/Results/SteganosaurusResults;
else
echo -e \ "${file}" "Signature & Atom Test - No Steganography found" >>
    /home/user/Results/Negatives;
fi

rm /tmp/tmp1
rm /tmp/tmp2

touch /tmp/tmp1
touch /tmp/tmp2
```

A.3 OurSecret - Signature Detection

```
#!/bin/bash

xxd -p "${file}" > /tmp/tmp1
tr -d '\n' < /tmp/tmp1 > /tmp/tmp2
if grep -c -q
    -"9e97ba2a008088c9a370975ba2e499b8c178720f88dddc342b4e7d317fb5e87039a8b84275687191"
    /tmp/tmp2; then
echo -e \ "Signature test - OurSecret signature detected in" "${file}"
    >> /home/user/Results/OurSecretResults
else
echo -e \ "Signature test - No OurSecret signature found " "${file}" >>
    /home/user/Results/Negatives

fi

rm /tmp/tmp1
rm /tmp/tmp2

touch /tmp/tmp1
touch /tmp/tmp2
```

A.4 OmniHide Pro - Signature Detection

```
#!/bin/bash

xxd -p "${file}" > /tmp/tmp1
tr -d '\n' < /tmp/tmp1 > /tmp/tmp2
tac /tmp/tmp2 > /tmp/tmp3
if grep -q -E -m1 \(20\)\{140,\} /tmp/tmp3; then
echo -e \ "Signature test - OmniHide signature found" "${file}" >>
    /home/user/Results/OmniHidePResults
else
echo -e \ "No Steganography Detected" "${file}" >>
    /home/user/Results/Negatives
fi

echo "${file}"

rm /tmp/tmp1
rm /tmp/tmp2
rm /tmp/tmp3

touch /tmp/tmp1
touch /tmp/tmp2
touch /tmp/tmp3
```

A.5 BDV DataHider - Signature Detection

```
content...#!/bin/bash

xxd -p "${file}" > /tmp/tmp1
tr -d '\n' < /tmp/tmp1 > /tmp/tmp2
if grep -c -q "fbead881250ff9" /tmp/tmp2; then
echo -e \ "Signature test - BDV signature detected " "${file}" >>
    /home/user/Results/BDVResults
else
echo -e \ "Signature test - No BDV signature found " "${file}" >>
    /home/user/Results/Negatives

fi

rm /tmp/tmp1
rm /tmp/tmp2

touch /tmp/tmp2
touch /tmp/tmp2
```

A.6 Generalised EOF Scanner

```
#!/bin/bash

catch="invalid atom size, extends outside parent atom"

mp4file --dump "${file}" > /tmp/tmp1

test=$(grep -o "invalid atom size, extends outside parent atom"
    /tmp/tmp1)
echo $test >/tmp/tmp2

if [[ $test == $catch ]]
then
echo -e \ "${file}" "EOF Test - EOF injection detected! We recommend
    running signature tests" >>/home/user/Results/EOFResults;
else
echo -e \ "${file}" "EOF Test - No steganography detected"
    >>/home/user/Results/Negatives

fi

rm /tmp/tmp1
rm /tmp/tmp2

touch /tmp/tmp1
touch /tmp/tmp2
```

A.7 OpenPuff - Flag Detection

```
#!/bin/bash

IFS=$'\n'
for line in $(< /tmp/tmp1); do

echo $line | sed 's/.*(\(.*\))/\1/' >>/tmp/tmp2

done

for line in $(< /tmp/tmp2); do
printf "%d\n" $line >> /tmp/tmp3
done
pvar=0
nvar=0
IFS=$'\n'
for line in $(< /tmp/tmp3); do

result=$(echo $line)
if [[ $result -gt 0 ]]; then
pvar=$((pvar+1))
else
nvar=$((nvar+1))
echo -n "" >/tmp/tmp2
echo -n "" >/tmp/tmp3
fi
done
echo $pvar
echo $nvar
if [[ $pvar -gt $nvar ]]; then

echo -e \ "${file}" "Flag test - Modified flags, OpenPuff Steganography
    detected " $pvar $nvar >> /home/user/Results/OpenPuffResults
else
```

```
echo -e \ "${file}" "Flag Test - No modified flags found" $pvar $nvar
    >> /home/user/Results/Negatives
pvar=0
nvar=0
fi
```

A.8 OpenPuff - PDF Steganography Detection

```
import re
import sys
import os
import glob

for filename in glob.glob('*.pdf'):

with open(filename, "rb") as f:


stream=f.read()
f10 = re.findall( b'\x65\x6E\x64\x73\x74\x72\x65\x61\x6D\x0D', stream)
    #Counting "endstream+CR"
occurf10=len(f10)
f11 = re.findall( b'\x65\x6E\x64\x73\x74\x72\x65\x61\x6D\x0A', stream)
    #Counting "endstream+LF"
occurf11=len(f11)
f20 = re.findall( b'\x65\x6e\x64\x6F\x62\x6A\x0D', stream) #Counting
    "endobj+CR"
occurf20=len(f20)
f21 = re.findall( b'\x65\x6e\x64\x6F\x62\x6A\x0A', stream) #Counting
    "endobj+LF"
occurf21=len(f21)
f30 = re.findall( b'\x20\x6F\x62\x6A\x0D', stream) #Counting " obj+CR"
occurf30=len(f30)
f31 = re.findall( b'\x20\x6F\x62\x6A\x0A', stream) #Counting " obj+LF"
occurf31=len(f31)

hidden_contents=((occurf10*occurf11)+(occurf20*occurf21)+(occurf30*occurf31))>0

if hidden_contents:
print "This PDF file has contents hidden with OpenPuff"
maxhiddensize=0.2204*occurf11+0.2117*occurf21+0.2115*occurf31-194.6563
```

```
print "Our estimate of the maximum size of the embedded data is",
    maxhiddensize, "bytes"
else:
print "This PDF file does not have contents hidden with OpenPuff"
if ((occurf10*occurf20)==0):
print "It looks like a scanned or a compressed file"
```

Appendix B

Overview and Description of the Framework

Work throughout this thesis has shown that, quite often, the steganography process introduces predictable changes in stego-objects when using practical stego-systems. In this work, modifications are identified using a method that is independent of the exact tool or media format. The results obtained are related to the particular tool used, though sometimes several tools can be affected by the same vulnerability, due to the use of common libraries. These changes extend beyond the unavoidable impact upon the signal itself, which is the subject of classical statistical steganalysis. These implementation issues are generally unintentional modifications that can be, however, very useful to distinguish between cover objects and stego-objects.

Implementation issues of this type can provide an analyst with a number of options for developing an effective steganalysis framework. Signatures which can be seen as consistent fingerprints from the stego-system are a classic candidate for these attacks. Signatures can be unique to the stego-system and appear solely in the stego-object. In an ideal scenario, the signature will consist of strings of enough length and entropy so that perfect accuracy (in terms of zero false positives and negatives) can be achieved [25], although of course this is not always the case.

B.0.1 Concepts and definitions regarding attribute extraction

In this section a specific example is used to describe some of the terms employed later when describing the procedure itself and all of its steps.

The objective is then the construction of a classifier able to separate cover objects from stego-objects. As in any machine learning technique, objects have to be described in terms of features, also called attributes. For instance, the existence of a given signature in an object would map to an attribute that has two possible values, *True* (if the signature is present) or *False* (otherwise). A supervised classification system is able to learn the correlation between the existence of those features and each of the object classes, if provided with adequate training data. This training data would incorporate the class of each of the objects (labelled data). The information the classifier learns can be called a "model". The final objective is twofold: to build a model that represents the information available for training, but also that successfully predicts the correct class for any previously unseen object, such as those found in a real-world scenario. This is usually called "generalisation".

Without any assumption on where potential signatures will be located in the object, attribute generation has to be modelled as a search for modifications produced by the steganographic process. This search can be performed by comparing a cover-object with its corresponding stego-object. Then, the system has to determine if the change is consistent throughout a significant part of the available labelled data. A classifier can only use these "consistent features", because they are the ones that represent information not particular to single pairs of cover and stego-objects. For the same reason, those attributes are likely to be the most useful to construct a model able to generalize, that is, a model able to successfully distinguish between stego- and cover-objects in the most generalizable way.

In this work, a simple grammar is used, shown in Table 40, to identify potential attributes of the classification model. With it, three different pattern types can be generated:

