New Algorithm for Text in Text Steganography

Dr. Nidhal K. El Abbadi
Kufa University, IRAQ
nidhalka@it.kuiraq.com

Abstract

Steganography is a technique to hide secret information in some other data (we call it a cover) without leaving any apparent evidence of data alteration. All of the traditional steganographic techniques have limited information- hiding capacity. They can hide only 10% (or less) of the data amounts of the cover. While much of the recent research in steganography has been on hiding data in images, many of the solutions that 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.
This paper present a new algorithm to hide a large amount of text in cover text without effecting the cover, by using many types of pointers (which are characters can interpreter as invisible character, or as apart of cover. Pointers used as single pointer or set of pointer to represent new single pointer. In this algorithm we can hide more than 40% of the data amounts of the cover.

Key Words: steganography, text in text, cover, embedded.

1. Introduction

Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out of the usual. Steganography is often confused with cryptology because the two are similar in the way that they both are used to protect important information. The difference between the two is that Steganography involves hiding information so it appears that no information is hidden at all. If a person or persons views the object that the information is hidden inside of he or she will have no idea that there is any hidden information, therefore the person will not attempt to decrypt the information. Steganography comes from the Greek words **Steganós** (Covered), and **Graptos** (Writing). Steganography in the modern day sense of the word usually refers to information or a file that has been concealed inside a digital Picture, Video, Text or Audio file. What Steganography essentially does is exploit human perception, human senses are not trained to look for that has information hidden inside of them, although there are programs available that can do what is called Steganalysis (Detecting use of Steganography.).The most common use of Steganography is to hide a file inside another file.

Within The past several years, there has been an exponential increase in the research community and industry's towards Information hiding techniques as opposed to the traditional cryptography area. Figure (1) expressively depicts this rapid increase in topic publications. There have been three international workshops related solely to Information hiding since 1996[1].
The remainder of the paper is structured as follows. First, Section 2 reviews uses of steganography. In Section 3, the principles of hiding text in text. In Section 4, we give reviews for the related work. While section 5 concentrated on proposal algorithm, and finally section 6 conclusions.

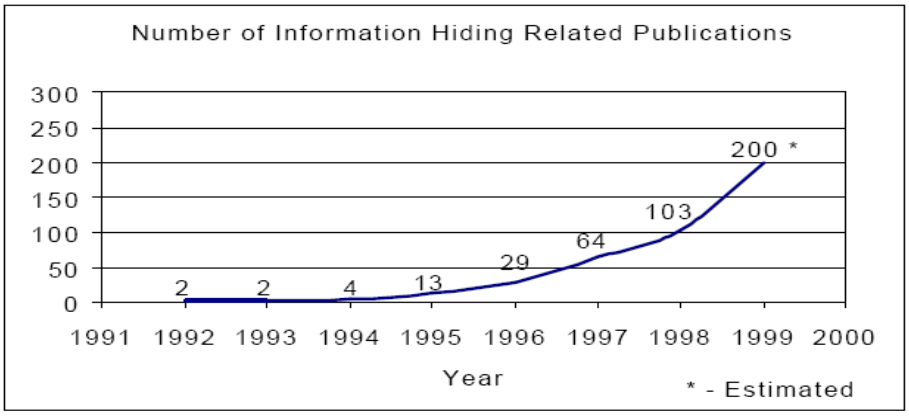

Figure (1): show increasing in Information Hiding Publications

2. What is Steganography Used for?

Like many security tools, steganography can be used for a variety of reasons, some good, some not so good. Legitimate purposes can include things like watermarking images for reasons such as copyright protection. Digital watermarks (also known as fingerprinting, significant especially in copyrighting material) are similar to steganography in that they are overlaid in files, which appear to be part of the original file and are thus not easily detectable by the average person. Steganography can also be used as a way to make a substitute for a one-way hash value (where you take a variable length input and create a static length output string to verify that no changes have been made to the original variable length input). Further, steganography can be used to tag notes to online images (like post-it notes attached to paper files). Finally, steganography can be used to maintain the confidentiality of valuable information, to protect the data from possible sabotage, theft, or unauthorized viewing.

