

**University of New Mexico**  
**UNM Digital Repository**

---

Electrical and Computer Engineering ETDs

Engineering ETDs

---

2-9-2010

# Information similarity metrics in information security and forensics

Tu-Thach Quach

Follow this and additional works at: [https://digitalrepository.unm.edu/ece\\_etds](https://digitalrepository.unm.edu/ece_etds)

---

## Recommended Citation

Quach, Tu-Thach. "Information similarity metrics in information security and forensics." (2010). [https://digitalrepository.unm.edu/ece\\_etds/212](https://digitalrepository.unm.edu/ece_etds/212)

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact [disc@unm.edu](mailto:disc@unm.edu).

Tu-Thach Quach

*Candidate*


Electrical and Computer Engineering

*Department*

This dissertation is approved, and it is acceptable in quality and form for publication:

*Approved by the Dissertation Committee:*

  
\_\_\_\_\_, Chairperson

  
\_\_\_\_\_

Major Tatixus  
\_\_\_\_\_

Chuki Sheller  
\_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

# Information Similarity Metrics in Information Security and Forensics

by

**Tu-Thach Quach**

B.S., Computer Engineering, University of New Mexico, 1999

M.S., Electrical Engineering, University of New Mexico, 2003

DISSERTATION

Submitted in Partial Fulfillment of the  
Requirements for the Degree of

Doctor of Philosophy  
Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

©2009, Tu-Thach Quach

# Acknowledgments

I would like to express my gratitude to my committee members: Professors Heileman, Pérez-González, Abdallah, and Pattichis. Professor Heileman has an amazing ability for seeing the big picture and connecting concepts that might seem disconnected at first. You have taught me how to approach problems from many different perspectives. Professor Pérez-González, you have taught me many technical skills, to be a deep thinker, to challenge assumptions, and to be skeptical. Professors Abdallah and Pattichis, thank you for your utmost support.

This dissertation would not have been possible without the excellent educational programs supported by Sandia National Laboratories and the Department of Energy. My manager, Stephen Kleban, thank you for supporting my effort and filling in for me during my absence from work.

I also would like to thank members of DRAKE for enduring many of my talks. Your comments certainly contributed to the improvement of my research.

# **Information Similarity Metrics in Information Security and Forensics**

by

**Tu-Thach Quach**

## **ABSTRACT OF DISSERTATION**

Submitted in Partial Fulfillment of the  
Requirements for the Degree of

Doctor of Philosophy  
Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

# Information Similarity Metrics in Information Security and Forensics

by

**Tu-Thach Quach**

B.S., Computer Engineering, University of New Mexico, 1999

M.S., Electrical Engineering, University of New Mexico, 2003

Ph.D., Engineering, University of New Mexico, 2009

## **Abstract**

We study two information similarity measures, relative entropy and the similarity metric, and methods for estimating them. Relative entropy can be readily estimated with existing algorithms based on compression. The similarity metric, based on algorithmic complexity, proves to be more difficult to estimate due to the fact that algorithmic complexity itself is not computable. We again turn to compression for estimating the similarity metric. Previous studies rely on the compression ratio as an indicator for choosing compressors to estimate the similarity metric. This assumption, however, is fundamentally flawed. We propose a new method to benchmark compressors for estimating the similarity metric. To demonstrate its use, we propose to quantify the security of a stegosystem using the similarity metric. Unlike other measures of steganographic security, the similarity metric is not only a true distance metric, but it is also universal in the sense that it is asymptotically minimal among all computable metrics between two objects. Therefore, it accounts for all similarities

between two objects. In contrast, relative entropy, a widely accepted steganographic security definition, only takes into consideration the statistical similarity between two random variables. As an application, we present a general method for benchmarking stegosystems. The method is general in the sense that it is not restricted to any coverttext medium and therefore, can be applied to a wide range of stegosystems. For demonstration, we analyze several image stegosystems using the newly proposed similarity metric as the security metric. The results show the true security limits of stegosystems regardless of the chosen security metric or the existence of steganalysis detectors. In other words, this makes it possible to show that a stegosystem with a large similarity metric is inherently insecure, even if it has not yet been broken.



# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Organization . . . . .	5
<b>2 Information Similarity Metrics</b>	<b>7</b>
2.1 Relative Entropy . . . . .	8
2.2 Algorithmic Distance . . . . .	8
2.2.1 Algorithmic Information Theory . . . . .	9
2.2.2 Information Distance . . . . .	18
2.3 Summary . . . . .	19
<b>3 Metric Estimation</b>	<b>20</b>
3.1 Entropy and Relative Entropy Estimation . . . . .	21

## Contents

3.1.1	The Burrows-Wheeler Transform . . . . .	23
3.1.2	Experimental Result . . . . .	26
3.2	Algorithmic Complexity Estimation . . . . .	27
3.2.1	Experimental Result . . . . .	29
3.2.2	Related Work . . . . .	33
3.3	Summary . . . . .	33
<b>4</b>	<b>Steganography and Steganalysis</b>	<b>35</b>
4.1	Steganography . . . . .	38
4.1.1	Least-significant Bit Embedding . . . . .	38
4.1.2	F5 . . . . .	40
4.1.3	Steghide . . . . .	42
4.1.4	Model-based . . . . .	44
4.2	Steganalysis . . . . .	46
4.2.1	Method-specific Steganalysis . . . . .	48
4.2.2	Universal Steganalysis . . . . .	49
4.3	Model-based Steganalysis . . . . .	53
4.3.1	Cover Image . . . . .	53
4.3.2	Stego Image . . . . .	54
4.3.3	Invariant Feature . . . . .	56
4.3.4	Detecting The Hidden Message . . . . .	57

## *Contents*

4.4	Summary . . . . .	65
<b>5</b>	<b>Benchmarking Stegosystems</b>	<b>67</b>
5.1	Steganographic Security . . . . .	68
5.2	Steganographic Capacity . . . . .	72
5.3	Experimental Result . . . . .	74
5.3.1	Information-Theoretic Result . . . . .	75
5.3.2	Algorithmic Complexity Result . . . . .	83
5.4	Summary . . . . .	86
<b>6</b>	<b>Conclusion and Future Work</b>	<b>88</b>
6.1	Enhancing Steganalysis Detector . . . . .	89
6.2	Steganalysis and Algorithmic Complexity . . . . .	90
	<b>Appendices</b>	<b>92</b>
<b>A</b>	<b>Mean and Standard Deviation of Security Estimates</b>	<b>93</b>
A.1	Information-Theoretic Model . . . . .	94
A.2	Algorithmic Complexity Model . . . . .	95
	<b>References</b>	<b>96</b>

# List of Figures

2.1	Turing machine . . . . .	10
3.1	Comparison of the LZ and BWT (a) entropy estimators and (b) relative entropy estimators. . . . .	26
3.2	$NCD(x, x)$ at different sequence lengths for (a) all compressors and (b) for the <i>lzma</i> compressor. $NCD(x, x)$ for the <i>lzma</i> compressor actually decreases as sequence length increases. . . . .	31
3.3	$ NCD(x, y) - NCD(y, x) $ at different sequence lengths. Each point is an average of 100 sequences. . . . .	32
4.1	A sender uses the encoder of a stegosystem to hide message $M$ into coartext $X$ to produce stegotext $Y$ , which is sent over a noiseless public channel to a receiver. The receiver uses the decoder to extract message $M$ from the stegotext. . . . .	36
4.2	LSB replacement overrides the least-significant bits of the coartext with the message bits to produce a stegotext. LSB embedding results in little visual distortion. . . . .	39

## List of Figures

4.3	LSB embedding equalizes the histogram bins of pixel values that differ only in the least-significant bit. Before embedding, the number of 6's is 60 and the number of 7's is 40. After embedding, the two numbers are likely to be equal when the message size is sufficiently large. . . . .	41
4.4	The decompressor uses the statistical model of the coverttext to decompress a random message so that the output sequence has the desired statistical properties of the model. The output sequence can then replace the least-significant portion of the coverttext to produce the stegotext. . . . .	45
4.5	Visual attack on LSB embedding exploiting the ability of the human visual system to recognize patterns: (a) cover image (b) least-significant bit image of cover image and (c) least-significant bit image of stego image where the left half is overwritten with the message bits. The outline of the cover image can be seen in the least-significant bits. LSB embedding destroys this pattern. . . . .	47
4.6	The scatter plot of $ r_{2,1}(5) $ , $ r_{2,1}(10) $ , and $ r_{2,1}(15) $ for 100 random cover images and stego images using LSB and Steghide with message size at 100% of the cover capacity. The stego images are generated from a different set of 100 cover images. . . . .	60
4.7	A detector's false-alarm probability versus detection probability for a perfectly secure stegosystem. . . . .	61
4.8	The ROC curves for our classifier along with the SSIS, LSB, and Steghide hiding schemes (a). The ROC curves for four different message sizes at 100%, 50%, 25%, and 5% of the cover capacity (b). . .	62

## List of Figures

4.9	The detection accuracy for different hiding algorithms and message sizes at 95% cover detection accuracy. . . . .	63
5.1	The security and capacity trade-off of the simple stegosystem. The relative entropy estimates are good approximations of the ideal values.	77
5.2	An example illustrating how JPEG coefficients are converted into a one-dimensional sequence. . . . .	77
5.3	Security and capacity trade-off of F5, Steghide, MB, and Outguess JPEG stegosystems. . . . .	78
5.4	Security and capacity trade-off of Steghide and LSB pixel-domain stegosystems using (a) vertical and (b) horizontal concatenation of pixels. . . . .	81
5.5	The security and capacity trade-off of the simple stegosystem using the <i>lzma</i> compressor. The estimates closely match the ideal values. .	84
5.6	Security and capacity trade-off of F5, Steghide, MB, and Outguess JPEG stegosystems using algorithmic complexity model. . . . .	85

# List of Tables

3.1	The Ziv-Lempel algorithm on the sequence <i>abbbbaaabba</i> . . . . .	22
3.2	The Ziv-Merhav algorithm with $x^n = \textit{abbbbaaabba}$ and $z^n = \textit{baababaabba}$ . . . . .	23
3.3	An example of the BWT algorithm. . . . .	24
3.4	The mean $NCD(x, x)$ and mean compression ratio on $x$ for three different compressors on randomly generated ASCII texts of 100,000 characters. . . . .	30
3.5	The mean $NCD(x, x)$ and mean compression ratio on $x$ for three different compressors on JPEG images. . . . .	30
3.6	The $NCD(x, x)$ and compression ratio on $x$ for <i>lzma</i> and DNACompress on <i>Plasmodium falciparum</i> DNA sequences. . . . .	31
A.1	Mean and standard deviation of relative entropy estimate at different $\alpha$ for F5, Steghide, MB, and Outguess JPEG stegosystems. . . . .	94
A.2	Mean and standard deviation of $NCD$ at different $\alpha$ for F5, Steghide, MB, and Outguess JPEG stegosystems. . . . .	95

# Chapter 1

## Introduction

The ability to quantify the similarity between two objects is of great importance in many applications such as data mining and pattern recognition. The Euclidean distance between two points in a plane measures how close they are in space. Hamming distance defines the distance between two binary strings of equal length as the number of positions in which the two strings differ. It quantifies the number substitutions required to make two strings match. This metric only captures the positional dependencies of two strings. It is not the minimal number of changes required to make two strings match. Edit distance is a generalization of Hamming distance, where the set of operations is expanded to include insertion and deletion. These distance metrics have been used successfully in many applications such as clustering, DNA matching, and, of course, spell checking.

In our information age, we are interested in metrics that quantify the information similarity between two objects, which might not have coordinates or well-defined operations. Information similarity has a wide range of important applications. In music applications, for example, the similarity of two songs can be used to recommend songs to consumers based on their past choices. Searching for similar images



## Chapter 1. Introduction

can apply information similarity to find images that have similar content rather than pixels. Information theory is an appropriate framework to study information similarity. One of the fundamental quantities in information theory is relative entropy, which quantifies the divergence between two random variables [1]. It is not surprising that relative entropy has been used in many applications including classification [2, 3] and intrusion detection [4]. An important application of relative entropy is steganography. In steganography, it is critical that the coverttext and the stegotext are similar to minimize detection. This similarity determines the security of a stegosystem and is defined as the relative entropy between the coverttext and the stegotext distributions [5, 6].

Despite its popularity, relative entropy has many drawbacks. It is not a distance metric because it is not symmetric and does not satisfy the triangle inequality. More importantly, information-theoretic metrics such as relative entropy inherently deal with probabilities and ensembles, not individual objects. In this sense, relative entropy is not general as it requires two probability distributions. What if we want to compare two objects that do not have distributions or their distributions are unknown? Even with distributions, relative entropy fails to capture the information similarity of two distributions. Two binary random variables  $X$  and  $Y$  with distributions  $p_X(0) = p$  and  $p_Y(0) = 1 - p$ , where  $p < 0.5$ , have non-zero relative entropy. Yet, they represent identical information sources. This is a consequence of the fact that relative entropy only considers statistically regularities between two distributions.

An alternative metric based on algorithmic complexity is the *information distance* [7] and its normalized form, the *similarity metric* [8]. The benefits of using algorithmic complexity is that it is a universal information quantity. As a result, the similarity metric is a universal metric in the sense that it is asymptotically minimal among all computable metrics between two objects. Universality is an important

## *Chapter 1. Introduction*

point to emphasize. This implies that any similarity between two objects is accounted for by the similarity metric. As an additional benefit, algorithmic complexity deals with individual objects, not ensembles, which means that it does not rely on any underlying probabilistic model. Whether two objects are probability distributions or two paintings, the similarity between them is defined.

In order to use these metrics in applications, they must be calculated. Relative entropy can be computed trivially when the distributions are known. In many applications, however, the distributions are not known. Digital images, for example, have high degrees of dependencies among pixels and the distributions are difficult to obtain. In these situations, the underlying distribution can only be estimated. As simple as it sounds, estimating relative entropy has been a road block due to the poor performance of existing estimators. We study in depth two estimators based on compression algorithms and show that the estimator based on the Burrows-Wheeler Transform performs better.

Computing the similarity metric proves to be more difficult due to the fact that algorithmic complexity itself is not computable. We can only hope to approximate it using compression. The idea is that the compressed version of an object serves as the shortest representation, and estimates the shortest program, for producing that object. Thus, the size of the compressed object can approximate its complexity. Identifying a good compressor to estimate the similarity metric has been an active area of research. This is critical as bad compressors lead to poor results. In the case of classification, for example, objects would be put in the wrong class. We examine several well-known compression programs to estimate the similarity metric. We show that related works using compression ratio to benchmark compressors are flawed as minimal compression ratio does not guarantee good estimates for the similarity metric. We propose a new method to benchmark compressors for estimating the similarity metric and test several compressors on different types of objects including

## Chapter 1. Introduction

text, images, and DNA sequences. Contrary to results reported in literature, we show that the popular *ppmd* compressor does not perform well as reported. The *lzma* compressor, on the other hand, is a general-purpose compressor that outperforms *ppmd* and task-specific compressors.

To demonstrate its use, we study the security of stegosystems. A fundamental requirement of a stegosystem is its security. This is the condition where the warden cannot distinguish stegotexts from coverttexts, i.e., the presence of the hidden message cannot be detected by the warden. From a practical approach, the security of a stegosystem is viewed from the perspective of a steganalysis detector. This approach, however, has many issues. The results are dependent on many factors including the performance of the detectors, the training process, and the data set. An alternative approach is define steganographic security independent of any steganalysis detector. There exist many works that try to define steganographic security using different quantities. The problem with these definitions, however, is that they only consider one aspect of security based on the chosen quantity. Therefore, it is difficult to compare the security of stegosystems under different definitions. For example, a stegosystem is shown to be more secure than another stegosystem under relative entropy, but is less secure with respect to a different security definition. This immediately raises an issue of the choice of security measure. Due to its universality, we propose to define steganographic security using the similarity metric which overcomes this problem.

As an application, we present a general-purpose method for benchmarking stegosystems. Unlike existing works, this method does not rely on any steganalysis detector. As a result, it does not have many of the drawbacks such as classifier parameters and training associated with using steganalysis detectors. Furthermore, it is general in the sense that it is not restricted to any coverttext medium and therefore, can be applied to a wide range of stegosystems. Our method requires only a security met-

ric and the steganographic capacity of a stegosystem. We benchmark several image stegosystems using the newly proposed similarity metric as the security metric. For comparison, we also use relative entropy as the security metric. We demonstrate the generality of our approach to show that it is not restricted to JPEG images or file formats by applying it to pixel-domain stegosystems. The results show the security and capacity trade-off of stegosystems and serve as a benchmark for stegosystems. More importantly, the results show the true security limits of stegosystems regardless of the chosen security metric or the existence of steganalysis detectors. In other words, this makes it possible to show that a stegosystem with a large similarity metric is inherently insecure, even if it has not yet been broken.

## **1.1 Organization**

The technical content of this dissertation is divided into four Chapters. Chapter 2 describes relative entropy, information distance, and the similarity metrics in detail. Fundamental properties of each metric are presented. In addition, algorithmic information theory is described in detail as well as its connection to prefix codes. Chapter 3 discusses methods to estimate relative entropy and the similarity metric. A new method to benchmark compressors for estimating the similarity metric is proposed. This is in contrast to existing methods that rely on optimal compression ratio to choose compressors. Experimental results are presented to compare each method. Chapter 4 describes steganography in detail including the common methods used in steganography as well as the challenges in developing secure stegosystems. It covers several popular image stegosystems. Common steganalysis techniques are also outlined. Our model-based steganalysis method is developed in detail. A discussion on how steganalysis detectors are used to quantify the security of stegosystems is presented. We re-examine the security issue of stegosystems in Chapter 5. We define

## *Chapter 1. Introduction*

the security of a stegosystem using the similarity metric and present our general-purpose approach to benchmark stegosystem. Experimental results for real image stegosystems are shown. Chapter 6 concludes our work and discusses future research directions.

## Chapter 2

# Information Similarity Metrics

Two theories that quantify information are classical information theory and algorithmic information theory. Shannon's approach in classical information theory was motivated as a communication problem. As such, it deals with the probabilistic nature of communication. Algorithmic information theory, on the other hand, was motivated from the complexity of objects, and how to describe those objects efficiently. It is remarkable that both approaches end up with many similarities. The entropy of a random variable, for example, is approximately equal to the length of the shortest program to describe it. In terms of information similarity, information theory offers a convenient metric, relative entropy, that quantifies how different two random variables are. Algorithmic information theory has a true distance metric that measures the information similarity between two objects. This is the information distance [7] and its normalized form is the similarity metric [8]. Both inherit many benefits from algorithmic complexity. An important property is universality. That is, the information distance is minimal among all computable distances between two objects. Therefore, it accounts for all similarities between two objects. This is truly a remarkable property. We formally define these metrics in this chapter with an emphasis on algorithmic complexity.

## 2.1 Relative Entropy

Perhaps the most popular metric in information theory that measures the divergence between two probability distributions  $p_X$  and  $p_Y$  is relative entropy:

$$D(p_X||p_Y) = \sum_x p_X(x) \log \frac{p_X(x)}{p_Y(x)}. \quad (2.1)$$

We are careful to use the word divergence rather than distance because relative entropy is not a true distance as it is not symmetric and does not satisfy the triangle inequality. It does, however, has some properties that make it resemble a distance metric as shown in the following theorem.

**Theorem 2.1.1** (Positivity).  $D(p_X||p_Y) \geq 0$  with equality iff  $p_X = p_Y$ .

## 2.2 Algorithmic Distance

Relative entropy as a distance measure has several drawbacks. It is not a metric because it is not symmetric and does not satisfy the triangle inequality. More importantly, information-theoretic metrics such as relative entropy inherently deal with probabilities and ensembles, not individual objects. In this sense, relative entropy is not general as it requires two probability distributions. What if we want to compare two objects that do not have distributions or their distributions are unknown? Even with distributions, relative entropy fails to capture the information similarity of two distributions. Two binary random variables  $X$  and  $Y$  with distributions  $p_X(0) = p$  and  $p_Y(0) = 1 - p$ , where  $p < 0.5$ , have non-zero relative entropy. Yet, they represent identical information sources. This is a consequence of the fact that relative entropy only considers statistical regularities between two distributions. This same situation can be extended to other popular distance metrics such as Hamming distance. A black and white picture and its negative have a large Hamming distance, but the

two pictures look similar. Clearly, relative entropy, Hamming distance, and other popular distances fail to capture this similarity. From a computation perspective, however, the negative image is easily generated. Turn all zeros into ones and turn all ones into zeros. It would be ideal if we can define a universal metric that takes into account all computable similarities between two objects. For this, we turn to *algorithmic information theory*.

### 2.2.1 Algorithmic Information Theory

We toss a fair coin 20 times and we obtain all heads. Instantly, we do not believe this is a fair coin because the sequence does not look random. Our argument is that since this is a fair coin, we *expect* some heads and some tails, but not all heads. Why? Probability tells us that any sequence is as likely and each has probability  $2^{-20}$ . So, why should we feel that a sequence of all heads is not random? The problem is that probability only works with expectations of ensembles, not individual objects. This simple example illustrates another problem with probability and classical information theory. The entropy of the source (the coin toss) is 1. However, the empirical entropy of the sequence of all heads is practically 0. It is not possible to embed any usable quantity of information into this sequence, but the source entropy tells us otherwise. Again, the problem is that classical information theory tells us about the expected quantity of information of ensembles, not individual objects.

