International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 6, Issue 1, Dec - Jan 2017; 89-96

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

UNLIMITED SIZE OF ENGLISH PLAIN TEXT-IN-TEXT HIDING ALGORITHM

MOHAMMED JAWAR KHAMI

Assistant Professor, Department of Computer Science, Basra Technical Institute,

Southern Technical University, Iraq

ABSRACT

When using internet as main communication infrastructure, people apart need their information to be protected from other third parties. Two widely techniques are used for it, cryptography and steganography. In cryptography the existence of the encrypted message is visible to the world. While steganography conceals the very existence of the message.

This paper concerns with steganography. It deals with text-in-text data hiding technique. The method of using non-printable characters of the Unicode standard characters are chosen to encode and hide English text into another English cover text to produce a stego text that can be used later, at the other end of the communication media, to extract and recover the exact secret text.

Two text-in-text and text-from-text hiding and extraction algorithms are written and coded in matlab programming language. Merits and drawbacks are shown for the proposed algorithms. The designed algorithms result in adding many enhancements to the implementation of the basic method of using the non-printable characters of the Unicode standard characters to text-in-text hiding technique. These enhancements include the size minimization of stego text file, reducing of hiding and extraction processing time and thus reducing communication time. From security point of view, key has been used to encrypt the secret text before hiding it and decrypt the text at the extraction stage.

KEYWORDS: Steganography, Text Steganograpy, Encryption, Data Hiding Algorithms

INTRODUCTION

Today, internet becomes as a key communication infrastructure for connecting peoples across the global. And hence, secure communications for both social and business fields through public and private channels become as more important issue. As this communication happened and developed into everyday activity, securing sensitive data became as a matter of great concern [1,2].

In many situations, people apart need their information to be protected from other third parties. Two widely used techniques are used, cryptography and steganography. Cryptography is the study of changing information appearance by scrambling secret data and keeping them overt. It is used when communicating over an untrusted medium such as internet [3]. Steganography is the art and science of writing hidden messages in such a way that no one apart from the sender and the receiver would realize that a secret communicating is taking place [4].

In cryptography the existence of the encrypted message is visible to the world. While steganography conceals the very existence of the message, and removes the unwanted attention coming to the hidden message. Cryptographic methods try to protect the content of a message, while steganography uses methods that would hide both the message as well as the

content. Thus, by combining steganography and cryptography one can achieve better security [5].

The general objective of steganography is to hide or embed secret message contents within another cover message or media of same type, or may be different one from that of the secret message. And without attracting attention, hide the message from anyone who does not know the existence of secret message [6]. The word **'hide'** here means not let human or devices see, show, print, or display the hidden message in normally used ways unless extract or recover it from the received message by specially designed tools.

Steganography methods and techniques are often called by type of the used cover (host), such as image, audio, video, and text data type and not by the data type of the embedded (secret) message[7]. Thus, data hiding techniques could be called as data hiding in image, in audio, or in text. In spite of its big needs, the last data hiding technique, data hiding in text, is less popular than the other data hiding techniques. This isn't because it less required in everyday activities but because text has less redundant elements (not like image, audio, and video data that have many redundant), and any change, even in one bit if any character ASCII code, will result in changing that character to another character from the ascii-table and hence change the text itself.

This paper deals with the text in text data hiding technique. And from the many methods and algorithms of how this technique works, the method of using non-printable characters of the Unicode standard characters are chosen to encode and hide English text into another English cover text to produce a stego text that can be used later, at the other end of the communication media, to extract and recover the exact secret text or with a minimum amount of perceivable degradation.

The proposed algorithm could be used to hide single letter, word, line, paragraph of many lines of text, or any number of plain text pages and hide them in any size of other cover English plain text. '**any size**' means here, that secret text size could be smaller, bigger than or equal to the size of the cover text. And also may mean an empty cover text (pages of lines, each line is made of null character concatenated with carriage-return and line-feed characters in computer environment). The basic idea implemented here is taken from researches by [8], and [9]. It employed the non-printing properties of some of the Unicode standard characters to encode the letters of English language first and then uses this code in embedding the secret message letter by letter into the cover-text.

Many enhancements have been added by the proposed algorithm to the origin basic hiding method of [8] include the way of how the algorithm deals with any size of both secret and cover text. And also, by taking some important points fromthe used language into consideration, like relative frequencies of letters in the English language, and how can this lead to the increase in processing speed by reducing total hiding time. Also, reducing the total required storage space for the stego text and thus reducing the transmission time at communication stage. The algorithm is programed and coded by MATLAB (R2015b) software and run on hp Pavilion dv6 PC.