Unfortunately, steganography can also be used for illegitimate reasons. For instance, if someone was trying to steal data, they could conceal it in another file or files and send it out in an innocent looking email or file transfer. Furthermore, a person with a hobby of saving pornography, or worse, to their hard drive, may choose to hide the evidence through the use of steganography. And, as was pointed out in the concern for terroristic purposes, it can be used as a means of covert communication. Of course, this can be both a legitimate and an illegitimate application.

3. Hiding a message inside a text

Since everyone can read, encoding text in neutral sentences is doubtfully effective. But taking the first letter of each word of the previous sentence, you will see that it is possible and not very difficult. Hiding information in plain text can be done in many different ways. The first-letter algorithm used here is not very secure, as knowledge of the system that is used, automatically gives you the secret. This is a disadvantage that many techniques of hiding secrets inside plain text have in common.

Many techniques involve the modification of the layout of a text, rules like using every nth character or the altering of the amount of whitespace after lines or between words. The last technique was successfully used in practice and even after a text has been printed and copied on paper for ten times, the secret message could still be retrieved.

Another possible way of storing a secret inside a text is using a publicly available cover source, a book or a newspaper, and using a code which consists for example of a combination of a page number, a line number and a character number. This way, no information stored inside the cover source will lead to the hidden message. Discovering it relies solely on gaining knowledge of the secret key.

Text is one of the oldest media used in steganography; well before the electronic age, letters, books, and telegrams hid secret messages within their texts. I'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

3.1 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.

3.2 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.

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.

3.3 Linguistic methods

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, these two has problems. A string of words with no semantic structure and no discernable semantic relationship can be detected by both humans and computers alike.
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’.

4. Related Work

Traditional linguistic steganography has used limited syntactically-correct text generation (sometimes with the addition of so-called "style templates") and semantically-equivalent word substitutions within an existing plaintext as a medium in which to hide messages. Wayner [7] introduced the notion of using precomputed context-free grammars as a method of generating steganographic text without sacrificing syntactic and semantic correctness. Note that semantic correctness only guaranteed if the manually constructed grammar enforces the production of semantically cohesive text.

Chapman And Davida [5] improved on the simple generation of correct text by Syntactically tagging large corpora of homogeneous data in order to generate grammatical "style templates"; these templates were used to generate text which not only had syntactic and lexical variation, but whose consistent register and "style" could potentially pass a casual reading by a human observer.

Chapman et al [4], later developed a technique in which semantically equivalent substitutions were made known plaintexts in order to encode messages. Semantically-Driven information hiding is a recent innovation, as described for watermarking schemes in Atallah et al [6].

Wayner [7] detailed text-based that are strictly statistical in nature. However, in general, linguistic approaches to steganography have been relatively limited. Damage To language is relatively easy for a human to detect. It does not take much modification of a text for a native speaker to judge it to be ungrammatical; furthermore, even syntactically and grammatically texts can violate semantic constraints.

5. Proposal Algorithm

This algorithm aim to the following:

1. Hide text (embed) in another text (cover) without leaving any sign on the cover to point to exist of embedded text, meaning the stego-cover (cover after hiding embed in it) unchanged and look the same as the original cover (cover) to third party when displaying on the screen .
2. Increasing embedded size that can hide in cover.
3. Increase complexity to analyze and break stego-cover.

General notes :

- Choose any text randomly to use it as cover with one restriction (size of cover should be two or more times size of embedded.
- The useful number of bits in each embedded character is less than six bits (with just five bits we can represent (32) character, (26) of them are English

alphabetic characters and remaining are the important character like (space, comma, bracket … etc)).

- To increase size of embedding and complexity we arrange embedded characters in table with numbers range (0 .. 31) (to reduce number of bits to represent its), this table arrange depend on user can make any arrange (to increase complexity). It's type of encryption.
- This table consist of three column first one for characters, second one for sequence numbers (0 .. 31), and third one is for the corresponding (ASCII) to the characters.

  As **example**:

  Table (1):Characters with suggestion numbers proposed for embedded Purpose used by this paper

characters	Sequence No.	ASCII No.
a	0	97
m	1	109
f	2	102
j	3	106
y	4	121
d	5	100
k	6	107
s	7	115
x	8	120
q	9	113
e	10	101

- To hide embedded in cover we have to pass two times on cover characters.
- First pass using pointer point to bits (use one bit from each cover character (except last character in each word) to hide one bit of embedded characters using specific pointer).

