Dictionary Module and UDC: Two new approaches to Enhance Embedding Capacity of a Steganographic Channel

Vidyasagar M. Potdar, Song Han, Elizabeth Chang

School of Information Systems, Curtin University of Technology, Perth, Western Australia
e-mail: Vidyasagar.Potdar, Song.Han, Elizabeth.Chang@cbs.curtin.edu.au

Abstract — **In this paper we present a new technique to enhance embedding capacity of a steganographic channel by preprocessing the secret data. Here we limit ourselves to textual data only. By preprocessing we don't mean compression; secret data can be compressed after it has been preprocessed by our technique. W e introduce the concept of Scrambled Letters, User Defined Codes and the Dictionary Module to explain our technique. All these concepts when applied together give a phenomenal embedding capacity. Theoretical results show that we can achieve at least 25-30% increase in embedding capacity. We introduce two techniques, one in which we only apply the dictionary concept while the other in which we use the user defined codes along with the dictionary approach. The former technique is generic and can be applied to any form of textual data, whereas the latter can only be applied to pure text, i.e. without any form of graphics like images or graphs. Once the data is preprocessed by using our technique it can be easily embedded in any steganographic cover medium by using any steganographic algorithm.**

Index Terms — **Scrambled Letters, User Defined ASCII Codes, Dictionary Module, Steganography, Data Embedding Capacity, GLM Algorithm.**

I. Introduction

Steganography, meaning "covered writing", includes methods of transmitting secret message through innocuous looking cover mediums in such a manner that the existence of the hidden message is undetectable. Using steganographic techniques we can hide secret information in digital image files, digital audio and video files, or any other digital media which have some redundant bits that can be replaced to hide secret data. Pre-computing steganography has a long history, but digital steganography as a research field is avante garde.

II. Addressed Problem

In this paper we address the issue of embedding capacity of a steganographic channel. All existing techniques compress the secret data before embedding it in a cover medium to achieve higher embedding capacity. We propose to preprocess the secret data before even compressing it. We introduce this step to reduce the size of data set which would be compressed and finally embedded. To preprocess the data we introduce two techniques. The first technique is generic and can be applied to any form of textual data, whereas the second technique can only be applied to pure text, i.e. text without any kind of formatting, or graphics like images or graphs. Both these techniques reduce the initial size of the data set. In the first technique we introduce the concept of Dictionary to reduce the data set, whereas in the second technique we introduce the concept of User Defined Codes. Both these concepts are explained in detail in the later sections.

This paper is organized as follows: in §III we give an overview of existing research in image steganography; in §IV we detail our proposed method; in §V we give examples to prove the concepts, and in §VI we discuss the preliminary results. Finally, our conclusions and future directions will be given.

III. Image Steganography

Simmons first introduced the concept of steganography in the early 1980s when he discussed the prisoners' problem [8]. He discussed the situation in which two prisoners who are locked in different cells have to communicate innocuously without raising any suspicion. He used the idea of subliminal channels instead of steganography. This was one of the first works in the field of steganography.

The literature shows the existence of a variety of techniques for using data so that it can be embedded in images [2,3,4,5,6,7]. Image steganographic techniques can be classified on the basis of the domains in which data is embedded. Basically there are two domains: the spatial domain and the transform domain. Steganographic techniques try to embed data in these domains.

In the spatial domain image steganography, the simplest technique is to embed data in the least significant bit (LSB) of each pixel in the cover image. The LSB Replacement technique alters the insignificant information in the cover image. It places the embedding data at the least significant bit (LSB) of each pixel in the cover image.

Recently, some steganographic techniques have been reported which directly modify the pixels to embed data. Some of them are reported here [9]. Potdar et al. (2004) showed how data can be directly embedded in the spatial domain of images by directly modifying the absolute values of pixels. Most of the techniques discussed in the literature uses error correcting codes (ECC) to manage signal distortions. However using ECC codes increases the amount of data that is embedded. In this paper we propose a practical approach to tackle the signal distortion issue.

IV. Conceptual Framework- Dictionary Module and UDC To Enhance Embedding Capacity

In this section we propose our conceptual framework. We discuss two different techniques to reduce the data set in order to achieve higher embedding capacity. Literature shows the existence of compression as the only technique to reduce the data set. However, we propose to preprocess the data before compressing it. Hence we add one extra step before compression. This is shown in Fig. 1. We initially have *'Raw Textual Data'*, which we process using our technique to get the *'Processed Data'*, which is then compressed to get the *'Compressed Data'*. Each of these steps reduces the size of the data by a certain factor. We assume (and later prove) that this process significantly reduces the amount of data that we need to embed in a steganographic cover medium. We initiate this step to reduce the size of data set which would be compressed and finally embedded. To preprocess the data we introduce two techniques. Both these techniques reduce the initial size of the data set. In the first technique we introduce the concept of Dictionary to reduce the data set, whereas in the second technique we introduce the concept of User Defined Codes. We will now explain them in detail.

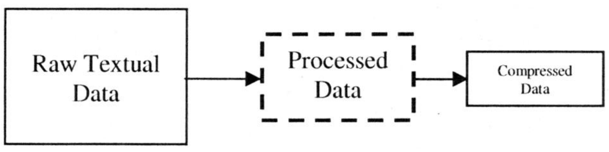

Fig.1 Rate of change in the data size with each step

A. *Technique One: Dictionary Approach*

This technique is a generic technique and can be applied to any form of textual data, i.e. word processor file which has some text, some graphs and some images. By using this technique we only process the textual portion. The basic idea behind the Dictionary Approach is to process each word one after the other to represent it in as minimum letters as possible. When we mean processing each word, we mean skipping some letters from a word in a predetermined order, in such a way that the word can be properly regenerated, if we know which letters are skipped and from where. In the proposed algorithm we decide to skip alternate letters in a word.

