TÉCNICO LISBOA

A Novel Approach for Semantic Pure Steganography

João Teixeira Figueira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Alexandre P. L. Francisco

Examination Committee

December 2020

Acknowledgments

I would like to thank my parents for their support throughout my entire academic life, they have my great respect for their effort in having three children pursue a higher education degree, and my appreciation for all they do for me.

I want to thank my friends and colleagues for being my main support group and being there for me whenever I was struggling with difficulties in my degree or in life, and without whom I would not have had even been able to get to the point of starting this thesis. Working together and towards the same goals is what pushed me to keep going forward.

I thank my roommate for the many thesis writing sessions we had together, and for pressuring me to focus on working whenever I was starting to slack off.

I want to thank my family, my friends, and my boyfriend for being a source of comfort and understanding, and for always being there for me, throughout all these years.

I would also like to acknowledge my dissertation supervisor Prof. Alexandre Francisco for initially listening to my ideas for research topics and for helping me guide one of those into the thesis that is here today. I thank him for his continued support, insight, patience, and knowledge that has made this possible.

To all those who have been by my side in this long journey. Thank you.

Abstract

Steganography is the practice of concealing a message within some other carrier or cover message. It is used to allow the sending of hidden information through communication channels where third parties would only be aware of the explicit information in the carrier message. In this article propose a novel approach for text steganography that can be classified as pure steganography. The proposed algorithm uses the redundancy in language semantics as the space for the hidden message. It improves on existing algorithms by not requiring the message receiver to be aware of the specific redundancies of any cover message. We contextualize our system by thoroughly reviewing semantic steganography and the concepts surrounding it, and by surveying published systems in this area. Our results show that a semantic pure steganographic system is possible and can realistically be used, despite being limited by very low embedding rates.

Keywords

Steganography; Linguistic; Semantics; Monte Carlo

Resumo

Esteganografia é a prática de esconder uma mensagem dentro de uma outra mensagem de transporte. É usado para permitir que informação seja enviada através de canais de comunicação onde terceiros estariam apenas cientes da informação explícita na mensagem de transporte. Neste artigo propomos um sistema novo para esteganografia que pode ser classificado como esteganografia pura. O algoritmo proposto usa a redundância na semântica da língua natural usada na mensagem de transporte como o espaço para a mensagem oculta. Este algoritmo apresenta melhorias perante algoritmos existentes por não necessitar que o recetor da mensagem identifique as redundâncias específicas existentes na mensagem de transporte. Contextualizamos o nosso sistema com uma revisão completa de esteganografia semântica e dos conceitos que a rodeiam, e através de uma pesquisa a sistemas publicados nesta área. Os nossos resultados mostram que um sistema de esteganografia pura semântica é possível e pode ser usado em situações realistas, apesar de ser limitado por taxas de informação escondida muito baixas.

Palavras Chave

Esteganografia; Linguística; Semântica; Monte Carlo

Contents

1 Introduction **1**

1.1 Encryption and Steganography . . . 3

1.2 Motivation . . . 3

1.3 Objectives . . . 4

1.4 Document Structure . . . 4

2 Background and Definitions **5**

2.1 Encryption . . . 7

2.2 Steganography . . . 8

2.2.1 Motivation for Steganography . . . 8

2.2.2 The Steganographic Process . . . 9

2.3 Embedding Rate . . . 10

2.4 Purity of Steganographic Systems . . . 10

2.4.1 Pure Steganography . . . 10

2.4.2 Secret Key Steganography . . . 10

2.4.3 Public Key Steganography . . . 11

2.4.4 Natural Randomness of Covertexts . . . 11

2.5 Embedding Functions . . . 11

2.5.1 Steganography by Cover Modification . . . 12

2.5.2 Steganography by Cover Synthesis . . . 12

2.5.3 Steganography by Cover Lookup . . . 12

2.6 Cover Messages . . . 12

2.6.1 Image Steganography . . . 13

2.6.2 Audio Steganography . . . 13

2.6.3 Steganography on other Multimedia Formats . . . 14

2.6.4 Text Steganography . . . 14

2.7 Text Steganography Systems . . . 15

2.7.1 Format-Based Text Steganography . . . 15

2.7.2 Statistical Text Steganography 15
2.7.3 Linguistic Text Steganography 16
2.8 Linguistic Steganography Systems 16
2.8.1 Syntactic Text Steganography 16
2.8.2 Semantic Text Steganography 16

3 Related Work **17**
3.1 Synonym Table Steganography Systems 19
3.1.1 Embedding Method 19
3.1.2 Extraction Method 20
3.1.3 Synonym Tables 21
3.1.4 Synonym Table Construction 23
3.1.5 Synonym Plurality 24
3.2 Other Approaches for Semantic Steganography 25
3.2.1 Synonym Distribution 25
3.2.2 Mimic Functions 26
3.2.3 Markov Chain Based Text Steganography 26

4 Approach **29**
4.1 Replaceable Substring Identification 31
4.2 Substring Replacement Set Construction 32
4.3 Section Replacement Labelling 33
4.4 Hidden Message Extraction 33
4.5 Embedding Failure Probability 34
4.5.1 Estimating Embedding Failure Probability 35
4.5.2 Estimating The Probability Distribution for Replacements of a Section 38

5 Implementation **41**
5.1 Natural Language Selection 43
5.2 Synonym Table Construction 43
5.2.1 Core Dataset 43
5.2.2 Expansion with Inflections 44
5.2.3 Synset Intersection 44
5.2.4 Public Availability 45
5.3 Steganography System Implementation 46
5.3.1 Project Structure 46
5.3.2 Text Parsing 46
5.3.3 Synonym Table Usage 47

5.3.4 Section Hashing . . . 47
5.3.5 Public Availability . . . 48
5.4 Probability Estimation Functions . . . 48

6 Evaluation **49**
6.1 Replacement Table . . . 51
6.1.1 Synonym Quality . . . 51
6.1.2 Synonym Probabilities . . . 52
6.1.3 Independence of Word Replaceability . . . 53
6.2 Steganography System . . . 54
6.2.1 Embedding Effectiveness . . . 54
6.2.1.A Distribution of Section Replacement Cardinality . . . 54
6.2.1.B Message Embedding Probability . . . 55
6.2.2 Embedding Capacity . . . 57
6.2.3 Statistical Undetectability . . . 57
6.2.4 Robustness . . . 58
6.3 Hidden Message Alphabet . . . 59
6.3.1 Alphabet Encoding . . . 59
6.3.2 Alphabet Multiplication . . . 60

7 Final Remarks **63**
7.1 Significant Contributions . . . 65
7.2 Comments on the System . . . 66
7.3 Future Improvements . . . 66
7.3.1 Context Aware Replacement Table . . . 66
7.3.2 Deep Learning Applications for Text Rewriting . . . 67
7.3.3 Replacement Recycling . . . 67
7.4 Conclusion . . . 68

A Usage Example **73**

List of Figures

2.1 Diagram showing a generic setting for symmetric cipher encryption. The sender encrypts the message m with the key k, producing the *ciphertext* c. This *ciphertext* is sent to the receiver who decrypts it with the same key k. A third party might see the *ciphertext* but is unable to extract the *plaintext* from it without k. 8

2.2 Diagram showing a generic setting for steganography. The sender embeds the *plaintext* message m into the *covertext* c, producing the *stegotext* s. This *stegotext* is sent to the receiver who uses the extraction function to recover the original *plaintext*. A third party might see the *stegotext* but it should be innocuous enough that no suspicion would be raised to the fact that it is carrying a message. 9

2.3 Demonstration of an image steganography system, using Stanley's implementation available in [1]. This is an LSB (lowest significant bits) encoding system. The three least significant bits in the pixels of the cover image (left) are replaced with the most significant bits of the payload image's pixels (middle). The resulting image (right) appears largely unchanged except for a slight quality loss and slight noise. These alterations are more obvious in the smoother areas of the image, such as the sky. 13

2.4 Diagram showing hierarchy of the major families of steganographic systems. 14

3.1 Diagram exemplifying the embedding process of the *plaintext* "11" into the *covertext* "The room was small and the air was cool." using a semantic steganographic system with the synonym table shown in Table 3.1. 20

3.2 Diagram showcasing the extraction process from the *covertext* generated in figure 3.1. This process requires the same synonym table shown in Table 3.1. 21

3.3 Examples for sets of synonyms and the bits they can encode, as described by Winstein in [2]. 24

3.4 Diagram showcasing the embedding of the hidden message "10010" into the *covertext* "The room was small and the air was cool.", using the *multi-base number* approach described by Winstein in [2]. 25

3.5 Diagram of a steganography system constructed using a Markov model, as described in [3]. To exemplify the function of the model, the hidden message “0100” would synthesize the *stegotext* “is there a great”, while “1001” would synthesize the *stegotext* “is not good enough”. 27

3.6 Diagram of grouped state transitions as described by Moraldo [4]. By grouping two consecutive transitions, the transition “is there” is made more probable than “is obfuscated”, as expected in natural text. In Dai’s approach, these transitions would be equiprobable. . 27

3.7 Diagram exemplifying the usage of “chained” Huffman trees to label transitions in a Markov model, as described by Zhongliang [5]. 28

4.1 Diagram showing the process of computing the replacement sets for fixed-size sections of a message. In this example, the size of each section is 5 words. 32

4.2 Diagram showing the embedding process of the proposed approach. This example uses the replacement sets computed for the cover message in figure 4.1. Here, the hidden message “cb” uses the alphabet $\Sigma = \{a, b, c\}$. 34

4.3 Diagram showing the extraction process of the proposed approach. This example uses the *stegotext* computed in Figure 4.2. 35

5.1 Examples of some synsets in Wordnet [6, 7]. 43

5.2 Examples of some synsets created using Inflect [8] and MLConjug [8], from the synsets in Figure 5.1. 44

5.3 Examples of some words and the synsets in which they appear. 45

5.4 Examples of some words and the respective sets of ”safe” replacements. 45

5.5 Simplified UML class diagram of the described implementation of the system. 47

6.1 Probability of a word being replaceable, by decade of document publication, from a 3 million word subset of the COHA corpus [9]. 53

6.2 $RP(w, r)$, Probability distribution for number of replacements up to 100, on sections of $w = 100$ words. The probability mass function described in Corollary 4.5.4 is compared to the results obtained from sampling the dataset described in Section 6.1.2. The replacement table described in Section 5.2 is used for the sampling, and the corresponding first order statistics in Table 6.2 are used as parameters for the estimation formula. 55

6.3 Probability that a hidden message of size n is embeddable into some cover message. Using sections of 200 words, and assuming a hidden message alphabet of size 29 (English alphabet with 2 punctuation marks and the space character). The ”measured probability” was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The ”expected lower bound” was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table. 56

6.4 Probability that a hidden message of 20 characters is embeddable into some cover message. Using sections of w words, and assuming a hidden message alphabet of size 29. The ”measured probability” was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The ”expected lower bound” was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table. Lemma 4.5.2 is plotted again, using the real $RP(w, r)$ distribution as measured and shown in Figure 6.2. 56

6.5 Embedding probability for one section, depending on embedding rate, and assuming a hidden message alphabet of size 29. The ”measured probability” was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The ”expected lower bound” was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table. Figure 6.2. 58

6.6 Number of replacements needed for a section to ensure 90% and 95% probability of embedding a hidden message character, depending on alphabet size. 59

6.7 Comparison between usage of a 32 character alphabet, and that same alphabet encoded into 5 bits of a binary alphabet, measured in embedding probability. The horizontal axis shows size of each section for the 32 character alphabet, or the size of 5 sections for the binary alphabet. Results were computed for a 10 character message, or 50 bits, after conversion, by sampling candidate covertexts from the dataset described in Section 6.1.2. 60

6.8 Comparison between usage of a 32 character alphabet Σ^1, and longer alphabets Σ^n created by the grouping of n characters of Σ^1. Results were computed for a 12 character message in Σ^{1*}, converted to a 6, 4, 3, and 2 character message using Σ^2, Σ^3, Σ^4, and Σ^6, respectively. Simulations were ran by sampling candidate covertexts from the dataset described in Section 6.1.2. The horizontal axis measures covertext words per character of the hidden message after being converted to Σ^1. 61

List of Tables

3.1 Example of a short synonym table, used by Bender in [10]. 19

3.2 Example of a short synonym table using words that have different spellings in American English and European English, used by Shirali-Shahreza [11]. 22

3.3 Example of a short synonym table using the acronyms and abbreviations commonly found in SMS messaging, partly taken from [12] and extended with some more modern expressions. 22

6.1 Example of a paragraph being rewritten according to the synonym table constructed as described in Section 5.2. This paragraph is a sample of the book ”To Kill a Mockingbird”, by Harper Lee, 1960. Replaced words are highlighted. 51

6.2 Probabilities of a word having n replacements, calculated over a 3 million word subset of the COHA corpus [9] and a million word subset of the GloWbE corpus [13]. 52

6.3 Probabilities of a word being replaceable, for the various document types within a 3 million word subset of the COHA corpus [9] and a million word subset of the GloWbE corpus [13]. 53

1 Introduction

Contents

1.1 Encryption and Steganography 3

1.2 Motivation 3

1.3 Objectives 4

1.4 Document Structure 4

1.1 Encryption and Steganography

When two parties want to communicate securely via the sending of messages, these messages should be encrypted. As described by Ferguson [14], encrypting a message (that in its initial state is called a *plaintext*) is the process of feeding it through an encryption function that outputs another message called a *ciphertext*, this message usually appears to be random bits or characters. This *ciphertext* can be sent to a second party which will use a decryption function to recover the original *plaintext*. If a third party were to intercept the message that is being sent, it would have no way (if the encryption system is secure enough) to extract the *plaintext* message from the *ciphertext* message, as only the desired recipients of the message would have the correct decryption function.

When a third party intercepts an encrypted message, it becomes aware that confidential communication is happening between the two other parties. This, in itself can be seen as a minor security problem. Alternatively (as exemplified by Katzenbeisser [15], with the prisoners and guard example), if a third party is needed to deliver messages between the two other parties, it might refuse to deliver messages that it cannot read (they might contain conspiracy against the third party). In specific situations like these, alternative encryption methods can be used in which the *ciphertext* appears innocuous, *i.e.* the secret message is hidden within some other “benign-looking” message [16]. These methods are called steganography.

Steganography methods take an “innocent” message, called *covertext* and embed it with the *plaintext*, outputting a *stegotext*. This *stegotext* is the equivalent to the *ciphertext* in that the *plaintext* can be extracted from it, but it maintains the innocuous property of the *covertext*.

Semantic steganography is the branch of text steganography that uses redundancies in the vocabulary of natural languages as the space for the *plaintext* message [17]. Most semantic steganography methods function in the following way: The message sender and receiver share a synonym table, this table (of usually two columns) pairs words with their interchangeable synonyms. The message sender will use this table to swap some of the words in the *covertext* with their synonyms. These swaps are done in a deliberate way that allows for the embedding of the *plaintext*. The message receiver is then able to, by consulting the same synonym table, know which words were swapped out with their synonyms, and then use this information to reconstruct the *plaintext*.

1.2 Motivation

One of the problems of most current semantic steganography methods is the requirement of the shared synonym table. In a realistic scenario where steganography would be required, the communication channel might not be sufficiently reliable that a big synonym table could be sent. Additionally, it is not possible to later improve the synonym table by adding new words or synonyms without having to update the other party's table.

Further, no existing methods for semantic steganography can be classified as pure steganography (defined as a steganography method that does not require the prior exchange of some secret information [15]) because of the need for the prior sharing of the synonym table.

1.3 Objectives

The purpose of this article is to first serve as a study on the various types steganography, survey existing systems and algorithms, and to propose a novel approach for semantic steganography. The proposed system should not require that a message receiver have access to the synonym table used by the sender to create the *stegotext*, and, as such, could be classified as pure semantic steganography. This system is to be described in detail and evaluated with regards to existing systems.

1.4 Document Structure

This document is structured as follows: In Section 2 we go over establishing important concepts that are relevant for the understanding of the article; Section 3 describes existing systems that are the state of the art for semantic steganography; In Section 4 we propose a novel system for semantic steganography and describe its function and properties in detail; The implementation of the described system is exposed in Section 5; In Section 6, the metrics that can be used to evaluate the proposed system are listed and used for its validation; In Section 7, final remarks regarding the development and applicability of the system are exposed, along with propositions for possible future developments and improvements that could be applied.

2

Background and Definitions

Contents

2.1 Encryption 7

2.2 Steganography 8

2.3 Embedding Rate 10

2.4 Purity of Steganographic Systems 10

2.5 Embedding Functions 11

2.6 Cover Messages 12

2.7 Text Steganography Systems 15

2.8 Linguistic Steganography Systems 16

Steganography, the focus of this article, is a very ancient area of research, despite this, it is not very actively studied in modern times. Scientific literature on the matter often offers very different and some times even contradictory information. In this section we brought together and conciliated the information from various articles while trying to provide the most consensual definitions and formalisations.

Definition 2.0.1. An alphabet Σ is defined as a finite set of symbols, called characters or letters [18] (the alphabets studied in the context of cryptography and steganography are usually natural language alphabets (such as the letters and punctuation used in the English alphabet, or specific encodings of these, such as utf-8) or the binary "alphabet" $\Sigma = \{0, 1\}$ as these are the most commonly used in communication).

Definition 2.0.2. A message m is a finite sequence (or string) of characters from an alphabet Σ.

2.1 Encryption

According to Ferguson in [14], "Encryption is the original goal of Cryptography" and is the most basic way of securing communication. In a context where two parties are sending messages through a communication channel that might be eavesdropped on (in the internet, messages usually pass through multiple intermediaries on the way to their destination), encryption is a process that makes the messages unreadable while travelling through the unsafe channel.

As described in [14]: A sender desires to send a message to a receiver. In the context of cryptography, the original message possessed by the sender is called the *plaintext*. The channel is of such a nature that if the sender sends the message to the receiver, it might be read by a third party. This is undesirable to the sender. Encrypting a message is the process of using an encryption function that takes the *plaintext* and outputs a *ciphertext*. This *ciphertext* can be sent to the receiver. If intercepted, it is not possible (if the encryption function is safe enough) for the third party to extract the original *plaintext* from the *ciphertext*. After receiving the *ciphertext*, the receiver will use the corresponding decryption function to extract the *plaintext*.

For further security, these encryption and decryption functions usually take an additional parameter, a key. Third parties might have access to the decryption function, but the key is usually something decided privately by the message sender and receiver so that there is no way that the third party might know what it is. When the decryption and encryption functions use the same key, the system is called a symmetric cipher (there are other types of encryption systems but they are not yet relevant for the focus of this article). A diagram of such a system can be seen in Figure 2.1.