```
<pattern> ::= <cnst> [ <var> | { <mconst> } ] [ <cnst> ]
<mcnst> ::= <var> <cnst> <var>
<var> ::= <skip> [ <val> ] | <val> [ <skip> ]
<cnst> ::= <hex> { <hex> }
<skip> ::= "*" { "*" }
<val> ::= "?" { "?" }
<hex> ::= [af_09]
```

Table 40: Grammar used for the pattern generation, in EBNF notation.

- Fixed patterns. Those are strings that can be found in cover objects but missing in stego-objects or, more frequently, the other way around. They generally correspond to some piece of information injected or deleted by the steganographic process. Those patterns become attributes, labelled by the value of the pattern itself, 0012-6e4d for instance, and take binary values, True or False.

- Variable patterns. They are strings in which a particular sub-string is constantly changed by the same value or values. For instance, pattern `0012????0000????6e4d6574` describes one where the variable part ???? is systematically changed for the same value or values, but the remaining 8 bytes stay constant (`0012`, `0000`, `6e4d6574`).

  Variable patterns become attributes labelled with the constant part. For instance, if `0012abcd000012346e4d6574` and `001212340000abcd6e4d6574` are both found, a new attribute called `0012????0000????6e4d6574` is included, where the only values permitted are: `abcd-1234`, `1234-abcd` and `Void`.

- Extended patterns. Those patterns correspond with situations where differences between the cover and stego-objects include a fixed part, a variable part that changes always to the same value or values, and a variable part that takes arbitrary values. For instance, the pattern `0012????****????6e4d6574` is similar to the previous example, but without taking into account the value of the third byte.

The search for patterns is performed by comparison of each pair of cover- and

stego-objects, using the aforementioned grammar as reference for identifying and generating attributes.

At the end of this process of attribute discovery, the entire object dataset is replaced by an attribute-value table, where each row corresponds to an object, stego or cover, each column to an attribute, and the cells are filled with the values of each attribute for each object.

In Table 41, a learning set with 30 objects (15 stego and 15 cover) and 3 attributes is shown. For instance, column 21 corresponds to an object where the string "6e4d6574" is not present, string "00123b2a00009218" is, and "00122dfb****e05d" is also present, no matter the value assigned to the asterisks. This table constitutes the training set for the naive-bayes algorithm, which is used in the next step to distinguish between cover- and stego-objects. In the next section, a more detailed description is introduced about how to built the training set.

Attributes Objects	6e4d-6574	0012-???? 0000-???? 6e4d-6574	0012-???? ****-???? 6e4d-6574
VideoS 1	True	Void	Void
VideoC 1	False	Void	Void
VideoS 2	True	Void	Void
VideoC 2	False	Void	Void
VideoS 3	True	Void	Void
VideoC 3	False	Void	Void
VideoS 4	True	Void	Void
VideoC 4	False	Void	Void
VideoS 5	False	Void	Void
VideoC 5	False	Void	Void
VideoS 6	False	3b2a-9218	Void
VideoC 6	False	Void	Void
VideoS 7	False	3b2a-9218	Void
VideoC 7	False	Void	Void
VideoS 8	False	3b2a-9218	Void
VideoC 8	False	Void	Void
VideoS 9	False	2dfb-e05d	Void
VideoC 9	False	Void	Void
VideoS 10	False	2dfb-e05d	Void
VideoC 10	False	Void	Void
VideoS 11	False	3b2a-9218	2dfb-e05d
VideoC 11	False	Void	Void
VideoS 12	False	3b2a-9218	3b2a-9218
VideoC 12	False	Void	Void
VideoS 13	False	3b2a-9218	2dfb-e05d
VideoC 13	False	Void	Void
VideoS 14	False	2dfb-e05d	3b2a-9218
VideoC 14	False	Void	Void
VideoS 15	False	2dfb-e05d	3b2a-9218
VideoC 15	False	Void	Void

Table 41: Example of a learning table.

B.0.2 Model construction from extracted features

The overall process proposed in this work proceeds in five steps:

1. Extract patterns from a set of paired objects (cover and stego), each one

labelled with the class it belongs to (as depicted in Figure 48).

2. Generalize patterns extracted in the previous step, and add the resulting extended patterns to the repository.

3. Build a raw model from all patterns by searching on the whole training set (Figure 49).

4. Refine model using predefined rules.

5. Validate model to obtain the success rate on objects whose class is unknown

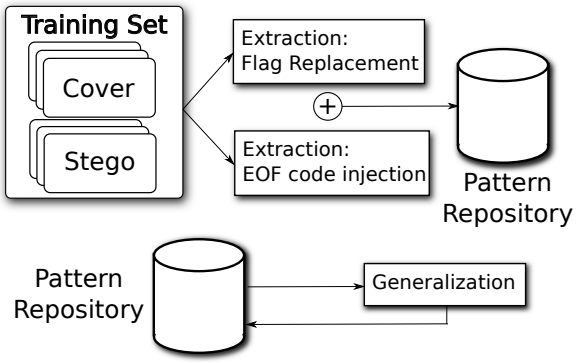

Figure 48: Block diagram of the process, Steps 1 and 2

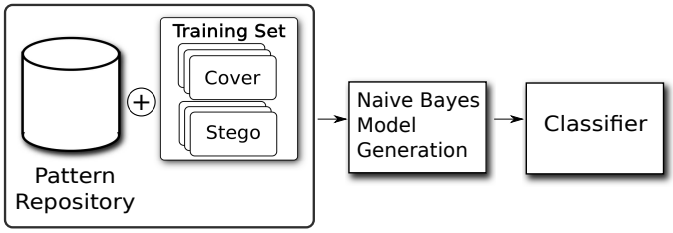

Figure 49: Block diagram of the process, Step 3

Note that training requires having paired objects, that is, both a cover-object and the corresponding stego-object, where a steganographic scheme has embedded a message of unknown length. This set of objects is what can be referred to as training set.

In the experimental procedure, a test set is also used, choosing randomly a fraction from the training set. The use of a separate test set is a common practice for parameter tuning and to stop over fitting [59]. The experiments were evaluated using k-fold cross-validation on the third group of objects, the validation set, so the split among train, test, and validation sets is performed 10 times.

In the following sections, detail is given for each of the aforementioned steps in the process.

B.0.3 Pattern extraction and pattern generalization

B.0.3.1 Step 1: Pattern extraction

This phase processes files from the training set to extract patterns that can be used to construct a model. These selective patterns have to be searched using a predefined criteria so they represent the modifications that a stego-system imprints onto the files that represent both cover-objects (called cover-files) and stego-objects (called stego-files) through its embedding scheme.

In these experiments, a search was initially conducted for two types of patterns, based on previous knowledge of two typical steganography techniques, namely Flag Replacement and End-of-File code injection.

- Patterns for Flag Replacement are extracted comparing a cover-file with its corresponding stego-file. The search finds consistent sequences of bytes located in parallel between both files.

  In Table 42, the extraction process is described for one of these patterns. In this example, an object pair is explored and it is shown that, at a given position, 11 bytes can be identified that had the same value in both files, then, 3 bytes are shown that were different in both files, followed by 4 bytes that had again the same value in both.

- Patterns for EOF Injection are sequences of bytes that are found anywhere (but frequently at the end) in a stego-file, but not in the corresponding cover-file. In this case, only constant patterns are returned, that is, patterns with no variable part.

File [Pos.]	Hexadecimal dump in file order
Cover [0xa]	`0000001200017b0000000000000002000a6f`
Stego [0xa]	`0000001200017b0000000004101002000a6f`