  In this case size of embedded hide when pass all cover characters and hide one bit in each, will equal to (16 % of cover size) { 6 bits of embedded hide in 8 bytes of cover }.
- The second pass different from the first pass, in this pass we seek for space between words from beginning to end of document file. In each space hide one of embedded characters, using special technique explain later depending on another pointer.

  In this stage size of embedded that can hiding in cover depended on number of spaces in cover (number of spaces is equal to number of words in cover).

By analyzing many different English text documents, I found the average of number of characters in words is equal to (6 characters (including space)), that mean size of embedded can hiding in cover by this technique is equal to (16 % of cover size).

- In all of above we talk about hiding embedded characters (except space) in cover. Embedded space can hiding in cover using different technique (it is important to hide embedded spaces in cover to recognize words). In this stage size of spaces that can hide in cover equal to about (16 % of cover size).
- In each of above passes we pass sequentially from first character to the last one in document file.
- From these three stages the total embed size that can hide in cover is about (48 % of cover size).

5.1 What is pointer?

Pointers that I suppose to use in this algorithm is characters or symbols uses to point to cover character or bit inside file code without any effect on displaying the cover contents.

Changing in cover header formatting will help word processor to display cover on screen without displaying pointers (word processor will neglect some of pointers, process other as invisible characters, and process other as part of cover).

Word processor process covers as follow:

- *Neglect pointer when displaying cover*, this pointer use to point to bit in character (let call it (P_1))… as example if we want to point to bits in characters of word (Baghdad) as follow :

B	**P_1**	a	**P_1**	g	**P_1**	h	**P_1**	d	**P_1**	a	**P_1**	d

At this case the word processor will neglect displaying pointer (P_1) and displaying only the characters of the word (Baghdad).

- *Treat pointer as part of cover* (it process it as cover character) (let call it (P_2))… as example if we use it with the word (Baghdad) as follow:

B	a	g	**P_2**	d	a	d

In above pointer (P_2) used instead of character (h) in word (Baghdad), and that mean there is an embedded character hided in place of pointer, in this case word

processor will replace character (h) instead of pointer (P_2) and display the word (Baghdad) on screen.

- *Neglect pointer when displaying cover like the first one*, but this pointer point to one character, and it call (P_3)… as example if we use it with the word (Baghdad) as follow

B	a	g	**P_3**	h	d	a	d

Here pointer (P_3) point to the character (g), and word processor will neglect it and display the word (Baghdad) on screen.

5.2 Pointers type

1. First type call (P_1), which is neglect from word processor when displaying cover on screen. This pointer use between cover characters inside document file, and point to one bit in each character. This pointer consists of one byte value.

2. Second type call (P_2), and it use either by it self or grouping with the first pointer (P_1) to make new pointers. This pointer consists of five different byte values, uses by it self or by grouping with pointer (P_1). These five pointers use to represent (30) different characters (can be 26 alphabetic characters and four for important symbol or usage).

3. Third type call (P_3), and it use to point to cover character which is similar to embedded character. This pointer represented by grouping more than one of first pointer (P_1).

5.3 Implementation

To implement this algorithm, follow the following steps:

1. Open the two documents files (embed and cover).

2. Files document consist of three parts (header, plaintext, and footer), pointers work on plaintext, and there are some changes in formatting of header and footer.

3. Make table to convert between characters and corresponding (ASCII) from one side and sequence number (0 .. 31) from other side, as in table(1).

4. Read one byte of embedded (b1), and converted it to corresponding number according to table building in step (3).

5. Read five characters (bytes value) from cover, and convert it to corresponding number according to table build in step (3).

6. Compare embedded byte (b1) with the five cover characters (bytes value) if (b1) is identical to any one of five bytes, then insert pointer (P_3) after the identical character, and go to step (4).

   Note: the remaining characters from five characters when insertion pointer should take in account in step (5) as example: let look how to account new five characters when insert pointer after third character as follow

C1	C2	C3	**P_3**	C4	C5

   In this case the remaining two characters after pointer (C4, and C5) will account with new five characters in step (5) to be the first two characters, and read just three new characters . Else go to step (6)

7. Partition the embedded character (byte value) to five single bits (embedded value range (0 .. 30))according to table (1).

