NEW DESIGNS FOR IMPROVING THE EFFICIENCY AND RESILIENCE OF NATURAL LANGUAGE WATERMARKING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mercan Karahan Topkara

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2007

Purdue University

West Lafayette, Indiana

This work is dedicated to (anneme) my mom Nuriye, (babama) my dad Mustafa, and (kardeşime) my brother Çağrı for their love and support, and to the strong women in my family (ve ailemdeki güçlü kadınlara, ozellikle anneanneme ve babaanneme,) for the inspiration (adanmıştır)...

ACKNOWLEDGMENTS

Thanks to Mikhail J. Atallah (for being the strongest inspiration, and for the laughs and the great stories); Cristina Nita-Rotaru (for believing in me when I needed it the most, and for the care and support); Edward J. Delp (for the great support, for the VIPER lab where I first learned what a lab group means, and for introducing me to the watermarking community); Giuseppe Riccardi (for creating the first push that taught me how to walk as a scientist and raise my voice); Dilek Hakkani-Tür (for being there when it all started, and for encouraging me to do a research internship which changed the way I look at many things); and Wojciech Szpankowski (for the "Curious Minds" meetings, and for his class on the Analysis of Algorithms on Sequences). Thanks to them all for actively serving on my PhD committee. Thanks to Cüneyt Taşkıran (for always energizing me and killing the inertia in me, and for teaching me how to write a scientific paper), and Eugene T. Lin (for the smiling face and the calm voice when I panicked the most). Thanks to all of the above names for the great mentoring.

Thanks to Turkish Student Association for being my big family away from home, for the great food, and the laughs.

Thanks to my friends for keeping me sane and making me believe that I am alive: Burak Bitlis, İlter Saygın, Ferit Erin, Ali Kumcu, Mehmet Derya Arıkkan, İlke Mollaoğlu, Fırat Y. Çelikler, Barış Çetin Çelikler, İrem Zeynep Yıldırım, Mehmet Koyutürk, Günnur Karakurt Koyutürk, Sibel Sayılı-Hurley, and Sinem Şenol. Thanks to the Turkish folk dance team for sharing all the excitement of dancing in front of a big crowd hand in hand: Eda Ünlü, Emre Ünlü, Celeste Akın, Derya Akın, Deniz Akın, Demir Akın, İrem Zeynep Yıldırım, İlter Saygın, Tamer Çakıcı, Levent Gün, Umut Topkara.

Thanks to my fellow CS graduate students Sundararaman Jeyaraman, Ronaldo A. Ferreira, Asad K. Awan, Hyojeong Kim, Jing Dong, and David Zage for being great friends, and for being there when I needed to talk to someone.

Thanks to William J. Gorman, Amy J. Ingram, Renate Mallus, Janice Thomaz, Deborah C. Frantz and Tammy S. Muthig for being very good in their job, for helping me in many things that seem too hard without their help, for being very understanding, and making me feel like a part of a family in the department.

Thanks to my friends Gönül Doğan, Gülçin Erdem, Harika Burcu Yeğen, Bedriye Azra Timur, Özge Pınar Ülkü, Burcu Özgör, Ceren Denk, Elif Çalıdağ, Zeynep Bediz Turna, Gülşah Doğan, and Aybike Tekin, who have entered my life when I was just a kid, and during all these years they were always there with their unconditional friendship and support.

Many thanks to Umut Topkara for the love, the joy, and the friendship; for sharing the adventure of life; for always encouraging me to believe that I can achieve everything I want; and for being the person he is.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Topkara, Mercan Karahan. Ph.D., Purdue University, August, 2007. New Designs for Improving the Efficiency and Resilience of Natural Language Watermarking. Major Professors: Mikhail J. Atallah, and Cristina Nita-Rotaru.

Contributing our own creativity (in the form of text, image, audio, and video) to the pool of online information is fast becoming an essential part of online experience. However, it is still an open question as to how we, as authors, can control the way that the information we create is distributed or re-used.

Rights management problems are serious for text since it is particularly easy for other people to download and manipulate copyrighted text from the Internet and later re-use it free from control. There is a need for a rights protection system that "travels with the content". Digital watermarking is a mechanism that embeds the copyright information in the document. Besides traveling with the content of the documents, digital watermarks can also be imperceptible to the user, which makes the process of removing them from the document challenging.

The goal of this thesis is to design practical and resilient natural language watermarking systems. I have designed and implemented several natural language watermarking algorithms that use the linguistic features of the cover text in order to embed information. Using linguistic features provides resilience through making the message an elemental part of the content of the text, and through the judicious use of ambiguity in the usage of natural language and richness of features of natural language constituents. In this thesis, I propose several practical and resilient natural language watermarking systems for a variety of genres of text (short, long, edited and cursory text) and analyze their resilience and feasibility.

Significant by-products of this research are as follows: a protocol for improving the stealthiness of information hiding systems; systems for using the proposed information hiding mechanisms to solve the problems of private communication and phishing defense; analysis of the evaluation methodologies and detection techniques for information hiding systems that use natural language text as cover.

1. INTRODUCTION

Even though being able to search and access immense amount of knowledge online has become a part of everyday life, the owner or the authors of digital text do not have control on how their data is distributed or re-used. Rights management pose a serious problem for text, since it is easy for any user to download copyrighted text and re-use it free from control.

There is a need for a rights protection system that "travels with the content", i.e., a technology that can protect the content even after it is decrypted, or after the digital signature is separated from the document in some other way. In this thesis, we investigate the ways of solving the rights protection problem for natural language text by using digital watermarking, an information hiding mechanism that embeds the copyright information within the document. Besides traveling with the content of the documents, digital watermarks are also imperceptible, making the process of removing them from the document challenging.

Digital watermarking technology is very well studied and developed for image, video, and audio. In these environments, imperceptibility of the watermark is achieved by exploiting the redundancy in the data representation format and the limitations of the human perceptual system. We cannot directly apply the information hiding techniques that were developed, for signal-based domains, to natural language text. Unlike images, video and audio, natural language has a discrete nature and follows syntactical and semantical constraints. Natural language has a combinatorial syntax and semantics, and the operations on natural language constituents (e.g., phrases, sentences, paragraphs) are constrained by the grammar of the natural language. In addition, it is very easy for a human to detect differences between an original and modified version of a natural language text document. This makes information hiding in general and digital watermarking in particular, very

challenging in the context of natural language text and may explain why to date work in information hiding in natural language text has been scarce.

1.1 Natural Language Watermarking

In this thesis we propose and evaluate techniques for hiding information in natural language text. Our focus is on using the linguistic features of the sentence constituents in natural language text in order to insert information (i.e., watermark, meta-data, fingerprint etc.) [1]. The goal of this work is to design practical and resilient watermarking systems for natural language text.

This approach is different from techniques, collectively referred to as "text watermarking," which embed information by modifying the appearance of text elements, such as lines, words, or characters [2]. Text watermarking, in that context, is achieved by altering the text format or fonts, such as modifying inter-word and inter-letter spacing in text. Watermarks inserted by most of these systems are not robust against attacks based on performing optical character recognition on a scan of the document, or re-formatting a digital version of the document. Text watermarking based on visual modifications of the digital text documents is outside the scope of this study.

Publicly available methods for information hiding in natural language text can be grouped under two branches. The first branch of methods are based on generating a new text document that will carry a given message. These methods are most commonly used for steganography where the cover text do not have any value and the adversary is passive. Spammimic [3] is an example of this first group. These methods are comparable to automatically generating images, videos or audio files to carry a given information.

The second branch of methods for information hiding into natural language text are based on linguistically modifying a given cover document in order to encode a given message in it. These methods are used for both steganography and water-

marking. Natural language watermarking systems fall under this second branch of methods, where there is also a need for robustness against an active adversary who is attempting to destroy the mark without destroying the *value* of the watermarked document in order to be able to re-use it. In this thesis, we aim to design information hiding systems that modify a given cover document.

A major challenge in watermarking natural language text is assuring imperceptibility (i.e., stealthiness) and preserving the value of the text while being resilient against attacks from an active adversary. The stealthiness requirements and the notion of *value* depend on the genre of the text, on the writer and on the reader characteristics. The common requirements can be summarized as follows:

- **Meaning :** The meaning of the text is the most important component that determines its "value", and it had to be preserved during watermarking in order not to disturb the communication. This is not the case for steganography [4], unless there is a concern of a human warden reading the marked document to check for the existence of a covert communication.

- **Grammaticality :** The embedding process should not generate a text that violates the grammar rules of the language. This is mainly due to preserving the readability and the meaning of the text. In addition, an information hiding system will be vulnerable to statistical attacks unless the grammaticality is preserved. Statistical attacks use statistical modeling techniques to find anamolies in a given object [5]. The aim is to decide whether the object is carrying hidden information (in case of steganography) and if so, which parts are used for embedding the message (in case of watermarking).

- **Fluency :** Fluency is required to represent the meaning of the text in a clear and a readable way. Preserving the fluency of the document is not a common concern for steganography.

- **Style :** Preserving the style of the author is important for certain genres such as literature writing or newspaper columns. It is also important for robustness:

attacks that are based on author detection might be successful in detecting the information carrying items in the document unless the style is preserved [6].

We base our work on well-established research in the area of statistical Natural Language Processing (NLP), in order to build a system that can perform fully automatic and stealthy information hiding in natural language text without destroying the value of the original text.

Compared to other media, natural language text presents unique challenges for information hiding. These challenges require the design of a robust algorithm that can work under the following constraints: (i) low embedding bandwidth, i.e., number of sentences is comparable with message length, (ii) not all transformations can be applied to a given sentence or a word (iii) the number of alternative forms for a sentence or a word is relatively small, a limitation governed by the grammar and vocabulary of the natural language, as well as the requirement to preserve the style and fluency of the document. In addition, almost all of the linguistic transformations defined at sentence level are reversible, which make natural language watermarking systems at this level vulnerable to removal attacks. The adversary can also permute the sentences (i.e., the information carriers), or select and use a subset of them, or insert new ones; a similar attack is harder to achieve with an image, audio or video even for tech-savvy users.

The algorithms presented in this thesis are applicable to any language that allows linguistic transformations on the language constituents (such as synonym substitution, typographical errors or sentence paraphrasing). For the sake of easy access to highly accurate off-the-shelf NLP tools and rich data resources, we apply our algorithms to the English language.

Even though NLP tools are very accurate in performing their isolated tasks (e.g., parsing, part-of-speech-tagging), they were not designed for building an information hiding system. For example, in order to modify a given cover document and re-write it, we need a robust natural language generation system that can perform text-to-text generation. Research in generic text-to-text generation systems is recently

emerging [7]. During the implementation phase of this thesis there were not any off-the-shelf generic text-to-text generation systems available. Most of the available natural language generation systems required the use of a complex input representation language, or they use application-specific models [7]. For testing the feasibility of our sentence level watermarking system, we used a natural language generation system, RealPro, [8] that takes as input a special form of deep sentence structure representation (i.e., DSyntS). The sentence level watermarking system we propose will perform better as the accuracy and coverage of NLP tools and resources improve.

Most of the natural language watermarking systems [9–14] and some of the linguistic steganography systems [4, 15] that insert the message into a given cover document perform the generic steps listed below:

1. **Linguistic Analysis :** A typical watermarking system needs to get information about the syntactic and semantic structure of the given document in order to be able to perform the embedding transformations. This step includes either Part-of-Speech tagging [12, 14], syntactic parsing [9, 11, 16] or semantic analysis [10]. Usually original document is the only input to this step. But, in a few information hiding systems, training corpus of this analysis system is part of the secret key and is chosen separately for every session of information embedding [4].

2. **Selection :** At this step, the words or sentences that will carry information are selected. Inputs to this step are the encrypted watermark message, the secret key and the user parameters. User parameters are usually set by the author in order to preserve the value of the text such as certain style features. For example, the author of a meal recipe would not mind changes in the style of the sentences, but s/he would set the system so that the order of the sentences are preserved.

3. **Embedding :** The watermark message is embedded by applying linguistic transformations. This step varies very widely for every system.

4. **Generating the surface form :** At this step the deep structure is converted into natural language sentence form. Natural Language Generation (NLG) is used in some of the systems that make changes to the forms of the words and linguistic properties of the sentences during the embedding transformations [16]. For example, use of an NLG system is required when the sentence "this frank discussion will **close** this chapter" is converted into "this chapter will **be closed** by this frank discussion". However, NLG is not required when embedding is performed by a transformation that either reorders the words in a sentence, inserts new words (without any need for linguistic restructuring) or removes some words. For example, the following sentence,"**In seconds,** Greg made his decision." can be converted into "Greg made his decision **in seconds**." directly by re-ordering the words [14, 17].

5. **Verification of Embedding :** As a last step, the newly generated text is analyzed to make sure that the embedding was successful. This step is usually performed after each embedding operation. The actual process of verification depends on the design of the system: in the sentence level watermarking system we proposed [16], this step consists of feature extraction on the watermarked sentence and verifying if the features are carrying required selection and embedding values (e.g., the sentence is passive and it has more than one preposition). T-Lex system [15], a synonym-based steganography system, checks if the system successfully replaced the selected word with its synonym that carries the exact mixed radix code required by the secret message (e.g. the word "wonderful" is replaced by "fine"). In addition to the automatic verification of the success of embedding, the author (i.e., the owner of the document) can also be involved in this step in order to audit the changes: (i) if there are more than one option for replacing an original sentence or word, the author can pick which one she/he prefers; (ii) if none of the embedding transformations is applicable on a sentence or a word, the author can be asked to re-write the sentence or replace the word.

See Figure 1.1 for a sketch of a generic natural language watermarking system.

Digital signatures may seem an adequate solution for rights protection for natural language text [18]. However, a major drawback of this approach is that digital signatures are "separable from" the documents because they are not part of the content. Another relevant technology is duplicate detection and document fingerprinting [19]. These methods are limited to detecting duplicates, whereas watermark messages can carry additional information such as meta-data (except in the case of 1-bit watermarks). Another difference between the capabilities of watermark methods and duplicate detection methods is that the duplicate detection methods cannot tell who had the original document first; they just detect duplicates. This is sufficient for the purpose of detecting plagiarism, but fails short of being useful in applications where ability to assert priority is essential.

Steganography applications dominate the previous work in information hiding to natural language text [3,4,15,20]. As mentioned above, in steganography preserving the meaning is less of a concern than being stealthy against a passive warden. This relaxation creates room for high bandwidth embedding. The number of publications in natural language watermarking has been increasing since 2000. To the best of the author's knowledge, using sentence-level syntactic transformations for information hiding (mainly for steganography) in natural language text was first proposed in [21]. When we started this thesis work there were only three papers on natural language watermarking [9, 10, 22]. First one of these three systems uses ASCII values of the words for embedding watermark information into text by performing synonym substitution. This system was vulnerable against random synonym substitution attacks. Both [9] and [10] modify the deep structure of sentences in order to embed the watermark message. The first one modifies syntactic parse trees of the cover text sentences for embedding the watermark message, while the second one uses semantic tree representations. The last two systems use the same feature (the tree structure) for both selection and embedding, which results in using two sentences for embedding 1 bit: first sentence selects, and second sentence (the successor of the first

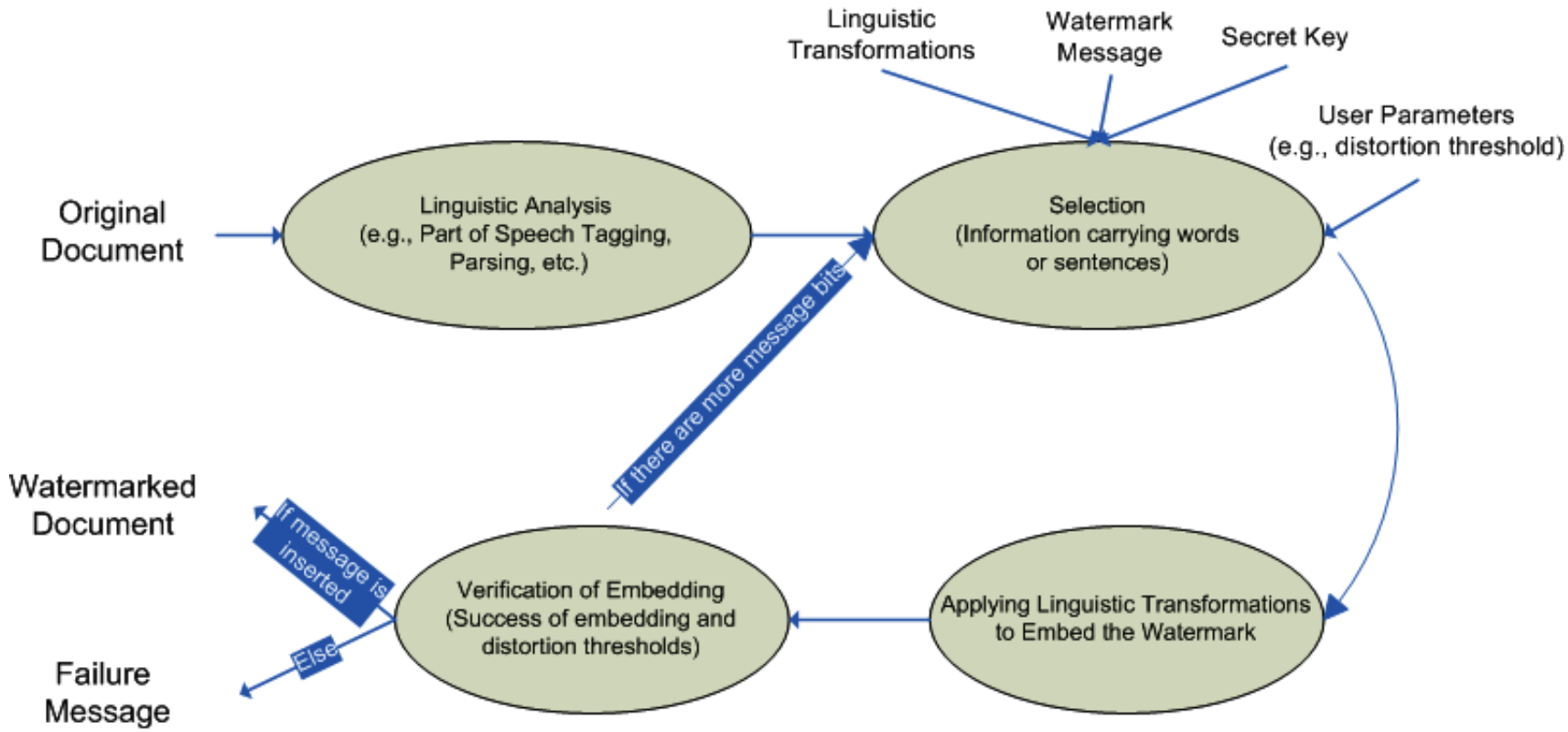


Fig. 1.1. A generic natural language watermarking system

in the cover text) embeds the bit. For this reason, they are vulnerable to re-ordering, insertion, and deletion attacks. Since the exact tree structure is used for embedding information, a change in one of the sentences has a $|M|/n$ chance of damaging an embedded bit, where $|M|$ is the message length and n is the number of sentences in the text. In addition, the second system requires automated semantic parsing and co-reference resolution, which is a very challenging problem [23]. See Chapter 7, for more detailed information about previous work in information hiding into natural language text. We also discuss relevant details of these systems as needed throughout this document.

Our work overcame most of the above-mentioned drawbacks. A brief introduction to the three watermarking systems that were developed during this thesis work will be given in the following sections.

Equmark's core technology, robust synonym substitution, is explained in detail in [24]. Robust synonym substitution is used in Equmark to make the natural language watermarking system more resilient to removal attacks. Details of Equmark has been published in [12]. We will briefly mention its main principles as these principles are essential to understand the merits of the other two systems.

1.2 Equmark: Natural Language Watermarking through Robust Synonym Substitution

Equmark is a lexical watermarking system [12] that achieves good embedding and resilience properties through synonym substitutions. When there are many alternatives to carry out a substitution on a word (that was selected as a message carrier), Equmark prioritizes these alternatives according to their ambiguity, and uses them in that order. In order to measure the "ambiguity" of a candidate word, Equmark uses several measures, including the number of senses of that word.

Equmark builds a weighted undirected graph, G, of (word,sense) pairs based on an electronic dictionary such as Wordnet. Each node in G represents one of the

(word,sense) pairs from the given dictionary. An edge between two nodes shows that they are synonyms. Edge weights show the measure of the similarity between its two endpoints. There are several metrics defined in NLP literature for measuring the similarity between two words [25]. See Figure 1.2 for a sketch of how the Equmark system works.

The algorithm proceeds as follows: A sub-graph, G_W, of G is selected using a secret key. Later, G_W is colored in three different colors corresponding to the one bit information carried by these words, (i.e., they are either carrying a "0", or a "1" or they are "non-encoding"). This coloring is performed in a way that the homographs in the same synonym set get opposite colors. At the embedding time a similarity metric is used to quantify the distortion, d_1, on the meaning of the text due to the transformations. In addition, Equmark quantifies the estimated distortion, d_2, that will be done by the adversary in case s/he decides to perform random synonym substitutions on the marked text in order to damage the watermark. The candidate message carrying word that maximizes d_2 while keeping d_1 below a user-set threshold is picked for embedding replacement [12]. While implementing these distortion metrics Equmark follows the information theoretic principles described in [26].

The main idea behind the design of Equmark is using Computationally Asymmetric Transformations (CAT) for embedding, where the process can easily be performed automatically without any time or CPU cost but reversal requires disproportionately larger computational resources or human intervention. Equmark uses the original text and the author's help for deciding on the sense of the words, however the adversary who is willing to un-do the embedding changes will need to perform word sense disambiguation on a more ambiguous text.

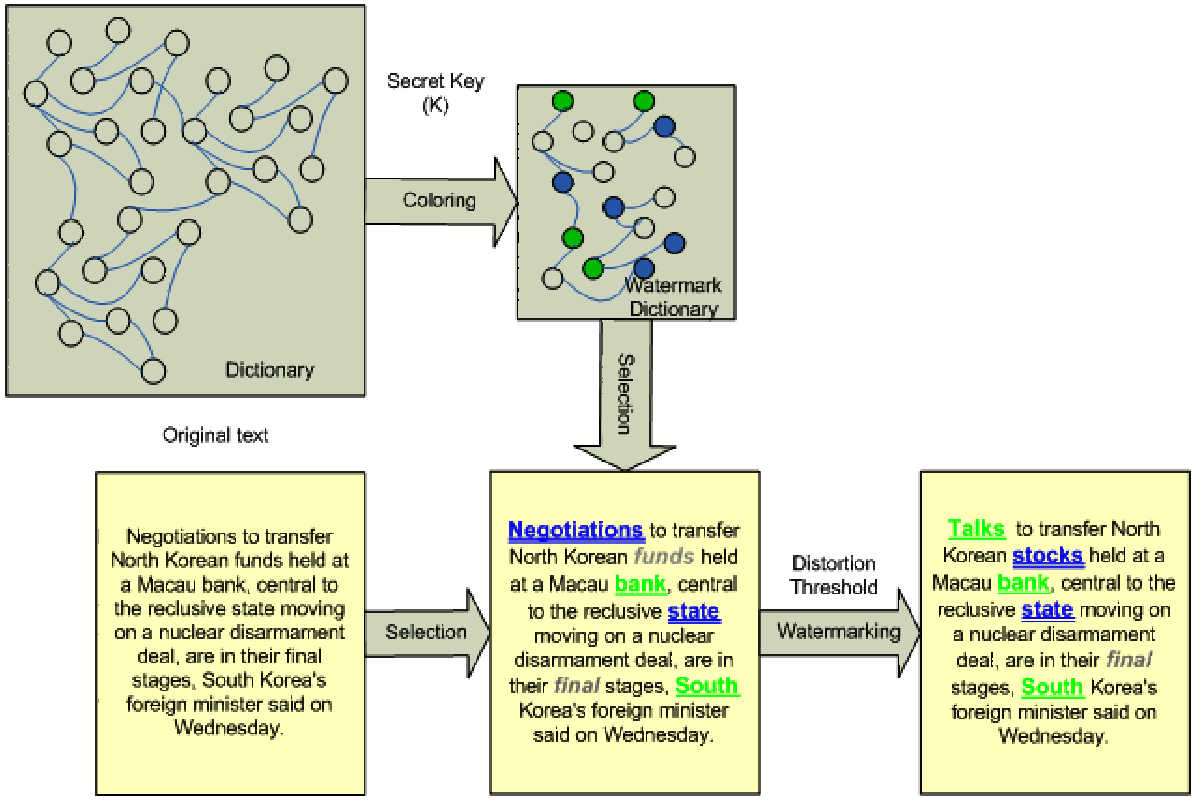


Fig. 1.2. Equmark System Diagram

1.3 MarkErr: Information Hiding through Lexical Errors

Natural language watermarking traditionally targets grammatical, or even edited text where preserving the grammaticality and style is one of the main concerns. The concern for quality of the watermarked text forces systems to perform at a low embedding bandwidth and to put emphasis on the accuracy of natural language processing components.

However, a large percentage of daily exchanged digital text is in the form of e-mails, blogs, text messages, or forums. This type of text is usually written spontaneously and is not expected to be grammatically perfect, nor to comply with a strict style. The freedom from being error-proof and from following a style, creates an opportunity for improved information hiding. This embedding process does not create much distraction in the ongoing communication since humans adapt to errors in this type of text and they are good in spelling correction.

MarkErr [13] uses the typographical errors for embedding information (both for steganography and watermarking) into a given cover text. For the case of existence of a passive adversary (i.e., steganography), MarkErr replaces a selected word with an ambiguous probable typo of it. For example, the word "world" is replaced by "worod", which is similar to the words "wood, word or world". Typos are ambiguous if they are equally similar to many vocabulary words. The probability of a typo is calculated by using a model generated by Kernighan et al. [27] using Associated Press Newswire corpus. MarkErr merges a word and its typos into one set, and uses these sets in a way that any word from these sets can equally be used to encode the same bit at a given position.

When an active adversary (i.e., watermarking) is present, MarkErr replaces a selected word with a probable typo that is also a word in the dictionary, such as replacing "party" with "patty". MarkErr increases the sizes of word-sets for the case watermarking, by merging a word, its synonyms, and their possible typos into one set. This precaution is taken in order to be robust against an adversary that

adds random spelling errors to the text or performs random synonym substitutions. See Figure 1.3 for a sketch of how MarkErr system works.

1.4 Enigmark: Sentence Level Watermarking using Orthogonal Features

While word-level watermarking techniques provide robustness, high-bandwidth and ease of use; they suffer from the *un-avoidable* distortion in the meaning of the cover text. In addition, unless the embedding process is adjusted to author's unique style of vocabulary usage, the style of the text may suffer from the substitutions made by the watermarking system.

We propose combining the powers of word-level and sentence-level watermarking techniques by devising a watermarking system, Enigmark [16], that uses orthogonal features of sentences separately for selection and embedding. The use of orthogonal features make it possible to embed multiple bits into one sentence depending on the richness of its features. A simple example of changing a sentence using orthogonal features is as follows: "I bought milk and cereal from the grocery store" changed into "I purchased cereal and milk from the grocery store", where the word "buy" is replaced with a synonym "purchase", and the places of words "milk" and "cereal" are swapped. This way we are able to embed at least 2 bits into this example sentence.

Enigmark generates a feature space for the sentences in the given cover text, and divides them into two sets: one for selection features and the other for embedding features. After the feature space is generated, the secret key is used for picking which features will be used for selection and embedding of each bit.

Enigmark uses a set of sentences to embed one bit. Note that one sentence can be used for embedding several bits if its part of several sets of sentences, or sets can have any number of sentences (including 1). Two sentences are in the same set if they have a common selection feature. Depending on the size of the cover document the sizes of the sets can be increased to gain robustness.

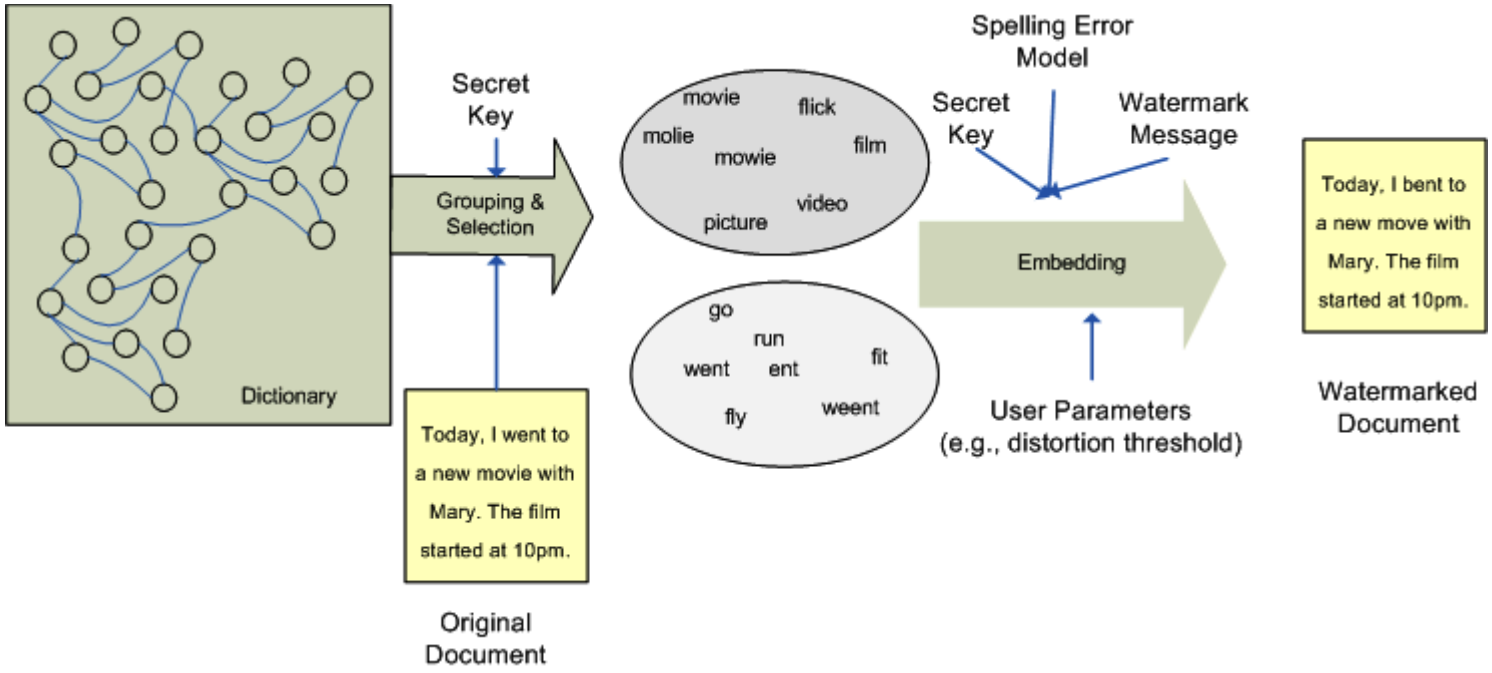


Fig. 1.3. MarkErr System Diagram

The algorithm we propose provides a way for using a lower-level (in this case word-level) marking technique to improve the resilience and embedding properties of a higher level (in this case sentence-level) marking technique. This is achieved by exploiting the orthogonality between some of the word-level features and some of the sentence-level features (i.e., using the word-level marking methods as a separate channel from the sentence-level methods). See Figure 1.4 for a sketch of how Enigmark system works.

Enigmark uses Equmark to assign bit values to the words of the sentences in the cover text (see Section 1.2 for details), and use the value of these bit values for selection. If need be, the robustness of this selection can be improved by performing synonym substitution using Equmark, which would come with the cost of damaging the style and meaning. Embedding features (e.g., the voice of the sentence, or the number of prepositions in a sentence) are modified using meaning preserving syntactic transformations. The complexity of marking process can be lowered by picking embedding features that would not require complex sentence analysis such as parsing. Part-of-speech tagging would be enough for evaluating most of the embedding features.

Unlike Equmark, whose resilience relied on the introduction of ambiguities, Enigmark provides more room for tuning to situations where very little change to the text is allowable (i.e., when style is important and slight changes to the meaning is not allowed.) and strong robustness is needed (where even one bit change to the watermark is not tolerable or the channel is very noisy). The drawback of Enigmark is the need for a rather long text and more complex sentence analysis at embedding time.

1.5 Contributions of this Dissertation

The focus of this thesis work is designing practical and resilient watermarking systems for natural language text. We use meaning-preserving linguistic transforma-

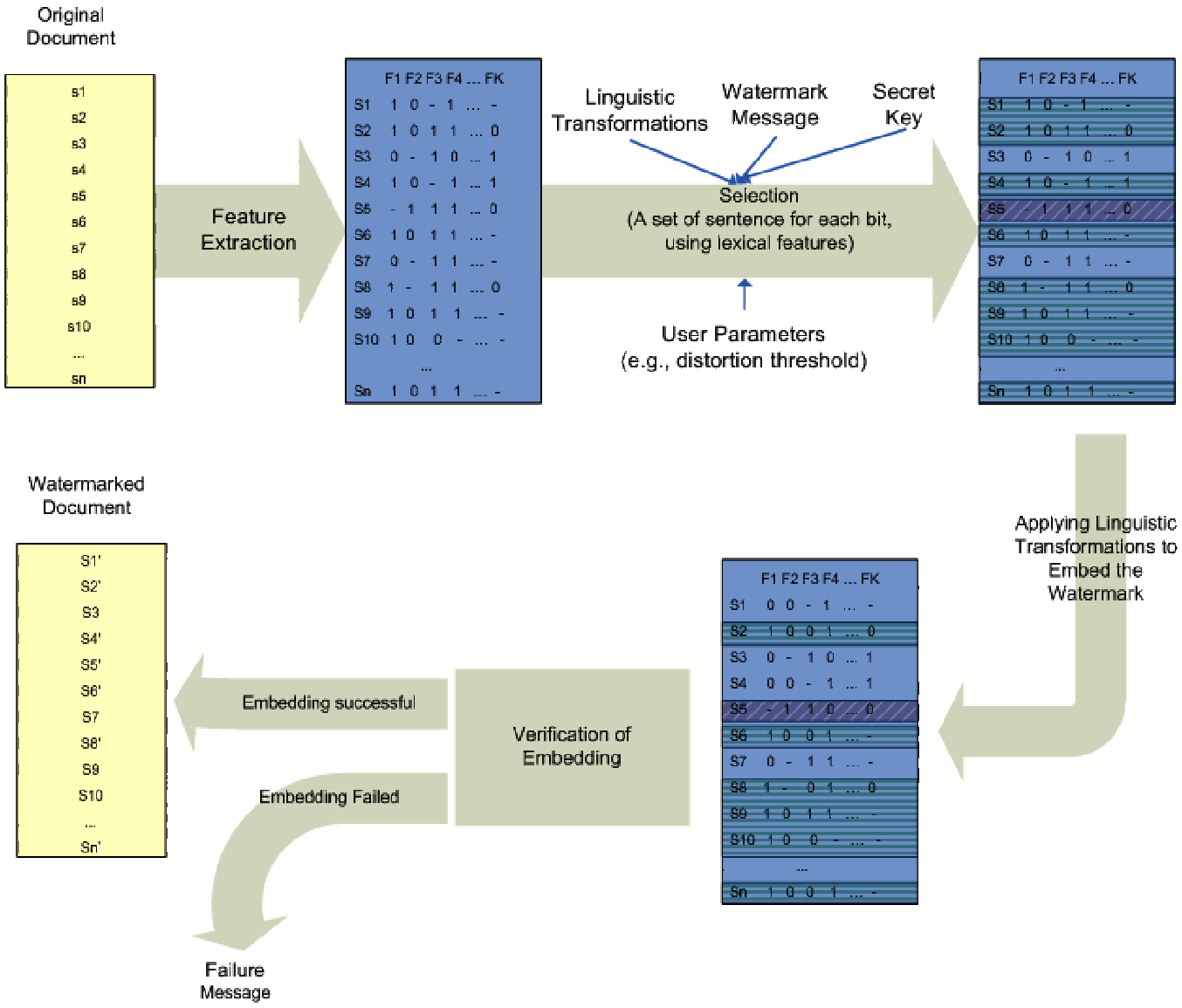


Fig. 1.4. Enigmark System Diagram

tions and statistical natural language processing technology in order to achieve this goal. Natural language watermarking systems modify a given text in order to embed the mark, where there is also a need for robustness against an active adversary who is attempting to destroy the mark without degrading the *value* of the watermarked document.