Pattern	`0000001200017b000000????????02000a6f`

Table 42: Example pattern for flag replacement.

An example of these patterns can be found in Table 43. The final portion of the cover-file is repeated in the stego-file, but there are some additional contents present in the stego-file. A section of these contents are selected (`2a50523131306d622020` in the example) and store it if that fragment is not found at any position of the cover-file.

File	Hexadecimal dump in file order
Cover:	`777bbbf0 EOF`
Stego:	`777bbbf0 2a50523131306d622020... EOF`
Pattern:	`2a50523131306d622020`

Table 43: Example pattern for EOF code injection.

B.0.3.2 Step 2: Pattern generalization

The result of the pattern extraction process is a pattern repository. On this repository, a generalization process can be run that introduces an additional number of extended patterns.

This mechanism was introduced to account for the fact that some object pairs had specific parts that were being introduced in the patterns, when considered those pairs individually, but were never repeated in other data. For instance, the file size or file name might be present in that section of the files, and would be consistent in cover and stego-objects, but would never (or rarely) be repeated in any other object. So those patters were useful, but had to be generalized to match more than one pair of cover and stego-objects.

This process is performed as follows:

1. Group the patterns in clusters, according to similarity

2. For each cluster formed that contains at least two patterns, create a generalized pattern by replacing each position that is not identical in all patterns with the (*) sign

The clustering method, hierarchical agglomerative clustering technique (HAC) with single linkage [80] is used, employing the Hamming distance as a measure of similarity between patterns. When comparing two patterns, their Hamming distance was computed as the ratio of symbols that are different for each position in the pattern. That is, two patterns where all positions have the same symbols have distance 0.0; if a third of the symbols are different, then distance is approximately 0.33 and so on. The patterns that were closer in terms of Hamming distance were clustered together, as long as their distance was not above a certain threshold "Th". The distance function assigned distance 1.0 to patterns of different length, thus ensuring that the cluster mechanism never placed those patterns in the same group.

As seen in Table 44, new patterns are obtained where some of the bytes are to be ignored. This step is relevant if the original patterns extracted when comparing a cover-object and its corresponding stego-object are specific only to that pair. Extended patterns are guaranteed to match at least two pairs of objects, and have much improved chances of matching new objects not included in the model generation phase.

Patterns	Hexadecimal representation
1	0000001200017b000000????????02000a6f
2	0000001200027b000000????????02000a6f
3	000000120ffefb000000a5??????02000a6f
Extended	000000120****b000000**??????02000a6f

Table 44: Example of the Pattern Generalization process. The repository contains three variable patterns, and single extended pattern is generated, abstracting all

B.0.3.3 Step 3: Model generation

Model generation consists in the application of every pattern to all object pairs in the training set, and the computation of two distributions of values for each

pattern, one for the cover-object and one for the stego-object. This is done by calculating the frequencies for the values obtained for each pattern, as described previously.

An example of model generation is presented in Table 45. The first section in Table 45 corresponds to the pattern in Table 42. It matched 90% of the cover-objects and also 90% of the stego-objects.

Pattern 1: `0000001200017b000000????????02000a6f`			
	Matches	Value	Frequency
Cover ($Class_0$)	90%	`0000 0000`	0.90
		`False`	0.10
Stego ($Class_1$)	90%	`0000 0000`	0.18
		`0004 1010`	0.01
		`... others ..`	...
		`False`	0.10
Pattern 2: `2a50523131306d622020`			
	Matches	Value	Frequency
Cover ($Class_0$)	0%	`False`	1.0
		`True`	0.0
Stego ($Class_1$)	75%	`False`	0.25
		`True`	0.75

Table 45: Model constructed from the example patterns

The value "False" is included for the cases where the pattern did not match. For the cases that the pattern matched, the frequencies of the values are obtained located among the training set. The result in this example was:

- For cover-objects the value was `0000 0000`.
- For stego-objects, the values were `0000 0000` in 18% of cases, and different from `0000 0000` (possibly random values) in 82% of cases. This may mean that a generally unused "flag" position was frequently (but not always) modified by the embedding scheme.

The second pattern in Table 45, corresponds to the example pattern shown in Table 43. As the pattern has only a constant part, only the frequently matched

files of both classes are computed. For this example, it can be assumed that it was never identified within cover-objects, and it was found in 75% of the stego-objects. Conventionally, the value "False" can be assigned when a pattern does not match, and "True" when it does.

These frequencies are then stored as CPTs for the model, and used for classification using the Maximum Likelihood rule (Eq. 9). In Appendix C, details are given into how this would be applied to a specific case using the current example.

B.0.3.4 Step 4: Model refining

After model construction, but before classification, filtering can be applied to eliminate patterns that are certainly not useful for classification. This pruning process consists in applying the following rules:

- Eliminate patterns that only match files in one file pair (the file pair from which they were obtained)
- Eliminate patterns whose distribution of values is the same for both cases (cover and stego).
- Eliminate patterns whose distribution of values is thought to be approximately random for both cases (cover and stego).

Also, the following two policies were used to modify the CPTs in the model, to address some practical issues in this process of model construction:

- The empirical distribution may not be complete, in the sense that some of the values for a feature may not be found in any of the cases used for training. This is usually called the problem of zero-observation (that is, a possible value is never observed when training). In this case, the simplest strategy used in literature is adopted, that is, assigning a fixed small value to the factor corresponding to unknown values. This ensures that the joint probability product does not become 0 due to unknown values [63].
- The CPT associated to a feature was transformed when the CPT for a feature had a single value V_0 with frequency 1.0 in one case (either cover or stego).