This brings us to algorithmic information theory that quantifies the information content of individual objects. Algorithmic information theory was independently discovered by Kolmogorov, Solomonoff, and Chaitin [9, 10, 11]. We are interested in a efficient way to describe objects. Some objects are easily described. A sequence of one million heads is easily described. We have just described this sequence in just a few bytes. The number 3.1415926535897932384626433832795..., or  $\pi$ , seems to be com-



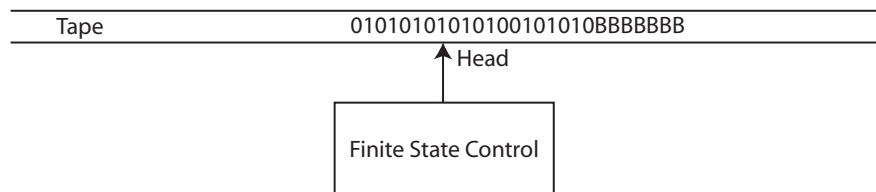


Figure 2.1: Turing machine

plicated, but it too can be described in just a few bytes. There are, however, many objects that are difficult to describe, the binary sequence 0110001101010110001... is difficult to describe other than to write the sequence itself.

In order to be useful, we must agree on a universal mechanism to describe objects. If such universal mechanism does not exist, then we cannot have a standard way of quantifying the information content of objects as this quantity would depend on the chosen description mechanism. If, for example, we do not understand the symbol  $\pi$ , then it would not mean anything to us and we cannot compute  $\pi$ . Similarly, some programming languages are more elegant at specific tasks than others and require only a few lines of codes to implement difficult algorithms. It turns out though that this universal mechanism exists and is due to the theory of computation developed in the 1930s.

In 1936, Alan Turing described a simple computer and showed that any function that can be computed by a human can be computed by the computer [12]. This Turing machine consists of a finite state control with a head and a tape as shown in Figure 2.1.

At each execution step, the machine is allowed to read the value on the tape at the head location and can perform only two basic operations:

1. write to the tape at the current head location

2. move the head one position to the left or right

The finite state control is allowed to change its state at the end of each step according to a determined set of rules based on its current state and the symbol read from the tape. We can view a Turing machine as a function that maps finite binary strings to finite or infinite binary strings. Some combinations of states and symbols lead the finite state control to enter a *halt* state and ends its execution. In these cases, the function is defined. Not all combinations halt execution. In these cases, the function is undefined. The set of functions that are computed by Turing machines are collectively referred to as *partial recursive functions*.

One of the most celebrated results of Turing's work is the *halting problem*, which states that there is no computer that can determine whether any arbitrary computer will halt on some input. This theorem has important consequences. One of the consequences, as we will see shortly, is that the algorithmic complexity of an object is non-computable.

**Theorem 2.2.1** (Halting Theorem). *Let  $T_1, T_2, \dots$  be the enumeration of computers. There is no computer  $H$  such that  $H(i, x) = 1$  if  $T_i(x)$  is defined (i.e., halts) and  $H(i, x) = 0$  if  $T_i(x)$  is undefined for all  $i, x$ .*

*Proof.* We prove by contradiction using circular referencing. Suppose that  $H$  does exist. Define computer  $\overline{H}$  as follows.  $\overline{H}(x)$  is defined if  $H(x, x) = 0$  and  $\overline{H}(x)$  is undefined if  $H(x, x) = 1$ . Let  $T_j = \overline{H}$  in the enumeration of computers. Now, what happens if we execute  $H(j, j)$ ? If  $H(j, j) = 1$ , then  $T_j(j)$  is defined, but according to the definition of  $T_j$ ,  $T_j(j)$  is defined only if  $H(j, j) = 0$ . Similarly, if  $H(j, j) = 0$ , then  $T_j(j)$  is undefined, but  $T_j(j)$  is undefined only if  $H(j, j) = 1$ . Thus, we have a contradiction.  $\square$

Turing's model of computation is quite simple. It turns out that Turing's model

## Chapter 2. Information Similarity Metrics

of computation is equivalent to any other models in the sense that they all can compute the same set of functions within a polynomial time factor [13]. This is now commonly known as *Church's Thesis*. Thus, we can construct a *universal* computer that simulates all Turing machines. This brings us back to algorithmic information theory.

Informally, the complexity or information content of an object is the length of the shortest program that generates the object. Formally, let  $x$  be a finite binary sequence and  $U$  be a universal computer. Let  $p$  be a program executable by  $U$ . Denote the length of  $p$  as  $l(p)$ . The algorithmic complexity of  $x$  with respect to  $U$  is defined as

$$K_U(x) = \min_{p:U(p)=x} l(p). \quad (2.2)$$

If no such program exists,  $K_U(x) = \infty$ .

Similarly, we can define a conditional version of algorithmic complexity of  $x$  given  $y$  with respect to computer  $U$  as

$$K_U(x|y) = \min_{p:U(p,y)=x} l(p). \quad (2.3)$$

If no such program exists,  $K_U(x|y) = \infty$ . In other words, the conditional complexity is the shortest program length that computes  $x$  if  $y$  is made available to computer  $U$ . This is equivalent to saying that  $y$  is a parameter or data made available to program  $p$  on computer  $U$ . To satisfy the symmetry of algorithmic information where  $K_U(x, y) = K_U(y|x) + K_U(x)$ ,  $K_U(x|y)$  is defined as  $K(x|y^*)$ , where  $y^*$  is the shortest program for  $y$ . For an in-depth discussion on the symmetry of information issue, please refer to the textbook by Li and Vitányi [14].

We instantly run into the problem described above where the complexity is now dependent on computer  $U$ . Fortunately, the *invariance theorem*, which relies on the existence of universal computers, solves this problem elegantly. This theorem

## Chapter 2. Information Similarity Metrics

is considered as one of the most fundamental theorems in algorithmic information theory since the entire theory relies on it.

**Theorem 2.2.2** (Invariance Theorem). *Let  $U$  be a universal computer and  $V$  be any other computer. Then,*

$$K_U(x) \leq K_V(x) + c, \quad (2.4)$$

where  $c$  is independent of  $x$ .

*Proof.* Let  $T_1, T_2, \dots$  be the set of all computers. Let  $p$  be a shortest program to compute  $x$  using computer  $V$ , i.e.,  $V(p) = x$ . Let  $V$  correspond to  $T_n$  in the enumeration of computers. Denote the length of  $n$  as  $l(n)$ . Universal computer  $U$  uses the following program to compute  $x$

$$p_U = 1^{l(n)}0np. \quad (2.5)$$

$U$  parses the input program to extract  $n$ , then simulates  $V$  and executes  $p$  to compute  $x$ . Thus,

$$K_U(x) \leq K_V(x) + 2l(n) + 1. \quad (2.6)$$

□

The constant itself can be quite large depending on  $n$ . That is not the point of the theorem, however. The key point is that the constant does not depend on  $x$ . Thus, the complexity of  $x$  depends only on  $x$ . Therefore, we will not refer to any specific computer  $U$  so that the algorithmic complexity of  $x$  is denoted as  $K(x)$ .

There are several versions of algorithmic complexity. The most well-known version is *prefix* algorithmic complexity. In prefix algorithmic complexity, the programs form a prefix-free set. That is, no program is a prefix of another program. This restriction has many connections with information theory. In particular, there is a

strong tie to Shannon's source coding theorem and Kraft inequality, which ultimately leads to the universality of algorithmic complexity.

**Definition** A *code*  $C$  for alphabet  $X$  is a mapping from the elements of  $X$  to binary strings, i.e.,  $C : X \rightarrow \{0, 1\}^*$ . The binary string  $C(x)$  is the corresponding *codeword* for  $x$ .

**Definition** A code is a *prefix code* if no codeword is a prefix of another codeword.

The following theorem, due to Kraft [15], establishes a constraint on the codeword lengths of any prefix code.

**Theorem 2.2.3** (Kraft Inequality). *There exists a prefix code with codeword lengths  $l_1, l_2, \dots$  if and only if*

$$\sum_i 2^{-l_i} \leq 1. \quad (2.7)$$

It is straightforward to see that  $\sum 2^{-K(x)} \leq 1$  since the programs form a prefix free set. An important consequence of Kraft inequality is the establishment of a lower bound on the expected codeword length of any prefix code for a given distribution. This remarkable result was discovered by Shannon in his original paper [16] and is commonly known as the noiseless coding theorem or source coding theorem.

**Theorem 2.2.4** (Source Coding Theorem). *Let  $l(x)$  be the codeword length for  $x$  of some prefix code and  $p(x)$  be the corresponding probability of  $x$ . The expected codeword length satisfies*

$$\sum_x p(x) \log \frac{1}{p(x)} \leq \sum_x p(x) l(x). \quad (2.8)$$

## Chapter 2. Information Similarity Metrics

In other words, the expected codeword length of any prefix code is lower bounded by the entropy of the random variable over the alphabet. Equality is achieved when the codeword lengths are optimal,

$$l^*(x) = \log \frac{1}{p(x)}. \quad (2.9)$$

An optimal prefix code exists only if the optimal lengths are integer, which is often not case. One way to construct a prefix code for a given distribution that is almost optimal is through Shannon-Fano coding. To solve the non-integer problem, Shannon-Fano coding rounds up the non-integer lengths to become

$$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil. \quad (2.10)$$

These codeword lengths still satisfy Kraft inequality since

$$\sum_x 2^{-\lceil \log \frac{1}{p(x)} \rceil} \leq \sum_x 2^{-\log \frac{1}{p(x)}} = 1. \quad (2.11)$$

Therefore, by the Kraft inequality, we can construct a prefix code with these round-up codeword lengths.

An important consequence of the above discussion is the following theorem, which establishes algorithmic complexity as a universal information metric.

**Theorem 2.2.5.** *For any computable real-valued function  $f(x)$  satisfying the inequality  $\sum_x 2^{-f(x)} \leq 1$ ,*

$$K(x) \leq f(x) + c, \quad (2.12)$$

*where  $c$  is a constant independent of  $x$ .*

*Proof.* Since  $\sum_x 2^{-f(x)} \leq 1$ , there exists a prefix code with codeword lengths  $\lceil f(x) \rceil$ . Thus, we can construct a computer  $\phi(x)$  that produces a codeword for  $x$ . Since the

## Chapter 2. Information Similarity Metrics

codewords form a prefix code,  $\phi(x)$  is unique for every  $x$ . Therefore, we can construct another computer using codewords  $\phi(x)$  as programs to compute  $x$ , i.e.,  $\varphi(\phi(x)) = x$ . The complexity of  $x$  is the length of  $\phi(x)$ , which is  $\lceil f(x) \rceil$ . Altogether, we have

$$K_\varphi(x) = \lceil f(x) \rceil \leq f(x) + 1. \quad (2.13)$$

Combined with the Invariance Theorem (Theorem 2.2.2), we have

$$K(x) \leq K_\varphi(x) + c \quad (2.14)$$

$$\leq f(x) + c + 1 \quad (2.15)$$

$$= f(x) + O(1). \quad (2.16)$$

□

**Corollary 2.2.6.** *For any computable function  $p(x) \geq 0$  satisfying  $\sum_x p(x) \leq 1$ ,*

$$2^{-K(x)} \geq cp(x). \quad (2.17)$$

We can view  $2^{-K(x)}$  as a universal probability for  $x$  in the sense that any other probability distribution on  $x$  is within a multiplicative constant factor of  $2^{-K(x)}$ . Furthermore,  $K(x) \leq \log \frac{1}{p(x)} + c$ , which resembles Shannon's definition of information suggesting that there is a connection between algorithmic complexity and entropy.

**Theorem 2.2.7.** *For any computable probability mass function  $p(x)$  with finite entropy, i.e.,  $H(p) < \infty$ ,*

$$0 \leq \sum_{x_i} p(x_i) K(x_i) - H(p) \leq c. \quad (2.18)$$

*Proof.* Since  $K(x)$  is the length of a prefix code, the first inequality follows immediately from Theorem 2.2.4. For the second inequality, Theorem 2.2.5 states that  $K(x) \leq \log \frac{1}{p(x)} + c$ . Thus,

$$\sum_{x_i} p(x_i) K(x_i) \leq \sum_{x_i} p(x_i) \left( \log \frac{1}{p(x_i)} + c_i \right) \quad (2.19)$$

$$\leq H(p) + c^*, \quad (2.20)$$

where  $c^* = \max\{c_i\}$ . □

This is truly a remarkable result. It states that  $K(x)$  is equivalent to the entropy of a random variable on average. Yet the two metrics came from two different approaches. Algorithmic complexity, however, has one major drawback due to the following theorem.

**Theorem 2.2.8.**  *$K(x)$  is non-computable.*

*Proof.* This is a result of the halting theorem (Theorem 2.2.1). Namely, we do not know whether a program will halt. So, the only way to find the shortest program for  $x$  is to run all programs. Some very short programs, however, do not halt. The following very short program, for example, does not halt:

```
while true do  
  
end
```

Since there is no way of knowing whether a program halts, we cannot determine whether the current shortest program to compute  $x$  is, in fact, the shortest. It follows that algorithmic complexity is not computable. □

Even though  $K(x)$  is not computable, we can approximate it from above. We can certainly wait for a very long time to allow more and more short programs to finish. The longer we wait, the better the estimate is. In other words,  $K(x)$  is *upper semicomputable*, but not computable. In practice, we resort to compression methods to approximate algorithmic complexity.



## 2.2.2 Information Distance

Algorithmic information theory offers a mechanism to define the absolute information content that is based on individual objects. As we have seen, algorithmic complexity is a universal information measure. This makes it even more attractive to define a distance metric based on algorithmic complexity. We want this metric to be a true distance metric so that it satisfies the following conditions:

- $d(x, y) = 0$  iff  $x = y$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$

The above conditions address the symmetry and triangle inequality issues not present in relative entropy. The conditional complexity  $K(x|y)$  is a good candidate, but it is not symmetric, i.e.,  $K(x|y) \neq K(y|x)$ , in general. Bennett et al. [7] addressed this issue by defining an information distance

$$E(x, y) = \max\{K(x|y), K(y|x)\}. \quad (2.21)$$

It is straightforward to verify that (2.21) is a metric in that it satisfies the above conditions. The triangle inequality is satisfied since

$$K(x|z) \leq K(x, y|z) = K(x|y, z) + K(y|z) \leq K(x|y) + K(y|z). \quad (2.22)$$

Furthermore, for any other computable metric  $d$  satisfying the density requirement

$$\sum_{x: x \neq y} 2^{-d(x, y)} \leq 1, \quad (2.23)$$

$E(x, y) \leq d(x, y) + c$ , where the constant  $c$  is independent of  $x$  and  $y$ . This is due to Theorem 2.2.5. Therefore, the information distance accounts for all computable

distance metrics between two objects and it is the minimal up to an additive constant among them. We note that the density requirement eliminates degenerate metrics where  $d(x, y) = 1$  for all  $x \neq y$ . In effect, there can only be a finite number of elements  $x$  from  $y$  at any distance.

In many situations, it is advantageous to use a normalized form of information distance rather than the absolute information distance. In this case, the normalized distance expresses the degree of similarity between two objects. Li et al. [8] defined the normalized information distance, also referred to as the similarity metric, as

$$e(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}. \quad (2.24)$$

## 2.3 Summary

A natural metric to measure the similarity between two random variables is relative entropy. As we have seen, however, relative entropy has many drawbacks. The information distance and the similarity metric based on algorithmic complexity addressed many of these drawbacks. Both are true metrics. Furthermore, they are universal in the sense that they account for all computable metrics between two objects.

## Chapter 3

# Metric Estimation

In order to use relative entropy and the similarity metric in applications, they must be calculated. Relative entropy can be computed trivially when the distributions are known. In many applications, however, the distributions are not known. Digital images, for example, have high degrees of dependencies among pixels and the distributions are difficult to obtain. In these situations, the underlying distribution can only be estimated. As simple as it sounds, estimating relative entropy has been a road block due to the poor performance of existing estimators. We study in depth two estimators based on compression algorithms and show that the estimator based on the Burrows-Wheeler Transform performs better.

Computing the similarity metric proves to be more difficult due to the fact that algorithmic complexity itself is not computable. We can only hope to approximate it using compression. The idea is that the compressed version of an object serves as the shortest representation, and estimates the shortest program, for producing that object. Thus, the size of the compressed object can approximate its complexity. Identifying a good compressor to estimate the similarity metric has been an active area of research. This is critical as bad compressors lead to poor results. In the case

of classification, for example, objects would be put in the wrong class. We examine several well-known compression programs to estimate the similarity metric. We show that related works using the compression ratio to benchmark compressors are flawed as minimal compression ratio does not guarantee good estimates for the similarity metric. We propose a new method to benchmark compressors for estimating the similarity metric and test several compressors on different types of objects including text, images, and DNA sequences. Contrary to results reported in literature, we show that the popular *ppmd* compressor does not perform well as reported. The *lzma* compressor, on the other hand, is a general-purpose compressor that outperforms *ppmd* and task-specific compressors.

### 3.1 Entropy and Relative Entropy Estimation

If the sequence is generated according to an i.i.d. process, then entropy and relative entropy estimation is trivial. In this case, a simple frequency count is sufficient. However, sources for coverttext, such as digital images, exhibit a high degree of correlation between adjacent data samples. For these sources, simple frequency counting would not be adequate and more advanced algorithms are needed. Several algorithms based on popular compression algorithms can be used as entropy estimators for stationary ergodic sources and relative entropy estimators for finite-memory Markov sources.

Universal compression algorithms can optimally compress a sequence generated from a source without having knowledge of the underlying distribution [17]. The Ziv-Lempel algorithm is an example of such algorithms [18]. Given an input sequence, the algorithm parses the sequence into distinct phrases. Each phrase is the shortest phrase not previously parsed. The algorithm works as follows. The algorithm looks along the sequence until it encounters a subsequence that has not been parsed. This subsequence or phrase is added to a dictionary of parsed phrases. The algorithm

Table 3.1: The Ziv-Lempel algorithm on the sequence *abbbbbaaabba*.

Current phrase	Parsed?	Output
<b>a</b> bbbbaaabba	no	a
a <b>b</b> bbbbaaabba	no	b
ab <b>bb</b> baaabba	yes	
abb <b>bb</b> baaabba	no	bb
abbbb <b>a</b> aaabba	yes	
abbbb <b>ba</b> aaabba	no	ba
abbbbba <b>a</b> abba	yes	
abbbbba <b>aa</b> abba	no	aa
abbbbbaaab <b>b</b> a	yes	
abbbbbaaab <b>ba</b>	yes	
abbbbbaaab <b>ba</b>	no	bba

continues where it left off until the entire sequence is parsed. For example, if the sequence is *abbbbbaaabba*, then the algorithm parses it as *a, b, bb, ba, aa, bba* as shown in Table 3.1.

Let  $c(x^n)$  be the number of phrases parsed by the algorithm on a sequence  $x^n$  of length  $n$  generated from a stationary ergodic source  $X^n$ . In the above example,  $c(x^n) = 6$ . Then, due to Wyner and Ziv [19],

$$\frac{c(x^n) \log c(x^n)}{n} \rightarrow \frac{H(X^n)}{n}. \quad (3.1)$$

Similar algorithms exist using slightly different calculations [20, 21].

Ziv and Merhav extended the idea of LZ parsing and proposed a relative entropy estimator for finite-memory Markov sources based on cross parsing of two sequences [2]. Given the two sequences  $x^n$  and  $z^n$ , the algorithm parses  $x^n$  into  $c(x^n|z^n)$  longest phrases that appear in  $z^n$ . The algorithm works as follows. Find the longest prefix of  $x^n$  that appears in  $z^n$ . In other words, find the largest  $m$  such that  $(x_1, x_2, \dots, x_m) = (z_i, z_{i+1}, \dots, z_{i+m-1})$  for some  $i$ . Then, start again from  $x_{m+1}$  until the entire sequence is parsed. For example, let  $x^n = \text{abbbbbaaabba}$  and  $z^n = \text{baababaabba}$ , the algorithm parses  $x^n$  with respect to  $z^n$  to produce

Table 3.2: The Ziv-Merhav algorithm with  $x^n = abbbbaaabba$  and  $z^n = baababaaabba$ .

Matching $x^n$	Matching $y^n$	Output
<b>abbbbaaabba</b>	baababaa <b>abba</b>	abb
abbb <b>baa</b> abba	baababaa <b>abba</b>	bba
abbbbaa <b>abba</b>	baababaa <b>abba</b>	aabba

$abb, bba, aabba$  as shown in Table 3.2.

Let  $c(x^n|z^n)$  be the number of output phrases resulting from parsing  $x^n$  with respect to  $z^n$ . In the above example,  $c(x^n|z^n) = 3$ . The corresponding relative entropy estimator is

$$\frac{1}{n} [c(x^n|z^n) \log n - c(x^n) \log c(x^n)], \quad (3.2)$$

where  $x^n$  and  $z^n$  are sequences generated from finite-order stationary Markov sources.

