EUROPEAN ACADEMIC RESEARCH
Vol. II, Issue 12/ March 2015

Impact Factor: 3.1 (UIF)
DRJI Value: 5.9 (B+)

ISSN 2286-4822
www.euacademic.org

Improve Capacity in Text in Text Steganography

SAMER THAABAN ABAAS
Education College, University of Kufa, Iraq
ZAID NIDHAL KHUDHAIR
Imam Kadhim College, Iraq
SAMI KADHIM KAREEM
Education College, University of Kufa, Iraq
ZAINAB NIDHAL KHUDHAIR
Education College, University of Kufa, Iraq

Abstract:

With the rapid development of Internet, safe covert communications in the network environment become an essential research direction. Steganography is a significant means that secret information is embedded into cover data imperceptibly for transmission, so that information cannot be easily aware by others. Text Steganography is low in redundancy and related to natural language rules these lead to limit manipulation of text, so they are both great challenges to conceal message in text properly and to detect such concealment.

This paper presents a novel algorithm to hide a large amount of text in cover text without affecting the cover format, by using many types of pointers (which are characters can interpreter as invisible character, or as a part of cover. Pointers used as single pointer or set of pointer to represent new single pointer. The suggested algorithm can hide more than 40% of the cover size, which represent more than four times of the best known method used to hide text in text.

Key words: steganography, text in text, hiding data, security.

1. Introduction

In today's world, keeping the digital information safe from being misused is one of the most important criteria. This issue gave rise to a new branch in computer science, named Information Security. Although new methods are introduced every day to keep the data secure, but computer hackers and unauthorized persons are always trying to break those methods or protocols to fetch the sensitive beneficial information from those data. For this reason, computer scientist and cryptographers are trying very hard to come up with permanent solutions to this problem (Somdip Dey, 2012).

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. Steganography in the modern day sense of the word usually refers to information or a file that has been concealed inside a digital Image, Video, Text or Audio file.

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.

2. 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.

Peter Wayner 2002, 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.

Monika 2013, introduced three approaches of text steganography. The first approach uses the theme of the missing letter puzzle where each character of message is hidden by missing one or more letters in a word of cover. The average Jaro score was found to be 0.95 indicating closer similarity between cover and stego file. The second approach hides a message in a wordlist where ASCII value of embedded

character determines length and starting letter of a word. The third approach conceals a message, without degrading cover, by using start and end letter of words of the cover. For enhancing the security of secret message, the message is scrambled using one-time pad scheme before being concealed and cipher text is then concealed in cover.

Ammar and Khaled 2012, proposed a new steganography algorithm for Arabic text. The algorithm employs some letters that can be joined with other letters. These letters are the extension letter, Kashida and Zero width character. The extension letter, Kashida does not have any change in the word meaning if joined to other letters. Also, the Zero width character (Ctrl+ Shift +1) does not change the meaning. The new proposed algorithm, Zero Width and Kashidha Letters (ZKS), mitigate the possibility to be discovered by steganoanalysis through using parallel connection and permutation function.

Wesam and et al. 2013, suggested text steganography method which takes into account the Font Types. This new method depends on the Similarity of English Font Types. It works by replace font by more similar fonts. The secret message was encoded and embedded as similar fonts in capital Letters of cover document. Proposed text steganography method can works in different cover documents of different font types. The size of cover and stego document was increased about 0.766% from original size.

3. Proposed 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 present 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 the 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 order depend on user to make any arrange (which increase complexity). It's type of scrambling.
- 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 suggested numbers proposed for embedding process 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 scan the cover characters two times.
- First scan used pointer which point to bits (each character will be converted to the corresponding ASCII numbers represented as binary number). From each cover character we used one bit (except last character in each word) to hide one bit of embedded characters by using specific pointer). In this case size of embedded hided when scan all cover characters and hide one bit in each will be equal to (16 % of cover size) { 6 bits of embedded hide in 8 bytes of cover }.
- The second scan differ from the first scan, in this scan 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, we 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 scans we scan 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).

3.1 What is pointer?

Pointers that we 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 used 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

  The 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:

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

3.2 Pointers type

1. First type call (P_1), which is neglect from word processor when displaying cover on screen. This pointer used 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).

3.3 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, plan text, and footer), pointers work on plan text, 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.

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 1: 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 2: 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 3: stego text

4. Conclusions

In this paper we suggested new method to hide text in text based on using many pointers, which are characters can interpreter as invisible character, or as a part of cover when the document display.

The suggested algorithm can hide up to 48% of cover size which is very good contribution in hiding the text in the text.
Also the proposed algorithm did not give any sign for embedding the data, and the stego text will not change by hiding the data, it is absolutely the same as the origin text.

REFERENCES

Ammar Odeh and Khaled Elleithy, 2012, "STEGANOGRAPHY IN ARABIC TEXT USING ZERO WIDTH AND KASHIDHA LETTERS", International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 3, DOI : 10.5121/ijcsit.2012.4301.

Monika Agarwal 2013, "TEXT STEGANOGRAPHIC APPROACHES: A COMPARISON", International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1, DOI : 10.5121/ijnsa.2013.5107.

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

Somdip Dey, 2012, An Image Encryption Method: SD-Advanced Image Encryption Standard: SD-AIES, International Journal of Cyber-Security and Digital Forensics (IJCSDF) 1(2): 82-88 The Society of Digital Information and Wireless Communications.

Wesam Bhaya, Abdul Monem Rahma and Dhamyaa AL-Nasrawi, 2013, "TEXT STEGANOGRAPHY BASED ONFONT TYPE IN MS-WORD DOCUMENTS" Journal

of Computer Science 9 (7): 898-904, doi:10.3844/jcssp.2013.898.904.