ASCII TABLE AND UNICODE STANDARD SET DESCRIPTION

Computers read bytes and people read characters, so computer users use encoding standards to map characters to bytes. ASCII (American Standard Code for Information Interchange), was the first widely used standard, but covers only Latin (7 bits/character can represent 128 different characters). The ASCII code is the numerical representation of a character. Text in ASCII format means 'plain' text with no formatting such as tabs, bold or underscoring (the raw format that any computer can understand).

Unicode is an international encoding standard for use with different languages and scripts. It covers all possible characters in the world (up to 1,114,112 characters). It provides a unique number (code point), for every character, irrespective of the used platform, program, or language. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. These encoding systems may conflict with one another. That is, two encodings can use the same number for two different characters, or use different numbers for the same character. Unicode enables a single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

Unicode contains certain characters, control characters, that are not displayed (non-printable or non-graphical characters) like (U+200B), ZERO WIDTH SPACE 'ZWS', (U+200C), ZERO WIDTH NON-JOINER 'ZWNJ' and (U=200D), ZERO WIDTH JOINER 'ZWJ' These characters can be inserted into Unicode text to hide some secret information there.

The current text hiding method allows hiding/extracting English plain text into/from another English plain text. The hiding approach can be summarized by two main steps:

- Create a coding system, with at most six binary digits, to encode only those characters usually be used in typing and configuring English plain text (about 126 characters), and then replace all zero's and one's digits of the created code system by U+200C and U+200D respectively.
- Encode each secret text character into its representative sequence from the above created code system and convert, in order, one cover text character from the one-byte ASCII code point representation to two-byte code point representation. Insert secret code sequence before the converted cover code to make the stego text.

As an example on implementation of the above two steps, let us hide the secret massage text "AB" ina cover text of "abc". According to step (1), let the code given to the letter "A" is "200C 200D" and the code given to "B" is "200C 200C". From step (2), the ASCII code point of "abc" with the representations of 2-byte for each are "0097", "0098", and

"0099" respectively. And by inserting one character from the secret text message in front of its corresponding character in the cover text, the combined text will be "AaBbc" and the stego text will be as in Figure(1).

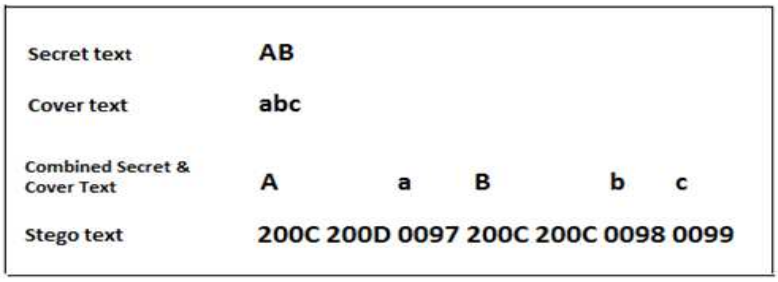

Figure 1: Text in Text Hiding Steps

And to extract secret text from the above stego text, sequences of secret text coded characters and character of the cover text must separated first and then the code values of each sequence of the secret text must be obtained as shown in Figure(2).