We summarize the main contributions of this thesis work as follows:

- We design practical and resilient natural language watermarking algorithms that embed the watermark information into the content of the document. In the meanwhile, we analyze the challenges in and the requirements for building natural language watermarking systems.

- We design mechanisms to preserve the meaning and the grammaticality of the cover text while embedding information. Our mechanisms rely on syntax based linguistic transformations such as synonym substitution, spelling error injection, or sentence level transformations.

- We design and implement natural language watermarking systems for a several genres of text, such as edited text and cursory text.

- We design and implement several natural language watermarking architectures for the algorithms we propose. Our architectures make use of existing natural language processing tools and resources such as Wordnet [28], XTAG parser [29] and Realpro [8], as well as new mechanisms that enables compatibility between these components.

- We quantify the effect (i.e. distortion) of our lexical watermarking systems using several word-similarity metrics and spelling error models. We evaluate the coverage of our sentence level watermarking architecture with BLEU metric, a well known evaluation metric for machine translation. We discuss several approaches to the evaluation of natural language watermarking systems.

- We propose two new applications for watermarking: one for private communication and another for phishing defense.

1.6 Organization of this Dissertation

Chapter 2: Background on Information Hiding and Natural Language Processing: This chapter provides the necessary background in information hiding and natural language processing in order to make the reader more comfortable with the terms and concepts related to natural language watermarking. The main goals of natural language watermarking and the challenges involved in building a natural language watermarking system are also discussed in this chapter.

Chapter 3: Lexical Natural Language Watermarking: In this chapter, we introduce the two systems, Equmark and MarkErr, that perform natural language watermarking at the word level.

Chapter 4: Sentence Level Natural Language Watermarking: In this chapter, we propose a rather generic information hiding algorithm, Enigmark, into natural language text at the sentence level, where the carrier medium and the adversary model presents unique challenges.

Chapter 5: Improving Stealthiness by Adaptive Embedding: This chapter presents a new protocol that utilizes a tree-structured hierarchical view of the cover object and determines regions where changes to the object for embedding message data would be easily revealed by an attacker, and are thus to be avoided by the embedding process.

Chapter 6: Applications of Information Hiding: In this chapter, we introduce two systems we have contributed to during the development of this thesis by applying watermarking to new problems, namely a private communication and a phishing defense system that uses watermarking.

Chapter 7: Previous Work in Information Hiding into Natural Language Text: This chapter gives detailed information about existing information hiding systems for natural language text.

Chapter 8: Conclusion This chapter summarizes the contributions of this thesis work, briefly points out the limitations as well as the future directions of this work.

2. BACKGROUND ON INFORMATION HIDING AND NATURAL LANGUAGE PROCESSING

This chapter provides the necessary background in information hiding and natural language processing in order to make the reader more comfortable with the terms and concepts that are related to natural language watermarking. In addition to providing this background, and its relevance to natural language watermarking, this chapter also includes discussions on several new topics that were studied during the course of this thesis work such as steganalysis for lexical steganography and adaptive embedding. The main goals of natural language watermarking and the challenges involved in building a natural language watermarking system are also discussed in this chapter in Section 2.1.2 and Section 2.3.

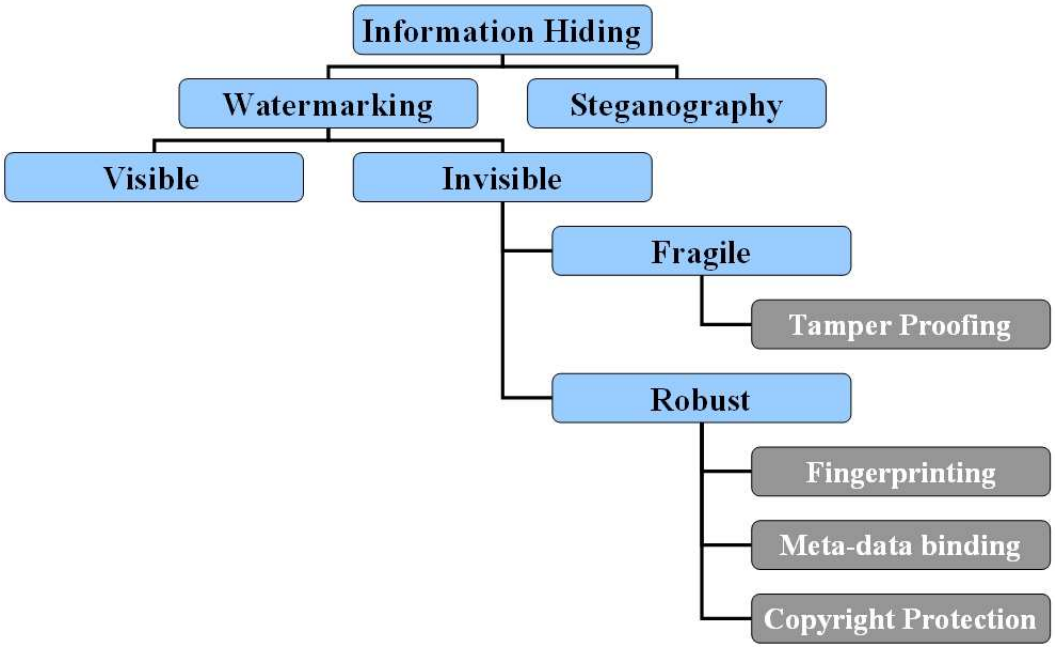


Fig. 2.1. Glance at Information Hiding

2.1 Information Hiding

Even though Information Hiding is a general term used for a wide range of problems, in the context of this thesis the term Information hiding refers to "making the information imperceptible or keeping the existence of information secret" [30]. See Figure 2.1 for a simple chart of the areas classified under information hiding.

Applications of information hiding include: (i) covert communication, (ii) authenticating the source of an object, (iii) proving or denying ownership on an object, (iv) controlling distribution and reuse of intellectual property, (v) meta-data binding, (vi) tamper-proofing, (vii) traitor tracing, (viii) fingerprinting.

2.1.1 Overview of Steganography

The goal of steganography is to send a message $\mathcal{M}$, using a stego object $\mathcal{S}$, in a covert manner such that the presence of the hidden $\mathcal{M}$ in $\mathcal{S}$ cannot be discovered by anyone except the intended recipient. $\mathcal{S}$ does not have any value besides carrying $\mathcal{M}$ and hiding the covert communication. $\mathcal{S}$ is either generated by altering a given cover object $\mathcal{C}$ or it is generated from scratch using a steganography algorithm (e.g., mimic functions [3]). Any type of digital object can be potentially used as a cover. For example, images, audio, streaming data, software or natural language text have been used as cover objects.

In steganography, there are two parties, who can exchange digital objects through a public communication channel, and they would also like to exchange a secret message $\mathcal{M}$. However, they do not want the existence of this secret communication to be noticed by others. Hence, they do not want to achieve confidentiality through encryption, because the exchange of encrypted messages would reveal the existence of their secret communication. For this reason, they use a steganography algorithm to embed $\mathcal{M}$ into a cover object, $\mathcal{C}$, to obtain a stego-object, $\mathcal{S}$, and exchange $\mathcal{S}$ through the public communication channel.

The objective of the adversary, is to construct a method for distinguishing stego-objects from unmodified objects with better accuracy than random guessing. Attack methods generally use statistical analysis to examine a suspicious object and search for characteristics which may indicate that some information has been embedded in the object. For example, the adversary might be looking for an unusual value in a characteristic of $\mathcal{S}$. Studies show that such statistical attacks are very successful on well-known image steganography systems [31–35].

One way to defend against attacks is to inflict as little change to the object as possible [36–38]. To this end, steganography systems try to minimize changes in the cover object $\mathcal{C}$ when $\mathcal{C}$ is converted to corresponding message-carrying stego object $\mathcal{S}$. Due to their statistical nature, some regions in the cover object experience non-significant change in their statistics after embedding. These message-carrying regions will be harder to identify for the adversary. Conversely, some regions will easily reveal their message-carrying characteristics. For example, in the case of an image steganography algorithm that uses random bit flipping, message-carrying regions are easier to identify when the algorithm is applied to smooth regions compared to the case when it is applied to regions with high texture. In this case a region with natural noise is more suitable for message embedding than a smooth region.

Statistical Attacks and Countermeasures to Steganography

Steganalysis is the study of methods and techniques to detect and extract hidden data in stego-objects that are created using steganography techniques. These techniques generally introduce some amount of distortion in the stego-object during message embedding, even though this distortion may not easily be detected by a human observer. Steganalysis methods aim to exploit this fact by detecting statistical effects caused by the distortion to distinguish between cover objects and stego-objects. The challenge of designing a steganography technique is to introduce the distortion in such a way as to minimize its statistical detectability by steganalysis.

One approach, which was taken by early steganography methods, was to try to minimize the detectability of data hiding by introducing as little distortion as possible during embedding. However, as pointed out by Fridrich and Goljan [39], advances in steganalysis have shown that this approach does not guarantee robustness against steganalysis.

One of the first practical works on robustness against statistical attacks was [31] by Pfitzman and Westfeld. This study introduced a statistical attack on stego-objects. This attack is based on the chi-square test, where the estimated color histogram distribution is compared with its observed values. Then the chi-square value, which shows the deviation from the expected values, is used to estimate the probability that a given image has information embedded in it.

Provos [32] proposed a generalized chi-square attack that is capable of detecting more subtle changes in stego-objects. He introduced two methods for decreasing the distortion of the embedding process and for defending against generalized chi-square attack. A pseudo-random number generator is used to create multiple groups of bit selection for embedding. The selection that causes the fewest changes to the cover object is used for embedding. Later, error correction is applied to compensate for detectability caused by the embedding process. Provos incorporated these ideas in his steganography system, *Outguess*, that embeds bits in the LSBs of DCT coefficients for JPEG images. He used a two-pass algorithm, where bits are embedded in the first pass and changes are made to coefficients in the second pass to match the histogram of DCT coefficients of the stego-image with that of the cover image. Since chi-square attacks rely on the first order statistics of the image, this makes the *Outguess* system immune to such attacks.

Westfeld, in his steganography system *F5* [40], decrements the DCT coefficient's absolute values instead of overwriting the LSBs, in order to defend against chi-square test proposed in [31]. *F5* also uses matrix encoding to restrict the necessary changes on the cover object to embed the message. Matrix encoding helps to improve embedding efficiency significantly. Embedding efficiency is the ratio of embedding rate

and necessary changes per message bit. Besides these, message bits are distributed over the whole cover image using permutative straddling.

Later a number of steganalysis algorithms successfully attacked these steganography systems. Fridrich et al. discuss a general methodology for developing attacks on steganography systems using the JPEG image format, which is also effective for the *Outguess* and *F5* system [41]. Their approach is based on the assumption that there is a macroscopic quantity that predictably changes with the length of the embedded secret message for a given embedding method. Lyu and Farid [34] propose an attack that universally works for any steganography system using images. It is based on higher-order statistical models of natural images, where use is made of a wavelet-like decomposition to model images and train a classifier with this model. This classifier is then used for classifying images as a cover image or a stego-image.

Sallee [42] proposed an information-theoretic method for both steganography and steganalysis. A statistical model of the cover media is used to estimate $\hat{P}_{X_\beta|X_\alpha}(X_\beta|X_\alpha = x_\alpha)$ where x_β is the part of the cover object that is used for embedding and x_α is the remaining part which is unperturbed. Then this model is used to select the value x'_β that conveys the intended secret message and is also distributed according to estimated $\hat{P}_{x_\beta|x_\alpha}$. This steganoghraphy method works for any type of cover media. Moreover, if this system is used, capacity of a cover medium can be measured using the entropy of the conditional distribution $\hat{P}_{x_\beta|x_\alpha}$ for a given x_α.

Fridrich et al. in [43] introduced wet-paper codes to provide a mechanism for embedding throughout the cover text while letting the user to mark several places in the cover object as *untouchable*. Wet-paper codes achieve stealthy steganography by allowing the sender to use an arbitrary selection channel (that inflicts minimum distortion on the cover object) for embedding a secret message without letting the receiver know about the selection channel. In [43], Fridrich et al. shows that the embedding efficiency improves when random linear codes of small codimension is used as wet-paper codes.

For in-depth discussion of other work on steganalysis and steganography techniques the reader is referred to [35] and [44].

Improving Stealthiness by Adaptive Embedding

In [38], we propose a new protocol that utilizes a tree-structured hierarchical view of the cover object and determines regions where changes to the object for embedding message data would be easily revealed by an attacker, and are thus to be avoided by the embedding process. This protocol is designed to work in conjunction with information hiding algorithms during the process of embedding in order to systematically improve their stealthiness. It is designed to work with many digital object types including natural language text, software, images, audio, or streaming data.

The protocol requires the existence of a heuristic *detectability* metric which can be calculated over any region of the cover object and whose value correlates with the likelihood that a steganalysis algorithm would classify that region as one with embedded information. By judiciously spreading the effects of message-embedding over the whole object, the proposed protocol keeps the detectability of the cover object within allowable values at both fine and coarse scales of granularity. Our protocol provides a way to monitor and to control the effect of each operation on the object during message embedding. More detailed discussion about this scheme is provided in Chapter 5.

Steganalysis on Lexical Natural Language Steganography

The increase in the significance of electronic text in turn creates increased concerns about the usage of text media as a covert channel of communication. These concerns are especially urgent for text media since it is easier for non-tech-savvy users to modify text documents compared to other types of multimedia objects, such as images and video.

Since the theory and practice of information hiding into natural language is still in its infancy, there has been little emphasis in previous literature on testing the security, stealthiness and robustness of the proposed methods using various attacks.

Natural language steganography methods employ lexical, syntactic, or semantic linguistic transformations to manipulate cover text and embed a message. In this section we will focus on a steganalysis method for *lexical steganography*. Lexical steganography is based on changing the words and other tokens in the cover text in order to hide a secret message.

In [45], we proposed a lexical steganalysis system that exploits the fact that the text manipulations performed by the lexical steganography system, though they may be imperceptible, nevertheless change the properties of the text by introducing language usage that deviates from the expected characteristics of the cover text. This method may be summarized as follows: First, the cover-text and the stego-text patterns are captured by training language models on unmodified and steganographically modified text. Second, a support vector machine (SVM) classifier is trained based on the statistical output obtained from the language models. Finally, a given text is classified as unmodified or steganographically modified based on the output of the SVM classifier. The SVM classifiers were previously used successfully for text classification [46] and were proven to be effective as a universal steganographic attack when images were used as cover objects [34,47]. The performance of this steganalysis approach was tested on a lexical steganography system proposed by Winstein [48].

2.1.2 Digital Watermarking

The hidden information in a object can serve several purposes, a common one is to ensure the genuineness of that information. One way of hiding information is *digital watermarking*, a mechanism which allows imperceptible, robust and secure information embedding directly into original data. Digital watermarking aims to embed information by modifying original data in a discreet manner, such that the

modifications are imperceptible when the watermarked data is consumed and the embedded information is robust against possible attacks. Digital watermarking is applied to several types of cover media such as image, video, audio, software, numerical databases and natural language text. Applications of digital watermarking include content protection, meta-data binding, tamper-proofing and traitor tracing. Readers are referred to [30, 44] for more in depth information about the general principles of digital watermarking.

Although it is not very common to sign an image or an audio file after creating it, signing a natural language text, such as e-mail messages, is very common. There are many secure e-mail tools such as Privacy Enhanced Mail (PEM), Secure Multipurpose Internet Mail Extension (S/MIME) and Pretty Good Privacy (PGP) [1]. But, still the creator or the owner loses control on how the document is distributed after it is separated from the digital signature. There is a strong need for an alternative or complement to these methods, a technology that can protect the content even after it is decrypted or digital signature is separated from the document [30]. Besides being inseparable from the documents, watermarks are also imperceptible.

More specifically, we would need watermarking for natural language text for the following reasons:

- Watermarking provides a robust solution when we want to combine meta-data information with the document in a way that they are not separable. Meta-data can be the information about the source of the content of the document, the security level of the document, the creator of the document, the original physical features of the document i.e. size of the document or the time stamp on the document. Watermarking will also be better than a header or a footnote that is separable from the document. We have used a similar idea of combining the content with a watermark in order to provide a mechanism to authenticate the owner of a web page to the users of the page [50]. See Chapter 6 for

[1] A good discussion of certificate based security is provided in [49].

a defense system we have build for mitigating phishing attacks using visible watermarking.

- Fragile watermarking is needed for tamper-proofing. Tamper-proofing is required when the owner of the document needs to be aware of or prove any tampering with the document. For example, manuals of nuclear submarines are in the class of documents where tampering would create highly dangerous results.

- Since watermarking is imperceptible, it can be used for traitor-tracing. In this case, the document is marked with a different watermark for each recipient. This idea was used by Margaret Thatcher, in early 1980s, to find out who was leaking information about the confidential cabinet documents to the English press.

- The text document and the digital signature can be separated. After the signature is removed, there is no information in the content of the document that can be used for proof of ownership or authentication. Moreover, the adversary can re-sign the document with his own signature. The document or the digital signature must carry a time-stamp for resolving a conflict of ownership in this case.

- It is not always expected from readers to have the capability to verify digital signatures when they are reading a web page. They may not as well be willing to deal with managing the overhead of approving the owner of a digital signature in their "leisure Internet surfing".

Achieving robustness and imperceptibility while embedding information, creates different challenges for different kinds of data. Throughout the rest of this section, we will discuss how natural language as an information carrier differs from other multi media objects that are used for information hiding.

We will mainly focus on image watermarking as a representative of watermarking non-text multimedia data, significant exceptions of audio and video watermarking will be mentioned at places that apply.

In image watermarking imperceptibility is achieved by exploiting the "redundancy" in images and the limitations of the human visual system. Similar approaches are used in other multimedia watermarking domains, such as video and audio. On the other hand, natural language has a syntactical structure that makes such techniques more difficult to apply. Specifically, natural language, and consequently its text representation, has several important properties that differ from image representations.

- Sentences have a combinatorial syntax and semantics. That is, structurally complex representations are systematically constructed using structurally simple (atomic) constituents, and the semantic content of a sentence is a function of the semantic content of its atomic constituents together with its syntactic/formal structure.

- The operations on sentences are causally sensitive to the syntactic/formal structure of representations defined by this combinatorial syntax. Not every embedding operation (linguistic transformation) can be performed on any given sentence (e.g., it is not possible to passivize a sentence with an intransitive verb: "I run every morning").

- The number of alternative (transformed) forms for a sentence is relatively small, a limitation governed by the grammar and vocabulary of the natural language, as well as the requirement to preserve the meaning, style and fluency of the document.

- The embedding bandwidth is very low compared to other multimedia domains. For example, the number of sentences in a typical document are comparable with the length of a typical watermark message.

- The adversary can permute sentences or words, insert new ones or delete some of the existing ones from the information carrying text. In fact, even a non-tech-savvy person can perform this kind of attacks on watermarked text.

Images in general do not lend themselves to a syntactical decomposition similar to the one for language [2].

Besides being perceptible, even small local changes (e.g. randomly swapping the places of two words) in a sentence can change its meaning and/or make it ungrammatical. This is not exactly the case in other multimedia domains, where small local changes will not necessarily effect the meaning or cohesiveness of the document.

Natural language watermarking is not possible without making perceptually significant changes to the content of the text (when a machine or a human compares a cover text and the information carrying version of it, it will be easy for them to point out the differences). For this reason, non-blind watermarking is extremely vulnerable to attacks in natural language watermarking. For example, take the Least Significant Bit (LSB) embedding used in image watermarking as a rough analog to synonym substitution technique used for natural language watermarking. When the user is given the original image and the image watermarked with LSB embedding, s/he will not be able to tell the difference between the two images by just looking at it. But it will be easy for the user to tell the difference between the watermarked and the original copies of the natural language text even when synonym substitution is used.

2.2 Statistical Natural Language Processing

The techniques developed in the field of natural language processing (NLP) aim to design algorithms that analyze, understand, and generate natural language automatically. Since some of the NLP terminology and techniques that will be referred to

[2]Although syntactic approaches to image analysis gained some success in the analysis of some simple, highly structured images, such as electrical circuits and maps, for the most part they have been abandoned since they are not robust for natural images [51].

in the rest of the thesis (and occasionally referred without definition in the natural language watermarking and linguistic steganography literature) may not be familiar to all, in this section we will briefly introduce terms and outline some techniques and resources used in NLP that are of interest for information hiding in natural language text. For an in-depth treatment of the NLP field the reader is referred to [52] and [53].

Data Resources

Success of an information hiding system depends on obtaining good models of the cover medium which requires large data sets. A statistically representative sample of natural language text is referred to as a *corpus*. Since most of NLP research is based on statistical analysis and machine learning systems, large corpora in machine readable form are essential. Therefore, a number of corpora in electronic form have been created and are commonly used in NLP research to train models or for benchmarking purposes, many of them are available from Linguistic Data Consortium [54]. Information about some of the frequently used English text corpora are provided in Table 2.1.

In order to make a corpus more useful for NLP research, the content of it is usually annotated with metadata. An example of such annotation is part-of-speech tagging where information about each word's part of speech (such as verb, noun, adjective) is added to the corpus in the form of tags. PennTreebank corpus has three version of sentence: raw, tagged, and parsed. An example from an entry in PennTreeBank is shown in Figures 2.2 and 2.3

In addition to corpora, there are also electronic dictionaries available that are designed as large databases of lexical relations between words. The most widely known such dictionary is Wordnet [28]. In Wordnet English nouns, verbs, adjectives, and adverbs are organized into synonym sets, each set representing an underlying lexical concept. There are also several semantic relations that link the synonym

Raw sentence (taken from Reuters Corpus):

```
‘‘The House voted to boost the federal minimuma wage for the first time
  since early 1981, casting a solid 382-37 vote for a compromise
  measure backed by President Bush.’’
```

Tagged with Penn Treebank Part-of-Speech Tags and marked noun phrases:

```
[ The/DT House/NNP ]
voted/VBD to/TO boost/VB
[ the/DT federal/JJ minimum/JJ wage/NN ]
for/IN
[ the/DT first/JJ time/NN ]
since/IN
[ early/JJ 1981/CD ]
,/, casting/VBG
[ a/DT solid/JJ 382-37/CD vote/NN ]
for/IN
[ a/DT compromise/NN measure/NN ]
backed/VBN by/IN
[ President/NNP Bush/NNP ]
./.
```

Fig. 2.2. An example sentence from the Penn Treebank.

Parsed Sentence:

```
( (S (NP-SBJ-1 The House)
     (VP voted
         (S (NP-SBJ *-1)
            (VP to
                (VP boost
                    (NP the federal minimum wage))))
         (PP-TMP for
                 (NP (NP the first time)
                     (PP-TMP since
                             (NP early 1981))))
         ,
         (S-ADV (NP-SBJ *-1)
                (VP casting
                    (NP (NP a solid 382-37 vote)
                        (PP for
                            (NP (NP a compromise measure)
                                (VP backed
                                    (NP *)
                                    (PP by
                                        (NP-LGS President Bush)))))))))
```

Fig. 2.3. An example sentence from the Penn Treebank.

Table 2.1
Properties of some of the well known corpora

Name of the Corpus	Size (app.)	Properties
Brown	1,000,000 words	American English 15 different categories of text printed in 1961 (balanced corpus)
Lanchester-Oslo-Bergen	1,000,000 words	British English counterpart of the Brown corpus
Susanne	130,000 words	Freely available subset of the Brown corpus
Wall Street Journal	40,000,000 words	American English financial news articles from 1987 to 1993
Reuters	810,000,000 words	British English 810,000 articles printed from 1996 to 1993
Penn Treebank II	1,000,000 words	Parsed sentences of 1989 Wall Street Journal articles

sets in WordNet such as "is-a-kind-of", or "is-a-part-of" relations. The content of Wordnet is summarized in Table 2.2 [52].

VerbNet [55] is another electronic dictionary which is a verb lexicon with syntactic and semantic information for English verbs, using Levin verb classes [56] to systematically construct lexical entries.

Linguistic Transformations

In order to embed information in a given natural language text, a systematic method for modifying or transforming the content of the text is needed. These transformations should preserve the grammaticality of the sentences to be robust

Table 2.2
Wordnet2.0 Database Statistics

Category	Unique Strings	Number of Senses
Noun	114648	141690
Verb	11306	24632
Adjective	21436	31015
Adverb	4669	5808
Total	152059	203145

against statistical attacks. Ideally, we also require that the differences in sentence meaning caused by such transformations should be imperceptible, in order not to interfere with the readers' experience and to be stealthy against a human warden. Three types of linguistic transformations are commonly used for modifying a given text: i) lexical transformations, where words are substituted for synonyms [12,13,15], ii) syntactic transformations, where the syntax of sentences are modified [9, 11, 14, 16, 57], iii) semantic transformations, where the meaning structure of the sentences are transformed [10].

Lexical transformation (especially the synonym substitution) is the most widely used linguistic transformation for information hiding systems [12, 15]. The Wordnet electronic dictionary is commonly used for this task. The challenge in using synonym substitution is determining the correct sense (i.e., meaning) of the word to be substituted so that the meaning is preserved during embedding [58]. For example, the noun "bank" has 10 different senses listed in Wordnet, including depository financial institution, sloping land, and a flight maneuver. Determining the correct sense of a given word from context, referred to as the *word sense disambiguation* task in NLP, is a challenging problem in general [59].

Syntactic transformations, such as passivization and clefting, change the syntactic structure of a sentence with little effect on its meaning [17]. Some of the common

Table 2.3
Some common syntactic transformations in English.

Transformation	**Original sentence**		**Transformed sentence**
Passivization	The slobbering dog kissed the big boy.	⇒	The big boy was kissed by the slobbering dog.
Topicalization	I like bagels.	⇒	Bagels, I like.
Clefting	He bought a brand new car.	⇒	It was a brand new car that he bought.
Extraposition	To believe that is difficult.	⇒	It is difficult to believe that.
Preposing	I like big bowls of beans.	⇒	Big bowls of beans are what I like.
There-construction	A unicorn is in the garden.	⇒	There is a unicorn in the garden.
Pronominalization	I put the letter in the mailbox.	⇒	I put it there.
Fronting	"What!" Alice cried.	⇒	"What!" cried Alice.

syntactic transformations in English are listed in Table 2.3. Many natural language watermarking methods use syntactic transformations for embedding [9, 14, 16, 57]

The third type of linguistic transformation that has been used for natural language watermarking is *semantic transformation*. These transformations are based on the semantic (i.e., meaning) relations among the words. The semantic relations usually span several sentences. One way of performing a meaning-preserving semantic transformation is using noun phrase coreferences [10]: two noun phrases are coreferent if they refer to the same entity. Based on the coreference concept different transformations may be introduced. One such transformation is *coreferent pruning*, where repeated information about the coreferences is deleted. The opposite of this

operation is *coreferent grafting* where information about a coreference is intentionally repeated in another sentence, or extra information about this concept is added to the text using a fact database. As an example of these semantic transformations, consider the following news story.

> Yet Iceland has offered a residency visa to ex-chess champion **Bobby Fischer** in recognition of a 30-year-old match that put the country on the map. **His** historic win over Russian Boris Spassky in Reykjavik in 1972 shone the international spotlight on Iceland as never before.

The focus of the analysis is the reference item "'Bobby Fischer". Pruning is applied to the first sentence and the extracted information is used to perform a substitution at the second sentence. The modified text is given below.

> Yet Iceland has offered a residency visa to **Bobby Fischer** in recognition of a 30-year-old match that put the country on the map. **Ex-chess champion's** historic win over Russian Boris Spassky in Reykjavik in 1972 shone the international spotlight on Iceland as never before.

One challenge in using the semantic transformations is performing accurate coreference resolution, which is one of the hardest tasks in NLP [23]. Furthermore, two phrases may be coreferent but may have different connotations; in these cases they cannot be substituted without significantly altering the semantic structure of sentences. The phrases *Spiderman* and *Peter Parker* in the following sentences illustrate such a case.

> Spiderman just saved us from death.
>
> Peter Parker just saved us from death.

Natural Language Parsing

In NLP *parsing* is defined as processing input sentences and producing some sort of structure for them [52]. The output of the parsing may either be the morphologic, syntactic, or semantic structure of the sentence or it may be a combination of these.

Parsing is essential to get more information about the sentence structure and the roles of the constituent words in this structure. Most parsers use part-of-speech taggers, which categorize words into predetermined classes (such as noun, adjective, or verb), and morphological analyzers, which break up words into their morphemes in pre-processing steps. Properties of some of the commonly used parsers are listed in Table 2.4. Accuracy of the parsers are tested on WSJ corpus. All parsers, except Link parser, were tested with the standard methodology, where Penn Wall Street Journal tree-bank sections 2-21 is used for training, section 23 is used for testing and section 24 is used for development (development and tuning) [60–62]. Link parser was tested on Switchboard corpus of conversational English. As reported by Grinberg et al.'s [63], even though only 22% of the sentences were grammatical Link parser was able to parse all sentences.

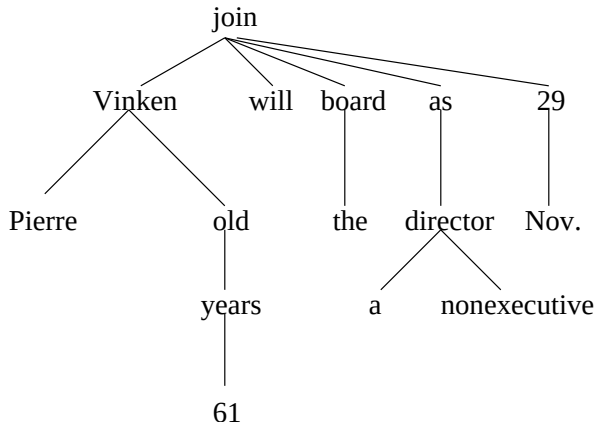


Fig. 2.4. Dependency tree for the sentence, "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."

The parser output may be viewed as a transformed representation of the given text. Various transforms used in image data hiding may be used as a simple analogy to parsing. The input text and the tree relationships produced by the parser are conceptually similar to the time and frequency domain representations of an image.

There are many tools that can convert phrase structures generated by syntactic parsers into dependency trees, which illustrate the argument or modifier relation

Table 2.4
Properties of commonly used syntactic parsers that are freely available.

Parser	Input Format	Output Format	Accuracy ($\leq$ 40 words)
Link, 1995	Raw sentence	Phrase level parse in PennTreebank Format	NA
Collins, 2000	Sentence with part-of-speech tags	Word level parse in PennTreebank Format	90.1%
Charniak, 2000	Raw sentence	Word level parse in PennTreebank Format	90.1%
XTAG, 2001	Raw sentence	Word level parse in Tree-Adjoining Grammar Format	87.7%

between words in the sentences [64]. The dependency tree generated for the sentence "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29." is shown in Figure 2.4.

Word Sense Disambiguation

Word Sense Disambiguation is the process of resolving the meaning of a word, words with more than one meaning are referred as "ambiguous" words. Homograph is a more specific linguistic term used for the "ambiguous" words. Two or more words are homographs if they are spelled the same way but differ in meaning and origin, and sometimes in pronunciation. For example the word "bank" is a homograph, and means either a financial institution, the edge of a stream, or a slope in the turn of a road.

Determining the correct sense of a given word from context, referred to as the *word sense disambiguation* task in NLP, is a challenging problem in general [59].

2.2.1 Statistical Language Models

A language model (LM) is a statistical model that estimates the prior probabilities of n-gram word strings [65]. Language models were previously used for steganalysis to generate a model of word usage patterns for unmodified and steganographically modified text [45]. They can also be used for improving the robustness of watermarking systems [13].

An n-gram LM models the probability of the current word in a text based on the $n-1$ words preceding it; hence, an n-gram model is a $n-1^{th}$ order Markov model, where, given the probability of a set of n consecutive words, $W = \{w_1, \ldots, w_n\}$, the LM probability is calculated using

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_0, \ldots, w_{i-1}), \tag{2.1}$$

where the initial condition $P(w_1|w_0)$ is chosen suitably.

In the NLP field, the goodness-of-fit of a LM to a given text data is usually measured using a quantity referred to as the *perplexity*, rather than using model entropy, as is common in signal processing. The perplexity for a LM, $\mathcal{L}$, is calculated using

$$\text{perplexity}(\mathcal{L}) = 2^{-\frac{1}{N}\sum \log_2 P(data|model)} \tag{2.2}$$

Natural Language Generation

The *natural language generation* (NLG) task is defined as the process of constructing natural language output from non-linguistic information representations according to some communication specifications. NLG maps meaning to text. NLG process can be divided into three phases [66]:

1. Discourse Planning : To select information and organize it into coherent paragraphs

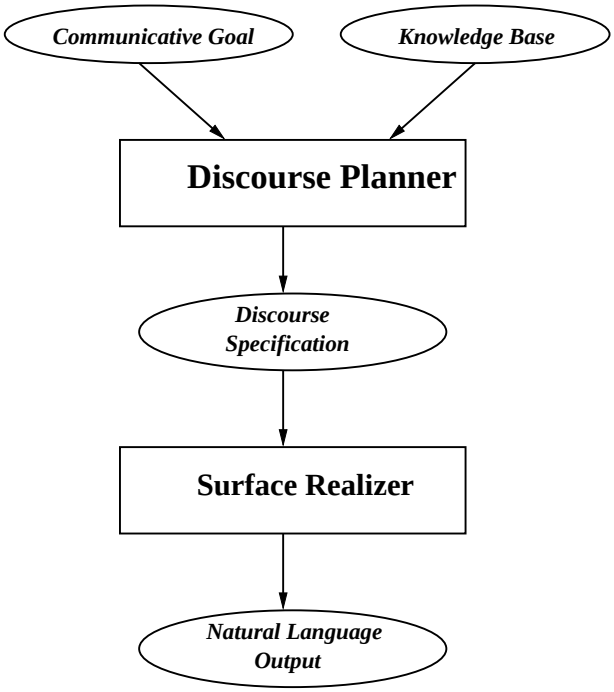


Fig. 2.5. Components of a typical natural language generation system.