Definition 2.1.1. A symmetric cipher encryption system can be defined as a quintuple $< M, C, K, E, D >$, where M is the set of possible *plaintexts* (the set of all messages over the *plaintext* alphabet Σ, this set

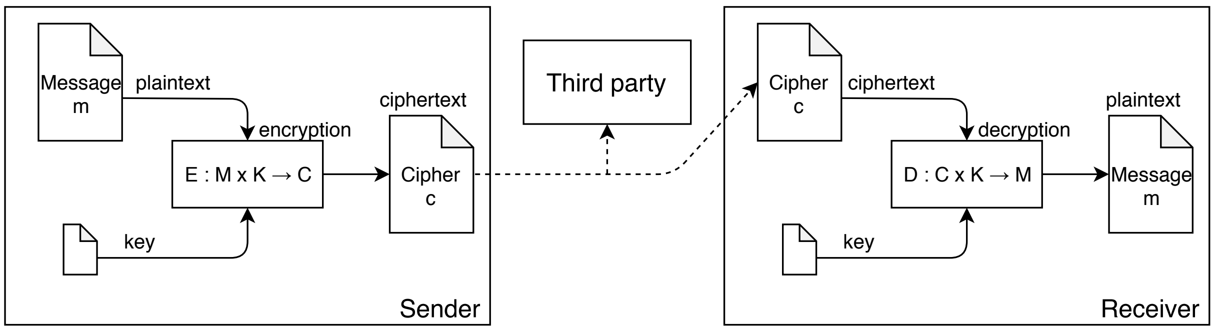

Figure 2.1: Diagram showing a generic setting for symmetric cipher encryption. The sender encrypts the message m with the key k, producing the *ciphertext* c. This *ciphertext* is sent to the receiver who decrypts it with the same key k. A third party might see the *ciphertext* but is unable to extract the *plaintext* from it without k.

can represented as Σ^*), C is the set of possible *ciphertexts* (the set of all messages over the *ciphertext* alphabet which is often the same as the *plaintext* alphabet), K is the set of possible keys (usually also the set of strings over some alphabet), $E : M \times K \to C$ is the encryption function, $D : C \times K \to M$ is the decryption function. The property $D(E(m,k),k) = m$ is verified for all $m \in M$ and $k \in K$.

2.2 Steganography

As explained by Katzenbeisser in [16], Steganography (first named by Johannes Trithemius, comes from the Greek words *"steganos"*, meaning "concealed", and *"graphein"*, meaning "writing" [15, 16, 19–21]) can be briefly described as "the practice of undetectably altering a Work to embed a message". This means the sender will take some random "innocent" message and embed it with the secret message. This produces an altered version of the host message that still appears "innocent" but carries the secret message.

In short, steganography can be seen as the process of communicating while preventing third parties from being aware that the communication itself is happening.

2.2.1 Motivation for Steganography

With standard encryption systems, a *ciphertext* is sent in the place of the *plaintext*. This *ciphertext* can be intercepted by third parties but they would not be able to extract the *plaintext* from it without the appropriate key. It will be, however, very apparent to third parties that the message has been encrypted. *Ciphertexts* usually appear as random sequences of characters or bits and it should be immediately obvious that they are not simple human readable messages. There are specific situations in which the message sender or receiver might see this as an issue and desire further security through iniquity. In certain contexts, the communication channel provider might refuse to relay messages that it does not

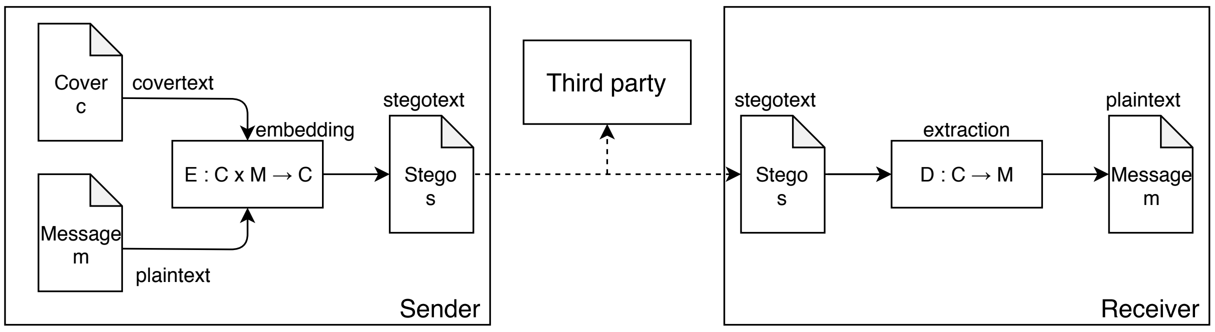

Figure 2.2: Diagram showing a generic setting for steganography. The sender embeds the *plaintext* message m into the *covertext* c, producing the *stegotext* s. This *stegotext* is sent to the receiver who uses the extraction function to recover the original *plaintext*. A third party might see the *stegotext* but it should be innocuous enough that no suspicion would be raised to the fact that it is carrying a message.

know are trustworthy. More recently, some governments are planning to outlaw or to regulate the usage of some encryption systems [22, 23] and others have already begun to do so [24, 25].

In the described contexts, Steganography is one of the more obvious solutions for these concerns.

2.2.2 The Steganographic Process

To paraphrase the previous sections, the process of using a steganographic system consists on inserting or embedding the *plaintext* (or payload) into some cover message (often referred to as the *covertext*), this produces an altered version of the *covertext* that is called the stego-object or the *stegotext*. This *stegotext* can be sent over an unsafe channel and should appear innocuous (should not raise suspicion that it is carrying a hidden message). The message receiver can then use the steganographic system to extract the *plaintext* from the *stegotext*.

As described by Kingslin in [17], a steganographic process can be divided into two components:

- An embedding or injection method, where a *covertext* is modified to receive the *plaintext*, outputting the *stegotext*. These functions make use of the redundancies in the *covertext* and exploit them as the space in which the *plaintext* will be inserted. This is performed by the message sender.
- An extraction method, where the *plaintext* is extracted from the *stegotext* (in some systems, the original *covertext* can also be obtained here). This is done by the message receiver.

A diagram explaining the usage of these two functions to hide and send messages can be seen in Figure 2.2.

Definition 2.2.1. A steganographic system (or scheme) can be defined as a quadruple $< C, M, E, D >$, where C is the set of possible *covertexts* (messages that are innocuous and would not raise suspicion to a third party), M is the set of possible *plaintexts* (the set of all messages over the *plaintext* alphabet Σ, Σ^*), with $|C| \geq |M|$, $E : C \times M \to C$ is the embedding or insertion function, $D : C \to M$ is the extraction function. The property $D(E(m)) = m$ is verified for all $m \in M$. The embedding and extraction functions might take additional parameters, such as keys, depending on the system.

2.3 Embedding Rate

The embedding rate, or hidden information capacity of a steganographic system refers to the amount of hidden information that can be embedded into a cover message. Usually, if the hidden message and the cover message can be expressed as binary files, the embedding rate can be given by the size of the *plaintext* file, divided by the corresponding *stegotext* file size.

2.4 Purity of Steganographic Systems

Steganographic systems can be classified according to the amount of required prior information exchange between the message sender and receiver. Usually this information exchange relates to some security measure of the steganographic system, such as a secret key. More generically, what needs to be exchanged are some additional parameters that are needed in both the embedding and extraction methods. Classifying steganographic systems based on this is relevant, since situations that require the usage of steganography usually deal with unsafe communication channels, and the prior exchange of information might not be feasible.

The following are the three classifications most commonly discussed when studying this property of steganographic systems [15, 26, 27]:

2.4.1 Pure Steganography

A pure steganographic system, as formalized by Katzenbeisser in [15], is a steganographic system that does not require the prior exchange of some secret information. These systems are solely secured by the iniquity of the *stegotext* and rely on the presumption that no third party would be aware that there exists some hidden message [26]. The system formalized in Definition 2.2.1 (with no additional parameters on the embedding and extraction functions) is one such system.

2.4.2 Secret Key Steganography

A secret key steganographic system is defined as a system that requires the prior sharing of a secret key. This secret key, often called *stego-key* [26], is required as an additional parameter in the embedding and extraction functions. Secret key steganography shares some similarities and is closely related to symmetric cipher encryption (described in Section 2.1). Many of these systems can be seen as applying a pure steganography system on top of a symmetric cipher encryption system. Katzenbeisser [15] considers that these systems that require the prior sharing of information subvert the original intention of information hiding.

Definition 2.4.1. A secret key steganographic system can be defined as a quintuple $< C, M, K, E, D >$, where C is the set of possible *covertexts*, M is the set of possible *plaintexts*, with $|C| \geq |M|$, K is the set of possible secret keys, $E : C \times M \times K \to C$ is the embedding or insertion function, $D : C \times K \to M$ is the extraction function. The property $D(E(m, k), k) = m$ is verified for all $m \in M$ and $k \in K$.

2.4.3 Public Key Steganography

Public key steganographic systems take concepts from public key cryptography (described in detail by Ferguson [14]) for added security. These systems require the usage of two keys, one public and one secret. The message receiver will generate both keys using some key generation function and will place the public key in some publicly available source. The public key can then be used by the message sender in the embedding function to generate the *stegotext*. To extract the *plaintext* from the message, the original secret key needs to be used in the extraction function. This key will only be accessible to the message receiver who generated it since it is never necessary to share it.

2.4.4 Natural Randomness of Covertexts

Steganography systems that make use of cryptography have the additional advantage that, if a third party knows the specific steganography system used to generate some message and intercepts said message, the third party will have no way to be sure of or prove that the message is carrying a hidden message. If the extraction function can be applied to any cover message and output something, the third party will have no way to prove that the output, which will likely be apparently random characters, is some encrypted message. The natural randomness of some *covertext* should be indistinguishable from the *ciphertext* produced by some cryptosystem [15].

To ensure the security of the steganographic system, it must be assumed that the third party that will intercept the message will know that a certain steganography scheme can be used and will test the message for that scheme.

2.5 Embedding Functions

The embedding and extraction functions are the most important part of a steganographic system and certainly the most difficult part to define. As inverse functions, these two methods are co-dependant and need to be jointly defined. For their relevance, steganographic systems can be classified according to the working principles of these functions.

In [16], Kaufmann proposes the following classifications for steganographic systems, according to the behavior of the embedding function (and consequentially of the extraction function):

2.5.1 Steganography by Cover Modification

Steganography systems in which the embedding function alters an existing *covertext* are called steganography by cover modification. This is, without doubt, the most common working principle of steganographic systems and is the one shown in Figure 2.2. In [28], Osman considers that this category can be further divided into substitution-based systems, in which parts of the cover message are replaced; and injection-based systems, in which new elements are inserted into the cover message.

2.5.2 Steganography by Cover Synthesis

The generation of a *stegotext* based on the *plaintext* is called steganography by cover synthesis (or generation). This type of steganography is less common due to the fact that it is complicated to generate a cover message that is innocuous and does not appear to be generated.

2.5.3 Steganography by Cover Lookup

Steganography by cover lookup describes steganographic systems in which the cover messages are preexisting and not modified in any way. In these systems, the message sender will use the extraction function on all available cover messages and choose the one that produces the desired *plaintext*.

2.6 Cover Messages

Most Steganography methods can embed a hidden message of any nature into a cover message of a specific nature, *i.e.* the embedding and extraction functions will be constructed for the specific given source of covers [16]. As such, Steganography methods are usually classified according to the type of cover message they work with [17, 21, 29, 30]. In the digital age, with such a wide variety of media types and file formats, steganography methods have been developed for almost all types of possible cover messages. The following are the families of cover message that are most frequently used for steganography.

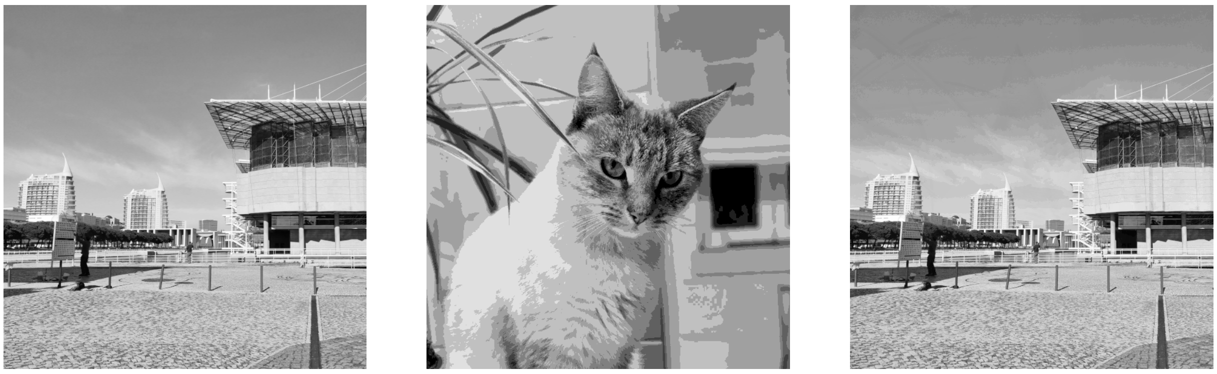
Figure 2.3: Demonstration of an image steganography system, using Stanley's implementation available in [1]. This is an LSB (lowest significant bits) encoding system. The three least significant bits in the pixels of the cover image (left) are replaced with the most significant bits of the payload image's pixels (middle). The resulting image (right) appears largely unchanged except for a slight quality loss and slight noise. These alterations are more obvious in the smoother areas of the image, such as the sky.

2.6.1 Image Steganography

Image steganography is the practice of using an image as the cover message. In modern times, image steganography has become one of the most common types of steganography. Images are very commonly shared over the internet, and as such, they are a very innocuous format for a cover message. Additionally, image files are usually very big compared to the ones used in other types of steganography. This means they are prone to have more redundancies that can be exploited by a steganographic system [30].

Image steganography systems usually deal with manipulating the least significant bits of certain (or all) pixels [30]. This results in a slight noise over the original image that should be unnoticeable. Images that are more noisy or textured are a better candidate for a cover in these systems. The usage of such a system is demonstrated in Figure 2.3.

Other more advanced methods can deal with parameters that are specific to certain compression methods or image formats, some also use operations over statistical properties of the image [29]. These systems include DCT [31] and DWT [32] based image steganography systems.

2.6.2 Audio Steganography

An audio steganography system is a steganography system that uses audio files as cover messages.

Much like image steganography systems, most audio steganography systems use digitized audio formats to perform operations on the file bits [30]. LSB (least significant bit) encoding is very common for these systems but they are usually used with some error diffusion methods to make the resulting distortion less noticeable [29].

Other systems apply operations directly on the audio signal, these include phase coding, which

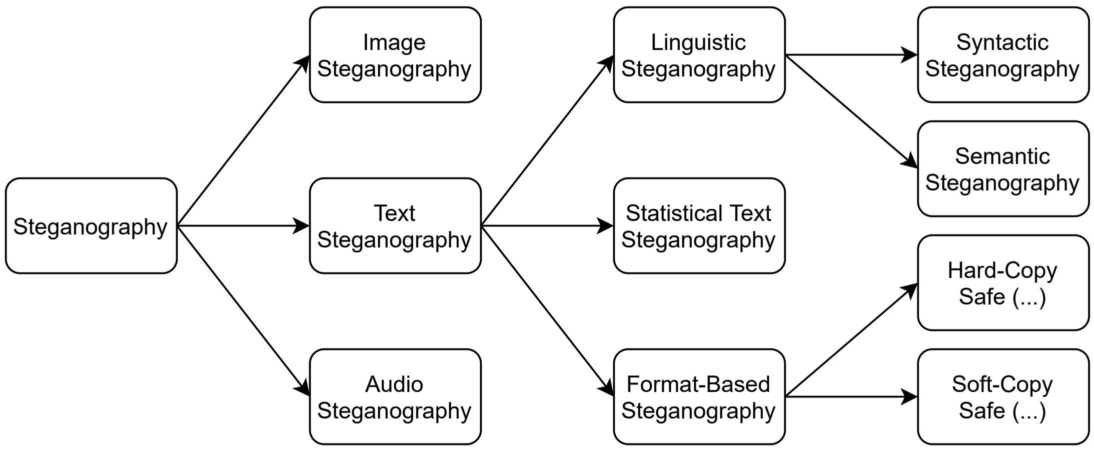

Figure 2.4: Diagram showing hierarchy of the major families of steganographic systems.

encodes the secret message as phase shifts in the phase spectrum of a digital signal, and spread spectrum systems, which multiply the digital signal by a certain noise signal [30].

2.6.3 Steganography on other Multimedia Formats

With the emergence of new multimedia formats, more applications for steganography have appeared. These new formats, such as video, can often be seen as combinations of other more “primary” formats, such as image and audio [29]. For this reason, steganography can be applied to such formats by using the existing image and audio steganography systems on the components of the cover message.

2.6.4 Text Steganography

Text steganography is the family of steganographic systems that use text as the cover message. Historically, writing has been one of the oldest forms of communication over long distances. As such, it is likely that text steganography is the oldest form of steganography. Despite this, text steganography is still seen as the most difficult kind of steganography. As Sharma describes it in [33], this is because a text file lacks a large scale redundancy of information in comparison to the other digital media formats described in this section.

A large variety of systems has been developed to work with these kinds of messages. Some systems are specific to the file format and usually deal with minor alterations to the form that the text is displayed in, while others perform changes to the text itself. In the following sections we will go over some of the families and classifications of text steganography systems. A diagram showing the hierarchy of such systems can be seen in Figure 2.4.

2.7 Text Steganography Systems

Due to its longer history, text steganography is an area of research that has seen the development of some very different approaches. In this section we will go over the broader classifications of text steganography, these differ mostly in which elements of the message are affected in order to receive the hidden message.

2.7.1 Format-Based Text Steganography

Text steganography systems that alter the formatting of the text are called format-based steganography systems. Altering the formatting of the text might involve things such as slightly altering the size or color of letters, moving words or sentences a few millimetres, or even adding extra spaces between words [10, 17, 21, 29, 30]. These systems are the most commonly used for text steganography.

In [10], Bender states that these systems can be further divided into two categories: "Soft-copy safe systems", which are the systems in which the hidden message is not lost if the text is copied onto a different file, these include the insertion of spaces between words; And "Hard-copy safe systems" which are systems in which the text formatting is closely related to the specific file format of the text, in these systems the hidden message is likely to be lost if the text is copied onto some other file, Bender [10] described that these systems can be treated as a "highly structured image".

