CERIAS Tech Report 2004-13

LINGUISTIC STEGANOGRAPHY: SURVEY, ANALYSIS, AND ROBUSTNESS CONCERNS FOR HIDING INFORMATION IN TEXT

by Krista Bennett

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Linguistic Steganography:

Survey, Analysis, and Robustness Concerns for Hiding Information in Text

Krista Bennett
Department of Linguistics
Purdue University
West Lafayette, IN 47906
kbennett@cerias.purdue.edu

Abstract. Steganography is an ancient art. With the advent of computers, we have vast accessible bodies of data in which to hide information, and increasingly sophisticated techniques with which to analyze and recover that information. While much of the recent research in steganography has been centered on hiding data in images, many of the solutions that work for images are more complicated when applied to natural language text as a cover medium. Many approaches to steganalysis attempt to detect statistical anomalies in cover data which predict the presence of hidden information. Natural language cover texts must not only pass the statistical muster of automatic analysis, but also the minds of human readers. Linguistically naïve approaches to the problem use statistical frequency of letter combinations or random dictionary words to encode information. More sophisticated approaches use context-free grammars to generate syntactically correct cover text which mimics the syntax of natural text. None of these uses meaning as a basis for generation, and little attention is paid to the semantic cohesiveness of a whole text as a data point for statistical attack. This paper provides a basic introduction to steganography and steganalysis, with a particular focus on text steganography. Text-based information hiding techniques are discussed, providing motivation for moving toward linguistic steganography and steganalysis. We highlight some of the problems inherent in text steganography as well as issues with existing solutions, and describe linguistic problems with character-based, lexical, and syntactic approaches. Finally, the paper explores how a semantic and rhetorical generation approach suggests solutions for creating more believable cover texts, presenting some current and future issues in analysis and generation. The paper is intended to be both general enough that linguists without training in information security and computer science can understand the material, and specific enough that the linguistic and computational problems are described in adequate detail to justify the conclusions suggested.

Introduction

Steganography is the art of sending hidden or invisible messages. The name is taken from a work by Trithemus (1462-1516) entitled "Steganographia" and comes from the Greek στεγανό-ς, γραφ-ειν meaning "covered writing" (Petitcolas *et al* 1999: 1062, Petitcolas 2000: 2, etc.). The practice of sending secret messages is nothing new, and attempts to cover the messages by hiding

them in something else (or by making them look like something else) have been made fpr millennia. Many of the standard examples used by modern researchers to explain steganography, in fact, come from the writings of Herodotus. For example, in around 440 BC, Herodotus writes about Histæus, who was being held captive and wanted to send a message without being detected. He shaved the head of his favorite slave, tattooed a message on his scalp, and waited for the hair to regrow, obscuring the message from guards (Petitcolas 2000: 3). Petitcolas mentions that this method was in fact still used by Germans in the early 20th century.

Modern steganography is generally understood to deal with electronic media rather than physical objects and texts. This makes sense for a number of reasons. First of all, because the size of the information is generally (necessarily) quite small compared to the size of the data in which it must be hidden (the *cover text*), electronic media is much easier to manipulate in order to hide data and extract messages. Secondly, extraction itself can be automated when the data is electronic, since computers can efficiently manipulate the data and execute the algorithms necessary to retrieve the messages. Also, because there is simply *so much* electronic information available, there are a huge number of potential cover texts available in which to hide information, and there is a gargantuan amount of data an adversary attempting to find steganographically hidden messages must process. Electronic data also often includes redundant, unnecessary, and unnoticed data spaces which can be manipulated in order to hide messages. In a sense, these data spaces provide a sort of conceptual "hidden compartment" into which secret messages can be inserted and sent off to the receiver.

This work provides an introduction to steganography in general, and discusses linguistic steganography in particular. While much of modern steganography focuses on images, audio signals, and other digital data, there is also a wealth of text sources in which information can be hidden. While there are various ways in which one may hide information in text, there is a specific set of techniques which uses the linguistic structure of a text as the space in which information is hidden. We will discuss text methods, and provide justification for linguistic solutions. Additionally, we will analyze the state-of-the-art in linguistic steganography, and discuss both problems with these solutions, and a suggested vector for future solutions.

In section 1, we discuss general steganography and steganalysis, as well as some well-known areas of steganography. Section 2 discusses the main focus of this paper, text steganography in general and linguistic steganography in particular. Section 3 explores the linguistic problems with existing text steganographic methods. Finally, Section 4 gives suggestions for constructing the next generation of linguistically and statistically robust cover texts based upon the methods described in Section 1 and 2, and the issues discussed in Section 3.

1 Steganography, Steganalysis, and Mimicking

Because the focus of this text is on linguistic steganography, it is important to understand just what we mean by this term. Chapman et al define linguistic steganography as "the art of using written natural language to conceal secret messages" (Chapman *et al* 2001: 156). Our definition is somewhat more specific that this, requiring not only that the steganographic cover be composed of natural language text or some sort, but that the text itself is either generated to have a cohesive linguistic structure, or that the cover text is natural language text to begin with. To further elaborate, we will first introduce steganography as a field and discuss current techniques in information hiding. We then show how these are applied to texts, differentiating between non-linguistic and linguistic methods. Section 1.1 describes modern steganography with some examples of steganographic techniques, and defines linguistic steganography within the context of text steganography in general. Section 1.2 introduces steganalysis and adversarial models, which are, in a sense, the driving force behind the creation of new steganographic methods. Finally, section 1.3 discusses "mimicking", which is an encapsulation of the idea of using the statistical properties of a normal data object as the basis for generating a steganographic cover. These are intended as background information in order to motivate the discussion of text steganography and cover generation in section 2.

1.1 Steganography

Steganographic information can be hidden in almost anything, and some cover objects are more suitable for information hiding than others. This section will simply detail a few common steganographic methods applied to various kinds of electronic media, along with an explanation of the steganographic techniques used. Techniques can be grouped in many different ways; Johnson and Katzenbeisser group steganographic techniques into six categories by how the algorithm encodes information in the cover object: *substitution systems, transform domain*

techniques, spread spectrum techniques, statistical methods, distortion techniques, and *cover generation methods* (2000: 43-44). In terms of linguistic steganography, we will be mainly concerned with cover generation methods, although some statistical methods and substitution systems will be described. Substitution systems insert the hidden message into redundant areas of the cover object, statistical methods use the statistical profile of the cover text in order to encode information, and cover generation texts encode information in the way the cover object itself is generated (44). The descriptions that follow are not supposed to be an exhaustive survey, but merely an introduction to some of the existing methods; for a much more comprehensive description of modern steganographic techniques, see Katzenbeisser and Petitcolas (2000) or Wayner (2002).

One further comment should be made; Kerkhoffs' principle, which states that one must assume that an attacker has knowledge of the protocol used and that all security must thus lie in the key used in the protocol, is not to be ignored (Anderson and Petitcolas 1998, Petitcolas 2000). While we do not specifically discuss secret keys here, it should be stated that we assume that the hidden message is encrypted before being hidden in the cover text. While this does not protect a protocol from being attacked if the introduction of random-looking data is inappropriate in the context of the cover data, but it does prevent the message from being read. In many cases, the fact that encrypted data looks like random data is intentionally used in spaces where such random noise could realistically occur. In any event, we assume that the message itself is cryptographically secure, and we therefore focus the protocols intended to hide such messages.

