A Text Steganographic System Based on Word Length Entropy Rate

https://doi.org/10.3991/ijes.v5i3.7521

Francis X. K. Akotoye
University of Cape Coast, Cape Coast, Ghana
fakotoye@ucc.edu.gh

Abstract—The widespread adoption of electronic distribution of material is accompanied by illicit copying and distribution. This is why individuals, businesses and governments have come to think of how to protect their work, prevent such illicit activities and trace the distribution of a document. It is in this context that a lot of attention is being focused on steganography. Implementing steganography in text document is not an easy undertaking considering the fact that text document has very few places in which to embed hidden data. Any minute change introduced to text objects can easily be noticed thus attracting attention from possible hackers. This study investigates the possibility of embedding data in text document by employing the entropy rate of the constituent characters of words not less than four characters long. The scheme was used to embed bits in text according to the alphabetic structure of the words, the respective characters were compared with their neighbouring characters and if the first character was alphabetically lower than the succeeding character according to their ASCII codes, a zero bit was embedded otherwise 1 was embedded after the characters had been transposed. Before embedding, the secret message was encrypted with a secret key to add a layer of security to the secret message to be embedded, and then a pseudorandom number was generated from the word counts of the text which was used to paint the starting point of the embedding process. The embedding capacity of the scheme was relatively high compared with the space encoding and semantic method.

Keywords—Steganography, entropy, cover text, stego text, and information security

1 Introduction

The introduction of computers and, for that matter, information technology has caused the problem of information sharing and data exchange to drastically reduce. With the proliferation and improvement in information exchange techniques such as the internet, search engines, data download services, file sharing; these have occasioned the improvement and advancement in data-manipulation software and hardware. A whole new world of creating and exchanging information has been opened. It is considerably easy and cheaper to create, reproduce or modify data and transfer them at an amazing speed over the internet to a variety of end users and or devices.

The ease with which data can be edited and manipulated has given rise to a lot of questions concerning ownership and unauthorized tampering. In fact, digital data such as document, video, audio and images are easy to edit or manipulate and transfer easily and at low cost over the internet to other users both authorized and unauthorized. This tends to cause a lot of financial loss and judicial litigation that could drag on endlessly. It is for these reasons that steganography still enjoys heightened interest in the information security research community.

The field of steganography has generated a lot of interest both in the academic and industrial arenas considering its potential in enabling secret communication and for that matter secret data exchange between consenting parties. Although a lot of investigation has gone into and continues to go on in this field, the volume of work done in text steganography is quite restricted compared with other digital media domains (video, image, audio, etc.). The lack of interest by researchers to venture into text steganography may not be far-fetched considering the fact that softcopy text is in many ways the most difficult place to hide data [1], [2]. Hard-copy text can be treated as a highly structured image and is readily amenable to a variety of techniques such as slight variations in letter forms, kerning, baseline, etc. This is due largely to the relative lack of redundant information in a text file compared with a picture or a sound byte. While it is often possible to make imperceptible modifications to a picture, even an extra letter or period in text may be noticed by a casual reader.

This underlying characteristic of text documents provides the driving force behind defining new boundaries and techniques for marking text documents for covert communication. Studies in linguistics have shown that the frequency distribution of four-letter words make them suitable candidates to hide information in a text document. This work seeks to explore that scope by examining the embedding capacity, robustness and imperceptibility of the embedding method employing the entropy properties of English n-grams.

1.1 Theoretical Context

An English text, to a certain extent, can be considered as a random signal composed of a finite number of symbols (mainly the letters of the alphabet) [3]. Applying statistical techniques which are grounded in information theory, it is possible to estimate the entropy rate of words as shown in [4], [5]. Shannon's entropy measure came to be taken as a measure of the uncertainty about the realization of a random variable. It thus served as a proxy capturing the concept of information contained in a message as opposed to the portion of the message that is strictly deterministic by inherent structures. For instance, redundancy in language structure or statistical properties relating to the occurrence frequencies of letters or word pairs. However, it will be more appropriate to define entropy for text as based on the Markov model of text. For a 0-order source, that is each character is selected independent of the last characters, the binary entropy is:

$$H(s) = -\sum Pi \, log_2 Pi \tag{1}$$

where Pi is the probability of i (i.e. the random text) [6]. For a first-order Markov source, that is one in which the probability of selecting a character is dependent only on the immediately preceding character, the entropy rate is:

$$H(S) = -\sum_i Pi \sum_j Pi(j) \log_2 Pi(j) \quad (2)$$

where i is a state of certain preceding characters and pi(j) is the probability of j given i as the previous character or characters.