These systems have the problem that many communication channels, such as online messaging systems, will detect and fix what they will consider "formatting errors" and the hidden message can be easily lost. Another vulnerability, described by Agarwal [21], is that these changes in formatting can be easily detected by opening the text in a word processor.

2.7.2 Statistical Text Steganography

Statistical text steganography, often also called random or generative text steganography [17, 20, 21], is the branch of text steganography that deals with hiding information in statistical properties of the *covertexts*. To achieve this, most statistical steganographic systems usually deal with generating the *stegotext* itself (a process mentioned in Section 2.5.2). The *stegotext* is generated in such a way that the desired statistical properties of the text are verified.

The most simple example of such a system would be the "cover lookup" system described by Kaufmann in [16]. In this system, the message sender has a set of possible cover messages (or generates them) and simply selects the one that, using a specific hashing function, hashes to a desired *plaintext*. More advanced systems of this branch include the mimic functions described by Wayner [34, 35].

2.7.3 Linguistic Text Steganography

Text steganography systems that deal with the linguistic properties of the *covertext* are called linguistic steganographic systems [17, 21, 29]. These systems perform modifications on the text itself and exploit the ambiguities or redundancies of natural languages. As the topic of this article, this family of systems is further studied in subsequent sections.

2.8 Linguistic Steganography Systems

As described by Kinglslin [17] and Singh [29], the family of linguistic steganography systems can be further divided according to which linguistic properties of the text are being used to embed the *plaintext*. As such, the following two sub-families of linguistic steganography can be formalized:

2.8.1 Syntactic Text Steganography

Linguistic steganography systems that deal with the syntax of text are called syntactic text steganography systems. Such systems might change the grammatical structures of sentences to embed a hidden message. Simpler systems in this family might, for example, simply add or remove commas from text in places where their necessity is arguable (such as the Oxford comma).

2.8.2 Semantic Text Steganography

Semantic text steganography is the branch of text steganography that uses the redundancy of words as the space for the hidden message. Languages are naturally ambiguous tools for communication and that makes them a viable vehicle for steganography.

Most steganographic systems in this family replace words in a cover message with their synonyms.

Trivial implementations of such systems label words and their synonyms with a binary value. The message sender identifies the words that can be replaced in the covertext, and, depending on the desired bit from the *plaintext*, will choose to keep the original word or replace it with its synonym. The message receiver will do the same process and identify the message sender's choices to determine the hidden message bits.

As the topic of this article, we will go over existing implementations of such systems in the following section.

3

Related Work

Contents

3.1 Synonym Table Steganography Systems 19
3.2 Other Approaches for Semantic Steganography 25

Table 3.1: Example of a short synonym table, used by Bender in [10].

big	large
small	little
chilly	cool
smart	clever
spaced	stretched

The focus of this article is to propose a novel approach for semantic steganography. To do so, we first survey existing semantic steganography systems and describe them in this section.

3.1 Synonym Table Steganography Systems

Semantic steganography systems use the redundancy in the words of natural languages as the space for a hidden message. The most trivial implementation of such a system would be one that replaces words in the *covertext* with their synonyms.

In our survey, the majority of steganographic systems that replace words with their synonyms make usage of a synonym table (exemplified in Table 3.1). These tables, of usually two columns, pair words with their synonyms.

In these systems, the hidden message is encoded into the choice of synonyms that was used in the *covertext*. This way, each word in the *covertext* (that can be replaced by a synonym) will encode a character of the *plaintext*, corresponding to which column of the synonym table it is in.

Synonym table steganography systems require that the synonym table is shared by the message sender and receiver, this means that these systems cannot be classified as pure steganographic systems (see Section 2.4.1).

In the approaches described by Bender [10], Rafat [36], and Shirali-Shahreza [11, 12], the *plaintext* is first converted into a binary string. This way, a two-column synonym table can be used to encode the hidden message (there is one column for each character of the hidden message alphabet $\Sigma = \{0, 1\}$).

3.1.1 Embedding Method

In all of these systems [10–12, 36], the embedding method functions as follows, for a given *covertext* and *plaintext*:

1. The *plaintext* is converted into an alphabet Σ such that $|\Sigma| = c$, where c is the number of columns in the synonym table.

2. The *covertext* is scanned and occurrences of words in the synonym table are identified.

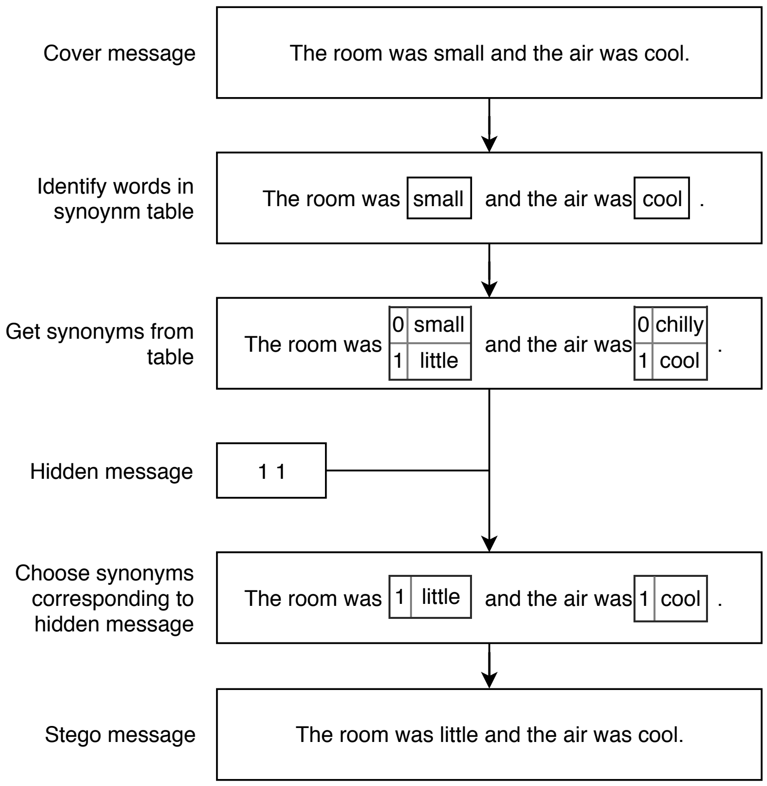
Cover message
The room was small and the air was cool.
Identify words in synoynm table
The room was small and the air was cool .
Get synonyms from table
The room was 0 small 1 little and the air was 0 chilly 1 cool .
Hidden message
1 1
Choose synonyms corresponding to hidden message
The room was 1 little and the air was 1 cool .
Stego message
The room was little and the air was cool.

Figure 3.1: Diagram exemplifying the embedding process of the *plaintext* “11” into the *covertext* “The room was small and the air was cool.” using a semantic steganographic system with the synonym table shown in Table 3.1.

3. The n^{th} identified word of the *covertext* is replaced with a synonym from the table’s column corresponding to the n^{th} character of the *plaintext*.

This embedding method is further clarified in Figure 3.1.

3.1.2 Extraction Method

The *stegotext* generated by the message sender using the aforementioned embedding method is sent to the message receiver which will apply the corresponding extraction method. The extraction method for these systems can be described as follows:

1. The *stegotext* is scanned and occurrences of words in the synonym table are identified.

2. The n^{th} character of the *plaintext* will correspond to the column of the n^{th} identified word of the *stegotext*.

This extraction method is further clarified in Figure 3.2.

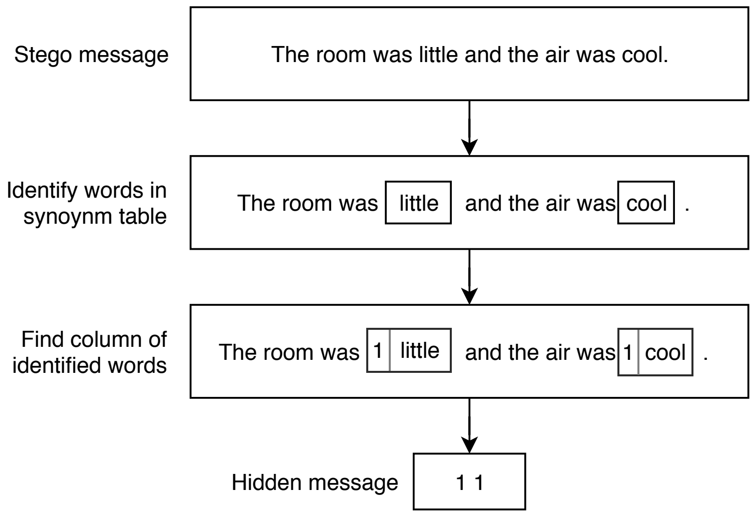
Stego message
The room was little and the air was cool.
Identify words in synoynm table
The room was little and the air was cool .
Find column of identified words
The room was 1 little and the air was 1 cool .
Hidden message
1 1

Figure 3.2: Diagram showcasing the extraction process from the *covertext* generated in figure 3.1. This process requires the same synonym table shown in Table 3.1.

3.1.3 Synonym Tables

In describing synonym table steganography, the authors in [10,33] explain its usage in a generic context. In [2], the author described this as the "naive" implementation of a semantic steganographic system. In regard to the synonym table itself, these authors simply described it as a table that pairs words with interchangeable synonyms and offered no source as to how the table would be constructed or to which specific words could be used. It is not entirely trivial how these tables should be constructed. Words that seem synonymous in certain contexts might not be interchangeable in other contexts [2, 10]. Other authors expanded on how these systems could be constructed by exemplifying specific sets of words that could be used for the synonym tables.

In [11], Shirali-Shahreza explored the usage of words that have different spellings in American English and European English. This approach is advantageous in that there should be no occurrence of two words in the table not always being interchangeable. In Table 3.2, a synonym table of such a system is exemplified. In their article, the authors pointed out that such words do not occur very frequently (or not as frequently as words that can have any synonyms) and that such a system would consequentially have a very low information capacity. This approach might also not be ideal for usage in the United States or in the United Kingdom, where the inconsistency with spelling might be more apparent to native speakers.

The substitution of acronyms and their unabbreviated counterparts can also be used in place of synonyms. In [12], the authors explored the application of the abbreviations and acronyms commonly used in SMS messages for such a system. Their approach would ideally be applied to SMS messaging where the usage of the aforementioned acronyms would be most innocuous. An obvious disadvantage

Table 3.2: Example of a short synonym table using words that have different spellings in American English and European English, used by Shirali-Shahreza [11].

American English	European English
account	bill
candy	sweets
color	colour
faculty	staff
fall	autumn

Table 3.3: Example of a short synonym table using the acronyms and abbreviations commonly found in SMS messaging, partly taken from [12] and extended with some more modern expressions.

Abbreviated	Unabbreviated
asap	as soon as possible
sry	sorry
lol	laughing out loud
np	no problem
hmu	hit me up

of this, however, is that SMS messaging is not an ideal channel for steganography, due to the character limit of text messages. In the described system, only about three bits of information could be sent per message. As such, hundreds of separate messages would be needed to send short hidden messages, which would damage the system's innocuity. Table 3.3 is an example of a synonym table that could be used for such a system.

The work of Shirali-Shahreza [12] was extended by Rafat's research [36]. In this article, the author explored the expansion of the security of the system by using a *stego-key* to shuffle elements of the synonym table between the left and right columns. For his implementation, the author used a process of XoR-Encryption supported by a Linear Feedback shift register to perform the shuffling. This system is more secure in that a third party that might know the system would still be unable to extract the hidden information without the *stego-key*. For this reason, this system would be classified as a secret key steganographic system (see Section 2.4.2).

Rafat's approach does, however, have some vulnerabilities. Shuffling the synonym table for security means that expressions in the table will always correspond to the same character of the hidden alphabet. For example, a third party that knows the system but does not know the *stego-key* would not know which bit is encoded in an instance of the acronym "np" in the *covertext*. However, this third party will know that all instances of "np" encode the same bit value which consequentially is different from the bit value of all instances of "no problem". This means the system might be vulnerable to some statistical analysis methods.

As described by Ktazenbeisser [15], a simpler and safer approach for security in these systems would be to simply stack it on top of a symmetric cipher encryption system (see Section 2.1). The *plaintext* is first encrypted into the *ciphertext* and that is embedded into the *covertext*. The *stego-key* is the key used in the encryption phase.

3.1.4 Synonym Table Construction

Manually constructing a synonym table for a semantic steganographic system is a long and arduous process. The size of the synonym table is vital for maximizing the hidden information capacity of a semantic steganographic system. For this reason, it is ideal to automate the generation of the synonym table.

In their survey for comprehensive public domain synonym file, the authors in [37] cite the Moby thesaurus [38] as the best source for a synonym table. The Moby thesaurus was developed by Gary Ward as part of the Moby project in the The Institute for Language, Speech, and Hearing at the University of Sheffield, England. It contains about 30,000 keywords with a total of more than 2.5 million synonyms and related terms. This dataset includes a large set of synonym clusters. Each synonym cluster is a set of words that share the same meaning in a certain context.

In Winstein's article [2], WordNet [6, 7] is cited as another good source for a synonym table. WordNet is a large lexical database of English developed in the Princeton University. Similarly to the Moby thesaurus, WordNet groups words into sets of cognitive synonyms (here referred to as *synsets*), each expressing a distinct concept. A word might appear in multiple *synsets*, depending on the various meanings it might take.

In both of the aforementioned articles, the authors mentioned that there were some concerns about pairs of synonyms not being interchangeable in every context. For example, in the sentence "We will be here for a long time.", the word "time" is interchangeable with its synonym "period", this, however, does not hold true for the sentence "Can you time my race?". To solve this problem, two approaches can be applied:

- Delete synonym pairs of words that are not interchangeable in every context.
- Find the context-specific synonyms for a word. This involves identifying the context in which the word is appearing, and finding the set of words that are interchangeable with it in that specific context.

Neither solution is trivial to implement. For their approaches, both of the authors decided to use the first solution and both implemented a very identical system. They used the *synsets* (or synonym clusters in the Moby thesaurus) to identify which pairs of words are always interchangeable.

Definition 3.1.1. A *synset* S_m is the set of all words that can, under some context, convey the meaning m.

A *synset* is a set labeled by a certain meaning or sense, and that contains all words that can convey that meaning. If a word is in a given context conveying a certain meaning, it can be replaced with any

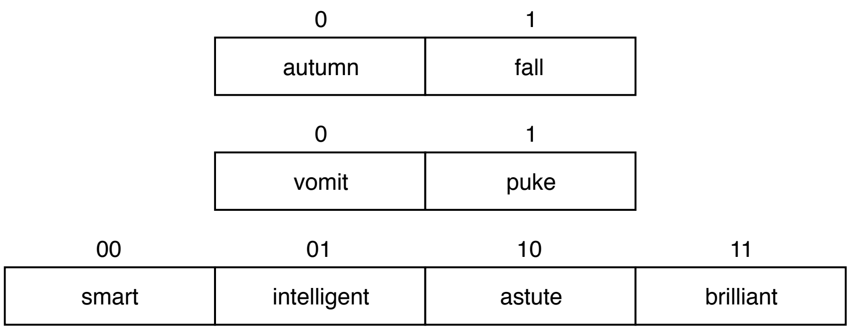

Figure 3.3: Examples for sets of synonyms and the bits they can encode, as described by Winstein in [2].

word in the *synset* of that meaning. Since it is not trivial, for a word that has multiple meanings and appears in multiple *synsets*, to determine which meaning it is being used for, it is not trivial to determine which words it can be replaced with. The solution provided in [37] and [2] is the following: Only replace words with synonyms that appear in all *synsets* that the word appears in. This way, regardless of the meaning that the word might be taking, the synonyms used are guaranteed to also convey that same meaning.

This process can be formalized as follows:

Lemma 3.1.1. *Let $S(w) = \{S_{m_1}, S_{m_2}, ..., S_{m_n}\}$ be the set of all synsets in which a word w, of n possible meanings, appears. The set of all words that can always replace w can be given by $R(w) = \bigcap\limits_{S \in S(w)} S$.*

3.1.5 Synonym Plurality

In the examples described in previous sections, the synonym table has a set number of columns and, as such, all words in such synonym tables are restricted to having that set number of possible replacements (usually just one, for the embedding of a binary string). This is rather restrictive since some words (usually the most common ones) can have a large number of synonyms. These words have the potential to encode more information. In our survey, we found some literature on approaches that allow for words to have a variable number of synonyms.

The most trivial solution for this problem is the one described by Winstein [2] in his description for a “naive algorithm”. The described approach groups words into sets of mutually interchangeable synonyms. The system embeds a binary message into the *covertext*, each word can embed as many bits as the base two logarithm of the number of words in its synonym set. As such, the number of elements in these sets of synonyms is restricted to being some power of 2. This approach is exemplified in Figure 3.3. A similar approach is also used in [37] and [39].

In [2], Winstein improves on the aforementioned “naive algorithm” by proposing a related system in which the synonym sets can have any number of words (as opposed to only powers of 2). His proposal

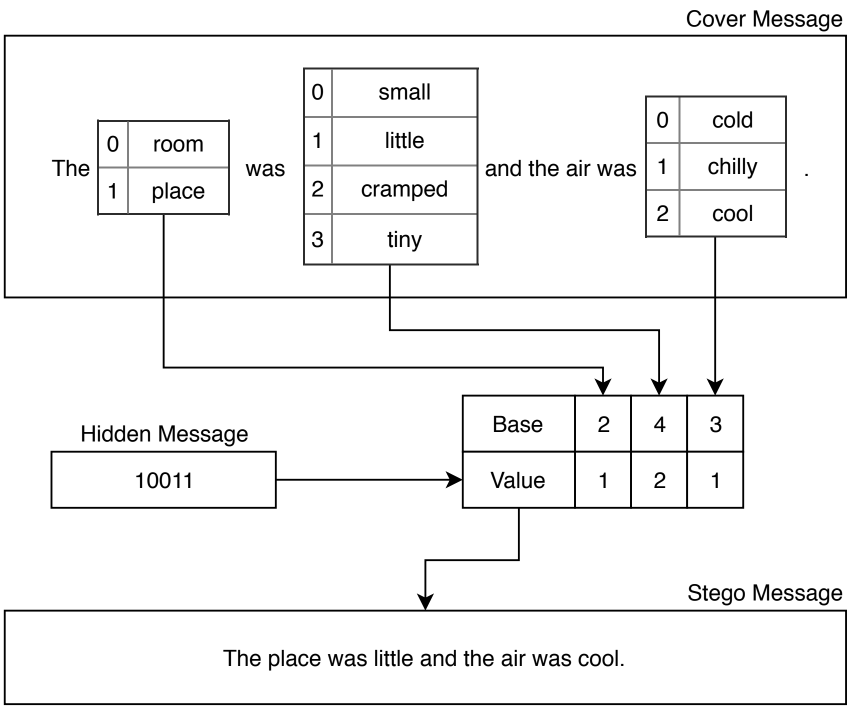

Figure 3.4: Diagram showcasing the embedding of the hidden message "10010" into the *covertext* "The room was small and the air was cool.", using the *multi-base number* approach described by Winstein in [2].

consists on converting the hidden message into a *multi-base number* (each digit may have a different base), where each digit corresponds to a word in the synonym table, and the base of each digit is the number of replacements that word can have. This solution can be visualised with the diagram in Figure 3.4.