2. Sentence Planning : To choose words and structures to fit information into sentence-sized units

3. Surface Realization : To determine surface form of output including word order, morphology and final formatting or intonation

The input to NLG systems varies according to the domain specified by the NLG tool. The components of a typical NLG system are illustrated in Figure 2.5. A good example of a fully functional NLG system is the Forecast Generator (FOG) [67], a weather forecast system that generates bilingual text in English and French. This system takes raw meteorological data and generates weather forecasts. There are several fully implemented NLG systems freely available for research purposes [68].

As far as NL information hiding is concerned, NLG is a crucial component. After information is added to a sentence by modifying its structural representation, this altered representation needs to be converted back to natural language using NLG

systems. NLG systems also play a crucial part in natural language steganography systems as cover text generation mechanisms.

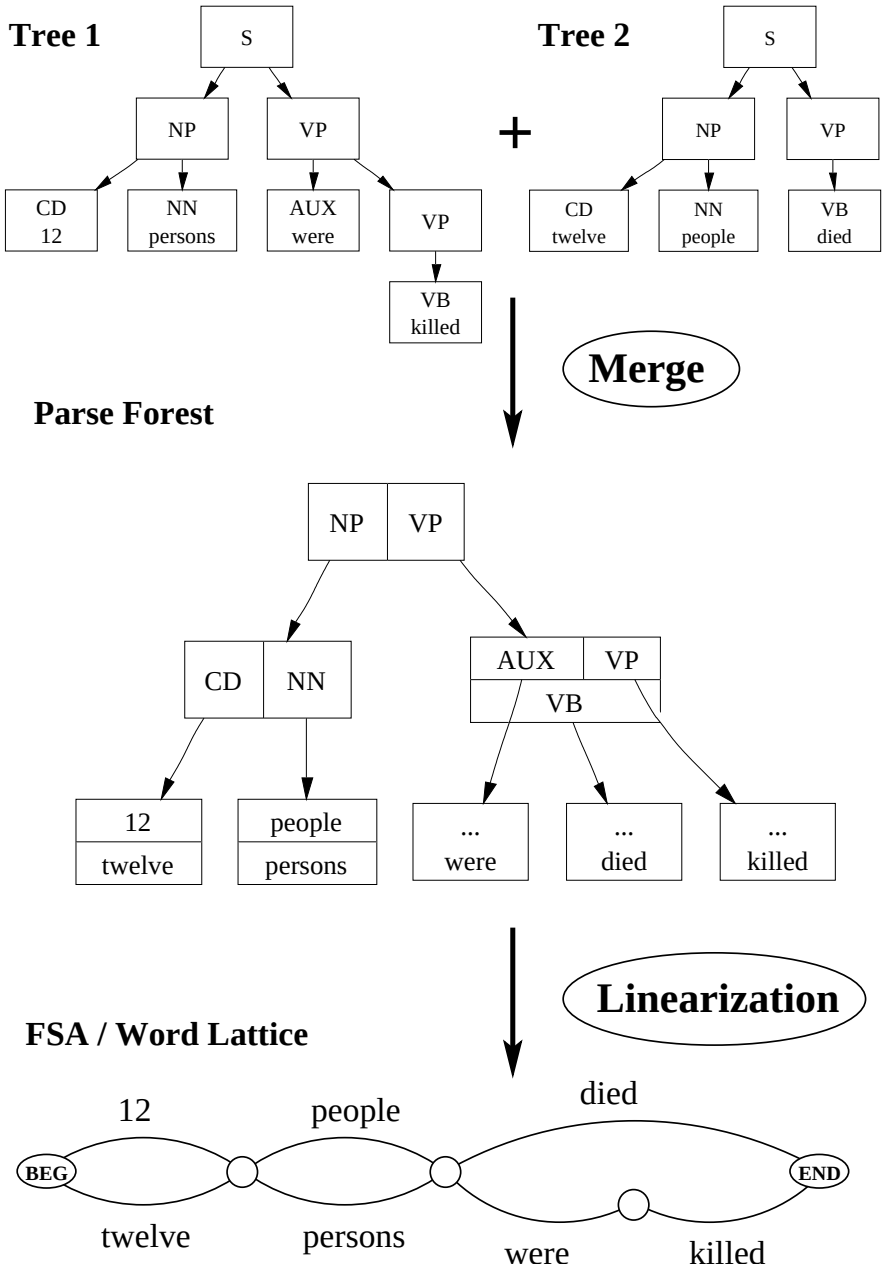


Fig. 2.6. An example of text paraphrasing using a finite-state approach

Natural Language Paraphrasing

The task of *text paraphrasing* entails changing text parameters such as length, readability, and style for a specific purpose without losing the core meaning of the text. Therefore, text paraphrasing is directly related to NL watermarking. Text paraphrasing is also similar to machine translation; however, rather than converting text from one language to another, it is modified from one form to another within the same language. Paraphrasing systems are mainly based on creating or collecting sets or pairs of semantically equivalent words, phrases, and patterns. For example, the sentences

```
After the latest Fed rate cut, stocks rose across the board.
Winners strongly outpaced losers after Greenspan cut interest rates
again.
```

form such a semantically related pair. Such training sentence pairs may be located in news stories covering the same event by using multiple sequence alignment techniques [69].

After the system is trained, given a sentence, it is possible to create a paraphrase using the best matching template pair. An example of text paraphrasing using a finite-state approach [70] is shown in Figure 2.6(c).

2.3 Technical Challenges in Building a Natural Language Watermarking System

Natural Language Processing (NLP) techniques are used for analyzing a natural language text, modifying it, or generating a new text for a given deep structure. NLP techniques are needed for all steps of embedding information into a given text, and in most cases NLP is also needed for extracting the message. Natural language analysis tools have to be accurate and consistent over time in generating the information vital for the functioning of the information hiding systems. One challenge for natural language analysis is the problem of ambiguity, which results in multi-

ple interpretations. The following are commonly used examples of ambiguity at the lexical, syntactic, and semantic levels, respectively:

- *Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.* (This is a grammatical sentence, where word *buffalo* is used in the same sentence with its different senses. It means that [Those] buffalo(es) from Buffalo [that are intimidated by] buffalo(es) from Buffalo intimidate buffalo(es) from Buffalo. [71])

- *I saw a woman with a telescope.* (The prepositional phrase can be attached to both the noun phrase or the verb phrase in the syntactic structure.)

- *Iraqi head seeks arms.* (The word *head* can be interpreted as 'chief' or 'body part' and *arms* can be interpreted as 'weapons' or 'body parts', respectively.)

Large amount of data is required to be able to build models that disambiguate generic natural language sentences.

Another challenge for NLP research is the flexibility and variety of language usage. For example, the vocabulary used in one domain (e.g. sports) varies highly from the vocabulary used in another domain (e.g. finance). Another example is the change in the complexity of the grammatical construction, the syntactic structure of the sentences in a Jane Austen novel is much more complex than the syntactic structure of the sentences in a typical news wire article.

The ultimate goal of NLP systems is to be able to process any natural language input. However, as a result of ambiguity and data coverage, the state-of-the-art tools' accuracies vary widely with the type of analysis. For example, for part-of-speech tagging of English, the best accuracy is around 98% for a given domain [72], for syntactic parsers the accuracy based on labeled precision and recall is around 91% [62].

These inaccuracies are limiting the bandwidth of information hiding. In the case of sentence level watermarking, certain sentence forms or contents that can not be accurately parsed or generated are avoided, or more sophisticated transformation rules are engineered to overcome these in-capabilities.

Note that, the inaccuracies in NLP tools are not always against the aims of information hiding tools. In some cases the in-capabilities of NLP tools are even used to the benefit of information hiding mechanisms [4, 12]. More information about these systems is provided in Chapter 3 and Chapter 7.

Another technical challenge involved in constructing a watermarking system, is the integration of NLP tools. Existing off-the-shelf NLP tools are not designed for the purpose of being part of a watermarking system. For example, natural language generation (NLG) tools are designed for specific purposes, such as converting raw meteorological data into human readable weather forecasts [67] or producing individual letters out of questionnaire results. NLG tools that are designed for specific generation tasks have a limited expression power. Text-to-text NLG is an emerging area [73]. Powerful NLP tools will make it possible to design more flexible natural language watermarking systems, since it will relax the restrictions on the selection of information carrying language constituents and increase the variety of embedding transformations. Following chapters give more details on the experimental results of building natural language watermarking systems using existing NLP resources and tools.

3. LEXICAL NATURAL LANGUAGE WATERMARKING: EQUMARK AND MARKERR

Text can be seen as a combination of units of several sizes from words to full document sizes, or even groups of documents. Any of these units can be used as an information carrier while embedding information, as there are several ways to change features of each: we can substitute a word with its synonym or inject typos in it, or merge some of them into an acronym; we can re-write phrases or sentences; we can change the number of sentences in a given paragraph; we can change the style of a document, or the number of paragraphs in it, or use the coreferences in it to change the flow of information.

In this chapter, we will introduce two systems that perform natural language watermarking at the word level. Ideas about one of these systems, Equmark, came out during the studies of this thesis when we were looking for a way to perform hard to undo meaning preserving changes on edited text. Equmark achieves good embedding and resilience properties through synonym substitutions. When there are many alternatives to carry out a substitution on a word (that was selected as a message carrier), Equmark prioritizes these alternatives according to their ambiguity, and uses them in that order. Besides having this one-wayness feature, Equmark allows the owner of the document to set a distortion threshold. Embedding process stays within this threshold, while maximizing the expected distortion that has to be applied by an adversary that is trying to remove the embedding. The second system, MarkErr, exploits the idiosyncrasies of cursory text in order to embed information. The challenge in performing embedding through idiosyncrasies is that they are easily detected and can be undone if the changes are not performed in a stealthy way. We

overcome this difficulty by using spelling errors that convert the original word again into a word from vocabulary using a human spelling error model as a guide.

3.1 Equmark: Natural Language Watermarking through Robust Synonym Substitution

Even though there is a growing interest in information hiding into natural language, there has not been much movement in the direction of quantification that makes possible using the considerable theoretical work on the analysis of the communication channel established by information hiding. To avail oneself of the information hiding model proposed by Moulin et al in [26] requires quantification of the distortion effect of each linguistic transformation. In this work we carry out such an analysis, using a natural language watermarking system based on a novel twist on the old idea of synonym substitution. Section 3.1.1 will discuss how we use the existing information hiding model for the natural language domain.

Equmark is based on improving resilience of synonym substitution based embedding by ranking the alternatives for substitution according to their ambiguity and picking the one that has maximum ambiguity within the synonyms (subject to not exceeding the maximum cumulative distortion limit). The encoding is designed in a way that the decoding process does not require the original text or any word sense disambiguation in order to recover the hidden message. This system follows the Kerckhoff's rule, namely, that the decoding process depends only on the knowledge of the secret key and public domain information (no "security through obscurity"). Refer to Section 3.1.1 for a discussion of the model of adversary. The details of the proposed watermarking system are explained in Section 3.1.2.

Even though we have focused our attention directly on synonym substitution based watermarking, the analysis and discussions made in this work shed light on the information theoretic analysis of other systems that achieve information hiding through approximately meaning-preserving modifications on a given cover text.

Equmark's core technology, robust synonym substitution, is explained in detail in [24]. Robust synonym substitution is used in Equmark to make the natural language watermarking system more resilient to removal attacks. Details of Equmark has been published in [12]. We will briefly mention its main principles as these principles are essential to understand the merits of the other two systems presented in this thesis, namely MarkErr and Enigmark.

3.1.1 The General Framework of Equmark

This section discusses the general framework we use, including our model of the adversary. Where appropriate, we explain how the peculiarities of the natural language application domain pertain to the framework.

Review of Distortion Quantification

Here we briefly review the general model proposed by Moulin et al in [26] and use the same notation, as applicable. In this notation, random variables are denoted by capital letters (e.g. S), and their individual values are donated by lower case letters (e.g. s). The domains over which random variables are defined are denoted by script letter (e.g. $\mathcal{S}$). Sequences of N random variables are denoted with a superscript N (e.g. $S^N = (S_1, S_2, ..., S_N)$).

Natural language watermarking systems aim to encode a watermark message, M, into a given source document, S^N, using a shared secret, K^N, where K^N is the only side information shared between the encoding and decoding processes. The goal of the encoding process is to maximize the robustness of watermark against possible attacks while keeping the distortion inflicted on S^N during watermarking within allowable limits. There are two distortion constraints on a given natural language watermarking system.

The first distortion constraint is introduced to capture the fact that the watermark encoding process, $f_N : \mathcal{S}^N \times \mathcal{M} \times \mathcal{K}^N \rightarrow \mathcal{X}^N$, has to preserve the "value" of

the source document, while creating the watermarked document X^N. Moulin et al formalizes this constraint as below:

$$\sum_{s^N \in \mathcal{S}^N} \sum_{k^N \in \mathcal{K}^N} \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} p(s^N, k^N) d_1^N(s^N, f_N(s^N, m, k^N)) \leq D_1 \tag{3.1}$$

where p is the joint probability mass function and d_1 is a nonnegative distortion function defined as $d_1 : \mathcal{S} \times \mathcal{X} \rightarrow R_+$. The distortion functions d_i, where $i \in \{1, 2\}$, are extended to per-symbol distortion on N-tuples by $d_i^N(s^N, x^N) = \frac{1}{N} \sum_{k=1}^{N} d_i(s_k, x_k)$.

The second constraint denotes the maximum distortion an adversary can introduce on the modified document, Y^N, without damaging the document's "value" for the adversary. The constraint on the attack channel for all $N \geq 1$ is formalized as below:

$$\sum_{x^N \in \mathcal{X}^N} \sum_{y^N \in \mathcal{Y}} d_2^N(x^N, y^N) A^N(y^N | x^N) p(x^N) \leq D_2 \tag{3.2}$$

where $A^N(y^N|x^N)$ is a conditional probability mass function that models an adversary who maps $\mathcal{X}^\mathcal{N}$ to $\mathcal{Y}^\mathcal{N}$, and d_2 is the adversary's distortion function (similar to d_1). The decoder process receives Y^N.

For image, video or numeric databases, the space can be modeled as a Euclidean space and the effect of changes on the objects can be quantified as a continuous function [26, 74]. However, it is rather hard to model the natural language text input. The value of a natural language document is based on several properties such as meaning, grammaticality and style. Thus, there is a need for designing a distortion function that measures the distortion in these circumstances.

In fact we cannot even talk of a distance in natural language processing, as the triangle inequality need not be satisfied. For example, both "lead" and "blend" are synonyms of different senses of the word "go".

The difference between the word "lead" and the word "go", and the difference between the word "blend" and the word "go", are rather low, whereas the difference between "blend" and "lead" is high.

We cannot use the part of Moulin et al.'s model [26] that assumes a Euclidean distance, since the triangle inequality does not hold in the natural language framework of our application. However, the other requirements that the difference function must obey, are satisfied, namely

Boundedness This is the requirement that the distortion is finite. This holds in our case, because no matter how different two sentences are, our difference function between them will produce a finite outcome.

Symmetry This is the requirement that $d(a, b) = d(b, a)$. The numbers we use for differences are weights of edges in an *undirected* graph (as will become apparent in section 3.1.2).

Equality This is the requirement that $d(a, b) = 0$ if and only if $a = b$. This holds in our case.

Model of the Adversary for Equmark

Our model of the adversary is one who fully knows our scheme (except the key) and has the same knowledge and computational capabilities (including automated natural language processing tools, and access to all the databases used by the encoding and decoding processes).

The approximately meaning-preserving changes that we make are in the direction of more ambiguity, and automated disambiguation is harder for the adversary than it is for us because we start with a less ambiguous (original) document than the one in the hands of the adversary (watermarked document). A human, however, is able to quickly disambiguate when reading the marked text [75].

We carry out substitutions not only for the purpose of encoding the mark in the text, but also for the purpose of getting as close as possible to the allowable cumulative distortion limit, an idea that was previously suggested in a broader framework (see [76]). That is, we keep doing transformations even after the mark is embedded, for the sole purpose of accumulating enough distortion to get close to the allowable limit. This is crucial: The adversary, not knowing the key, does not know *where* we carried out the modifications (as that choice is key-based), and trying to "un-do" them by wholesale application of transformations will cause the adversary to exceed the allowable distortion limit (because s/he started out close to it in the first place).

In practice the adversary is not limited to synonym substitutions; s/he can also make meaning-preserving syntactic changes which effect the ordering of words without altering them [1]. The watermarking mechanism can use an auxiliary fixed syntax with a fixed word order for watermark embedding and detection purposes (e.g. subject, object, verb).

Note that the adversary in our scheme uses an automated process to attack the watermark. Our aim is to raise the bar for the cost of removing the watermark message. In this sense, our scheme can be considered successful if it forces the adversary to manually process the document for removing the watermark.

3.1.2 Synonym Substitution Based Watermarking System

Whereas previous work in this area typically benefits from progress in natural language processing, we propose a watermarking system that benefits from the difficulty of automated word sense disambiguation, as it increases the adversary's complexity of removing the hidden message.

We propose a lexical watermarking system that is based on substituting certain words with more ambiguous words from their synonym set. Here by ambiguous word, we mean a word that is a member of several synonym sets and/or has many

senses. The difficulty of the adversary's task of automated disambiguation is widely accepted in the natural language processing community.

As also explained in Chapter 2, homograph is a more specific linguistic term used for the "ambiguous" words; two or more words are homographs if they are spelled the same way but differ in meaning and origin, and sometimes in pronunciation. We have implemented our system to consider the words with more than one sense as homographs, and only homographs within a synonym set are considered as the target words for synonym substitution.

An example of what our system does carry out today is when we encounter the word "impact" as a verb in our cover text: We will find that it is a member of {affect, impact, bear upon, bear on, touch on, touch} synonym set. Let's assume that the verbs "affect" and "touch" are possible alternatives for replacing the verb "impact" (they both carry the same bit). Our system favors replacing the word "impact" with the word "touch" over the word "affect", because the expected distortion that will be imposed by the verb "touch" on the adversary, $E(d_2(touch; impact, s2))$, is higher than the expected distortion, $E(d_2(affect; impact, s2))$, that will be imposed by the verb "affect". $E(d_2(w_c; w_o, s_o))$ is the average difference of every sense of watermark carrying word, w_c, to the original (word,sense) pair, (w_o, s_o).

In our scheme, more information is available about the sense (meaning) of the words at the watermark embedding time, since the original document and the author is available. The watermarking process presented here, replaces as many as possible words with one of the homographs in their synonym set. Hence the watermarked text has "blurred" meaning and it becomes harder for an adversary to perform word sense disambiguation on it (i.e., the ambiguity has increased in such a way that it is harder to find the correct synonym of the words without human intervention). In such a setting, the adversary will not be willing to replace every homograph word with a non-homograph automatically and the watermark will be successfully retained.

As an example, consider the sentence "he went without water and food for 3 days" coming from a watermarked text. If the adversary had replaced the word "went"

with the word "survived" then the change in the meaning is minimal. However, if he had replaced "went" with "died", the meaning of the sentence would be taken very far from its original meaning. Yet both "survive" and "die" are synonyms of different senses of the word "go".

The decoding process is not dependent on the original text and there is no need to know the sense of a word in order to decode the message. This simplicity of decoding process makes it computationally light, and it enables the copyright infringement detection to be performed by a web crawler on a large number of online documents.

The details of the encoding and decoding processes are explained in the next subsection.

The Encoding and Decoding Algorithms of Equmark

Our system is based on building a weighted undirected graph, G, of (word,sense) pairs, where an edge between two nodes represents that they are synonyms. In our experimental implementation, the synonym sets of words are taken from WordNet [28]. Each weight on a graph's edge is a measure of the similarity between its two endpoints.

Several different techniques and similarity functions have been proposed in the natural language processing literature to quantify the similarity of two words. A large number of these techniques are based on WordNet, which is an electronic dictionary that organizes English nouns, verbs, adjectives and adverbs into synonym sets, each representing one underlying lexical concept [28]. There are several semantic relations that link the synonym sets in WordNet such as "is-a-kind-of", or "is-a-part-of" relations. Some of the word similarity functions are available as a Perl Library called WordNet::Similarity [25, 77]. WordNet::Similarity package implements six different similarity measures that are in some way based on the structure or the content of WordNet.

After graph G is formed, we select a subgraph, G^W of G using the secret key k. This subgraph selection is performed over the words that have homographs in their synonym sets. After this, we use k once more to color the graph in such a way that approximately half of the homograph neighbors of a non-homograph word are colored with blue to represent the encoding of "1", and the other half are colored with green to represent the encoding of "0", while non-homographs are colored with black to represent "no-encoding".

At encoding time, we calculate the expected distortion value for the adversary, which in some sense measures how hard it would be for the adversary to find the original word, w_o, given the mark carrying word, w_c. Note that, if the adversary can replace w_c with w_o, then not only the mark bit encoded by that word will be removed, the distortion introduced by the watermarking process will also be undone. In our implementation, $E(d_2(w_c; w_o, s_o))$ is calculated by summing up the differences of every sense of w_c to the original (word,sense) pair, (w_o, s_o) normalized over the number of senses of w_c, which is denoted with $|S(w_c)|$. This is formalized as below:

$$E(d_2(w_c; w_o, s_o)) = \frac{\sum_{s_i \in S(w_c)} sim(w_c, s_i; w_o, s_o)}{|S(w_c)|} \tag{3.3}$$

where s_o is the sense of the original word, w_o, in the original document, and $sim(w_c, s_i; w_o, s_o)$ is the similarity based difference between (word,sense) pairs, it increases as the words get more dissimilar.

If there are more than one candidate homograph with the same color (the color that is required to encode the current bit of the message, m) then the one with the maximum $E(d_2())$ value is picked. The following are summaries of the encoding and decoding algorithms, based on the above discussion.

Steps of the encoding algorithm:

- Build graph G of (word,sense) pairs. Use WordNet to find synonym sets of (word, sense) pairs. Two nodes are neighbors of they are synonyms of each other. In addition, connect different senses of the same word with a specially

marked edge in order to follow the links to every neighbor of a word independent from its senses.

- Calculate distances between the (word,sense) pairs, $d(wi_{sense_k}, wj_{sense_l})$, using a similarity measure. Assign these values as edge weights in G.
- Select a subgraph G^W of G using the secret key k.
- Color the graph G^W. Detect the pairs of words (w_i, w_j), where w_i and w_j are in the same synonym set with one of their senses, and have more than one sense. In other words, these words act as homographs. Color w_i and w_j with opposite colors in graph G^W, using k to decide which one gets to be colored in blue (i.e, encodes a "1") and which one gets to be colored in green (i.e., encodes a "0"). Color non-homographs as black.
- $c = 1$
- For each word w_i in the cover document S
  - $bit_c = M[c]$
  - if $w_i \in G^W$ then replace w_i with the neighbor that carries the color that encodes bit_c (coloring makes sure that every word either itself encodes "0" or has at least one neighbor that encodes "0", same for encoding "1", refer to [12] for details.)
    if there are more than one neighbor that encodes bit_c
    for each, w_j, of these neighbors calculate
    $E(d_2(w_j; w_i, s_k)) = \frac{\sum_{s_l \in S(w_j)} sim(w_j, s_l; w_i, s_k)}{|S(w_j)|}$
    pick the neighbor with the maximum $E(d_2(w_j; w_i, s_k))$ value and replace w_i with that neighbor
    Increment c (if $c = |M| + 1$ then set $c = 1$)

If the cover document's size is long enough the message, M is embedded multiple times. (The cover document is rejected, if its size is not long enough to embed

the message.) We assume that the message M, that is input to the watermarking system, has already been encrypted and encoded in a way that it is possible to find the message termination point when reading it sequentially from an infinite tape. The encrypted M could have an agreed-upon fixed length (symmetric encryption preserves length, so we would know how to chop the decoded message for decryption). Or, alternatively, if the length of M is unpredictable and cannot be agreed upon ahead of time, the encrypted M could be padded at its end with a special symbol, i.e. #, that would act as a separator between two consecutive copies of the encryption.

Steps of the decoding algorithm:

- Build graph G using just the words. Note that there is no need for sense information or weights of edges between synonyms, only the coding color of a node is needed.
- Select a subgraph G^W of G using the secret key k (This is a symmetric key algorithm. The same key, that was used in the encoding process, has to be used in order to generate the same sub graph.)
- Color the graph G^W using k
- $c = 1$
- For each word w_i in the cover document S
  - if $w_i \in G^W$ then check the color of the node that represents w_i.
    if it is black, move to the next word
    if it is blue, assign 1 to $M[c]$ and increment c
    if it is green, assign 0 to $M[c]$ and increment c

The decoding algorithm is simply a series of dictionary lookups. We envision that this simplicity will enable our system to be used for watermarking online text documents. Then, web crawlers that are indexing web pages can also check for watermarks or metadata embedded using our system in the pages they visit.

3.2 MarkErr: Information Hiding through Errors

A substantial portion of the text available online is of a kind that tends to contain many typos and ungrammatical abbreviations, e.g., emails, blogs, forums. It is therefore not surprising that, in such texts, one can carry out information-hiding by the judicious injection of typos (broadly construed to include abbreviations and acronyms). What is surprising is that, as this work demonstrates, this form of embedding can be made quite resilient. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations that can be carried out inexpensively, yet reversing them requires much more extensive semantic analyses (easy for humans to carry out, but hard to automate). An example of CAT is transformations that consist of introducing typos that are *ambiguous* in that they have many possible corrections, making them harder to automatically restore to their original form: When considering alternative typos, we prefer ones that are also close to other vocabulary words. Such encodings do not materially degrade the text's meaning because, compared to machines, humans are very good at disambiguation. We use typo confusion matrices and word level ambiguity to carry out this kind of encoding. Unlike robust synonym substitution that also cleverly used ambiguity, the task here is harder because typos are very conspicuous and an obvious target for the adversary (synonyms are stealthy, typos are not). Our resilience does not depend on preventing the adversary from correcting without damage: It only depends on a multiplicity of alternative corrections.

Natural language watermarking traditionally targets grammatical, or even edited text where preserving the grammaticality and style is one of the main concerns [10, 16]. The concern for quality of the watermarked text forces systems to perform at a low embedding bandwidth and to put emphasis on the accuracy of natural language processing components.

However, a large percentage of daily exchanged digital text is in the form of e-mails, blogs, text messages, or forums; which we will call *cursory text*. This type

of text is usually written spontaneously and is not expected to be grammatically perfect, nor to comply with a strict style. The freedom from being error-proof and from following a style, creates an opportunity for improved information hiding by applying completely new approaches tailored for cursory text, or by adapting the existing mechanisms that were proposed for edited text.

It is possible to use many idiosyncrasies of cursory text for information hiding, by modifying them or judiciously injecting them to the text. Such idiosyncrasies include:

- Unintentional typographical errors (character typing errors such as "teh" instead of "the").
- Well-known abbreviations and acronyms (e.g. using "ur" instead of "you are" or "omg" for "oh my god", "b4" for "before").
- Transliterations such as leet-speak (e.g. "l33t" for "leet"), pig latin, gyaru-moji or inversion of syllables such as verlan (e.g. "my nopia is kenbro" instead of "my piano is broken").
- Free formatting, such as unnecessary carriage returns or arbitrary separation of text into paragraphs, or varying line sizes.
- Usage of emoticons to annotate text with emotions or attitudes (e.g. ":)" for annotating a pun).
- Colloquial words or phrases (e.g. "gonna" or "ain't nothin"').
- Jargon specific to the age or interest group (e.g. "DCT" is used for "Discrete Cosine Transform" in engineering jargon, and it is used for "Divine Command Theory" in philosophy jargon.)
- Free usage of capitalization and fonts for richer annotation (e.g. "I AM joking").

- Mixed language use, where words from several different languages are used together in the same text (e.g. "We always commit the same mistakes again, and 'je ne regrette rien'!").

- Replacing native characters of an alphabet with latin characters such as writing in "faux cryllic" or writing "sh" instead of letter "ş" in Turkish.

- Grammatical errors.

In this work, we focus on using the typographical errors (henceforth referred to as *typos*), broadly construed to include the above-mentioned acronyms and abbreviations, for increasing the bandwidth of natural language information hiding. When resilience is important (as in watermarking), we make use of ambiguity to make it harder for the adversary to correct the typo. To illustrate this, consider the following example, based on the "lol" acronym that is uncommon, if not unacceptable, in formal text, but very common in cursory text. According to Wikipedia, "lol" has 17 different meanings that depend on the context in which it is used, including the following few:

- Internet slang: "lots of laughs".

- Legal and financial texts: "limits of liability".

- Culinary texts: "Land O Lakes" butter.

- Travel texts: "Derby Field Airport" (less strange than "ord" for O'Hare airport – Derby Field is located in Lovelock, Nevada)

Our current implementation has not yet automated the use of hard-to-reverse acronym substitutions such as "lol" (we used it as an example because it has such a high degree of ambiguity[1] to a text analysis software, yet practically no ambiguity for a human reader).

[1]The list of different meanings of "lol" can be made even longer – "acronymfinder.com" lists 62 different meanings for "lol".

As mentioned above, another way to introduce ambiguity is using latin characters while writing in a non-latin language. The short sentence "iyi isitir" written in ASCII characters, has three different meaningful and grammatical mappings in Turkish script. One mapping is "iyi işitir" ("he/she/it hears well"), another is "iyi ısıtır" ("it heats well"), and the last one is "iyi ışıtır" ("it illuminates well"). This ambiguity occurs due to the many-to-one mapping of Turkish characters to ASCII characters. In its context, a human reader would not have any problem in disambiguating the correct Turkish script of above phrase when written in ASCII. Even though currently available automatic "de-asciifying" techniques [78] have an impressive average case performance on real text, they will not resolve the ambiguities that have been injected with the intention to confuse an automated "de-asciifying" system, such as the one in the above example.

Typos have the advantageous property of being common to all types of cursory text, e.g. emails, text messages, forum posts, etc.; hence an information hiding system based on this notion has a wide range of applications. They are usually injected into the text by their authors as a result of speed typing (e.g "teh" instead of "the"), or incorrect spelling knowledge (e.g. "principal" instead of "principle" or "tommorow" instead of "tomorrow") among other reasons. Typos can occur at any part of the text. However some words are naturally harder to spell or type with keyboard, hence typos occur more frequently with them.

Spelling checkers usually use the edit distance of typos between vocabulary words to suggest corrections. In some cases several corrections are viable for the typo, and spelling correctors use additional side information such as previously observed regularity of typos and models of the underlying natural language or similarity of pronunciation to prioritize this correction list [79, 80]. The regularities in typos are usually due to proximity of key locations on the keyboard, or mental proximity of syllables or words resulting from phonetic similarity. See Section 3.2.5 for more information about spell checkers.

Humans are usually better than automated spelling correctors in typo correction, for this reason, spelling correctors are usually insufficient to completely correct typos, and the author's intervention is required for the final decision. Human fluency in spelling correction also plays a role in the implicit correction of typos by readers of cursory text. Humans use a combination of pattern matching (e.g., resolving "imlipcit" stands for "implicit") and a wide variety of side information that surpass the boundaries of one message exchange, such as shared experience with the author (e.g., a reference to "Jerry" as "Gary"), and real life knowledge ("bear footprint" caption under a human bare footprint picture).

While designing an information hiding system for cursory text using the typographical errors, the following challenges should be taken into account:

- Preserving the value of the document. Even though the editorial quality, grammaticality or fluency of text are less of a concern in cursory text, the result of the information hiding process still has to be easily comprehensible by a human reader. Furthermore, some portions of the text may not tolerate *any* variations since they are central to the meaning of the cover text, such as the meeting place in a memo.

- Automatic spelling checkers can be used to reduce the possibility of using typos for information hiding.

- The adversary might have access to a good model of cover text including a model of the text that was previously generated by each author. The message exchanges can be strictly monitored, such as in company mail accounts, or in blogs.

- The embedding bandwidth in natural language text is lower when compared to that of image, video or audio, i.e., the number of words or sentences are comparable with the message length. For this reason the hidden messages are embedded more densely in the cover text documents when compared to cover objects of other media types.

We have designed two methods of embedding, one for an active adversary (which can be used for watermarking applications), and one for a passive adversary (which can be used for steganography applications). The methods use individual words to embed bits of the secret message. The only information required to read the hidden message is the shared key that was used to embed the message in the first place. Both of the embedding methods spread the modifications evenly throughout the cover text. They also provide the author with the flexibility to exclude parts of the text from being modified by the embedding process. We will discuss these methods in Section 3.2.2 and Section 3.2.3. See Section 3.2.4 for examples of marked text.

Section 3.2.5 briefly covers the literature on the evolution of written language use on the Web, as well as the literature on spelling correction, and other relevant work in information hiding.

3.2.1 Computationally Asymmetric Embedding with Typographical Errors

Robust synonym substitution cleverly uses sense ambiguity to insert watermarks into text that are resilient against an adversary who uses automated synonym substitution to remove a watermark [12]. This resilience is achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations (modifications to a cover document to embed a mark) that can be carried out inexpensively, yet reversing them requires much more extensive analyses that necessitates strong artificial intelligence (easy for humans to carry out, but hard to automate).

The task of robust watermarking with typo injection is harder than that using synonym substitution. There are two major difficulties: i) typos are very conspicuous and an obvious target for the adversary (synonyms are stealthy, typos are not) ii) adversaries can use spelling correction tools to undo the effect of embedding. In order to overcome these difficulties we construct CATs with typos. We replace words with typos such that a spelling checker will produce a long list of possible "corrections",

hence force the adversary to achieve the capability to understand the underlying text to single out the original word from this list. The resilience of such CATs depend on a multiplicity of alternative typo corrections.