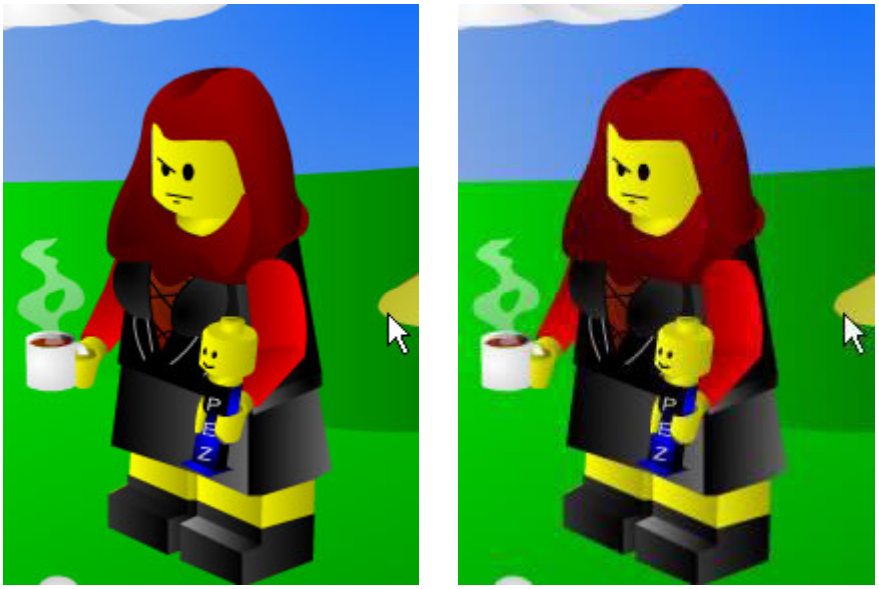
1.1.1 Image steganography

Image steganography has gotten more popular press in recent years than other kinds of steganography, possibly because of the flood of electronic image information available with the advent of digital cameras and high-speed internet distribution. Image steganography often involves hiding information in the naturally occurring "noise" within the image, and provides a good illustration for such techniques.

Most kinds of information contain some kind of noise. Noise can be described as unwanted distortion of information within the signal. Within an audio signal, the concept of noise is obvious. For images, however, noise generally refers to the imperfections inherent in the process

of rendering an analog picture as a digital image. For example, the values of colors in the palette for a digital image will not only not be the exact colors in the real image, and the distribution of these colors will be also be imperfect. As Wayner mentions, the instantaneous measurement of photons made by a digital camera also captures the randomness inherent in their behavior, leading to a set of "imperfect" measurements which balance out to become a digital image (Wayner 2002: 152). By changing the least significant bit (LSB) in the color representation for selected pixels, information can be hidden while often not significantly changing the visual appearance of the image; this is known as "image downgrading" (Katzenbeisser 2000: 29, Johnson and Katzenbeisser 2000: 49). The greater the number of bits used to represent colors, the less obvious the changes in palette values are in the visual representation of the final image. While changing the LSB in order to hide information is a widely used steganographic method, Petitcolas *et al* note that it is "trivial for a capable opponent to remove" such information (1999: 1065). Furthermore, lossy compression and other image transformations can easily destroy hidden messages (Johnson and Jajodia 1998: 30).

There are many other methods for image steganography, however. For the images below, an algorithm was used that attempts to avoid statistical distortion by taking advantage of the discrete cosine transforms that are used to compress and approximate a digital image. The F5 algorithm improves upon previous methods for steganographic JPEGs by not only modifying the compression coefficients, but by attempting to spread the modifications through the file in such a way that their statistical profile still approximates a non-steganographic JPEG image (Westfeld 2001; Wayner 2002: 181). The image below is a bitmap image which is compressed to a JPEG image by the tool as the secret message is embedded.



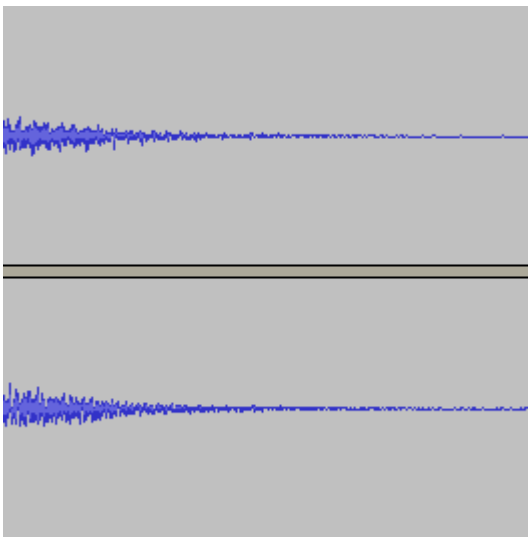
(Original image thanks to *reasonablyclever.com*)

(An original image (182k bitmap – left), and the same image with a 1k file embedded in it using the F5 steganography tool (Westfeld, 2003) – note that the distortions in the background of the second image resemble image distortions commonly seen when converting images; however, the distortion in the two images is also clearly detectable by a human. This might or might not tip off an attacker, depending upon whether or not such image distortion can be expected in context. Furthermore, the statistical profile of the distortion may tell an attacker much more about the stego-object.)

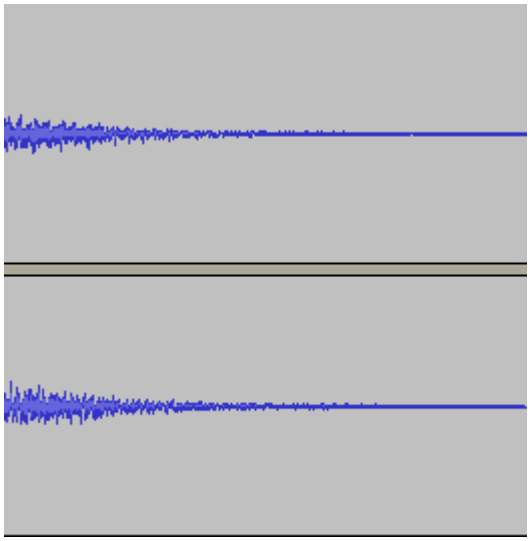
1.1.2 Audio steganography

Audio steganography, the hiding of messages in audio "noise" (and in frequencies which humans can't hear), is another area of information hiding that relies on using an existing source as a space in which to hide information. Audio steganography can be problematic, however, since musicians, audiophiles, and sound engineers have been reported to be able to detect the "high-pitched whine" associated with extra high-frequency information encoded in messages. In addition to storing information in non-audible frequencies or by distorting the audible signal to include additional noise, Johnson and Katzenbeisser also mention a technique known as "echo hiding." An echo is introduced into the signal, and the size of the echo displacement from the original signal can be used to indicate 1's or 0's, depending on the size (2000: 62). Regardless of the kind of signal modification used, as with many steganographic techniques applied to images, changing an existing signal modifies its statistical profile in addition to potentially changing the audible qualities of the signal. Making such steganography less detectable depends on making the changes look like legitimately occurring noise or signal distortion.