Although convergence was proven for the above algorithms, the required number of samples can be large, as shown in the sequel. Furthermore, the relative entropy estimator in (3.2) tends to produce negative values especially when the two sequences differ only by a small number of samples. For these reasons, we use estimators based on the Burrows-Wheeler Transform (BWT), described next, rather than the LZ scheme.

### 3.1.1 The Burrows-Wheeler Transform

The Burrows-Wheeler Transform is a block-sorting lossless data compression algorithm [22]. The input to the algorithm is a sequence of  $n$  symbols. The algorithm generates  $n$  cyclic shifts or rotations of the sequence, sorts them lexicographically, and extracts the last symbol of each rotation. The algorithm itself does not compress the sequence, but groups similar symbols together making it easy to compress using simple compression algorithms such as Huffman or run-length coding. For exam-

Table 3.3: An example of the BWT algorithm.

Rotations	Sorted	Output
<i>banana</i>	<i>abanan</i>	<i>n</i>
<i>ananab</i>	<i>anaban</i>	<i>n</i>
<i>nanaba</i>	<i>ananab</i>	<i>b</i>
<i>anaban</i>	<i>banana</i>	<i>a</i>
<i>nabana</i>	<i>nabana</i>	<i>a</i>
<i>abanan</i>	<i>nanaba</i>	<i>a</i>

ple, let the input sequence be *banana*, then the algorithm produces the rotations as shown in Table 3.3.

The key idea to use the BWT as an entropy estimator is the fact that sorting the rotations lexicographically results in grouping together rotations that start with the same sequence, and the output symbol is the symbol that precedes the sequence in each row. If the BWT is performed on the reversed input sequence, for finite memory sources, the algorithm groups together symbols that share the same state, and the conditional distributions can be estimated.

Cai et al. proposed an entropy estimator, and proved almost sure convergence, for stationary ergodic sources using this key observation [23]. The method is as follows:

1. Apply the BWT on the reversed input sequence of  $n$  symbols.
2. Partition the BWT output into  $T$  segments. Uniform segmentation with  $T = \sqrt{n}$  is sufficient.
3. Estimate the first-order distribution within each segment. Denote the number of occurrences of symbol  $a \in \mathcal{X}$  in segment  $j$  as  $N_j(a)$ . The probability estimate of symbol  $a$  in segment  $j$  is then

$$\hat{p}(a, j) = \frac{N_j(a)}{\sum_{b \in \mathcal{X}} N_j(b)}. \quad (3.3)$$

The contribution of segment  $j$  to the entropy estimate is

$$\log \hat{p}(j) = \sum_{a \in \mathcal{X}} N_j(a) \log \hat{p}(a, j). \quad (3.4)$$

4. The entropy estimate is

$$\hat{H}(p) = -\frac{1}{n} \sum_{j=1}^T \log \hat{p}(j). \quad (3.5)$$

The BWT entropy estimator can be extended to act as a relative entropy estimator for finite-memory Markov sources [24]. The method is as follows:

1. Concatenate the two sequences  $x^n$  and  $z^n$  together.
2. Apply the BWT on the reversed concatenated sequence.
3. Partition the BWT output into  $T$  segments according to  $z^n$ . Uniform segmentation according to  $z^n$  is sufficient.
4. Estimate the first-order distribution within each segment. Denote symbols from  $x^n$  with lowercase letters and the same symbols from  $z^n$  with uppercase letters. The probability estimate of symbol  $A$  from  $z^n$  in the  $j$  segment is

$$\hat{q}(A, j) = \frac{N_j(A) + \Delta}{\sum_{B \in \mathcal{X}} N_j(B) + |\mathcal{X}| \Delta}, \quad (3.6)$$

where  $N_j(A)$  is the number of occurrences of symbol  $A$  in segment  $j$ . A small bias  $\Delta > 0$  is needed to deal with the potential for zero probability. The contribution of segment  $j$  to the cross entropy estimate is

$$\log \hat{q}(j) = \sum_{a \in \mathcal{X}} N_j(a) \log \hat{q}(A, j). \quad (3.7)$$

5. Estimate the relative entropy. Averaging over all  $T$  segments,

$$\hat{S}(p||q) = -\frac{1}{n} \sum_{j=1}^T \log \hat{q}(j). \quad (3.8)$$



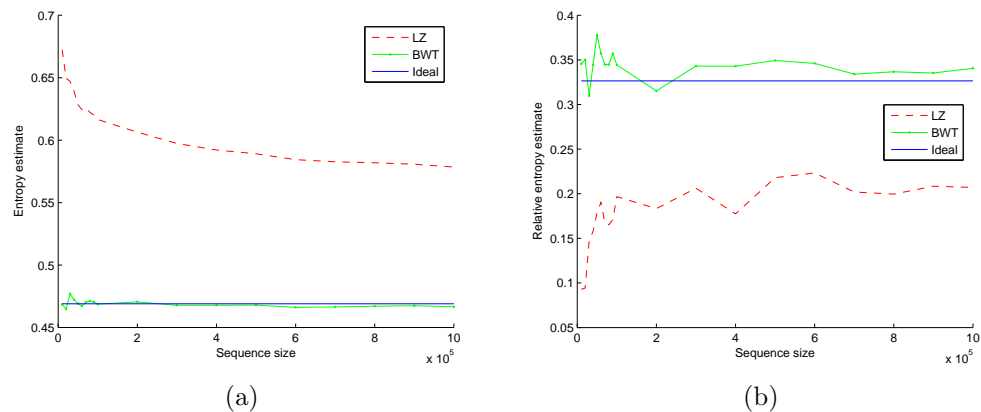


Figure 3.1: Comparison of the LZ and BWT (a) entropy estimators and (b) relative entropy estimators.

The relative entropy estimate is then

$$\hat{D}(p||q) = \hat{S}(p||q) - \hat{H}(p). \quad (3.9)$$

### 3.1.2 Experimental Result

For comparison, we test the LZ and the BWT entropy and relative entropy estimators using binary first-order Markov sources. The result of averaging over 100 runs is shown in Figure 3.1. The ideal curves are shown along with the approximations produced by the estimators. Estimators based on the BWT converge quickly to the ideal curves. The LZ estimators, on the other hand, do not offer good approximations even with  $10^6$  data samples.

## 3.2 Algorithmic Complexity Estimation

In our discussion on algorithmic information theory, we established that algorithmic complexity is not computable (Theorem 2.2.8). We can only hope to approximate it using compression. The idea is that the compressed version of an object serves as the shortest representation, and estimates the shortest program, for producing that object. Thus, the size of the compressed object can approximate its complexity. With a real-world compressor  $Z$ , the similarity metric (2.24) is expressed as

$$NCD(x, y) = \frac{Z(xy) - \min\{Z(x), Z(y)\}}{\max\{Z(x), Z(y)\}}, \quad (3.10)$$

where  $Z(x)$  is the compressed size of  $x$  and  $Z(xy)$  is the compressed size of the concatenation of  $x$  and  $y$ . This technique was used to construct mammalian phylogeny trees and to classify biological sequences [8, 25, 26].

Of course, this opens the question of what compressors are good estimators of the similarity metric. It is not clear what metrics we should use to compare compressors to identify good compressors. Ideally, we would like to use compressors that are *optimal*. A compressor  $Z$  is optimal if

$$\frac{1}{n}Z(x^n) \rightarrow \frac{1}{n}H(X^n). \quad (3.11)$$

Among the optimal compressors, the ones based on LZ77 [27] and LZ78 [18] are the most popular. For a proof of the optimality of LZ77 and LZ78, see [17].

As with our discussion on entropy estimators, optimality does not guarantee good results in practice. In fact, many compressors are optimal, but do they all estimate the similarity metric well? Clearly, optimality alone is not a sufficient condition. It is surprising that this optimality condition, also referred to as compression ratio, is used extensively as a quantity to compare compressors for estimating the  $NCD$ . Compression ratio ignores an important component of  $NCD(x, y)$ :  $Z(xy)$ .  $Z(x)$  might be minimal but  $Z(xy)$  might not be. Even if  $Z(x)$  and  $Z(xy)$  are minimal, that

### Chapter 3. Metric Estimation

is still not a sufficient condition for minimal  $NCD(x, y)$ . Suppose that compressor  $Z_1$  compresses better than  $Z_2$  so that

$$Z_1(x) < Z_2(x) \quad (3.12)$$

and

$$Z_1(xy) < Z_2(xy) \quad (3.13)$$

for all  $x$  and  $y$ . The following relation is not guaranteed to hold:

$$\frac{Z_1(xy) - \min\{Z_1(x), Z_1(y)\}}{\max\{Z_1(x), Z_1(y)\}} < \frac{Z_2(xy) - \min\{Z_2(x), Z_2(y)\}}{\max\{Z_2(x), Z_2(y)\}}. \quad (3.14)$$

For simplicity, let  $Z_i(x) < Z_i(y)$  for  $i \in \{1, 2\}$  so that the above condition becomes

$$\frac{Z_1(xy) - Z_1(x)}{Z_1(y)} < \frac{Z_2(xy) - Z_2(x)}{Z_2(y)}. \quad (3.15)$$

It is straightforward to verify that minimal compression is not sufficient for the above relation. Let  $Z_1(xy) - Z_1(x) = Z_2(xy) - Z_2(x)$ .

Many existing applications of the  $NCD$  or its relatives rely on the compression ratio to identify good compressors [8, 25, 28, 29]. It has been pointed out that compression ratio is not a good quantity to compare compressors [30]. Another approach compared compressors based on how well machine classifiers were able to classify known DNA sequences using the computed  $NCD$  [26]. This, too, is not a good indicator as it depends on the classifier used, its parameters, and the training and testing data sets. The result is therefore dependent on many factors.

Rather than using some indirect metrics, we take our clues directly from the definition of a distance metric:

1.  $d(x, y) = 0$  iff  $x = y$
2.  $d(x, y) = d(y, x)$

$$3. d(x, y) \leq d(x, z) + d(z, y).$$

Specifically, the first two properties, identity and symmetry, are quantities that can be used to compare compressors. Therefore, we evaluate compressors by computing

1.  $NCD(x, x)$
2.  $NCD(x, y)$  and  $NCD(y, x)$ .

Ideally,  $NCD(x, x)$  should be zero. For a practical compressor, however, there will be some overhead associated with storing the additional information of the replica. A good compressor should yield values close to zero. Similarly, we would like  $NCD(x, y)$  to be close to  $NCD(y, x)$ .

### 3.2.1 Experimental Result

We test three compressors: *bzip2*, which is based on the BWT algorithm [22], *lzma*, which is based on the Lempel-Ziv compression algorithm [27], and *ppmd*, which is based on the Prediction by Partial Matching algorithm [31]. All compressors use their best compression setting. This means that the window size is 900 KB for *bzip2*, the dictionary size for *lzma* is 64 MB, and the dictionary size for *ppmd* is 192 MB.

#### Identity Property

We randomly generate English-alphabet text sequences consisting of  $n = 100,000$  characters. For each sequence we compute  $NCD(x, x)$  and the compression ratio on  $x$  for all three compressors. The result of averaging over 100 sequences is shown in Table 3.4.

Table 3.4: The mean  $NCD(x, x)$  and mean compression ratio on  $x$  for three different compressors on randomly generated ASCII texts of 100,000 characters.

	$NCD(x, x)$	Compression ratio on $x$
<i>bzip2</i>	0.368200	4.785793
<i>lzma</i>	0.002287	4.844836
<i>ppmd</i>	0.001778	4.893315

Table 3.5: The mean  $NCD(x, x)$  and mean compression ratio on  $x$  for three different compressors on JPEG images.

	$NCD(x, x)$	Compression ratio on $x$
<i>bzip2</i>	0.865825	1.305979
<i>lzma</i>	0.003059	1.375825
<i>ppmd</i>	0.533291	1.203729

The compression ratio for all three compressors are quite good. Note that the entropy of the source is  $\log 26 \approx 4.7$ . This is the lower bound on the compression ratio. It is clear that *bzip2* has the best compression ratio of the three. It is, however, a poor  $NCD$  estimator. This is not surprising given our above discussion that minimal compression ratio is not a sufficient condition for minimal  $NCD$ .

The dependency of the  $NCD$  on the sequence length  $n$  can be seen in Figure 3.2. The *bzip2* compressor is poor for  $n > 400,000$ , which is expected as its window size is 900,000 KB. The *ppmd* compressor falls apart for  $n > 500,000$ . This is surprise given that its dictionary size is quite large. The *lzma* compressor on the other hand, performs better as  $n$  increases.

We further validate our results using JPEG images. Image dimensions vary from  $1012 \times 1600$  to  $1600 \times 1600$  yielding a minimum of  $1.6 \times 10^6$  pixels. We use the JPEG coefficients as our  $x$ . The result of averaging over 100 images is shown in Table 3.5. Again, *lzma* outperforms the other two even though it has the largest compression ratio of the three.

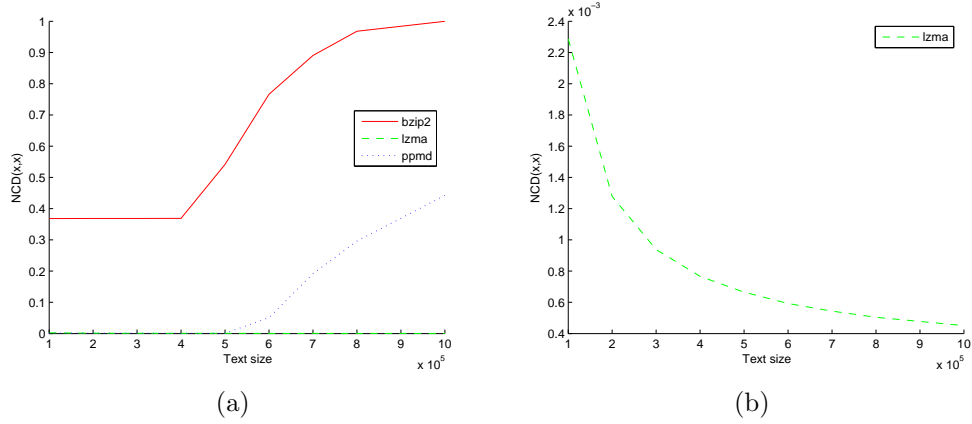


Figure 3.2:  $NCD(x, x)$  at different sequence lengths for (a) all compressors and (b) for the *lzma* compressor.  $NCD(x, x)$  for the *lzma* compressor actually decreases as sequence length increases.

Table 3.6: The  $NCD(x, x)$  and compression ratio on  $x$  for *lzma* and DNACompress on *Plasmodium falciparum* DNA sequences.

Sequence	Length	<i>lzma</i>		DNACompress	
		$NCD(x, x)$	ratio on $x$	$NCD(x, x)$	ratio on $x$
AE001362	947102	0.001121	0.2148	1	0.2082
AE014185	1694445	0.000930	0.2132	1	0.2073
AE014186	2035250	0.000854	0.2181	1	0.2064

We test the *lzma* compressor against a special-purpose compressor DNACompress [32] optimized for DNA sequences. This compression algorithm uses Pattern-Hunter [33] to find similar, not necessarily identical, sequences and encode the differences between them. This method works because DNA sequences are quite similar. We use long DNA sequences of *Plasmodium falciparum*. The result of computing  $NCD(x, x)$  is shown in Table 3.6. Similar to the above experiments using general-purpose compressors, *lzma* has larger compression ratio than DNACompress, but it is a much better  $NCD$  estimator than DNACompress. In fact, DNACompress does not work at all for long DNA sequences.

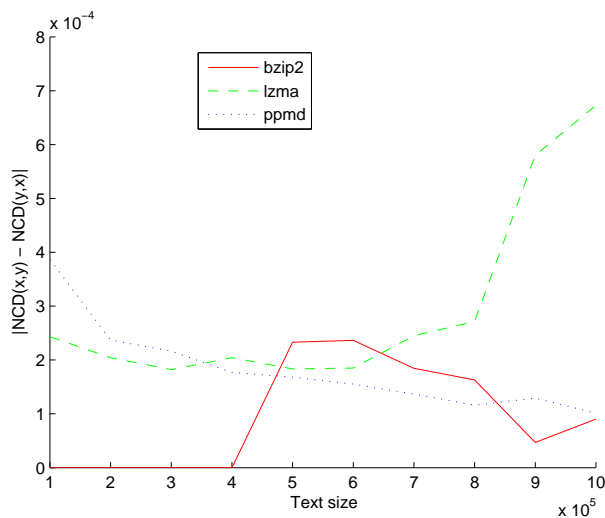


Figure 3.3:  $|NCD(x,y) - NCD(y,x)|$  at different sequence lengths. Each point is an average of 100 sequences.

### Symmetry Property

We now test the same three compressors on the symmetry property. We randomly generate English-alphabet text sequences of various lengths. For each sequence  $x$ , we create a sequence  $y$  by randomly changing every tenth character. This is done on purpose so that the resulting values are not equal to 1. Then,  $NCD(x,y)$  and  $NCD(y,x)$  are computed. From this, we calculate  $|NCD(x,y) - NCD(y,x)|$ . The result is shown in Figure 3.3. Each point is an average of 100 sequences. Since *bzip2* uses the Burrows-Wheeler Transform, it is expected that

$$|NCD(x,y) - NCD(y,x)| = 0$$

for sequence lengths less than 450 KB. Only on longer sequences that *bzip2* starts to diverge from zero. For all compressors, the symmetry does not play a significant role.

### 3.2.2 Related Work

Estimating the similarity metric has been an active area of research. There exist many works using compression programs to estimate the  $NCD$  for phylogeny tree construction [8, 25, 28, 29]. These works rely on short mitochondrial DNA sequences less than 20,000 bases. Compression ratio has been the metric of choice to compare compressors. Besides special-purpose compressors such as GenCompress [28] and DNACompress, *ppmd* was the top performing general-purpose compressor in all of these studies. Our experiments show that *ppmd* does not work well for long sequences of lengths larger than 500,000. We also show that DNACompress does not work for long DNA sequences either. On the other hand, we show that the *lzma* compressor is the best compressor.

Rather than using compression ratio, Ferragina et al. used machine learning as an indicator to compare compressors [26]. This approach, however, is problematic as it is dependent on many factors including the chosen classifier, its parameters, and the training and testing sets. They also used short mitochondrial DNA sequences. Their results showed that the top general-purpose compressor is *ppmd* while *bzip2* is the worst. Gencompress also did well. Our experiments using long sequences show that neither *bzip2* nor *ppmd* are good compressors.

## 3.3 Summary

We have shown better methods for estimating entropy and relative entropy based on the Burrows-Wheeler Transform that outperform estimators based on Lempel-Ziv compression. Estimating the similarity metric is more tricky as algorithmic complexity is not computable. For this, we resort to data compression with the justification that the compressed version of an object serves as the shortest representation, and



### Chapter 3. Metric Estimation

estimates the shortest program, for producing that object. We show that minimal compression ratio is not sufficient to guarantee minimal  $NCD$ . Our experiments validate this idea. Unlike previous studies which show that the general-purpose compressor *ppmd* is a good compressor for estimating the  $NCD$ , we show that it does not work well on long sequences of lengths larger than 500,000. Our results also show that DNACompress, a special-purpose compressor for DNA sequences, also does not work well for long DNA sequences. More importantly, we show two criteria for evaluating compression algorithms: identity and symmetry. This does not have the disadvantages of using compression ratio or other approaches based on machine classification, which depends on many factors including the chosen classifier, its parameters, and the data set. Our results show that only the *lzma* compressor does well in all cases which include text, images, and DNA sequences.

In the next chapter, we study steganography and steganalysis, which sets the stage for the following chapter where we use these estimators to study the security of stegosystems.

## Chapter 4

# Steganography and Steganalysis

Steganography is the term used to refer to a branch of information hiding where the goal is to communicate covertly. The prisoners problem is often used to describe steganography [34]. The prisoners, Alice and Bob, are locked up in separate cells. They wish to communicate with each other in order to come up with an escape plan. They are allowed to send messages to each other, provided the warden examines all messages. If the warden suspects they are devising an escape plan, no further communication between them will be allowed. Alice and Bob are aware of the situation and are prepared for the event that they are arrested. They agree to use a steganographic method or *stegosystem* to communicate with each other to devise an escape plan. Using the agreed upon steganographic method, Alice embeds an escape plan into an innocent looking *coverttext* to create a *stegotext*. The stegotext is then given to the warden who may pass it along to Bob under the conditions described above. They succeed if they can communicate their escape plan without being detected by the warden. The situation depicted is often referred to as steganography with a passive adversary. Steganography with an active adversary refers to the situation where the warden is allowed to alter messages deliberately. We restrict our discussion to the former.