For instance, a spelling checker will easily point the typo in the sentence "Don't forget to bring the ake". However, the correction list will be rather long (ispell version 3.1.18 lists "Abe, ace, age, AK, AK e, AK-e, ale, ape, are, ate, Ave, awe, axe, aye, bake, cake, eke, fake, Ike, jake, lake, make, rake, sake, take, wake" as alternative corrections that are 1 unit edit-distance away from "ake"). Unless the context in which this sentence appears is known, it is hard even for a human to figure out the original word for the typo. Note that some of the alternative words in this list are 2 units edit distance away from the original word "cake". Eventually the adversary is likely to end up choosing a replacement word for the typo, which is further away in terms of edit distance from the original word.

Typos whose correction processes are hard to automate are preferred in robust information embedding. Their correction lists contain several words that have same part of speech and have similar meaning as the original word.

Using the same example as above: an embedding system based on the CAT principle will prefer to use the typo "ake" to the typo "cakw" in the sentence "Don't forget to bring the cake". There are two reasons for this: i) "cakw" has a smaller correction list (ispell version 3.1.18 lists "cake, caw" as alternative corrections for "cakw") ii) while words "ale","sake" (from correction list of "ake") have a similar meaning with "cake" and can be used in the same context, it is unlikely that "caw" (from correction list of "cakw") could be used in this sentence instead of "cake".

Furthermore, CATs can be used to achieve stealthiness of typos, besides achieving resilience. We do this by choosing typos that are themselves legal words from the vocabulary (English vocabulary in this case). This deliberate choice for typos forces the adversary to perform the complex task of detecting such typo words that are used in the wrong context.

The sentence "It's going to be a great party" can be changed to "It's going to be a great patty" by the typo injection mechanism. Since the typo "patty" is also a vocabulary word, a spelling checker will not detect this typo. The adversary does not know which word(s) in this sentence is not from the original sentence. Hence, from the point of view of the adversary, the original sentence is only one of a long list of possible sentences that could have been used to create this watermarked sentence. In order to remove the watermark, the adversary will need to first come up with this long list of sentences (which is expected to include the original sentence before watermarking). Then the adversary will need to make a best guess among these sentences to select the original sentence. This last step of the adversary is similar to the hypothesis ranking problem of automated speech recognition and machine translation [52]. It is known that automated solutions to this problem perform well most of the time in practice. However, in this case, as an important difference from the average case behavior of natural language text or speech, the watermarking process will deliberately choose the typos which will make sure the automated ranking process will perform at its worst. In order to foil the automated adversary, the watermarking system is designed to pick the vocabulary words that create the maximum ambiguity (i.e., longest list of alternative corrections with similar probability).

3.2.2 Watermarking with Typographical Errors

Without loss of generality, we will describe the method assuming one bit is embedded per word. For now we assume that there are no *untouchable* word occurrences in the text (words that the user forbids to be modified in the encoding process). At the end of this section we generalize the scheme to work for untouchable words as well.

Let V be the vocabulary from which the words in the cover text D are picked. Let K be the shared secret key. Let M be the secret message.

Embedding Algorithm

1. Replace V by another vocabulary V' obtained from V by merging all words, their synonyms and their possible typos (that are not a word from V) into groups, where each group is represented by one (key-selected) word in V'. The representative word that corresponds to w is denoted by $G(w)$ and it is same for all words in a group. For example, {aircraft, airplane, airliner, jetliner, aeroplane, ..., airpane, arplane, airplan, aiplane, ..., aircaft, arcraft, aircrft, aicraft, ..., etc. } are grouped together and only one of them (say, airplane) is chosen as the group's representative. If a word is eligible to appear in multiple groups then ties are broken arbitrarily using the key K to flip a coin.

2. A word token w in D, is denoted by a pair (w, s), if this corresponds to the sth instance of the group represented by $G(w)$ in D.

3. (w, s) is used for information carrying only if the least significant bit of $H_K(G(w)||s)$ is 1, where H_K denotes a keyed cryptographic hash with K as key, and $||$ denotes concatenation. We make sure that any word from the same group will encode the same bit at a given sequence order (by always using the representative of the group to compute the keyed hash). If the adversary uses synonym substitution, the bit value encoded by the word can still be successfully recovered since all synonyms carry the same bit value. If the adversary chooses to inject non-word typos, it is highly likely that the replacement string will be in the same group as the original word. Otherwise there is a 50% chance to flip the encoded bit value.

4. Process each bit m_i in the message M in the following way:

   (a) Scan through the message and find the leftmost information carrying (w, s) in D that has not yet been processed.

   (b) Use the second least significant bit of $H_K(G(w)||s)$ to determine the message bit value carried by (w, s). w already carries m_i with 50% probability;

in this case we are done and we move on to embed the next message bit m_{i+1}.

(c) If w does not carry m_i, then we try injecting different typos into w and collect, in a set of candidates $\mathcal{C}$, the following two types of outcomes for these typo-injections: (i) the resulting typo word $\bar{w}$ is such that the least significant bit of $H_K(G(\bar{w})||\bar{s})$ is 0; (ii) the resulting typo word $\bar{w}$ is such that the least significant bit of $H_K(G(\bar{w})||\bar{s})$ is 1 but its second least significant bit matches m_i. A central technical issue is: Which of the candidates in $\mathcal{C}$ to select? This is tackled separately in the Candidate Selection algorithm given below. For now we note that if a type (i) candidate is selected, $\bar{w}$ is not used to carry any message bits, and in such a case we skip to next information carrying word to encode m_i. If on the other hand, a type (ii) candidate is selected then it corresponds to using $\bar{w}$ to carry the message bit m_i, and in such a case we continue to embed m_{i+1}.

The next algorithm explains how we select from candidate set $\mathcal{C}$ the best alternative.

Candidate Selection Algorithm

1. Partition the typos, that were used to produce the candidate words, into 2 classes: *Conspicuous* typos, and *stealthy* typos. A candidate word $\bar{w}$ that is *not* in the dictionary (such as "imlipcit" instead of "implicit") is considered to be conspicuous. Respectively, a candidate that is in the dictionary (such as "mat" instead of "man") is considered to be stealthy.

   Note that, some of the conspicuous typos of a word are already in the group of the word, and do not change the encoded bit if they are used; the previous steps of the algorithm does not include such words in the list of candidates. The conspicuous typos that are considered at this step come from other groups; they were assigned to another group as a result of a random coin flip when they were eligible for more than one group.

2. If there are candidates in $\mathcal{C}$ whose typo is of the stealthy kind, then we prune $\mathcal{C}$ by removing from it all the candidates whose typo is of the conspicuous kind. Note that the remaining candidates in $\mathcal{C}$ are all stealthy or all conspicuous. In either case, the next step uses the same criteria for selecting the best candidate.

3. For each candidate typo $\bar{w}$ in $\mathcal{C}$, compute the following function $h(\bar{w})$ for it.

   (a) Let $N(\bar{w})$ be the set of *neighbors* of $\bar{w}$: Word a is in $N(\bar{w})$ if the edit distance from a to $\bar{w}$ is no more than a user set threshold, let's say 2. Intuitively, a is in $N(\bar{w})$ if $\bar{w}$ *could* have resulted from a mis-typing of a. In fact the probability of occurrence of such a typo, the conditional probability $\Pr(a|\bar{w})$, is obtained from confusion matrices that quantify the probabilities of various mis-typings [79]. Assume this has been done for all $a \in N(\bar{w})$. Note that the user set threshold can also be given for the $\Pr(a|\bar{w})$, the probability of $\bar{w}$ being the result of a typo in a.

   (b) Having obtained $\Pr(a|\bar{w})$ for all $a \in N(\bar{w})$, we compute $h(\bar{w})$ as follows:

$$h(\bar{w}) = -\sum_{a \in N(\bar{w})} \Pr(a|\bar{w}) \log \Pr(a|\bar{w})$$

   which is the entropy that we should seek to maximize (thereby maximizing the adversary's uncertainty about where $\bar{w}$ could have come from). Note that, here we have assumed that the adversary will only use the knowledge about typo confusion matrices. A more advanced adversary could use n-gram language models, and in that case ambiguity measure of the watermarking algorithm should be changed accordingly.

4. Select from $\mathcal{C}$ the candidate $\bar{w}$ with the largest $h(\bar{w})$.

Typos are injected in a way that maximizes the possible correction alternatives, while staying within a distortion threshold that captures the damage incurred on the cover document after the injection. The distortion threshold is defined by the owner of the document. Viable typos are generated with respect to the confusion matrices

used by the spelling checkers, which yields all typos of the cover word that could be typed by a human user. See the next subsection for other objective functions that can be used in watermark embedding.

The decoding is a simple key-based reading of the bit values of the (also key-selected) words in the watermarked text.

We note that extending our system so it can handle 3-letter acronyms such as "lol" would require checking (by pattern matching) whether the current word and its 2 successors (predecessors are not used, since they might have been already used for encoding) can form a 3-letter acronym: If so then replacement by the acronym provides one of the alternative "typos" that we could use.

Extending the above system to include the conversion of non-latin characters into latin characters is straightforward, since such conversions can be treated as typos.

In this method, every typo gives away some information to the adversary about encoding words (i.e., they indicate that one of the dictionary words listed by the spell checker in the correction list is an encoding word). The adversary can use this list to flip some of the message bits by injecting typos into occurrences of all the words in the correction list. We can prevent this damage by inserting unnecessary typos into the watermarked document. This measure also helps to increase the amount of distortion incurred on the document by the watermarking process, hence limiting the error tolerance of the adversary.

Candidate Selection Heuristics

The heuristic introduced in step 3.*b* of the Candidate Selection Algorithm maximizes the possible correction alternatives for encoding words. This increases the uncertainty about the original versions of encoding words, and make it harder for the adversary to revert the watermarked text back into original form.

We can alternatively use another heuristic to maximize the probability that the encoded bit will stay the same even if the adversary chooses to randomly replace the encoding word:

For all the candidate words that are encoding the desired bit m_i, pick the information carrying neighbors of $\bar{w}$ that encode m_i and put them into set $N'(\bar{w})$, then compute $\Pr(m_i|\bar{w})$ as follows:

$$\Pr(m_i|\bar{w}) = \sum_{a \in N'(\bar{w})} \Pr(a|\bar{w})$$

Then pick the candidate word that maximizes $\Pr(m_i|\bar{w})$. Here we seek to maximize the probability that adversary will pick a word that is still encoding m_i even though s/he replaces $\bar{w}$ with a word from its neighbors $N(\bar{w})$.

Extension to Untouchable Words

One of the disadvantages of information hiding in text is the low bandwidth of the medium. Since the document units (e.g., words, phrases, etc.) that carry the embedded bits is scarce, every available unit is used for message carrying. Cursory text tolerates such saturation of the text with modifications for most of the time. However, it is not uncommon to have parts of the text which are too sensitive and modifications to the original are not tolerable (such as a date, a salary figure, a military rank). It is important that information hiding systems have a way of accommodating "untouchable document areas" by avoiding modifications to such portions.

It is possible to avoid untouchable areas if "untouchable words" are known beforehand, by skipping the individual words or the phrases and sentences that they occur. However this is not a practical approach for a general purpose information hiding system.

Untouchable words are like the "defective memory cells" in Wet-Paper Codes (WPC), and hence efficient wet-paper codes can be used to handle them [43] for text steganography, see Section 3.2.3 for more details.

In our algorithm, "untouchable words" are problematic only if the word is used to carry the watermark (i.e., least significant bit of $H_K(G(w)||s)$ is 1). An alternative encoding could easily solve this problem when untouchable words are rare and isolated (as opposed to occurring in chunks of text): Encode a message bit m_i (i.e., the bit value carried by jth information carrying word w_j in D) as the XOR of the *third* least significant bit of $H_K(G(w_{j-1})||s_{j-1})$ with the second least significant bit of $H_K(G(w_j)||s_j)$.

This may entail having to backtrack and modify w_{j-1} for the sake of encoding bit m_i in w_j while still satisfying the old requirement imposed on w_{j-1} by its encoding of m_{i-1} or of the (non)information carrying property of w_{j-1}. The drawback of this approach is that it will cause a substantial decrease in the number of candidates in $\mathcal{C}$ for w_{j-1} (approximately by a factor of two), and hence a lower conditional entropy for the chosen candidate from that $\mathcal{C}$. Of course no such burden is imposed on w_{j-1} if w_j is not untouchable. Note that the bandwidth is also largely unchanged – it does *not* go down by a factor of 2 because, when we get to m_j, we do not need to backtrack to m_{j-1} because the 3rd bit of $H_K(G(w_{j-1}), s_{j-1})$ is already fixed by then and its presence in the XOR makes no difference to the success (or lack thereof) of encoding m_i in w_j. Also note that the bandwidth would have dropped by a factor of 2 if we had used the second bit of $H_K(G(w_{j-1}), s_{j-1})$ in the XOR.

3.2.3 Steganography with Typographical Errors

In steganography, stealthiness of the marked document is a more important concern than the robustness of embedding against modifications by an adversary or the value of the cover document. Here the adversary is passive and only interested in detecting the covert communication.

We use the same notation that the watermarking algorithm uses in Section 3.2.2: V is the vocabulary from which the words in the cover text D are picked; K is the shared secret key; M is the secret message.

Let T be the user defined total distortion threshold. We define distortion, $d()$ to be the probability of a typo word, $\bar{w}$, occurring in a text as a result of a typing error made by a human typist that intended to write the original word, w. Hence $d(\bar{w}, w) = \Pr(w) \Pr(\bar{w}|w)$. We use the spelling error probability metric defined by Kernighan et al. in [79]. See Section 3.2.5 for more details on this spelling error correction method. The steganography process ensures that $\sum_{w \in D} d(\bar{w}, w)$ stays below T. Note that this distortion measure assumes an adversary would know the original word w, and use this information to detect a steganography. Hence the adversary model of the steganography system we describe in this section is more sophisticated than an automated system.

We propose a steganography system that has the flexibility to let the user forbid some of the words from being modified. Such words are considered "untouchable" and the marking process finds a way to encode the secret message without touching these words in the cover text. This way, users may i) conserve the value of the cover document from being destroyed by the embedding process, ii) achieve stealthiness to the human eye, which would not tolerate extensive typing errors at critical words. Usually, preserving the value of the cover document is not a primary concern in steganography. However, there might be automated and manual filtering systems that have been installed to remove the documents that do not serve the purpose of the communication channel (e.g., spam filters, or a collaborative filtering system in a forum in which other users vote to remove posts with no relevant information).

Our steganography algorithm uses efficient Wet-Paper Codes (WPC) technique presented in [43] to achieve minimum distortion while allowing the user to mark certain sections of the cover text as untouchable. These untouchable sections will act as "defective memory cells" in WPC.

Let L denote the sections that are untouchable.

Embedding Algorithm

1. Replace V by another vocabulary V' obtained from V by merging all words and their possible typos into groups, where each group has a (key-selected) *representative* word for that group. For example, {word, owrd, wrod, wodr ...etc. } would be grouped together and only one of them is chosen as the group's representative. The representative word that corresponds to w is denoted by $G(w)$ and it is same for all words in a group. If a word is eligible to appear in multiple groups then ties are broken arbitrarily using the key K to flip a coin.

2. A word token w in D, is denoted by a pair (w, s), if this corresponds to the sth instance of the group represented by $G(w)$ in D.

3. (w, s) is used for information carrying only if the least significant bit of $H_K(G(w)||s)$ is 1, where H_K denotes a keyed cryptographic hash with K as key, and $||$ denotes concatenation.

4. Read the bit string that D carries (before the message embedding) into the string B as follows:

   (a) For each information carrying word w in D, assign the second least significant bit of $H_K(w||s)$ to b_i in B. (Note that the word w is directly included in the hash in this case.)

5. Use the efficient wet-paper code technique [43] to generate a set of bit strings, $\mathcal{B}$, that can encode M without using the bits that fall into the sections marked by L and whose hamming distance to B is below a threshold (this threshold is set a priori).

6. For each $B' \in \mathcal{B}$ use the embedding transformations defined in the watermarking algorithm in Section 3.2.2 to embed B' into D and get the marked text D'. The only difference here is the candidate selection criteria: while picking a typo from the possible candidates in the set $\mathcal{C}$, we pick the candidate, $\bar{w}$, that

is the most probable typo (i.e. $\bar{w} = argmax_w \Pr(\bar{w}|w)$) in the list of typos that can encode the desired bit, m_i.

7. Pick the B' that inflicts minimum total distortion on D, and output the corresponding stego text, D' that carries B'.

The decoding process is similar to watermark reading, where the bits encoded by the key-selected words in the stego text is read using the keyed hash.

See Section 3.2.4 for an example.

3.2.4 Experiments of MarkErr

The current implementation of MarkErr has two main characterizing inputs: i) the vocabulary of words ii) distortion measure. The vocabulary determines which words are eligible to be considered for message carrying. The vocabulary also determines the words that can be used as replacement words by the watermarking algorithm. The user can limit the choice of words that can be injected to the cover document (i.e., by removing such words from the vocabulary if they are inappropriate). We used the master English word list of Aspell Version 0.50.5 as the vocabulary in our implementation.

We use two different distortion measures to quantify the cost of a unit transformation of the cover text for watermarking and steganography. The distortion measure of steganography is based on a model of typos that have been extracted from AP newswire text in 1988 [79]. It is $-log(P(t|w))$, logarithm of the probability of a particular error given the correct word. A stego embedding with a lower cost should be similar to a human-made typo, hence be less detectable by an adversary.

The watermarking distortion measure should quantify the distortion of the value of the text for human readers. Even though we cannot perfectly capture this as a number, we believe that the similarity of the alternative corrections of a typo may capture the confusion of the readers when faced with a typo and hence approximate the degradation in the value of the cover text. The entropy, $h(\bar{w})$ in step 3.b of the

A substantial portion of the text available online is of a kind that tends to contain **mane** typos and ungrammatical abbreviations, e.g., emails, blogs, forums. It is therefore not surprising **chat**, in **suck** tests, one can **tarry** out information-hiding by the judicious injection of **tyros**. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations that can be **married** out inexpensively, yet reversing them requires much **mere** extensive semantic analyses (easy for humans to carry out, but **hark** to automate).

Fig. 3.1. A sample watermarked version of the first three sentences of Section 3.2. This text is carrying 16 bits, and changed words are shown in bold font.

A substantial **potion** of the text available **onlne** is of a kind that tends to **conain** many typos and ungrammatical abbreviations, e.g., **emals**, **blgs**, forums. It is therefore not **surprisng** that, in **sich** texts, one can **carrsy** out **infomation-hiding** by the judicious injection of **tpyos**. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for **hsort**): Transformations that can be carried out inexpensively, yet reversing them **requirs** much more extensive semantic **analsyes** (**esasy** for **humas** to **carrsy** out, but **harsd** to **uatomate**).

Fig. 3.2. An example of applying proposed steganography techniques on the first three sentences of Section 3.2. This text is carrying 16 bits, and changed words are shown in bold font.

Candidate Selection Algorithm in Section 3.2.2 quantifies this confusion of the user. In this implementation, we did not use context information when computing the entropy of the correction word (i.e., an injection of **hat** to replace **that** is less confusing than when **hat** is used to replace **cat**). A more sophisticated implementation should use side information (e.g., language models, word clusters, part of speech, etc.) to determine words which are commonly used in similar contexts and can be accidentally typed in place of each other.

Figure 3.1 shows a watermarked version of the first three sentences of Section 3.2. We embedded 16 bits into this text. Note that the embedding transformations were performed by using "stealthy" typos where the mark carrying word is a vocabulary word. (Changed words are shown in bold font.)

In our implementation for steganography we used the conspicuous typos as embedding transformations. These typos are generated by applying letter changes to the original words in the cover text which usually yield non-vocabulary tokens in the marked text. Even though these transformations can easily be detected by the adversary, they still have the CAT property as the adversary has to find out the original correct wording of the typographical error among many viable alternatives

in order to revert the transformation. For instance a deletion transformation of "change" into "chage" will force the adversary to choose one of "achage, cage, chafe, Chaga, Chane, change, chape, chare, charge, chase, cha ge, cha-ge, phage" to revert the typo. An example of this embedding is shown in Figure 3.2. The original text is the same as the watermarking example (first three sentences of the abstract) and again we embedded 16 bits into this text.

3.2.5 Related Work for MarkErr

In her 1995 book [81], Turkle published the results of her analysis on the behavior of users in a multi-user game (e.g. Multi-User Dungeon (MUD)) that allows players to chat. She states that onomatopoeic expletives and a relaxed attitude toward sentence fragments and typographic errors suggest that the new writing is somewhere between traditional written and oral communication. This type of language used online is commonly referred as "NetSpeak" [82]. While Turkle's focus is mainly on the psychological effects of the Internet environment, Crystal's work focuses on analyzing the evolution of the language used in chatgroups, emails, and text messages send over mobile phones [83, 84]. Crystal mentions that due to the 160 character limitation in text messages, users tend to shorten many words, and these acronyms or abbreviations stick in a community. Crystal also discusses the ambiguity in NetSpeak in his article [84]. He mentions that there is a two way ambiguity in this language, one way is while interpreting what an acronym stands for, for example "N" can mean "no" and "and", "Y" can mean "why" or "yes". It is up to the receiver to decode a sender's message when it involves an ambiguous acronym, "GBH" can mean "great big hug" or "grievous bodily harm". The other way of ambiguity occurs when shortening a term, for example, "good to see you" can be "GTCY", "GTSY", "G2CY" or "G2SY", and "thanks" can be "THNX", "THX", "TX" or "TNX". Even though usage of acronyms are two-way ambiguous, embedding information through them forms CAT type of a transformation, where

computational complexity of forming an acronym out of a word or a phrase is much more lower than disambiguating the meaning of an acronym.

The studies in spelling error detection and correction research have focused on three problems [52, 85]:

- **Non-word Error Detection** This problem involves finding out whether a word is not in a given dictionary. Several efficient pattern matching and *n-gram* analysis techniques have been proposed for solving this problem, which requires correctly parsing a given word into its stem and its suffix; a fast search capability and a well-designed dictionary. The Unix ®*spell* program is one of the commonly used non-word error detection tools.

- **Isolated-Word Error Correction** Analysis of word typing errors occurring in several applications such as newswire text, search engine queries or optical character recognition data has shown that error rate and error type (e.g. single or multiple character errors) varies from application to application. There have been several techniques designed for detecting and proposing corrections for a misspelled word, such as minimum edit distance technique, rule-based techniques, n-gram based techniques, or probabilistic techniques–such as the one we have used in our experiments [79]. All of these techniques have proven to be successful in a given domain, but an isolated word error correction technique that works efficiently for any given domain has not yet been introduced.

- **Context-Dependent Word Correction** This problem involves dealing with errors where an actual word is substituted for another actual word. This can happen in many forms: due to typos (e.g typing "form" instead of "farm", "lave" instead of "leave"); due to cognitive or phonetic mistakes (e.g. "there" instead of "their", or "ingenuous" instead of "ingenious" or typing "f" instead of "ph"); due to use of wrong function word (e.g. "of" instead of "on"); due to improper spacing (e.g. "my self" instead of "myself"); due to insertion or deletion of whole words (e.g. "I cut myself when while cooking."); due to

grammar errors (e.g. "he come" instead of "he comes") etc. Devising a solution for correcting this type of errors requires strong natural language processing capabilities including the challenging topics of robust natural language parsing, semantic understanding, pragmatic modeling and discourse structure modeling. Only a few spell correction tools attempted to perform context-dependent word correction, and so far none of them have been successful in solving this problem beyond a domain dependent setting that allows only a very restricted type of errors (e.g. at most one misspelled word per sentence, each misspelling is the result of a single point change, and the relative number of errors in the text is known) [86].

As also mentioned in Section 3.2.4, while implementing MarkErr, we used the probabilistic spelling correction technique introduced by Kernighan et al. [79] This technique uses a Bayesian argument that one can often recover the intended correction, c, from a typo, t, by finding the correction that maximizes $\Pr(c)\Pr(t|c)$. $\Pr(c)$ is the word probability learned from a corpus by using the frequency count of a word, and $\Pr(t|c)$ is a model of the noisy channel that accounts for spelling transformations on letter sequences (i.e. insertion, deletion, substitution and reversal). There is one confusion matrix for each spelling transformation. This confusion matrix shows the probabilities of the transformation occurring between the two letters such as $count(sub(t_p, c_p))/count(chars(c_p))$ shows the probability of a character c_p being substituted by t_p. The matrix for $\Pr(t|c)$ is computed using the four confusion matrices computed for each spelling transformation. Kernighan et al. used Associated Press Newswire corpus for training these probability models. Given the typo word "acress", this spelling correction method (when trained on AP newswire corpus) produces the following list where the correction words are sorted according to their scores: {acres (0.45), actress (0.37), across (0.18), access (0.00), caress (0.00), cress (0.00)}. Confusion matrices for all four spelling transformations are provided in [79].

Besides the above mentioned challenges spelling correction for cursory text – similar to spelling correction for search engine queries [86] – has unique challenges such as maintaining a dynamic dictionary which should be updated to include terms emerging in daily life: acronyms (e.g. "asap", "lol"), emoticons (e.g.":-D"), new terms (e.g "blogging", "googling", "phishing", "pwned"), uncommon person names (e.g. "Suri", "Shiloh"), newly generated words for marketing purposes (e.g. a recent movie directed by Gabriele Muccino is titled "The Pursuit of Happyness", one of the popular songs performed by Avril Lavigne is titled "Sk8er Boi"). Such requirements make devising a highly accurate spelling correction tool for cursory text very hard.

Most of the studies in information hiding into natural language text is based on re-writing the cover document using linguistic transformations such as synonym substitution [15, 22], or paraphrasing [10, 16]. T-Lex is one of the first implemented systems that embed hidden information by synonym substitution on a cover document [15, 87]. T-Lex first generates a database of synonyms by picking the words that appear only in the same set of synonym sets from WordNet. The intersections between distinct synonym sets are eliminated to avoid usage of ambiguous words for encoding. This filtering causes the use of uncommon words (e.g. replacing "nothing" with "nada") due to the fact that common words tend to span through several unrelated synonym sets and this property can easily be exploited by steganalysis techniques that use language modeling such as the one introduced in [45].

In [87], Bergmair provides a survey of linguistic steganography. He also discusses the need for an accurate word sense disambiguator for a fully automated synonym substitution based steganography, where sense disambiguation is required both at decoding and encoding time. The lack of accurate disambiguation forces the synonym substitution based information hiding systems to restrict their dictionaries to a subset of words with certain features. Besides decreasing the communication bandwidth, such restrictions cause the systems to favor use of rare words for encoding information [45].

In another work, Bergmair et al. proposes a Human Interactive Proof system which exploits the fact that even though machines can not disambiguate senses of words, humans can do disambiguation highly accurately [58].

3.3 Summary

In this chapter, we first presented and discussed a synonym-based natural language watermarking system, Equmark, that we designed and built. This is the first instance of the use of quantified notions of differences between sentences in natural language information hiding. The use we make of such differences is two-fold. First, we use them to maximize capacity without exceeding the maximum allowable cumulative distortion, and achieve resilience by giving preference to ambiguity-increasing transformations that are harder for the adversary to un-do. Second, we achieve additional resilience by getting close to the maximum allowable cumulative distortion ourselves, as a way of preventing the adversary from carrying out attacking transformations (as these are likely to push the text beyond the allowable distortion limit).

In the second part of this chapter, we have presented a robust information hiding system that is based on the clever use of idiosyncrasies (such as typing errors, use of abbreviations, and acronyms) that are common to cursory text (e.g, e-mails, blogs, forums).

We use computationally asymmetric transformations (CAT), that are computationally inexpensive to perform but hard to revert back (without disproportionately larger computational resources, or human intervention), such as replacing a word with a typo that has a long list of equally possible corrections.

We have designed and implemented two different systems, one for watermarking (robust against an active adversary) and one for steganography (stealthy against a passive adversary).

The language of cursory text is evolving, and getting richer by new acronyms (e.g, "lol") or new words (e.g., "phishing") added daily to the language by repeated usage in many online communities. There is much room for improvement in information hiding in cursory text. Typing error is only one type of idiosyncrasy that opens room for information hiding in cursory text, and new opportunities develop as the language develops.

4. SENTENCE LEVEL NATURAL LANGUAGE WATERMARKING: ENIGMARK

In this chapter, we propose a rather generic information hiding algorithm, Enigmark, for embedding messages into natural language text at the sentence level, where the carrier medium and the adversary model presents unique challenges. We provide a highly flexible system where (i) the watermark reading process is freed from using the exact same statistical Natural Language Processing (NLP) tools that were used while the watermark was being embedded, (ii) the watermark detection requirements are adjusted to be able to stand a given amount of attacks (i.e., embedding threshold is higher than detecting threshold) (iii) the complex and rich feature set of sentences are exploited to increase the bandwidth and robustness.

Compared to other media, natural language text presents unique challenges for information hiding. These challenges require the design of a robust algorithm that can work under following constraints: (i) low embedding bandwidth, i.e., number of sentences is comparable with message length, (ii) not all transformations can be applied to a given sentence (iii) the number of alternative forms for a sentence is relatively small, a limitation governed by the grammar and vocabulary of the natural language, as well as the requirement to preserve the style and fluency of the document. The adversary can carry out all the transformations used for embedding to remove the embedded message. In addition, the adversary can also permute the sentences, select and use a subset of sentences, and insert new sentences. In this chapter, we propose a scheme that overcomes these challenges, together with a partial implementation and its evaluation for the English language.

The present application of this scheme works at the sentence level while also using a word-level watermarking technique that was recently designed and built into a fully automatic system ("Equmark") introduced in Chapter 3. Unlike Equmark,

whose resilience relied on the introduction of ambiguities, Enigmark is more tuned to situations where very little change to the text is allowable (i.e., when style is important). Secondarily, this work shows how to use lower-level (in this case word-level) marking to improve the resilience and embedding properties of higher level (in this case sentence level) schemes. We achieve this by using the word-based methods as a separate channel from the sentence-based methods, thereby improving the results of either one alone. The sentence level watermarking technique we introduce is novel and powerful, as it relies on multiple features of each sentence and exploits the notion of orthogonality between features.

4.1 Algorithm for Multiple Feature Based Information Hiding

In this section we describe an algorithm that can be used to hide information into any data as long as it has multiple features. Although the algorithm is presented for the specific case of natural language text, it can potentially be used in other domains.

Let D be a natural language text document consisting of n sentences $d_1, \ldots, d_n$. Let F be a set of Boolean returning functions on sentences, where each such function indicates the presence (or lack thereof) of a particular property in a sentence, e.g., an $f_i(d_j)$ could be an indicator of whether the sentence d_j is passive or active, or whether it contains two nouns, or whether it contains a particular class of words, or whose hash is a least significant bit of 1, etc. We call each such function f_i a feature function.

Let T be the set of transformations that are available for modifying the sentences (e.g., synonym substitution, passivization). For each $t \in T$, $t(d_i)$ denotes the outcome of applying that particular transformation to d_i (the "transformed" version of d_i). We use $\delta_1(t(d_i), d_i)$ to denote the amount of distortion that d_i undergoes as a result of using transformation t on it. Likewise $\delta_2(t(d_i), d_i)$ denotes the expected

distortion that the adversary will cause after modifying $t(d_i)$ without the knowledge of d_i.

We henceforth use M to denote the message to be embedded.

Our algorithm for information hiding into natural language text works under the following demanding constraints:

- The number of sentences n can be small, i.e., comparable with message length $|M|$; contrast this with the earlier work in [9, 10] where n needed to be much larger than $length(M)$.

- Relatively few transformations (if any) could be applicable to a given d_i, e.g., it is not possible to passivize a sentence with an intransitive verb ("I run every morning").

- A sentence may be transformable into a relatively small number of alternative forms, as there may only be a small number of transformations applicable to it. This limitation is governed by the grammar and vocabulary of the natural language (e.g. small number of synonyms, small number of paraphrases, rigid word ordering).

- The adversary can permute sentences, select a subset of the sentences, and insert new sentences. The resilience we achieve can handle arbitrary permuting, and extensive but not massive subset selection (e.g., selecting zero sentences) and insertion (e.g., hugely many new sentences). More on how this resilience is quantified will be given in the experimental section.

The feature functions in F serve two distinct purposes: (i) some of them serve as indicators of the presence of a mark (we will generically denote functions used for this selection purpose with f_s); (ii) others will be used to actually help encode the bit(s) of M that are embedded in a sentence (we will generically denote functions used for this embedding purpose with f_e). We said "help encode" because the $f_e(d_i)$ need *not* necessarily agree with the bit(s) of M that d_i is helping encode: The relevant bit

of M is encoded in the aggregate distribution properties of all such sentences that encode that bit (more on this later); a similar technique of using aggregate properties for encoding was done in [76], although in our case the sentence subsets that encode different bits can overlap which helps increase both capacity and resilience (in [76] these subsets were disjoint – no two items contributed to more than 1 bit of M).

In this framework, the process of embedding 1 bit, consists of transforming a number of sentences d_i so that their $f_s(d_i) = 1$ (i.e., they are selected for embedding), and making their $f_e(d_i)$ collectively encode the appropriate bits of M by deviating significantly from expected distribution of $f_e(d_i) = 1$. Embedding transformations either set $f_s(d_i) = 0$ to de-select a data unit, or set both $f_s(d_i) = 1, f_e(d_i) = 1$. The detection of this bit, consists of finding out whether the aforementioned statistical deviation holds.

It is important to have the flexibility to unmark a data unit, since in many occasions a transformation t will not be able to yield $f_e(t(d_i)) = 1$.

The f_s and f_e need to be defined on an indivisible data unit, such as a word, a phrase or a sentence, depending on the adversary model of a particular information hiding application. For example one model could assume that the adversary cannot divide a sentence into two sentences.

The embedding process will be subject to a maximum allowed distortion threshold $\sum_{d_i \in D} \delta_1(d'_i, d_i) \leq \hat{\Delta}_1$, where d'_i is a message carrying sentence derived from d_i. $\hat{\Delta}_1$ captures the tolerable loss in value of D (in case of watermarking) or the loss of stealthiness of covert channel (in case of steganography). In case of watermarking the embedding process also aims to maximize $\sum_{d_i \in D} E[\delta_2(d'_i, d_i)]$, which captures the expected distortion that the adversary will cause while attempting to remove the embedded message from d'_i.