3.2 Other Approaches for Semantic Steganography

More recently, some approaches for semantic steganography have been proposed that do not make use of synonym tables. Most of these proposals are classified as cover synthesis steganography (see Section 2.5.2) since they generate the natural language cover message that will embed a specific *plaintext*.

3.2.1 Synonym Distribution

The authors in [39] have explored how, in some semantic steganographic systems, common words might be replaced by very uncommon synonyms and how this might lead to suspicion from third parties. Some statistical analysis methods can be used to detect what the authors called "linguistic distortion". In their article, the authors described how this "linguistic distortion" metric can be used to select which words to replace, and propose a semantic steganography algorithm that maintains iniquity even in the statistical properties of the text. Some other approaches as described in this section do also tackle this issue.

3.2.2 Mimic Functions

A well known approach for semantic steganography is the one proposed by Wayner in his articles [34] and [35]. Here Wayner described the construction of mimic functions and their applications for text steganography.

A mimic function f is described as the function that alters the statistical properties of a text file A to be the same as some other file B. Formally, if $p(t, A)$ is the probability of a substring t occurring in A, then the mimic function f encodes A so that $p(t, f(A))$ approximates $p(t, B)$.

Wayner introduces mimic functions as the inverse of Huffman compression functions. A Huffman function (or Huffman code) is a type of optimal prefix code that is commonly used for lossless data compression, it was first proposed by Huffman [40].

The proposed approach is to construct the Huffman compression functions for the files A and B, f_A and f_B. The inverse of f_B, g_B is then computed. The composite function $g_B(f_A(A))$ is the first order mimic function that converts A to have the statistical properties of B. Larger order mimic functions can be computed by joining sequences of n characters together (for an n^{th} order mimic function) and interpreting them as being a single character.

This system can be seen as a cover synthesis steganographic system in which, for a hidden message A, and a cover message B, the mimic function $g_B(f_A(A))$ will generate a *stegotext* message that has the statistical properties of the *covertext* B. The *stegotext* outputted by this system will be text that contains words or even short expressions found in the *covertext* but that lacks any grammatical structure or sense. For a human third party, this *stegotext* will obviously raise suspicion. Wayner improved on his system by joining it with context-free grammars to ensure the sentences maintain grammatical consistency. This improved the iniquity of the *stegotext*, but it still remained mostly devoid of meaning.

3.2.3 Markov Chain Based Text Steganography

In [3], Dai introduced the usage of Markov chains for text staganography, this research was continued in [41]. Dai's proposal involves constructing a Markov model for the desired *covertext*.

A Markov model constructed from some text corpus would maintain the probabilities of any two consecutive words appearing in the corpus $p(w_i|w_{i-1}) = count(w_{i-1}, w_i)/count(w_{i-1})$. The model could be used to construct new text samples that mimicked the statistical properties of the corpus (much like the mimic functions described in Section 3.2.2). Higher order Markov models can be used, these models take in account more words to provide a more accurate probability of the next. An n^{th} order Markov model would compute and use the probabilities $p(w_i|w_{i-1}, w_{i-2}, ..., w_{i-n})$.

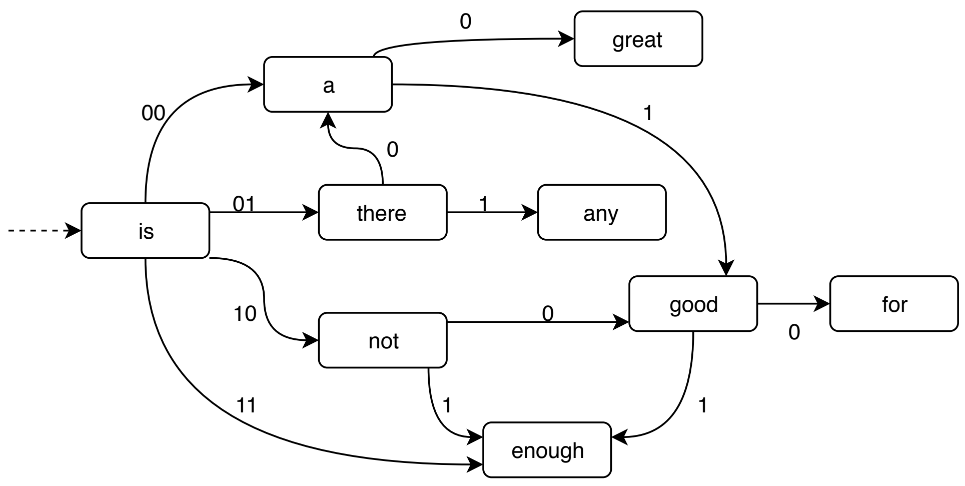

Figure 3.5: Diagram of a steganography system constructed using a Markov model, as described in [3]. To exemplify the function of the model, the hidden message "0100" would synthesize the *stegotext* "is there a great", while "1001" would synthesize the *stegotext* "is not good enough".

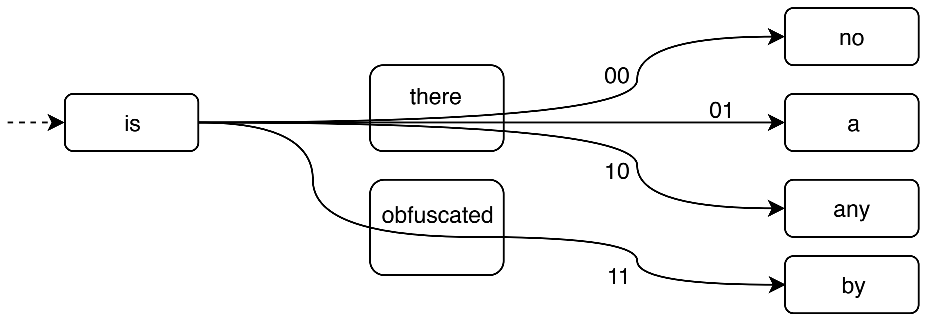

Figure 3.6: Diagram of grouped state transitions as described by Moraldo [4]. By grouping two consecutive transitions, the transition "is there" is made more probable than "is obfuscated", as expected in natural text. In Dai's approach, these transitions would be equiprobable.

In Dai's approach, the transitions of the Markov model are labelled with parts of the hidden message. To synthesise the *stegotext*, it is only necessary to use the *plaintext* to determine the sequence of state transitions that is done on the model. This process is exemplified Figure 3.5.

In [4], Moraldo described how these systems produce "unnatural" looking text by not taking into account the probability of transitions. With the way that transitions are labelled, any outgoing transition from any given state has the same probability of occurring in a *stegotext*.

Moraldo's solution involves grouping multiple consecutive transitions together and labeling these groups with parts of the hidden message. More probable state transitions will occur in more of these labelled groups. This way, the resulting *stegotext* will have more natural word sequences that occur with the frequency that is expected of a real text. This system is exemplified in Figure 3.6.

In [5], the authors also explore the problem of ensuring a natural probability distribution of transitions on a Markov based steganographic system. For their approach, the authors make use of Huffman coding

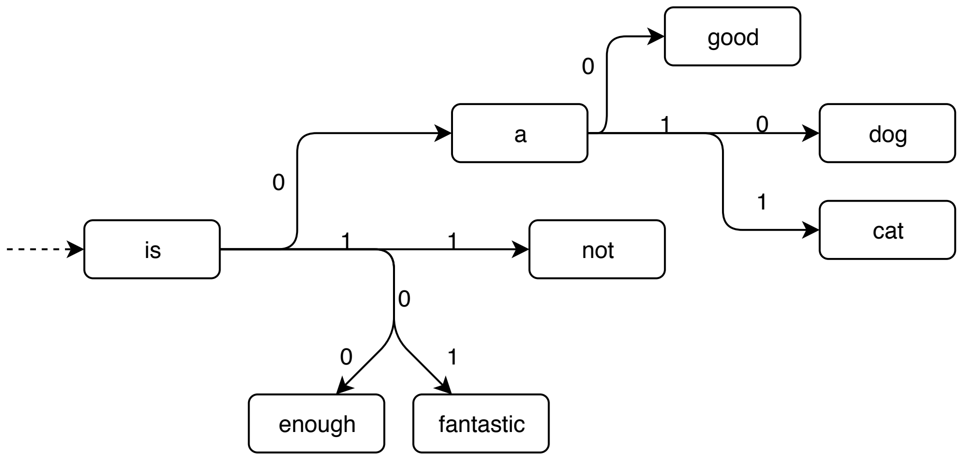

Figure 3.7: Diagram exemplifying the usage of “chained” Huffman trees to label transitions in a Markov model, as described by Zhongliang [5].

to construct a tree for the transitions at each step of the Markov model. More frequent transitions are labelled with shorter labels and are thus more likely to appear in the hidden message. This is exemplified in Figure 3.7. This system shares a lot of similarities with the mimic functions described by Wayner [34] and with Moraldo’s approach [4].

4

Approach

Contents

4.1 Replaceable Substring Identification 31

4.2 Substring Replacement Set Construction 32

4.3 Section Replacement Labelling 33

4.4 Hidden Message Extraction 33

4.5 Embedding Failure Probability 34

Semantic steganographic systems that use synonym tables can be thought of as requiring two main operations that are shared by the embedding and extraction functions.

- The identification of replaceable substrings in the cover message.
- The labeling of the possible replacements for the identified substrings with characters from the hidden message alphabet.

In synonym table semantic steganography, the hidden message is constructed from the concatenation of the labels of the identified replaceable sections.

It is the fact that these two operations can be performed independently by the message sender and receiver and provide the same results that allows for the sending of the hidden information. Synonym table steganographic systems enforce that the message sender and receiver use the same synonym table to ensure that both identify the same replaceable substrings and that both label each replaceable substring the same way.

Definition 4.0.1. A substring s of a message $m = m' \cdot s \cdot m''$ over an alphabet Σ is called replaceable in m if there exists another message $n = m' \cdot r \cdot m''$ that conveys the same meaning or information as m, with $s \neq r$. In the context of m, s is called interchangeable with, or a replacement of r.

In this section, we propose a novel algorithm for semantic steganography that does not require the message sender and receiver to share a synonym table. To do so, we first define how the two aforementioned operations can be performed in the absence of a shared synonym table.

For the following sections describing our approach, it is assumed that the message sender has access to some unspecified synonym table. This table can have a variable number of synonyms for each word.

4.1 Replaceable Substring Identification

Our approach to have the message sender and receiver agree on which will be the replaceable sections is the following: The cover message is divided into substrings (or sections) of a fixed size. This fixed size should be large enough to ensure that at least one replaceable substring can be found by the message sender inside the fixed-size sections. If each fixed-size section contains a replaceable substring, then the whole section can be thought of as a replaceable substring.

Lemma 4.1.1. *If a substring s of a message m is replaceable, then any substring $c \supseteq s$ of m is also replaceable in m.*

Proof. Given two substrings s and c of a message m. And given that s is replaceable in m. If $s \subseteq c$, then c can be written as $c' \cdot s \cdot c''$, and m can be written as $m' \cdot c \cdot m''$ or $m' \cdot c' \cdot s \cdot c'' \cdot m''$. Since we know that

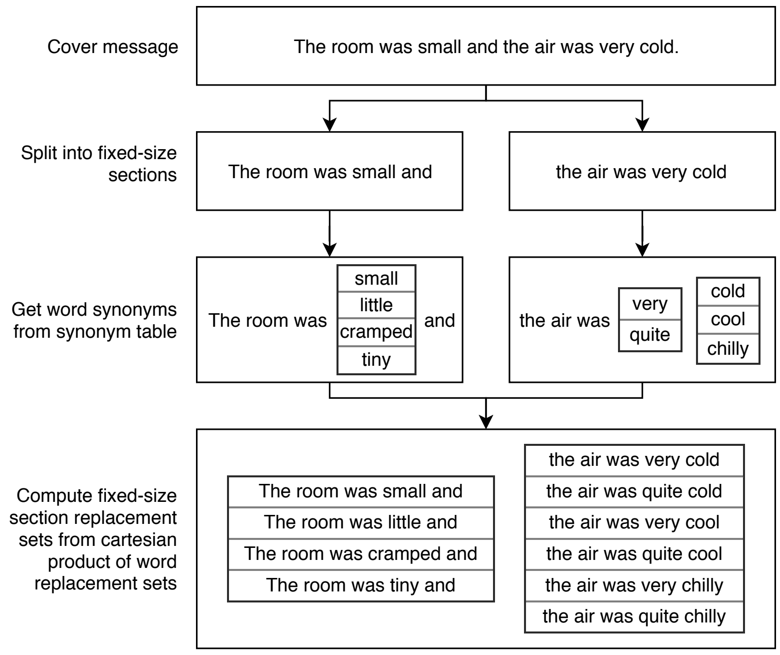

Figure 4.1: Diagram showing the process of computing the replacement sets for fixed-size sections of a message. In this example, the size of each section is 5 words.

s is replaceable, then there exists a message $n = m' \cdot c' \cdot r \cdot c'' \cdot m''$ that shares the same meaning as m. If we define $d = c' \cdot r \cdot c''$, and observe that $d \neq c$, then $m = m' \cdot c \cdot m''$ and $n = m' \cdot d \cdot m''$ share the same meaning, and d is, therefore, a replacement of c, and c is replaceable. $\square$

To try to maintain the replacement sets of each fixed-size section independent from other fixed-size sections, we can define the size of each section as a number of words. This way no word is cut between two sections.

To determine the ideal size of the fixed-size sections, the statistical properties of the text and the synonym table need to be studied. We go over the analysis of these properties in the following sections.

4.2 Substring Replacement Set Construction

With the identification of the replaceable sections established, the message sender needs to identify the possible replacements for each section. To do this, the message sender can use his synonym table to identify the replaceable words within each section and then the set of possible replacements for each word. If we are treating the entire fixed-size section as one replaceable substring of the message, we can compute its set of replacements from the Cartesian product of replacement sets of the identified replaceable words. A section will have as many replacements as the product of the sizes of replacement sets contained in it. This procedure can be visualised in Figure 4.1.

4.3 Section Replacement Labelling

In most synonym table steganographic system, word replacements are labelled according to their column in the synonym table. In our approach, since we are assuming that the message receiver does not have access to the table used by the message sender, an alternative labelling method needs to be implemented.

The proposed solution is one that is closely related to the "cover lookup" system described by Morgan [16]. A function that behaves similarly to a hashing function can be used to map fixed-size sections to characters of the hidden message alphabet. This function, which we will refer to as a stego-hashing function, should, for any given fixed-size section, deterministically output a character of the hidden alphabet, with uniform probability over the alphabet.

Definition 4.3.1. A stego-hashing function $H : S \to \Sigma$ is a function that maps strings over the cover message alphabet to the hidden message alphabet Σ and is selected from a strongly 2-universal family of hash functions [42] $\mathcal{H} = \{H : S \to \Sigma\}$. For any string s, hidden message character $m \in \Sigma$, and H a stego-hashing function uniformly selected from $\mathcal{H}$, **P**$(H(s) = m) = 1/|\Sigma|$.

To construct a stego-hashing function, any existing hashing function that operates on strings can be used. The output of such function just needs to be limited to the size of the hidden message alphabet, this can be done with the modulo operator. As such, if $h : S \to \mathbb{N}$ is an existing hashing function that operates over strings, we can define the stego-hashing function $H(s) = h(s) \bmod |\Sigma|$. The resulting value is used to index a character of the hidden message alphabet.

The message sender can use this stego-hashing function to compute the label for each section replacement of the cover message.

To embed the hidden message, the message sender will select any replacement of each section that hashes to (has been labelled with) the desired character of the hidden message. The n^{th} character of the hidden message will correspond to the label of the n^{th} selected fixed-size section replacement. This process is exemplified in Figure 4.2.

4.4 Hidden Message Extraction

To extract the *plaintext* from the *stegotext*, the message receiver simply needs to split the *stegotext* and compute the stego-hash for each section. The concatenation of these stego-hashes is the *plaintext*. This process does not make use of any synonym table. This is exemplified in Figure 4.3.

The major advantage of our approach over related systems becomes apparent in the extraction method. The extraction process is very light and does not require the sharing of a synonym table. The

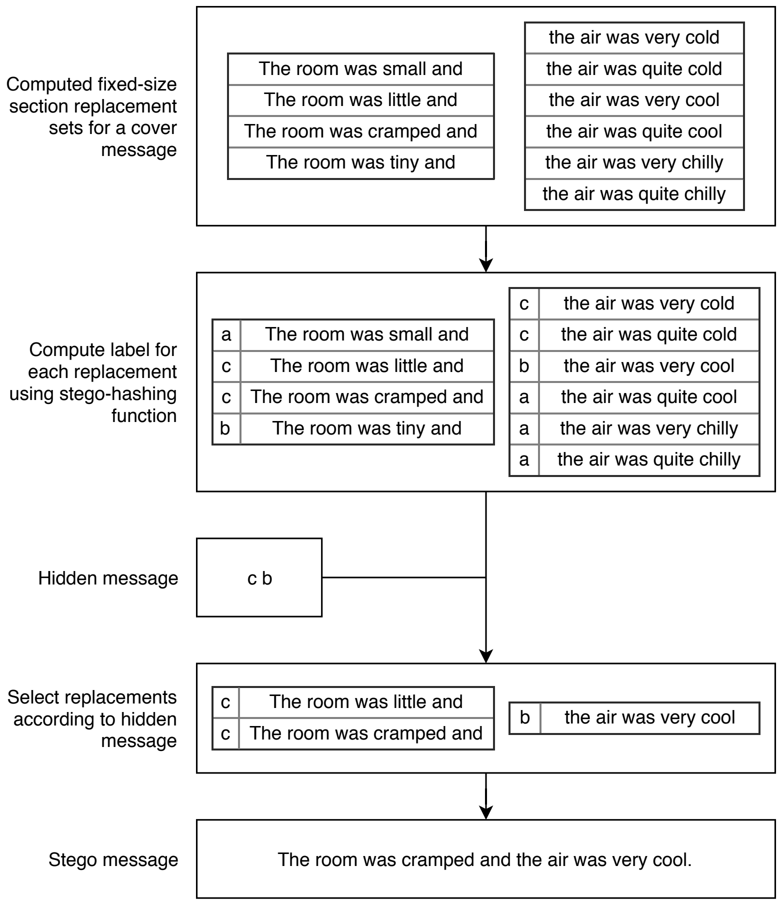

Figure 4.2: Diagram showing the embedding process of the proposed approach. This example uses the replacement sets computed for the cover message in figure 4.1. Here, the hidden message “cb” uses the alphabet $\Sigma = \{a, b, c\}$.

message sender and receiver need only to agree on the size of sections and on the used stego-hashing function, which should be encompassable in a very short message.