The following is a short visual sample of happens when a 13k audio file is embedded into a 168k .wav file using Steganos (Steganos GmbH, 2004); the audio of the steganographic file sounds fuzzy, like a radio that is not well-tuned (the original sound file is quite clear). The representation captured below shows the end of the sound trailing off into silence. While both images are similar, an examination of the steganographic representation shows that there is additional noise (this is made particularly obvious by the fact that line at the end of the right sample is much thicker than that of the left sample):



Clean .wav file (cover object)



Stego .wav file (images from the *Audacity* audio program)

While the original audio is still clearly discernable in the stego-object, the changes are detectable by the human listener. While they fall within the range of the kind of noise heard on a radio station or through interference, if such noise is not expected in context, it may raise questions about the audio file itself.

1.1.3 Text steganography

Text steganography, which is what this paper specifically deals with, uses text as the medium in which to hide information. Our definition of text steganography remains broad in order to differentiate it from the more specific "linguistic steganography". Text steganography can involve anything from changing the formatting of an existing text, to changing words within a text, to generating random character sequences or using context-free grammars to generate readable texts. With any of these methods, the common denominator is that hidden messages are

imbedded in character-based text. What sets each method apart from another, coarsely speaking, is whether or not the text is preexisting or is completely generated by the steganographic tool, and whether or not the resulting text is a result of random generation/modification, "statistical" generation, or linguistically-driven generation/modification. Only linguistically-driven generation and modification of cover texts qualifies as *linguistic steganography* by our definition, and will be further detailed in section 2. Each of these text-based methods has its own unique characteristics, and from an both a linguistic and security standpoint, problems.

To give an idea of what text steganography looks like, here is a short sample of steganographic text generated by a tool called **spammimic**. **spammimic** is a web-based steganography tool which uses a "spam" grammar and Wayner's mimic algorithm for context-free-grammars to produce spam-like steganographic text (spammimic.com, 2000). For example, entering "Lovely spam" as a secret message into the web interface creates the following (lengthy) steganographic message:

> Dear Friend , This letter was specially selected to be sent to you ! This is a one time mailing there is no need to request removal if you won't want any more ! This mail is being sent in compliance with Senate bill 1622 ; Title 6 ; Section 305 ! This is NOT unsolicited bulk mail . Why work for somebody else when you can become rich in 10 days ! Have you ever noticed nearly every commercial on television has a .com on in it and people love convenience . Well, now is your chance to capitalize on this ! WE will help YOU turn your business into an E-BUSINESS and process your orders within seconds ! You are guaranteed to succeed because we take all the risk ! But don't believe us ! Mr Jones of Indiana tried us and says "Now I'm rich, Rich, RICH" ! This offer is 100% legal . Because the Internet operates on "Internet time" you must act now ! Sign up a friend and you'll get a discount of 50% . Thanks .

The message is created through a grammar of spam phrases, each of which expresses a bit or series of bits. In some sense, the **spammimic** variant of Wayner's mimic grammars is one of the more convincing text methods, since so much essentially meaningless spam content is readily available on the internet (one estimate claims (as of May 2004) that 82% of all email in the U.S. was spam (Gaudin 2004)). Wading through the volumes of "legitimate" spam to find hidden messages would be a hefty job indeed; in fact, one of the ways to improve the chances that a steganographic message won't be found is to camouflage the data in such a way that there is a

high cost for the attacker to search for the message (Petitcolas *et al* 1999: 1065). While this "needle in a haystack" metaphor is appropriate and is relied upon by many steganographic methods, given the grammar used to create the message or a known steganographic message created with the grammar, the task becomes significantly less daunting.

1.2 Steganalysis

As with most areas of information security, steganography is an arms race. This is perhaps best exemplified by Ross Anderson's comment that "[e]very few years, computer security has to re-invent itself. New technologies bring new threats, and force us to invent new protection mechanisms" (Anderson 2000: xv). When a weakness in a steganographic protocol is found, new protocols (and modifications to existing protocols) are developed to counteract the weakness. These, in turn, lead to new weakness analyses and attack methods. This cycle continues over time, raising the stakes for new steganographic protocols. What this emphasizes, however, is that it is very difficult to consider protocols without considering how they will be attacked. Furthermore, many ideas for new directions in steganography are arrived from attack analyses. This process of analyzing steganographic protocols in order to detect and extract secret messages is called *steganalysis*. This is the steganographic analogue to *cryptanalysis*, which refers to attempts to break cryptographic protocols. With cryptographic protocols, cryptanalysis is generally considered to be successful if the encrypted message can be retrieved by the adversary. Steganography adds the additional requirement that the steganographically hidden message is not even detectable by the adversary; that is, not only should the attacker not be able to find the message, but he should not even know it exists. Steganalysis, then, is generally considered to be successful when the existence of a message is detected (Zöllner *et al*: 344, Provos 2001: 1). This section briefly discusses steganalysis in order to give background information for robustness concerns discussed later in the text.

1.2.1 Adversarial models

Successful approaches to steganalysis raise the bar for researchers developing steganographic protocols. New protocols must usually be robust enough to resist analysis by existing steganalysis; thus, existing methods of steganalysis frame and define future steganographic protocols. Steganalysis also frames the *adversarial models* used by steganographers. The adversarial model is a theoretical description of the attacker against which the proposed protocol

is to be judged. The description of this adversary describes both his resources and his expected behavior. In the case of text steganography, we will be referring not only to the kinds of computational resources available to the attacker, but also to whether or not the adversary might be a human, a computer, or both. Traditionally, steganalysis has been mostly concerned with computational analysis. Since a computer can often detect statistical anomalies that a human cannot (by hand), and since the data is often electronic, the human factor has often been ignored. However, since it is possible to generate text which "looks" normal to a computer, but which would be instantly recognizable to a human as abnormal (ungrammatical, or nonsensical), the human adversary must be increasingly considered.

1.2.2 Methods of Steganalysis

Johnson lists 6 different kinds of steganographic attacks: *stego-only, known cover, known message, chosen stego, chosen message,* and *known stego* (2000: 81). These methods are partially classified by what the attacker has available to him in analyzing the steganographic text. The following table summarizes what the attacker has available to him in each case:

	stego object	original cover object	hidden message	stego algorithm or tool
stego only	X			
known cover	X	X		
known message	X		X	
chosen stego	X			X
chosen message	X	(see explanation)		
known stego	X	X		X

(generated from Johnson 2000: 81)

While the attack types are essentially explained by the table, it should additionally be noted that the "known message" attack is an attack used to derive information about the algorithm used to create the stego-object with the message in hopes of being able to detect and extract future hidden messages, and the chosen message attack is one in which the attacker feeds different messages to various algorithms in order to see if the stego-object seems to have attributes that indicate the use of one of those algorithms.

This classification is useful in talking about the kind of attacks a particular method is vulnerable to. For example, text methods which modify words in an original cover text become vulnerable to the known cover attack because both the original cover text object and the stego-object are available – comparison of the two will show that the original has been modified and perhaps provide information about the hidden message. In some sense, this set of classifications helps to generally describe the adversarial model one is working with, as well as the kinds of available attacks.