## Chapter 4. Steganography and Steganalysis

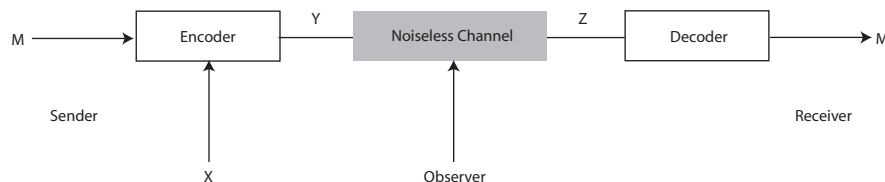


Figure 4.1: A sender uses the encoder of a stegosystem to hide message  $M$  into cocontext  $X$  to produce stegotext  $Y$ , which is sent over a noiseless public channel to a receiver. The receiver uses the decoder to extract message  $M$  from the stegotext.

A stegosystem consists of an encoder and a decoder. The input to the encoder, in general, consists of a cocontext and a message to encode or embed. The encoder embeds the message into the cocontext to produce a stegotext. The sender sends the stegotext over a public channel monitored by a warden who also has access to the cocontext. The receiver uses the decoder to extract the hidden message from the stegotext. While other forms of steganography exist that use side information at the decoder, the described stegosystem is widely popular due to its simplicity and ease of use. This is especially the case for image steganography. We restrict our discussions to stegosystems that do not rely on side information at the decoder, however, our work applies to this case as well. A typical stegosystem is depicted in Figure 4.1.

Although similar, steganography differs from cryptography in the mechanisms they use to communicate privately. In cryptography, only the intended recipients can understand the content, which is otherwise unintelligible to others. Steganography goes one extra step to conceal the presence of the private communication itself. In practice, these two are commonly used together where a secret message is first encrypted and subsequently embedded into a cocontext.

The desire to communicate secretly has a long history with man. For example, Romans shaved the heads of their soldiers and tattooed secret messages onto the bare skin. Once the hair grew back, the soldiers can safely deliver the messages. During World War II, a spy sent the following message:

## *Chapter 4. Steganography and Steganalysis*

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

On the surface, the message appears to be a casual statement. Combining the second letter of every word, however, reveals "Pershing sails from NY June 1." A recent espionage case involving a former State Department official and his wife highlights the use of steganography [35]. The couple was arrested and accused of providing classified information to the Cuban government. They passed information to the Cuban officials by changing shopping carts in a grocery store.

Today, steganography has many more uses beyond covert communication. The popularity of the Internet and the ease of sharing digital content have revolutionized the way we use and distribute information. Digital images, books, music, and videos, for example, are easily accessible via the Internet. This accessibility, however, is also a concern for publishers and copyright owners. For these entities, tracking ownerships and enforcing license agreements, are vital to their business. There is an entire research field called digital rights management devoted to this problem, but certainly, steganography is a part of the solution. Intellectual property owners, for example, can use steganography to identify customers who violate their license agreement. The transmission and storage of medical records benefit from steganography in that patient information and other sensitive data remain hidden and coupled with the records. The storage and transmission of highly sensitive information can use an additional layer of security through the use of steganography. The act of sending encrypted content alone alerts unauthorized users of its significance. However, if the encrypted content is hidden in an innocent coverttext, eavesdroppers will have to take additional steps to figure out which message contains the original encrypted sensitive information.

We study several stegosystems and the techniques they use to hide data. One of

the burning questions in steganography is determining the security of a stegosystem. A commonly used approach is to define the security of a stegosystem in terms of its detectability with respect to a steganalysis detector. We cover common steganalysis techniques and illustrate how they can be used to define the security of a stegosystem.

## 4.1 Steganography

While many forms of steganography exist, in our digital world, it is not surprising that digital steganography is ubiquitous. Digital steganography manipulates digital contents such as images, video, audio, and text documents to embed messages. Among the many digital formats, digital images have emerged as a popular choice of coverttext for concealing messages. Digital images are easily accessible. Anyone with a digital camera can produce digital images. Digital images also tend to have high redundancy. Stegosystems can exploit this redundancy to hide secret messages. The concept of steganography is intriguing on its own. The rich set of embedding techniques used in steganography makes it even more fascinating. In general, it is best to hide a message so that the resulting stegotext resembles the original coverttext. In digital images, this implies, at a minimum, that the two images look similar and that there is no visible artifact in the stegotext, which would otherwise raise suspicion. Each embedding method aims to achieve this and they do so in their own way. Some methods are more advanced and try to preserve statistical properties of the coverttext so that they are not vulnerable to statistical attacks.

### 4.1.1 Least-significant Bit Embedding

One of the simplest embedding methods involves replacing the least-significant bits (LSB) of the coverttext with the message bits. The idea of *LSB replacement* is that it

## Chapter 4. Steganography and Steganalysis

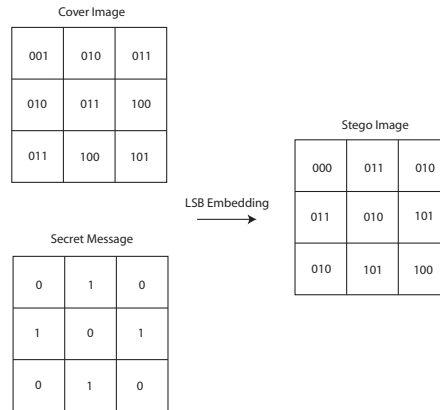


Figure 4.2: LSB replacement overrides the least-significant bits of the coverttext with the message bits to produce a stegotext. LSB embedding results in little visual distortion.

results in minimal visual distortion. For example, a pixel value of 100 does not look different from 101. Thus, given a coverttext and a message to hide, LSB replacement overrides the least-significant bits of the coverttext with the message bits. This process is illustrated in Figure 4.2.

It can be seen that the secret message can be recovered from the stegotext by the intended recipient by extracting the least-significant bits of the stegotext. In practice, the message bits are typically shuffled before embedding to increase randomness and security. Shuffling is done using a random number generator and a private key known only to the sender and recipient. Thus, an observer cannot simply extract the least-significant bits to reveal the message. This would likely result in a random sequence of bits. Only the intended recipient with the appropriate key can extract the hidden message.

Another form of LSB embedding is called *LSB matching*. Rather than overwriting the least-significant bits of the coverttext with the message bits, LSB matching randomly adds or subtracts one from the coverttext value when their least-significant

bits do not match.

LSB replacement is easy to implement and has the added benefit of minimal visual distortion. Yet, the statistical properties of the stegotext are quite different from the coverttext, which makes it vulnerable to detection (a topic we will explore in the next Chapter). The concept of LSB embedding, however, is far-reaching. Today, many advanced stegosystems borrow this LSB concept to hide messages. These stegosystems differ primarily on how they select pixels to change to embed a secret message.

#### 4.1.2 F5

The inherent problem with the simple LSB method is that it introduces statistical irregularities. In particular, LSB embedding tends to equalize the histogram bins of pixel values that differ only in the least-significant bit. Consider two pixel values 6 and 7. Let the number of 6's in the coverttext be  $n_6$  and the number of 7's in the coverttext be  $n_7$ . Assume that  $n_6 \neq n_7$ . Let the message bits consist of an equal number of 0's and 1's. After LSB embedding, the number of 6's is now  $\frac{n_6+n_7}{2}$ . So is the number of 7's. An illustration of this problem is shown in Figure 4.3. This issue becomes more apparent as the message size increases and the method is easily detected by just looking at pixel pairs that share all bits other than the least-significant bit.

A simple method to overcome this problem is to correct the histogram after embedding so that the distribution of the values in the stegotext is identical to the coverttext. First-generation JPEG stegosystems such as Outguess [36] utilized this technique to preserve the distribution of the values in the coverttext. Of course, the disadvantage of such approach is that the embedding capacity is reduced since some coverttext values cannot be used for hiding.

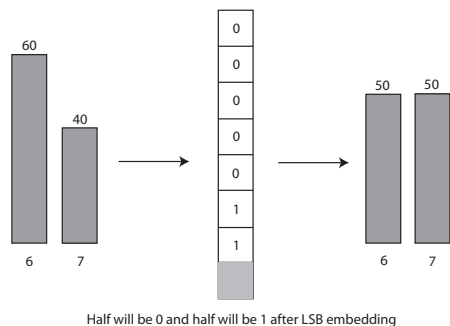


Figure 4.3: LSB embedding equalizes the histogram bins of pixel values that differ only in the least-significant bit. Before embedding, the number of 6's is 60 and the number of 7's is 40. After embedding, the two numbers are likely to be equal when the message size is sufficiently large.

Westfeld approached the histogram problem using an embedding technique called *matrix embedding* in his F5 stegosystem [37]. Matrix embedding was discovered by Crandall [38] and independently by Dijk and Willems [39], Galand and Kabatiansky [40], and later studied by Fridrich and Soukal [41]. Matrix embedding borrows the idea from Hamming codes with the purpose of increasing the embedding efficiency, where efficiency is defined as the number of bits embedded per change. Efficiency is a quantity that is frequently used to determine how good an embedding technique is. In general, the higher the efficiency is, the better the embedding method is. Assuming that the message bits are uniformly distributed, then after LSB embedding, half of the message bits change the covertext, while the other half does not change anything. So, the embedding efficiency for LSB embedding is 2 bits per change. To illustrate the advantage of matrix embedding, consider a simple scenario where we wish to embed two bits  $m_1, m_2$  using three bits  $x_1, x_2, x_3$  from the covertext. Let  $m_1 = x_1 \oplus x_3$  and  $m_2 = x_2 \oplus x_3$ , where  $\oplus$  denotes exclusive or. With this configuration, we have four possible outcomes:



$$\begin{aligned}
 m_1 &= x_1 \oplus x_3, & m_2 &= x_2 \oplus x_3, & \text{no change} \\
 m_1 &\neq x_1 \oplus x_3, & m_2 &= x_2 \oplus x_3, & \text{change } x_1 \\
 m_1 &= x_1 \oplus x_3, & m_2 &\neq x_2 \oplus x_3, & \text{change } x_2 \\
 m_1 &\neq x_1 \oplus x_3, & m_2 &\neq x_2 \oplus x_3, & \text{change } x_3.
 \end{aligned}$$

The point is that, at most, only one change is needed to embed two bits. This type of code is denoted as a triple  $(d_{max}, n, k)$ , where  $n$  is the number of modifiable symbols from the coverttext,  $k$  is the number of message bits to embed, and  $d_{max}$  is the maximum number of changes. In the above example, we have  $(1, 3, 2)$ . The expected efficiency for this code is  $\frac{8}{3} = 2.667$  bits per change, which is larger than the efficiency of LSB embedding. The example shows the value of using matrix embedding as a mechanism to improve efficiency. The efficiency can further be improved by using codes with larger  $k$  such as  $(1, 7, 3)$  which yields an efficiency of 3.429. For the class of  $(1, n, k)$  codes,  $n = 2^k - 1$ . The expected number of changes for each block of  $n$  symbols is  $\frac{n}{n+1}$ . Thus, the expected efficiency is  $\frac{k(n+1)}{n}$ . Note that efficiency is always larger than  $k$ . The penalty for using large  $k$  is low embedding rate, which is defined as  $\frac{k}{n}$ . Low embedding rate reduces the overall capacity of the coverttext.

### 4.1.3 Steghide

A different approach to embedding is based on exchanging rather than overwriting pixels. By exchanging pixels, the histogram is automatically preserved. Hetzl and Mutzel proposed a graph-theoretic stegosystem based on exchanging called Steghide [42]. In this scheme, the coverttext is partitioned into groups of  $k$  elements. Each group is used to hide one message bit. The parity of the least-significant bits of each group is compared with their corresponding message bit. If the two do not match, a node is created for that group. An edge is formed between two groups if an element in one group can be swapped with another element in the other group so

#### Chapter 4. Steganography and Steganalysis

that their resulting parity bits match the message bits. As an example, let the least significant bits of the coverttext be 1,0,1,1,1,0,0,0,1. Let the message be 1,0,0. Let  $k = 3$ . Then, the coverttext is partitioned into three groups of threes:

Group	1			2			3		
LSB	1	0	1	1	1	0	0	0	1
Parity Bit	0			0			1		
Message Bit	1			0			0		

Since the parity bit of group 2 matches the message bit, no change is required. For groups 1 and 3, their parity bits do not match with their message bits. Therefore, one value from group 1 needs to be swapped with another value from group 3. For example, swapping the first value of group 1 with the first value of group 3 yields the desired message bits.

In a real situation, there will be many nodes and many edges. Two nodes can have multiple edges and the resulting graph is dense. There are multiple ways of picking values to swap. In our example, we could have picked the first value of group 1 to swap with the second value of group 3 to obtain the desired message bits. If the coverttext values are binary, then this does not become a major issue. For non-binary coverttext values, choosing the appropriate swapping arrangement is critical. For example, if the first value of group 1 is 101 and the first value of group 3 is 100 while the second value of group 3 is 10, swapping the first value of group 1 with the first value of group 3 is clearly a better choice than the alternative. To guide the search for a good swapping arrangement, each edge has a weight that is based on the distance between the two values to swap. The algorithm tries to find a swapping arrangement on the graph so that the number of matching vertices is maximized while minimizing the total exchanged edge weights. This problem is referred to as the *maximum cardinality minimum weight matching* problem. The general algorithm is not limited to using just the least-significant bits, but can be

applied to a value function  $v$  which maps covertex values to a range of values from 0 to  $m - 1$ . This is accomplished by replacing the parity bits described above with a modulo  $m$  sum operation. The approach is generic and does not limit itself to any specific covertex format. The implementation supports palette images, JPEG images, waveform audio, and  $\mu$ -law audio files.

Other than preserving the histogram of the covertex, Steghide is similar to LSB replacement. It has an expected efficiency of 2 bits per change. Depending on  $k$ , however, the overall capacity of the image is also reduced by a factor of  $\frac{1}{k}$ . For example, when  $k = 2$ , only half of the image can be used to hide messages. The main benefit of this approach is that it automatically preserves first-order statistics and is therefore resilient against statistical attacks that rely on first-order statistics.

#### 4.1.4 Model-based

Other than trying to preserve first-order statistics, the above stegosystems do not make use of any statistical model of the covertex. With a statistical model, it might be possible to build a more secure stegosystem. Sallee proposed a Model-based (MB) stegosystem using statistical modeling and data compression [43]. The relationship between embedding and data compression can be seen as follows. Supposed that we have a perfect compressor for digital images, then the compressor produces random binary sequences for any input image. If the output sequence is given to the corresponding decompressor, the original image is reproduced perfectly. Now, what if a random secret message is given to the decompressor? Surely, a perfectly legitimate image is also produced. The idea is then to construct a model of the covertex to guide a decompressor to produce a sequence from the message that has the desired statistical properties of the model. This sequence can then replace the least-significant portion of the covertex to produce the stegotext. The process is



Figure 4.4: The decompressor uses the statistical model of the coverttext to decompress a random message so that the output sequence has the desired statistical properties of the model. The output sequence can then replace the least-significant portion of the coverttext to produce the stegotext.

illustrated in Figure 4.4. The receiver can recover the message by extracting the sequence from the stegotext and compresses the sequence using the same statistical model.

A specific form of the generalized Cauchy distribution is used to model the distribution of the JPEG coefficients:

$$f_X(x) = \frac{p-1}{2s} \left( \left| \frac{x}{s} \right| + 1 \right)^{-p}. \quad (4.1)$$

This distribution has a closed form equation for its cumulative density function and is preferred over other distributions such as the generalized Gaussian. The model parameters  $s$  and  $p$  are obtained using the maximum likelihood method. Arithmetic coding [44] is used for compression and decompression, which achieves better compression than Huffman.

An important issue with this approach is that the compressor and decompressor must use the same statistical model. If the compressor uses a slightly different model from the decompressor, the message would not be recoverable. Since the coverttext and the stegotext are not identical, some care must be taken so that both compressor and decompressor end up with the same model. To solve this problem, the algorithm puts multiple JPEG coefficients into one bin. For example, coefficient values 1 and 2 are put into the same bin. Each coefficient is then identified by its bin index and its offset in the bin. Coefficient 1 has offset 0, while coefficient 2 has offset 1. These bins are referred to as low precision histograms. The model parameters are fit using

these low precision histograms. The probability of each offset can be calculated using the cumulative distribution of the model. These probabilities can now be used by the decompressor to produce a sequence carrying the message that has the desired statistical properties. Note that the use of the low precision histograms guarantee that the coartext and stegotext end up with the same model. Thus, the message is recoverable by the recipient.

Let the probability of offset 0 be  $p$  and the probability of offset 1 be  $1 - p$ . Assuming that the compressor is ideal, the average number of bits that can be embedded for each coefficient is equal to the entropy:  $-(p \log p + (1 - p) \log(1 - p))$ . The expected number of changes is  $p(1 - p) + (1 - p)p = 2p(1 - p)$ . Therefore, the expected efficiency is

$$\frac{-(p \log p + (1 - p) \log(1 - p))}{2p(1 - p)}. \quad (4.2)$$

For  $0 < p < 1$ , the above equation lower bounded by 2 with equality when  $p = \frac{1}{2}$ . Thus, MB has better efficiency than the simple LSB method.

## 4.2 Steganalysis

Advances in steganography have not gone unnoticed. There exist many interesting steganalysis techniques. In the previous section, we described LSB embedding which tries to minimize visual distortion. It does not guarantee, however, that visual attack is not possible. In fact, early stegosystems employing LSB embedding are vulnerable to visual detection [45]. The premise of LSB embedding is that the human visual system has difficulty discerning the minor differences in the least-significant bits. Thus, the least-significant bits can be treated as noise and can be used to hide messages. The human visual system, however, is also excellent at recognizing patterns. Coupled with the fact that the least-significant bits of color images carry

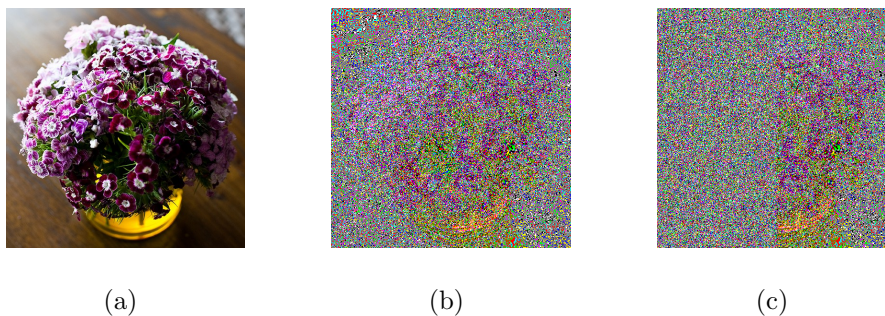


Figure 4.5: Visual attack on LSB embedding exploiting the ability of the human visual system to recognize patterns: (a) cover image (b) least-significant bit image of cover image and (c) least-significant bit image of stego image where the left half is overwritten with the message bits. The outline of the cover image can be seen in the least-significant bits. LSB embedding destroys this pattern.

enough information to convey visible patterns, the human visual system can be used as a steganalysis detector. A naive stegosystem that does not distribute the message bits randomly over the covertext is particularly vulnerable to this visual attack. For example, in Figure 4.5, a cover image is shown along with its least-significant bit image from the three color components. Also shown is the least-significant bit image of the stego image generated using LSB embedding on the left half of the cover image (without shuffling). It can be seen that the least-significant bit image of the cover image contains enough visual information to convey the outline of the cover image. LSB embedding puts noise into the cover image, the resulting stego image destroys this visible pattern. Thus, an observer can exploit this weakness to detect stego images.

Visual attacks work on simple stegosystems that are unaware of this weakness. It is easy to see that this type of attack would not be as successful if a stegosystem employs shuffling to distribute the message bits over the cover image or that the cover image is grayscale. It illustrates, however, that stegosystems that seem secure are in fact vulnerable. More advanced statistical steganalysis techniques exist

exploiting the statistical anomalies left behind by the embedding process. Steganalysis techniques are divided into two broad categories: method-specific and universal. Method-specific techniques aim to defeat a specific steganographic algorithm by exploiting the embedding process of the target stegosystem. Universal steganalysis makes no prior assumption about the embedding method. The goal is to be able to detect the presence of a hidden message embedded by a variety of hiding schemes. The two categories offer complimentary views on steganography and steganalysis.

### 4.2.1 Method-specific Steganalysis

One of the earliest statistical attacks aiming at LSB replacement is the  $\chi^2$  method [45]. The idea was outlined in our discussion on Steganography. Assuming that the message bits are uniformly distributed, replacing the least-significant bits of the coverttext with the message bits has the effect that values that differ only in the least-significant bits are equally distributed after embedding. This concept was illustrated in Figure 4.3. A simple observation shows that the mean of the two frequencies do not change by LSB replacement. Thus, the expected histogram of the values are known for any given image. The method does not need any information about the original image. Given an image, the method constructs the histogram of the image and compare it against the expected histogram using the  $\chi^2$  method. The  $\chi^2$  method measures how close the two distributions are. If the two are close, we can declare the image to be steganographic. This method is simple and it illustrates the idea of statistical steganalysis by exploiting statistical irregularities in the stegotext. The  $\chi^2$  method works well on early stegosystems. Advanced stegosystems such as F5, Steghide, and MB are difficult to detect with this method as they try to preserve first-order statistical properties that this method relies on.