4.5 Embedding Failure Probability

When the message sender computes the possible replacements for each fixed-size section, there is a probability that none of the replacements will hash to the desired hidden alphabet character. If this happens, the embedding might be considered impossible for that specific *covertext* and *plaintext* pair. Because of this failure probability, the embedding algorithm can be considered a Monte Carlo randomized algorithm. In [16], the concept of embedding effectiveness is described as relating to this probability of an embedding failure.

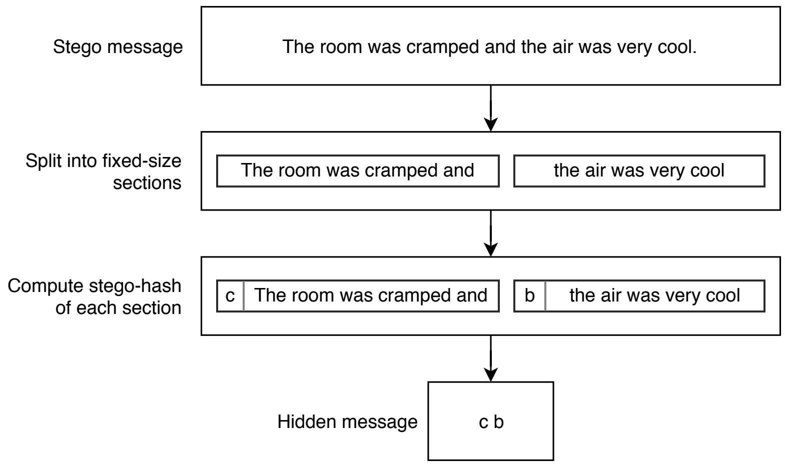

Figure 4.3: Diagram showing the extraction process of the proposed approach. This example uses the *stegotext* computed in Figure 4.2.

4.5.1 Estimating Embedding Failure Probability

The message sender and receiver will want to negotiate parameters for the system that minimize this failure probability. As such, it is relevant to estimate it.

If a hidden message can be described as an ordered set of characters from the hidden alphabet $M = \{m_1, m_2, ..., m_n\} \in \Sigma^*$ and a cover message can be described as an ordered set of fixed-size sections $C = \{c_1, c_2, ..., c_n, ...\}$, then the probability of M being embeddable in C (embedding probability of M in C), $EP(M, C)$ can be defined as the probability that each character of the hidden message is embeddable into its corresponding fixed-size section,

$$EP(M, C) = \prod_{i=1}^{n} EP(m_i, c_i).$$

Lemma 4.5.1. *Using the described system, the embedding probability for a hidden message character $m \in \Sigma$ on a fixed-size section c with r possible replacements can be formulated as:*

$$EP(m, c) = f(r) = 1 - (1 - \frac{1}{|\Sigma|})^r.$$

Proof. Any given section replacement s can embed a hidden message character m if it stego-hashes to that character. Due to the uniform nature of the stego-hashing function H that was uniformly selected from $\mathcal{H}$, the probability of s stego-hashing to m is

$$\mathbf{P}(H(s) = m) = \frac{1}{|\Sigma|},$$

and the failure probability for a single replacement is therefore

$$\mathbf{P}(H(s) \neq m) = 1 - \mathbf{P}(H(s) = m) = 1 - \frac{1}{|\Sigma|}.$$

Failure on a fixed-size section c happens when all its r replacements fail to hash to m,

$$FP(m,c) = \prod_{i=1}^{r} \mathbf{P}(H(s_i) \neq m) = \prod_{i=1}^{r} (1 - \frac{1}{|\Sigma|}) = (1 - \frac{1}{|\Sigma|})^r,$$

which is complementary to the embedding probability of m in c:

$$EP(m,c) = 1 - FP(m,c) = 1 - (1 - \frac{1}{|\Sigma|})^r.$$

□

Corollary 4.5.1. *Using the described system, the embedding probability for a hidden message $M = \{m_1, m_2, ..., m_n\}$ on a cover message $C = \{c_1, c_2, ..., c_n, ...\}$, where the i^{th} section of C has r_i possible replacements, can be computed as:*

$$EP(M,C) = \prod_{i=1}^{n} EP(m_i, c_i) = \prod_{i=1}^{n} f(r_i) = \prod_{i=1}^{n} (1 - (1 - \frac{1}{|\Sigma|})^{r_i}).$$

Corollary 4.5.2. *Since the embedding probability for a section only depends on the number of possible replacements for that section, we can group sections with the same number of replacements together. If we define $O(C,r)$ as the number of sections in C that have r possible replacements (occurrences of r in C), then the formula provided in Corollary 4.5.1 can also be written as:*

$$EP(M,C) = \prod_{i=1}^{n} f(r_i) = \prod_{r=1}^{\infty} f(r)^{O(C,r)} = \prod_{r=1}^{\infty} (1 - (1 - \frac{1}{|\Sigma|})^r)^{O(C,r)}.$$

The provided formulas are useful for computing the embedding probability for a known cover message that has been “pre-processed”. To use these formulas, the number of section replacements needs to be known for each fixed-size section in the cover message.

It is useful to establish a more “generic” formula for the embedding probability on a cover message. If the individual sections and their replacements are not known, the probability distribution of the number of replacements can be studied.

Lemma 4.5.2. *Using the described system, a hidden message M of length n, a cover message C divided into fixed-size sections of w words, where the probability of a section with w words having r*

replacements $RP(w,r)$ is known. The embedding probability of M in C has a lower bound:

$$EP(M,C) \geq (\prod_{r=1}^{\infty} f(r)^{RP(w,r)})^n = (\prod_{r=1}^{\infty}(1-(1-\frac{1}{|\Sigma|})^r)^{RP(w,r)})^n$$

Proof. By computing the expected value of the formula for the embedding probability provided in Corollary 4.5.2, with $O(C,r)$ as our random variable,

$$\begin{aligned}
\mathbf{E}[EP(M,C)] &= \mathbf{E}[\prod_{r=1}^{\infty} f(r)^{O(C,r)}] \\
&= \mathbf{E}[\exp(\ln(\prod_{r=1}^{\infty} f(r)^{O(C,r)}))] \\
&= \mathbf{E}[\exp(\sum_{r=1}^{\infty} O(C,r) \times \ln(f(r)))] \\
&\geq \exp(\mathbf{E}[\sum_{r=1}^{\infty} O(C,r) \times \ln(f(r))]) && \text{(Jensen's inequality)} \\
&= \exp(\sum_{r=1}^{\infty} \mathbf{E}[O(C,r)] \times \ln(f(r))) \\
&= \prod_{r=1}^{\infty} \exp(\mathbf{E}[O(C,r)] \times \ln(f(r))) \\
&= \prod_{r=1}^{\infty} f(r)^{\mathbf{E}[O(C,r)]}.
\end{aligned}$$

Jensen's inequality can be applied in this context since $\exp$ is a convex function [43].

The expected value of $O(C,r)$ can be computed from the size of C and the probability of a given section having r possible replacements. This way we get:

$$\begin{aligned}
\mathbf{E}[EP(M,C)] &\geq \prod_{r=1}^{\infty} f(r)^{RP(w,r)\times n} \\
&= (\prod_{r=1}^{\infty} f(r)^{RP(w,r)})^n.
\end{aligned}$$

□

This formula is more useful in that it can be used to estimate the embedding probabilities for unknown cover messages, based only on their length.

The probability of a section of w words having r replacements, $RP(w,r)$, is very dependant on the synonym table and it is not trivial to compute. To use this formula, this probability should be estimated from simulated samples of cover texts.

4.5.2 Estimating The Probability Distribution for Replacements of a Section

$RP(w, r)$ is defined in the previous section as the probability that a section of w words may have r replacements.

The number of replacements for a section is the number of possible combinations of replaceable words within the section. As such, this value is computed as the product of the number of alternatives for each word in the section. These numbers of alternatives for a word are obtained from the synonym table in use with the system.

Corollary 4.5.3. *The number of possible replacements for a section c, defined as an ordered set of words $c = \{x_1, x_2, ..., x_w\}$, where the i^{th} word has a_i alternatives (including itself), can be computed as*

$$R(c) = \prod_{i=1}^{n} a_i.$$

Alternatively, if we group words with the same number of alternatives together, and define $O(a, c)$ as the number of words in c that have a replacements (occurrences of a in c), the previous formula can be rewritten as

$$R(c) = \prod_{a=1}^{\infty} a^{O(a,c)}.$$

If we assume that the numbers of alternatives for words in a section are independent or almost independent, it is easy to compute the probability distribution for the number of alternatives per word from a synonym table. This can be done by randomly sampling words from candidate covertexts and counting their replacements. This way we compute the probabilities $\{p_1, p_2, ..., p_k\}$ of a word having $\{1, 2, ..., k\}$ alternatives, respectively.

Given these assumptions, each word can be interpreted as one of w independent trials, and that each will have an outcome that is a number a of alternatives with a known probability $\{p_1, p_2, ..., p_k\}$. As such, $O(a, c)$, the number of words in c that have a replacements follows a multinomial distribution.

To compute the probability that a section of w words may have r replacements, $RP(w, r)$, it is first necessary to determine how r can be described as a multiplication of the possible numbers of alternatives that words might have.

Corollary 4.5.4. *If for some value of $r \in \mathbb{N}$ there exist l sets $E_1, E_2, ..., E_l$, such that each set $E_i = \{e_{i,1}, e_{i,2}, ..., e_{i,k}\} \in \mathbb{N}_0^*$ verifies*

$$\sum_{j=1}^{k} e_{i,j} = w \quad \textit{and} \quad \prod_{j=1}^{k} j^{e_{i,j}} = r.$$

Then the probability that a section of w words may have r replacements can be computed as

$$RP(w,r) = \sum_{i=1}^{l} MPr(e_{i,1}, e_{i,2}, ..., e_{i,k}; w; p_1, p_2, ..., p_k),$$

where MPr is the probability mass function for a multinomial distribution, which can be written as

$$RP(w,r) = \sum_{i=1}^{l} \frac{w!}{e_{i,1}!e_{i,2}!...e_{i,k}!} p_1^{e_{i,1}} p_2^{e_{i,2}} ... p_k^{e_{i,k}}.$$

Using the described system, we first compute the combinations of word alternatives that are needed to get r replacements in a section. Then, knowing that the numbers of word replacements follow multinomial distributions, we compute the probability of each combination. The sum of these probabilities is equal to the probability of finding r replacements to a section of w words, or $RP(w,r)$.

5 Implementation

Contents

5.1 Natural Language Selection 43

5.2 Synonym Table Construction 43

5.3 Steganography System Implementation 46

5.4 Probability Estimation Functions 48

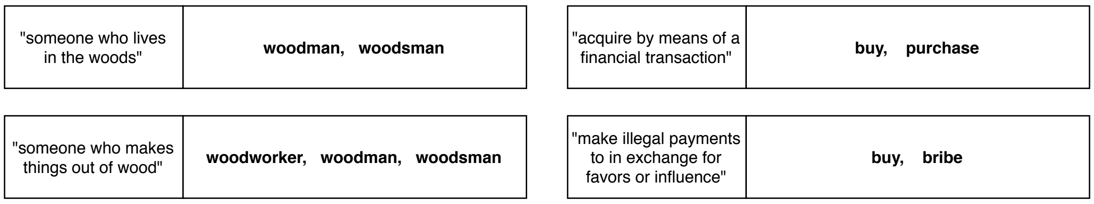
"someone who lives in the woods"
woodman, woodsman
"acquire by means of a financial transaction"
buy, purchase
"someone who makes things out of wood"
woodworker, woodman, woodsman
"make illegal payments to in exchange for favors or influence"
buy, bribe

Figure 5.1: Examples of some synsets in Wordnet [6, 7].

To implement the semantic steganography system described in Section 4, we first constructed a complete and static table of word replacements. This synonym table, mainly constructed with the Python programming language, is then imported to and used by a program that was constructed in Java, and implements the described embedding and extraction procedures.

5.1 Natural Language Selection

Our system was implemented in such a way that the synonym table and the steganography system are two somewhat divorced entities. As such, the language of cover messages is solely dictated by the language of words present in the synonym table. The steganography system is agnostic to the language being used and can change language by simply changing between the appropriate synonym tables.

For the language of the synonym table, we opted for the English language. This was motivated by the fact that, firstly, as the language of this document, using English allows for examples of the system to be shown and understood by readers of this document. And secondly, given its place as a *lingua franca* in the academic world, there is a plethora of resources that are available and facilitate the construction of an English synonym table.

5.2 Synonym Table Construction

5.2.1 Core Dataset

The base for our synonym table is WordNet [6, 7]. WordNet is a large lexical database of English constructed and made available by Princeton University. This database is composed of synsets. Each synset is a set of words labelled by a meaning. Each word in the synset can, in some context, take the meaning of the synset. As such, words that appear on multiple synsets can have multiple meanings and might be harder to replace if the specific meaning is not identified. An example of some synsets ins available in Figure 5.1.

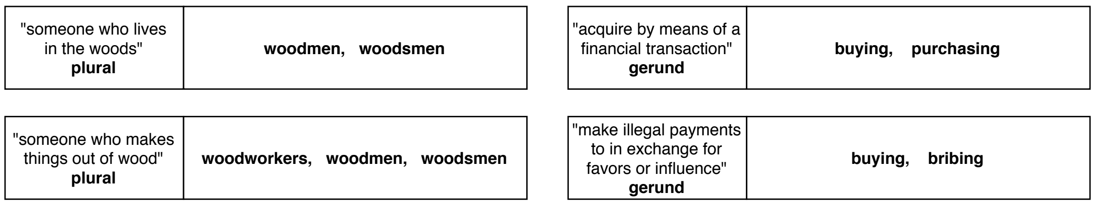
"someone who lives in the woods"
plural
woodmen, woodsmen
"acquire by means of a financial transaction"
gerund
buying, purchasing
"someone who makes things out of wood"
plural
woodworkers, woodmen, woodsmen
"make illegal payments to in exchange for favors or influence"
gerund
buying, bribing

Figure 5.2: Examples of some synsets created using Inflect [8] and MLConjug [8], from the synsets in Figure 5.1.

5.2.2 Expansion with Inflections

One limitation of WordNet is that all words are in their basic, non-inflected forms. As such, to create a complete dataset of words in the English language, a synset of plural nouns needs to be created for each noun synset in WordNet. The equivalent must be done for verb synsets and their conjugated counterparts.

To perform these operations, we first created a Python parser for the WordNet database. This is beneficial due to the large number of code libraries for natural language operations available for Python. To pluralize the words in noun synsets, the Inflect [8] library was used, which provides a number of operations for words and is mainly focused on pluralization. To inflect and conjugate the verb synsets, the MLConjug [44] library was used, which provides methods for conjugating verbs six different languages, including English. When each of these operations is applied to a synset, a new synset is created, and each word in this new synset is the converted counterpart of the related word in the original synset. An example of some of the generated synsets is provided in Figure 5.2.

5.2.3 Synset Intersection

The desired synonym table should map each word to a set of words that can replace it in any context, we define these as the ”safe” replacement words. This is the alternative to having a system that identifies which meaning is being taken by a word in a specific context. To do this, we use the various synsets that a word appears in to identify which words can always replace it, regardless of the meaning that it takes on in a given context. As such, the set of possible replacements for a word can be defined as the intersection of all synsets in which that word appears.

As was previously shown in Lemma 3.1.4, for a given database of synsets S and a word w, the set of ”safe” replacements for w, $R(S, w)$ can be computed as:

$$R(S,w) = \bigcap_{s \in S, s \ni w} s$$

To perform the described operation, we first mapped each word the synsets in which it appears, as shown in Figure 5.3.

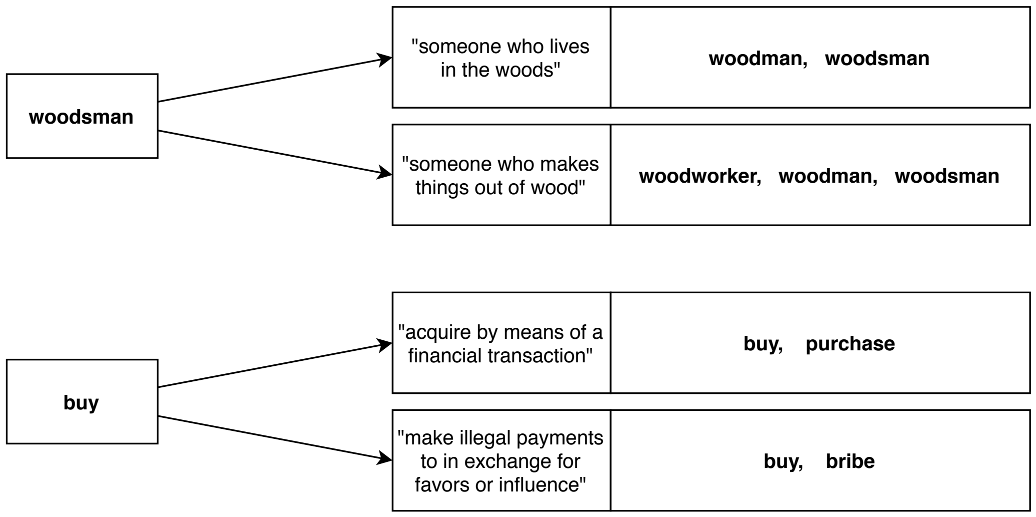
woodsman
"someone who lives in the woods"
woodman, woodsman
"someone who makes things out of wood"
woodworker, woodman, woodsman
buy
"acquire by means of a financial transaction"
buy, purchase
"make illegal payments to in exchange for favors or influence"
buy, bribe

Figure 5.3: Examples of some words and the synsets in which they appear.

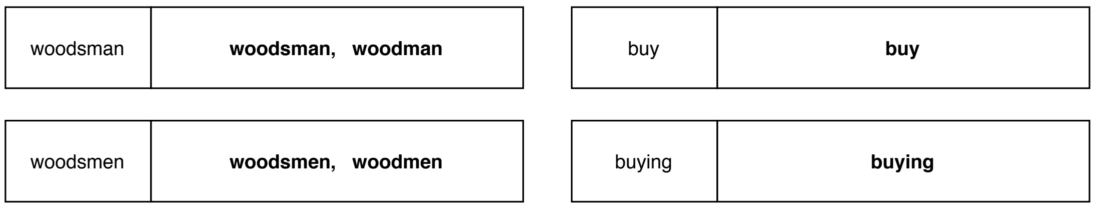

Figure 5.4: Examples of some words and the respective sets of ”safe” replacements.