There are other ways to break up attack-types, however, and these are also useful in describing the vulnerabilities of various methods. Wayner divides common attack methods by functional properties rather than adversarial assumptions; attacks are divided into *visual or aural attacks*, *structural attacks,* and *statistical attacks* (2002: 305) Visual and aural attacks describe the human factor in attacks; humans can often perceive the modifications in the cover object because it doesn't look or sound right. In text steganography this can be extended to format-based, lexical, grammatical, semantic, and rhetorical attacks, among others. Structural attacks refer to detecting the patterns in modifications made in the data format (for example, using extra space in files or encoding schemes to store information is often detectable through structural attacks). Statistical attacks detect anomalies in the statistical profile of the stego-object (for example, images whose color palette has been changed to hide information often contain non-standard colors or ranges of colors which would not normally be generated by image software) (see, for example, Provos 2001).

We will use both Johnson's and Wayner's classifications in describing vulnerabilities in existing protocols, as well as for motivation of future work.

1.3 Mimicking

Wayner's 1992 paper entitled "Mimic Functions" took the idea of statistical steganalysis and turned it into an approach for steganographic generation. Because steganalysis has relied heavily detecting data that does not fit the statistical profile of a particular data type, Wayner decided that the statistical profile itself should be used to generate steganographic text. This is called "mimicking", and the functions and algorithms used to generate the data which fit a particular statistical profile are called "mimic functions". Framing the generation of steganographic cover

data as “mimicking” provides a convenient set of formalisms for the generation of seemingly meaningful data which is in fact nonsense, used to cover a steganographic message.
There are, however, issues with assuming that mimicking solves the problem of statistical steganalysis. Successful mimicking requires that one knows the statistical properties that an adversary may be analyzing. A generated text which matches a statistical profile of letter frequencies and word lengths in the English language may not in fact stand up under statistical profiling of syntactic variation or semantic cohesiveness. It is here that the interplay between steganalysis and steganography becomes quite apparent. For every new statistical attack on a mimicking system, a new feature must be added to that system which responds to this attack and makes the generated data more robust. Aura recognizes this interplay in the context of noise-based steganography when he warns that “it is likely that someone with greater resources and more time will be able to develop a better model of the noise and still differentiate between the signal and the replacement” (1996: 268). This applies no less to statistical generation; it is always likely that someone will develop a better statistical model based upon a different set of factors and will be able to detect statistical anomalies. We will specifically be looking at this interaction in regard to generated text, and section 4 offers our suggestions for both the next step in statistical steganalysis of generated texts and for the properties of steganographic systems which respond to such steganalysis.

2 Text steganography

Now that a general description of steganalysis and steganographic methods have been presented, we will focus on text steganography. The wealth of electronic textual information available as well as the difficulty of serious linguistic analysis make this an interesting medium for steganographic information hiding. Text is also one of the oldest media used in steganography; well before the electronic age, letters, books, and telegrams hid secret messages within their texts. In this section, we will explore text steganography in greater depth, discussing the state-of-the-art and introducing existing linguistic methods. This, it is hoped, will give a solid introduction to what linguistic steganography is in the context of information hiding in general, why it exists, and where it is going.

As mentioned earlier, text steganography refers to the hiding of information within text (i.e. character-based) messages. We’ll examine three basic categories of text steganography here:

format-based methods, random and statistical generation, and linguistic methods. Note that within each category, the text can either be generated from scratch or embedded within known plaintext. Specific linguistic issues with text methods will be detailed in section 4 of the paper.

2.1 Format-based methods

Format-based methods use the physical formatting of text as a space in which to hide information. Format-based methods generally modify existing text in order to hide the steganographic text. Insertion of spaces or non-displayed characters, deliberate misspellings distributed throughout the text, and resizing of fonts are some of the many format-based methods used in text steganography. Some of these methods, such as deliberate misspellings and space-insertion, might fool human readers who ignore occasional misspellings, but can often be easily detected by a computer. On the other hand, a computer might not recognize font resizing as a problem, particularly if it is only concentrating on text contents within a document for which text-resizing might be expected (reports with figures, etc.); however, a human might detect strange font sizes almost immediately. Additionally, if the original plaintext is available, comparing this plaintext with the suspected steganographic text would make manipulated parts of the text quite visible. It should be noted the idea of shifting fonts on the page is not a new idea; in the 16th century, Francis Bacon manipulated fonts to hide information in texts (Petitcolas 2000: 6). More recently, Brassil *et al* came up with a line and word shifting scheme in order to mark texts; while their intent was to hide a watermark in order to "deter illicit dissemination", the technique could also hide steganographic information (Brassil *et al* 1994).

2.1.1 Random and statistical generation

In order to avoid comparison with a known plaintext, steganographers often resort to generating their own cover texts. While this often resolves the problem of a known cover attack, the properties of the generated text may still give rise to suspicions that the text is not legitimate. Such generation generally tries to simulate some property of normal text, usually by approximating some arbitrary statistical distribution found in real text. In this section, we detail non-linguistic text generation.

2.1.1.1 Character sequences

The hiding of information within character sequences is appealing because so much character-based information is available and transmitted over networks. One approach to text

steganography might hide information in what appears to be a random sequence of characters. Of course, to both the person sending and receiving the message, this sequence is far from random, but it must appear to be random to anyone who intercepts the message. Not only must it appear to be random, however, but since we are also concerned with detection of the fact that this is a steganographic text, it must not appear to be suspicious. Random sets of characters which all fall within one character set but have no apparent meaning might indeed raise red flags.

Because of this, a second approach to character generation is to take the statistical properties of word-length and letter frequency in order to create "words" (with no lexical value) which will appear to have the same statistical properties as actual words in a given language. These words might convince a computer which is only doing statistical analysis (and this is much less likely now that we are in an age where enormous dictionaries can be used to check the validity of words), but has clear problems with modern computer systems in terms of appearing suspicious.

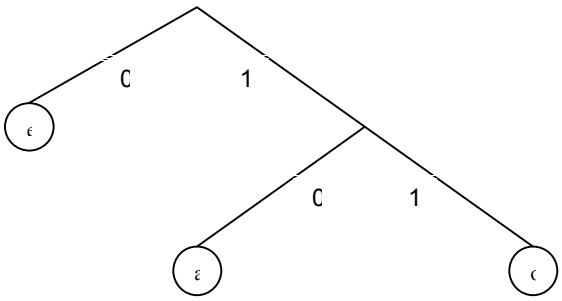
2.1.1.2 Word sequences

To solve the problem of detection of non-lexical sequences, actual dictionary items can be used to encode one or more bits of information per word. This might involve a code-book of mappings between lexical items and bit sequences, or the words themselves (length, letters, etc.) might encode the hidden information. However, this too has problems. A string of words with no semantic structure and no discernable semantic relationship can be detected by both humans and computers alike. Even in situations where computers cannot do complex syntactic analysis, simply classifying words and noticing extremely unlikely patterns (too many verbs or determiners in a row, no prepositions, etc.) within sequences of a certain length may be enough to alert the attacker to anomalous behavior. Even grammar-checkers used by modern word processors may be helpful tools in discovering ungrammatical texts. While legitimate ungrammatical texts certainly exist, given a certain context and threshold, such methods could be used to flag texts with no syntactic structure for further attention.