Encoding Algorithm

Let M be a message $(m_1 \dots m_w)$

Let $D[]$ be a document of n sentences, $d_i = D[i]$

Let K be a secret key

Let $T[i]$ be the set of transformations applicable to d_i

Let F be a set of boolean "feature" functions on D ($f(d_i)$
returns a 1 if d_i contains feature $f \in F$)

Let $F_s \subset F$ be the subset of message-presence
indicator functions

Let $F_e \subset F$ be the subset of message-embedding
indicator functions

Let $\hat{\Delta}_1$ be the maximum allowable distortion

Let $C[i]$ be the subset of message bits that d_i
contributes to encoding

Let $\textsc{Gain}(d_j, d_k) \leftarrow \frac{E[\delta_2(d_j,d_k)]}{\delta_1(d_j,d_k)}$(intuitively, this is the
"resilience gained", the distortion caused by the adversary
per unit of distortion caused by the embedding)

Let $\textsc{BitSuccess}(f_s^i, f_e^i, D)$ return 0 if the i^{th} message bit m_i
was not successfully encoded in the (modified) D using f_s^i
and f_e^i; otherwise it returns a positive number that
measures the statistical significance of the existence of m_i
in D (the "strength" of the signal using, e.g., χ^2 or Fisher's Exact Test)

for each $l = 1, 2, \ldots, |M|$
Use K, l, m_l as seeds to randomly select from F an f_s^l
and an f_e^l, that are different features

$D^0[] \leftarrow D[]$

$\Delta_1 \leftarrow 0$

while $T \neq \emptyset$
For the bit l that is in most need for improvement of
embedding signal (In other words, the bit that has the
lowest χ^2 score. This can also be checked
using "odds ratio" to compare the statistical significance of
deviation from the normal for each bit), try to help it as
follows:
For each sentence d_j, choose the transformation

$t_{l,j} \in T[j]$ that helps the encoding (the χ^2) of that

bit l while maximizing GAIN$(t(d_j), d_j^0)$

(i.e., maximizing resilience).

Among all such pairs $(d_j, t_{l,j})$ choose the one that

has highest GAIN$(t(d_j), d_j^0)$, call that pair (d_i, t).

$d_i' \leftarrow t(d_i)$

$\Delta_1' \leftarrow \Delta_1 - \delta_1(d_i^0, d_i) + \delta_1(d_i^0, d_i')$

if$(\Delta_1' > \hat{\Delta}_1)$

$T[i] \leftarrow T[i] - t$

if$(T[i] = \emptyset)$

$T \leftarrow T - T[i]$

continue

$\Delta_1 \leftarrow \Delta_1'$

$d_i \leftarrow d_i'$

$C[i] \leftarrow C[i] \cup \{l\}$

Update $T[i]$ such that it includes only the transformations

that would improve the current strength of all the

message bits in $C[i]$.

if any of the $|M|$ bits was not successfully encoded, i.e., if some

BitSuccess(f_s^l, f_e^l, D) = False

return False

return D

Note that the above algorithm continues to perform transformations until the maximum distortion $\hat{\Delta}_1$ is reached, even after the message is successfully embedded. This is necessary to limit the flexibility of adversary.

Some of the modifications that the adversary performs on a sentence will not change the contribution of the sentence to the embedded message. We do not leverage on such difference among modifications of the adversary. The current scheme simply tries to maximize the number of alternative sentences that the message is embedded

in order to maximize its resilience against adversaries' modifications. An improved scheme should prefer to embed in those sentences which have a higher likelihood to carry the same embedded message even after the attackers' modifications.

We now describe the decoding algorithm, which reads an embedded message from a document that has undergone a message embedding. This algorithm does not, require the message M to be available. If M is available, the algorithm can be modified to use the BitSuccess for quantifying the confidence that the cover document D carries M. We require that M carries an error correction code, and it is possible to detect the termination of M when M is received as a growing string.

Decoding Algorithm

Let M, D, K, F be defined as above

Let MaxMessage be the largest message size

Let Terminated(M) be a boolean function that decodes the partial message M and returns True if the message has terminated

for each $i = 1 \ldots$MaxMessage

Use K, l, 0 as seeds to randomly select from F an f_s^0 and an f_e^0

Use K, l, 1 as seeds to randomly select from F an f_s^1 and an f_e^1

if(BitSuccess(f_s^0, f_e^0, D) > BitSuccess(f_s^1, f_e^1, D))

$M[i] \leftarrow 0$

else

$M[i] \leftarrow 1$

if (Terminated(M))

break

return M

The algorithms that we have given in this section can be used to embed messages into any kind of collections of data units, where we are under similar constraints as

natural language text but at the same time have a flexibly large number of features and a limited number of transformations in the arsenal of information hiding.

4.2 Sentence Level Watermarking

We distinguish two types of modifications that can be used for watermarking text: The robust synonym substitution introduced in [12], and syntactic sentence-paraphrasing. Compared to naive synonym-substitution, robust synonym substitution introduces ambiguities in order to make it harder for the modification to be undone. Such modifications can somewhat damage the precision of the individual words used in the text, e.g., replacing " a slope in the turn of the road" with "bank". Sentence-level paraphrasing, on the other hand, typically does little damage to the precision of words, but may damage the stylistic value of the sentence. An example that points out to a possible one-wayness of sentence-level watermarking happens when the original sentence is "April had to hold a party", and gets transformed into "A party must be held by April". It will be hard for an adversary to undo this embedding without performing a co-reference resolution and context analysis on the full text. See Section 4.3.1 for several examples of sentence-level paraphrasing.

Depending on the type of the text (e.g., multimedia content, editorials, news reports, user manuals, etc), the requirements for the preservation of precision and the preservation of style can both vary. In user manuals, style requirements are less stringent, whereas precision cannot be compromised. Style is a more important value of editorials, whereas precision is more important in newswire. A text that accompanies video, audio or pictures as a secondary information resource, may have less stringent requirements on both style and precision. In addition to the above-mentioned differences between this work and [12], another difference is that whereas [12] focuses on precision, in this work we investigate a method that can be used to trade precision for style.

4.2.1 Selection of Sentences

As stated earlier, the sentence features used for selection are different and orthogonal to those used for embedding the message bit(s). We next discuss two alternatives for sentence-selection (the embedding of message bits is covered in the next section).

A subset of the vocabulary is pre-selected as mark-carrying (that subset is not known to the adversary). The message bits are inserted only in those sentences that contain a word from that subset. Of course this means that some inputs will not contain enough words from that special subset, and hence will be deficient in terms of their "markability". To avoid such situations, the selected vocabulary subset is chosen using a language model for the specific domain, to insure that long enough sentences from this domain will usually not be so deficient; a language model for financial analysis texts will be different from a language model for Jack London's works, and the vocabulary-subset for the former will be very different from the latter's.

Alternatively to the above language-model based approach to select mark-carrying sentences, synonym substitution can be used to flexibly and adaptively mark sentences. In such a case, mark words are added to the text by replacing their synonyms in the original text. The slight shift in meaning due to synonym-replacement can be viewed as a robustness advantage: The adversary trying to do it wholesale will degrade the value of the work beyond desired limits.

4.2.2 Embedding

We now assume that the sentence at hand is selected for message-bits insertion (possibly using one of the two methods described earlier).

As the features for embedding are orthogonal to those for selection, we can carry out the embedding without changing the "selected" status of a sentence. The way embedding is done by modifying the embedding features until they "speak the desired

“the democratic party has denied the allegations”

Fig. 4.1. A sample sentence taken from the Reuters Corpus. Its publication day is 8th of January 1997.

message bits”. In the framework that is described here we distinguished between selection and embedding features. The selection features are determined by Equmark, where a sentence that has a word from the selected subset of the vocabulary is an information-carrier, and the embedding features are based on sentence-level linguistic features which can be “number of prepositions in a sentence”, “a sentence being passive or active”, “distance of certain functional words”, or “the verb classes [56] of the verbs in a sentence”.

The task of creating a statistically significant deviation in the distribution of embedding features in a selected set of sentences is not independent from the features that are used for selection and embedding. This distribution is based on the correlation between selection features and embedding features. For example, Sigmund Freud has a tendency of using double-negation(e.g. “this is not insignificant”), and if we were to watermark a text that heavily quotes from him, and if the words that are related to psychological research are used as selection features (“mark-carrying” words), the embedding feature of “sentence carrying double-negation” will be correlated with these selection features.

4.3 System Implementation and Experiments

The purpose of our experiments is not to stress-test the embedding capacity of our scheme, rather, it is to demonstrate the possibility of applying it on a real-life test case (Reuters [88] is a common benchmark used in NLP research). Therefore the reported embedding rates are not indicative of the potential of our proposed scheme, because what we implemented is only a partial system that uses (i) a small fraction of the available repertoire of transformations (only two of them), and (ii)

```
( S_r ( NP_r ( D the )
             ( NP_f ( N_r ( A democratic )
                          ( N_f party ) ) ) )
      ( VP_r ( V has )
             ( VP ( V denied )
                  ( NP_r ( D the )
                         ( NP_f ( N allegations ) ) ) ) ) )
```

Fig. 4.2. Syntactically parsed sentence, output of XTAG Parser on the sentence given in Figure 4.1

```
( alphanx0Vnx1[denied] ( alphaNXN[party]<NP_0>
                         betaAn[democratic]<N>
                         betaDnx[the]<NP> )
                       ( alphaNXN[allegations]<NP_1>
                         betaDnx[the]<NP> )
                       betaVvx[has]<VP> )
```

Fig. 4.3. Sentence Dependency Structure, output of XTAG. See Figure 4.4 for a depiction of this tree.

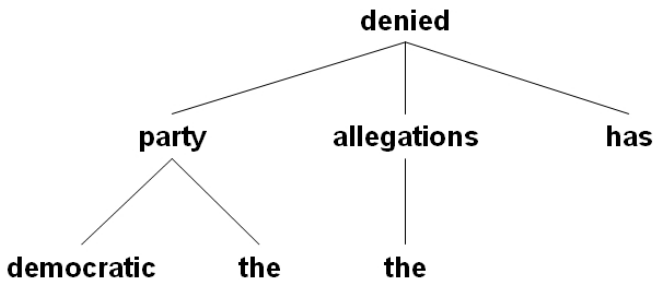


Fig. 4.4. Depiction of the dependency tree in Figure 4.4 for the sentence in Figure 4.1.

```
( alphanx0Vnx1[denied] ( betanxPnx[by]<NP_r> ( alphaNXN[party]<NP_1>
                                                betaAn[democratic]<N>
                                                betaDnx[the]<NP> ) )
                        ( alphaNXN[allegations]<NP_0>
                          betaDnx[the]<NP>)
                        betaVvx[has]<VP> )
```

Fig. 4.5. Sentence dependency structure for the watermark carrying sentence in Figure 4.8 generated by passivization process.

```
DSYNTS:
deny[ class:verb voice:act mood:past-part case:obj  taxis:perf
                                                    tense:pres ]
(   I by[  ]
    (   II party[ class:common_noun article:no-art case:nom
                                          person:3rd number:sg  ]
        (   ATTR democratic[ class:adjective  ]
            ATTR the[ class:article  ]        ) )
    I allegation[ class:common_noun article:no-art case:nom
                                          person:3rd number:pl  ]
    (   ATTR the[ class:article  ]         ) )
END:
```

Fig. 4.6. Partial DSyntS format for the watermark carrying sentence shown in 4.8. This is generated by the result of using the original sentence's XTAG parse output and the dependency tree generated by the passivization transformation(shown in Figure 4.5).

```
DSYNTS:
deny[ class:verb voice:pass mood:past-part case:obj  taxis:perf
                                                     tense:pres  ]
(   II by[  ]
    (   II party[ class:common_noun article:no-art case:nom
                                          person:3rd number:sg  ]
        (   ATTR democratic[ class:adjective  ]
            ATTR the[ class:article  ]             ))
    I allegation[ class:common_noun article:no-art case:nom
                                          person:3rd number:pl  ]
    (   ATTR the[ class:article  ]                  ))
END:
```

Fig. 4.7. Final DSyntS format for the watermark carrying sentence shown in 4.8, generated by the passivization process when the dsynts file in Figure 4.6 is given.

"the allegations have been denied by the democratic party"

Fig. 4.8. Watermarked version of the sample sentence in Figure 4.1. In this example, passivization is used for watermarking.

```
DSYNTS:
deny[ class:verb voice:act mood:past-part case:obj  taxis:perf
                                                    tense:pres ]
(  I party[ class:common_noun article:no-art case:nom person:3rd
                                                      number:sg  ]
    (  ATTR democratic[ class:adjective  ]
       ATTR the[ class:article  ]   )
   II allegation[ class:common_noun article:no-art case:nom
                                           person:3rd number:pl  ]
   (  ATTR the[ class:article  ]    )
 )

END:
```

Fig. 4.9. The DSyntS format generated for the sentence in Figure 4.1, if it was directly processed by conversion process without any transformation process' interference.

the specific implementation of these transformations has a very restrictive domain of input sentences to which they apply (for ease of implementation).

The approach described in this chapter is based on syntactically modifying the sentence structure. In order to be able to automatically manipulate the sentences, we first derive a structural representation of the sentences (through parsing [52]) and later revert this representation into surface sentence form (through generation [52]).

The output of the parsing may represent either the morphological, syntactical, or semantical structure of the sentence or it may represent a combination of these. Figures 4.1, 4.2 and 4.3 show a sentence in surface form, its syntactic parse tree and derivation tree (dependency tree) obtained using XTAG parser. We can use output of XTAG parser to find out features of sentences [11, 29] such as voice, question, superlative etc. Refer to [11] or [29] for the list of features that are evaluated for

each node of the syntactic parse tree generated by XTAG. For example, given a sentence, before trying to passivize it, we verify the value of "<passive>" feature being marked as "−" (i.e. negative) for the main verb, together with the label of this verb in the XTAG derivation tree showing that it is transitive (XTAG parser uses "nx0Vnx1" to denote transitive verbs in derivation (dependency) tree outputs, which can be observed in Figure 4.3).

Our transformations use both the parse tree and the derivation tree in order to perform embedding transformations.

We transform a sentence that has been selected for watermark embedding as follows:

1. Parse the sentence by XTAG parser.

2. Verify if the sentence already carries the embedding feature. If so return, else go to next step.

3. For each available transformation;

   (a) Verify if the transformation is applicable to the sentence (e.g. for passivization, the root of the syntactic tree has to be a transitive verb). Refer to Figure 4.1, Figure 4.2 and Figure 4.3 for a sample of information used at this step.

   (b) Embedding operation is performed in two steps:

      i. Re-write the dependency tree based on the design of the transformation. Refer to Figure 4.5 to see a transformed dependency tree generated during passivization. Note that "by" is added and made the parent of the subtree that has the subject of original sentence.

      ii. Convert the modified XTAG output into a deep syntactic structure (in DSyntS format) that reflects the "transformed" features of the sentence. Refer to Figure 4.7 for the deep syntactic structure rep-

Table 4.1
The cumulative evaluation of performance of the presented system on direct conversion from English back into English sentences. 1804 sentences (length ranging from 2 to 15 words) from Reuters corpus are used.

			Cumulative N-gram scoring		
	1-gram	2-gram	3-gram	4-gram	5-gram
NIST:	7.7169	9.7635	10.0716	10.1172	**10.1269**
BLEU:	0.8548	0.6768	0.5580	**0.4705**	0.4030

resentation of the sentence in Figure 4.1 after going through a passivization transformation.

(c) Use RealPro to convert the resulting deep syntactic structure into surface sentence form. Figure 4.8 shows the result of realization for our example case.

(d) Verify if the transformed sentence carries the embedding feature. If so, record the distortion value.

4. Commit the embedding transformation that imposes minimum distortion.

Comparing Figure 4.7 and Figure 4.9 will show the main idea behind the design of passivizing transformation implemented for this framework.

Data Resources We tested our system on 1804 sentences from the Reuters corpus [88]. We picked eleven publication days at random[1]. Later, from the articles that were published on these days, we picked the first 1804 sentences that are parsed with the XTAG parser. We are also using Wordnet [28] as a data resource for converting plural nouns to singular forms, and verbs into their base forms. This conversion is required for complying with the requirements of DSyntS.

[1]24th of August 1996, 20th of October 1996, 19th of August 1997 and 8 consecutive days from 1st of January 1997 to 8th of January 1997

Parsers Our implementation uses *XTAG* parser [2] [29] for parsing, dependency tree generation (which is called a derivation tree in the XTAG jargon) and linguistic feature extraction.

Generator We used *RealPro*[3] [8] for natural language generation.

Refer to Figure 4.10 for the depiction of the currently tested baseline system. Table 4.1 shows an evaluation of this system without the watermarking step. As explained in Section 4.4, these scores are generated by systems that were specifically designed for evaluating machine translation systems, and they do not perfectly capture semantic resemblance of two sentences. 23% of the 1804 sentences were identically re-generated.

We would like to emphasize that the current system is limited by the capabilities of the parser and the surface realizer. XTAG may not be able to analyze a given sentence into its structural representation. Even though the XTAG parser is very powerful, it is not 100% accurate. Moreover, it has a limited coverage of vocabulary, and adding new words to its dictionary is not trivial, because every word in its dictionary is represented with several tree structures that conform to its usage in the language grammar. RealPro may not be able to generate an expected realization of a given deep syntactic structure in DSyntS format. RealPro is not designed for English to English translation, hence it has limitations when used for this purpose. For instance it can only handle a subset of uses of punctuation. Refer to RealPro General English Grammar User Manual [89] for further details on the capabilities and shortcomings of RealPro. A natural language watermarking system that has overcome these limitations will have more coverage while selecting sentences and performing embedding transformations on them. Therefore as the NLP systems improve, Enigmark will get more resilient and will provide higher bandwidth.

[2]Available at http://www.cis.upenn.edu/ xtag/swrelease.html. In our experiments, we used *lem-0.14.0.i686.tgz*

[3]See http://www.cogentex.com/technology/realpro/ for access to RealPro.

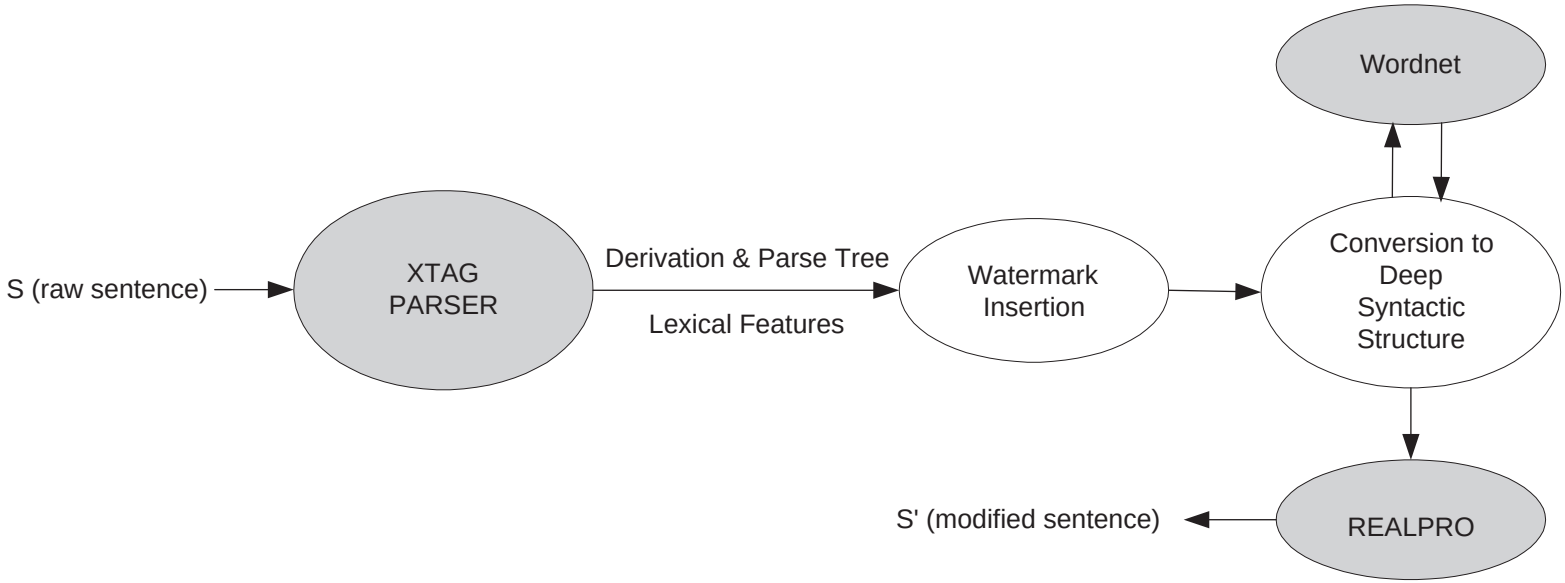


Fig. 4.10. A schema of the system that is being developed and tested for the baseline evaluations of the proposed sentence-level natural language watermarking system. This implementation extracts features, generates parse and derivation trees with *XTAG* parser and uses *RealPro* for surface realization.

Table 4.2
Review of linguistics transformation success on the dataset of 1804 sentences from Reuters corpus.

	Applicable sentences	Successfully transformed sentences
Passivization:	54	20
Activization:	24	11

4.3.1 Sentence Level Linguistic Transformations

We have implemented specific transformation algorithms for two linguistic transformations: "activization" and "passivization". Their success rate is listed in Table 4.2. The sentences that we marked as successfully transformed are the ones that are grammatical. But there were cases, where even though the sentence is grammatical and the transformation process worked as expected, the result was destroying the meaning of the original sentence. One example is as follows:

- Original : presidential elections must be held by october
- Transformed : october had to hold presidential elections

In addition to these two transformations, if a sentence is analyzed using XTAG and then RealPro is used to generate a surface sentence from this analysis, this process may generate an output sentence that differs from the original sentence. In cases where such output sentences are grammatical, we observe that these sentences have gone through some syntactic transformations. Two of the transformations that occur consistently are special versions of "adjunct movement" and "topicalization".

In [17], Murphy provides a detailed insight into applicability and coverage of several sentence level transformations for information hiding purposes. He shows that conjunct movement is the most applicable transformation within the list of

transformations he tested. Conjunct movement is applicable to 1 sentence in every 15 sentences, and 87% of the time the transformation rule for conjunct movement results in grammatical and meaningful sentences. See Table 7.1 in Chapter 7 for the results presented in [17]. In our experiments, we have evaluated the ubiquity (i.e., coverage) of passivization to be 1/33 and reliability (i.e., applicability) of our implementation of passivization to be 37%; the coverage of activization was 1/75 and applicability of our implementation of this transformation was 45%.

Examples for the aforementioned transformations are given below, these sentences are taken from the data set introduced above, which is a subset of Reuters Corpus [88]:

Adjunct Movement

Original: now they are just attractive

Transformed: they are now just attractive

Passivization (x 2)

Original: this frank discussion will close this chapter

Transformed:

(i)this chapter, by this frank discussion, will be closed

(ii)this chapter will be closed by this frank discussion

Activization

Original: the devices were disrupted safely by the
washington bomb squad

Transformed: the washington bomb squad safely disrupted
the devices

Topicalization

Original: doctors said he was alive but in critical condition

Transformed: he was alive but in critical condition
doctors said

An example of two transformations that can be performed on the same sentence is as below:

Original

he said canada and britain recently rejected the idea

After passivization

he said the idea was recently rejected by canada and britain

After adjunct movement

he said the idea was rejected recently by canada and britain

4.3.2 Resilience Discussion

The reader may have observed that the transformations we use are typically reversible, i.e., the adversary can apply them wholesale everywhere. There two answers to this.

The first is that wholesale application of transformations (so as to "flatten" everything) has serious drawbacks: It is computationally expensive, it significantly changes the style of a document, and ambiguity can make it hard to automatically carry out (e.g., "A party must be held by April"). When embedding, we do not suffer the ambiguity drawback (because the initial "April had to hold a party" was not ambiguous), nor do we apply the process wholesale (we use the secret key to choose where to selectively apply it).

The second point is that the resilience of our scheme does not hinge on the non-reversibility of the transformation (e.g., passivization is easily reversible), rather, it relies on the fact that the adversary does not know the key-selected embedding features: The transformation is usually reversible in a multiplicity of ways (even if by trivial adjunct-movement), and the adversary does not know the impact of each of these ways on the secret embedding features (one of them neatly un-does the embedding action, but the adversary does not know which one). When a transformation is reversible in a unique way, we can either introduce multiplicity (e.g., by doing non-embedding transformations combined with the uniquely reversible embedding one), or we can combine the uniquely reversible embedding transformation with the robust

synonym-substitution mechanism of [12] or with judicious (and ambiguity-increasing) removal of repeated information (a special and tractable case of co-references).

4.4 Evaluation of Natural Language Watermarking

The goal of digital watermarking is to provide copyright protection for digital documents; achieving this goal requires watermarking techniques to embed watermark message in a discreet manner (i.e. without damaging the value of the copyrighted material) while being court-provable (i.e. providing high capacity embedding) and robust against malicious attacks, channel noise or distortion.

The value of a watermarking system lies in its success in fulfilling these requirements. Building techniques for evaluation of watermarking systems is one of the main branches of watermarking research. There are several benchmarking systems available for image and audio; such as Stirmark [90] and Audio Watermark Evaluation Testbed [91]. Refer to [92] for an analysis of automatic benchmarking systems for images.

4.4.1 Evaluation of Perceptibility

There are two types of methods for evaluating the perceptibility of watermarking systems: *subjective* evaluation and *objective* evaluation. *Subjective* evaluation consists of asking the actual users (human beings) evaluate the quality of watermarked document compared to the original document. Performing subjective tests are very costly and time consuming. For this reason, in this section, we will focus on analyzing the *objective* evaluation methods for evaluating the perceptibility of natural language watermarking systems.

Objective evaluation methods are based on statistically comparing the similarity of watermarked and original objects. Mean Square Error (MSE) or peak-Signal-to-Noise-Ratio (PSNR) are commonly used metrics for evaluating the quality of image watermarking [92]). If a watermark image has a low MSR when compared to

the original image, it is accepted that, with high probability, the perceptibility of the watermarking will also be low. Similarly, Objective Difference Grade (ODG) is widely used for evaluating audio watermarking [91].

We have to point out the fact that while being very cost effective, the objective evaluation methods suffer from not being able to capture the user model perfectly. For example, even though users can not notice the difference when the audio document is lengthened or shortened while the pitch is kept the same, ODG metrics report that the modified audio is significantly different from the original. We will discuss more about a similar evaluation pitfall in natural language watermarking later in this section.

Evaluation of Machine Translation output has many similarities to the evaluation of Digital Watermarking. Machine Translation (MT) aims to provide a high quality translation (in the target language) of the input text (in the original language). The quality criteria for machine translation is the similarity to a reference translation generated by a human translator using the same input text. There are many studies on improving the objective evaluation methods for MT quality, for a survey of previous research on MT evaluation refer to [93].

NL watermarking bears a close resemblance to the machine translation task of NL processing. Rather than converting sentences from one language to another, their style and other properties are modified in a single language in order to embed information.

One way of using MT evaluation systems for evaluating perceptibility of natural language watermarking is to check the success of a natural language watermarking system in *re-generating* a given sentence as close to the original as possible. The results of this test shows the *coverage* of a natural language watermarking system; defined as how applicable it is to various sentences. In a sense the coverage of a natural language watermarking system shows the success of the system in being able to process any given input sentence and convert it back to surface level preserving

its meaning and form (e.g. voice, mood, tense); hence, showing the success of the system in being able to output any sentence.

In [11], Topkara et al. used the MT Evaluation Tool Kit [4] of NIST (available at [94]) to evaluate the coverage of a sentence level watermarking system. This toolkit outputs scores for BLEU (BiLingual Evaluation Understudy) metric [95] and NIST metric [96].

BLEU computes the geometric mean of the variable length phrase matches (precision) against reference translations. The BLEU metric ranges from 0 to 1. Only the translations that are identical to a reference translation will attain 1. BLEU measures translation accuracy according to the phrase matches with one or more high quality reference translations. BLEU has been found to generally rank systems in the same order as human assessments.

In the same year with BLEU, in 2002, the NIST metric was introduced [96]. The NIST metric is a modified version of BLEU where the arithmetic mean of information weight of the variable length phrase matches are used, instead of arithmetic mean of N-gram precisions. For previous research on MT evaluation refer to [93].

Both BLEU and NIST metrics are sensitive to the number of reference translations. The more reference translations per sentence there are, the higher the BLEU and NIST score are. Papineni et al. states in [95] that on a test corpus of about 500 sentences (40 general news stories), a human translator scored 0.3468 against four references and scored 0.2571 against two references. However, there is only one reference translation, the original sentence, available for the evaluation of natural language watermarking system's re-generation success hence their coverage.

Using just BLEU for sentence by sentence distance evaluation is not enough and accurate for the task of evaluating natural language watermarking. BLEU is very sensitive to precision in words and their position in the generated sentence. Some of the transformations (e.g. passivization) change the word order heavily while keeping

[4] mteval-v11b.pl, release date: May 20th, 2004. Usually length of phrases range between unigram to $4gram$ for BLEU metric and unigram to $5gram$ for NIST metric. In the tables presented here the range is between 1 to 9.

the meaning very close to original. Better way of evaluating the distortion made by a natural language watermarking system is also measuring the distortion at the full text level. Such an evaluation can be done in several ways: (i) by counting the number of sentences changed, (ii) by assigning weights to different types of changes (i.e. transformations) to indicate the amount of the distortion they impose on the sentences, for example verb particle movement can get higher weight than removal of double-negation, (iii) by generating a language model of the author and measuring the change in the probability of a watermark carrying sentence (iv) using summarization to detect the change in similarity between the original document and watermarked document.

Preserving the language characteristics (i.e. language model) of a document depends on being able to automatically evaluate the characteristics of a writer's style. Length of sentences and paragraphs, or usage of clauses or percentage of the passive sentences can be counted as style characteristics. See [6] for a discussion of using style and expression for identifying linguistic similarity. If the author's characteristics can be quantified (e.g. percentage of passive sentences, or the histogram of word frequencies), it is possible to perform on-the-fly damage control system in order to minimize the deviation from the expected characteristics' value while embedding information. In a previous work [38], we presented a new protocol that works in conjunction with the information hiding algorithm to systematically improve the stealthiness.

It is important to note here that natural language watermarking has an advantage of being able to take the consent of the author of the original document, since it is easy to prompt the author at the time of watermarking (or as a post-process after watermarking) to ask whether the watermarked sentence generated by the system is good quality or which one of the several alternatives should be used. This is more feasible for natural language watermarking than signal-based data watermarking, since in signal-based domains it is hard for the owner of the file (a human being) to

differentiate between possible alternatives for watermark carrying portions of image (i.e. pixels), or video, or audio document.

4.4.2 Evaluation of Robustness

Model of adversary is very important in designing an evaluation mechanism for robustness. In natural language watermarking area, it is assumed that the adversary has the same processing power and data resources with the watermark embedder. However, the adversary does not have access to the original document. Having the original document provides a big advantage to the watermark embedding tools due to the fact that the document carries more information (e.g. context [12], original language [97], exact characteristics of the author etc.). For example, the richness of context information increases the accuracy of word sense disambiguation which provides an advantage in synonym substitution based information hiding systems [12, 15].

In [45], Taskiran et al. has used a statistical attack for steganalysis of Tyrannosaurus Lex (T-Lex) system proposed by Winstein. Their approach relies on the fact that the text manipulations performed by the lexical steganography system change the properties of the text by introducing language usage that deviates from the expected characteristics of the cover text. Their technique proceeds as follows: First, they capture cover-text and stego-text patterns by training language models on unmodified and steganographically modified text. Second, they train a support vector machine (SVM) classifier based on the statistical output obtained from the language models. After this, trained SVM is used to classify a given text as unmodified or steganographically modified based on the output. The accuracy on steganographically modified sentences was found to be 84.9%.

Same statistical approach can be used to measure the robustness of other natural language watermarking and linguistic steganography systems as well. However, as they have mentioned in [45], this technique is heavily based on the presence of a

finite lexicon, its performance may suffer for steganalysis applications to inflectional and compounding languages such as German, Finnish and Turkish.

Another previously used evaluation method is probabilistic evaluation. Atallah et al. quantified the robustness of their natural language watermarking schemes probabilistically in [9].

Besides the above techniques, being able to benefit from the supervision of the author is a big advantage against many statistical attacks. Given many alternatives the author himself would pick the watermark carrying version to be the one that resembles most to his style, hence preserve the characteristics of the document against a statistical attack.

4.4.3 Evaluation of Capacity

Capacity evaluation can be done at different levels in natural language watermarking, such as evaluating the capacity of a word (how many bits can we insert in a word, if we are performing synonym substitution), or capacity of a sentence, or capacity of full text.

Capacity of a natural language watermarking system depends on its coverage. As also mentioned in Section 4.1 and in Section 4.4.1, in natural language watermarking not every embedding operation can be performed on any given word or sentence. For example, if a word does not have synonym, it is not possible to use that word as an information carrier in a synonym substitution based system. However, if a system is capable of changing any given word or sentence, every word or sentence will become a potential information carrier. The capacity improves even more, when it possible to perform orthogonal (i.e. independent) transformations on one information carrier.

Number of bits that can be carried by one word or one sentence depends on the number of synonyms (words or sentences with the same meaning) a word or sentence has. Take the word “go” as an example, one sense of “go” has 17 synonyms (i.e. can carry 4 bits), on the other side the verb “lie” has at most 4 synonyms(i.e.

can carry 2 bits) in one of its senses. A similar phenomenon happens in sentences. In the below example there are at least four different transformation which can be performed independently on the original sentence:

Original

he said canada and britain recently rejected the idea

After passivization

he said the idea was recently rejected by canada and britain

After adjunct movement

he said the idea was rejected recently by canada and britain

After conjunction argument switching

he said the idea was rejected recently by britain and canada

After topicalization

the idea was rejected recently by britain and canada, he said

However, it is not possible to perform a well-known meaning preserving linguistic transformation (except synonym substitution) on the sentence, "I run every morning".

According to the use of watermark the capacity requirements changes, also in watermarking the genre of the text that is being modified for watermarking has an important effect on the capacity limitations as well. For example, when watermarking a magazine article or a novel, the emphasis may be on the preservation of the author's style. On the other hand, when watermarking a cooking recipe or a user manual, preserving the preciseness and jargon would be more important but not the style.

Atallah et. al [10] states that their system's bandwidth is around 8 bits per typical watermark-bearing sentence. However they have not quantified the coverage of the system.

4.5 Summary

We have presented a generic information hiding algorithm that works on any cover document that consists of multiply featured data units. This algorithm is designed to overcome the challenges of low embedding bandwidth, small number of transformations that can not be applied to any given data unit, and there are only a limited number of alternatives that a data unit can be transformed into in order to embed information in it.

We have also presented and analyzed the application of this generic algorithm to sentence level watermarking, which is a novel and powerful technique, as it relies on multiple features of each sentence and exploits the notion of orthogonality between features. We verified the practicality of this technique on a prototype natural language watermarking system and presented the performance results on this baseline system tested on a data set of 1804 sentences.

What is needed is designing an evaluation system that handles the idiosyncrasies of natural language watermarking, as well as improving the implemented system to adjust to the limitations of the NLP tools used in the process. The accuracy of the transformations can be improved by adding a more informed dictionary to increase the coverage and to overcome the conversion mistakes such as “october had to hold presidential elections”.