8. Specify which bit from the (8 bits) of cover byte will use to hide embedded bit.

9. Compare each bit of embedded with the specified bit of five cover bytes (each one with the corresponding one of cover, first embedded bit with the bit of first cover byte, second embedded bit with the bit of second cover byte, and so on). If the two compared bits were equal then insert pointer (P_1) after corresponding cover byte. This step repeat five times until all five bits compared.

10. Check end of files, for both embedded and cover. If no one reach end of file then go to step (4), else close the ended file and go to step (11).

11. If embedded file was ended then go to step (18), else open the cover file again and start from beginning of plain text of cover file (from first cover byte).

12. Build table to represent characters according to orientation of pointer (P_2).

13. Continue reading one character from embedded (from the point or character reach it).

14. Seek for spaces in cover (space between words) sequentially, this done by reading cover character (byte value) one character each time until find space.

15. Insert pointer (P_2) instead of space value. This pointer represent embedded byte value reading in step (12).

16. Check end of files, for both embedded and cover. If one of them ended then go to step (17), else go to step (13).

17. If cover file ended, then change cover with one with larger size, and go to step (1). And if the embedded file end then go to step (18).

18. Close both files.

19. End.

6. Conclusions

6.1 Stego generated from scratch.

6.2 Steganography involves hiding data in an overt message and doing it in such a way that it is difficult for an adversary to detect and difficult for an adversary to remove. Based on this goal, three core principles can be used to measure the effectiveness of a given steganography technique: amount Of data, difficulty of detection, and difficulty of removal. Amount of data suggests that the more data you can hide, the better the technique, and in this algorithm we can hide a large amount of data.

6.3 Difficulty of detection relates how easy it is for somebody to detect that a message has been hidden. There is usually a direct relationship between how much data can be hidden and how easy it is for someone to detect it. As you increase the amount of information that is hidden in a file, we increase the chance that someone will be able to detect that there is information hidden in the file, but by this algorithm there is no any different between origin cover and stego-cover.

6.4 Difficulty Of removal is the principle that someone intercepting your file should not be able to remove easily, and this algorithm give more complexity to remove embedded text.

For many potential applications, however, it is more convenient to grow the size of the data cube dynamically to suit the data. For example, astronomers who are analyzing stars might form a data cube for their star database. They expect to discover more stars in the future. Clearly it would not be efficient to create a data cube that initially contains cells for all possible locations of star systems in the Universe, particularly since the vast majority of the resulting cells would always be empty. Rather, it is more practical to create the data cube initially only for locations of existing star systems; as additional systems are discovered, new cells can be added to the data cube. New star systems, however, can be found in any direction relative to existing systems, therefore the data cube must be able to grow in any direction relative to its existing cells.

Figure (2): Embedded text

Decision Support Systems

As the name implies, decision support systems are designed to empower the user with the ability to make effective decisions regarding both the current and future state of an organization. To do so, the DSS(decision support system) must not only encapsulate static information, but it must also allow for the extraction of patterns and trends that would not be immediately obvious. Users must be able to visualize the relationships between such things as customers, vendors, products, inventory, geography, and sales. Moreover, they must understand these relationships in a chronological context, since it is the time element that ultimately gives meaning to the observations that are formed. Discussions of DSS applications often implicitly assume that such systems are part and parcel of a single homogeneous technology. In fact this is typically not the case. Rather, knowledge-based systems come in a number of distinct forms, each extending the functionality of the core Database Management System (DBMS). Below, is a description of three primary DSS models, including OLAP.

Information Processing. It is concerned with fundamental querying and reporting functions. IT professionals typically design queries when supporting database management systems. Some form of visualization module may also be utilized to streamline the user interface. Analysis at this point is likely to be quite simple, consisting of sorting and basic aggregation, and lacks the sophistication to uncover anything but the most obvious features of the stored data. OLAP, Online Analytical Processing extends the basic reporting capabilities of Information processing systems by allowing a robust multidimensional analysis of the archived data from a variety of perspectives and hierarchies. Operations such as *drill-down*, *roll-up* and *pivot* provide insights into corporate growth, spending, and sales patterns that would simply not be possible other -wise. Additional OLAP functionality may include operations for ranking, moving averages, growth rates, statistical analysis, and “what if" scenarios. Note that the terms “DSS" and “OLAP" are sometimes used interchangeably.