2.1.1.3 Statistical generation of sequences – text mimicking

In addition to using the statistical frequency of letters or words in order to generate cover text, Wayner proposed a clever method of generating text which can be fairly convincing lexically and syntactically (and often semantically). This is not done by linguistic generation; rather, all

possible strings of length *n* are taken from a text or set of texts which are used as a sort of generator for the cover text. These strings are then organized in a table by how often they occur in the text. One string is picked at random to start the steganographic text, and the next letter in the text is chosen by looking at a window of the last *n – 1* characters in the steganographic text and finding all of the strings in the table which start with those characters. The next letter is chosen from the last letter of these strings by using the data structures from the Huffman compression algorithm in reverse; the statistical frequency of all of the possible next letters that end the strings that start with the desired *n – 1* characters is used to generate a tree which uses the frequency of each of the selected strings to organize the last letters into an encoding tree. Letters which occur more frequently encode less information; this results in more frequently-occurring letters having to be used more often in the text to carry the same amount of information as one occurrence of a letter which occurs less frequently. For example, the data structure Wayner showed for the choice between the letters *e*, *a*, and *o* as the next letter in the text, where *e* occurred most frequently at the end of the strings we're examining, and *o* occurred least frequently, looks like the following:



(Wayner 2002: 88)

The next letter chosen in this example, then, depends on the bits we need to encode. If a "0" needs to be encoded, we use an *e* as the next letter. If a "1" needs to be encoded, we use either an *a* or an *o*. The choice depends upon whether or not a "0" or a "1" follows the "1" we need to encode. *a* encodes two bits, "10", and *o* encodes "11". Thus, *a* and *o* provide more information in a shorter space and need to be used less frequently. This, then, generates a profile not unlike the real text from which it was derived.

One of these data structures needs to exist for each set of strings with the same first *n – 1* characters in the data set. Practically speaking, the larger *n* becomes, the better the results, but also the larger the amount of intermediate data structures which must be created for generation. For different sizes of *n*, Wayner gives examples of the algorithm's output using the text of a book chapter as the steganographic text generation seed. Each corresponding text is labeled as an *n-th order text*, where *n* is the string size mentioned above. For illustration, we show the first four lines of the first and fifth order output from Wayner 2002, pp 83 – 85:

> *First order text*:
> `islhne[hry saeeooisnre uo 'w nala al coehhs pebl e to agboean ce ed cshcenapch nt sibPah ea m n [tmsteoia lahid egnndl y et r yf arleo awe l eo rttnntnnhtohwiseoa a dri 6oc7teit2t...`
>
> *Fifth order text*:
> `The letter compression or video is only to generate a verbatim> followed by 12 whiter 'H' wouldn't design a perfective reconomic data. This to simple hardware. These worked with encodes of...`

While the first text is clearly gibberish, the second text is almost readable. The syntactic and semantic structures are still prone to serious error, but most of the words are actual lexical items and might pass a simple computer scan or even an uninterested human reader. More comprehensive linguistic details, of course, could cause detection problems. Additionally, it is possible that given the same generation "seed text", an attacker could figure out where the text was generated from. If that were to happen, given the right *n*, the attacker could generate the tables himself and figure out both that the text was generated by this scheme and what the original message was.

2.1.2 Linguistic methods – syntax and semantics

Computational power is increasingly able to analyze more and more complex linguistic structures. While the texts generated in previous examples may approximate some of the appearance of legitimate text, "simulating the statistical properties of text" must increasingly handle linguistic properties as well as lexical and orthographic distribution. Linguistic steganography specifically considers the linguistic properties of generated and modified text, and in many cases, uses linguistic structure as the space in which messages are hidden. This section describes existing methods of linguistic steganography as background for the linguistic issues to be discussed in section 3 and the proposed solution space in section 4.

Because syntactic analysis can be used to detect ungrammatical sequences of lexical items, one solution to detection by syntactic steganalysis is to ensure that structures are syntactically correct from the start. In fact, steganographic data can be hidden within the syntactic structure itself. As such, Wayner proposed that context-free grammars (CFGs) be used as a basis for generation of steganographic texts (Wayner 1992, 2002). Because the text is generated directly from the grammar, unless the grammar is syntactically flawed, the text is guaranteed to be syntactically correct. Furthermore, the tree structure created by a CFG is a natural device to use for encoding bits. Tree structures are used as optimizing data structures in many areas of computer science, from compilers to sorting algorithms. In the simplest scheme, at any point where there is a branch in the syntax tree, the left branch might mean '0' and the right branch might mean '1'. This section describes existing methods of using CFGs in order to generate syntactically correct steganographic text.

The easiest way to generate syntactically correct texts is to generate them from the syntax itself. This seems obvious, but finding a way to hide information within such text may not necessarily be. Wayner proposes use of a custom-made CFG in which any choice branch in a production indicates a bit or a sequence of bits. The **spammimic** example shown in section 1.1.3 uses such a grammar, and we present a very simple example here with discussion.

A grammar, like the Huffman tree shown in 2.1.1.2, provides a natural encoding structure for binary sequences. We give a very simple grammar below, and explain how data is encoded.

Wayner's method assumes that the grammar is in Greibach Normal Form (GNF) in order to prevent left recursion in parsing; that is, non-terminals come only at the end of productions. Given the following grammar:

```
Start := subject predicate
subject := propernoun || The noun
predicate := is adjective || can't be adjective
adjective := asleep || crusty
noun := moose || ball of cheese
propernoun := Oscar the Grouch || Trogdor the Burninator
```

by deciding that for every stage in the grammar where there is a choice, the first choice is a 0 and the second choice is a 1, we can encode bit sequences, and when parsing text created from this

grammar, retrieve them. Note that if this grammar is expressed as a tree, bits are encoded according to a pre-order traversal. Thus, when a decision is made at a non-terminal node, the bit from that decision is encoded, followed by all of the decisions from the subtree of its left child, and then its right child(ren).

In order to encode the bit string 0110, we do a pre-order traversal of the grammar. The Start symbol is processed first, from left to right; the first decision requires a 0 bit, so we choose a **propernoun** as a **subject** and encode a 0. We then need a 1, so when choosing a **propernoun**, we choose the second option, *Trogdor the Burninator*. Since all subtrees of **subject** have been processed, we now process **predicate** and its subtrees. Since the next bit we need in the sequence is a 1, we choose the second option for **predicate**, which is *can't be* **adjective.** Finally, we need a 0, so we choose the first option for **adjective**, which is *asleep*. So to encode the bit string 0110 with this grammar, we output the string "Trogdor the Burninator can't be asleep."

On the other hand, given the phrase "The ball of cheese is crusty", we get "1101". The small grammar we've presented is impractical, of course, since only $2^4 = 16$ sentence possibilities can be made from it, and there would be too much repetition to be a real text. As an illustration, the bit string 1011011111100000 would produce the following text:

```
The moose can't be crusty. Trogdor the Burninator can't be
crusty. The ball of cheese can't be asleep. Oscar the Grouch is
asleep.
```

While this text is syntactically correct, it is easily detectable as utter nonsense because the semantics are strange and there is no rhetorical structure. One rarely discusses crusty moose or sleeping balls of cheese; a ball of cheese cannot be an agent of the verb *sleep*, and so without careful construction of a CFG used to produce cover text, semantics go by the wayside. Furthermore, each sentence is an island unto itself; there is no structural relationship between the sentences, and the result is a string of sentences which have no relation to one another and are not about anything as a whole. These issues will be discussed in section 3.