5. IMPROVING STEALTHINESS BY ADAPTIVE EMBEDDING

In this chapter, we present a new protocol that utilizes a tree-structured hierarchical view of the cover object and determines regions where changes to the object for embedding message data would be easily revealed by an attacker, and are thus to be avoided by the embedding process. This protocol works in conjunction with information hiding algorithms during the process of embedding in order to systematically improve their stealthiness. It is designed to work with many digital object types including natural language text, software, images, audio, or streaming data.

The protocol requires the existence of a heuristic *detectability* metric which can be calculated over any region of the cover object and whose value correlates with the likelihood that a steganalysis algorithm would classify that region as one with embedded information. By judiciously spreading the effects of message-embedding over the whole object, the proposed protocol keeps the detectability of the cover object within allowable values at both fine and coarse scales of granularity. Our protocol provides a way to monitor and to control the effect of each operation on the object during message embedding.

The goal of steganography is to embed a message $\mathcal{M}$ in a cover object $\mathcal{C}$ in a covert manner such that the presence of the embedded $\mathcal{M}$ in the resulting stego-object $\mathcal{S}$ cannot be discovered by anyone except the intended recipient. Steganographic applications only require the flexibility to alter $\mathcal{C}$ in order to be able to embed the hidden information. For this reason any type of digital object can be potentially used as a cover. For example, images, audio, streaming data, software or natural language text have been used as cover objects.

Let Alice and Bob be two parties who exchange digital objects through a public communication channel. Alice and Bob would also like to exchange a secret message

$\mathcal{M}$, however, they do not want the existence of this secret communication to be noticed by others. Alice and Bob do not want to achieve confidentiality through encryption, because the exchange of encrypted messages would reveal the existence of their secret communication. For this reason, they use a steganographic algorithm to embed $\mathcal{M}$ into a $\mathcal{C}$ to obtain a stego-object, $\mathcal{S}$, where $\mathcal{S} = (\mathcal{M}, \mathcal{C})$ and exchange $\mathcal{S}$ through the public communication channel.

The objective of the attacker Eva, is to construct a method for distinguishing stego-objects from unmodified objects with better accuracy than random guessing. Attack methods generally use statistical analysis to examine a suspicious object and search it for characteristics which may indicate that some information has been embedded in the object. For example, Eva might simply be looking for an unusual value of a characteristic that Alice has overlooked while modifying $\mathcal{C}$. Eva might also be looking for anomalies in the statistics of $\mathcal{S}$ that are different (e.g., finer) than the statistics Alice paid attention to when inserting the mark. Studies have shown that such statistical attacks are very successful on well-known image steganographic systems [31–35].

One way to defend against Eva's attacks is to inflict as little change to the document as possible [36, 37]. To this end, steganographic systems try to minimize changes in the cover object $\mathcal{C}$ when they are converted to corresponding message-carrying regions in the stego object $\mathcal{S}$. Due to their statistical nature, some regions in the cover object will experience less change in their statistics after embedding. These message-carrying regions will be harder to identify for the attacker. Conversely, some regions will easily reveal their message-carrying characteristics. For example, in the case of an image steganography algorithm that uses random bit flipping, message-carrying regions will be easier to identify when the algorithm is applied to smooth regions compared to the case when it is applied to regions with high texture. In this case a region with natural noise is more suitable for message embedding than a smooth region.

This chapter presents a general protocol for improving the stealthiness of a given steganographic algorithm by providing an efficient method to determine the most suitable regions to embed information. In our approach, we first partition the cover object $\mathcal{C}$ and impose a hierarchical structure $\mathcal{T}$ on it using this partitioning, where each node in $\mathcal{T}$ corresponds to a partition in the cover object $\mathcal{C}$. Then we use $\mathcal{T}$ both to monitor and to control the change in the statistics of the stego-object during the process of embedding the message, and to determine where the message bits are embedded.

Our protocol successfully masks the statistical effects caused by embedding both at fine and coarse levels from the attacker, since it allows constraints to be enforced on all levels of $\mathcal{T}$. Moreover the hierarchical nature of $\mathcal{T}$ allows us to impose an upper bound on the *detectability* in an arbitrary region even though the shape of this region may not be aligned with the boundaries that define the hierarchy.

For this work we have chosen color images as cover objects. However, our protocol is applicable to other steganographic application domains, such as software, audio, streaming data, or natural language watermarking.

5.1 General Framework for Adaptive Embedding

We define a protocol that can be used in conjunction with any embedding algorithm to control and improve the algorithm's stealthiness. We only require that a partitioning of the document is possible and that for any region a quantifiable measure, $d()$, that we denote as the *detectability* of the region, is defined to measure the likelihood that any steganalysis algorithm would classify that region as one with embedded information. However, this measure is hard to derive in practice. Therefore, we use a metric based on the degree the statistics of the region deviate from aggregate behavior of similar regions in a collection. For example, the detectability of an image block may be defined as the distance of the statistics of the block from

the estimated statistics obtained for that block using an image model trained on the image or on a collection of related training images.

In the following subsections we discuss the properties of the hierarchical representation. We describe the details of the hierarchical representation in Section 5.1.1, and its advantages in Section 5.1.2. We conclude in Section 5.1.4 with a proof on the upper bound of detectability caused when the hierarchical representation is used during embedding.

5.1.1 Hierarchical Representation of the Cover-Document

In our approach the cover document is partitioned into blocks and a hierarchical structure is imposed on the document using this partitioning. This hierarchical structure is used to update the statistical properties of the document during embedding. Once this information is available, it can as well be used to efficiently manage the computational complexity of the process of choosing the suitable regions to embed information. More significantly, if the detectability caused by embedding is kept below a threshold at each node in the hierarchical representation, then we are guaranteed an upper bound on the detectability of any arbitrary region of interest in the object.

Let $\mathcal{T}$ be a tree used to represent the cover document $\mathcal{C}$. Each node N_i in this tree corresponds to a block in the partition of $\mathcal{C}$, denoted by $R(N_i)$, as illustrated in Figure 5.1. We use $\mathbf{T}(N_i)$ to refer to the vector of values that contain statistical information about block $R(N_i)$. The height of the subtree rooted at N_i is $h(N_i)$. The parent and the set of child nodes of N_i are denoted by parent(N_i) and children(N_i).

The nodes for which $h(N_i) = 0$ in $\mathcal{T}$ are called leaf nodes. If N_i is a leaf node, then we refer to $R(N_i)$ as an *elementary block*. n is the number of elementary blocks, which is equal to the number of leaf nodes in $\mathcal{T}$. The elementary blocks may correspond to paragraphs in natural language text, where we can perform either syntactic or semantic analysis of sentences [10] as well as text formatting analysis [2].

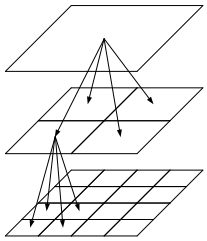


Fig. 5.1. Hierarchical representation in the form of a quad-tree for a two-dimensional stego-document. Lower levels of the tree correspond to finer partitioning of the cover object.

In software watermarking these elementary blocks might correspond to control flow blocks, whereas in images they could be blocks of pixels or regions of interest.

For a given message $\mathcal{M}$ and a cover object $\mathcal{C}$, the *embedding algorithm* $f(\mathcal{M}, \mathcal{C})$ produces the stego-object, $\mathcal{S}$. We assume that f embeds each bit of the message, M_j, by performing one or more transformations on a block of $\mathcal{C}$. For example, the transformation could be the flipping of least significant bits in an image or the changing of active sentences into passive sentences in text. This transformation is called an *embedding operation*. More precisely, the embedding operation $G(M_j, R(N_i))$ takes the j^{th} bit of $\mathcal{M}$, embeds it into the region $R(N_i)$ of $\mathcal{C}$ and produces $R'(N_i)$ of $\mathcal{S}$.

Depending on the structure of $\mathcal{C}$, $\mathcal{T}$ can be implemented as a binary tree, a quad-tree, or some other tree structure that need not have a fixed branching factor. $\mathcal{T}$ is formed such that $\mathbf{T}(N_i)$ may be obtained from $\sum_{\mathbf{v} \in \text{children}(N_i)} \mathbf{T}(\mathbf{v})$. We can reflect the statistical effects of $G(M_j, R(N_i))$ on $\mathcal{C}$ at leaf-level, upward, to all ancestor nodes of N_i in $O(height(\mathcal{T})$, which is $O(\log n)$ time.

5.1.2 Advantages of the Hierarchical Representation

Using the hierarchical representation in conjunction with an embedding algorithm provides the following advantages:

- A structured view of the statistical properties of the document is obtained for different resolutions, which will point out the *hot-spots*, which are the regions where the local statistics have anomalies compared to the global statistics of the document.

- It is possible to efficiently keep track of the changes in the statistics of the cover object after each embedding step. This is provided by reflecting the updates in statistics to higher levels in the hierarchical representation, which requires only $O(\log n)$ updates. n is the number of *elementary blocks*.

- Our protocol can set an upper bound on the detectability of arbitrary regions in the cover object if we preserve a threshold on detectability values at each level of the hierarchy. Section 5.1.4 contains a derivation of this upper bound.

- We can efficiently query document statistics. During the embedding process, some steganographic algorithms try to find the most suitable regions to embed information, as well as regions that require compensation for damage to the detectability incurred during information embedding. In the hierarchical representation only the statistics on the path to the root are relevant. Whenever we detect an anomaly in statistics of regions on this path, we will be able to focus on one subtree for corrections, whose root stands out with an abnormal value. Siblings will cooperate in "fixing" the abnormality in their parent's statistics in this process of correction.

One drawback of using a pre-computed detectability metric or model of the cover medium, is that it does not keep track of the document statistics that change during embedding, which may affect the detectability. This may cause the algorithm to incur detectability that is larger than what was initially quantified by the cost metric. Another drawback is that there is no mechanism for backtracking from a change made in the document in favor of a better embedding option that appears later during embedding, which may cause suboptimal embedding performance. Our protocol, on

the other hand, dynamically updates document statistics by monitoring statistical properties of candidate embedding regions using the hierarchical structure on-the-fly during embedding. Stealthiness is achieved through an efficient representation of the embedding costs, and it allows the embedding system to avoid regions whose use might result in poor embedding performance.

If our protocol is used in conjunction with error correction, then making only one pass through the stego-document is enough. Contrast this with steganographic methods like *Outguess* [32], that try to preserve the statistics of the cover image through a two-pass approach. In the first pass, message data is embedded into regions which are found to be suitable using a static detectability metric. In the second pass additional non-embedding changes are made to compensate for the changes in the statistical properties of the object introduced in the first pass.

5.1.3 The Protocol

In this section we will describe the protocol that ensures that the detectability measure for a region, $d(R(N_i))$ after applying $G(M_j, R(N_i))$ stays below a threshold τ. This will allow our protocol to limit the increase in detectability introduced by the embedding algorithm, thereby increasing its stealthiness. An upper-bound on the detectability is derived in the next section.

For each node we define a binary-valued function $S(N_i)$ which we will refer to as the *suitability function*. $S(N_i) = 1$ if embedding any bit from $\mathcal{M}$ in N_i will not increase $d(R(N_i))$ beyond τ, i.e. $d(G(M_j, R(N_i))) < \tau$. We also keep track of whether a message bit was embedded in $R(N_i)$, in indicator $Z(N_i)$. At each step during the embedding N^* is the suitable node selected for the embedding operation.

Let $D(\mathbf{T}(N_i))$ be a function that returns the detectability value for node N_i given the statistics, $\mathbf{T}(N_i)$. $d(G(b, R(N_i)))$ is the detectability measure after applying the embedding operation over the region $R(N_i)$, where b is the part of the message that can be embedded in $R(N_i)$.

INITIALIZATION PHASE

for each N_i in $\mathcal{T}$ in a bottom-up manner

do $Z(N_i) \leftarrow 0$

$S(N_i) \leftarrow 1$

if N_i is a leaf node

perform analysis on $R(N_i)$ to obtain $\mathbf{T}(N_i)$

else

$\mathbf{T}(N_i) \leftarrow \Sigma_{\mathbf{v} \in \text{children}(N_i)} \mathbf{T}(\mathbf{v})$

$d(R(N_i)) \leftarrow D(\mathbf{T}(N_i))$

for each N_i in $\mathcal{T}$ in a top-down manner

do if $d(G(b, R(N_i))) > \tau$

then $S(N_i) \leftarrow 0$

for each N_j in the subtree with root N_i

do $S(N_j) \leftarrow 0$

EMBEDDING & DYNAMIC UPDATE PHASE

for each M_j in M

do repeat obtain N^* from embedding algorithm

until $S(N^*) = 1$

$R'(N^*) \leftarrow G(M_j, R(N^*))$

perform analysis on $R(N^*)$ to obtain $\mathbf{T}(N^*)$

$N_p \leftarrow \text{parent}(N^*)$

while N_p is not root

$\mathbf{T}(N_p) \leftarrow \Sigma_{\mathbf{v} \in \text{children}(N_p)} \mathbf{T}(\mathbf{v})$

$d(R(N_p)) \leftarrow D(\mathbf{T}(N_p))$

if $d(G(b, R(N_p))) > \tau$

then $S(N_p) \leftarrow 0$

for each N_j in the subtree with root N_p

do $S(N_j) \leftarrow 0$

$N_p \leftarrow \text{parent}(N_p)$

In addition to the embedding protocol described above we also need to specify an extraction protocol. The extraction has to be modified to handle identification of the regions that were avoided during embedding. This can be done in a number of ways, of which we discuss two. One is by providing the extraction algorithm with the fixed threshold that was used to identify these avoided regions. This threshold information should be secret and known only to the extractor and the embedder. It may as well be embedded in the stego object in a way that the extractor can recover it before starting to extract $\mathcal{M}$. This has a couple of drawbacks. First, it imposes a constraint on embedding, namely, that the modifications done for the purpose of embedding do not cause an increase above that threshold. Second, as pointed out to us by an anonymous reviewer, it makes possible a "try-all-thresholds" attack whereby the attacker exploits the fact that there exists a threshold below which nothing was avoided at embedding time. These problems are mitigated by the fact that even though the attacker can successfully find the fixed threshold and restrict the region of attack to a smaller area, it will be harder to apply statistical attacks on that area since this region was picked for embedding for the reason that it was considered to be less vulnerable to statistical attacks.

An alternative mechanism to identify the avoided regions, one that avoids both drawbacks (but that sacrifices some capacity), would consist of augmenting the original message $\mathcal{M}$ with *markers* that identify the avoided regions. One way to do this is by embedding information about each forbidden region immediately prior (or after) that region – e.g., through a special marker symbol followed by avoided-region size. The tree structure should then be used to keep track of the boundaries of avoided regions in order to decrease the amount of bandwidth used up for such marking. At extraction time, the extractor will use this marker information to ignore the avoided regions. Note that, in this second scheme, we no longer impose the constraint that the embedding does not cause a used region to exceed the threshold τ used to identify avoided regions (although of course we would impose a constraint to not exceed

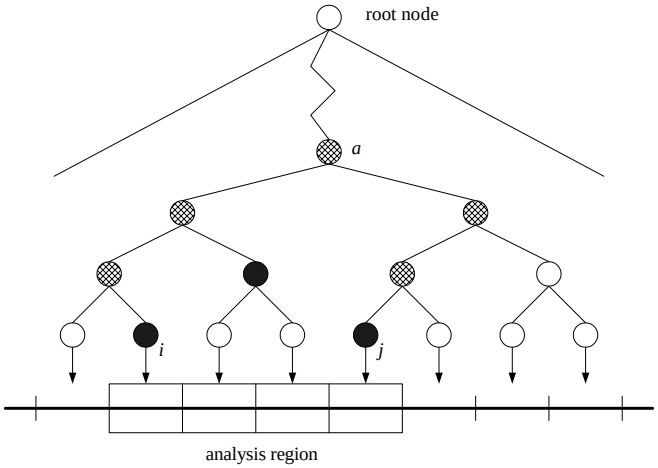


Fig. 5.2. Example of how the hierarchical representation efficiently keeps track of the changes done in the cover document for the one-dimensional case.

some other threshold $\tau' > \tau$); in this manner there is no threshold below which all was used and none avoided. Having more than one threshold can be achieved by increasing the threshold after the initialization phase. This way if embedding causes a region's statistics to exceed initial threshold, τ, but keeps them below τ', the embedding will still be allowed. If embedding causes a higher increase in statistics that exceeds τ', the algorithm should restore the original values of the region and mark the region as avoided.

5.1.4 The Upper Bound on Detectability: A Complexity Analysis

If a message embedding algorithm, is used in conjunction with the proposed protocol to monitor the statistical properties of a cover object, we are able to prove an upper bound on the detectability of the statistical features of a region of arbitrary shape in the stego-object. This upper bound provably provides robustness against attacks based on statistical analysis of the anomalies in a region of the object such as the sliding window in the generalized chi-square attack [98]. The proof we give

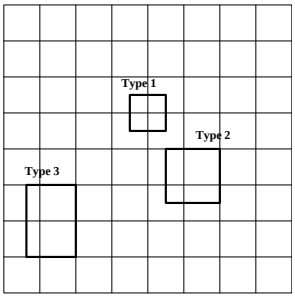


Fig. 5.3. Three basic types of regions at a fixed height h of a quad-tree $\mathcal{T}$ that are used to decompose any arbitrary region at this height.

relies on the fact that any such region can be decomposed into one or more blocks corresponding to the internal and leaf nodes of the tree structure. In the specific case of watermarking, Merhav et.al. [99] have shown that if a maximum distortion constraint can be imposed on the embedding, it is possible to quantify the capacity of the watermarking system in an information theoretic model with a non-malicious adversary.

Using the threshold of the detectability for each node as τ and an additive detectability model, where $d(R(N_i)) = \sum d(R(\text{children}(N_i))$, we show that for any region $R(N_i)$ in the document the detectability, $d(R(N_i))$, will be

- $O(\tau \log_2 n)$ for one dimensional data with a binary tree representation (e.g., audio, natural language text, software, streaming data)
- $O(\tau\sqrt{n})$ for two-dimensional data with a quad-tree representation(e.g., images).

Suppose that we are interested in obtaining the statistical properties, $\mathbf{T}(R)$, of an arbitrary region R of the one dimensional cover object shown in Figure 5.2. The region R is bounded by the elementary blocks $R(N_i)$ and $R(N_j)$. The smallest set of nodes selected to represent R are called *representative nodes* and are shown in black in the figure. $\mathbf{T}(R)$ may then be obtained using only these representative

nodes. The number of these nodes can be shown to be $O(\log_2 n)$ using the following argument: First, we search for nodes N_i and N_j starting from the root node. Let N_a be the common ancestor of nodes N_i and N_j with smallest height. We find the paths from N_a to the node N_i and pick all the right children of the nodes on the path and similarly pick the left children while searching for N_j from N_a as representative nodes. The shaded nodes in the figure are the nodes visited during this search. By this argument, since the length of the paths from N_a to N_i and N_j will be at most $\log_2 n$ the number of representative nodes will also be $O(\log_2 n)$. If we then sum up the detectability values for these nodes, we get a worst case upper bound of $O(\tau \log_2 n)$ on $d(R)$.

A similar approach can be used to derive an upper-bound in the quad-tree case. We define three basic types of regions, R_1, R_2, and R_3. We use the notation $R_1(h)$ to refer to a type R_1 region at height h. An $R_1(h)$ region does not cover any block in full at height h. An $R_2(h)$ fully covers a block in one corner and partially covers three neighboring blocks at height h. An $R_3(h)$ totally covers two blocks at one side, and partially covers two neighboring blocks height h. Any arbitrary region at height h may be decomposed into a combination of $R_1(h)$, $R_2(h)$, and $R_3(h)$. Refer to Figure 5.3 which illustrates these regions.

The detectability for $R_1(h)$ is given by $d(R_1(h))$, which we will refer to simply as $d_1(h)$. Similar definitions apply for regions of types R_2 and R_3. We can write the detectability values for regions at height h in terms of detectability values for regions at lower levels of the tree as

$$\begin{aligned} d_1(h) &\leq 4d_2(h-1) && (5.1)\\ d_2(h) &\leq \tau + d_2(h-1) + 2d_3(h-1) && (5.2)\\ d_3(h) &\leq 2\tau + 2d_3(h-1) && (5.3) \end{aligned}$$

By using the recursion on $d_3(h)$, we obtain

$$\begin{aligned} d_3(h) &\leq \tau(2^h 3 - 2) && (5.4)\\ &= O(\tau 2^h) && (5.5) \end{aligned}$$

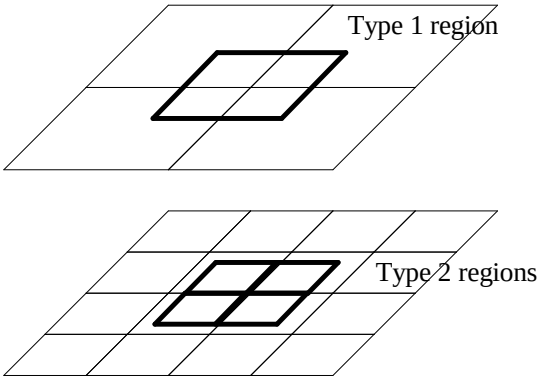


Fig. 5.4. Decomposition of a type R_1 region

$$= \quad O(\tau\sqrt{n}) \tag{5.6}$$

where we have used the fact that $h = \log_4 n$. We can use this result to solve for $f_2(h)$ as

$$\begin{aligned} d_2(h) &= \tau + d_2(h-1) + 2O(\tau\sqrt{n}) && (5.7)\\ &= \tau \log_4 n + \tau + 2O(\tau\sqrt{n}) && (5.8)\\ &= O(\tau\sqrt{n}) && (5.9) \end{aligned}$$

which shows that $f_1(h) = O(\tau\sqrt{n})$.

5.2 Experimental Results for Adaptive Embedding

We have performed experiments to illustrate the effectiveness of our protocol in increasing the stealthiness of a steganographic algorithm. For our experiments we have chosen a simple least significant bit (LSB) embedding steganographic algorithm for color TIFF images, however any other embedding scheme may be employed. A quad-tree structure is used for $\mathcal{T}$.

The embedding algorithm first pads $\mathcal{M}$ with random bits to produce a message $\mathcal{M}'$ with a size in bits equal to the number of pixels in $\mathcal{C}$. $\mathcal{M}$ is located at a random place within $\mathcal{M}'$. A small part of $\mathcal{M}'$ is used to for storing the starting point of

$\mathcal{M}$ within $\mathcal{M}'$ and the size of $\mathcal{M}$. Both red and green planes of $\mathcal{C}$ are used for embedding. Each pixel of $\mathcal{C}$ carries only one bit of $\mathcal{M}'$. Bits of $\mathcal{M}'$ are XOR'ed with a random bit, which is generated by a pseudo random bit generator that takes the stego key as a seed. This randomizes the bits of $\mathcal{M}'$. The embedding length is equal to the number of pixels in $\mathcal{C}$. The message length, length of $\mathcal{M}$, is smaller than the number of pixels in $\mathcal{C}$.

The elementary blocks in $\mathcal{C}$ were chosen to be 8×8 pixel blocks. For the experiments reported in this chapter we chose the pixel variance of the elementary blocks as the statistical information at the leaf nodes, or $T(N_i) = \mathrm{Var}(R(N_i))$. For internal nodes, we have $\sum_{\mathbf{v} \in \mathrm{children}(N_i)} \mathbf{T}(\mathbf{v})$. The detectability measure for N_i was simply selected to be equal to $-T(N_i)$, in other words, we have

$$d(R(N_i)) = \begin{cases} -\mathrm{Var}(R(N_i)), & \text{for leaf nodes} \\ \sum d(R(\mathrm{children}(N_i))), & \text{for internal nodes} \end{cases}$$

This choice is motivated by the following observation. Usually the message $\mathcal{M}$ that is embedded is an encrypted version of the secret message to be sent, in which case, the sequence of bits in $\mathcal{M}$ will have noise-like characteristics, which will cause an increase in the variance of $\mathcal{C}$. Let the variance of a region of the cover image be σ_c^2 and suppose that after message embedding the variance of that region increases to $\sigma_s^2 = \sigma_c^2 + \epsilon$. For regions with small σ_c^2, the contribution ϵ may make the region visible to steganalysis. Therefore, regions with high variance should have low detectability values and are suitable for embedding. A sample image and the corresponding 8×8 block variances are shown in Figure 5.5 and Figure 5.6, respectively.

A quad-tree structure $\mathcal{T}$ is initialized using the initialization phase of the algorithm given in Section 5.1.3. Let V_h be the set of nodes at height h of $\mathcal{T}$. For each height, h, we calculate the threshold on detectability values, τ_h, as

$$\tau_h = c \left(\frac{1}{|V_h|} \sum_{N_i \in V_h} d(R(N_i)) - \min_{N_i \in V_h} d(R(N_i)) \right) \tag{5.10}$$

where c is a parameter that controls the number of suitable regions selected. In our experiments we have chosen $c = 0.5$.

The suitability of the node N_i is set using

$$S(N_i) = \begin{cases} 1 & \text{if } d(R(N_i)) < \tau_{h(N_i)} \\ & \quad d(R(\text{parent}(N_i))) < \tau_{h(N_i)+1} \\ 0 & \text{otherwise} \end{cases}$$

Note that the detectability values of both N_i and parent(N_i) are taken into consideration in deciding if $R(N_i)$ is a suitable region. This is a relaxation on the algorithm described in Section 5.1.3 in order to avoid setting large blocks of $\mathcal{C}$ as unsuitable for embedding and also taking into account the detectability measures of the siblings of N_i, which are reflected in $d(\text{parent}(N_i))$. This relaxation can be tuned to take into account ancestors of N_i that are further up in $\mathcal{T}$ than parent(N_i) for achieving better stealthiness.

During the embedding, our protocol restricts the embedding system to use only the suitable regions. The unsuitable regions after the initialization phase of the algorithm for the image in Figure 5.5 are shown in white in Figure 5.7. After the final phase of the algorithm the number of unsuitable regions increase for this image, as you can see in Figure 5.8.

Figure 5.11 and Figure 5.12, show the difference images between the cover image shown in Figure 5.5 and stego-images produced using two different approaches. The gray regions in Figure 5.12 represent the regions that are the same in both the cover and stego images. From these images it can be seen that our protocol guided the embedding algorithm to avoid regions with high variance.

We tested the performance of our system using the steganalysis attack proposed in [34]. Since the feature extraction of this system was designed for grayscale images, we processed the red, green and blue channels independently. In our experiments we used 141 TIFF images of size 512×512 pixels obtained from the Watermark Evaluation Testbed (WET) [100].

In order to perform the classification between cover and stego images we have used both support vector machine (SVM) and the Fisher linear discriminant (FLD) classifiers. LIBSVM tools [101] were used for SVM classification. Given the em-

Fig. 5.5. A sample cover image.

Fig. 5.6. Variances of elementary blocks of the sample image. Higher values are represented by lighter regions. Note that variance values are inversely proportional to detectability.

bedding algorithm itself randomizes the message, we inserted a text message, the first chapter of the Tale of Two Cities by Charles Dickens [102]. Although, actual message length is 18%, embedding length is 100% for plain embedding, and it varies

Fig. 5.7. Initial suitability map for sample image. The regions shown in white are the ones that are labeled as unsuitable for embedding.

Fig. 5.8. Final suitability map for sample image. The regions shown in white are the ones that are labeled as unsuitable for embedding.

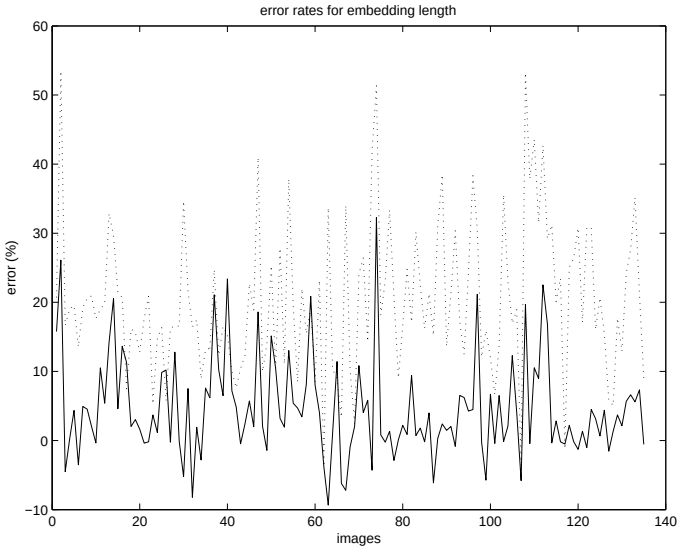


Fig. 5.9. Error of RS-Analysis for the green channel using LSB embedding only and using LSB embedding with hierarchical protocol

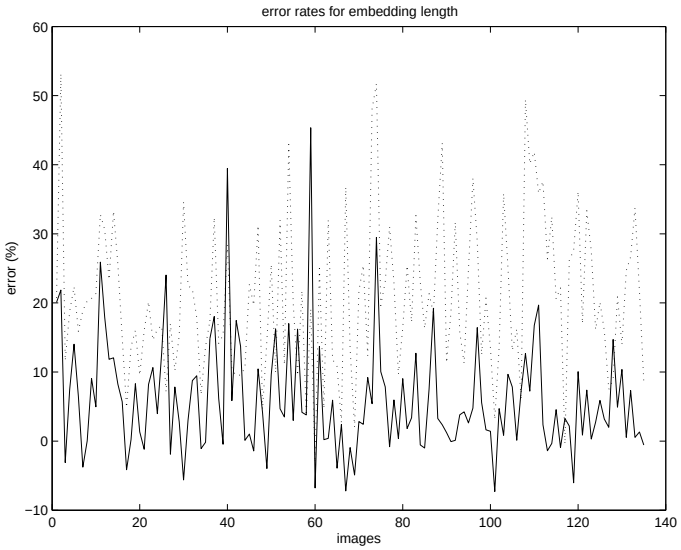


Fig. 5.10. Error of RS-Analysis for the red channel using LSB embedding only and using LSB embedding with hierarchical protocol

for each image when embedding is combined with the protocol. While we force the system to stay out of avoided regions, we decrease the size of random part of $\mathcal{M}'$. The average embedding length was 42% for the embedding with the protocol.

classification method	plain embedding	embedding with hierarchical protocol
SVM	%49.65	%42.65
FLD	%76.92	%69.23

Table 5.1
Classification results.

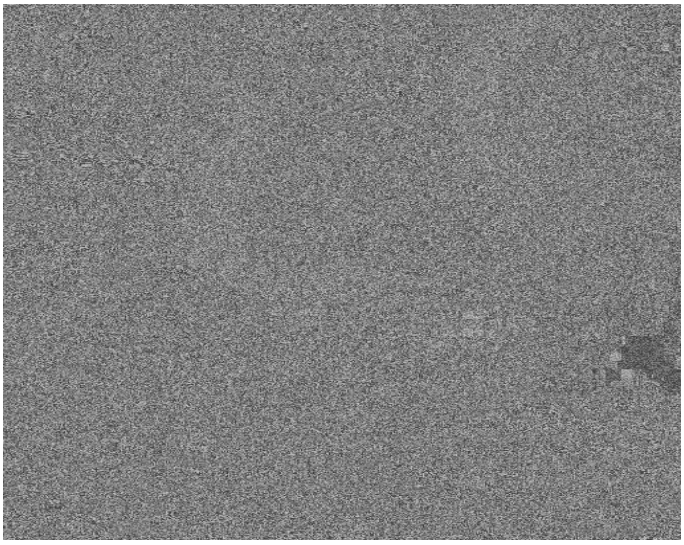


Fig. 5.11. Difference of cover image and stego image generated using LSB embedding only

The accuracy of classification for the images in our test set are given in Table 5.1. Although both classifiers are not very accurate at detecting LSB embedding, from this table it can be seen that our protocol was still able to decrease the detectability of the steganographic method.

We have also performed tests using *RS steganalysis* [103] over the green and red color planes which were used as the embedding channel. Our aim was not to evaluate RS steganalysis *per se* but rather to evaluate the impact of our technique on increasing the stealthiness against statistical steganalysis. This attack is specifically designed to detect LSB embedding. However, as it is also stated in [103] and [35],

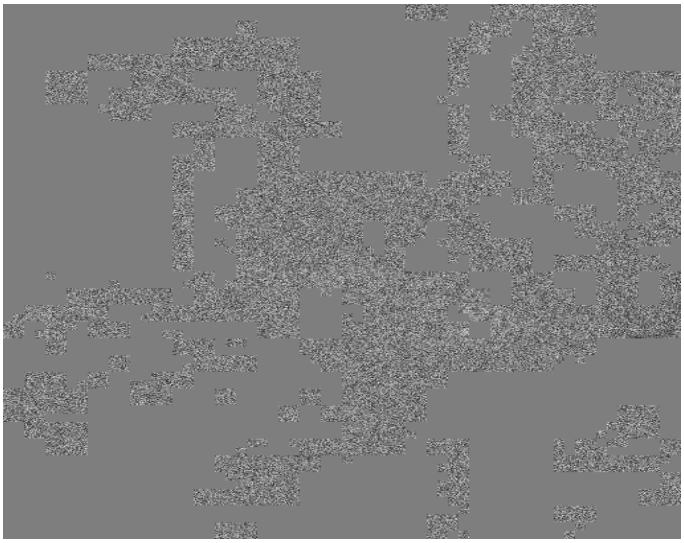


Fig. 5.12. Difference of cover image and stego image generated using LSB embedding with hierarchical protocol

RS steganalysis is more successful with grayscale images and for messages that are randomly scattered over the stego-image. This is not the case for our embedding algorithm. Even with the plain embedding the error rates were high, because the LSB algorithm perturbs LSBs of all pixels. Therefore, estimated embedding lengths are sometimes higher than 100%. Still, detection errors increase when our protocol is used, as you can see in Figure 5.9 for green color plane and in Figure 5.10 for red color plane.

5.3 Summary

In this chapter, we describe a protocol for improving the stealthiness of information-hiding schemes, and give the test results of an implementation of it on LSB based image steganography. Although our protocol does not completely eliminate the statistical anomalies caused by embedding that are a major threat to the embedding algorithm's stealthiness, it effectively controls their severity and decreases their total number.

Guided by a continuously updated detectability representation of the cover object, our protocol provides a mechanism for controlling statistical anomalies at both fine and coarse scales of granularity. We use a hierarchical representation to manage the complexity of dynamically keeping track of the detectability of the cover object during embedding.

We also quantify how bounds on the detectability of regions from the hierarchy translate into detectability bounds for arbitrary regions.

6. APPLICATIONS OF INFORMATION HIDING TO PRIVATE COMMUNICATION AND DEFENSE AGAINST PHISHING

There are many other applications of watermarking besides the immediate applications of copyright protection, fingerprinting or meta-data binding. During the development of this thesis we have contributed to the design and building of a private communication system that uses watermarking. In addition to that we have built a phishing defense system, ViWiD, that makes use of visible watermarking in order to provide a usable verification mechanism for corporate that provide services to their users over the Internet.