In these cases, the CPT was considered binary, that is, with only two cases: $Pr(V_0|Class_k)$ and $Pr(\neg V_0|Class_k)$, where $\neg V_0$ means "any value different from V_0"

These subtle changes to the general process generate what can be defined as the Refined Model, that is the final result of training and is returned as the solution to the steganalysis problem at hand.

B.0.3.5 Step 5: Model validation

This step simply classifies a set of objects that was reserved from the original labelled data, and reports the results in terms of classification success rate. This is the expected success rate for the classifier on new data that might be presented in the future.

Appendix C

Example of Classification using a Naive Bayes Model

Let us assume a case with the example model in Table 45 is being tested. For each pattern in the model, two factors are obtained. One is the (estimated) likelihood for cover-objects, and the other for the stego-objects. These products of likelihoods are used as in Eq. 9 to obtain the class.

Using this example, Pattern 1 would first be tested. If the pattern does not match the file, the value can be considered as "False". If it matches, then its value can be examined for that case, and go to the proper part of the Model in Table 45 to find the factors to be used in Eq. 9:

- If the value is `0000 0000`, the following values are used:

$$Pr(\texttt{0000 0000}|Class_0)=0.9$$
$$Pr(\texttt{0000 0000}|Class_1)=0.18$$

- If the value is different than `0000 0000`, then the following is used:

$$Pr(\neg(\texttt{0000 0000})|Class_0)=0.1$$
$$Pr(\neg(\texttt{0000 0000})|Class_1)=1-0.18=0.82$$

, where ¬(0000 0000) means that the value is **different** from 0000 0000.

Then you would proceed pattern 2, where it is determined whether the pattern matches the file ($True$) or not ($False$).

- If the pattern matches the file, $Pr(True|Class_0) = 0.0$ is used [1] and $Pr(True|Class_1) = 1.0$.
- However, if the pattern does not match the file, $Pr(False|Class_0) = 1.0$ and $Pr(False|Class_1) = 0.25$ is used.

So let's say that a file has 0000 0000 for the first pattern and the second pattern **matched** this file; then one would calculate:

- For $Class_0$,

$$
\begin{aligned}
&Pr(\texttt{0000 0000}|Class_0) = \\
&\quad = Pr_1(\texttt{0000 0000}|Class_0) \times Pr_2(True|Class_0) = \\
&\quad = 0.9 \times 0.0 = 0.0 \\
&Pr(Class_0|\texttt{0000 0000}) = \\
&\quad = \alpha \times Pr(Class_0) \times Pr(\texttt{0000 0000}|Class_0) \\
&\quad = \alpha \times 0.5 \times Pr(\texttt{0000 0000}|Class_0) \\
&\quad = \alpha \times 0.5 \times 0.0 = 0
\end{aligned}
$$

[1] Usual practice in these classifiers is to *smooth* values that are equal to 0, because any factor whose value is 0.0 would make the rest of the patterns useless for classification. The simplest method found in literature is to replace 0.0 for a small, non-zero value [63]

- For $Class_1$,

$$
\begin{aligned}
&Pr(\texttt{0000 0000}|Class_1) = \\
&\quad = Pr_1(\texttt{0000 0000}|Class_1) \times Pr_2(True|Class_1) = \\
&\quad = 0.18 \times 0.75 = 0.135 \\
&Pr(Class_1|\texttt{0000 0000}) = \\
&\quad = \alpha \times Pr(Class_1) \times Pr(\texttt{0000 0000}|Class_1) \\
&\quad = \alpha \times 0.5 \times Pr(\texttt{0000 0000}|Class_1) \\
&\quad = \alpha \times 0.5 \times 0.135 > 0
\end{aligned}
$$

Because training is being carried out with balanced classes, the values of $P(Class_0)$ and $P(Class_1)$ are both 0.5 and the factor α is the same for both. Thus, instead of calculating the value for the posterior probability ($P(Class_0|\texttt{0000 0000})$), it is possible to simply compare the likelihoods (0 and 0.135) and conclude that the file should be classified as $Class_1$, the class that has the greater likelihood.

Appendix D

Kent Digital Media Archive

Internet access to digital media resources, although ubiquitous are often subject to strict policies regarding user download and distribution. These resources are usually managed by copyright infringement policies or the host servers place strict guidelines upon these media files to limit or block user access with techniques such as digital watermarks, anti-crawling and many more. While the benefits of this can be self-evident, easy access to a large archive of varying media files for research purposes is a common request amongst academics and researchers, especially for those in the domain of cyber-security, digital forensics, and machine learning, or content based retrieval where authors will often have to use their own datasets or do not specify the source [131]. In this sense, files can be used for testing a various avenues of research, such as steganography, digital content forensics, compression, and content-based image retrieval (CBIR). This is also inclusive to research and development through image and video application benchmarking. This can be a challenge to carry out due to availability of datasets for testing [130], this is especially relevant in cases where a significantly large number of datasets are required for analysis and testing over multiple frameworks. Availability of files for these purposes are scarce and under the circumstances that they are freely accessible with no strings attached, download policies usually restrict or limit access to large numbers of these files, often through Captcha technology.

Having identified a significant gap where media files are needed for large-scale testing and benchmarking, this project has endeavoured to build an archive that

provides an open source multimedia repository available to all for research and academic purposes. Although use of these files will be applicable to a large number of fields and disciplines, emphasis will be placed upon research in cyber-security, digital forensic, and benchmarking. As such, all content and datasets created for the archive will be created by the project authors and additional participants using state of the art devices such as cameras, smartphones and tablets.

The foundation of this archive is built around an extensive amount of diverse multimedia content, consisting of videos, images, and in the future, audio. This spans a range of file formats such as MP4, AVI, M4V, MOD, and JPEG. All content produced for the archive is freely available and easily accessible with a current archive size of 25,000 images, and 2,000 video files.

Bibliography

[1] Acharya, U. D., Kamath, P. R. et al. (2013). A secure and high capacity image steganography technique. *arXiv preprint arXiv:13043629*.

[2] Adelson, E. H. (1990). Digital signal encoding and decoding apparatus. US Patent 4,939,515.

[3] Adonis (2007). Cve details - securekit. `http://www.cvedetails.com/cve/CVE-2007-0163`, [Accessed 15 August 2018].

[4] Alís, J. B. A. (2012). *Information leakage and steganography: detecting and blocking covert channels*. Ph.D. thesis, Universidad Carlos III de Madrid.