It is known that image pixels are highly correlated. For example, two adjacent

pixels do not vary much. By taking into account the statistical dependency among pixels, better steganalysis attacks can be developed. Fridrich et al.'s RS (Regular/Singular) steganalysis detector uses this principle to detect LSB replacement embedding [46, 47]. Pixels are divided into two groups called the R and S groups based on flipping the LSB and some discrimination function such as variation. The observation is that the frequencies of the two groups approach each other as more message bits are embedded. This is due to the equalization property of LSB replacement described above. The advantage of this approach is that it is possible to estimate the length of the message as well. A similar method based on partitioning groups of pixels into sets is sample pair analysis [48].

Attacks on other stegosystems also try to exploit weakness of their targeted stegosystems. For example, Outguess reserves half of the JPEG coefficients to correct the histogram after embedding. However, Outguess leaves traces of evidence at the boundaries of the 8x8 pixel blocks. By measuring the discontinuities at the boundaries, Fridrich et al. was able to attack Outguess successfully [49].

An attack on F5 is based on estimating the original histogram of the cover image [50, 51]. By decompressing an image, crops it by four pixels in both horizontal and vertical directions, recompresses it using the same quality factor, the original cover image histogram can be obtained. By comparing the histogram of the stego image with the estimated histogram of the original cover image, the method can detect stego images. Furthermore, the two histograms provide an estimate of the number of changes made by F5, which gives an indication on the size of the message.

### **4.2.2 Universal Steganalysis**

The above detectors give a flavor of the diversity of the techniques used to attack stegosystems. They provide valuable insights into the weaknesses of stegosystems.



For large scale steganalysis, however, it is more practical to use universal detectors that can detect a number of stegosystems. Universal detectors do not exploit weaknesses of any particular stegosystem. Rather, they try to identify measures that are natural in cover images and are altered in stego images by the embedding process. Thus, a major challenge in developing universal steganalysis detectors is identifying these natural measures. This becomes even harder given that cover images vary widely.

An approach to universal steganalysis uses image quality metrics such as Mean Square Error and Spectral Magnitude Distance via analysis of variance (ANOVA) technique [52]. However, the use of ANOVA requires knowledge of various hiding schemes *a priori*. It was shown though that these metrics can still be used to detect unknown hiding schemes. A similar approach relies on the correlation between the 7th and 8th bit planes for detecting LSB embedding [53]. The assumption is that there is a correlation between these two bit planes and that embedding destroys this correlation. Several binary similarity measures are used to capture this correlation. These measures are used to train a Support Vector Machine (SVM) classifier to detect stego images.

Another detector aims at detecting spatial-domain stegosystems where cover images are originated from JPEG compressed images [54]. The method works by detecting the compatibility of 8x8 blocks of pixels with a given JPEG quantization table estimated from the image. Without embedding, each block is compatible with the estimated quantization table. Embedding introduces noise and the resulting block is not compatible with the estimated quantization table.

Harmsen and Pearlman model information hiding as an additive process [55]. With the assumption that the embedding process is independent of the cover image, the resulting histogram of the stego image is the convolution of the distribution of the message and the histogram of the cover image. In other words, if  $h_c$  is the histogram

of the cover image,  $f_m$  is the distribution of the message values,  $h_s$  is the histogram of the stego image, then  $h_s = h_c * f_m$ , where  $*$  is the convolution operator. Let  $H_c[k]$ ,  $H_s[k]$ , and  $F_m[k]$  be the discrete Fourier transform of  $h_c$ ,  $h_s$ , and  $f_m$ , respectively. Then,

$$H_s[k] = H_c[k]F_m[k]. \quad (4.3)$$

$H_c[k]$  and  $H_s[k]$  are referred to as the histogram characteristic function (HCF) of  $h_c$  and  $h_s$ , respectively. Since  $f_m$  is a probability mass function,  $|H_s[k]| \leq |H_c[k]|$ .

The HCF is then used to calculate the center of mass (COM):

$$C(H[k]) = \frac{\sum_i k |H[i]|}{\sum_i |H[i]|}. \quad (4.4)$$

It can be shown that

$$C(H_s[k]) \leq C(H_c[k]). \quad (4.5)$$

For color images, each COM is calculated for each color component: red, green, and blue. The three values form a point in a 3-dimensional space, which can be used to detect stego images as the COMs for stego images are smaller than those for cover images.

Fridrich's approach to universal steganalysis utilizes the idea of cropping an image to obtain an estimate of the original image [56]. This technique has been used successfully to attack Outguess and F5 [50, 51]. In this method, an image is cropped 4 pixels horizontally and vertically. The choice of 4 is critical due to the way JPEG compression works on 8x8 blocks. By cropping 4 pixels in both directions, the obtained image has minimal influence from the previous quantization process. This also eliminates the noise introduced by embedding. Thus, the obtained image is a good estimate of the cover image. The method uses several features obtained from the two images such as global histogram, individual histogram for various coefficient

types, dual histograms, variation, and blockiness. These features can then be used to train a classifier to detect stego images.

Lyu and Farid proposed a universal steganalysis method for digital images using wavelet decomposition [57]. Their motivation for using wavelet is that spatial or frequency representation of digital images is insufficient to discriminate between an image and a noise. In the pixel or spatial representation, the histogram is a typical model of images. However, it is possible to have a noisy image that still has the exact histogram of a perfectly looking image. Thus, this pixel representation is not sufficient to discriminate an image from noise. Similarly, a frequency representation based on the Fourier transform, for example, would also not be sufficient to discriminate between an image and a noise pattern. This problem is due to the fact that the spatial representation is localized in space, but not in frequency. The frequency representation, on the other hand, is localized in frequency, but not in space. To get the best of both worlds, wavelet is used to get partial spatial and frequency localization. First-order and higher-order magnitude and phase statistics are collected from the wavelet coefficients. These statistics can then be used to train a classifier to detect stego images.

As we have seen, just as with method-specific steganalysis, there are many different techniques used for universal steganalysis. What is common among these methods is the quest for reliable features that can be used to classify cover and stego images. Ideally, these features should have similar values for cover images and vary for stego images. The differences can then be detected and classified appropriately by a steganalysis detector using these features. In the next section, we discuss a model-based steganalysis method. This method differs from existing methods in that it relies on specific models of images to derive invariant features [58]. The approach was motivated by the work on Benford's law [59].

## 4.3 Model-based Steganalysis

The inherent difficulty with steganalysis is that the original cover image is not available. As such, various attempts have been made to identify image properties that are altered when a hidden message is embedded into an image. Our approach to steganalysis is to analyze the frequency components of images. This motivation comes from the fact that existing steganography techniques primarily try to preserve first-order statistics. Image pixels, however, are highly correlated beyond first-order dependency. This suggests that analysis of the frequency components can reveal clues on whether there is a hidden message in an image. We analyze the frequency components using the DCT coefficients. The choice of using the DCT is due to the fact that the distribution of the coefficients can be modeled as a Laplacian [60]. This gives us a closed form equation which simplifies analysis. We seek to identify the Fourier coefficients of the Laplace distribution and how embedding alters these coefficients. Using these coefficients, we identify invariant features that can be used to detect stego images. We evaluate our detector against three different pixel-domain stegosystems.

### 4.3.1 Cover Image

Let  $C$  be the cover image in the DCT domain. Then, the distribution of the DCT coefficients of  $C$  is modeled as a Laplacian,

$$f_C(x) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}, \quad (4.6)$$

where  $\mu$  and  $b$  are the mean and scale parameter, respectively. The Laplace distribution is symmetric at  $x = \mu$  and the standard deviation of the Laplace distribution is  $\sqrt{2}b$ .

Rather than analyzing the distribution on the entire space in  $\mathbb{R}$ , we analyze the

distribution of  $\hat{C} = C \bmod m$  for some  $m \in \mathbb{R}^+$ . The effect of the modulo operator on the distribution of  $\hat{C}$ ,  $f_{\hat{C}}(x)$ , can be seen through an intermediate function  $f_{\tilde{C}}(x)$ ,

$$f_{\tilde{C}}(x) = f_C(x) * \sum_{k=-\infty}^{\infty} \delta(x - km) \quad (4.7)$$

$$= \sum_{k=-\infty}^{\infty} f_C(x - km), \quad (4.8)$$

where  $*$  is the convolution operator,  $\delta$  is Dirac's delta function.

It is clear that  $f_{\tilde{C}}(x) = f_{\hat{C}}(x)$ ,  $x \in [0, m)$ . Since  $f_{\tilde{C}}(x)$  is periodic, its Fourier coefficients are,

$$a_k = \frac{1}{m} \Phi_C(\omega) \Big|_{\omega_k}, \quad (4.9)$$

where  $\Phi_C(\omega)$  is the Fourier transform of  $f_C(x)$ ,  $\omega_k = \frac{2\pi k}{m}$ .

Without loss of generality, let  $\mu = 0$ , the Fourier transform of  $f_C(x)$  is

$$\Phi_C(\omega) = \frac{1}{1 + \omega^2 b^2}. \quad (4.10)$$

It follows that the Fourier coefficients in (4.9) are,

$$a_k = \frac{1}{m \left( 1 + \left( \frac{2\pi k}{m} \right)^2 b^2 \right)}. \quad (4.11)$$

The Fourier coefficients depend on two parameters: the scaling parameter  $b$ , and the modulo  $m$ .

### 4.3.2 Stego Image

Similar to the derivation of the Fourier coefficients for cover images, we now derive the coefficients for stego images. Our assumption is that embedding is an additive process. The goal is to understand how embedding alters the Fourier coefficients. In

# Chapter 4. Steganography and Steganalysis

the additive model, the stego image is  $S = C + M$ , where  $M$  is the hidden message to embed. The distributions for  $S$  and  $M$  are  $f_S(x)$  and  $f_M(x)$ , respectively. The Fourier coefficients for  $f_S(x)$  are,

$$a_k = \frac{1}{m} \Phi_S(\omega) \Big|_{\omega_k}, \quad (4.12)$$

where  $\Phi_S(\omega)$  is the Fourier transform of  $f_S(x)$ . From probability theory, the distribution of the sum of two independent random variables is the convolution of the distributions. Therefore,

$$f_S(x) = f_C(x) * f_M(x). \quad (4.13)$$

The Fourier transform of a convolution is the product of the transforms. It follows that,

$$\Phi_S(\omega) = \Phi_C(\omega) \Phi_M(\omega), \quad (4.14)$$

and

$$a_k = \frac{\Phi_M\left(\frac{2\pi k}{m}\right)}{m \left(1 + \left(\frac{2\pi k}{m}\right)^2 b^2\right)}. \quad (4.15)$$

The relationship between the cover and stego images is clear from (4.11) and (4.15). Let  $c_k$  be the coefficients in (4.11) and  $s_k$  be the coefficients in (4.15), then the relationship between the two coefficients is

$$s_k = c_k \Phi_M\left(\frac{2\pi k}{m}\right). \quad (4.16)$$

In addition to being dependent on  $b$  and  $m$ , the stego coefficients have a multiplicative factor that is dependent on the message distribution. Thus, the message distribution plays an important role in shaping the Fourier coefficients of stego images and ultimately, its detectability.

### 4.3.3 Invariant Feature

Since  $f_M$  is a density function,  $|\Phi_M(\omega)| \leq 1$ , and we have the following inequality from (4.16)

$$|s_k| \leq |c_k|. \quad (4.17)$$

In other words, the magnitude of the stego coefficients is at most the magnitude of the cover coefficients. This suggests that these coefficients can be used to differentiate cover and stego images. However, a steganalysis detector does not have access to the cover image and hence, cannot compute the cover coefficients for comparison against the stego coefficients.

It is tempting to train a classifier using two sets of coefficients: one from cover images and one from stego images. The resulting classifier can then be used to detect stego images. This, too, would not work well in practice. The problem is that although the distribution of images can be modeled as a Laplacian, the scale parameter  $b$  can vary among images. This parameter is present in (4.11) and (4.15). Thus, the coefficients are not invariant across images. This hints at the limitation of using the coefficients directly for steganalysis. What we want is a feature that is invariant of  $b$  for cover images. Furthermore, this feature must be altered by embedding. Such a feature would allow a classifier to detect stego images without needing access to the original cover images. Our approach is then to eliminate the scale parameter  $b$  from (4.11).

If we choose  $m = \frac{b}{c}$ , where  $1 \leq c < \infty$ , then (4.11) is simplified as,

$$a_k = \frac{1}{m(1 + (2\pi kc)^2)}. \quad (4.18)$$

Define the ratio of two coefficients  $a_k$  and  $a_l$  for a choice of  $c$  as,

$$r_{k,l}(c) = \frac{a_k}{a_l} = \frac{1 + (2\pi lc)^2}{1 + (2\pi kc)^2}. \quad (4.19)$$

For a choice of  $c$ ,  $r_{k,l}(c)$  is invariant of  $b$  for cover images. As such, it is less sensitive to the differences among cover images.

For stego images,  $r_{k,l}(c)$  is,

$$r_{k,l}(c) = \frac{a_k}{a_l} = \left( \frac{\Phi_M(\frac{2\pi k}{m})}{\Phi_M(\frac{2\pi l}{m})} \right) \left( \frac{1 + (2\pi lc)^2}{1 + (2\pi kc)^2} \right). \quad (4.20)$$

The alteration depends only on the distribution of the DCT coefficients of the hidden message. Hence, the distribution of the message plays a crucial role in its detectability.

The ratio for cover images simply says that for any cover image, we can find two points such that their ratio is invariant of the image parameter, namely  $b$ . This is not true for stego images due to the contribution from the message distribution.

#### 4.3.4 Detecting The Hidden Message

We now propose a simple steganalysis method using the above ratio to detect pixel-domain hiding schemes. The derivation of  $r_{k,l}(c)$  makes no reference to any particular steganographic algorithm other than the fact that the hiding method changes the cover additively. Note that this is not a restriction as stegosystems can be viewed as additive embedding. This is desirable, as in practice, we often do not know what embedding method was used to hide messages. Thus, our steganalysis detector is a universal detector.

##### Ratio Selection

From (4.20), it is clear that the performance of the detector depends not only the distribution of the message, but also the choice of  $k$ ,  $l$ , and  $c$ . A steganalysis detector using just one ratio is prone to evasion. After all, with just one ratio, we are only looking at two frequencies in the spectrum. An advanced hiding scheme can easily



## Chapter 4. Steganography and Steganalysis

evade detection. Ideally, we want to sweep the entire spectrum and measure  $r_{k,l}(c)$ . This can be expensive and time consuming. Instead, we can achieve a similar effect using different techniques. For example, for a choice of  $c$ , we vary  $k$  and  $l$ . In theory, we can choose any  $a_k$  and  $a_l$  to form our ratio, but it is advantageous not to choose large  $k$  or  $l$ . This observation is a direct consequence of (4.11). As  $k \rightarrow \infty$ ,  $a_k \rightarrow 0$ . To minimize numerical errors, small  $k$  should be used. Consequently, this approach is limited.

Alternatively, we can fix  $k$  and  $l$  to some small integers and vary  $c$  to obtain a vector  $\mathbf{r}$  as,

$$\mathbf{r} = (r_{k,l}(c_1), r_{k,l}(c_2), \dots, r_{k,l}(c_n)). \quad (4.21)$$

In our experiments,  $\mathbf{r}$  is,

$$\mathbf{r} = (r_{2,1}(5), r_{2,1}(10), r_{2,1}(15), r_{2,1}(20), r_{2,1}(25)). \quad (4.22)$$

In general,  $r_{k,l}(c)$  is complex. Therefore, we use its magnitude  $|r_{k,l}(c)|$  instead. Our choice of  $c$  covers a wide range of frequencies. If we assume that the message has a Gaussian distribution, then its Fourier transform is also a Gaussian. As a result, (4.20) would vary for different values of  $c$ . By detecting this variation, we should be able to detect stego images.

### Experimental Result

We evaluate our steganalysis detector against three stegosystems: Spread Spectrum Image Steganography (SSIS) [61], Steghide [42], and LSB replacement. Our image database consists of 2460 images collected from various Internet image repositories. Many of the images are photos contributed by different photographers using different cameras. Images span a wide range of scenes. Some images are computer generated. Image dimensions vary from 228x451 to 640x640. All images are in JPEG format.

For our experiments, we convert all images into gray-scale BMP format. For each image, we apply the 16-by-16 DCT to obtain the DCT coefficients. We then calculate  $\mathbf{r}$ . To do this, we apply the modulo operator on the DCT coefficients. Then, the resulting coefficients are binned into 100 bins to obtain the frequency or distribution. Next, the FFT is applied to the distribution to obtain  $a_k$  and, consequently,  $\mathbf{r}$ .

Stego images are generated from the cover images using SSIS, LSB, and Steghide with random messages generated at 100%, 50%, 25%, and 5% of the cover capacity. The cover capacity is the maximum message size, in bytes, that can be embedded. This value is image and hiding method dependent. Steghide's executable has the option to estimate the cover capacity. We use this feature to generate random messages with sizes close to the estimated capacity. At 100% cover capacity, the average message sizes are 28 KB, 28 KB, and 18KB for SSIS, LSB, and Steghide, respectively.

The messages for SSIS are generated using a Gaussian distribution  $\mathcal{N}(0, 4.5^2)$ . This yields average SNRs of -23 dB, -25 dB, -28 dB, and -35 dB for message sizes at 100%, 50%, 25%, and 5% of the cover capacity, respectively. This is consistent with the smallest SNR Marvel et al. did in their experiments at -23 dB. We assume SSIS fails to recover the message for smaller SNR. Thus, we embed beyond the capability of SSIS. For LSB, the message consists of randomly generated 0's and 1's of equal probability. For each stego image, we compute the vector  $\mathbf{r}$  as described above for cover images.

The invariant ratio in (4.19) suggests that cover images have similar values. This clustering property can be seen in the 3-dimensional scatter plot using three different ratios  $|r_{2,1}(5)|$ ,  $|r_{2,1}(10)|$ , and  $|r_{2,1}(15)|$  on 100 random images as shown in Figure 4.6. Also shown are the ratios for stego images generated using LSB and Steghide with message size at 100% of the cover capacity. To strengthen the point, the stego images are generated from a different set of 100 cover images. The ratios for SSIS stego images are not shown because it occupies a region that is relatively far from the other

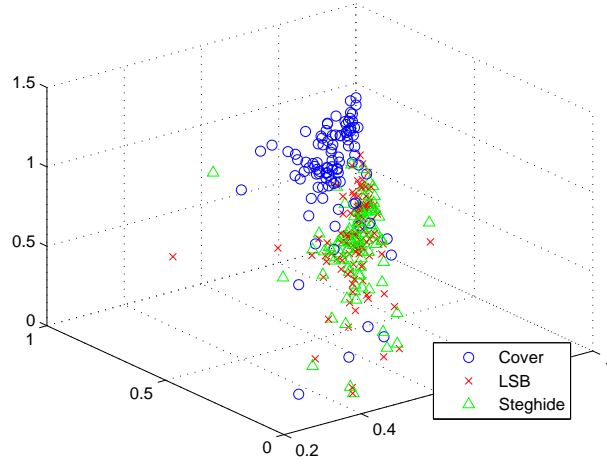


Figure 4.6: The scatter plot of  $|r_{2,1}(5)|$ ,  $|r_{2,1}(10)|$ , and  $|r_{2,1}(15)|$  for 100 random cover images and stego images using LSB and Steghide with message size at 100% of the cover capacity. The stego images are generated from a different set of 100 cover images.

two in the three dimensional space. From the figure, we note that  $|r_{2,1}(c)| \approx 0.7$  for the cover images. The ideal value from (4.19) should be approximately  $\frac{1}{4}$ . This discrepancy might be an artifact of numerical binning or the fact that the Laplace distribution is not a perfect representation of the distribution of the DCT coefficients. Perhaps using a subset of the DCT coefficients might yield better estimates. Nevertheless, the clustering property is a good candidate for applying machine classification techniques such as Support Vector Machine (SVM). We use Joachims SVM implementation for our classifier [62]. Half of the cover images and the corresponding half of the stego images are used for training. The other half is used for testing. For training, we use the default radial basis kernel (RBF) with  $\gamma = 1$ . We do not optimize any parameters.

As suggested by Chandramouli and Memon, the security of a stegosystem can be

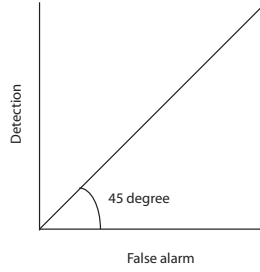


Figure 4.7: A detector's false-alarm probability versus detection probability for a perfectly secure stegosystem.

viewed from a steganalysis perspective via a receiver operating characteristics (ROC) plot [63]. The ROC shows the performance of the detector by comparing its false alarm probability versus its detection probability. The detection probability is the number of correctly identified stego images over the total number of stego images. The false alarm or error probability is the number of cover images classified as stego over the total number of cover images. Thus, the ROC shows the trade-off between the false-alarm and detection probabilities. Let  $\alpha$  be the false-alarm probability and  $\beta$  be the detection probability. If we assume the probability of an image carrying a message is equal to the probability of an image not carrying a message, then the average probability of error is

$$p_e = \frac{1}{2} (\alpha + (1 - \beta)). \quad (4.23)$$

If  $\alpha = \beta$ , then  $p_e = 0.5$ . In this case, the detector is making random guesses and is no better than flipping a fair coin. Thus, when the ROC is a straight line at 45 degrees, a stegosystem is said to be perfectly secure with respect to the specified steganalysis detector. The ROC curve of a perfectly secure stegosystem is depicted in Figure 4.7.