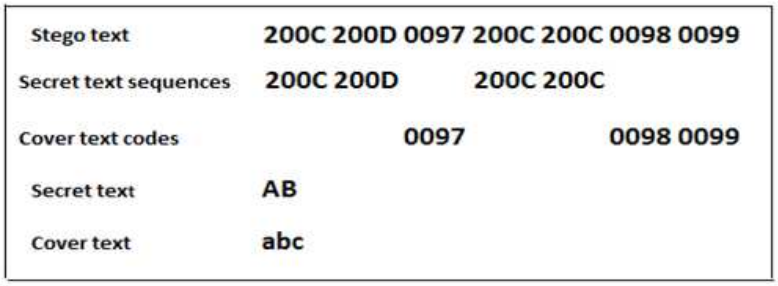

Figure 2: Extraction Hidden Text Steps

DESCRIPTION OF TEXT-IN-TEXT HIDING ALGORITHM

- Open secret and cover text files as a source of input, and one third file for output (to hold stego text).
- ***Loop-while-1***: While still there is un-read line of text in any of both secret and cover text files do:
  - Read all characters belong to one secret text line except those of its carriage return characters [combination of two characters: char (13) and char (10)]. Call it as the **current secret text line**.
  - Read all characters belong to one cover text line except those of its carriage return characters. Call it as the **current cover text line**.
  - Calculate lengths (in character), of both current secret and cover text lines.
  - Using current secret and cover text lines, the following two conditions must be satisfied:
- For each secret text line there must be a corresponding cover text line. And if there is no such line (cover text line), then create new line of spaces instead and consider it as the current cover text line.
- Each current cover text line must contain number of characters at least equal in number to those of the corresponding current secret text line. And if not, then append to the current cover text line number of spaces equal in number to the difference (in characters) between the two lines.
  - Some text editor's programs use few non-printable characters (control characters), in configuring and arranging text in their environment. These characters may disturb this configuration when the text is used in text-in-text hiding programs. Thus, it is needed to remove these non-printable characters from current secret and cover text. It can be done by calling a specially written function [Printable Char Only ()].
  - Append current secret and cover text lines with carriage return characters [char (13) + char (10)].
  - Since one or both of current secret and cover text line's lengths may change due to the implementation of Printable Char Only () function, thus both text line's lengths must be re-calculated again.
  - ***Loop-for-1***: Loop for each character in cover text line:
- If sequence number of current cover character is less than or equal to length of current secret text line, then encode the current secret character by calling specially written function [tablesearch ()], and append the obtained code-sequence to stego text line variable (StegoText).
- Append current character ASCII code of cover text to stego text line (StegoText).

- End ***Loop-for-1***.
    - Each character ASCII value in stego text line must be converted from one-byte (if it is), to two-bytes per character, i.e. as same as the ASCII coding used in Unicode character set.
    - Write stego text line to output file.
    - End ***Loop-while-1***.
- Close all opened files.
- End of text-in-text hiding algorithm.

DESCRIPTION OF TEXT-FROM-TEXT EXTRACTION ALGORITHM

- Open stego text file as a source of input, and another file for output (to hold the extracted secret text).
- Read the whole content of the stego text file. And partition it into lines depending on search operation for any character equal to char (10). Locations of char (10), found in stego text, can be used to determine start/end of each text line in stego text, number of lines, and also enable to determine all line's lengths.
- ***Loop-for-1***: Loop for each StegText line do:
    - Initialize new temporary variable and call it (TemSecret) to hold the extracted secret text from one stego text line.
    - ***Loop-for-2***: Loop for each character in current stego text line:

If current Unicode character value is equal to 200C or 200D then

- Append character Unicode value to a temporary variable (TemStego).
- loop to Loop_for-2.

Else

- Call teblesearch() function to get TemStego corresponding character and append it to TemSecret variable.
- Re-initialize temporary variable (TemStego).

End if

- End ***loop-for-2***.
- Write TemSecret on output file.
- End ***Loop-for-1***.

- Close all opened files.
- End of secret text extraction program section.

RESULTS OF IMPLEMENTATION OF THE ALGORITHMS

Let us apply the proposed text-in-text hiding algorithm to equal to, larger, shorter secret plain texthan a cover plain text. On each run of the proposed algorithm, sizes of secret, cover, and stego text files and total hiding time are recorded as in Table (1), in Table (1) sizes of the used files (secret, cover and stego text files), will be shown as pair of integers (C,L),where C is total characters contained in the text file while L is the total text lines in that file. Also assuming file lines may contain zero (empty line), some character in each, or all characters in one line and other lines are empty.

CONCLUSIONS

The general notes and conclusion points that may come out from applying the idea of Unicode character representation in encoding and then hiding secret text in known cover text by the proposed two algorithms, allow the user to compare it with other available methods of text in text hiding methods. Many experiments have been carried out on different sizes for both secret and cover text as in Table (1). The following merits and drawbacks of the method can be drawn as in following:

Table 1: Size of Input and Output Files

Input Files		Output File	Hiding Time (Seconds)
Secret Text (C, L)	Cover Text (C, L)	Steg Text s (C, L)	
(34,1)	(34,1)	(228, 1)	0.11
(34,1)	(54,1)	(268, 1)	0.11
(54,1)	(34,1)	(374, 1)	0.14
(54,1)	(175,3)	(616, 3)	0.14
(175,3)	(54,1)	(1218, 3)	0.23
(175,3)	(244,9)	(1470, 9)	0.26
(244,9)	(175,3)	(1758, 9)	0.26
(384,7)	(42359,425)	(87030, 425)	4.97
(42350,425)	(384,7)	(315938, 425)	31.49
(42350,425)	(42350,425)	(315408, 425)	31.14