[5] Alizadeh-Fahimeh, F. et al. (2012). Using steganography to hide messages inside pdf files. *SSN Project Report*.

[6] Altwaijry, H. and Algarny, S. (2012). Bayesian based intrusion detection system. *Journal of King Saud University - Computer and Information Sciences*, 24(1), pp. 1 – 6.

[7] Aly, H. A. (2011). Data hiding in motion vectors of compressed video based on their associated prediction error. *IEEE Transactions on Information Forensics and Security*, 6(1), pp. 14–18.

[8] Anguita, D. et al. (2012). The'k'in k-fold cross validation. In *ESANN*.

[9] AOL (2015). Dmoz open-content directory. `https://www.dmoz-odp.org//`, [Accessed 15 August 2018].

[10] Apple (2014). *Overview of Movie Atoms - Quicktime File Format Specification*. [Accessed 15 August 2018].

[11] Bai, J. and Chang, C.-C. (2016). A high payload steganographic scheme for compressed images with hamming code. *Int J Netw Secur*, 18(6), pp. 1122–1129.

[12] Baker, W. et al. (2011). 2011 data breach investigations report. *Verizon RISK Team, Available: www verizonbusiness com/resources/reports/rp_databreach-investigationsreport-2011_en_xg pdf*, pp. 1–72.

[13] Bas, P., Filler, T. and Pevnỳ, T. (2011). break our steganographic system: The ins and outs of organizing boss. In *International Workshop on Information Hiding*, Springer, pp. 59–70.

[14] Bell, G. and Lee, Y.-K. (2010). A method for automatic identification of signatures of steganography software. *IEEE Transactions on Information Forensics and Security*, 5(2), pp. 354–358.

[15] Bender, W. et al. (1996). Techniques for data hiding. *IBM systems journal*, 35(3.4), pp. 313–336.

[16] Bender, W. et al. (2000). Applications for data hiding. *IBM systems journal*, 39(3.4), pp. 547–568.

[17] Böhme, R. (2010). *Advanced statistical steganalysis*. Springer Science & Business Media.

[18] Böhme, R. (2010). Principles of modern steganography and steganalysis. *Advanced Statistical Steganalysis*, pp. 11–77.

[19] Borse, D. and Patil, S. (2015). Review and analysis of multifarious spatial domain steganography techniques. *International Journal of Engineering Research & Technology (IJERT) ISSN: 2278*, 181.

[20] Cachin, C. (1998). An information-theoretic model for steganography. In *International Workshop on Information Hiding*, Springer, pp. 306–318.

[21] Cantrell, G. and Dampier, D. D. (2004). Experiments in hiding data inside the file structure of common office documents: a stegonography application. In *Proceedings of the 2004 international Symposium on information and Communication Technologies*, Trinity College Dublin, pp. 146–151.

[22] Cao, Y., Zhao, X. and Feng, D. (2012). Video steganalysis exploiting motion vector reversion-based features. *Signal Processing Letters, IEEE*, 19(1), pp. 35–38.

[23] Cao, Y. et al. (2011). Video steganography with perturbed motion estimation. In *International Workshop on Information Hiding*, Springer, pp. 193–207.

[24] Chae, J. J. and Manjunath, B. (1999). Data hiding in video. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 1, IEEE, pp. 311–315.

[25] Chen, M. et al. (2006). Analysis of current steganography tools: classifications & features. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, IEEE, pp. 384–387.

[26] Chiew, K. L. (2011). *Steganalysis of Binary Images*. Macquarie University.

[27] Choudhury, B., Das, R. and Baruah, A. (2015). A novel steganalysis method based on histogram analysis. In *Advanced Computer and Communication Engineering Technology*, Springer, pp. 779–789.

[28] Chuntian, S. Y. W. L. Z. (2007). New video steganalysis algorithm for msu stego video. *Journal of Southeast University (Natural Science Edition)*, p. S1.

[29] Council, W. S. C. (2018). Halnaker windmill west sussex. `https://www.westsussex.gov.uk/leisure-recreation-and-community/places-to-visit-and-explore/halnaker-windmill/`, [Accessed 16 August 2018].

[30] Cox, I. J. and Miller, M. L. (1997). Review of watermarking and the importance of perceptual modeling. In *Electronic Imaging'97*, International Society for Optics and Photonics, pp. 92–99.

[31] Cox, I. J. et al. (1997). Secure spread spectrum watermarking for multimedia. *IEEE transactions on image processing*, 6(12), pp. 1673–1687.

[32] Crandall, R. (1998). Some notes on steganography. *Posted on steganography mailing list*.

[33] Cruz, J. P., Libatique, N. J. and Tangonan, G. (2012). Steganography and data hiding in flash video (flv). In *TENCON 2012-2012 IEEE Region 10 Conference*, IEEE, pp. 1–6.

[34] Dai, Y., Zhang, L. and Yang, Y. (2003). A new method of mpeg video watermarking technology. In *Communication Technology Proceedings, 2003. ICCT 2003. International Conference on*, vol. 2, IEEE, pp. 1845–1847.

[35] Datta, B., Mukherjee, U. and Bandyopadhyay, S. K. (2016). Lsb layer independent robust steganography using binary addition. *Procedia Computer Science*, 85, pp. 425–432.

[36] Devarakonda, N. et al. (2012). Intrusion detection system using bayesian network and hidden markov model. *Procedia Technology*, 4, pp. 506 – 514.

[37] Dickman, S. D. (2007). An overview of steganography. *Department of Computer Science, James Madison University Infosec Techreport*.

[38] Dixon, R. C. (1994). *Spread spectrum systems: with commercial applications*. John Wiley & Sons, Inc.

[39] Dmitriy Vatolin, O. P. (2007). MSU stegovideo. *http://googl/XqiWi1*.

[40] Domingos, P. and Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Machine Learning*, Morgan Kaufmann, pp. 105–112.

[41] Dongardive, P., Gupta, N. and Barde, C. (2015). Improved security color grace steganography with grace to text encoding and lsb. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, IEEE, pp. 1–6.

[42] Duda, R. O., Hart, P. E. and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

[43] Fang, D.-Y. and Chang, L.-W. (2006). Data hiding for digital video with phase of motion vector. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, IEEE, pp. 4–pp.

[44] Fillder, T., Pevny, T. and Bas, P. (2010). Break our steganographic system. `http://agents.fel.cvut.cz/boss/index.php?mode=VIEW&tmpl=about/`, [Accessed 15 August 2018].

[45] Filler, T., Judas, J. and Fridrich, J. (2011). Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3), pp. 920–935.

[46] Fridrich, J. (2009). *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

[47] Fridrich, J. and Goljan, M. (2002). Practical steganalysis of digital images: state of the art. In *Electronic Imaging 2002*, International Society for Optics and Photonics, pp. 1–13.

[48] Fridrich, J. and Goljan, M. (2004). On estimation of secret message length in lsb steganography in spatial domain. In *Electronic Imaging 2004*, International Society for Optics and Photonics, pp. 23–34.

[49] Fridrich, J., Goljan, M. and Du, R. (2001). Detecting lsb steganography in color, and gray-scale images. *IEEE multimedia*, 8(4), pp. 22–28.