Outside of the fact that actual phrases and indeed entire sentences could be repeated in carelessly constructed grammars, it is also true that unless huge grammars are constructed (often requiring

thousands of grammatical rules), syntactic structures may be detectably repeated (Chapman *et al* 2001 159). Creating grammars of this size can be a gargantuan task. Even if the sentences generated by these grammars are readable by humans, the "writing style" that comes out of such generation may be so strange that both humans and computers may be able to detect the anomalies in vocabulary usage, register, syntactic constructions, and so forth. None of the methods so far has any concept of the semantics of words within a syntactic structure. One step toward such an approach comes from Chapman and Davida, who created the NICETEXT system in order to do two things: first of all, they aimed to create "interesting sequences of parts-of-speech", and secondly, they aimed to categorize such parts-of-speech by "types" (which are essentially semantic categories) and to incorporate these types into the syntactic structures used for generation in order to make the text more believable to a human reader (Chapman and Davida, 1997). This was done by compiling large code dictionaries which categorized words by these types, and by creating style sources which acted as syntactic templates in which the words could be inserted (Chapman 1998: 6). Dictionaries are compatible with style sources if they contain all of the types needed by the style source templates. These code dictionaries, in addition to having words categorized by type, also contain the bit values which correspond to each word for the encoding of steganographic data.

Style sources begin with a *sentence model*, which is a syntactic (typed) template with additional information for formatting the sentence (punctuation and capitalization, for example). When generating text, a syntactic structure is chosen from a *sentence model table*, which is a set of syntactic sentence structures (which have the same semantic categories as the vocabulary words in the created dictionary), and one of the words which matches the current part of speech and semantic type with the desired bit value is inserted accordingly (1998:34).

Such texts have a distinct "style" which is derived by creating the sentence model table through large corpus analysis. Information is not encoded in the grammar itself, then, but by the choice of word which is inserted into the current sentence model (sentence models are chosen at random from the sentence model table). When the corpora used as sources for sentence model tables come from technical or obscure language sources (medical terms, legal proceedings, government

policies, archaic literature, etc.), there is a higher chance that the generated text will fool even human readers because the style differs greatly from more commonly read literature.

The following is a sample of text from the generated cover texts given in Chapman 1998. This comes from a style template generated from *The Complete Works of William Shakespeare* (65: Example 1):

> Not before the buttock, fair fathom, but by my will. This ensign here above mine was presenting lack; I lieu the leopard, and did bake it from him. Him reap upon, the taste boyish. Captain me, Margaret; pavilion me, sweet son. At thy service. Stories, away. I will run no chase wrinkle. …

Linguistic issues with this approach will be further explored in section 3, but note that not only does the text not make sense, but there are semantic inconsistencies, such as the fact that while *bake* is a transitive verb, one cannot *bake* something *from* someone, even in Shakespearean English. While syntacticians and semanticists may argue about whether or not this is a syntactic problem or a semantic problem, it clearly creates problems for the reader, and given a sufficiently sophisticated semantic analyzer, would be rejected.

Chapman states that the main concern for this system is not how a text looks statistically, but how it looks semantically (1998: 5). While this is not terribly precise, since actual semantics are only dealt with by tagging dictionary words with coarse semantic categories (which are integrated into the grammatical templates containing the same semantic categories (and generation beyond that is purely syntactic)), the NICETEXT system is a first attempt at going beyond syntax when doing natural language cover generation. However, since the coded dictionaries do not actually come from the same source as the sentence model templates, there is no sense of semantic coherence between sentences, and furthermore, the syntactic style used may not match the vocabulary used. A text which uses the structure of modern English with the vocabulary of Shakespeare would be decidedly odd. An informal "letter" produced with words used only in a formal register would be no less bizarre. Chapman *et al*, in response to some of these problems with both NICETEXT and simple CFG systems, came up with an improved system. The revised NICETEXT approach involves what the authors call "extensible contextual templates" combined with synonym replacement (2001). Known texts were again used to produce the sentence models; however, the sentence models themselves were extended into

"full-blown contextual templates". These no longer randomly choose sentence models at random from a sentence model table, but rather have a large contextual template of sentence models which are derived from the corpus text (161-163). Furthermore, what were "types" in the templates of the original NICETEXT are now synonym categories based upon the words used in the original text. The code dictionaries, then, are now synonym dictionaries with bit values assigned to synonym choices for each synonym type.

This provides a text which has a semantic coherence which previous methods of generating cover texts have lacked. Of course, sometimes synonym replacement causes problems; even when words are synonymous, they are not necessarily interchangeable within the same semantic structure. The classic example of such a set of synonyms is "eat" and "devour". One can *eat lunch*, and one can *devour lunch*. However, while one may have *eaten in the morning*, he cannot have *devoured in the morning*. The fact that *eat* may omit the direct object while *devour* requires one means that using them interchangeably will create semantic problems. The authors recognize such issues in the paper, and suggest culling the dictionary of such problem words; however, for a fully automated system with many large corpora and huge dictionaries, this may be an unscaleable task. Additionally, if the original text is available to the adversary, semantic analysis would quickly uncover the fact that the stego object was generated from a template derived from the original text.

3 Linguistic concerns with existing methods

While there are statistical issues with various kinds of text mimicking (depending upon where the adversarial threshold is set), there are also linguistic issues present. In some sense, one approach to linguistic steganalysis is to develop an analysis which is able to detect more complex linguistic structures than those which were used to produce the initial text. On the other hand, an approach to better steganography is to develop a steganographic technique which models more complex linguistic structures than current steganography is able to detect. While this may sound obvious, it is generally far from trivial to "go one better" than the current linguistic methods; it is, however, something to keep in mind when analyzing and developing protocols. Note that one level of linguistic correctness is often implied by another; a syntactically correct text will necessarily contain valid lexical items, while a semantically coherent text is generally also syntactically correct, and a rhetorically valid text has coherent semantics and correct grammar.

While not a perfect hierarchy for linguistic issues, it does show the increasing levels of complexity needed for increasingly deeper linguistic solutions. This section explores the progression from simple text-based methods to increasingly more linguistically complex solutions, and the linguistic issues with the methods discussed.

3.1 Generation of character and word sequences

When character-based generation schemes are used, a number of linguistic methods can be employed to detect potentially suspicious text. While it is true that one should not assume that all texts which are analyzed follow the rules of English (or any other given language), there are a number of methods which could be used in order to flag texts which might have hidden content. While section 2.1.1 discusses using statistical frequencies of letters and word lengths as parameters in generating cover texts as an improvement over simply using arbitrary characters, there are still linguistic features which can be examined in order to detect non-lexical items, even when the letter and word-length frequency are appropriate to the lexical model. The most obvious method for detecting non-lexical items would be to compare strings to a set of dictionaries (possibly in multiple languages) to see if the "words" in the text are in fact lexical items. Words can also be examined to see whether or not they follow common syllabification rules for widely-spoken languages; words in which too many vowels or consonants occur together or in which unlikely combinations of obstruents co-occur (violating sonority principles in codas or onsets, for example) might flag a text as suspicious. Even using inconsistent character sets (e.g. text containing characters with umlauts, accents and tildes) might be enough to cause a computer to flag the text for more careful review.