6.1 Watermarking for Private Communication: WaneMark

The WWW increasingly allows people to create and update content for public access. Some of this information is collaboratively owned (created and maintained), while other information is privately owned and maintained (but still publicly accessible). Whereas it is unethical to modify the former for covert communication, it is quite legitimate to do so with the latter, and we give a design for doing so while achieving both plausible deniability and automatic perishability of the covert message (the message disappears unless periodically refreshed by the encoder). Traditional information-hiding has looked at the problem of embedding a message in a static version of an online document, the problem of doing so for rapidly evolving document collections has not been considered in the past. This work [104] shows that it is possible to do so, and in a manner that actually makes use of the rapidly evolving nature of the documents to achieve the above-mentioned property of evanescence: That the message decays over time and eventually becomes completely erased

unless it is refreshed. Therefore the mark needs to be continuously maintained as the document evolves, in a manner that prevents the adversary from knowing who is doing the refreshing yet that allows the intended reader of the mark to recover it without any form of explicit communication. One advantage of our scheme is that the mark's reach is now unbounded: It can be read by any authorized entity on the web (anyone with the secret key), and the reading of it is indistinguishable from normal web access patterns. Another advantage is the "hiding in the crowd" effect: Many people are updating the documents, thereby providing a cover for the one person surreptitiously injecting and refreshing the mark, or replacing it with another mark message. We have also demonstrated the feasibility of the proposed technique, and shown that remarkably little effort is required to implement our scheme over today's web – refer to [104] for details.

Although messages that self-destruct were featured in spy movies, their potential usefulness is not limited to the original purpose of their self-destruction in such movies, which was to prevent their being read by a hostile adversary after they had served their useful purpose of communicating the next mission to Mr Phelps. The case for making such messages perishable includes many possible reasons: (i) Such messages can become stale and thereby convey misleading information to their intended recipient (e.g., the message "Alice and Bob are both doing fine" after one of them ceases to be doing fine) – more generally, because information is perishable and becomes stale, useless, or even dangerous as the world changes, it stands to reason that stealthy messages that convey that information should be similarly perishable and self-efface; (ii) if the message involves a secret key, then the longer it lingers, the more it is likely that it (and the key used to hide it) may eventually be compromised by an adversary who has enough computational resources (or will have such resources in the future - systems have to be resilient not only against the computational power of today but also against that of the future); (iii) the desired updating of the message (either refreshing it or replacing it with another message) by the person who wrote it may become infeasible through that person's accidental loss of the secret key, loss

of access to the online world, or other physical inability to take such action; (iv) the automatic disappearance of the message can be used to communicate the very fact that (iii) has occurred; (v) the person in charge of removing the message may be negligent, or may erroneously believe that someone else was supposed to do the removal, etc.

Providing privacy preserving Web-based communication is an active research area [105–107]. Achieving this is hard because many players (such as authoritarian governments, aggressive marketeers, etc) want to have a complete profile of Web users and a log of their actions on the Web. We propose a private communication channel, that ensures plausible deniability, and automatic perishability of the messages. We achieve this goal through the use of collaborative web content available on the Internet, without unethically interfering with the functionality of these valuable services, and without any need for modification of publicly available data (defined as data not owned by the sender of the message). In a nutshell, our scheme is based on pairing a privately owned web page with a collaboratively owned web page, and to use this pair as a cover document. The embedding changes are only performed on the privately owned web page.

There are many challenges involved in designing such a system:

- how to use, for marking purposes, the content that we cannot modify either because we have no control over it (e.g., news portals), or because it is unethical to use the ability to modify it for marking purposes (e.g., wikis, forums)
- heterogenous content; most published marking schemes assume one type of content, whereas we are now faced with a semi-structured collection of different content types (text, images, audio, video, annotations, etc)
- not interfering with the proper functioning of the publicly owned covers used
- providing controlled perishability

- being stealthy while using a publicly accessible cover document (as is customarily required in information hiding)
- providing plausible deniability of the covert communication
- providing high covert communication bandwidth (especially challenging when the document consists of data that has low embedding capacity, such as text)

The difficulty of these challenges is exacerbated by the increasing power that a potential adversary can muster – the repertoire of information sources for such an adversary now includes forum or blog boards (most or least accessed pages, most active member etc.), web-bots, ISP logs, search engines, web page tracking engines [108, 109], to mention a few.

Our system fulfills most of the requirements listed for a Web based publishing system listed by Waldman et al. [107]: censorship-resistant, tamper evident, source anonymous, updatable, deniable, fault tolerant, persistent (i.e., no expiration date), extensible, freely available. The only requirement we do not provide is persistency over time (which is inherently incompatible with a self-destructing message).

Previously proposed private communication systems use a third party distributor (e.g., e-mail services [105]) to store and distribute the message to intended receivers. In our system, the sender does not use a third party distributor and therefore has a greater degree of control. The sender also has the option of privately storing the cover document (until it is cached by another Internet company [110, 111]) if she/he wishes to maintain the privately owned web page that is used as part of the cover document. Refer to [104] for more information about this system.

6.2 Watermarking for Phishing Defense: ViWiD

Natural language watermarking provides a way to hide meta-data information into the cover text in a way that it is not easy to separate the meta-data from the

text without changing the semantic form or syntactic form of the sentences in the text.

In addition to meta-data binding for copyright protection, natural language watermarking provides an integrity checking method that "travels with the content" whenever the content is (mis)used, which makes it very valuable for applications that involve private communication. For example, phishing - a web based deception technique - exploits the fact that customers of online services trust e-mail communication. This attack is successful partially due to the fact that secure e-mail systems are not commonly employed by the non-tech-savvy users. There is a great need for binding the source information to the documents involved in private communication. See [50] for a system that uses watermarking against phishing.

Another relevant problem is enforcing security policies on private communications. An example of such a system would be e-mail communications that involve groups of people where each of the participants has a different level of access control rights. In such systems, unless the security level is bound to the text content, there is no possibility of enforcing security policies automatically when the encryption or digital signature is separated from the document (whereas a watermark inherently "travels with the content"). In addition, robust natural language watermarking algorithms will enable a wide range of applications such as text auditing, tamper-proofing, and traitor tracing.

As a future research topic, we will work on watermarking ensembles of text documents collectively. This would create new application areas that concern the access control policies for digital libraries.

In this chapter, we present a watermarking based approach, and its implementation, for mitigating phishing attacks - a form of web based identity theft. ViWiD [1] is an integrity check mechanism based on visible watermarking of logo images. ViWiD performs all of the computation on the company's web server and it does not require installation of any tool or storage of any data, such as keys or history logs, on the

[1]Stands for VIsible WatermarkIng based Defense against Phishing

user's machine. The watermark message is designed to be unique for every user and carries a shared secret between the company and the user in order to thwart the "one size fits all" attacks. The main challenge in visible watermarking of logo images is to maintain the aesthetics of the watermarked logo to avoid damage to its marketing purpose yet be able to insert a robust and readable watermark into it. Logo images have large uniform areas and very few objects in them, which is a challenge for robust visible watermarking. We tested our scheme with two different visible watermarking techniques on various randomly selected logo images.

Our society has increasingly become a digital society where many critical applications and services are provided on-line. Examples of such applications are financial services, retail services, on-line news channels and digital libraries. This paradigm shift has had a beneficial effect on business and education by providing faster and easier access to services and information. Unfortunately, it has also exposed these services to malicious attacks that are more difficult to detect and defend against. One of the major security concerns in cyberspace, having impact on individuals as well as businesses and organizations, is identity theft. According to a recent Congressional Statement of the FBI Deputy Assistant Director [112], on-line identity theft represents a significant percentage of the total number of crimes committed in cyberspace.

Phishing is a form of on-line identity theft in which attackers send fraudulent e-mails and use fake Web sites that spoof a legitimate business in order to lure unsuspecting customers into sharing personal and financial data such as social security numbers, bank account numbers, passwords, etc. The incidence of phishing attacks has increased significantly over the last couple of years. By the end of December 2004, Symantec Brightmail AntiSpam antifraud filters were blocking an average of 33 million phishing attempts per week, up from an average of 9 million per week in July 2004 [113]. Acknowledging that phishing is a significant threat to e-commerce, over 600 organizations formed the Anti-Phishing Working Group [114] focused on eliminating identity theft due to phishing.

Due to the rapid growth in the impact and number of phishing attacks, there is a considerable research effort going on both in academy and industry for developing robust and easy to use defense systems. Most of the currently available defense systems against phishing either limit the access of the user or display warning messages when they detect suspicious activities. Examples of such systems include e-mail spam filtering or browser plug-ins specially designed for monitoring user's transactions, e.g. SpoofGuard [115], Netcraft [116] or Ebay [117] toolbar. Another approach focuses directly on mitigating man-in-the-middle phishing attacks through a multi-factor authentication scheme [118]. We will briefly review these existing approaches in Section 6.2.1.

We propose a defense system, ViWiD, that mitigates phishing attacks through an integrity check mechanism built on visible watermarking techniques. This mechanism is based on asking the user to check the validity of the visible watermark message on the logo images of the web pages. We propose two types of watermark messages: The first type is the time only watermark when the company's web site embeds only the current date and time of the user's time zone into the logo image. Recall that IP address can be used to determine the time zone of the user machine. An example of this type of watermarked logo can be seen in Figure 6.2[2](a). The second type of watermark message includes a secret shared between the user and the company together with the time stamp, as shown in Figure 6.2(b). The logo images with this shared secret watermark message can be displayed either after the user logs in, or through the usage of cookies. Since this watermarked logo displays a secret shared only between the user and the genuine company, the appearance of such information on the logo is enough for the user to confirm the genuineness of the web site.

The integrity checking system is designed to include a shared secret between the company and the user in order to prevent the phisher from performing the current "one-size-fits-all" attack. This means that even if the phisher is successful

[2]There is a quality loss in the displayed images through out the chapter due to the conversion from Graphics Interchange Format (GIF) to Post Script (PS) format

in removing the watermark, he can not insert back the expected watermark without knowing the shared secret between the company and the user.

The reasons for following this particular approach are as follows. First, phishing is primarily a social engineering attack which involves the active participation of the users to succeed. Thus, the approach towards mitigating such attacks must also include the co-operation of the users to some extent. Indeed even today, the company web sites advise the users to follow well-known safety measures such as checking the padlock at the bottom of the screen and the 'https' sign in the URL, both of which signify a SSL connection. But, most of the victims of phishing attacks today are naive users who are not tech savvy enough to check the certificates or security sign. Also, the presence of a SSL connection by itself does not confirm the true identity of the web site. Any site, even a spoofed site, can establish a SSL connection. Communicating to naive users the true identity of the web site is a challenging problem. Hence, we propose the use of a shared secret which the user chooses himself when he registers with the original site. This shared secret can be easily recalled and recognized by the user. Using this secret, the company authenticates itself to the user. In the remaining of this chapter, we will refer to this secret as a *mnemonic*.

Second, we chose *the web site logo* as a carrier for the watermark message, since the user always expects to see a logo on a web page. Besides this, the phisher always has to re-use the web site logo when he imitates the pages of the original web site. Since the original logos are always watermarked in our approach, it is not trivial for the phisher to remove them and insert his own watermarks. Even if the phisher is able to remove the watermark, he will not be able to insert the mnemonic for each user. More details about the proposed framework are presented in Section 6.2.2.

The rest of the chapter is structured as follows. Next section provides a brief introduction to the anatomy of phishing attacks, state-of-the-art defense systems against phishing and summarizes the visible watermarking technique we use. We

introduce our experimental set up and results in Section 6.2.3 and discuss possible attack models in Section 6.2.4. We conclude in Section 6.2.5.

6.2.1 Related Work on Defense Against Phishing

In a typical phishing attack, a person receives an email apparently sent by an organization that the person interacted with before, and with which he has possibly built a trust relationship (e.g., his bank or a major retail on-line store). The email usually projects a sense of urgency, and asks the client to click on a link that, instead of linking to the real web page of the organization, will link to a fake web page that is subsequently used to collect personal and financial information. There are two victims in phishing attacks: the customer being tricked into giving away personal information and thus allowing the attacker to steal its the identity, and the company that the phisher is posing as, which will suffer both financial loss and reputation damage due to the attack.

Unauthenticated E-mail The major mechanism to start the attack is using forged e-mails. The phisher can forge e-mails by faking the source information displayed on the e-mail programs. Moreover, phishers can forge the content of the e-mail by getting a template of the style of legitimate e-mails when they subscribe to the company. The attack has a great impact because e-mail is the main communication channel for the online services. The subscribers or customers are expected to follow their transactions and receive confirmations via e-mails.

User Actions Phishing requires human interaction as like many of other on-line attacks do. However, unlike other attacks (worms or viruses spreading via e-mail) where one click is enough to trigger the attack, phishing requires active participation of the user at several steps, including providing personal information.

Deceptive View The core of the phishing attack lies in the ability of the phisher to create a web page looking very similar to a web page of the legitimate organizations by simply copying the logos, and using a style and structure similar to those on the legitimate page. In other words, the information displayed on web pages is not tied to its creator or owner in a way that removing that tie, will deteriorate the data beyond repair. In addition, many browsers are modifiable on the client side, allowing a phisher to remove buttons, not to display certain information, or to mislead the user by playing with the graphics.

A major challenge in addressing phishing attacks lies in designing mechanisms that are able to tie the data displayed on a web page (or related with a web page) to its legitimate owner. This is a difficult task because of the nature of the information displayed, its heterogeneous nature, and the dynamic characteristic of web pages.

Previous Approaches to Prevent Web-Based Identity Theft

Secure Email Many forms of phishing attacks can be prevented by the use of secure email tools such as Privacy Enhanced Mail (PEM), Secure Multipurpose Internet Mail Extension (S/MIME) and Pretty Good Privacy (PGP). However, to this date, secure email is not widely used over the Internet, because of scalability, trust, and difficulty to deploy it. A good discussion of certificate based security is provided in [49] by Ellison and Schneier.

Client-side Defense One direction in addressing the phishing attack was to provide the client with more accurate information about the web sites that he accesses. Various tools empowering clients with more information have been designed to mitigate phishing attacks. One such tool is SpoofGuard [115] which computes a spoof index and warns the user if the index exceeds a safety level selected by the user. The computation of the index uses domain name, url, link and image checks to

evaluate the likelihood that a particular page is a spoof attack. One component of SpoofGuard maintains a database of hash of logo images and corresponding domain names. Later on a web page when the hash of the logo image matches a hash in the database, the current url is compared with the expected domain name, if these do not match the user is warned.

Netcraft, [116] also has released an anti-phishing toolbar that provides information about the web sites that are visited by a client such as the country hosting the sites and enforces the display of browser navigational controls (toolbar and address bar) in all windows.

Herzberg and Gbara [119] proposed establishing, within the browser window, a trusted credentials area (TCA). It is the browser that protects the TCA by preventing its occlusion. The scheme has its costs (it requires logo certification, logo certificate authorities, etc), but tolerates more naive users.

Cryptography-based Defense TriCipher, Inc. very recently introduced TriCipher Armored Credential System (TACS) against man-in-the-middle phishing attacks [118]. TACS works when the SSL client authentication is turned on. This means that the SSL protocol will have three steps: authenticate the web server to the client browser, set up encrypted communications and authenticate the end user to the web server. Common usage of SSL consists only of the first two steps. TACS uses two different types of credentials. The first one is called *double armored* credentials, and requires the users to install the TriCipher ID protection tool on their machine. The tool automatically pops up when the user goes to a page that is protected by SSL and encrypts (signs) the password using a key stored in the Trusted Platform Module or Windows® Key Store. Then the TACS appliance at the web server side authenticates the user. The second type of credentials is called *triple armored* credentials which uses, besides the user password and the key stored on the user's machine, a smart card or a USB memory stick to store a key or a biometric. The user's password is signed both with the key on the user's machine and another

key stored elsewhere. The *triple armored* credential system raises the bar for the phisher because even if he is able to steal the key on the user's machine, he also has to steal the key stored on an outside system.

Shared Secret Schemes More recently two new authentication schemes, similar in nature to our system, have been brought to our attention. PassMark Security, Inc.'s [120] *2-Way Authentication Tool* helps the users identify known servers. In this scheme, the user provides the server with a shared secret, an image or a text phrase, in addition to his regular password. The server presents the user with this image, and the user is asked to recognize it before entering his password and authenticating himself to the server. Passmark images are randomly assigned to users from a pool of over 50,000 images and later the users can change their Passmarks, like they change their passwords, by selecting new images from the pool or by uploading an image of their choice.

Dhamija and Tygar proposed using Dynamic Security Skins [121] as a defense against phishing. Their system is based on having a *Trusted Window* in the browser and using the Secure Remote Password Protocol (SRP) [122] for authentication. Spoofing of trusted window is prevented by providing an image which is a shared-secret between the user and his browser. This window is dedicated to username and password entry. SRP is a *verifier-based protocol*. SRP provides the functionality for the server and user to authenticate each other over an un-trusted network by independently generating a session key based on a verifier. User sends the verifier to the server only once when he is registering. In Dynamic Security Skins, this verifier is used by the browser and the server to generate *a visual hash* that is displayed in the background of the trusted window and in the server's web site. To authenticate the server, the user needs to visually compare the two images to check if they match.

Limitations of Previous Approaches

Even though the client-side defense tools raise the bar for the attackers, they do not provide a complete solution. Many checks and enforcements used by the client-based defense tools can be fooled by attackers having a reasonable understanding of web site construction [115]. For example, the image check system of SpoofGuard can be fooled by a mosaic attack where the attacker partitions the logo image into pieces, but displays them in appropriate order so that the user thinks that he is looking at a legitimate logo.

Moreover, any "client side only" defense mechanism will suffer from false positives. Too many warnings will interfere with the user's browsing experience and the user will simply turn off the protection mechanism in such cases.

In addition to the above limitations, the "client side only" schemes leave all of the defensive actions and computational costs up to the user's machine, even though the companies have larger computing power at their disposal and can do more to mitigate the risks. Moreover it is the companies who create the content (logo, style etc) that the attackers seek to imitate and/or misuse. Therefore, we believe that companies can play a larger role in the overall defense strategy to mitigate phishing attacks.

On the other hand, cryptography based tools require the user to download a tool on every machine he uses to access his online accounts, and/or the user is required to carry another medium e.g. a smart card or USB memory stick with him when he wishes to access his accounts. One other limitation for TACS [118] is that it is designed to work only for man-in-the-middle phishing attacks. When the phisher directs the users to his web page which might have a SSL connection but without the client authentication module turned on, the TriCipher ID protection tool will not pop up and sign the password.

The shared secret schemes introduced in Section 6.2.1 are similar to our approach in the sense that they focus on how a legitimate server can authenticate itself to the

user. However, our approach and these two approaches diverge on the generation and presentation of the shared-secret. The main drawback of PassMark approach is that the shared secret is not bound to a particular location on the original web page. This makes the scheme less user-friendly as across different service providers, the users will have to look at different places to find their shared secret on the web page. On the other side, Dynamic Security Skins [121] scheme suffers from asking the users to dedicate part of their browser window to the Trusted Window. Besides, this scheme trusts the client's browser on vital security processes such as storing the verifier and generating the visual hash.

Overall, a complete solution for defense against phishing, must address all three causes that allow a phishing attack to be possible: unauthenticated e-mail, user actions and deceptive view. Thus, a complete solution should include mechanisms that can analyze what the user sees, analyze the e-mail and web page content, and provide integrity checks for these components. In addition, such a system should be easy to use and deploy.

Visible Watermarking Overview

Visible watermarking is the insertion of a visible pattern or image into a cover image [123]. A useful visible watermarking technique should meet the following requirements: preserving the perceptibility of the cover image, providing reasonable visibility of the watermark pattern and robustness [124]. Huang and Wu summarize the insertion of a visible pattern into the cover image as:

$$I' = K_1 \cdot I + K_2 \cdot W \tag{6.1}$$

$$D(E_I(I'), E_I(I)) < Threshold_I \tag{6.2}$$

$$D(E_W(I'), E_W(I)) < Threshold_W \tag{6.3}$$

In Equation 6.1, I represents the cover image, W represents the watermark image and I' represents the watermarked image. Equation 6.2 represents the boundary on

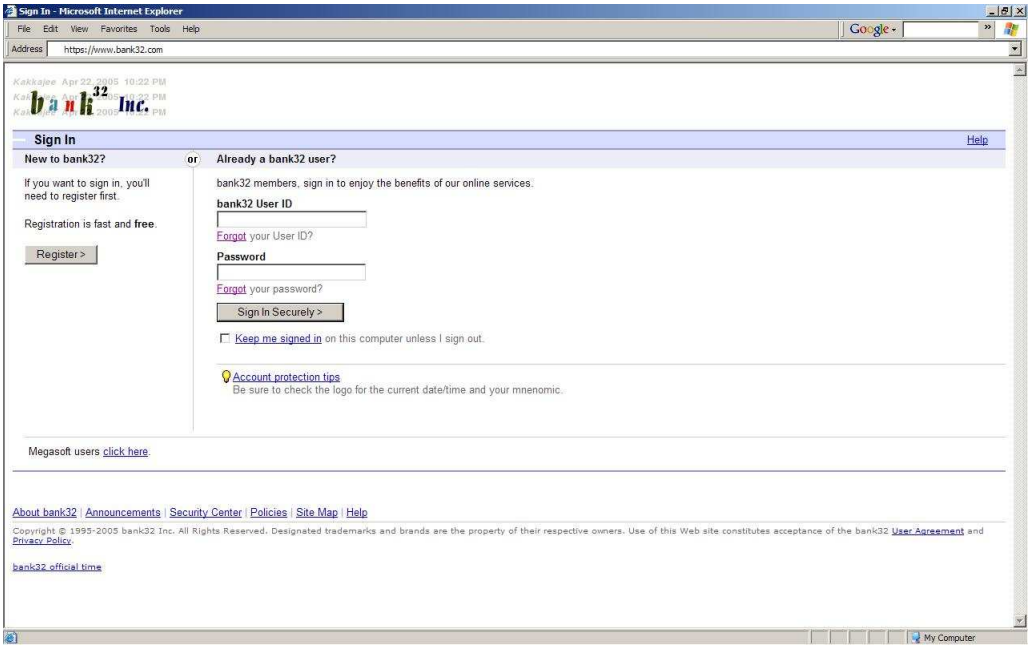


Fig. 6.1. A generic login page with a watermarked logo image, scaled to half of its original size for space requirements.

the distortion of the perceptibility of the cover image, while Equation 6.3 represents the boundary on the distortion of the visibility of the watermark patterns. D is a distance function measuring the perceptible difference of its two entries. E_I is a image feature extraction function for the cover and watermarked images. E_W is a separate image feature extraction function for the watermark pattern. $Threshold_I$ and $Threshold_W$ represent the largest allowable distortion on perceptibility of the cover image and on the visibility of the watermark pattern respectively.

In ViWiD, we use visible watermarking in order to provide the users with visibly watermarked logo images and the visible watermark pattern is generated dynamically depending on a shared secret between the user and the company.

6.2.2 Proposed Approach

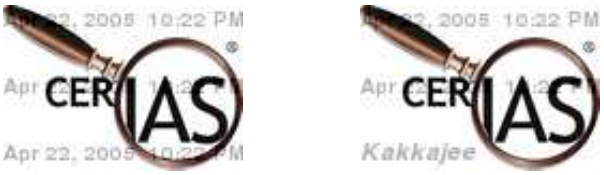
The content of the e-mail and the spoofed page are the means through which the "social engineering" aspect of phishing is carried out. The phisher tricks the

user into submitting sensitive information by using the content and the style stolen from the legitimate company. A good defense mechanism must require an integrity check method that "travels with the content" when it is used or misused. One way to achieve this is digital watermarking. Our approach watermarks the content on the legitimate web page in a way that provides an integrity check. We use the logo images as the watermark carriers, based on the observed fact that nearly all phishing attacks re-use the logo images.

Design Goals and Motivation

The user can be tricked into a phishing attack, only if the phishing e-mail is imitating a company with which the user has previously established a trust relation. All companies, targeted by phishing attacks, have large numbers of users using their online services. Many of the users use several varieties of browsers and more than one computer to access their account online. A key-based watermark detection system requires the keys for detection and extraction to be distributed to all the users. We avoid the key distribution problem by using a visible watermark, with a human involved in the detection process. This way we also give the user an active role in the defense against a social engineering attack.

We seek to thwart the "one size fits all" attacks by designing the visible watermark message such that it is unique and varies with time. ViWiD embeds a local time stamp which is updated periodically and a mnemonic selected by the user while the online account established. The rationale for using the time stamps is that phishing sites are usually up for 6 to 7 days [114], and unless the phishers are able to remove the watermark, their stolen logo cannot display a fresh time to all the intended victims. Also, this system should never ask for the user's mnemonic after the online account is established in order to avoid the possibility of revealing the mnemonic even if the user mistakenly enters his login and password to a spoofed site.



(a) (b)

Fig. 6.2. Logo images watermarked with $ImageMagick^{TM}$: (a) time only watermark (b) watermark with both time and mnemonic, in this image the mnemonic is *Kakkajee*.

Framework Description for Viwid

On the publicly available web pages, the logo images display the date and time of the day as a visible watermark. An example is shown in Figure 6.2 (a). In these logo images, date and time are periodically updated to show the current time according to the user's time zone. The user will be trained to expect to see the current date and time as a visible watermark on the publicly available web pages.

When the user establishes an account with the legitimate company, he is asked to select a mnemonic. We assume that there is a secure connection between the web server and the client side at that time to prevent the disclosure of the mnemonic to eavesdroppers. When cookies are enabled at the user's machine, the web site can use it to recognize the user the next time he is visiting the site. Using the cookie information, the web site knows which mnemonic to embed as a watermark in the logo images without authenticating the user. On the other hand, if cookies are disabled, then the mnemonic can only be added to the visible watermark after the user logs into the established account. This is a less satisfactory form of protection, as the alarm comes after the user has given his login and password. An example of a logo image carrying both the time stamp and the mnemonic is shown in Figure 6.2 (b).

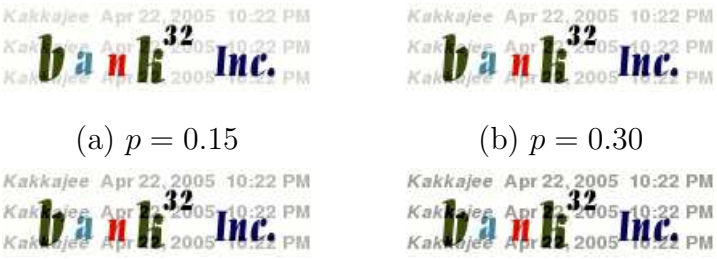
In order to make the user expect these watermarks, the companies need to display messages that remind the user to verify the validity of the watermark displayed on the logo images. An example login page can be seen in Figure 6.1.

6.2.3 Experimental Setup and Results

We collected logo images from randomly selected web pages of 60 Fortune 500 companies and the Center for Education and Research in Information Assurance and Security (CERIAS). All of these logo images were colored Graphics Interchange Format (GIF) images. GIF is the preferred format for displaying logos on web pages because GIF images are 8-bit palette based images, hence their sizes are small. In our experiments, we tested the effectiveness of several visible watermarking algorithms on 61 logo images. The size of these logo images ranges from (18x18) to (760x50).

Even though there is a vast amount of literature on invisible image watermarking techniques, there have been relatively fewer visible image watermarking schemes developed to date [124]. We tested several different visible watermarking techniques on our logo images database. Visibly watermarking color logo images brings many challenges compared to watermarking gray scale images or JPEG images. The main challenge is to maintain the aesthetics of the watermarked logo so as to not to damage its marketing purpose yet be able to insert a robust and readable watermark into it. Moreover, visible watermarking on the logo images is rather less robust because these logo images have large uniform areas and very few objects in them. Besides these the time and memory requirements of the watermarking operation should be very low in order for the web server to be able to dynamically update the time stamp on the logo images frequently. We used the following two techniques in order to verify the applicability. In all these tests, we used a watermark image that is the same size as the cover image.

- $ImageMagick^{TM}$**'s embedded watermarking module [125]** $ImageMagick^{TM}$ is a free software suite for the creation, modification and display of bitmap im-



(a) $p = 0.15$ (b) $p = 0.30$

(c) $p = 0.30$ (d) $p = 0.40$

Fig. 6.3. Logo images watermarked with $ImageMagick^{TM}$ using various p values

ages. $ImageMagick^{TM}$ version 6.2.0 watermarking scheme updates brightness component of HSB color space of every pixel in the cover image using the following equations to embed the watermark:

$$B'_{i,j} = B_{i,j} + \frac{(p \cdot offset^{w}_{i,j})}{midpoint} \tag{6.4}$$

where $B'_{i,j}$ is the brightness of the watermarked image pixels, and $B_{i,j}$ is the brightness of the cover image pixels.

$$offset^{w}_{i,j} = I^{w}_{i,j} - midpoint \tag{6.5}$$

where $I^{w}_{i,j}$ is the intensity of the watermark image pixels.

$$midpoint = \frac{maxRGB}{2} \tag{6.6}$$

$maxRGB$ is the maximum value of a quantum, where a quantum is one of the red, green, or blue elements of a pixel in the RGB color space. In our experiments, $ImageMagick^{TM}$ was compiled with 16 bits in a quantum, thus giving $maxRGB$ equal to 65x535.

p is a user selected parameter for the percentage brightness of the watermark pixel. An example of this embedding with $p = 0.3$ can be seen in Figure 6.2.

Hue and saturation of the cover image are not affected in the watermark embedding process. The value of the p parameter controls the visibility of the watermark. Figure 6.3 shows and example of the watermark embedding where the same watermark is embedded with varying p.

In order to preserve the aesthetics of the cover logo image, we used RGB (midpoint, midpoint, midpoint) as the background color in our watermark images. This is because, with these RGB values, the corresponding $offset^w$ values, in Equation 6.5, become 0.

We have observed that the background color, the text geometry on the watermark image and parameter p have to be adjusted according to the cover image properties in order to reach an acceptable level of watermarked image quality. Figure 6.4 shows examples of (a) a light and (b) a dark background logo images watermarked. In both Figure 6.4 (a) and (b) background of watermark image is RGB (midpoint, midpoint, midpoint) and $p = 0.40$. The color of the text of watermark image is black in Figure 6.4 (a), and white in Figure 6.4 (b).

- **Mohanty et al.'s approach [126]** In their visible watermarking scheme, the modification of the gray values of the host image is based on its local as well as global statistics.

$$I'_n = \alpha_n \cdot I_n + \beta_n \cdot I^w_n \tag{6.7}$$

where I'_n is the intensity of the n_{th} block of the watermarked image. I_n and I^w_n are the corresponding intensity values of the cover and watermark images respectively. α_n and β_n are the scaling and embedding factors depending on the mean and the variance of each block, and the image mean gray value. In [126], it is stated that for color images the watermark should be put in the Y component (luminance). However, when this approach is applied on logo images with white background, even a small change in the luminosity of the background will disturb the aesthetics of the logo image. An example of this

(a) (b)

Fig. 6.4. Logo images watermarked with $ImageMagick^{TM}$ parameter $p = 0.40$ (a) a white background and (b) a dark background

phenomenon can be seen in Figure 6.5 (a). On the other hand, logo images with dark background gave better results, see Figure 6.5 (b) for an example.

However, we observed that the K component of the $CMYK$ colormap can also be used to insert the watermark into logo images. This modified approach gave us better results on logo images with white background, see Figure 6.6.

We are not able to provide samples from the watermarked version of the logo images we collected from Fortune 500 companies' web pages due to copyright issues. In addition, there is a quality loss in the displayed images through out the chapter due to the conversion from GIF to Post Script (PS) format. In order to provide GIF versions of the watermarked logo images and a controlled access to these logo images.

6.2.4 Security Analysis and Discussion

A phisher can try to break the above system through the following three attacks. First attack is to insert a valid watermark message after removing the existing watermark from the logo image. The second attack is to recreate the logo image from scratch and later insert a valid watermark message. The third attack is to perform a man-in-the-middle attack. We explain below why these attacks are not easy for an attacker to carry out.

Success of the first attack depends on the robustness of the underlying visible watermarking algorithm and on the success of the phisher at generating the valid watermark messages for the targeted users.

(a) (b)

Fig. 6.5. Logo images watermarked with Mohanty et al.'s watermarking algorithm (a) a white background and (b) a dark background

Huang and Wu , in [124], show successful attacks on well known visible watermarking systems [126, 127] with the help of human intervention. Huang and Wu's system requires the shapes of the watermark patterns to be marked manually. Results in [124] show that the image inpainting techniques are very effective in removing simple watermark patterns composed of thin lines or symbols. For more sophisticated watermark patterns such as thick lines or bold faced and multi-textured text, Huang and Wu propose an improved scheme where thick watermarked areas are classified into edges and flat areas. Later flat watermarked areas are recovered by refilling them with unaltered flat neighbours. Edged watermarked areas are recovered by approximated prediction based on adaptation information of nearby pixels.

However, in ViWiD, even if the attacker is able to remove the watermark successfully from the watermarked image, he can not insert a completely valid watermark message. The valid watermark message consists of the date and local time of the day for the user's time zone, and the user's mnemonic. The mnemonic is unique for every user and the attacker does not have access to any user's mnemonic. If he can have such access, his attack ceases to be a "one-size-fits-all", and thus we have succeeded in increasing the attacker's cost.

The second attack, which requires recreating the logo image from scratch, can also be thwarted by the fact that the attacker is unable to generate the valid watermark message for every user.

The man-in-the-middle attack is one of the most successful ways of gaining control of customer information [128]. However, besides directing the user to his machine through social engineering, it is difficult for the phisher to be successful in this attack.

(a) (b)

Fig. 6.6. Logo images watermarked with modified version of Mohanty et al.'s watermarking algorithm (a) Time only watermarked logo image (b) Watermarked logo image with Time and Mnemonic

He has to either manipulate the DNS or proxy data on the user's machine, or locate the attacking machine on the real company's web server's network segment or on the route to the real company's web server. Even if the phisher performs a man-in-the-middle attack in order to bring a fresh logo every time a user requests the phisher's web page, the web site would only provide the logo specifically watermarked for the time zone that is assigned to the attacker's IP address. In such a case the attacker would need to have available as many man-in-the-middle's as the number of time zones he wants to attack.

6.2.5 Summary

We have presented a defense system, ViWiD, that mitigates phishing attacks through integrity checking of web site logos using visible watermarking techniques. The valid watermark message consists of the date and local time of the day for the user's time zone, and the user's mnemonic. The watermark message is designed to be unique for every user and carries a shared secret between the company and the user in order to thwart the "one size fits all" attacks.