- The method allows hiding any quantity of secret text (could be file of many pages), in any quantity of cover text. Quantity means here number of text pages, lines, and number of characters in any lines.
- Number of secret text lines must be at least one line. While actual number of cover text lines and number of characters in each of them can be zero (empty text) or more.
- Size of stego text file is not always equal to the direct sum of secret and cover text files sizes. But it can be much bigger than that. This comes from two reasons: the first cause is due to the special coding system applied to the characters of the secret text before hiding it in the stego text. and the second reason is due to converting the one-byte ASCII representation of any character in cover file to two-bytes long for representing its characters in Unicode character set representation.

The increment in stego file size can be considered as one of the major drawback in implementation of this method especially from communication side of view and when one needs to transfer stego text by internet applications such as WhatsApp, Telegrams, skype, e-mail and many others social communication programs.

- From security point of view, the algorithms use auto generated key seed (key seed value depends on either current secret or cover text line length), for redistribute the obtained secret code sequence of the secret text characters

between cover text characters before assigning it to the stego text. This method makes it difficult to the intruder to know the exact meaning of the hidden text even if intruder gets the hidden text by one way or another. The key implementation can be considered here as an encryption technique.

- Hiding text-in-text by applying this method (using some of the Unicode character as the base idea for text hiding in text), works very well from human visual system side view. But from other side it is not so well, since it can let the intruder to think, that something is not right with text currently using. So, if intruder looks to the size of text file (the one which is using it at that moment), and comparing it with the size of actual displayed text size, then definitely he/she will find the big difference between them, then intruder can easily think of something is hidden in and this may surely lead to challenge him/her to discover it.
- Computer time consumption is highly proportional to both secret and cover text file's sizes. This is because the method works in steps of single character at time. In addition, the method uses different character's manipulating and processing flow (it needs longer processing time when dealing with secret text characters than the time for cover text characters), It is due to the different processing flow for characters in both files.
- When file storage capacity, and transferring needs are not so important, text-in-text hiding by Unicode characters method has good features, advantages, and characteristics over the other hiding methods. These advantages include:
- High hiding capacity. In fact, unlimited text hiding capacity.
- It can hide text in an empty white page. i.e., it doesn't require usual written text as in other hiding methods.
- The recovered text is 100% undistorted and the method could produce exactly the same original text relating to number of text characters and their arrangement or locations on the original text pages.

REFERENCES

1. Abdul Monem S. Rahma, Wesam S.Bhaya, Dhamyaa A. Al-Nasrawi , "Text Steganography Based On Unicode of Characters in Multilingual" ,International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, www.ijera.com Vol. 3, Issue 4, Jul-Aug 2013, pp.1153-1165.
2. William Stallings, "Computer Security: Principles and Practice", Prentice Hall, 2nd edition, 2011.
3. Yaman F.Abdullah, Hebah H. O. Nasereddin, "Proposed Data Hiding Technique – Text under Text", American Academic & Scholarly Research Journal (aasrj),Vol. 5, No. 3, April 2013, (Special Issue).
4. Youssef Bassil, "A Text Steganography Method Using Pangram and Image Mediums ", International Journal of Scientific & Engineering Research (IJSER), ISSN 2229-5518, Volume 3, Issue 12, Dec. 2012.
5. Gayathri C., Kalpana V., "Study on Image steganography techniques", Hnternational Journal of Engineering and Technology (IJET), ISSN: 0975-4024, Vol. 5 No.2, Apr-May 2013.
6. Anuradha A., Hardik B. Pandit, "Review on information hiding techniques: a comparative analysis", International Journal of Research in Engineering and Technology (IJRET), pISSN:2321-7308, Vol. 05, Issue: 02, Feb. 2016.
7. Hebah H. o. Nasereddin, Murad Saleh Al Farzaeai,"Proposed data hiding technique text image inside image

(TIII)", IJRRAS, Vol. 4, Issue:2, Aug. 2010.

8. Akbas E. Ali," A New Text Steganography Method by Using Non-Printing Unicode Characters", Eng. & Tech. Journal, Vol. 28, N0o.1, 2010.
9. M.Shirali-Shahreza and S.Shirali-Shahreza, high capacity Persian/Arabic text steganography", Journal of applied sciences 8 (22): 4173-4179, 2008.