Shown in Figure 4.8(a) is the ROC curve for our detector. It also shows the ROC curves corresponding to the three stegosystems. SSIS is easily detectable and Steghide is harder to detect. Similarly, shown in Figure 4.8(b) are the ROC curve

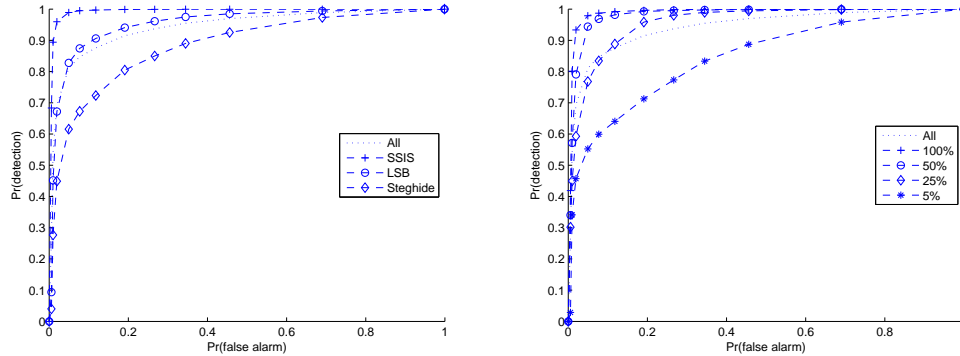


Figure 4.8: The ROC curves for our classifier along with the SSIS, LSB, and Steghide hiding schemes (a). The ROC curves for four different message sizes at 100%, 50%, 25%, and 5% of the cover capacity (b).

for our classifier and the ROC curves for different message sizes at 100%, 50%, 25%, and 5% of the cover capacity. The smaller the message size, the more difficult it is to detect. Overall, our classifier performs best at 5% false alarm or 95% cover detection, where the total probability of error (false alarm plus false positive) is minimal. The detail breakdowns for each stegosystem and message size at 95% cover detection accuracy is shown in Figure 4.9. Clearly, SSIS is easy to detect. At 5% of the cover capacity or 1.4 KB message size, we can still detect 97.5% of the SSIS stego images and 58.5% of the LSB stego images. For Steghide, at 5% of the cover capacity or 0.9 KB message size, we can detect about 10% of the stego images.

## Related Work

To put our results into perspective, we compare our method against other universal pixel-domain steganalysis methods. However, it is difficult to have a direct comparison against existing methods due to the differences in the image set, the number of images, the stegosystems used, message size, and the classifiers and their parameters.

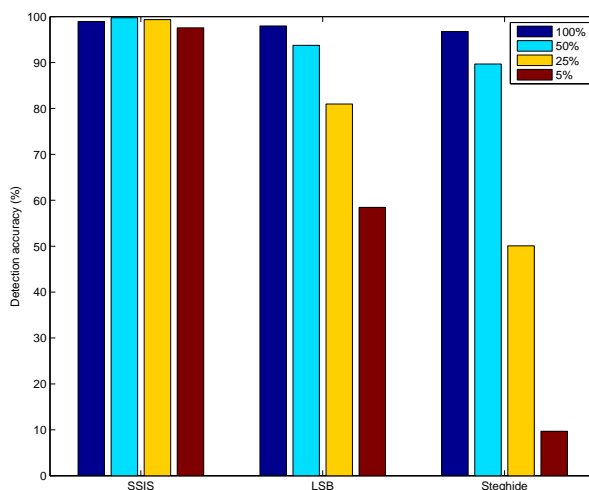


Figure 4.9: The detection accuracy for different hiding algorithms and message sizes at 95% cover detection accuracy.

For example, a detector using only simple stegosystems tends to lead to better results than another detector that uses a mixture of stegosystems including more advanced ones. Different message sizes would also lead to bias results. Machine classification methods are usually used to classify coverttexts and stegotexts. Comparing results using different classifiers is difficult. To further complicate things, some reported results optimized their classifier parameters to obtain optimal results. Never the less, we will point out some important differences as we go.

Harmsen and Pearlman used histogram characteristic function and center of mass to classify cover and stego images [55]. They tested their method on 24 color images using SSIS, LSB and DCT hiding schemes with message sizes at 100% of the cover capacity. This is clearly different from our experiment. In particular, using only message size at 100% of the cover capacity is advantaguous in their experiment. As we have seen from our results, larger message sizes are easier to detect than smaller

message sizes. Using only 100% of the cover capacity leads to better detection probabilities and lower false alarm probabilities. Regardless, at 94.88% cover detection, they detected 96.87% of SSIS stego images, and 100% LSB stego images. In contrast, our detection accuracies at 95% cover detection with message size at 100% of the cover capacity for SSIS and LSB are 98.94% and 97.97%, respectively.

Lyu and Farid used wavelet decomposition to capture higher-order image statistics to classify cover and stego images [57]. They tested their method on 40,000 images using 432 features. Though they focused on detecting JPEG hiding methods, they also experimented with LSB hiding on TIFF images. However, for the TIFF images, only LSB hiding was used. This is no longer blind or universal steganalysis as it is now tuned to a specific hiding scheme. Similar to the above situation, this gives an advantage in their experiment. At 99% cover detection, they detected 72.3%, 52.9%, 11.3% and 1.2% LSB stego images with message sizes at 100%, 89%, 22%, and 5% of the cover capacity, respectively. In contrast, at 95% cover detection, our blind detector detects 97.97% , 93.74%, 80.98%, and 58.46% LSB stego images with message sizes at 100%, 50%, 25%, and 5%, respectively. Since the cover detection percentages are not the same in the two experiments, it is better to compare the total probability of error, which is the sum of the false alarm plus the missed detection. Their total errors are 28.7% and 99.8% for message sizes at 100% and 5% of the cover capacity, respectively. Our total errors are 7.03% and 46.54% for message sizes at 100% and 5% of the cover capacity, respectively. Clearly, using a large set of features or higher-order statistics does not necessarily give better results. Our blind steganalysis method uses only five features, but results in better performance. The difference is that the feature selection using wavelet decomposition is arbitrary.

## 4.4 Summary

There are many interesting embedding techniques employed in steganography. As we have seen, LSB is a simple embedding method aiming at minimizing visual distortion. We have shown three other stegosystems using three different approaches to embedding. In particular, F5 uses matrix embedding, Steghide uses the concept of exchanging rather than overwriting, while MB uses statistical modeling and compression for embedding.

Other embedding methods that are not based on LSB embedding exist. Spread Spectrum Image Steganography (SSIS) is one example [61]. In SSIS, a Gaussian noise carrying the hidden message is added to the cover image. The idea is that the noise is not easily distinguishable from common natural image distortions. The method uses image restoration and error-correcting codes to extract the hidden message without requiring the original cover image.

We have seen a breadth of techniques used in steganalysis. The premise of steganalysis is that embedding leaves traces of evidence that can be used to detect the presence of the hidden message. Least-significant bit embedding, for example, tries to exploit to weakness of the human visual system by embedding in the least-significant bit plane so that it is difficult for the human eyes to pick up any distortions. A simple visual attack, however, shows that it is possible to visually inspect an image to determine whether it contains a hidden message. This reinforces the idea that steganalysis is possible, but at the same time forces steganography to adapt to become more evasive. Advances in steganography forces the development of more advanced statistical attacks. Development of new attacks and improvement of existing detectors are ongoing active research areas [64, 65, 66, 67, 68].

The availability of steganalysis detectors makes it possible to understand the security of stegosystems from a steganalysis perspective. The security of a stegosystem



with respect to a detector is defined in terms of the false-alarm and detection probabilities, which is obtained from the ROC plot. While this approach to analyzing the security of stegosystems aims to be practical, it suffers many problems in practice. As we have seen, the detection probabilities of stegosystems vary among detectors. The detectors are also sensitive to the training process, which leads to the problem that the same detector trained using two different training data sets might produce two different ROC curves. Ultimately, the dependency on a steganalysis detector is problematic as it is difficult to interpret results. In the next chapter, we re-examine the security issue and present a general method for benchmarking stegosystems.

## Chapter 5

# Benchmarking Stegosystems

The most critical requirement of a stegosystem is security. If the warden is able to differentiate coverttexts from stegotexts, the stegosystem is rendered useless. There exist many works that try to define steganographic security using different quantities [5, 63, 69, 70, 71, 72, 73]. The problem with these definitions, however, is that they only consider one aspect of security based on the chosen quantity. Therefore, comparing the security of stegosystems using different definitions would be difficult. For example, it could be that a stegosystem is shown to be more secure than another stegosystem under relative entropy, but is less secure with respect to a different security definition. This immediately raises an issue of the choice of security measure.

We propose a universal steganographic security metric based on algorithmic complexity theory that overcomes this problem [74]. This metric is the similarity metric. It is not only a metric, but it is also universal in the sense that it is asymptotically minimal among all computable metrics between two objects. Therefore, it accounts for all similarities between two objects. This is significant as it illuminates the true security of a stegosystem. That is, if a stegosystem is perfectly secure with respect to some metric, then it is also perfectly secure with respect to this metric. Analo-

gously, a stegosystem with a large similarity metric is inherently insecure as all other computable metrics must also be large. In contrast, relative entropy, a widely accepted steganographic security definition, only takes into consideration the statistical similarity between two random variables.

As an application, we present a general-purpose method for benchmarking stegosystems. Unlike existing works [63, 75], our method does not rely on any steganalysis detector. As a result, it does not have many of the drawbacks such as classifier parameters and training associated with using steganalysis detectors. Furthermore, it is general in the sense that it is not restricted to any coverttext medium and therefore, can be applied to a wide range of stegosystems. In contrast, the approach proposed by Pevný and Fridrich is limited to JPEG images [76]. Our method requires only a security metric and the steganographic capacity of a stegosystem. We benchmark several image stegosystems using the newly proposed similarity metric as the security metric. For comparison, we also use relative entropy as the security metric. We demonstrate the generality of our approach to show that it is not restricted to JPEG images or file formats by applying it to pixel-domain stegosystems. The results show the security and capacity trade-off of stegosystems and serve as a benchmark for stegosystems. More importantly, the results show the true security limits of stegosystems regardless of the chosen security metric or the existence of steganalysis detectors. In other words, this makes it possible to show that a stegosystem with a large similarity metric is inherently insecure, even if it has not yet been broken.

## 5.1 Steganographic Security

Among the different models defining steganographic security, Cachin’s definition is the most widely used [5, 6]. This model quantifies the security of a stegosystem in terms of the relative entropy between the coverttext and stegotext distributions.

## Chapter 5. Benchmarking Stegosystems

Let the probability distributions of the coverttext and the stegotext be  $p_X$  and  $p_Y$ , respectively. Then, a steganographic system is said to be *perfectly secure* if

$$D(p_X||p_Y) = \sum_x p_X(x) \log \frac{p_X(x)}{p_Y(x)} = 0, \quad (5.1)$$

and  $\epsilon$ -secure if

$$D(p_X||p_Y) \leq \epsilon, \quad (5.2)$$

where the logarithm is base 2.

In a perfectly secure stegosystem, the coverttext and the stegotext are statistically identical. The warden, who has full knowledge of the distribution of the coverttext, cannot statistically differentiate between the two and allows the stegotext to go through. Consequently, Alice and Bob can communicate their escape plan without being detected by the warden.

Differentiating between coverttexts and stegotexts is ultimately a hypothesis testing problem. In hypothesis testing, two types of errors can occur: type I and type II. In a type I error, or false-alarm error, the warden mistakenly asserts that the object is a stegotext when in fact it is a coverttext. Denote this probability by  $\alpha$ . In a type II error, the warden mistakenly asserts that the object is a coverttext when in fact it is a stegotext. The probability of type II error occurring is  $\beta$ . Using the following theorem, the relationship between the error probabilities and the security of a stegosystem can be developed.

**Theorem 5.1.1** (Information Processing Inequality). *Let  $p_X$  and  $q_X$  be two distributions over alphabet  $X$ . Let  $Y = f(X)$ , where  $f$  is any deterministic function, then*

$$D(p_X||q_X) \geq D(p_Y||q_Y). \quad (5.3)$$

*Proof.*

$$D(p_X||q_X) = \sum_x p_X(x) \log \frac{p_X(x)}{q_X(x)} \quad (5.4)$$

$$= \sum_y \sum_{x:y=f(x)} p_X(x) \log \frac{p_X(x)}{q_X(x)} \quad (5.5)$$

$$\geq \sum_y \left( \sum_{x:y=f(x)} p_X(x) \right) \log \frac{\sum_{x:y=f(x)} p_X(x)}{\sum_{x:y=f(x)} q_X(x)} \quad (5.6)$$

$$= \sum_y p_Y(y) \log \frac{p_Y(y)}{q_Y(y)} \quad (5.7)$$

$$= D(p_Y||q_Y), \quad (5.8)$$

where inequality (5.6) is due to the log-sum inequality.  $\square$

Thus, deterministic processing cannot increase the relative entropy between two distributions. Relating back to our hypothesis testing problem, the warden ultimately makes a binary decision: 0 for covertexts and 1 for stegotexts. For covertexts, the probabilities of 0 and 1 are  $1 - \alpha$  and  $\alpha$ , respectively. For stegotexts, the probabilities of 0 and 1 are  $\beta$  and  $1 - \beta$ . The relative entropy between these two distributions is

$$D(\alpha||\beta) = (1 - \alpha) \log \frac{1 - \alpha}{\beta} + \alpha \log \frac{\alpha}{1 - \beta}. \quad (5.9)$$

Since we can view the decision process of the warden as a deterministic function, according to Theorem 5.1.1, we have

$$D(\alpha||\beta) \leq D(p_X||p_Y). \quad (5.10)$$

For the special case when  $\alpha = 0$ , the above equation yields a lower bound on the type II error probability,

$$\beta \geq 2^{-D(p_X||p_Y)}. \quad (5.11)$$

Defining the security of a stegosystem using relative relative entropy solves the problems associated with the approach based on steganalysis detectors and allows for a “fair” analysis of stegosystems. We are careful to emphasize “fair” because relative entropy is not without criticism. As we have seen, there are many different security definitions. What makes one definition better than another? It could be that a stegosystem is shown to be more secure than another stegosystem under relative entropy, but is less secure with respect to a different security definition. This immediately raises an issue of the choice of security measure. What is needed is a notion of universal steganographic security that takes into account all computable metrics. To overcome this problem, we define the security of a stegosystem using the similarity metric.

**Definition** Let  $x$  be a covertext and  $y$  be the corresponding stegotext. A stegosystem is called *universal perfectly secure* if

$$e(x, y) = 0, \quad (5.12)$$

within a additive factor, and is called *universal  $\epsilon$ -secure* if

$$e(x, y) \leq \epsilon. \quad (5.13)$$

The universality of this steganographic security definition is justified as follows. If a stegosystem is perfectly secure with respect to some normalized metric  $d(x, y)$ , i.e.,  $d(x, y) = 0$ , then it is also perfectly secure within an additive factor with respect to the similarity metric due to the fact that  $e(x, y) \leq d(x, y) + \delta$ . We note that the additive term  $\delta$  is  $O(1/\min\{K(x), K(y)\})$ . For long sequences, it is negligible. In those cases, if  $d(x, y) = 0$ , then  $e(x, y) \rightarrow 0$  as the length goes to infinity. Furthermore, a stegosystem with a large similarity metric is inherently insecure as all other computable metrics must also be large. This implies that the stegosystem can be broken with any of these metrics.

## 5.2 Steganographic Capacity

While security is an important measure of stegosystems, it is incomplete by itself; it does not specify how much information can be embedded. A stegosystem that can only embed one bit of information into a covertext has limited uses as compared to a stegosystem that can embed one thousand bits of information into the same covertext. Analogously, a compressed image typically does not contain as much information as its raw counterpart. Embedding the same message into these two images might produce different security results or might not be possible altogether. Therefore, the amount of information that can be embedded is dependent on the source of covertexts. Clearly, a notion of steganographic capacity is needed to quantify the amount of information that can be hidden.

The steganographic capacity problem has been an active area of research. Moulin and O’Sullivan studied the channel capacity problem for Bernoulli and Gaussian sources [77]. Similarly, Moulin and Wang analyzed the steganographic capacity of a stegosystem where the covertext is Bernoulli( $\frac{1}{2}$ ) and the distortion constraint is a Hamming distance [78, 79]. Comesaña and Pérez-González derived the capacity of a stegosystem where the covertext and the message are Gaussians [80]. Harmsen and Pearlman defined steganographic capacity for arbitrary channels [81] using the general channel capacity formula developed by Verdú and Han [82]. Chandramouli and Memon proposed that steganographic capacity should be defined from a steganalysis perspective [63, 83].

We define the steganographic capacity as the maximum amount of information that can be communicated subject to being perfectly secure. From an information-theoretic perspective, this quantity is defined using the channel capacity. Using the variables from Figure 4.1, the steganographic capacity is,

$$C = \max_{p_Y: p_Y = p_X} I(Y; Z), \quad (5.14)$$

where  $I(Y; Z)$  is the mutual information between  $Y$  and  $Z$ . Since this is a noiseless channel,

$$I(Y; Z) = H(Y) \quad (5.15)$$

$$= H(X), \quad (5.16)$$

where the last equality is due to the perfectly secure constraint. Thus, for a fixed covertex source  $X \sim p_X$ , the maximum rate of information that can be communicated in a perfectly secure stegosystem is bounded by  $H(X)$ . This has a natural interpretation in that the maximum amount of information is the entropy of the covertex.

The information-theoretic security and capacity definitions apply to stationary ergodic covertex sources. In this case, the relative entropy rate is used to define the perfectly secure constraint as,

$$\lim_{n \rightarrow \infty} \frac{1}{n} D(p_X || p_Y) = 0. \quad (5.17)$$

For a fixed covertex source  $X^n \sim p_X$ , the steganographic capacity of a perfectly secure stegosystem is defined using the entropy rate,

$$C = \lim_{n \rightarrow \infty} \frac{1}{n} H(X^n). \quad (5.18)$$

Parallel to the information-theoretic model, we derive the universal steganographic capacity using algorithmic complexity. Denote the encoder of a stegosystem as  $f$  which embeds a message  $m$  into a covertex  $x$  to produce a stegotext  $y = f(m, x)$ . For a fixed covertex  $x$ , we define the steganographic capacity of a stegosystem as the maximum message complexity subject to being perfectly secure:

$$C = \max_{m: e(x, f(x, m))=0} K(m). \quad (5.19)$$

The above definition is intuitive in the sense that the complexity of an object is a measure of the absolute information content of that object. Thus, the capacity of a



stegosystem is the maximum information content that can be hidden while remaining perfectly secure. This leads to the following intuitive result that the capacity is upper bounded by  $K(x)$ .

Due to the perfectly secure constraint, it follows that

$$K(x|y) = K(x, y) - K(y) = 0 \quad (5.20)$$

$$K(y|x) = K(x, y) - K(x) = 0. \quad (5.21)$$

Thus,  $K(y) = K(x)$ . Since  $y$  carries the information content of  $m$ ,  $K(y) \geq K(m)$ . Therefore,  $K(m) \leq K(x)$ , yielding

$$C = K(x). \quad (5.22)$$

It is interesting to note that the information-theoretic steganographic capacity in (5.16) and algorithmic complexity steganographic capacity in (5.22) are the information content of the coverttext. This is expected because the stegotext cannot contain more information than the coverttext without being detectable. Furthermore, both quantities are equivalent due to the fact that the complexity of a random variable is approximately equal to its entropy.

### 5.3 Experimental Result

Our general method for benchmarking stegosystems works as follows. For a fixed coverttext, the analysis is done by embedding random binary messages of various sizes and computing the corresponding security metric. The sizes, in bits, are determined according to  $\alpha C$ , where  $C$  is the steganographic capacity and  $\alpha$  increases until the stegosystem fails to embed the message due to its large size. The result shows the security and capacity trade-off of the stegosystem and can be used to benchmark stegosystems. It is important to emphasize that our method is independent of the

choice of steganalysis detector. The dependency on a chosen steganalysis detector is undesirable due to the following reasons. Steganalysis detectors, in general, rely on machine learning to classify coartexts and stegotexts. Therefore, the results are dependent on many factors including the performance of the chosen detector, its parameters, the training process, and the data set. To avoid these problems, our method does not make use of any steganalysis detector. A similar work was proposed by Pevný and Fridrich [76]. The authors used a nearest neighbor entropy estimator to calculate relative entropy. However, due to large errors in the estimates, the Maximum Mean Discrepancy was used to quantify security. They also used bits per non-zero coefficients or *bpac* to define steganographic capacity of JPEG images. This limits their method to JPEG images. In contrast, our method is not restricted to any coartext medium and therefore, can be applied to a wide range of stegosystems. For comparison, we use both relative entropy and the similarity metric in our experiments.