Unlike the other systems proposed for preventing phishing attacks, ViWiD performs all of the computation on the company's web server and does not require installation of any tool or storage of any data, such as keys or history logs, on the user's machine. ViWiD also involves the user in the integrity checking process, which

makes it harder for the phisher to engineer an attack, since the integrity checking mechanism is not fully automated.

One of the pre-requisites of the proposed scheme is that it requires the users to be trained to expect a valid message to be displayed on the logo images when they perform sensitive transactions. Users are also provided the opportunity to adjust the parameters of the watermark and logo image according to their reading needs and appeal. For example, a user might select a larger font size for the embedded watermark message, or he can as well select a larger logo image.

A large scale user study for validating the effectiveness of our approach is needed as a future work. In addition to that, the robustness of the watermarking techniques can be improved by using high quality logo images in JPEG format or by spreading the message over all images in a web page.

7. PREVIOUS WORK IN INFORMATION HIDING INTO NATURAL LANGUAGE TEXT

7.1 Previous Approaches to Natural Language Steganography

Compared to similar work in the image and video domain, work in natural language watermarking and steganography has been scarce. The previous work in this area has concentrated on natural language steganography. This is probably due to the fact that it is hard to derive robust watermarking methods for text, which will be discussed in more details in the following sections.

A typical scenario for steganography is the case of two parties who want to exchange digital objects through a public communication channel; however, they do not want the existence of this covert communication to be detected by third parties. They sure do not want to achieve confidentiality through encryption, because even the exchange of encrypted messages would reveal the existence of their secret communication, which is why some authoritarian countries forbid the exchange of encrypted messages. In this case, a steganographic algorithm is employed to embed secret messages into innocent looking cover objects to obtain *stego-objects*, while trying to keep statistical properties of these stego-objects as close to natural objects as possible. Later, these stego-objects are exchanged through a communication channel (possibly a public one). While traversing a communication channel, the stego-objects may be subject to intentional or unintentional attacks. Examples of unintentional attacks are transmission errors, changing the visual properties of the stego-document or in the case of pure natural language text, an unintentional attack would be automatic spelling correction. Intentional attacks, on the other hand, are deliberate attempts to distinguish stego-objects from unmodified objects and thus detect the presence

of covert communication. Attack methods for steganographic systems are generally based on detecting the small statistical deviations due to embedding [5].

7.1.1 Using Probabilistic Context-Free Grammars to Generate Cover Text

A *probabilistic context-free grammar* (PCFG) is a commonly used language model where each transformation rule of a context-free grammar has a probability associated with it [53]. A PCFG can be used to generate strings by starting with the root node and recursively rewriting it using randomly chosen rules. Conversely, a string belonging to the language produced by a PCFG can be parsed to reveal the sequence of possible rules that produced it. The PCFG can be changed through different messages in order to be stealthy against statistical attacks.

In the mimicry text approach described in [3], a cover text is generated using a PCFG that generates strings with statistical properties close to normal text. This is achieved by assigning a Huffman code to each grammar rule based on the probability of the rule. The payload string is then embedded by choosing the grammar rule whose code corresponds to the portion of the message being embedded. The PCFG and the corresponding rule probabilities are learned using a corpus.

One drawback of this method is the need for training a PCFG that models natural language to the extend that meaningful texts (containing several sentences) can be generated using this model. Due to this hardship cover texts produced by PCFGs, with limited coverage, tend to be nonsensical to a human reader. Therefore, this method can only be used in communication channels where computers act as wardens.

7.1.2 Information Embedding through Synonym Substitutions

In the T-Lex system [15] a subset of words from the text are selected and the synonym set of each selected word is determined using Wordnet. The synonyms in

each set are indexed in alphabetical order. A simplified example of this embedding is given in [129] as follows: Suppose we have the sentence,

$$\texttt{Midshire is a} \left\{ \begin{array}{cc} 0 & wonderful \\ 1 & decent \\ 2 & fine \\ 3 & great \\ 4 & nice \end{array} \right\} \texttt{little} \left\{ \begin{array}{cc} 0 & city \\ 1 & town \end{array} \right\},$$

where the words in the braces form the synonym sets of two words in the original sentence. If the current string to be embedded is $(101)_2 = 5$, it is first represented in mixed radix form as

$$\left(\begin{array}{cc} a_1 & a_0 \\ 5 & 2 \end{array} \right) = 2a_1 + a_0 = 5,$$

with the constraints $0 \leq a_1 < 5$ and $0 \leq a_0 < 2$. Thus, we obtain the values $a_1 = 2$ and $a_0 = 1$ which indicates that we should use the words *fine* and *town* in the modified sentence.

Three examples of problematic modifications made by the T-Lex system, when a short message is embedded into Jane Austen's novel *Pride and Prejudice*, are shown below. The first sentence fragment is the original version and the second is the steganographically modified version.

. . . I can tell you, to be	making	new acquaintances every day . . .
. . . I can tell you, to be	fashioning	new acquaintances every day . . .
An invitation to dinner was soon	afterwards	dispatched;
An invitation to dinner was soon	subsequently	dispatched;
. . . and make it still better, and say	nothing	of the bad–belongs to you alone.
. . . and make it still better, and say	nada	of the bad–belongs to you alone.
Bingley likes your sister	undoubtedly;	
Bingley likes your sister	doubtless;	

The above examples illustrate two shortcomings of the T-Lex system. First, it sometimes replaces words with synonyms that do not agree with correct English usage, as seen in the phrase *soon subsequently dispatched*. Second, T-Lex also substitutes synonyms that do not agree with the genre and the author style of the given text. It is clear that the word *nada* does not belong to Jane Austen's style. Furthermore, the string *say nada of* is not part of typical English usage in literature writings.

Both types of errors made by the T-Lex system are caused by the fact when choosing synonyms from synsets, important factors such as genre, author style, and sentence context are not taken into account. Synonyms that are not frequently used in common texts of that style can be detected using language models trained on a collection of typical text that has the same genre and style as the one being analyzed. This shortcoming is not unique to the T-Lex system but is a problem with all synonym substitution methods. One can argue that these systems may be improved by making use of information derived from language models during the embedding process. However, such synonym substitution methods would have high computational complexity.

7.1.3 Generating Cover Text using Hybrid Techniques

The *NICETEXT* system [20,130] is a steganography system that generates natural-like cover text according to a given message string. It uses a mixture of lexical transformations and the PCFG technique to generate cover text. The system has two components: a dictionary table and a style template. The dictionary table is a large list of (*type,word*) pairs where the *type* may be based on the part-of-speech [130] of *word* or its synonym set [20]. Such tables may be generated using a part-of-speech tagger or Wordnet. The style template, which is conceptually similar to the PCFG of Section 7.1.1, improves the quality of the cover text by helping generation of natural (i.e., expected) sequences of part-of-speech while controlling the word generation, capitalization, punctuation, and white space. Different style templates may be learned using online corpora (such as Federal Reserve Board meeting minutes or Aesop's Fables) and employed in the system.

7.1.4 Translation Based Steganography

This work [4], investigates the possibilities of steganographically embedding information in the noise created by machine (i.e., automatic) translation of natural language documents. Current state-of-the-art machine translation systems inherently create plenty of room for variation, which makes it ideal to use them for steganography. Since there are frequent errors in legitimate automatic text translations, additional errors inserted by steganographic system appear to be part of the normal noise associated with translation and would be hard to detect.

The LiT system proposed in [4] uses the keyed hash of translated sentences in order to encode information. LiT employs many machine translation systems in order to generate variations of translations of sentences from a given cover text.

LiT suffers from the fact that an adversary can perform statistical analysis to learn the language models of full texts generated by machine translation systems in

order to detect a stego-text that does not carry the statistical properties of being translated by using only one machine translation system but several of them.

7.2 Previous Approaches to Natural Language Watermarking

To the best of authors' knowledge the idea of using linguistic transformations for natural language watermarking was first mentioned by Bender *et al.* in [21]. The first published implementation for natural language watermarking came in 2000, and there was 4.5 publications appearing on the average per year since 2001 [131].

7.2.1 Synonym Substitution Based on Quadratic Residues

Atallah *et al.* [22] devised a watermarking system that uses ASCII values of the words for embedding watermark information into text by performing synonym substitution.

Let $m_{i \bmod k}$ be the bit of watermark message that is to be embedded and w_i be the current word being considered in the cover text with ASCII value $A(w_i)$. If $m_{i \bmod k} = 1$ and $A(w_i) + r_{i \bmod k}$ is a quadratic residue modulo p, then w_i is kept same. Otherwise it is modified. Here p is a 20 digit prime key, k is the number of bits in the watermark message, and $r_0, r_1, ..., r_{k-1}$ is a sequence of pseudo-random numbers generated using p as seed.

This system does not provide much security against an active adversary that uses synonym substitution in order to add another watermark of her/his own to overwrite the previously embedded watermark or just to scramble it to be unrecoverable.

7.2.2 Embedding Information in the Tree Structures of Sentences

In later work [9, 10] Atallah *et al.* have proposed two algorithms that embed information in the tree structure of sentences. These techniques modify the deep structure of sentences in order to embed the watermark message. In other words, the

watermark is not directly embedded to the text using only the surface properties, as is done in lexical substitution, but to the parsed representation of sentences. Utilizing the deep structure makes these algorithms more robust to attacks that built language models based on the surface text properties of the marked documents and make it harder for the adversary to overwrite or modify the embedded information.

The difference between these two algorithms is that the first one modifies syntactic parse trees of the cover text sentences for embedding the watermark message, while the second one uses semantic tree representations. Examples of a *syntactic tree* can be seen below:

```
I took the book.
(S (NP I) (VP took (NP the book)) (. .))

The book was taken by me.
(S1 (S(NP (DT The) (NN book))
      (VP (VBD was) (VP (VBN taken) (PP (IN by) (NP (PRP me)))))
      (. .)))
```

By contrast a *semantic tree* is a tree-structured representation that is imposed over the flat text meaning representation of a sentence [10]. Such representations of sentences may be generated by using ontological semantics resources [132].

Selection of sentences that will carry the watermark information depends only on the tree structure and proceeds as follows: The nodes of the tree T_i for sentence s_i of text are labeled in pre-order traversal of T_i. Then, a node label j is converted to 1 if $j + H(p)$ is a quadratic residue modulo p, and to 0 otherwise. p is a secret key and $H()$ is a one-way hash function. A binary sequence, B_i, for every sentence, s_i, is then generated by traversing T_i following a post-order traversal. A rank, d_i, is then derived for each sentence for s_i using $d_i = H(B_i) \,\mathrm{XOR}\, H(p)$ and the sentences are sorted by their rank. Starting from the least-ranked sentence s_j, the watermark is inserted to s_j's successor in the text. The sentence s_j is referred as a *marker*

sentence, since it points to a watermark carrying sentence. Watermark insertion continues with the next sentence in the rank-ordered list. Once the mark carrying sentences are selected, the bits are embedded by applying either a *syntactic* or a *semantic transformation*, which are explained in detail in Chapter 2.

These studies were important first steps but (unlike the present paper) had the following drawbacks:

- They used only one feature of the sentence to both select and embed, thereby implying that a sentence could not do both (it was the sentence that comes immediately after a selected sentence that carried embedded information).

- The above-mentioned requirement for immediate proximity between a select-marked sentence and its corresponding message-carrying sentence, implies not only lower embedding capacity, but also an increased vulnerability to re-ordering of sentences, selection of a subset of sentences, as well as insertion of new sentences.

- The proximity was actually not the only (or even the main) source of such vulnerability in these previous schemes: A more serious one was the fact that a random change in any sentence had a probability of around $|M|/n$ of damaging an embedded bit. This is negligible only for very large texts ($n \gg |M|$).

- The previous work required fully automated semantic parsing and co-reference resolution, which current natural language processing technology does not satisfactorily provide (it is currently very domain-specific and hence not widely applicable).

7.2.3 Linguistically Analyzing the Applicability of Sentence Level Transformations for Natural Language Watermarking

Another work that deals with sentence level syntactic watermarking is by Murphy [17]. His thesis presents the results of linguistic analysis of several sentence level

Table 7.1
Brian Murphy presented these results on the evaluation of linguistic transformations in [17] on the SUSANNE corpus

	Ubiquity	Reliability
Conjunct Movement	1/15	87%
Switching Relativiser	1/30	94%
Dropping Relativiser	1/30	98%
Expanding Relatives	1/40	53%
Verb Particle Movement	1/65	80%
Adjunct Movement	1/90	94%
Extraposition	1/100	88%
Reducing Relatives	1/160	90%
There Insertion	1/300	100%
Object Raising	1/800	87%
Subject Raising	1/14019	97%

syntactic transformations (including adjunct movement, adjective reordering, verb particle movement) on a hand parsed corpus of 6000 sentences [133]. This work provides the first detailed insight into applicability and coverage of several sentence level transformations for information hiding purposes, see Table 7.1 for the results presented in [17]. In our experiments, we have evaluated the ubiquity (i.e., coverage) of passivization to be 1/34 and reliability (i.e., applicability) of our implementation of passivization to be 37%; the coverage of activization was 1/69 and applicability of our implementation of this transformation was 42%.

Murphy also provides a detailed analysis of the challenges that are involved in writing a generic transformation rule for a natural language. The number of transformations that were analyzed was limited due to the fact that transformations were performed without the use of a surface level generator, thus they mainly cover the

transformations that re-orders the words in a sentence (e.g. adjunct movement) or adds a fixed structure to a sentence (e.g. clefting) or removes a fixed structure from the sentence(e.g. *that/who be* removal).

8. CONCLUSION

This dissertation has demonstrated the feasibility of practical and resilient natural language watermarking while preserving the meaning and grammaticality of the cover text. Other contributions include techniques for special classes of text; exploring different watermarking architectures; and quantification of the distortion introduced by the marking process.

Our work was an important step in the direction of information hiding in natural language text, but much remains to be done. A major remaining challenge is achieving higher embedding bandwidth, an issue that is particularly important (and difficult) for short text such as news briefs, communiquès, etc. Another challenge is extending the preliminary quantification work we started: not only quantification of the distortion, but broader metrics that are context dependent. For example, a specific modification to text might be completely innocuous in one text, yet completely unacceptable in another context (e.g., typing errors in a government press release would not be acceptable).

Looking forward, the Internet has dramatically increased the importance of natural language processing techniques, which stand to revolutionize the way humans interface with the network. Perhaps the most promising impact of information hiding will lie in the tagging of documents; through the use of watermarking a document can carry copyright or meta-data information (about its author, its integrity, generation date, access rights, etc.) on itself as an elemental part of its content.

LIST OF REFERENCES

LIST OF REFERENCES

[1] M. Topkara, C. M. Taskiran, and E. Delp, "Natural language watermarking," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, 2005.

[2] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proceedings of the IEEE*, vol. 87, pp. 1181– 1196, July 1999.

[3] P. Wayner, "Mimic functions," *CRYPTOLOGIA*, vol. XVI, pp. 193–214, July 1992.

[4] R. Stutsman, M. J. Atallah, C. Grothoff, and K. Grothoff, "Lost in just the translation," in *Proceedings of 21st Annual ACM Symposium on Applied Computing*, (Dijon, France), April 2006.

[5] J. Fridrich and M. Goljan, "Practical steganalysis of digital images – State of the art," in *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, (San Jose, CA, USA), January, 2002.

[6] O. Uzuner and B. Katz, "Style vs. expression in literary narratives," in *Proceedings of the Twenty-eighth Annual International ACM SIGIR Conference Workshop on Using Stylistic Analysis of Text for Information Access*, 2005.

[7] R. Soricut and D. Marcu, "Stochastic language generation using WIDL - Expressions and its application in machine translation and summarization," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, 2006.

[8] B. Lavoie and O. Rambow, "A fast and portable realizer for text generation systems," in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, (Washington, DC), 1997.

[9] M. Atallah, V. Raskin, M. C. Crogan, C. F. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik, "Natural language watermarking: Design, analysis, and a proof-of-concept implementation," in *Proceedings of the Fourth Information Hiding Workshop*, vol. LNCS 2137, (Pittsburgh, PA), 25-27 April 2001.

[10] M. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, U. Topkara, and K. E. Triezenberg, "Natural language watermarking and tamperproofing," in *Proceedings of the Fifth Information Hiding Workshop*, vol. LNCS 2578, (Noordwijkerhout, The Netherlands), 7-9 October 2002.

[11] M. Topkara, G. Riccardi, D. Hakkani-Tur, and M. J. Atallah, "Natural language watermarking: Challenges in building a practical system," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, 2006.

[12] U. Topkara, M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions," in *Proceedings of ACM Multimedia and Security Workshop*, (Geneva, Switzerland), September 26-27, 2006.

[13] M. Topkara, U. Topkara, and M. J. Atallah, "Information hiding through errors: A confusing approach," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, January 2007.

[14] B. Murphy, "The syntax of concealment: Reliable methods for plain text information hiding," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, January 2007.

[15] K. Winstein, "Lexical steganography through adaptive modulation of the word choice hash," 1998. http://www.imsa.edu/ keithw/tlex/.

[16] M. Topkara, U. Topkara, and M. J. Atallah, "Words are not enough: Sentence level natural language watermarking," in *Proceedings of ACM Workshop on Content Protection and Security (in conjuction with ACM Multimedia)*, (Santa Barbara, CA), October, 2006.

[17] B. Murphy, "Syntactic information hiding in plain text," Master's thesis, CLCS, Trinity College Dublin, 2001.

[18] A. S. R. L. Rivest and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications*, 1978.

[19] D. S. W. S. Schleimer and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 76–85, 2003.

[20] M. Chapman and G. Davida, "Plausible deniability using automated linguistic stegonagraphy," in *roceedings of the International Conference on Infrastructure Security*, (Bristol, UK), pp. 276–287, October 1-3 2002.

[21] 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.

[22] M. Atallah, C. McDonough, S. Nirenburg, and V. Raskin, "Natural Language Processing for Information Assurance and Security: An Overview and Implementations," in *Proceedings 9th ACM/SIGSAC New Security Paradigms Workshop*, (Cork, Ireland), pp. 51–65, September, 2000.

[23] V. Ng and C. Cardie, "Improving machine learning approaches to coreference resolution," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 104–111, 2001.

[24] U. Topkara, M. J. Atallah, and M. Topkara, "Passwords decay, words endure: Secure and re-usable multiple password mnemonics," in *ACM Symposium on Applied Computing*, 2007.

[25] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::Similarity - Measuring the Relatedness of Concepts," in *Proceedings of Fifth Annual Meeting of the NAACL*, (Boston, MA), May 2004.

[26] P. Moulin and J. A. O'Sullivan, "Information-theoretic analysis of information hiding," *IEEE Transactions on Information Theory*, vol. 49, pp. 563–593, 2003.

[27] M. D. Kernighan, K. W. Church, and W. A. Gale, "A spelling correction program based on a noisy channel model," in *Proceedings of the 13th conference on Computational linguistics*, (Morristown, NJ, USA), pp. 205–210, Association for Computational Linguistics, 1990.

[28] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[29] XTAG, Research, and Group, "A lexicalized tree adjoining grammar for English," Tech. Rep. IRCS-01-03, IRCS, University of Pennsylvania, 2001.

[30] I. Cox, M. Miller, and J. A. Bloom, *Digital Watermaking*. Morgan Kaufmann Publishers, 2002.

[31] A. Pfitzmann and A. Westfeld, "Attacks on steganographic systems," in *Third Information Hiding Workshop*, vol. LNCS, 1768, (Dresden, Germany), pp. 61–76, Springer-Verlag, 1999.

[32] N. Provos, "Defending against statistical steganalysis," in *Proceedings of* 10^{th} *USENIX Security Symposium*, (Washington DC, USA), August 2001.

[33] H. Farid, "Detecting steganographic message in digital images," Tech. Rep. TR2001-412, Darmouth College, Computer Science, Hanover, NH, USA, 2001.

[34] S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *Proceedings of the Fifth Information Hiding Workshop*, vol. LNCS, 2578, (Noordwijkerhout, The Netherlands), Springer-Verlag, October, 2002.

[35] J. Fridrich and M. Goljan, "Practical steganalysis of digital images - state of the art," in *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, vol. 4675, (San Jose, CA, USA), pp. 1–13, January, 2002.

[36] R. J. Anderson and F. A. Petitcolas, "On the limits of steganography," *IEEE Journal of Selected Areas in Communications*, vol. 16, pp. 474 – 481, May 1998.

[37] J. Zollner, H. Federrath, H. Klimant, A. Pfitzmann, R. Piotrascke, A. Westfeld, G. Wicke, and G. Wolf, "Modeling the security of steganographic systems," in *Second Information Hiding Workshop*, vol. LNCS, 1525, (Portland, Oregon, USA), Springer-Verlag, 1999.

[38] M. Topkara, U. Topkara, C. Taskiran, E. Lin, M. Atallah, and E. Delp, "A hierarchical protocol for increasing the stealthiness of steganographic methods," in *Proceedings of the ACM Multimedia and Security Workshop*, (Magdeburg, Germany), 20–22 September 2004.

[39] J. Fridrich and M. Goljan, "Digital image staganography using stochastic modulation," in *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, vol. 5020, (San Jose, CA), pp. 191–202, 21 – 24 January 2003.

[40] A. Westfeld, "F5 – A steganographic algorithm: High capacity despite better steganalysis," in *Fourth Information Hiding Workshop*, vol. LNCS, 2137, (Pittsburgh, USA), pp. 289–302, Springer-Verlag, April 2001.

[41] J. Fridrich, M. Goljan, and D. Hogea, "New methodology for breaking steganographic techniques for jpegs," in *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, vol. 5020, (San Jose, CA), pp. 143–155, 21 – 24 January 2003.

[42] P. Sallee, "Model-based steganography," in *International Workshop on Digital Watermarking*, (Seoul, Korea), 20–22 October 2003.

[43] J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 102–110, March 2006.

[44] S. Katzenbeisser and F. Petitcolas(Ed.), *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[45] C. M. Taskiran, U. Topkara, M. Topkara, and E. Delp, "Attacks on lexical natural language steganography systems," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, 2006.

[46] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of 16th International Conference on Machine Learning*, (Bled, SL), pp. 200–209, 1999.

[47] I. Avcibas, N. Memon, and B. Sankur, "Steganalysis of watermarking and steganographic techniques using image quality metrics," *IEEE Transactions on Image Processing*, vol. 2, no. 12, pp. 221–229, 2003.

[48] K. Winstein, "Lexical steganography at http://alumni.imsa.edu/ keithw/tlex/," March, 1999.

[49] C. Ellison and B. Schneier, "Inside risks: Risks of PKI: secure email," *Communications of the ACM*, vol. 43, no. 1, p. 160, 2000.

[50] M. Topkara, A. Kamra, M. Atallah, and C. Nita-Rotaru, "Viwid: Visible watermarking based defense against phishing," in *International Workshop on Digital Watermarking*, (Siena, Italy), 15–17 September 2005.

[51] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons Inc., 2001.

[52] D. Jurafsky and J. Martin, *Speech and Language Processing*. Upper Saddle River, New Jersey: Prentice-Hall, Inc, 2000.

[53] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[54] The Linguistic Data Consortium, "http://www.ldc.upenn.edu/."

[55] K. Kipper, H. T. Dang, and M. Palmer, "Class-based construction of a verb lexicon," in *AAAI-2000 Seventeenth National Conference on Artificial Intelligence*, (Austin, TX), July 30 - August 3, 2000.

[56] B. Levin, *English Verb Classes and Alternations: A preliminary investigation.* Chicago, IL: University of Chicago Press, 1993.

[57] H. M. Meral, E. Sevinc, E. Unkar, B. Sankur, A. S. Ozsoy, and T. Gungor, "Syntactic tools for text watermarking," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, January 2007.

[58] R. Bergmair and S. Katzenbeisser, "Towards human interactive proofs in the text-domain," in *Proceedings of the 7th Information Security Conference*, vol. 3225, pp. 257–267, Springer Verlag, September, 2004.

[59] N. Ide and J. Vronis, "Word sense disambiguation: The current state of the art," *Computational Linguistics*, vol. 24, no. 1, 1998.

[60] D. Chiang, "Data oriented parsing csli publications," in *Information and Communications Security — First International Conference*, pp. 299– 316, 2000.

[61] E. Charniak, "A maximum-entropy-inspired parser," in *Proceedings of the first conference on NAACL*, pp. 132–139, 2000.

[62] M. Collins, "Head-driven statistical models for natural language parsing," *Computational Linguistics*, vol. 29, Issue 4, December 2003.

[63] D. Grinberg, J. Lafferty, and D. Sleator, "A robust parsing algorithm for LINK grammars," Tech. Rep. CMU-CS-TR-95-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.

[64] F. Xia and M. Palmer, "Converting dependency structures to phrase structures," in *Proceedings of the Human Language Technology Conference*, (San Diego, CA), 18–21 March 2001.

[65] A. Stolcke, "SRILM - An extensible language modeling toolkit," in *Proceedings of International Conferrence on Spoken Language Processing*, 2002.

[66] E. Reiter and R. Dale, *Building natural language generation systems.* Cambridge University Press, 2000.

[67] L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguère, "Bilingual generation of weather forecasts in an operations environment," in *Proceedings of the Thirteenth International Conference on Computational Linguistics*, (Helsinki, Finland), pp. 318–320, 1990.

[68] "http://www.fb10.uni-bremen.de/anglistik/langpro/nlg-table/nlg-table-root.htm."

[69] R. Barzilay and L. Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment," in *Proceedings of NAACL Human Language Technology Conference*, (Edmonton, Canada), 2003.

[70] B. Pang, K. Knight, and D. Marcu, "Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences," in *Proceedings of NAACL Human Language Technology Conference*, (Edmonton, Canada), 2003.

[71] Wikipedia Buffalo Entry, "http://en.wikipedia.org/wiki/buffalo_buffalo_buffalo_buffalo_buffalo_buffalo_buffalo_buffalo."

[72] H. van Halteren, W. Daelemans, and J. Zavrel, "Improving accuracy in word class tagging through the combination of machine learning systems," *Comput. Linguist.*, vol. 27, no. 2, pp. 199–229, 2001.

[73] R. Soricut and D. Marcu, "Towards Developing Generation Algorithms for Text-to-Text Applications," *Ann Arbor*, vol. 100, 2005.

[74] R. Sion, M. Atallah, and S. Prabhakar, "Rights protection for discrete numeric streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, May, 2006.

[75] P. Resnik., "Selectional preference and sense disambiguation," in *the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, (Washington D.C., USA), April 1997.

[76] R. Sion, M. Atallah, and S. Prabhakar, "Power: A metric for evaluating watermarking algorithms," in *IEEE ITCC*, (Las Vegas, Nevada), pp. 95–99, 2002.

[77] P. Resnik., "Using information content to evaluate semantic similarity in a taxonomy.," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453, 1995.

[78] G. Tur, "Turkish text de-asciifier," 1998. http://www.hlst.sabanciuniv.edu/TL/deascii.html.

[79] M. D. Kernighan, K. W. Church, and W. A. Gale, "A spelling correction program based on a noisy channel model," in *Proceedings of the 13th conference on Computational linguistics*, (Morristown, NJ, USA), pp. 205–210, Association for Computational Linguistics, 1990.

[80] L. Philips, "Hanging on the metaphone," *Computer Language Magazine*, vol. 7, no. 12, pp. 39–44, 1990.

[81] S. Turkle, *Life on the Screen: Identity in the Age of the Internet*. New York: Simon and Schuster, 1995.

[82] L. Truss, *Eats, Shoots & Leaves*. New York: Gotham Books, 2004.

[83] D. Crystal, *Language and the Internet*. Cambridge CUP, 2001.

[84] D. Crystal, "Txt, ne1?," in *In Susan Tresman and Ann Cooke (eds), The Dyslexia Handbook*, pp. 179–183, British Dyslexia Association, 2006.

[85] K. Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.

[86] S. Cucerzan and E. Brill, "Spelling correction as an iterative process that exploits the collective knowledge of web users," in *Proceedings of EMNLP 2004*, pp. 293–300, 2004.

[87] R. Bergmair, "Towards linguistic steganography: A systematic investigation of approaches, systems, and issues.," tech. rep., University of Derby, November, 2004.

[88] Reuters Corpus, "http://about.reuters.com/researchandstandards/ corpus/."

[89] CogenTex, "Realpro general English grammar user manual." http://www.cogentex.com/papers/realpro-manual.pdf.

[90] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems," in *Information Hiding, Second International Workshop*, vol. LNCS 1525, (Portland, Oregon), pp. 219–239, 1998.

[91] A. Lang and J. Dittmann, "Profiles for evaluation - the usage of audio wet," in *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents VI*, (San Jose, CA), January 2006.

[92] B. Macq, J. Dittmann, and E. J. Delp, "Benchmarking of image watermarking algorithms for digital rights management," *Proceedings of IEEE*, vol. 92. Issue 6, pp. 971–984, June, 2004.

[93] E. Hovy, M. King, and A. Popescu-Belis, "Principles of context-based machine translation evaluation," *Machine Translation*, vol. 16, pp. 1–33, 2002.

[94] N. I. of Standards and Technology, "Machine translations benchmark tests provided by national institute of standards and technology," in *http://www.nist.gov/speech/tests/mt/resources/scoring.htm*.

[95] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of 40th Annual Meeting of the ACL*, (Philedelphia), July 2002.

[96] G. Doddington, "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics," in *Proceedings of ARPA Workshop on Human Language Technology*, 2002.

[97] C. Grothoff, K. Grothoff, L. Alkhutova, R. Stutsman, and M. Atallah, "Translation-based steganography," in *Proceedings of Information Hiding Workshop (IH 2005)*, p. 15, Springer-Verlag, 2005.

[98] J. Fridrich, M. Goljan, and D. Soukal, "Higher-order statistical steganalysis of palette," in *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, vol. 5020, (San Jose, CA), 2003.

[99] A. Levy and N. Merhav, "An image watermarking scheme based on information theoretic principles," Tech. Rep. HPL-2001-13, HPL Technical Report, January 2001.

[100] H. C. Kim, H. Ogunleye, O. Guitart, and E. J. Delp, "Watermarking evaluation testbed (WET) at Purdue University," in *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. SPIE 5306, (San Jose, CA), 1822 January 2004.

[101] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[102] M. Hart, *Project Gutenberg*. http://www.gutenberg.net/, 2004.

[103] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of lsb steganography in color and grayscale images," in *Proceedings of the ACM Workshop on Multimedia and Security*, (Ottowa, Canada), 2001.

[104] U. Topkara, M. Topkara, and M. J. Atallah, "This message will self-destruct: Self-erasing communication," in *Under Submission*, 2007.

[105] K. Butler, W. Enck, J. Plasterr, P. Traynor, and P. McDaniel, "Privacy-preserving web-based email," in *International Conference on Information Systems Security*, 2006.

[106] I. Goldberg, "Privacy-enhancing technologies for the internet ii: Five years later.," in *Workshop on Privacy-Enhancing Technologies*, 2002.

[107] M. Waldman, A. D. Rubin, and L. F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant, web publishing system," in *Proceedings of 9th USENIX Security Symposium*, pp. 59–72, August 2000.

[108] Google Analytics, "http://www.google.com/analytics."

[109] Extreme Tracking, "http://extremetracking.com."

[110] The Internet Archive, "http://www.archive.org."

[111] Google, "http://www.google.com/."

[112] S. M. Martinez, "Identity theft and cyber crime," September 2004. Federal Bureau of Investigation, http://www.fbi.gov/congress/congress.htm.

[113] "Symantec internet security threat report highlights rise in threats to confidential information," in *http://www.symantec.com/press/2005/n050321.html*.

[114] "The Anti-Phishing working group," in *http://www.antiphishing.org*.

[115] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft," in *Proceedings of the Network and Distributed System Security Symposium*, 2004.

[116] "Netcraft," in *http://www.netcraft.com*.

[117] "ebay: buyer tools: toolbar," in *http://pages.ebay.com/ebay_toolbar/*.

[118] "Preventing man in the middle phishing attacks with multi-factor authentication," in *http://www.tricipher.com/solutions/phishing.html*.

[119] A. Herzberg and A. Gbara, "Trustbar: Protecting (even naíve) web users from spoofing and phishing attacks," in *Cryptology ePrint Archive, Report 2004/155*, 2004.

[120] P. Security, "Protecting your customers from phishing attacks – An introduction to passmarks," in *http://www.passmarksecurity.com/*.

[121] R. Dhamija and J. Tygar, "The battle against phishing: Dynamic security skins," in *Symposium on Usable Provacy and Security (SOUPS)*, July, 2005.

[122] T. Wu., "The secure remote password protocol," in *In Internet Society Network and Distributed Systems Security Symposium (NDSS)*, pp. 97–111, Mar 1998.

[123] N. Memon and P. W. Wong, "Protecting digital media content," *Commun. ACM*, vol. 41, no. 7, pp. 35–43, 1998.

[124] C.-H. Huang and J.-L. Wu, "Attacking visible watermarking schemes," *IEEE Transactions on Multimedia*, vol. 6, no. 1, February, 2004.

[125] "ImageMagick Studio LLC," in *http://www.imagemagick.org*.

[126] S. P. Mohanty, K. R. Ramakrishnan, and M. Kankanhalli, "A dual watermarking technique for images," in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pp. 49–51, ACM Press, 1999.

[127] G. W. Braudaway, K. A. Magerlein, and F. C. Mintzer, "Protecting publicly available images with a visible image watermark," in *Proceedings of the SPIE International Conference on Electronic Imaging*, vol. 2659, (San Jose, CA), February 1-2, 1996.

[128] G. Ollman, "The phishing guide," in *http://www.ngssoftware.com/papers/NISR-WP-Phishing.pdf*.

[129] R. Bergmair, "Towards linguistic steganography: A systematic investigation of approaches, systems, and issues.," tech. rep., University of Derby, August 2004.

[130] M. Chapman and G. Davida, "Hiding the hidden: A software system for concealing ciphertext in innocuous text," in *Proceedings of the International Conference on Information and Communications Security*, vol. LNCS 1334, (Beijing, China), 1997.

[131] R. Bergmair, "A bibliography of linguistic steganography," in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, January 2007.

[132] S. Nirenburg and V. Raskin, *Ontological Semantics*. MIT Press, 2004.

[133] G. Sampson, *English for the computer: the SUSANNE corpus and analytic scheme*. Oxford: Clarendon, 1995.

VITA

VITA

Mercan Topkara received her PhD degree from the Computer Science Department of Purdue University in 2007 and her Bachelor of Science degree from the Computer Engineering and Information Sciences Department of Bilkent University in 2000. She started her graduate studies at Purdue University in August 2001. Her PhD thesis is focused on designing, building and evaluating natural language watermarking systems. Her research interests are within the areas of digital watermarking, statistical natural language processing, usable security and machine learning. She has previously worked as a research intern at AT&T Research Labs, IBM T. J. Watson Research, and Google Research.