For a second-order Markov source, the entropy rate is:

$$H(S) = -\sum_i P_i \sum_j P_i(j) \sum_k P_{i,j}(k) \log_2 P_{i,j}(k) \quad (3)$$

In general the b-ary entropy of a source S = (S, P) with source alphabet S = {a1, ..., an} and discrete probability distribution P = {p1, ... , pn} where pi is the probability of ai (say pi = p(ai)) is defined by:

$$H_b(S) = -\sum_{i=1}^{n} P_i \, log_b \, P_i \quad (4)$$

2 Method

2.1 Proposed Algorithm

The basic principle of the proposed algorithm is based on the statistical nature of English words. The frequency of letters in English is not uniform, for example the most common letter "E" has a frequency of about 13% while "Q" and "Z" have a frequency of only 0.1% [3]. A further example of combining pairs of letters reveals that "TH" have a frequency of 3.15% [3] to top the list as the most occurring pairs in a word. It is in view of this statistics that this algorithm was proposed to harness the frequency and ubiquity of certain characters and, for that matter words, to hide data in. The data hiding process involves selecting words of at least four character long. The choice of at least 4-letter words is based on the statistical fact that the average length of English words in a text is 5 characters [7]. Each character starting with the first (x_0) is compared with the succeeding character (x_{n+1}) using their ASCll value. If $x_0 < x_{n+j}$, then a 0 bit of the secret message would be embedded, however, if $x_0 > x_{n+j}$ the position of x_{n+j} and x_0 are transposed and a 1 bit is embedded. The embedding continues with the next pair of two characters in the word. An obvious benefit of the proposed scheme is that the file size of both cover object and stego-object remain the same, also there is no noticeable modification of words involved in the embedding process.

The system takes a secret message, encrypts it with a chosen key and extracts the bit sequence of the encrypted messages. The embedding process begins only when the number of candidate words is at least the length of the bit strings of the encrypted message plus the additional bits. The candidate words are stored in a lookup table. The generated look-up table serves as a register to record the candidate words used in the embedding process as well as what changes were made in terms of the alphabetic ordering of their constituent characters. The key is used to generate a pseudo-random

number which is used to pick the starting point of the embedding process. In order to ensure that the file size of the stego-object does not vary too widely from the file size of the cover object, the lookup table is stored separately from the stego-object.

Input: Secret message *M* to be embedded, cover-text *C* and key *K*
Output: Stego-text *C′*
Step 1**:** Use stego-key $K \rightarrow K_E(M)$ to encrypt *M* and convert into bit string denoted as *M′,* set the
length of *M′* as *w*
Step 2: Add a start composed of three repeated sequential bit string of '0000 1111', and the
8-length bits of *w* before *M′,* denotes the entire bit string embedded into the text as *M″,*
set the length of *M″* as *I*
Step 3: Traverse all word in the text to compute the possible hidden capacity of *C*, denotes
capacity
as *L(C)* and the number of words as *N.* If $L(C) < I$, exit
Step 4**:** For each word, if the length of the word is less than 4, then skip it, else do the following
operation until reaching the end of *M″*
First, denote the word as $x_0\, x_1 \ldots x_m$, i-1, i ∈ $[1, \lfloor (m-1)/2 \rfloor * 2]$
1) when $x_{i-1} < x_i$, if the embedded bit is ' 1 ', then transpose x_{i-1} and x_i, else encode the
bit ' 0'
2) when $x_{i-1} > x_i$, if the embedded bit is ' 0', then transpose x_{i-1} and x_i, else encode the
bit '1 '

Fig. 1. Embedding Algorithms

Input: The Stego-text *C′* and the stego-key *K*
Output: The secret message *M*
For each word, if the length of the word is less than 4, then skip it, else do the following
operator until reaching the end of *M″*
Step 1**:** Denote the word as $x_0\ x_1 \ldots x_m\ i-1, i\ \in [1, \left\lfloor \frac{(m-1)}{2} \right\rfloor * 2]$, if $x_{i-1} < x_i$, extract the bit ' 0',
else extract the bit ' 1'
Step 2: If the first 24 bits are three repeated bit string '00001111 ', then extract the following
8 bits *w,* as the length of encrypted secret message, else return the result of non-secret
message, exit.
Step 3: Continue the process described by Step 1, and extract the encrypted secret message *M′*
that has length *w.*
Step 4: Decrypt *M′* to *M*, *using stego-key* $K \rightarrow K_E(M')$, output the secret message *M*

Fig. 2. Extraction Algorithm

The extraction process is simply an inverse of the embedding procedure. After supplying the stego-key which is seeded by the random number generator to identify the starting point of the embedding, each candidate word is extracted and processed according to its entry in the lookup table. The extracted bit strings are then decrypted into the secret message.