### 5.3.1 Information-Theoretic Result

Before we analyze real stegosystems, it is illustrative to analyze a simple stegosystem. In this setting, coartexts of length  $n$  are generated according to a uniform distribution over  $N$  consecutive integer values. The stegosystem adds 1 or -1 to a coartext value to hide a message bit. We assume that the embedding process is independent of the coartext. This is a variation of the simple least-significant bit embedding method. Assuming that the message bits are uniformly distributed so that the number of 1's is equal to the number of -1's, then the distribution of the message is,

$$p_M(x) = \frac{1}{2}\alpha H(X)\delta(x+1) + (1 - \alpha H(X))\delta(x) + \frac{1}{2}\alpha H(X)\delta(x-1), \quad (5.23)$$

where  $H(X)$  is the entropy of the coverttext,  $\delta(x)$  is Dirac's delta function and  $0 \leq \alpha \leq \frac{1}{H(X)}$ . The  $\alpha$  parameter controls how much information is embedded. At  $\alpha = 0$ , the coverttext is identical to the stegotext. At  $\alpha = 1$ , the largest possible message is embedded into the coverttext. The ideal relative entropy is,

$$D(p_X||p_Y) = -\frac{2}{N} \log \left( 1 - \frac{1}{2}\alpha H(X) \right). \quad (5.24)$$

We analyze this simple stegosystem using the BWT estimators. The parameters are  $N = 5$ ,  $n = 10^5$ . We vary  $\alpha$  in increments of 0.01 starting at 0.01. The plot of the result is shown in Figure 5.1. The estimate curve is an average of 100 runs. Also shown is the ideal relative entropy values from (5.24). The estimates closely match the ideal values. The convexity of relative entropy can also be seen from the plot.

### JPEG Steganography

We apply our proposed method to analyze the performance of several practical JPEG and pixel-domain image stegosystems. Our image set consists of 812 grayscale JPEG images taken using different cameras spanning a wide range of scenes. These images have varying compression qualities ranging from 50% to 97%. Image dimensions vary from  $1012 \times 1600$  to  $1600 \times 1600$  yielding a minimum of  $1.6 \times 10^6$  pixels.

For each image, the JPEG coefficients are converted into a one-dimensional array by converting each 8x8 block into a one-dimensional sequence and then concatenating all sequences to form the final array as illustrated in Figure 5.2. The entropy  $H(X^n)$

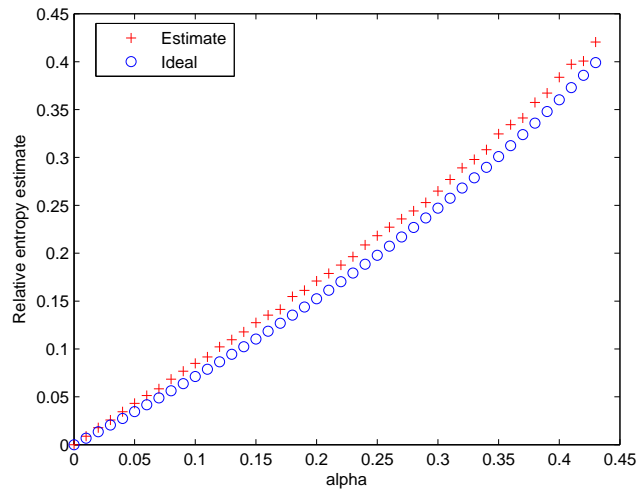


Figure 5.1: The security and capacity trade-off of the simple stegosystem. The relative entropy estimates are good approximations of the ideal values.

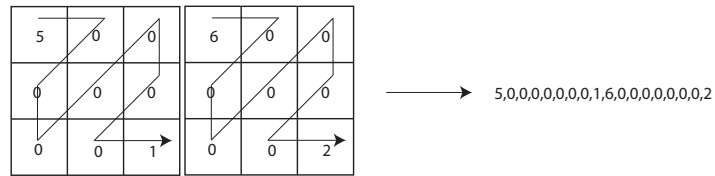


Figure 5.2: An example illustrating how JPEG coefficients are converted into a one-dimensional sequence.

of the entire sequence is estimated using the BWT entropy estimator. Recall that this quantity is the steganographic capacity of perfectly secure stegosystems.

Then, for each image, random binary messages of various sizes are generated and embedded into the image using F5 [37], Steghide [42], Model-based (MB) [43], and Outguess [36]. The message sizes start at  $\alpha = 0.01$  and, in increments of

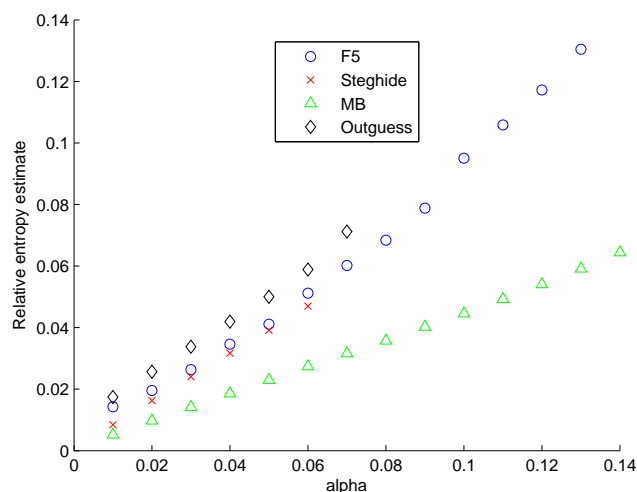


Figure 5.3: Security and capacity trade-off of F5, Steghide, MB, and Outguess JPEG stegosystems.

0.01, go up to the point where the stegosystems fail to embed the messages. Both Steghide and MB modify the JPEG coefficients directly. However, F5 decompresses and recompresses the cover image to generate the stego image. To minimize double compression problems, which lead to unfair analysis of F5, 703 images are chosen from the collection such that they use the same quality factors supported by F5.

For each stego image, the JPEG coefficients are converted into a one-dimensional sequence as described above for the cover image. The BWT relative entropy estimator is used to obtain the relative entropy estimate between the cover and stego images. Shown in Figure 5.3 is the average relative entropy estimate for each stegosystem at different  $\alpha$  values. Clearly, MB is the best of the four stegosystems. For any message size, MB is more secure than the others. This is expected since MB tries to embed according to the entropy of the source. Thus, it is favorable under relative entropy. Steghide is slightly better than F5 in terms of security. This is a consequence of Steghide preserving first-order statistics while F5 does not. It is interesting to note

that Outguess is the least secure system even though it does preserve first-order statistics just like Steghide. How they preserve the statistics, however, is different. Outguess flips bits to preserve statistics after embedding. Steghide, on the other hand, does this automatically as it swaps pixels. In a sense, Steghide does not have to correct for its embedding process, which might leave further evidence. The results also reveal that MB and F5 support larger message capacity than Steghide and Outguess. This is expected since Steghide uses  $k > 1$  symbols to hide just one bit. Similarly, Outguess only uses half the coefficients for embedding. The other half is used to preserve covertex statistics.

We compare our results against the detection accuracy of universal steganalysis methods. Universal steganalysis aims to detect the presence of a hidden message independent of the stegosystem. In this setting, a universal steganalysis detector acts as the warden that classifies an image as cover or stego using features extracted from the image. The classification process generally uses a machine learning technique such as the Support Vector Machine (SVM). As a consequence, the detection accuracy of a steganalysis detector depends on many factors including the image set, learning algorithm and parameters, and the chosen stegosystems. Despite this drawback, comparing our results against the detection accuracy of universal steganalysis detectors is still useful.

Lyu and Farid used wavelet decomposition to obtain magnitude and phase statistics of images to train linear and non-linear SVM classifiers to detect stego images [57]. They tested their detector using several JPEG stegosystems including F5, Outguess, and Steghide. Their results showed that F5 is slightly more secure than Steghide, which in turn is more secure than Outguess. Our results, on the other hand, show that Steghide is slightly more secure than F5, which is more secure than Outguess.

Fridrich used features such as the histogram and the dual histogram of the JPEG

## Chapter 5. Benchmarking Stegosystems

coefficients obtained from the stego image and the same set of features obtained from an estimate of the original image to classify cover and stego images using a Fisher Linear Discriminant classifier [56]. Several JPEG stegosystems were tested including F5, MB, and Outguess. The results showed that MB is more secure than F5, especially for small message sizes. Outguess is easily detected. This matches with our results.

Pevný and Fridrich used another steganalysis detector to detect several JPEG stegosystems including F5, MB, and Steghide [84]. Their results show that Steghide is the least secure, followed by F5, and MB. Our results show that Steghide is more secure than F5.

Kharrazi et al. benchmarked four JPEG stegosystems including F5, MB, and Outguess using three different universal steganalysis detectors trained with linear SVM classifiers [75]. The first detector used image quality metrics identified using analysis of variance techniques [52]. The second detector was wavelet decomposition as proposed by Lyu and Farid above. The third was Fridrich’s feature-based detector. For the image quality metrics and wavelet based detectors, the results showed that MB is slightly more secure than F5, which is more secure than Outguess. However, for the feature-based detector, MB is significantly more secure than F5, which is still more secure than Outguess. These findings match our results, especially for the feature-based detector.

The above comparisons illustrate the weakness of defining the security of a stegosystem from a steganalysis perspective [63]. As we have seen, one detector might detect a stegosystem poorly, but the same stegosystem is easily detected with another detector. Even with the same detector, the result can be different as the detector relies on machine learning and training for classification.

We now compare our results with one that does not rely on any universal ste-

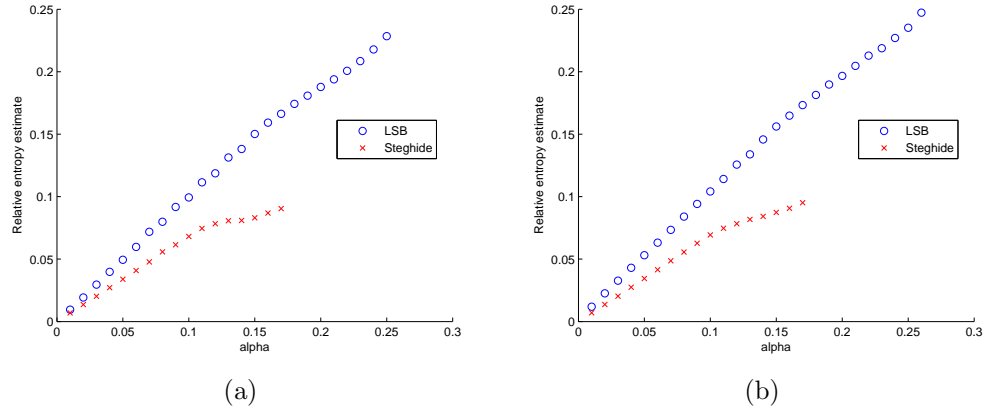


Figure 5.4: Security and capacity trade-off of Steghide and LSB pixel-domain stegosystems using (a) vertical and (b) horizontal concatenation of pixels.

ganalysis detector. Pevný and Fridrich benchmarked several JPEG stegosystems including F5, Steghide, and MB [76]. The approach was to calculate the relative entropy between a set of cover images and a set of stego images collectively. A nearest neighbor entropy estimator was used to calculate the relative entropy. However, the relative entropy estimates are poor and the Maximum Mean Discrepancy was used instead of the relative entropy. Their results showed that Steghide is the least secure of the three. For smaller message sizes, F5 is more secure than MB. For larger message sizes, MB is more secure than F5. In contrast, our results show that MB is more secure than F5 regardless of message size.

### Spatial Steganography

Unlike the approach in [76], which uses bits per non-zero AC coefficients or *bpac* to quantify the steganographic capacity of JPEG images, our approach relies on the entropy of the coverttext. It does not make any assumption about the content format of the coverttext. That is, coverttexts can be JPEG images, BMP images, audio or



video files, etc. As long as there exist mechanisms for approximating entropy and relative entropy, our approach can be used to analyze stegosystems. To demonstrate this point, we analyze two stegosystems for BMP image files. The first stegosystem is Steghide as it supports both JPEG and BMP images. The second system is LSB replacement where the least-significant bits of the image pixels are replaced with the message bits.

We convert all 812 JPEG images into BMP images. Stego BMP images are generated using the same procedure described above for JPEG images. For comparison purposes, the pixels are converted into a one-dimensional array by concatenating pixels vertically and horizontally. The BWT algorithms are used to estimate the entropy and relative entropy. Shown in Figure 5.4 is the average relative entropy estimate for each stegosystem at different  $\alpha$ . The results from both experiments are virtually identical; there is no significant difference on how the pixels are converted into a one-dimensional sequence.

Steghide is clearly more secure in contrast to the simple LSB method. This is as expected since the simple LSB method does not take into account any statistical regularity among pixels. Steghide, on the other hand, tries to preserve the first-order histogram of the pixels. Our results match the detectability of Steghide and LSB replacement using a model-based steganalysis detector [58].

One particular observation that is worth noting is the similarity between Figures 5.3 and 5.4 for Steghide. The relative entropy estimates for the corresponding  $\alpha$  values (between 0.01 and 0.06) in the two figures are similar. This is due to the fact that the JPEG coefficients and the pixel values are related directly by the DCT transform. Ignoring some small losses due to quantization, this transformation is invertible. Combining this with the fact that entropy is invariant under invertible transformations, and we see that this similarity is expected.

### 5.3.2 Algorithmic Complexity Result

Since relative entropy is not a universal metric, it does not take into account other similarities that might exist between the coartext and the stegotext. Therefore, the above results can only be interpreted with respect to relative entropy. Moreover, it would be difficult compare the security of stegosystems if they were analyzed using different security measures. To show the true security limits of these stegosystems, we benchmark them using the similarity metric.

It would be ideal if we could obtain analytical result using our complexity model on the simple stegosystem used in the information-theoretic model. However, due to the fact that algorithmic complexity is non-computable, an analytical result is not possible. As an alternative, we make use of the fact that the complexity of a random variable is approximately equal to its entropy

$$\sum_x p(x)K(x) \approx H(p). \quad (5.25)$$

With this in mind, the similarity metric in (2.24) has an entropy counterpart:

$$e(X, Y) = \frac{\max\{H(X|Y), H(Y|X)\}}{\max\{H(X), H(Y)\}}. \quad (5.26)$$

Recall that the simple stegosystem hides a message bit  $M$  into a coartext symbol  $X$  by adding a 1 or -1 ( $Y = X + M$ ). First, observe that  $H(Y|M) = H(X|M)$ . Under our assumption that  $X$  and  $M$  are independent,

$$H(Y) \geq H(Y|M) = H(X|M) = H(X). \quad (5.27)$$

Since  $H(Y) \geq H(X)$ , it follows that  $H(X|Y) \leq H(Y|X)$ . The above equation becomes

$$e(X, Y) = \frac{H(Y|X)}{H(Y)} = \frac{H(M)}{H(Y)}, \quad (5.28)$$

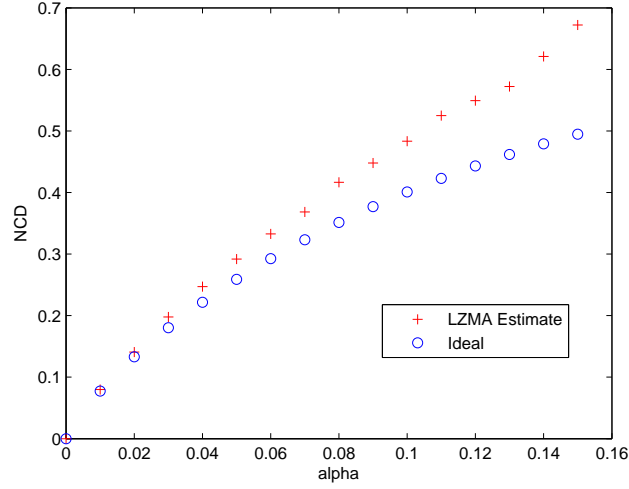


Figure 5.5: The security and capacity trade-off of the simple stegosystem using the *lzma* compressor. The estimates closely match the ideal values.

where  $H(M)$  is the entropy of the message distribution according to (5.23). This gives us a closed-form equation relating security and capacity which we can use to compare with our experimental results.

The covertexts are sequences of length  $10^5$  with symbols drawn i.i.d. over the set  $\{1, 2, 3, 4, 5\}$ . We vary  $\alpha$  from 0.01 through 0.15 in increments of 0.01. We only plot up to  $\alpha = 0.15$ , which is sufficient for the JPEG stegosystems, due to the fact that the compressor does not produce good estimates for large  $\alpha$ . The plot of the result is shown in Figure 5.5. The estimates are an average of 100 runs. Also shown is the ideal curve. The estimates closely match the ideal values for small values of  $\alpha$ . The concavity of entropy can also be seen in the plot.

We analyze the same four JPEG stegosystems (F5, Steghide, MB, and Outguess) using our complexity model. For each cover image  $x$ , the JPEG coefficients are compressed. The size of the compressed coefficients gives an approximation for the

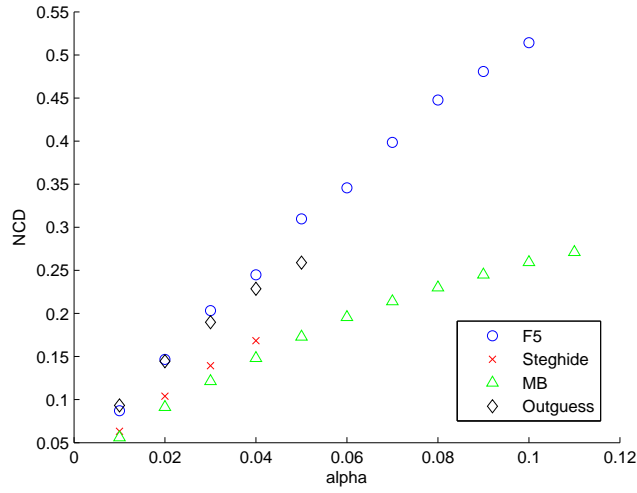


Figure 5.6: Security and capacity trade-off of F5, Steghide, MB, and Outguess JPEG stegosystems using algorithmic complexity model.

complexity of the cover image,  $K(x)$ . Then, for each image, random messages of various sizes are generated and embedded into the image using F5, Steghide, MB, and Outguess. The message sizes start at  $\alpha = 0.01$  and, in increments of 0.01, go up to the point where the stegosystems fail to embed the messages. For each stego image  $y$ , the JPEG coefficients of  $x$  and  $y$  are compressed together to obtain  $K(x, y)$ . The similarity metric can then be calculated using (3.10). Similar to our above experiment with JPEG stegosystems using the classic information-theoretic model, only 703 images are chosen from the collection such that they use the same quality factors supported by F5.

Shown in Figure 5.6 is the average  $NCD$  for each stegosystem at different  $\alpha$  values. Similar to our previous results using relative entropy, MB is the best of the four stegosystems. For any message size, MB is more secure than the others. Steghide is better than F5 in terms of security. The results also reveal that MB and F5 support larger message capacity than Steghide and Outguess. There is one difference between

the relative entropy result and this result: Outguess is more secure than F5 for larger messages. This is due to the fact that the similarity metric takes into account more than just statistical regularities, but all computable similarities between two objects. Since Outguess preserves first-order statistics of the coverttext, it has not changed the information content of the coverttext, to the first-order. F5 does not preserve first-order statistics and is therefore accounted for by the similarity metric. The details on the means and standard deviations at each  $\alpha$  for all four stegosystems are shown in Table A.2 in Appendix A.

The importance of this result is that reveals the universal steganographic security of these stegosystems. That is, it shows the true security limits of these stegosystems regardless of the chosen security metric or the existence of steganalysis detectors.

## 5.4 Summary

Defining steganographic security that is independent of any steganalysis detector solves the problems associated with the definition based on steganalysis detectors and allows for a fair analysis of stegosystems. Analysis based on steganographic security definitions, however, is not without problems. Namely, they only address one aspect of security based on the chosen quantity. This makes it difficult to compare the security of stegosystems using different definitions. To overcome this limitation, we have proposed a universal steganographic security metric called the similarity metric that takes into account all other computable metrics between the coverttext and the stegotext. This metric is asymptotically minimal among all computable metrics between the coverttext and the stegotext. As a result, a stegosystem with a large similarity metric is inherently insecure as all other computable metrics are also large. Consequently, the security of stegosystems can be analyzed using this newly proposed metric without requiring knowledge of any other metric, *a priori*.

## *Chapter 5. Benchmarking Stegosystems*

To illustrate this concept, we present a general approach for benchmarking stegosystems. It is general in the sense that it is not limited to just JPEG images or digital images and the same approach can be applied to text-based stegosystems, digital audio and video, for example. We show, for the first time, the security and capacity trade-off of several image stegosystems that is independent of any steganalysis detector. More importantly, the results show the true security limits of stegosystems regardless of the existence of steganalysis detectors. We hope that future stegosystems will use our method for benchmarking as it is easy to implement.