The fact that even the slowest of modern personal computers is capable of spell-checking lengthy texts in several languages, coupled with the other suggested analyses above, should be enough to suggest that trying to simulate lexical items in steganographic cover text has serious drawbacks. The logical answer to this dilemma is to use actual lexical items. While the use of random legitimate lexical items (with no syntactic structure) to encode information may immediately draw scrutiny from a human reader, if we assume an automated adversary which is efficient enough to do dictionary searches, but which either has too much accumulated text volume to do more complex text analysis or which is too slow to do linguistic processing of the text, this solution suffices. In fact, it should be noted that one common technique used by spammers to

defeat spam filters is to flood their mail messages with random dictionary words so that the filter assumes that the message has some other purpose than as a sales pitch. For a certain set of adversaries, then, the random lexical solution may suffice. That said, it does not make non-syntactic text generation a robust solution for steganography; because a syntactic analysis will reveal that there is no structure to these strings of words, it would be flagged as potentially suspicious text. Even without solid syntactic parsing capabilities, too many nouns, verbs, determiners, or adjectives in sequence (or any of these things in the wrong sequence) can conceivably be detected by automated tools without too much expense, making syntactic correctness a goal in successful automated text steganography.

3.2 Statistical sequences

In section 2.1.1.3, we examined Wayner's statistical solution which employs reverse Huffman encoding to encode texts through statistical analysis of an existing text as a basis for generation. As mentioned earlier, using higher-order mimic functions is extremely computationally expensive; however, even with the highest order functions, linguistic analysis can be employed to detect anomalies in the generated text. Even with the fifth order text shown in the example, both syntactic and lexical analysis would find errors. Furthermore, while the fact that the text is generated from a coherent piece of cover text might give greater semantic coherence than CFGs which are not written with semantics in mind, semantic analysis may well find that the ideas contained in a sentence are incoherent. For example, use of the analysis implied by Raskin and Nirenburg's ontological semantics framework could be used to generate text-meaning representations (TMRs) from the text (Raskin and Nirenburg 2003); if such structures fail to be generated or are found to have problems (such as arguments to non-transitive verbs), those texts could be flagged as potential stego-objects. Finally, rhetorical analysis, such as that discussed in Marcu (2000), is a possible method of determining that the text produced, even where it appears to form coherent sentences, is unconnected on anything but the sentence level.

3.3 Context-free grammars

As mentioned at the end of section 2.1, syntactic correctness is a logical goal for successful text steganography. The use of CFGs to generate a cover text guarantees syntactic correctness insofar as the CFG itself is correct (it should be noted that humans don't actually always produce syntactically correct text, so the necessity for complete syntactic correctness should not be taken

as an absolute requirement). The linguistic issues raised by this set of steganographic methods is less clear-cut than with random character and word generation. CFGs can be written such that they are consistent in terms of topic, and in fact, Wayner suggests a method which uses different grammars at different stages of the text so that the discourse and rhetorical structures even seem to be relatively coherent (the example he uses is a grammar which generates different voiceover texts for different innings of a baseball game (Wayner 2002); there have also been attempts to make such variable grammars work by hiding as dialogue on an IRC channel (Mystic, 2003)). The chances of such specialized grammars being used on a large-scale basis, however, are slim; because of the amount of effort that must be invested in generating a large grammar which conforms to the requirements (such as syntactic correctness and being in Greibach Normal Form, as well as having big enough productions to create reasonably undetectable texts), the fact that so little information is carried in so much text makes it unscaleable as a solution. To decrease the effort required and make the solution scalable, one must accept smaller grammars with less text associated with each production. This will necessarily create texts which are neither semantically nor rhetorically coherent. Furthermore, it resurrects the problem of repetition within and between texts, making recognition of a single stego-text possible, as well as increasing the effectiveness of using preexisting known stego-objects as tools for recognizing other stego-objects from their grammar and lexicon.

3.4 Semantic replacement

Even the solution proposed by Chapman *et al* as an improvement on NICETEXT has linguistic issues. As mentioned in section 2.1.2, the possibility of using unviable synonym replacement (*e.g.* verbs with different transitivity values, or non-compositional phrasal verbs (see Televnaja *et al* 2004) which may be split into synonyms for the lexical items which make up the phrasal verb but which have nothing to do with the actual meaning of the phrasal verb itself) can be a problem. Furthermore, and this is potentially more important, using ontological semantics to create text-meaning representations of such text would immediately show that the texts have the same semantic structure, since different lexical items can still have the same underlying semantic structure (Raskin and Nirenburg). While Chapman *et al*'s attention to semantics provides an advantage over previous methods both in terms of computational detectability and human readability, the vulnerability to a known cover attack through semantic analysis gives motivation for a linguistic approach which attempts to remain semantically (and rhetorically) cohesive while

using cover generation (without an extant text as a basis) to avoid known cover attacks. This is the motivation for the class of approaches suggested in section 4 as next steps in effective linguistic steganographic cover generation.

4 Future directions in constructing linguistically and statistically robust cover texts

As discussed in sections 1 and 2, current methods used in text steganography are particularly vulnerable to linguistic steganalysis. Like the use of a CFG to mimic the syntactic statistical profile of normal text, making a steganographic text conform to semantic and rhetorical standards forces the stego text into the statistical profile of semantically and rhetorically cohesive texts which do not hold steganographic content. Thus, linguistic cover generation aims to defend against both statistical and linguistic steganalysis. This section discusses current and future goals for linguistic steganography and the standards by which we should anticipate having to measure them.

We must assume that in the relatively near future, semantic attacks on linguistic steganography will become an issue; Atallah, et al, propose methods of semantically watermarking text (embedding undetectable bits of information using the syntax and semantics of the text), and the semantic improvements to NICETEXT indicate that this is one direction that linguistic steganography is headed. As stated earlier, new steganographic methods usually induce new analysis mechanisms which are able to detect them. Such an arms race must lead researchers to expect (and even devise) attacks on emerging linguistic steganography systems which will require increasingly sophisticated linguistic solutions. Also, with increasing attention being paid to the human reader as a potential detection mechanism, semantic and rhetorical correctness within steganographic text is an important consideration for future work. At the very least, linguistic solutions should be able to endure lexical, syntactic, and semantic scrutiny. To pass the human reader, however, this is not enough. Being able to generate semantically coherent sentences is a necessary element of a good linguistic solution; however, it is not sufficient, since a string of semantically coherent sentences does not necessarily make for a semantically coherent text. Rhetorical and discourse structure must also be considered if the text is in any way required to “make sense”. Since rhetorical analyses based upon Rhetorical Structure Theory (Mann and

Thompson 1988) are being developed for other uses, there is no reason these could not be applied to linguistic steganalysis (Marcu 2000).

Because future solutions should pay attention to both coherent semantics and rhetorical structure, two suggestions for future steganographic work would be to include ideas from both ontological semantics (which provides a useful tree-based inheritance structure for semantic concepts) and RST in the actual generation of the text. This comes from the same intuition which inspired Wayner's CFG-based syntactic mimicking; that is, when there is the possibility of detecting statistical inconsistencies in a cover text because of a structural factor, that structural factor should then be used as a means for generating the solution, guaranteeing that the statistical profile of the generated cover approximates that of a normal text in terms of that particular structural factor.