Then, the set of ”safe” replacements of each word is computed from the intersection of all sets in which the word appears. This end result is exemplified in Figure 5.4.

Words whose replacement sets contain only the word itself are omitted. These words are considered non-replaceable and, for the purpose of the described system, they are no different from new words that were not expected by WordNet with the inflections.

5.2.4 Public Availability

The resulting replacement table is usable in many semantic steganography systems beyond the system proposed in this document.

Given that the licensing on all the components used allows for modification and commercial and non-commercial distribution, this replacement table was made openly available on a Github repository [45], along with the code used to create it.

5.3 Steganography System Implementation

The described steganography system was implemented with the Java programming language. This selection was motivated by the ease of programming string-related operations without sacrificing much on the speed of the embedding process, and the code readability that comes from object oriented programming.

5.3.1 Project Structure

The project was divided into multiple classes to help delegate the various operations and steps of the embedding and extraction processes.

The foremost classes are: The parser, which receives the covertext string and divides it into its various word and non-word elements; The replacer, which is an abstract class that encapsulates the synonym tables and provides the possible replacements for a single word or for an entire section. In the project this class is inherited once into the ”WordnetReplacer” class, which includes the code to parse and import the synonym table that was exported into a text file as described in Section 5.2; The hashing function is another abstract class, it includes the single method ”hash” which receives a string and returns a character from the hidden message alphabet; The embedder uses the previous classes to perform the embedding procedures, it contains the ”embed” function that receives the parsed covertext and the hidden message, and returns the corresponding stegotext message; The extractor performs the inverse ”extract” operation, which receives a parsed stegotext and returns the corresponding hidden message. A UML diagram showcasing the core structure of our implementation is provided in Figure 5.5.

5.3.2 Text Parsing

To separate the covertext into a sequence of words and non-words, we defined a word as any contiguous sequence of letters from the Latin alphabet. This meant that some possible nonsense words are parsed as words. This is not a problem since these words will simply be marked as not replaceable. It is important to have this simple definition of what a word is because section size is defined in number of words. To maintain the purity of the steganographic system, it must be assumed that the message receiver does not have access to some word table that can verify if something is a real word. This meant cutting out some parts of the synonym table, which originally included some numerical values and some hyphenated words. **10.465%** percent of entries in the synonym table were removed to avoid changes in the perceived number of words between the covertext and the stegotext.

Words can appear with an uppercase or lowercase leading character. It is desirable that this is maintained for the replacements of a word. For this reason, words are parsed into a data structure that stores this information.

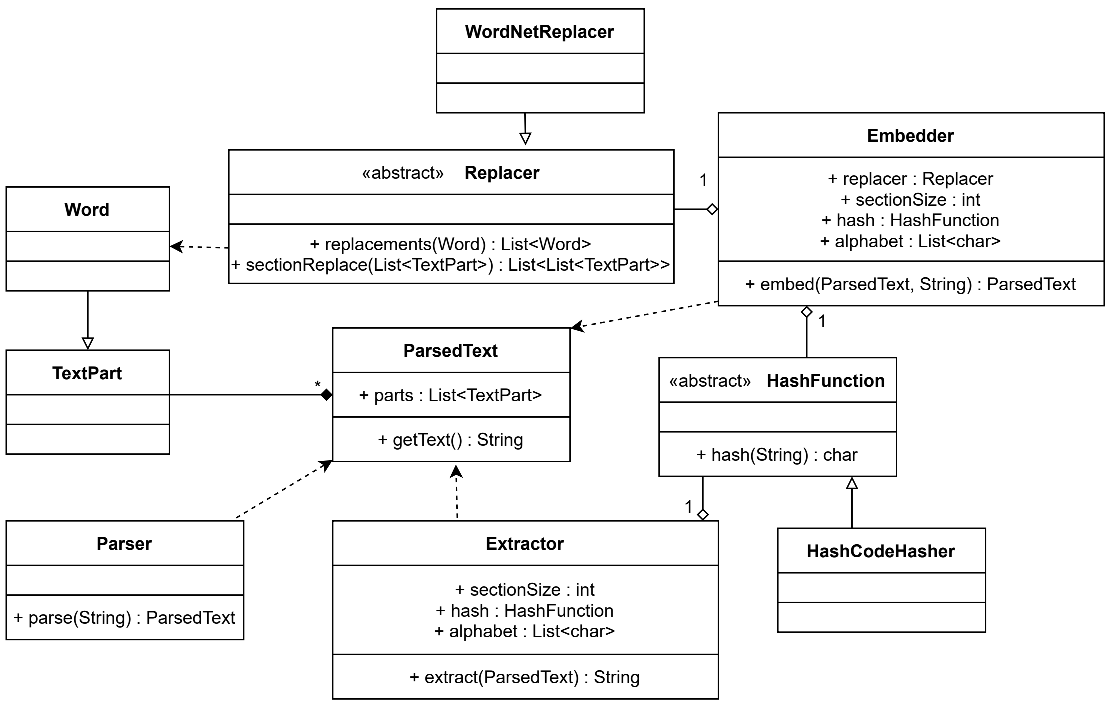

Figure 5.5: Simplified UML class diagram of the described implementation of the system.

5.3.3 Synonym Table Usage

The synonym table described in Section 5.2 is exported as a text file to be imported by the embedding program. For each line of the synonym table, a pair word-replacements is created. For efficiency, these are stored in a hash map, which allows for the replacement set of a word to be obtained in constant time.

To compute the set of replacements for a section, the replacements for each word in this section are firstly identified. Replacements for the section will come from the combinations of the different replacements for each word.

Given that the embedding process will only need one replacement of the section that hashes to the desired character, and that the number of replacements might be orders of magnitude more than needed. The system does not immediately compute all alternatives for a section, instead it returns them one at a time until a usable one is found. This led to a significant improvement in computation speed and memory usage.

5.3.4 Section Hashing

For the purpose of hashing, our implementation of the described steganography system used the immediately available Java string ”hashcode” function. This hashing function computes, for the string as a sequence of integer ascii values $S = s_1, s_2, ...s_n$, the formula $H(S) = \sum_i s_i(31^{n-i})$. The resulting value is used to map a character in the hidden message alphabet Σ by computing $H(S) \mod |\Sigma|$ and using it as an index for a position in the alphabet.

5.3.5 Public Availability

Our implementation of the steganography system, as described in this section, is made available, in working condition, on a Github repository [46].

5.4 Probability Estimation Functions

In Section 4.5.1, formulas are provided to predict the embedding probability of the described steganography system, based on the various parameters that are used. To validate these formulas, they were implemented and are provided alongside the Java implementation of the project.

The formula in Corollary 4.5.4 proved the most challenging. Firstly, all E sets need to be computed for the number of replacements r. These sets represent all the ways that r can be expressed as a multiplication of positive integers. To compute these sets, we first decompose r into the set of its prime factors, which is our first E set. All other E sets are the possible combinations of these primes. For example, for the number 12, we compute the prime set $\{2, 2, 3\}$, and by combining these primes we get the remaining E sets $\{4, 3\}$, $\{2, 6\}$, and $\{12\}$. Our implementation for this algorithm has $O(p^p)$ time complexity, where p is the number of primes in the primal decomposition of r. Due to this very fast growing time complexity, our computations of this algorithm stopped before $r = 512$, which is the first natural number with 9 primes.

The last formula in the Corollary also showed some challenges. The various factorials in this formula often causes value overflows. To solve this, our implementation computed all the factorials in parallel. If that part of the formula is computed as

$$\frac{w!}{e_{i,1}!e_{i,2}!...e_{i,k}!} = \frac{w}{e_{i,1}e_{i,2}...e_{i,k}} \frac{(w-1)!}{(e_{i,1}-1)!(e_{i,2}-1)!...(e_{i,k}-1)!}$$

and recursion is used, then there will not be a value overflow. The shown formula is simplified in that these values cannot be decremented bellow 1.

6 Evaluation

Contents

6.1 Replacement Table 51
6.2 Steganography System 54
6.3 Hidden Message Alphabet 59

Table 6.1: Example of a paragraph being rewritten according to the synonym table constructed as described in Section 5.2. This paragraph is a sample of the book ”To Kill a Mockingbird”, by Harper Lee, 1960. Replaced words are highlighted.

Original	Rewritten
The Radleys, welcome anywhere in town, kept to themselves, a predilection unforgivable in Maycomb. They did not go to church, Maycomb's principal recreation, but worshiped at home; Mrs. Radley seldom if ever crossed the street (...)	The Radleys, welcome anywhere in town, kept to themselves, a preference inexcusable in Maycomb. They did not go to church, Maycomb's principal recreation, but worshiped at home; Mrs. Radley rarely if ever crossed the street (...)

6.1 Replacement Table

One of the main contributions of this thesis was the synonym table constructed as described in Section 5.2, this table was constructed for usage with the described system but can be also be used in the context of many other systems for semantic steganography.

The utility of a synonym table is dictated by how frequently it can find a replaceable word, how many replacements can it find, and how natural are the replacements.

6.1.1 Synonym Quality

Given the way that the synonym table was constructed, it is ensured that it will never replace a word with another that would not be a fit for that context. This has the shortfall that the synonym table is somewhat restricted and will find fewer replaceable words than if this was not verified.

The WordNet dataset only includes nouns, verbs, adjectives, and adverbs, which are the types of words that can usually be replaced with synonyms. It does not include, however, more ”grammar-like” words such as pronouns or prepositions. This is usually not a problem since these words tend to not be replaceable in a text. But the omission of these words caused some problems when homographs appeared as other word types. For example, the pronoun ”what” is not in WordNet and should not be replaceable, but this word is a homograph to the past tense of the verb ”to wham”, meaning ”to hit hard”. As such, the system would often replace ”what” with past tenses of synonyms of ”to wham”, such as ”whacked”. To solve this, situations like these were identified and special synsets were added. Words like ”what” were added into their own synsets of only one word. This guarantees that the system will never identify them as replaceable.

An example of the alterations provided by the synonym table is provided in Table 6.1, all words identified as replaceable are highlighted and replaced.

Table 6.2: Probabilities of a word having n replacements, calculated over a 3 million word subset of the COHA corpus [9] and a million word subset of the GloWbE corpus [13].

Replacements	Probability
not replaceable	96.260%
2	2.040%
3	0.722%
4	0.488%
5	0.186%
6	0.125%
7	0.101%
8	0.044%
9	0.012%
10	0.010%
11	0.007%
12	0.002%
13	0.002%
14 or more	0.001%

6.1.2 Synonym Probabilities

Given that different words appear in text with different frequencies, the size of the synonym table is not sufficient to determine the frequency at which it finds replaceable words. The frequencies for each word need to be accounted for. As such, an appropriate way to extract some first order statistics of the synonym table is to randomly sample words from candidate covertexts and to count the number of replacements found for each. To do this we used a 3 million word subset of the COHA corpus [9], along with a 2 million word subset of the GloWbE corpus [13]. The Corpus of Historical American English (COHA) contains a wide variety of published English text from the 1810s-2000s, including magazines, newspapers and fiction and non-fiction books. The corpus of Global Web-based English (GloWbE) contains the plaintext contents of various English-language international websites. Together, these two corpora provide a very wide and unfocused sample of the English language. Table 6.2 shows the probabilities of words being replaceable with a certain number of replacements.

As is shown, the replacement table can find replacements for about **3.74%** of words randomly sampled from English language texts. Meaning that, for a random covertext, the system will find replacements for slightly more than 1 in every 30 words. These results fell short of the expected, as they imply that, for a steganographic system, there are not many degrees of freedom to perform changes in the text, and the resulting embedding rate should be quite low.

The replacement probability was also independently computed for each of the document types within the dataset, to study if the nature of document had any implication regarding the frequency of replaceable words. These values are shown in Table 6.3.

Due to the fact that the COHA corpus [9] is focused on the historical development of the English language, documents within this dataset span a wide range of publication dates are are accordingly

Table 6.3: Probabilities of a word being replaceable, for the various document types within a 3 million word subset of the COHA corpus [9] and a million word subset of the GloWbE corpus [13].

Document Type	Replacement Probability
overall	3.740%
fiction books	3.383%
magazines	3.959%
newspapers	4.209%
non-fiction books	3.931%
websites	3.846%

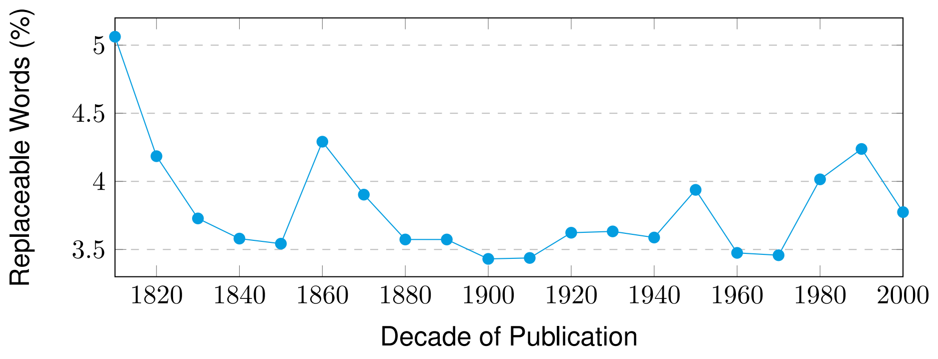

Figure 6.1: Probability of a word being replaceable, by decade of document publication, from a 3 million word subset of the COHA corpus [9].

labeled. We used this to study a possible correlation between the age of the document and the frequency of replaceable words. The results are plotted in Figure 6.1.

From the shown results, the variation of frequency of replaceable words, between decade of publication, or document type, does not seem substantial enough to allow for any confident statement regarding correlations between these factors.

6.1.3 Independence of Word Replaceability

In corollary 4.5.2 we stated that a multinomial distribution can be used to explain the distribution in the number of replacements of a section, if we assume that the number of alternatives for words in a section are independent from other words. To defend this statement, we made the simple experiment of computing the probability of a word being replaceable if the previous word is known to be replaceable, and comparing it to the overall probability of a word being replaceable. If the number of alternatives for words are perfectly independent, then it is expected that

$$\mathbf{P}(w_i \text{ is replaceable}) \approx \mathbf{P}(w_i \text{ is replaceable} \mid w_{i-1} \text{ is replaceable}).$$

Using the same dataset as described in Section 6.1.2, we counted occurrences of replaceable words, and occurrences of consecutive replaceable words. From our measurements it resulted that each word has a **3.740%** probability of being replaceable, and that a word that comes after a replaceable word has a **3.836%** probability of being replaceable itself. These values are very similar and show that the replaceability of a word is very independent from the replaceability of words in its immediate neighbourhood.

6.2 Steganography System

To evaluate our system, we first surveyed the metrics used for evaluation of semantic steganography systems.

In [16], Kaufmann dedicates a section to the evaluation of steganographic systems. In their articles, many authors of the systems listed in Section 3 described the methods that were used to validate their approaches.

In this section we go over these metrics and use them to evaluate our system.

6.2.1 Embedding Effectiveness

Embedding effectiveness relates to the probability that the embedding function fails or alters the hidden message [16].

In Section 4.5 we go over predicting the embedding probability for cover messages. The provided formulas can be validated by sampling cover messages that could be used in real-life applications and testing them for the embedding of random hidden messages.

6.2.1.A Distribution of Section Replacement Cardinality

In Corollary 4.5.4, a formula is deduced to compute the probability mass function for the number of replacements that a section is expected to have. To validate this formula, the computed expected probability is compared to the relative frequencies of numbers of section replacements, as sampled from the dataset described in Section 6.1.2. These results are plotted out in Figure 6.2.

The predicted probabilities are a very close fit to the measured values, which validates the provided formula. A more staggering difference between the projected and measured values is seen for the number of replacements ”1”, that is, sections that are not replaceable. With the formula predicting a **2.210%** frequency for these sections, but a real occurrence rate over the dataset of **4.358%**. We estimate that this discrepancy is caused, in part, by the not perfect independence of word replacements, but mainly for the existence of ”noisier” text within the dataset. If there are parts of the dataset with non natural language text, or text in languages other than English, then, these substrings will not have replaceable words within them. This causes clusters of non-replaceable words that can span multiple

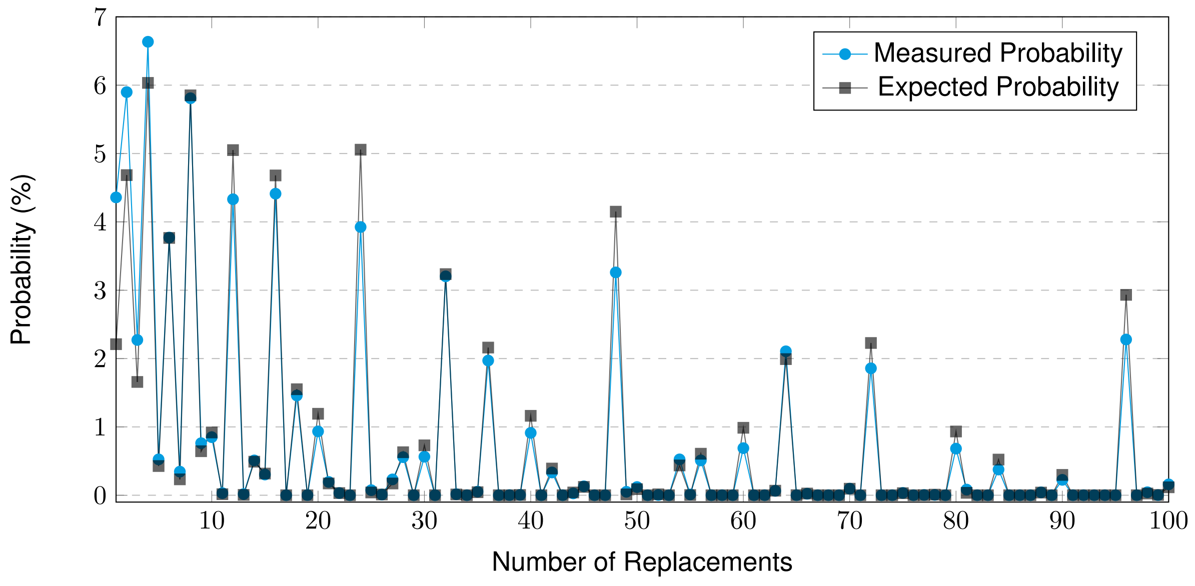

Figure 6.2: $RP(w, r)$, Probability distribution for number of replacements up to 100, on sections of $w = 100$ words. The probability mass function described in Corollary 4.5.4 is compared to the results obtained from sampling the dataset described in Section 6.1.2. The replacement table described in Section 5.2 is used for the sampling, and the corresponding first order statistics in Table 6.2 are used as parameters for the estimation formula.

sections, each of with will have no replacements, increasing the probability of non-replaceable sections, as measured.