[50] Fridrich, J., Goljan, M. and Hogea, D. (2002). Attacking the outguess. In *Proceedings of the ACM Workshop on Multimedia and Security*, vol. 2002, Juan-les-Pins, France, pp. 976–982.

[51] Fridrich, J., Goljan, M. and Hogea, D. (2002). Steganalysis of jpeg images: Breaking the f5 algorithm. In *International Workshop on Information Hiding*, Springer, pp. 310–323.

[52] Fridrich, J., Goljan, M. and Soukal, D. (2004). Searching for the stego-key. In *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, International Society for Optics and Photonics, pp. 70–83.

[53] Fridrich, J., Goljan, M. and Soukal, D. (2005). Perturbed quantization steganography. *Multimedia Systems*, 11(2), pp. 98–107.

[54] Fridrich, J., Goljan, M. and Soukal, D. (2006). Wet paper codes with improved embedding efficiency. In *Electronic Imaging 2006*, International Society for Optics and Photonics, pp. 607215–607215.

[55] Fridrich, J. and Long, M. (2000). Steganalysis of lsb encoding in color images. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 3, IEEE, pp. 1279–1282.

[56] Fridrich, J. et al. (2005). Forensic steganalysis: determining the stego key in spatial domain steganography. In *Electronic Imaging 2005*, International Society for Optics and Photonics, pp. 631–642.

[57] Fridrich, J. et al. (2011). Steganalysis of content-adaptive steganography in spatial domain. In *International Workshop on Information Hiding*, Springer, pp. 102–117.

[58] Fridrich, J. J. (2004). Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. In *Information Hiding*, vol. 3200, Springer, pp. 67–81.

[59] Friedman, J., Hastie, T. and Tibshirani, R. (2001). *The elements of statistical learning*, vol. 1. Springer series in statistics New York.

[60] Gallagher, S. (2012). Steganography: how al-qaeda hid secret documents in a porn video. `http://arstechnica.com/business/2012/05/steganography-how-al-qaeda-hid-secret-documents-in-a-porn-video/`, [Accessed 14 August 2018].

[61] Gao, H. et al. (2010). Mp4 file creator for svc adaptive video streaming. In *Internet Technology and Applications, 2010 International Conference on*, IEEE, pp. 1–4.

[62] Ge, H., Huang, M. and Wang, Q. (2011). Steganography and steganalysis based on digital image. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 1, IEEE, pp. 252–255.

[63] Gelman, A. et al. (2015). *Bayesian Data Analysis 3rd edn.* Chapman & HallCRC Texts in Statistical Science.

[64] Gloe, T. and Böhme, R. (2010). The dresden image database for benchmarking digital image forensics. *Journal of Digital Forensic Practice*, 3(2-4), pp. 150–159.

[65] Goljan, M., Fridrich, J. and Holotyak, T. (2006). New blind steganalysis and its implications. In *Electronic Imaging 2006*, International Society for Optics and Photonics, pp. 607201–607201.

[66] Haddaji, R. et al. (2016). Comparison of digital signature algorithm and authentication schemes for h. 264 compressed video. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 7(9), pp. 357–363.

[67] Hansmann, F. and Gabriel, Y. (2008). Steganos security suite.

[68] Hartung, F. H. and Girod, B. (1996). Digital watermarking of raw and compressed video. In *Advanced Imaging and Network Technologies*, International Society for Optics and Photonics, pp. 205–213.

[69] Hemalatha, M. and Prasanna, A. (2014). Vinoth kumar d.,image steganography using hbc and rdh technique. *International Journal of Computer Applications Technology and Research*, 3, pp. 136–139.

[70] Hosur, P. I. and Ma, K.-K. (1999). Motion vector field adaptive fast motion estimation. In *Second International Conference on Information, Communications and Signal Processing (ICICS99)*, pp. 7–10.

[71] Hu, S. D. et al. (2011). A novel video steganography based on non-uniform rectangular partition. In *Computational Science and Engineering (CSE), 2011 IEEE 14th International Conference on*, IEEE, pp. 57–61.

[72] Hu, Y., Zhang, C. and Su, Y. (2007). Information hiding based on intra prediction modes for h. 264/avc. In *Multimedia and Expo, 2007 IEEE International Conference on*, IEEE, pp. 1231–1234.

[73] Incorporated, A. S. (2006). Pdf reference, version 1.7, sixth edition. `http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf`, [Accessed: 15 August 2018].

[74] Ismail, I., Nor, S. M. and Marsono, M. N. (2014). Stateless malware packet detection by incorporating naive bayes with known malware signatures. *Appl Comp Intell Soft Comput*, 2014, pp. 5:5–5:5.

[75] Jia-Fa, M. et al. (2016). A steganalysis method in the dct domain. *Multimedia Tools and Applications*, 75(10), pp. 5999–6019.

[76] Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer*, 31(2), pp. 26–34.

[77] Johnson, N. F. and Jajodia, S. (1998). Steganalysis of images created using current steganography software. In *International Workshop on Information Hiding*, Springer, pp. 273–289.

[78] Johnson, N. F. and Jajodia, S. (1998). Steganalysis: The investigation of hidden information. In *Information Technology Conference, 1998. IEEE*, IEEE, pp. 113–116.

[79] Johnson, N. F. and Sallee, P. A. (2008). Detection of hidden information, covert channels and information flows. *Wiley Handbook of Science and Technology for Homeland Security*.

[80] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), pp. 241–254.

[81] Jones, J. and Laskey, K. B. (2014). Using bayesian attack detection models to drive cyber deception. In *Proceedings of the Eleventh UAI Bayesian Modeling Applications Workshop, BMA@UAI 2014, Quebec City, Quebec, Canada, July 27, 2014.*, pp. 60–69.

[82] Jordan, F. (1997). Proposal of a watermarking technique for hiding/retrieving data in compressed and decompressed video. *ISO/IEC Doc JTC1/SC 29/QWG 11 MPEG 97/M 2281*.

[83] Kelley, J. (2001). Terror groups hide behind web encryption. `http://usatoday30.usatoday.com/life/cyber/tech/2001-02-05-binladen.htm`, [Accessed 15 August 2018].

[84] Ker, A. D. (2004). Improved detection of lsb steganography in grayscale images. In *International workshop on information hiding*, Springer, pp. 97–115.

[85] Ker, A. D. (2005). Steganalysis of lsb matching in grayscale images. *IEEE signal processing letters*, 12(6), pp. 441–444.

[86] Ker, A. D. and Pevny, T. (2012). Batch steganography in the real world. In *Proceedings of the on Multimedia and security*, ACM, pp. 1–10.

[87] Ker, A. D. and Pevnỳ, T. (2012). Identifying a steganographer in realistic and heterogeneous data sets. In *Media Watermarking, Security, and Forensics 2012*, vol. 8303, International Society for Optics and Photonics, p. 83030N.

[88] Ker, A. D. et al. (2013). Moving steganography and steganalysis from the laboratory into the real world. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, ACM, pp. 45–58.

[89] Khalind, O. S., Hernandez-Castro, J. C. and Aziz, B. (2013). A study on the false positive rate of stegdetect. *Digital Investigation*, 9(3-4), pp. 235–245.