## Chapter 6

# Conclusion and Future Work

We have studied two information similarity metrics: relative entropy and the similarity metric. The ability to compute the similarity between two objects is critical in many applications. We have shown various methods to estimating relative entropy and the similarity metric. We have shown that compression ratio is not a good indicator for choosing compressors to estimate the similarity metric and proposed a new method for benchmarking compressors. Our experiments support our analysis and show that compressors previously reported in literature to be good estimators of the similarity metric do not perform well for long sequences even though they have minimal compression ratios. Our results show that only the *lzma* compressor does well in all cases which include text, images, and DNA sequences.

Our growing reliance on digital information has created many issues. Accessibility, theft, and privacy are examples of the problems we face. While there is no single solution to all problems, steganography has emerged as a strong contender. In order to utilize steganography effectively, it is essential to understand the security limits of stegosystems. To this end, we define the security of a stegosystem using the similarity metric and propose a new general-purpose method for benchmarking

stegosystems. The results show the true security limits of stegosystems regardless of the chosen security metric or the existence of steganalysis detectors. In other words, this makes it possible to show that a stegosystem with a large similarity metric is inherently insecure, even if it has not yet been broken. We have chosen to use image steganography to demonstrate our method due to the fact that it is popular and widely available. Our general-purpose method, however, can be applied to other forms of steganography including text, audio, and video.

The primary goal of this dissertation is to develop the similarity between objects, methods for computing similarity, and applying it to steganography. In doing so, we have gained new perspectives and obtained satisfying results. Still, there are more problems to address. Since our benchmarking method is not limited to images, we plan on investigating other types of stegosystems such as audio steganography in the future. We also plan on extending this work to systems with an active adversary such as watermarking. We offer some future research directions based on the work presented in this dissertation.

## **6.1 Enhancing Steganalysis Detector**

The model-based steganalysis detector discussed in Chapter 4.2 uses all DCT coefficients. This mixes the distributions of all coefficients and might not be accurately represented with a Laplacian distribution. It might be possible to obtain better results by grouping each coefficient type separately. That is, for each 8x8 block, we have 64 coefficient types. The distribution of each type follows a Laplacian. The difference among each distribution is primarily the scale parameter  $b$  in (4.6). Thus, we can obtain the vector  $\mathbf{r}$  as in (4.22) for each type. Since the DC coefficient is rarely affected by embedding, only the AC coefficients should be considered. If five types are used, then the total number of features is 25 rather than 5 as we originally



had. If these distributions are more accurately represented by Laplacians, then the resulting ratios should improve detection accuracy.

## 6.2 Steganalysis and Algorithmic Complexity

As we have seen, algorithmic complexity is a universal information measure. Any other deterministic processing on an object only reduces the contained information. Rather than building a steganalysis detector using some features extracted from an object, which loses information, we can use algorithmic complexity directly. If  $x$  is the coartext and  $y$  is the stegotext, we can compute  $K(x|y)$  and  $K(y|x)$  using a compression algorithm as we did in our analysis of stegosystems. Note that  $K(x|x)$  is negligible. However,  $K(x|y) > 0$  and  $K(y|x) > 0$ . The assumption here, of course, is that we have knowledge of the coartext, which is often not the case in practice.

We can, however, try to approximate the coartext. To illustrate the idea, let us consider JPEG images. We can view embedding as adding a small noise signal to a cover image. So, to recover or approximate the original cover image given the stego image, we can filter out the noise. One way to do this is to use JPEG compression on the stego image. If the stego image was compressed with quality factor  $Q$ , then we approximate the original cover image by re-compressing the stego image with quality factor  $Q - 1$ . The smaller quality factor serves as a filter to filter out the message signal. We do not want to use a much smaller quality factor as it destroys too much of the cover image. Similarly, we do not want to use a quality factor higher than  $Q$  as that does not filter the message signal. With the resulting approximation  $\hat{x}$ , we can calculate  $K(\hat{x}|y)$  and  $K(y|\hat{x})$ . The idea is that both  $K(\hat{x}|y)$  and  $K(y|\hat{x})$  are approximately 0 if  $y$  does not contain any hidden message. On the other hand, if  $y$  is a message-carrying stego image,  $K(\hat{x}|y)$  and  $K(y|\hat{x})$  would be larger than 0.

Assuming that the approximation is good, an added benefit of using this ap-

## Chapter 6. Conclusion and Future Work

proach is that it might be possible to approximate not only the size of the hidden message, but potentially the locations of the message bits, which is of great interest in *quantitative steganalysis*.

# Appendices

<b>A</b>	<b>Mean and Standard Deviation of Security Estimates</b>	<b>93</b>
A.1	Information-Theoretic Model . . . . .	94
A.2	Algorithmic Complexity Model . . . . .	95

## Appendix A

### Mean and Standard Deviation of Security Estimates

## A.1 Information-Theoretic Model

Table A.1: Mean and standard deviation of relative entropy estimate at different  $\alpha$  for F5, Steghide, MB, and Outguess JPEG stegosystems.

$\alpha$	F5		Steghide		MB		Outguess	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
0.01	0.0143	0.0134	0.0084	0.0031	0.0052	0.0022	0.0174	0.0361
0.02	0.0196	0.0139	0.0163	0.0059	0.0097	0.0040	0.0256	0.0357
0.03	0.0263	0.0147	0.0241	0.0086	0.0142	0.0058	0.0338	0.0354
0.04	0.0346	0.0165	0.0317	0.0113	0.0186	0.0075	0.0419	0.0358
0.05	0.0411	0.0173	0.0392	0.0139	0.0230	0.0091	0.0500	0.0356
0.06	0.0512	0.0196	0.0470	0.0152	0.0274	0.0106	0.0589	0.0365
0.07	0.0602	0.0213			0.0316	0.0121	0.0712	0.0377
0.08	0.0684	0.0226			0.0357	0.0136		
0.09	0.0788	0.0245			0.0402	0.0149		
0.10	0.0950	0.0283			0.0446	0.0162		
0.11	0.1058	0.0296			0.0493	0.0174		
0.12	0.1172	0.0304			0.0540	0.0186		
0.13	0.1304	0.0298			0.0591	0.0196		
0.14					0.0644	0.0204		

## A.2 Algorithmic Complexity Model

Table A.2: Mean and standard deviation of  $NCD$  at different  $\alpha$  for F5, Steghide, MB, and Outguess JPEG stegosystems.

$\alpha$	F5		Steghide		MB		Outguess	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
0.01	0.0871	0.0469	0.0633	0.0151	0.0560	0.0095	0.0933	0.0455
0.02	0.1467	0.0831	0.1039	0.0236	0.0912	0.0159	0.1446	0.0453
0.03	0.2032	0.0803	0.1393	0.0308	0.1214	0.0207	0.1898	0.0471
0.04	0.2448	0.0798	0.1683	0.0325	0.1481	0.0238	0.2286	0.0470
0.05	0.3098	0.0855			0.1730	0.0268	0.2590	0.0439
0.06	0.3458	0.0800			0.1957	0.0293		
0.07	0.3985	0.0703			0.2140	0.0261		
0.08	0.4477	0.0511			0.2300	0.0210		
0.09	0.4808	0.0449			0.2449	0.0161		
0.10	0.5143	0.0436			0.2595	0.0137		
0.11					0.2711	0.0111		

## References

- [1] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [2] J. Ziv and N. Merhav, “A measure of relative entropy between individual sequences with application to universal classification,” *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1270–1279, 1993.
- [3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. Wiley-Interscience, 2000.
- [4] C. Jia, D. Chen, and K. Lin, “The application of the relative entropy density divergence in intrusion detection models,” vol. 3, pp. 951–954, 2008.
- [5] C. Cachin, “An information-theoretic model for steganography,” in *2nd International Workshop on Information Hiding*, ser. Lecture Notes in Computer Science. Springer, 1998, pp. 306–318.
- [6] ———, “An information-theoretic model for steganography,” *Information and Computation*, vol. 192, no. 1, pp. 41–56, 2004.
- [7] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitányi, and W. H. Zurek, “Information distance,” *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1407–1423, 1998.
- [8] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, “The similarity metric,” *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3250–3264, 2004.
- [9] A. Kolmogorov, “Three approaches to the quantitative definition of information,” *International Journal of Computer Mathematics*, vol. 2, no. 1, pp. 157–168, 1968.
- [10] R. J. Solomonoff, “A formal theory of inductive inference: Part 1 and 2,” *Inform. Contr.*, vol. 7, no. 1-22, pp. 224–254, 1964.

## References

- [11] G. J. Chaitin., “On the length of programs for computing finite binary sequences,” *Journal of the ACM*, vol. 13, no. 4, pp. 547–569, 1966.
- [12] A. M. Turing, “On computable numbers with an application to the entscheidungsproblem,” in *London Math. Soc.*, vol. 42, 1936, pp. 230–265, correction, *Ibid.*, 43:544–546, 1937.
- [13] A. Church, “A note on the entscheidungsproblem,” *Journal of Symbolic Logic*, vol. 1, pp. 40–41, 1936.
- [14] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer, 2008.
- [15] L. G. Kraft, “A device for quantizing, grouping, and coding amplitude modulated pulses,” Master’s thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1949.
- [16] C. E. Shannon and W. Weaver, “A mathematical theory of communications,” *Bell System Technical Journal*, vol. 27, no. 2, pp. 632–656, 1948.
- [17] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.
- [18] J. Ziv and A. Lempel, “Compression of individual sequences via variable length coding,” *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978.
- [19] A. D. Wyner and J. Ziv, “On entropy and data compression,” *IEEE Trans. Inf. Theory*, 1991.
- [20] —, “Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression,” *IEEE Trans. Inf. Theory*, vol. 35, no. 6, pp. 1250–1258, 1989.
- [21] I. Kontoyiannis, P. H. Algoet, Y. M. Suhov, and A. J. Wyner, “Nonparametric entropy estimation for stationary processes and random fields, with applications to english text,” *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1319–1327, 1998.
- [22] M. Burrows and D. J. Wheeler, “A block-sorting lossless data compression algorithm,” Digital Systems Research Center, Tech. Rep. 124, 1994.
- [23] H. Cai, S. R. Kulkarni, and S. Verdú, “Universal entropy estimation via block sorting,” *IEEE Trans. Inf. Theory*, vol. 50, no. 7, pp. 1551–1561, 2004.
- [24] H. Cai, S. Kulkarni, and S. Verdú, “Universal divergence estimation for finite-alphabet sources,” *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3456–3475, 2006.



## References

- [25] R. Cilibrasi and P. Vitányi, “Clustering by compression,” *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1523–1545, 2005.
- [26] P. Ferragina, R. Giancarlo, V. Greco, G. Manzini, and G. Valiente, “Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment,” *BMC Bioinformatics*, vol. 8, no. 1, p. 252, 2007.
- [27] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 337–343, 1977.
- [28] X. Chen, S. Kwong, and M. Li, “A compression algorithm for dna sequences and its applications in genome comparison,” in *10th Workshop on Genome Informatics*, 2000, pp. 51–61.
- [29] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, “An information-based sequence distance and its application to whole mitochondrial genome phylogeny,” *Bioinformatics*, vol. 17, no. 2, pp. 149–154, 2001.
- [30] M. Cebrián, M. Alfonseca, and A. Ortega, “Common pitfalls using the normalized compression distance: what to watch out for in a compressor,” *Communications in Information and Systems*, vol. 5, no. 4, pp. 367–384, 2005.
- [31] D. Shkarin, “PPM: One step to practicality,” in *Data Compression Conference*. IEEE Computer Society, 2002, pp. 202–211.
- [32] X. Chen, M. Li, B. Ma, and J. Tromp, “Dnacompress: fast and effective dna sequence compression,” *Bioinformatics*, vol. 18, no. 12, pp. 1696–1698, 2002.
- [33] B. Ma, J. Tromp, and M. Li, “Patternhunter: faster and more sensitive homology search,” *Bioinformatics*, vol. 18, no. 3, pp. 440–445, 2002.
- [34] G. J. Simmons, “The prisoners problem and the subliminal channel,” in *CRYPTO*, vol. 83, 1984, pp. 51–67.
- [35] “Former state department official and wife arrested for serving as illegal agents of cuba for nearly 30 years,” <http://washingtondc.fbi.gov/dojpressrel/pressrel09/wfo060509a.htm>, June 2009.
- [36] N. Provos, “Defending against statistical steganalysis,” in *10th USENIX Security Symposium*.

## References

- [37] A. Westfeld, “F5 - a steganographic algorithm,” in *4th International Workshop on Information Hiding*, ser. Lecture Notes In Computer Science, vol. 2137. Springer, 2001, pp. 289–302.
- [38] R. Crandall, “Some notes on steganography,” *posted on Steganography Mailing List*, <http://os.inf.tu-dresden.de/westfeld/crandall.pdf>, 1998.
- [39] M. van Dijk and F. Willems, “Embedding information in grayscale images,” in *22nd Symposium on Information and Communication Theory*, 2001, pp. 147–154.
- [40] F. Galand and G. Kabatiansky, “Information hiding by coverings,” in *Information Theory Workshop*, 2003, pp. 151–154.
- [41] J. Fridrich and D. Soukal, “Matrix embedding for large payloads,” *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 3, pp. 390–395, 2006.
- [42] S. Hetzl and P. Mutzel, “A graph-theoretic approach to steganography,” in *Communications and Multimedia Security*, ser. Lecture Notes in Computer Science, vol. 3677. Springer, 2005, pp. 119–128.
- [43] P. Sallee, “Model-based steganography,” in *Digital Watermarking, Second International Workshop*, ser. Lecture Notes in Computer Science, vol. 2939. Springer, 2004, pp. 154–167.
- [44] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.
- [45] A. Westfeld and A. Pfitzmann, “Attacks on steganographic systems,” in *3rd International Workshop on Information Hiding*, ser. Lecture Notes In Computer Science, vol. 1768. Springer-Verlag, 1999, pp. 61–76.
- [46] J. Fridrich, M. Goljan, and R. Du, “Reliable detection of lsb steganography in color and grayscale images,” in *Workshop on Multimedia Security and Watermarking*. ACM, 2001.
- [47] —, “Detecting LSB steganography in color and grayscale images,” *Magazine of IEEE Multimedia: Special Issue on Security*, vol. Oct-Dec, pp. 22–28, 2001.
- [48] S. Dumitrescu, X. Wu, and Z. Wang, “Detection of lsb steganography via sample pair analysis,” *IEEE Trans. Signal Process.*, vol. 51, no. 7, pp. 1995–2007, 2003.
- [49] J. Fridrich, M. Goljan, and D. Hoge, “Attacking the outguess,” in *Workshop on Multimedia and Security*. ACM, 2002.

## References

- [50] —, “Steganalysis of JPEG images: Breaking the F5 algorithm,” in *5th International Workshop on Information Hiding*, ser. Lecture Notes In Computer Science, vol. 2578. Springer-Verlag, 2003, pp. 310–323.
- [51] J. Fridrich, M. Goljan, D. Hoge, and D. Soukal, “Quantitative steganalysis: estimating secret message length,” *ACM Multimedia Systems. Special issue on Multimedia Security*, vol. 9, no. 3, pp. 288–302, 2003.
- [52] I. Avcibas, N. Memon, and B. Sankur, “Steganalysis using image quality metrics,” *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 221–229, 2003.
- [53] I. Avcibas, M. Kharrazi, NasirMemon, and B. Sankur, “Image steganalysis with binary similarity measures,” *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 17, pp. 2749–2757, 2005.
- [54] J. Fridrich, M. Goljan, and R. Du, “Steganalysis based on JPEG compatibility,” in *Multimedia Systems and Applications IV*. SPIE, 2001, pp. 275–280.
- [55] J. J. Harmsen and W. A. Pearlman, “Steganalysis of additive noise modelable information hiding,” in *Electronic Imaging*, vol. 5022. SPIE, 2003.
- [56] J. Fridrich, “Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes,” in *6th International Workshop on Information Hiding*, ser. Lecture Notes In Computer Science, vol. 3200. Springer, 2004, pp. 67–81.
- [57] S. Lyu and H. Farid, “Steganalysis using higher-order image statistics,” *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 1, pp. 111–119, 2006.
- [58] T.-T. Quach, F. Pérez-González, and G. L. Heileman, “Model-based steganalysis using invariant features,” in *Media Forensics and Security XI*, vol. 7254. SPIE Electronic Imaging, 2009.
- [59] F. Pérez-González, T.-T. Quach, G. L. Heileman, and C. T. Abdallah, “Benford’s law and multimedia signals,” in *Workshop on Applications of Benford’s Law*, Santa Fe, NM, USA, December 2007.
- [60] R. Reininger and J. Gibson, “Distributions of the two-dimensional dct coefficients for images,” *IEEE Trans. Commun.*, vol. 31, no. 6, pp. 835–839, 1983.
- [61] L. M. Marvel, C. G. Boncellet Jr., and C. T. Retter, “Spread spectrum image steganography,” *IEEE Trans. Image Process.*, vol. 8, pp. 1075–1083, 1999.

## References

- [62] T. Joachims, *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.
- [63] R. Chandramouli and N. Memon, “Steganography capacity: a steganalysis perspective,” in *Security and Watermarking of Multimedia Contents V*, vol. 5020. SPIE, 2003, pp. 173–177.
- [64] A. D. Ker, “Improved detection of LSB steganography in grayscale images,” in *6th International Workshop on Information Hiding*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 87–115.
- [65] —, “Quantitative evaluation of Pairs and RS steganalysis,” in *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306. SPIE, 2004, pp. 83–96.
- [66] —, “Steganalysis of LSB matching in grayscale images.” in *IEEE Signal Processing Letters*, 2005, pp. 441–444.
- [67] M. Goljan, J. Fridrich, and T. Holtyak, “New blind steganalysis and its implications,” in *Security, Steganography and Watermarking of Multimedia Contents VIII*, vol. 6072. SPIE, 2006, pp. 101–113.
- [68] A. D. Ker, “Steganalysis of embedding in two least significant bits.” *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 1, pp. 46–54, 2007.
- [69] M. J. Ettinger, “Steganalysis and game equilibria,” in *2nd International Workshop on Information Hiding*, ser. Lecture Notes in Computer Science, vol. 1525. Springer, 1998, pp. 319–328.
- [70] J. Zöllner, H. Federrath, H. Klimant, A. Pfitzmann, R. Piotraschke, A. Westfeld, G. Wicke, and G. Wolf, “Modeling the security of steganographic systems,” in *2nd International Workshop on Information Hiding*, ser. Lecture Notes in Computer Science, vol. 1525. Springer, 1998, pp. 344–354.
- [71] T. Mittelholzer, “An information-theoretic approach to steganography and watermarking,” in *3rd International Workshop on Information Hiding*, ser. Lecture Notes in Computer Science, vol. 1768. Springer, 1999, pp. 1–16.
- [72] N. J. Hopper, J. Langford, and L. von Ahn, “Provably secure steganography,” in *Advances in Cryptology: CRYPTO*, August 2002, pp. 77–92.
- [73] S. Katzenbeisser and F. A. P. Petitcolas, “Defining security in steganographic systems,” in *Security and Watermarking of Multimedia Contents IV*, vol. 4675. SPIE, 2002, pp. 260–268.

## References

- [74] T.-T. Quach, F. Pérez-González, and G. L. Heileman, “Universal steganographic security,” *Submitted to IEEE Trans. Inf. Forensics Security*.
- [75] M. Kharrazi, H. T. Sencar, and N. D. Memon, “Benchmarking steganographic and steganalysis techniques,” in *Security, Steganography, and Watermarking of Multimedia Contents VII*, vol. 5681. SPIE, 2005, pp. 252–263.
- [76] T. Pevný and J. Fridrich, “Benchmarking for steganography,” in *10th International Workshop on Information Hiding*, ser. Lecture Notes In Computer Science. Springer-Verlag, 2008, pp. 251–267.
- [77] P. Moulin and J. OSullivan, “Information-theoretic analysis of information hiding,” *IEEE Trans. Inf. Theory*, vol. 49, no. 3, p. 563, 2003.
- [78] P. Moulin and Y. Wang, “New results on steganographic capacity,” in *Information Science and Systems*, Princeton, NJ, USA, March 2004.
- [79] Y. Wang and P. Moulin, “Perfectly secure steganography: Capacity, error exponents, and code constructions,” *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2706–2722, 2008.
- [80] P. Comesana and F. Pérez-González, “On the capacity of stegosystems,” in *9th Workshop on Multimedia & security*. ACM, 2007, pp. 15–24.
- [81] J. J. Harmsen and W. A. Pearlman, “Capacity of steganographic channels,” in *7th Workshop on Multimedia and security*. ACM, 2005, pp. 11–24.
- [82] S. Verdú and T. S. Han, “A general formula for channel capacity,” *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 1147–1157, 1994.
- [83] R. Chandramouli and N. Memon, “Analysis of lsb based image steganography techniques,” in *International Conference on Image Processing*, vol. 3. IEEE, 2001, pp. 1019–1022.
- [84] T. Pevný and J. Fridrich, “Merging markov and dct features for multi-class jpeg steganalysis,” in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505. SPIE Electronic Imaging, 2007.