Data Mining. This third class represents the “natural evolution" of knowledge-based data management systems. In this case, the goal is to automate the discovery process so that trends and patterns can be retrieved with minimal user input. The patterns are not necessarily those that are embedded directly in the aggregated fields of the data warehouse. Rather, they may consist of subtle regularities that cross hierarchical and /or dimension boundaries and, as such, would be less likely to be discovered by conventional OLAP techniques.

Figure (3): Text Cover

Decision Support Systems

As the name implies, decision support systems are designed to empower the user with the ability to make effective decisions regarding both the current and future state of an organization. To do so, the DSS(decision support system) must not only encapsulate static information, but it must also allow for the extraction of patterns and trends that would not be immediately obvious. Users must be able to visualize the relationships between such things as customers, vendors, products, inventory, geography, and sales. Moreover, they must understand these relationships in a chronological context, since it is the time element that ultimately gives meaning to the observations that are formed. Discussions of DSS applications often implicitly assume that such systems are part and parcel of a single homogeneous technology. In fact this is typically not the case. Rather, knowledge-based systems come in a number of distinct forms, each extending the functionality of the core Database Management System (DBMS). Below, is a description of three primary DSS models, including OLAP.

Information Processing. It is concerned with fundamental querying and reporting functions. IT professionals typically design queries when supporting database management systems. Some form of visualization module may also be utilized to streamline the user interface. Analysis at this point is likely to be quite simple, consisting of sorting and basic aggregation, and lacks the sophistication to uncover anything but the most obvious features of the stored data. OLAP, Online Analytical Processing extends the basic reporting capabilities of Information processing systems by allowing a robust multidimensional analysis of the archived data from a variety of perspectives and hierarchies. Operations such as *drill-down*, *roll-up* and *pivot* provide insights into corporate growth, spending, and sales patterns that would simply not be possible other -wise. Additional OLAP functionality may include operations for ranking, moving averages, growth rates, statistical analysis, and “what if" scenarios. Note that the terms “DSS" and “OLAP" are sometimes used interchangeably.

Data Mining. This third class represents the “natural evolution" of knowledge-based data management systems. In this case, the goal is to automate the discovery process so that trends and patterns can be retrieved with minimal user input. The patterns are not necessarily those that are embedded directly in the aggregated fields of the data warehouse. Rather, they may consist of subtle regularities that cross hierarchical and /or dimension boundaries and, as such, would be less likely to be discovered by conventional OLAP techniques.

Figure (4): Stego-cover

Refrences

1. **Christian Grothoff, Org Krista, Org Ryan, Edu Mikhail Atallah** "*Translation-Based Steganography*", 2007

2. **Eric Cole, *Ronald D. Krutz*** *"HidinginPlainSight: Steganography and the Art of Covert Communication*", Published by Wiley Publishing, Inc., Indianapolis, Indiana , 2003

3. **Jordon T. Cochran, Captain** "*STEGANOGRAPHIC COMPUTER WARFARE*", THESIS introduced to **AIR FORCE INSTITUTE OF TECHNOLOGY, Wright- Patterson Air Force Base,Ohio**

4. **Mark Chapman, George Davida, and Marc Rennhard**. "*A practical and effective approach to large- scale automated Linguistic steganography*". In Proceedings of the Information Security Conference (ISC ' 01), pages 156– 165, Malaga,Spain, 2001.

5. **Mark Chapman and George Davida**. "*Hiding the hidden:A software system for concealing ciphertext in innocuous text*". In Information and Communications Security First International Conference, volume Notes In Computer Science 1334, Beijing, China, 11– 14 1997.

6. **Mikhail J. Atallah, Viktor Raskin, Christian Hempelmann, Mercan Karahan, Radu Sion, and Katrina E. Triezenberg**. "*Natural language watermarking and tamperproofing"*. In Proceedings of the 5th International Information Hiding Workshop 2002, 2002

7. **Peter Wayner**. "*Disappearing Cryptography: Information Hiding: Steganography and Watermarking*", Morgan Kaufmann, 2nd edition edition, 2002.