3 Analysis

A cover text of 197 words was used in the testing process. A word length count and frequency table was extracted from the test document and is presented below. For every 4-letter word, the maximum embedding capacity of the word can only be 2-bits. That is one bit hidden by the first two letters of the word and the second bit hidden by the last two letters of the word. In fact every 4-letter word in the document has a constant hiding capacity of 2 bits. However, this constant factor does not apply to varying word lengths as can be deduced from the third column of Table 1. Thus taking an average text line of 10 words per line which in fact is the minimum average [8] and a frequency distribution average of 12.3% for four letter words, the proposed scheme is capable be of embedding at least 2 bits of data on each line of text if assuming the lime contains only one four letter word and the other words are two or three letter words; though this in fact is not the case in most instances. The embedding bandwidth as compared to other marking techniques is relatively high [9]. Although almost half of the available bandwidth for embedding in the test data (885 bit space) was not used, the scheme could still embed a secret message of 16 bytes.

Table 1. Word count statistics of test document

Word Length (number of characters in a word)	Word Count	Embedding Capacity of each Word (bits)	Embedding Capacity of each Class of Word (bits)	Frequency of Word in Document (%)
1 character	5	-	-	2.5
2 characters	35	-	-	17.8
3 characters	39	-	-	17.8
4 characters	28	2	56	14.2
5 characters	12	2.5	30	6.1
6 characters	16	3	48	8.1
7 characters	13	3.5	45.5	6.6
8 characters	12	4	48	6.1
9 characters	18	4.5	81	9.1
10 characters	8	5	40	4.1
11 characters	3	5.5	16.5	1.5
12 characters	2	6	12	1.0
13 characters	2	6.5	13	1.0

4 Conclusion

As the capabilities of the internet continues to experience rapid expansion and diversity, so also is the production, distribution and use of digital media which is the main traffic of exchange on the internet. These changes have brought about new challenges in how the integrity of digital documents could be protected whilst still maintaining their economic and aesthetic values. The answer to this challenge no doubt lies in steganography. In this work we explored the possibility of harnessing the entropy feature of at least four-letter words in a normal text document, by exploiting the alphabetic ordering of their constituent characters to hide information. The experimental results have shown promising features in the sense that embedding capacity is quite appreciable, also there is no noticeable change in file sizes between the cover file and the stego file. The proposed algorithm is also simple to implement considering its relatively high embedding capacity. These features make this algorithm suitable for fingerprinting and authentication system applications.

5 References

[1] H. Singh, P. K. Singh, and K. Saroha, "A survey on text based steganography," in *Proceedings of the 3rd National Conference*, 2009, pp. 26–27.

[2] A. Al-Azawi and M. A. Fadhil, "Arabic text steganography using kashida extensions with huffman code. J," *Applied Sci*, vol. 10, pp. 436–439, 2010. https://doi.org/10.3923/jas.2010.436.439

[3] "entropy of English," entropy of English - Everything2.com. .

[4] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001. https://doi.org/10.1145/584091.584093

[5] C. E. Shannon, "Prediction and entropy of printed English," *Bell Labs Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951. https://doi.org/10.1002/j.1538-7305.1951.tb01366.x

[6] G. Barnard, "Statistical Calculation of Worcl Entropies for Four Western Languages," *IRE Transactions on Information Theory*, no. 1, pp. 49–53, 1955. https://doi.org/10.1109/TIT.1955.1055123

[7] J. Johansson, "The great debates: Pass phrases vs. passwords," *Security Management Ocotber*, 2004.

[8] "Generally accepted number of words per page," *Google answers*. Dec-2005.

[9] D. Zou and Y. Q. Shi, "Formatted text document data hiding robust to printing, copying and scanning," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 4971–4974.

6 Author

Francis Xavier Kofi Akotoye received his M.Eng. in Computer Science and Technology from the Hunan University, Changsha in 2007. In 2008 he was appointed as lecturer in the Department of Computer Science and Information Technology of the University of Cape Coast. His research interests are in data security, cloud computing and internet of things.

Article submitted 02 August 2017. Published as resubmitted by the author 14 September 2017.