6.2.1.B Message Embedding Probability

The formula described in Lemma 4.5.2 provides a lower bound for the expected value of the embedding probability (or effectiveness) of our system. This formula (and the tightness of its bound) can be validated by sampling candidate cover messages and comparing the measured embedding probability to the lower bound given by the formula. These values were computed and are plotted out in Figure 6.3.

The effect of section size on the embedding probability of messages can also be studied using the described setup. The measured values for embedding probability are compared to the values predicted by Lemma 4.5.2 and Corollary 4.5.4 as plotted in Figure 6.4.

The results shown in Figure 6.3 show that the provided formula offers a very close lower bound to the real probability values. This allows for this formula to be used as a fully analytical tool for the negotiation of parameters in a system like this. If the first order properties of the synonym table are known, this formula would be sufficient to compute whether a set of parameters allows for reliability in the system and sufficient embedding effectiveness, without the need to run any simulations. As is immediately intuitive but clarified in this plot, longer messages have lower embedding probability due to requiring embedding successes over more sections.

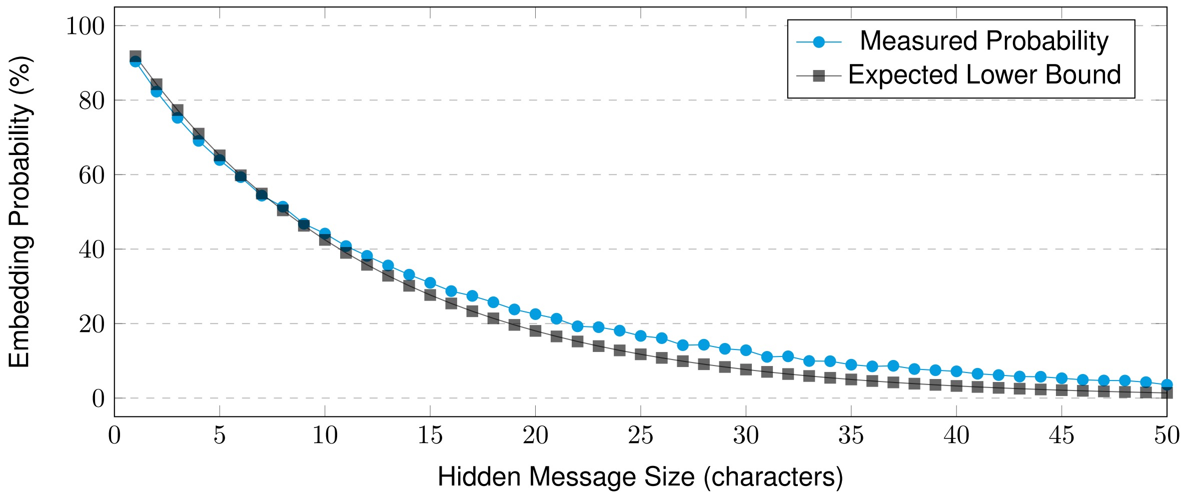

Figure 6.3: Probability that a hidden message of size n is embeddable into some cover message. Using sections of 200 words, and assuming a hidden message alphabet of size 29 (English alphabet with 2 punctuation marks and the space character). The ”measured probability” was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The ”expected lower bound” was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table.

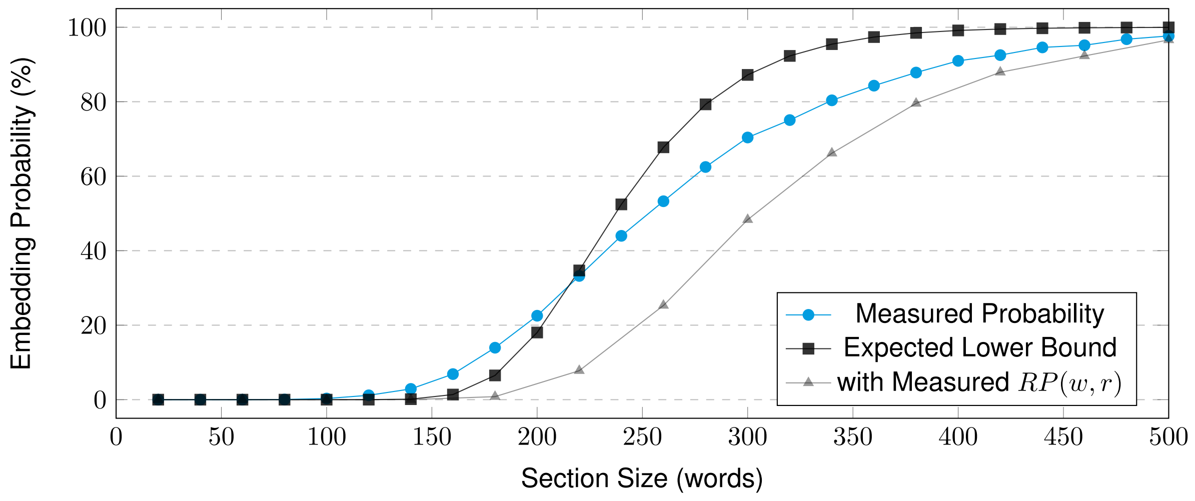

Figure 6.4: Probability that a hidden message of 20 characters is embeddable into some cover message. Using sections of w words, and assuming a hidden message alphabet of size 29. The ”measured probability” was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The ”expected lower bound” was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table. Lemma 4.5.2 is plotted again, using the real $RP(w, r)$ distribution as measured and shown in Figure 6.2.

In Figure 6.4 it is shown that, while the predicted values do closely follow the measured probabilities, the expected lower bound can take values that are above the real probabilities. This can be explained by the fact that the formula in Corollary 4.5.4 predicted a lower frequency for sections with no replaceable words, and that, as was explained in Section 6.2.1.A, since the number of replacements of words is not fully independent of neighboring words, there are some clusters with no replaceable words in the dataset. In this Figure it is shown that, if instead of the values predicted by Corollary 4.5.4, the real measurements for the distribution of section replacements are used, then, as is demonstrated in Lemma 4.5.2, the formula will provide a true lower bound for the embedding probability.

Another important take-away from the plot in Figure 6.4 is the correlation between section size and embedding probability. As is clear with the nature of the system, longer section sizes will have higher probability of having more replacements, and provide greater embedding effectiveness.

6.2.2 Embedding Capacity

The embedding capacity is a very relevant parameter on the selection of steganographic systems. It relates to the amount of information that can be hidden on a certain cover message. For text steganography, this is calculated as the size of the hidden message divided by the size of the cover message (given that both messages are written in, or converted to, the same alphabet). For the provided system, since we embed one character of the hidden message in each fixed size section, we get that the embedding capacity is $\frac{1}{s}$, where s is the average size, in characters, of sections. From the described dataset, we measured an average value of **6.215** characters per word (this includes non-word characters). We compare the effect of embedding rate on embedding probability with the results exposed in Figure 6.5.

As is shown, greater values for embedding probability can be obtained by sacrificing the embedding rate, this is done by increasing the section size. The values plotted here show the embedding probability for 1-character messages. For these values, the formula in Lemma 4.5.2 provides a very loose lower bound that gets tighter for longer hidden message sizes, such as the one used in Figure 6.3.

6.2.3 Statistical Undetectability

Third parties that might be suspicious of a cover message can use certain methods to try to detect the presence of a hidden message. These methods, that usually analyse the statistical properties of the message, are called steganalysis [15, 16].

Statistical anomalies in the distribution of words are used to detect the presence of semantic steganography systems. It is therefore ideal to evaluate and minimize this “byproduct” of the proposed system. In [5], the authors propose the evaluation of *perplexity* as the ideal benchmark for this. *Perplexity* is a measurement of how well a probability model predicts a sample and is a standard metric in natural

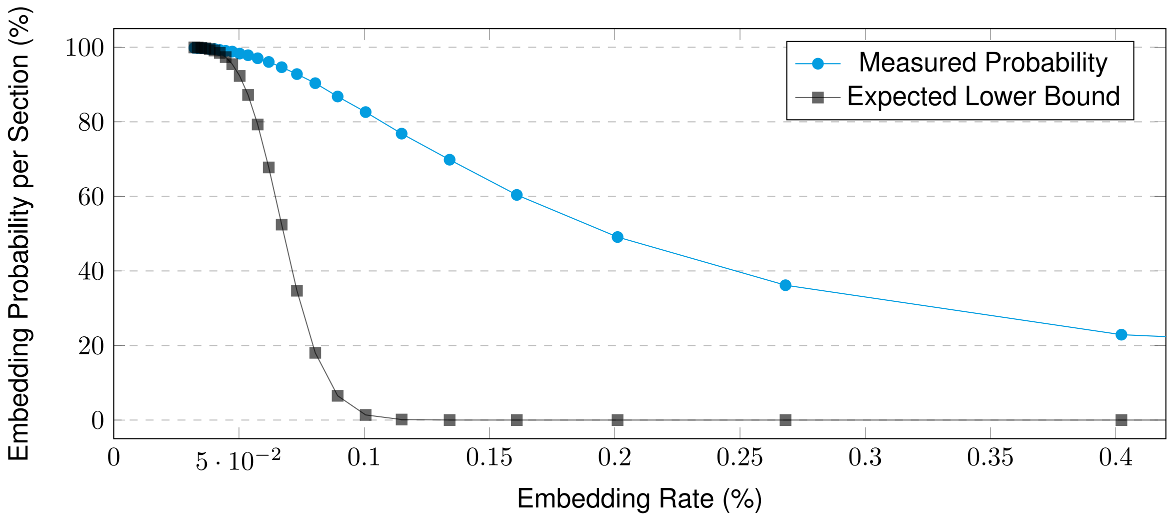

Figure 6.5: Embedding probability for one section, depending on embedding rate, and assuming a hidden message alphabet of size 29. The "measured probability" was measured from candidate cover messages that were sampled from the the dataset described in Section 6.1.2, and had possible replacements computed with the replacement table described in Section 5.2. The "expected lower bound" was computed using the formula described in Lemma 4.5.2, with the replacement distribution $RP(w, r)$ computed with the probability mass function described in Corollary 4.5.4 that uses the first order statistics of the same replacement table. Figure 6.2.

language processing for assessing the quality of sentences. For a fair comparison of systems using perplexity, the embedding rates of systems are matched and the perplexities are compared for a given probabilistic language model.

Given the nature of the described system, and the fact that the synonym table is not shared so as to provide a system of pure steganography, it is not directly competing with systems that do not verify this. Additionally, the range for embedding rates is very low compared to other systems that presented these values [3–5, 41] and it is not possible to match it. Further, given the fact that the stegotexts produced by this system will differ from the corresponding covertext in only at most **3.740%** of words, and that the replaced words are guaranteed to be a fit for the context, as described in Section 5.2, then the perplexity measured by any model and stegotext should be virtually identical to that of the covertext. As such, perplexity is not a viable metric for the described system.

6.2.4 Robustness

The robustness of a steganography system relates to how resistant the hidden message is to having the cover message go through simple transformations such as lossy compression systems [17]. A big advantage with semantic steganography systems is that, to damage the hidden message, it is necessary to replace or remove words. For this reason, semantic steganography systems can be regarded as having great robustness.

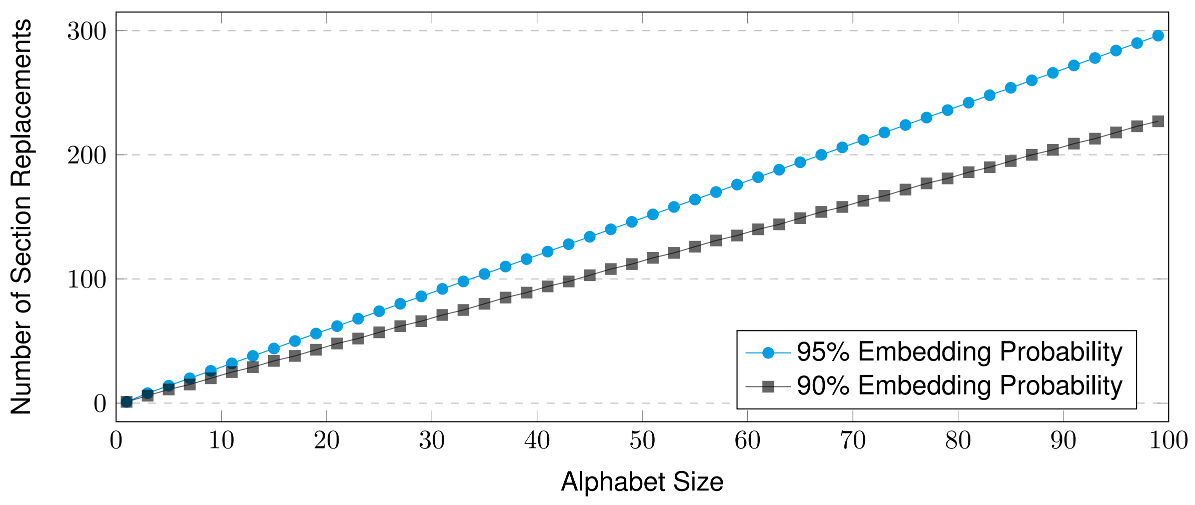

Figure 6.6: Number of replacements needed for a section to ensure 90% and 95% probability of embedding a hidden message character, depending on alphabet size.

6.3 Hidden Message Alphabet

Using small alphabets is important due to the scarcity in space for the hidden message. Larger alphabets require more replacements per section to ensure that the desired character is found on one of them. In Figure 6.6 we show this correlation between alphabet size and the number of replacements needed to ensure a certain embedding probability. These results can also be obtained by solving the formula in Lemma 4.5.1 for the number of replacements r, as follows.

$$\begin{aligned}
1-(1-\frac{1}{|\Sigma|})^r &\geq p \\
-(1-\frac{1}{|\Sigma|})^r &\geq p-1 \\
(1-\frac{1}{|\Sigma|})^r &\leq 1-p \\
r\ln(1-\frac{1}{|\Sigma|}) &\leq \ln(1-p) \\
r &\geq \frac{\ln(1-p)}{\ln(1-\frac{1}{|\Sigma|})}.
\end{aligned}$$

6.3.1 Alphabet Encoding

For the previous simulations, a hidden alphabet of size 29 was chosen. This is sufficient to encode the English alphabet, a space character, and two punctuation marks. Alphabets like these can be converted into smaller alphabets. For example, if the binary alphabet $\Sigma = \{0,1\}$ is used, then characters in alphabets of up to 32 characters can be encoded into 5 bits each. In doing this, sections can be shorter

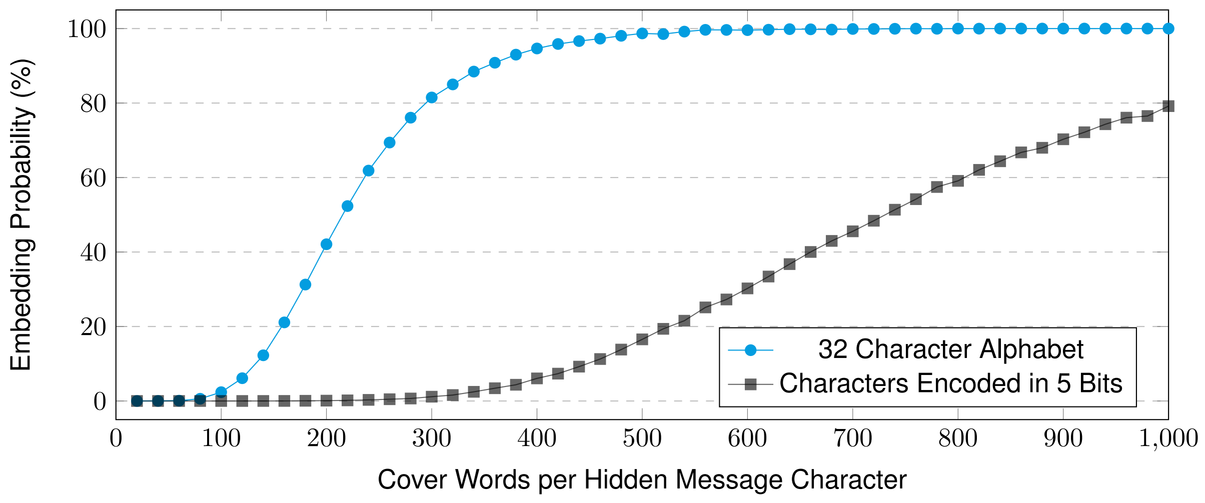

Figure 6.7: Comparison between usage of a 32 character alphabet, and that same alphabet encoded into 5 bits of a binary alphabet, measured in embedding probability. The horizontal axis shows size of each section for the 32 character alphabet, or the size of 5 sections for the binary alphabet. Results were computed for a 10 character message, or 50 bits, after conversion, by sampling candidate covertexts from the dataset described in Section 6.1.2.

while still ensuring high embedding probabilities per section, but 5 times as many sections are needed. This trade-off is compared in Figure 6.7.

As is shown by the plotted results, converting alphabets into multiple characters of smaller alphabets does not improve embedding probability. For cover messages of equal size, if the hidden message alphabet is mapped into n characters of a smaller alphabet, then each section must be divided into n parts. The conversion might increase the embedding probability per section, but the fact that there are n times as many sections results in a diminished embedding probability for the whole message.

6.3.2 Alphabet Multiplication

The inverse operation was also studied. Multiple characters of an alphabet can be grouped to construct a larger alphabet. The alphabet $\Sigma^1 = \{"a", "b", ...\}$ can be converted to the alphabet $\Sigma^2 = \{"aa", "ab", ..., "ba", ...\}$, with $|\Sigma^2| = |\Sigma^1|^2$ by grouping every pair of characters in Σ^1 into a character for Σ^2. This is a Cartesian multiplication of Σ^1 and itself. Larger alphabets Σ^n can be constructed by grouping n characters of Σ^1 and will verify $|\Sigma^n| = |\Sigma^1|^n$. The set Σ^n can be called the n-ary Cartesian power of Σ^1.

If a message in Σ^{1*} has length s and is being embedded into a cover message with s sections of w words each, then, using the same cover message, the hidden message converted to Σ^{n*} will have length s/n will be embedded into s/n sections of $w \times n$ words each. A simulation of this setup is shown in Figure 6.8.