[90] King, J. C. (2005). Adobe introduction to the insides of pdf.

[91] Kumar, M. (2011). *Steganography and steganalysis of joint picture expert group (JPEG) images*. University of Florida.

[92] Kurak, C. and McHugh, J. (1992). A cautionary note on image downgrading. In *Computer Security Applications Conference, 1992. Proceedings., Eighth Annual*, IEEE, pp. 153–159.

[93] Lerch-Hostalot, D. and Megías, D. (2013). Lsb matching steganalysis based on patterns of pixel differences and random embedding. *Computers & security*, 32, pp. 192–206.

[94] Levkov, M. (2010). Understanding the mpeg-4 movie atom. `https://www.adobe.com/devnet/video/articles/mp4_movie_atom.html`, [Accessed 15/08/2018].

[95] Li, B. et al. (2011). A survey on image steganography and steganalysis. *Journal of Information Hiding and Multimedia Signal Processing*, 2(2), pp. 142–172.

[96] Lingjun, L. et al. (2008). Detection of word shift steganography in pdf document. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, ACM, p. 15.

[97] Liu, B. et al. (2011). Thwarting audio steganography attacks in cloud storage systems. In *Cloud and Service Computing (CSC), 2011 International Conference on*, IEEE, pp. 259–265.

[98] Liu, G. et al. (2014). Adaptive steganography based on block complexity and matrix embedding. *Multimedia systems*, 20(2), pp. 227–238.

[99] Liu, Q., Sung, A. H. and Qiao, M. (2008). Video steganalysis based on the expanded markov and joint distribution on the transform domains detecting msu stegovideo. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, IEEE, pp. 671–674.

[100] Liu, Y. et al. (2009). Sidd: A framework for detecting sensitive data exfiltration by an insider attack. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, IEEE, pp. 1–10.

[101] Lu, P. et al. (2004). An improved sample pairs method for detection of lsb embedding. In *International Workshop on Information Hiding*, Springer, pp. 116–127.

[102] Luo, W., Huang, F. and Huang, J. (2010). Edge adaptive image steganography based on lsb matching revisited. *IEEE Transactions on information forensics and security*, 5(2), pp. 201–214.

[103] Machado, R. (2001). Ezstego. `http://www.jjtc.com`, [Accessed 16 August 2018].

[104] Mao, Q. (2014). A fast algorithm for matrix embedding steganography. *Digital Signal Processing*, 25, pp. 248–254.

[105] Mielikainen, J. (2006). LSB matching revisited. *IEEE signal processing letters*, 13(5), pp. 285–287.

[106] Mohamed, M. H. and Mohamed, L. M. (2016). High capacity image steganography technique based on lsb substitution method. *Applied Mathematics & Information Sciences*, 10(1), p. 259.

[107] Mozo, A. et al. (2009). Video steganography using flash video (flv). In *Instrumentation and Measurement Technology Conference, 2009. I2MTC'09. IEEE*, IEEE, pp. 822–827.

[108] Mstafa, R. J. and Elleithy, K. M. (2014). A highly secure video steganography using hamming code (7, 4). In *Systems, Applications and Technology Conference (LISAT), 2014 IEEE Long Island*, IEEE, pp. 1–6.

[109] Mukherjee, S. and Sharma, N. (2012). Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4, pp. 119 – 128.

[110] Nations, U. (2012). The use of the internet for terrorist purposes. `https://www.unodc.org/documents/frontpage/Use_of_Internet_for_Terrorist_Purposes.pdf`, [Accessed: 15 August 2018].

[111] Newman, L. H. (2018). Mysterious muslimcrypt appl helps jihadists send cover messages. `https://www.wired.com/story/muslimcrypt-steganography/`, [Accessed 19 April 2018].

[112] Nissar, A. and Mir, A. (2010). Classification of steganalysis techniques: A study. *Digital Signal Processing*, 20(6), pp. 1758–1770.

[113] Oliboni, C. (07/07/2012). Openpuff v4.00 steganography and watermarking. `https://embeddedsw.net/OpenPuff_Steganography_Home.html`.

[114] Paganini, P. (2014). Hackers exfiltrating data with video steganography via cloud video services. `http://securityaffairs.co/wordpress/30624/cyber-crime/hackers-used-data-exfiltration-based-video-steganography.html`, [Accessed 15 August 2018].

[115] Pang, W. et al. (2016). Rapid detection of stego images based on identifiable features. In *Advanced Communication Technology (ICACT), 2016 18th International Conference on*, IEEE, pp. 708–716.

[116] Pathak, P. and Selvakumar, S. (2014). Blind image steganalysis of jpeg images using feature extraction through the process of dilation. *Digital Investigation*, 11(1), pp. 67–77.

[117] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[118] Petitcolas, F. A., Anderson, R. J. and Kuhn, M. G. (1999). Information hiding-a survey. *Proceedings of the IEEE*, 87(7), pp. 1062–1078.

[119] Piva, A. and Barni, M. (2007). The first bows contest: Break our watermarking system. In *Security, Steganography, and Watermarking of Multimedia Contents*, p. 650516.

[120] Project, T. G. (2017). The guardian project. `https://guardianproject.info/`, [Accessed 19 April 2018].

[121] Proofpoint (2016). Massive adgholas malvertising campaigns use steganography and file whitelisting to hide in plain sight. `https://www.proofpoint.com/us/threat-insight/post/_massive-adgholas-malvertising-campaigns-use_-steganography-and-file-whitelisting-to-hide-in-plain-sight`, [Accessed 15 August 2018].

[122] Provos, N. (2001). Defending against statistical steganalysis. In *Usenix Security Symposium*, vol. 10, pp. 323–336.

[123] Provos, N. and Honeyman, P. (2001). Detecting steganographic content on the internet. *Ann Arbor*, 1001, pp. 48103–4943.

[124] Provos, N. and Honeyman, P. (2003). Hide and seek: An introduction to steganography. *Security & Privacy, IEEE*, 1(3), pp. 32–44.

[125] Rafat, K. F. and Sher, M. (2013). Secure digital steganography for ascii text documents. *Arabian Journal for Science and Engineering*, 38(8), pp. 2079–2094.

[126] Ridgway, J. (2013). Steganosaurus. `http://steganosaur.us/download`, [Accessed 10 August 2018].

[127] Sadek, M. M., Khalifa, A. S. and Mostafa, M. G. (2014). Video steganography: a comprehensive review. *Multimedia Tools and Applications*, pp. 1–32.

[128] Saha, S. (2000). Image compressionfrom dct to wavelets: a review. *Crossroads*, 6(3), pp. 12–21.

[129] Samidha, D. and Agrawal, D. (2013). Random image steganography in spatial domain. In *Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT), 2013 International Conference on*, IEEE, pp. 1–3.

[130] Schaefer, G. (2010). An uncompressed benchmark image dataset for colour imaging. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, IEEE, pp. 3537–3540.

[131] Schaefer, G. and Stich, M. (2003). Ucid: an uncompressed color image database. In *Electronic Imaging 2004*, International Society for Optics and Photonics, pp. 472–480.

[132] Schneier, B. (2007). Did nsa put a secret backdoor in new encryption standard? *URL: http://www wired com/politics/security/commentary/securitymatters/2007/11/securitymatters*, 1115, p. 2007.