Future work in semantic and rhetorical mimicking will need to include structural analyses of semantics and rhetorical structure; Marcu's RST-based generation and analysis of rhetorical structure could, at the very least, be used to ensure that generated texts would look rhetorically coherent, even if information is not encoded at the rhetorical level. Rhetorical relations between parts of text such as *justification, evidence, elaboration, background, motivation*, etc. are recursively associated with sections of text such that the elaborate and compositional rhetorical structure of a whole text unit can be expressed. With some concept of what makes a valid rhetorical structure, such trees could be built as scaffolds for generated text in order to ensure rhetorical correctness.

Semantics (and the interface of semantics with both syntax and rhetoric) must also be dealt with; Raskin and Nirenburg's system of ontological semantics has already been used to structure a watermarking system which takes advantage of the text structure of semantic representations in order to encode and restructure text in which bits are hidden (Atallah *et al* 2001, Atallah *et al* 2002). Further exploitation of the tree-based structure of the ontology could be used as a source not only for encoding bits through concepts and the relations between them, but also for ensuring that texts stayed "on topic" in the generation process by paying attention to concepts and the classes of concepts and lexical items that they are associated with, derived from, and composed

of. Future work will explore these options in the course of designing semantically coherent cover generation mechanisms. In combination with RST-based generation techniques, it is hoped that a system can be devised which can produce semantically, syntactically, lexically, and rhetorically correct cover texts which will be an advance in increasing human readability of cover texts without suspicion, as well as evading statistical and linguistic attacks against such methods. Ontological semantics and rhetorical parsing and generation will be discussed in a future paper on the theoretical linguistic aspects of believable, coherent cover generation.

References

Anderson, Ross. Foreward to *Information Hiding: Techniques for Steganography and Digital Watermarking*, Stefan Katzenbeisser and Fabien A.P. Petitcolas (eds.). Boston,:Artech House, 2000.

Anderson, Ross and Fabien Petitcolas. "On the Limits of Steganography." *IEEE Journal of Selected Areas in Communications*, 16:4. 474-481. 1998.

Atallah, Mikhail J., Victor Raskin, Christian P. Hempelmann, Mercan Karahan, Radu Sion, Umut Topkara, and Katrina E. Triezenberg. "Natural Language Watermarking and Tamperproofing." *Proceedings of the 5th International Information Hiding Workshop, Lecture Notes in Computer Science.* 196-212. 2002**.**

Atallah, Mikhail, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. "Natural Language Watermarking: Design, Analysis, and a Proof-of-Concept Implementation." *Pre-Proceedings of the 4th Information Hiding Workshop*, 2001. 193-208**.**

Aura, Tuomas. "Practical Invisibility in Digital Communication." *Information hiding: First International Workshop*, *Lecture Notes In Computer Science* 1174. 266-278. 1996.

Brassil, J, S. Low, N. Maxemchuk, and L. O'Gorman. "Document marking and identification using both line and word shifting." Technical report, AT&T Bell Laboratories, 1994.

Brubeck, Matt, Joshua Haberman, and Dominic Mazzoni. "Audacity audio editor". http://audacity.sourceforge.net, 2004.

Chapman, Mark T. *Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text*. Milwaukee: University of Wisconsin-Milwaukee. Master's Thesis. 1998.

Chapman, Mark and George Davida. "Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text." *Proceedings of Information Security, First International Conference, Lecture Notes in Computer Sciences 1334*. Berlin: Springer, 333-345. 1997.

Chapman, Mark, George I. Davida, and Marc Rennhard. "A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography." *Proceedings of the Information Security Conference (ISC '01)*, Malaga, Spain, October 2001. 156-165

Doyle, Chris. "The Mini-Mizer." http://www.reasonablyclever.com/mini/.

Gaudin, Sharon. "Record Broken: 82% of U.S. Email is Spam". *IT Management: Security*, May 5, 2004. http://itmanagement.earthweb.com/secu/article.php/3349921

Johnson, Neil F. "Steganalysis" In *Information Hiding: Techniques for Steganography and Digital Watermarking*. Boston,:Artech House. 79-93. 2000.

Johnson, Neil F. and Sushil Jajodia. "Exploring Steganography: Seeing the Unseen." *IEEE Computer*, 32:2. 26-34. 1998.

Johnson, Neil F. and Stefan Katzenbeisser. "A Survey of Steganographic Techniques." In *Information Hiding: Techniques for Steganography and Digital Watermarking*. Boston,:Artech House. 43-78. 2000.

Katzenbeisser, Stefan. "Principles of Steganography." In *Information Hiding: Techniques for Steganography and Digital Watermarking*. Boston,:Artech House. 17-41. 2000.

Katzenbeisser, Stefan and Fabien A.P. Petitcolas (eds). *Information Hiding: Techniques for Steganography and Digital Watermarking*. Boston,:Artech House, 2000.

Mann and Thompson. "Rhetorical Structure Theory: Toward a Functional Theory of Text Organization." *Text*, 8:3. 243-281. 1988.

Marcu, Daniel. *The Theory and Practice of Discourse Summarization and Parsing*. Cambridge: MIT Press. 2000.

Mystic. *ircMimic*. From "Mimicry: An Introduction", DefCon 11, Las Vegas, 2003. http://www.inventati.info/pub/defcon11/Mimic-Mimicry/

Petitcolas, Fabien A.P., Ross J. Anderson, and Markus G. Kuhn. "Information Hiding – A Survey." *Proceedings of the IEEE*, 87:7. 1062-1078. 1999.

Petitcolas, Fabien A.P. "Introduction to information hiding". In *Information Hiding: Techniques for Steganography and Digital Watermarking*, Stefan Katzenbeisser and Fabien A.P. Petitcolas (eds.). Boston,:Artech House. 1-14. 2000.

Provos, Niels. "Defending Against Statistical Steganalysis." CITI Technical Report 01-4, University of Michigan, February 2001.

Raskin, Victor and Sergei Nirenburg. *Ontological Semantics*. Cambridge: MIT Press, 2003.

"Spam Mimic." http://www.spammimic.com. 2000.

Steganos GmbH. "Steganos Security Suite 6". http://www.steganos.com. Frankfurt am Main: 2004.

Televnaja, Julija, Krista Bennett, Christian Hempelmann and Katrina E. Triezenberg. ”Semantic Representation of English Phrasal Verbs” *Intelligent Information Systems 2004: New Trends in Intelligent Information Processing and Web Mining (IIPWM 2004) May, 2004, Zakopane, Poland* (to appear).

Wayner, Peter. “Mimic Functions.” *Cryptologia* XVI:3. 192-213. 1992.

Wayner, Peter. *Disappearing Cryptography: Information Hiding: Steganography & Watermarking* (second edition). San Francisco: Morgan Kaufmann. 2002.

Westfeld, A. “High Capacity Despite Better Steganalysis: F5 – a Steganographic Algorithm.” *Proceedings of the 4th Information Hiding Workshop, Lecture Notes in Computer Science 2137*, 301-314. 2001.

Westfeld, Andreas. “F5 Steganography Software.” http://wwwrn.inf.tu-dresden.de/~westfeld/f5.html. 2003.

Zöllner, J., H. Federrath, H. Klimant, A. Pfitzmann, R. Piotraschke, A. Westfeld, G. Wicke, and G. Wolf. 1998. “Modeling the Security of Steganographic Systems.” *Proceedings of the 2nd Workshop on Information Hiding, Lecture Notes In Computer Science.* Springer-Verlag, 344-354.