As is shown, there is substantial increase to the embedding probability of a message when the

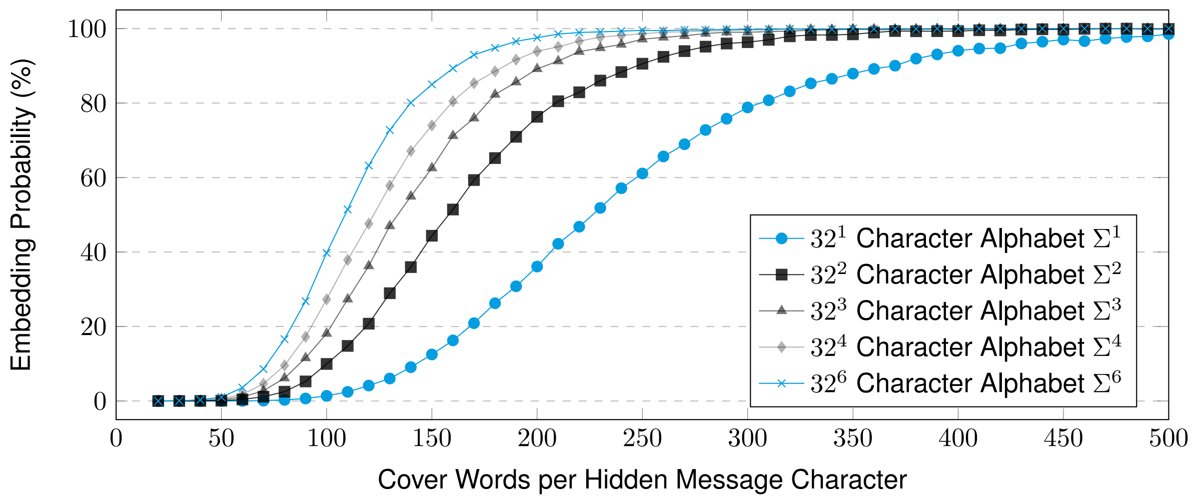

Figure 6.8: Comparison between usage of a 32 character alphabet Σ^1, and longer alphabets Σ^n created by the grouping of n characters of Σ^1. Results were computed for a 12 character message in Σ^{1*}, converted to a 6, 4, 3, and 2 character message using Σ^2, Σ^3, Σ^4, and Σ^6, respectively. Simulations were ran by sampling candidate covertexts from the dataset described in Section 6.1.2. The horizontal axis measures covertext words per character of the hidden message after being converted to Σ^1.

alphabet is expanded to Σ^n as described, with the difference being less significant with each increment of n. This increase in embedding probability has a significant trade-off in computation complexity. Given that the size of the alphabet Σ^n grows exponentially with n, the number of replacements that needs to be tested for each section will also grow exponentially. This implies an exponential increase in string hashings. For example, in the results plotted in Figure 6.8, the used alphabet Σ^6 has size 32^6, which implies that, on average, 1.073×10^9 section replacements will have to be checked per section, which can take up a substantial amount of computation time on most implementations.

7

Final Remarks

Contents

7.1 Significant Contributions 65
7.2 Comments on the System 66
7.3 Future Improvements 66
7.4 Conclusion 68

Throughout the body of this thesis we have provided a deep investigation and review of the core concepts of semantic steganography and of steganography in general. This included a thorough survey of published systems for semantic steganography. In doing so, we laid the groundwork to describe how a system for semantic steganography could be a pure steganographic system, and put forth the first description of one such system. We provided a rigorous analytical analysis of the statistical properties of our a system as a Monte-Carlo algorithm. And used this analysis to evaluate our own implementation of the described algorithm.

7.1 Significant Contributions

Despite being an ancient area of research, steganography is not often a topic of research in modern times. Further, many publications on the area offer inconsistent and occasionally outright contradictory statements. In reviewing the background for this thesis, we have provided structured information and definitions that help conciliate a lot of what has been published regarding steganography, and specifically about text steganography. This includes a complete hierarchy of the areas of text steganography that was not found in entirety on a previously published article.

In preparing to develop our system, we surveyed existing publications for semantic steganography systems. This survey provides the most thorough listing of different approaches to semantic steganography and is certain to be useful in bench-marking future applications. In explaining the mentioned systems, we have created detailed diagrams that help expose and simplify their function and were not present in the corresponding original publications.

Our approach for a steganographic system as shown in Section 4 is the first approach for a pure semantic steganography system, as such, it can be used between a sender and receiver without the prior sharing of a synonym table or a language model. This is in contrast to previous approaches for semantic steganography. The resulting system, as evaluated in Section 6, has a very low embedding rate, which implies that the chosen cover messages will have to be very long text artifacts. This severely limits the applicability of our system, however, given that this is the first such system, there can be situations where a pure steganographic system is the only viable option. Beyond its applicability, we hope that our system can serve as a starting point and benchmark for future semantic pure steganographic systems.

For usage with our system, we developed a synonym table to identify replaceable words in an English text and list possible replacements for these words. This synonym table can be applied to many steganographic systems beyond our own, including some listed and described in our survey in Section 3. Because of this, it was made publicly available as an independent part of our project.

7.2 Comments on the System

In this document, we described the function of our approach for a system of semantic steganography in Section 4. This system was implemented as described in Section 5.

In our evaluation of this system, whose results are available in Section 6, we found that there are some properties to it that might limit its applicability in real life situations.

To ensure a reliable probability of embedding success, we found that cover messages would have to be divided into sections of at least 200 to 300 words. Each of these sections can encode a single character of the hidden message. This implies that cover messages for usage as input to our implementation of the system have to be extremely long documents. The most viable option for these, if the cover messages are not written from scratch, is that documents like books are used. A security risk with the usage of these cover messages is that, if the original document is publicly available, a third party might find compare the original document with the resulting covertext and identify that it has been tampered with, nullifying the point of using steganography in the first place.

The main cause for the low embedding rate are the low degrees of freedom for modifying the covertext. These degrees of freedom are dictated from the possible replacements of words as provided by the replacement table. Our synonym table has the low rate of only finding replacements for **3.74%** of words randomly sampled from English texts. Greater rates of replacements would result in greater embedding rates for the system, but a different approach to construct a synonym table might be necessary.

7.3 Future Improvements

As the first system for semantic pure steganography, our proposed approach has room for further developments in eventual future research. Near the end of the development of our system, some possible improvements were proposed that could not be fit into the development schedule of this project and are now exposed here.

7.3.1 Context Aware Replacement Table

To construct our synonym table, we assume that the specific meaning being taken by a word is not known. As such, the possible replacements for a word are defined as words that are synonyms to it in any given context. This means discarding many possible replacements for words by intersecting all synsets in which it appears, this is first mentioned in Section 3.1.4 and further explained in Section 5.2.

A significant improvement in the construction of the synonym table would come from building a system that could identify the meaning being taken on by a word and identify the correct synset from which to get the possible synonyms. This would largely increase the number of possible replacements per word and the number of replaceable words, which would, in turn, result in a system with greater embedding rate.

Possibilities for implementing this modification include analysing semantic relations between the synsets of a word and the other words in the sentence, or using systems for sentence quality evaluation [47] to choose the synset that provides better quality sentences.

7.3.2 Deep Learning Applications for Text Rewriting

The usage of deep learning for natural language processing is an area of research that is very quickly growing and proving ever more outstanding results for what computers can do with man-made writing artifacts.

Given these recent developments, there are now systems that can rewrite sentences with different wording. One such system is the one developed by Xu [48] which uses the BERT deep learning model [49]. A system like this one could be used to replace the synonym table altogether. The system would list the possible rewritings of a sentence and and the embeder would perform the normal operation of selecting a replacement that hashes to the desired hidden message character.

7.3.3 Replacement Recycling

From the evaluation of our system, we saw that the number of replacements for a section could diverge by many orders of magnitude. This resulted in that the overall embedding probability would effectively be dictated by sections with fewer replacements, and that sections with many replacements were just a guaranteed embedding, regardless of how many millions of replacements they could have. Our system could have greater embedding rate if it could utilize the extra degrees of freedom given by these sections with many replacements.

If the hashing stage of our system would also take in information from previous sections, then the past choices of replacements would have an effect on the result of other hashings. This way, if there is an embedding failure on a section with fewer replacements, then the system could backtrack to a section with multiple choices for viable replacements and change the selection. This would effectively perform a tree search algorithm over the replacements of sections. This modification to our system could greatly improve the embedding rate. We expect that, if this was implemented, sections would only need to have, on average, as many replacements as characters in the hidden message alphabet.

7.4 Conclusion

With this thesis we have provided the groundwork for constructing a system for semantic pure steganography, and have implemented it and thoroughly analysed its properties. In doing this, we have made multiple contributions to the field of steganography and more specifically semantic and text steganography. These contributions go beyond our implementation of the system itself.

Despite the downfalls of our system, we see that, as a whole, it is an important step in the development of future systems for semantic pure steganography, and that, with further developments such as the ones described in Section 7.3, it can become a strong tool for truly innocuous communication.

Bibliography

[1] J. Stanley, “Image steganography,” https://incoherency.co.uk/image-steganography/, accessed: 12/2020.

[2] K. Winstein, “Lexical steganography through adaptive modulation of the word choice hash,” 1998.

[3] W. Dai, Y. Yu, and B. Deng, “Bintext steganography based on markov state transferring probability,” 11 2009, pp. 1306–1311.

[4] H. Moraldo, “An approach for text steganography based on markov chains,” 09 2014.

[5] Z. Yang, S. Jin, Y. Huang, Y. Zhang, and H. Li, “Automatically generate steganographic text based on markov model and huffman coding,” 2018.

[6] G. A. Miller, “Wordnet: A lexical database for english.” *Communications of the ACM*, vol. 38, 1995.

[7] C. Fellbaum, “Wordnet: An electronic lexical database.” 1998.

[8] J. R. Coombs, “Inflect,” https://github.com/jaraco/inflect.

[9] M. Davies, “Corpus of historical american english (coha),” https://www.english-corpora.org/coha/, 2010.

[10] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, “Techniques for data hiding,” *IBM Systems Journal*, vol. 35, no. 3.4, pp. 313–336, 1996.

[11] M. H. Shirali-Shahreza and M. Shirali-Shahreza, “A new synonym text steganography,” in *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008, pp. 1524–1526.

[12] M. Shirali-Shahreza and M. H. Shirali-Shahreza, “Text steganography in sms,” in *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, 2007, pp. 2260–2265.

[13] M. Davies, “Corpus of global web-based english (glowbe),” https://www.english-corpora.org/glowbe/, 2013.

[14] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*. Wiley, 2010.

[15] S. Katzenbeisser and F. A. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st ed. USA: Artech House, Inc., 2000.

[16] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[17] S. Kingslin and N. Kavitha, "Evaluative approach towards text steganographic techniques," *Indian Journal of Science and Technology*, vol. 8, 11 2015.

[18] S. Wintner, *Formal Language Theory for Natural Language Processing*. Wiley, 2001.

[19] M. G. K. F. A. P. Petitcolas, R. J. Anderson, "Information hiding - a survey," *Proceedings of the IEEE*, vol. 87, 1999.

[20] S. Roy and M. Manasmita, "A novel approach to format based text steganography," 01 2011, pp. 511–516.

[21] M. Agarwal, "Text steganographic approaches: A comparison," *International Journal of Network Security and Its Applications*, vol. 5, 02 2013.

[22] Z. Doffman, "U.s. may outlaw messaging encryption used by whatsapp, imessage and others, report," https://www.forbes.com/sites/zakdoffman/2019/06/29/u-s-may-outlaw-uncrackable-end-to-end-encrypted-messaging-report-claims/, accessed: 12/2020.

[23] V. Verma and R. Chawla, "Image block-based steganography using varying size approach," *International Journal of Computer Applications*, vol. 97, pp. 58–61, 2014.

[24] J. Fingas, "China passes law regulating data encryption," https://www.engadget.com/2019/10/27/china-cryptography-law/, accessed: 12/2020.

[25] D. Cooper, "China's new cybersecurity laws are a nightmare," https://www.engadget.com/2016-11-07-china-s-new-cybersecurity-laws-are-a-nightmare.html, accessed: 12/2020.

[26] S. H. Abdullah, "Steganography methods and some application(the hidden secret data in image)," 2009.

[27] B. A. Sheelu, "An overview of steganography," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 11, 2013.

[28] B. Osman, A. Yasin, and M. Omar, "An analysis of alphabet-based techniques in text steganography," vol. 8, pp. 109–115, 01 2016.

[29] H. Singh, "Analysis of different types of steganography," *International journal of scientific research in science, engineering and technology*, vol. 2, pp. 578–582, 2016.

[30] M. Nosrati, R. Karimi, and M. Hariri, "An introduction to steganography methods," *World Applied Programming*, vol. 1, pp. 191–195, 08 2011.

[31] J. J. M. Gunjal, "Image steganography using discrete cosine transform (dct) and blowfish algorithm," *International Journal of Computer Trends and Technology*, vol. 11, 2014.

[32] D. Babya, J. Thomasa, G. Augustinea, E. Georgea, and N. R. Michael, "A novel dwt based image securing method using steganography," *International Conference on Information and Communication Technologies*, 2015.

[33] S. Sharma, A. Gupta, M. C. Trivedi, and V. K. Yadav, "Analysis of different text steganography techniques: A survey," in *2016 Second International Conference on Computational Intelligence Communication Technology (CICT)*, 2016, pp. 130–133.

[34] P. Wayner, "Mimic functions," *Cryptologia*, vol. 16, no. 3, pp. 193–214, 1992.

[35] ——, "Strong theoretical stegnography," *Cryptologia*, vol. 19, no. 3, pp. 285–299, 1995.

[36] K. F. Rafat, "Enhanced text steganography in sms," in *2009 2nd International Conference on Computer, Control and Communication*, 2009, pp. 1–6.

[37] M. Chapman, G. I. Davida, and M. Rennhard, "A practical and effective approach to large-scale automated linguistic steganography," in *Information Security*, G. I. Davida and Y. Frankel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 156–165.

[38] G. Ward, "Moby project thesaurus," http://moby-thesaurus.org/, accessed: 12/2020.

[39] H. Huanhuan, Z. Xin, Z. Weiming, and Y. Nenghai, "Adaptive text steganography by exploring statistical and linguistical distortion," in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, 2017, pp. 145–150.

[40] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[41] W. Dai, Y. Yu, Y. Dai, and B. Deng, "Text steganography system using markov chain source model and des algorithm," *JSW*, vol. 5, pp. 785–792, 07 2010.

[42] P. R. R. Motwani, *Randomized Algorithms*. Cambridge University Press, 2000.

[43] R. McEliecen, *The Theory of Information and Coding*. Cambridge University Press, 1977.

[44] S. Diao, “Mlconjug,” https://github.com/SekouD/mlconjug.

[45] J. Figueira, “A wordnet replacement table,” https://github.com/joaoperfig/semsteg/tree/main/wordnet_parser.

[46] ——, “A novel system for semantic steganography,” https://github.com/joaoperfig/semsteg.

[47] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, “A Survey on Text Classification: From Shallow to Deep Learning,” *arXiv e-prints*, p. arXiv:2008.00364, Aug. 2020.

[48] L. Xu, I. Ramirez, and K. Veeramachaneni, “Rewriting Meaningful Sentences via Conditional BERT Sampling and an application on fooling text classifiers,” *arXiv e-prints*, p. arXiv:2010.11869, Oct. 2020.

[49] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv e-prints*, p. arXiv:1810.04805, Oct. 2018.

A

Usage Example

The following pages are an example of an output stegotext message produced by our system.

The hidden message ”secret” was embedded into the first two pages of the book ”1984”, published by George Orwell in 1949. The hidden alphabet was the 29 character alphabet comprised of the English alphabet, a space character, and two punctuation marks. A section size of 130 words was used.

All replaceable words are highlighted. Words that were altered from the original text are shown in light blue, and words that are replaceable but were not altered are shown in dark blue.

Nineteen eighty-four

George Orwell

PART ONE - Chapter 1

It was a bright cold day in Apr, and the clocks were striking xiii. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

The hallway smelt of boiled cabbage and old rag mats. At one end of it a coloured poster, too large for indoor display, had been tacked to the wall. It depicted simply an enormous face, more than a metre wide: the face of a man of about forty-five, with a heavy black moustache and ruggedly handsome features. Winston made for the stairs. It was no use trying the lift. Even at the best of times it was seldom working, and at present the electric current was cut off during daylight hours. It was part of the economy drive in preparation for Hate Week. The flat was seven flights up, and Winston, who was xxx-nine and had a varicose ulcer above his right ankle, went slowly, resting several times on the way. On each landing, opposite the lift-shaft, the poster with the enormous face stared from the wall. It was one of those pictures which are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption below it ran.

Inside the flat a fruity voice was reading out a list of figures which had something to do with the production of pig-iron. The voice came from an oblong metal plaque like a dulled mirror which formed part of the surface of the right-hand wall. Winston turned a switch and the voice sank somewhat, though the words were still distinguishable. The instrument (the telescreen, it was called) could be dimmed, but there was no way of shutting it off completely. He moved over to the window: a smallish, frail figure, the leanness of his body merely emphasized by the blue overalls which were the uniform of the party. His hair was very fair, his face naturally sanguine, his skin roughened by coarse soap and blunt razor blades and the cold of the winter that had just ended.

Outside, even through the shut window-pane, the world looked cold. Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemed to be no colour in anything, except the posters that were plastered everyplace. The black-moustachio'd face stared down from every commanding corner. There was one on the house-front immediately opposite. BIG BROTHER IS WATCHING YOU, the caption said, while the dark eyes looked deep into Winston's own. Down at street level another poster, torn at one corner, flapped fitfully in the wind, alternately covering and uncovering the single word INGSOC. In the far distance a helicopter skimmed down between the roofs, hovered for an instant like a bluebottle, and darted away again with a curving flight. It was the police patrol, siding into people's windows. The patrols did not matter, however. Only the Thought Police counted.

Behind Winston's back the voice from the telescreen was still babbling away about pig-iron and the overfulfilment of the Ninth Three-Year Plan. The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it, moreover, so long as he remained inside the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even imaginable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live--did live, from habit that became instinct--in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinised.

Winston kept his back turned to the telescreen. It was safer; though, as he well knew, even a back can be revealing. A kilometre away the Ministry of Truth, his place of work, towered vast and white above the grimy landscape. This, he thought with a sort of vague distaste--this was London, chief city of Airstrip One, itself the third most populous of the provinces of Oceania. He tried to squeeze out some childhood memory