[133] Seals, T. (2017). Steganography sees a rise in 2017. `https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf/`, [Accessed 15 August 2018].

[134] Shachtman, N. (2010). Fbi: Spies hid secret messages on public websites. `http://www.wired.com/2010/06/alleged-spies-hid-secret-messages-on-public-websites/`, [Accessed 15 August 2018].

[135] Shang, Y. (2007). A new invertible data hiding in compressed videos or images. In *Third International Conference on Natural Computation (ICNC 2007)*, vol. 5, IEEE, pp. 576–580.

[136] Sharp, T. (2001). An implementation of key-based digital signal steganography. In *International Workshop on Information Hiding*, Springer, pp. 13–26.

[137] Sherly, A. and Amritha, P. (2010). A compressed video steganography using tpvd. *International Journal of Database Management Systems (IJDMS) Vol*, 2, pp. 764–766.

[138] Simmons, G. J. (1984). The prisoners problem and the subliminal channel. In *Advances in Cryptology*, Springer, pp. 51–67.

[139] Simon, M. K. et al. (1994). *Spread spectrum communications handbook*, vol. 2. Citeseer.

[140] Sloan, T. and Hernandez-Castro, J. (2015). Forensic analysis of video steganography tools. *PeerJ Computer Science*, 1, p. e7.

[141] Sloan, T. and Hernandez-Castro, J. (2015). Steganalysis of openpuff through atomic concatenation of mp4 flags. *Digital Investigation*, 13, pp. 15–21.

[142] Software, B. (2010). BDV DataHider. `http://www.bdvnotepad.com/products/bdv-datahider/`, [Accessed 16/08/2018].

[143] SourceForge (2006). Atoms, boxes, parents, children and hex. `http://atomicparsley.sourceforge.net/changelog.html`, [Accessed 14 August 2018].

[144] Stanley, C. A. (2005). Pairs of values and the chi-squared attack. *Department of Mathematics, Iowa State University*.

[145] Stocker, R. (2018). Jpeg compression and jpeg quantization. `http://www.robertstocker.co.uk/jpeg/jpeg_new_10.htm`, [Accessed 14 August 2018].

[146] Stokes, J. (2010). How even the dumbest russian spies can outwit the nsa. `http://arstechnica.com/tech-policy/2010/07/how-even-the-dumbest-russian-spies-outwit-the-nsa/`, [Accessed 15 August 2018].

[147] Su, Y., Zhang, C. and Zhang, C. (2011). A video steganalytic algorithm against motion-vector-based steganography. *Signal Processing*, 91(8), pp. 1901–1909.

[148] Trivedi, S. and Chandramouli, R. (2005). Secret key estimation in sequential steganography. *IEEE Transactions on Signal Processing*, 53(2), pp. 746–757.

[149] Upham, D. (1997). Jpeg-jsteg. *Computer Software-Modification of the Independent JPEG groups JPEG Software (release 4)*.

[150] Wang, H. and Wang, S. (2004). Cyber warfare: steganography vs. steganalysis. *Communications of the ACM*, 47(10), pp. 76–82.

[151] Wang, K., Zhao, H. and Wang, H. (2014). Video steganalysis against motion vector-based steganography by adding or subtracting one motion vector value. *IEEE Transactions on Information Forensics and Security*, 9(5), pp. 741–751.

[152] Wang, Y. and Moulin, P. (2008). Perfectly secure steganography: Capacity, error exponents, and code constructions. *IEEE Transactions on Information Theory*, 54(6), pp. 2706–2722.

[153] Warkentin, M., Bekkering, E. and Schmidt, M. B. (2008). Steganography: Forensic, security, and legal issues. *The Journal of Digital Forensics, Security and Law: JDFSL*, 3(2), p. 17.

[154] Welstead, S. T. (1999). *Fractal and wavelet image compression techniques*. SPIE Optical Engineering Press (Bellingham, WA.

[155] Westfeld, A. (2001). F5 a steganographic algorithm. In *Information hiding*, Springer, pp. 289–302.

[156] Westfeld, A. (2002). Detecting low embedding rates. In *International Workshop on Information Hiding*, Springer, pp. 324–339.

[157] Westfeld, A. (2006). Steganalysis in the presence of weak cryptography and encoding. In *International Workshop on Digital Watermarking*, Springer, pp. 19–34.

[158] Westfeld, A. and Pfitzmann, A. (1999). Attacks on steganographic systems. In *International workshop on information hiding*, Springer, pp. 61–76.

[159] Wu, D.-C. and Tsai, W.-H. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters*, 24(9-10), pp. 1613–1626.

[160] Wu, J. et al. (2010). Steganalysis of msu stego video based on discontinuous coefficient. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 2, IEEE, pp. V2–96.

[161] Xu, C. and Ping, X. (2007). A steganographic algorithm in uncompressed video sequence based on difference between adjacent frames. In *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, IEEE, pp. 297–302.

[162] Xu, C., Ping, X. and Zhang, T. (2006). Steganography in compressed video stream. In *First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06)*, vol. 1, IEEE, pp. 269–272.

[163] Yang, G. et al. (2011). An information hiding algorithm based on intra-prediction modes and matrix coding for h. 264/avc video stream. *AEU-International Journal of Electronics and Communications*, 65(4), pp. 331–337.

[164] Yang, Y. (2013). *Information analysis for steganography and steganalysis in 3D polygonal meshes*. Ph.D. thesis, Durham University.

[165] Young, L. (2015). The dark side of steganography. `http://spectrum.ieee.org/tech-talk/telecom/security/the-dark-side-of-steganography/`, [Accessed 15 August 2018].

[166] Zambelli, A. (2009). Iis smooth streaming technical overview. `https://docs.microsoft.com/en-us/iis/media/on-demand-smooth-streaming/smooth-streaming-technical-overview`, [Accessed 16 August 2018].

[167] Zhang, C., Su, Y. and Zhang, C. (2008). A new video steganalysis algorithm against motion vector steganography. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, IEEE, pp. 1–4.

[168] Zhang, J., Li, J. and Zhang, L. (2001). Video watermark technique in motion vector. In *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, IEEE, pp. 179–182.

[169] Zhang, X. and Wang, S. (2006). Efficient steganographic embedding by exploiting modification direction. *IEEE Communications Letters*, 10(11).

[170] Zhao, L. and Guan, L. (2010). An optimized method and implementation for parsing mp4 metadata. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, vol. 2, IEEE, pp. 984–987.

[171] Zhao, Y., Ni, R. and Cao, G. (2013). A study on embedding efficiency of matrix encoding. *Emerging Digital Forensics Applications for Crime Detection, Prevention, and Security*, p. 92.

[172] Zhao, Y. et al. (2015). Video steganalysis based on intra prediction mode calibration. In *International Workshop on Digital Watermarking*, Springer, pp. 119–133.

[173] Zhong, S., Cheng, X. and Chen, T. (2007). Data hiding in a kind of pdf texts for secret communication. *IJ Network Security*, 4(1), pp. 17–26.

[174] Zielińska, E., Mazurczyk, W. and Szczypiorski, K. (2014). Trends in steganography. *Communications of the ACM*, 57(3), pp. 86–95.