To explain this we give an example. Suppose we have a word 'Steganography'. We keep the first and the last letter intact, while we delete the alternate letters within, i.e. instead of '**S**t**e**g**a**n**o**g**r**a**p**h**y**' we can have 'S e a o r p y'. As a result of this step we reduce the size of the data. We would only transit 'S e a o r p y' instead of the complete word 'Steganography'. But does that mean we can automatically recover the word 'Steganography' from 'S e a o r p y'? No! Hence to recover the word properly we need some black box, which takes in the processed word [S e a o r p y] and returns the original word [Steganography]. This black box we term the *'Dictionary Module'* (DM).

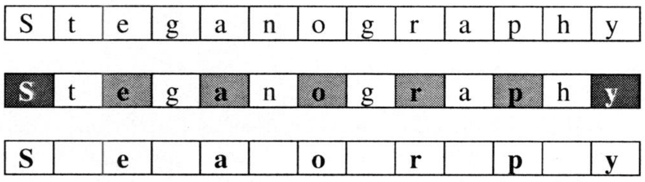

Figure 2: Keep the first and the last letter intact, while we delete the alternate letters within.

The DM is a small database of words similar to a dictionary. But these words are stored in a sorted order, which is different from a normal dictionary. The sorting criteria are:
# Alphabetical Order: Firstly all the words are stored in an alphabetical order.
# Ending Letter: Secondly the words are sorted according to the letter with which they end but maintaining the alphabetical order. E.g. [Steganographi**c**, Steganograph**y**]
# Number of letters in a word: This sorted list is finally sorted based on the number of letter they have maintaining the above two criterion.

DM takes the processed word and some parameters to uniquely identify the proper word. In case of the example discussed earlier we can recover 'Steganography' from 'S e a o r p y' using this module. This process is explained in Figure 2.

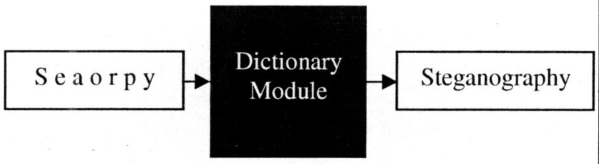

Figure 2: Dictionary Module

We now discuss the technical details of the DM. Basically DM is a search algorithm which takes four parameters:

# *Length of the original word*: This can be deduced form the processed word.
# *The Starting and the Ending letter of the processed word*: This is the same as in the original word, e.g. S and Y.
# *List of letters from the processed word*: E.g. In case of 'Seaorpy', the list of letters would be [e,a,o,r,p]
# *Location of these letters*: E.g. 'e' is 3rd, 'a' is 5th , 'o' is 7th etc.

After having all these parameters, the DM conducts a search looking for words that satisfy this criterion. The outcome of this search is the original word. Thus we can regenerate the processed word based on the DM. The DM that we build had all the words listed in an Oxford Dictionary.

Original Word	Processed Word
Information	I f r a i n
Embedding	E b d i g
Secret	S c et
The	T e
Dictionary	D c i n ry
Technique	T c n q e

Table 1: Sample outcomes form the DM

The concept of DM was inspired by the article posted on the Internet [1] which said misspelled words can be interpreted properly as long as the first and last letters are correct and are in their proper place, even though the letters in between can be scrambled or even missing. If you read the paragraph in Fig 3 you can see that there are several spelling mistakes. Even though there are so many spelling mistakes one can easily interpret each word and understand what the sentence tries to convey. One more interesting fact about the words in Fig 3 is that in every word the first and the last letter are in its proper location and the letters in between are jumbled up. If we maintain this minimum constraint we can interpret each word easily.

'Aoccdrnig to a rscheearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht frist and lsat ltteer is at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae we do not raed ervey lteter by it slef but the wrod as a wlohe.'

Fig.3 Words with scrambled letters

This is because humans can assign meaning to data by means of the known conventions used in their representation[1]. Thus when we developed the DM, we introduce the constraint of skipping alternate letters from a word, and this worked very well. Another important point that we want to convey from this description is that if we want to transmit information we can rely on partial data, provided we can regenerate is completely. This also supports Shannon's Information Theory.

B. *Technique Two: Dictionary Approach with User Defined Codes*

This technique is more specific and can be applied to only pure form of textual data, i.e. without any other objects like graphs, tables, images etc. By using this technique we can process purely textual information. The basic idea behind this technique is to define a set of codes which can be used to represent the textual information. We term such codes as User Defined Codes (UDC). An example of such a code set is shown in Table 2. UDC is a set of codes that represent alphabets, numbers and some special characters rather than using the standard 8 bit ASCII code. Users can define it according to their needs.

Character	Code	Character	Code	Character	Code
a	00000	k	01010	u	10100
b	00001	l	01011	v	10101
c	00010	m	01100	w	10110
d	00011	n	01101	x	10111
e	00100	o	01110	y	11000
f	00101	p	01111	z	11001
g	00110	q	10000	.	11010
h	00111	r	10001	;	11011
i	01000	s	10010	,	11100
j	01001	t	10011	>	11101
				space	11110

Table 2: UDC using 5bit Codes

Table 2 shows such a representation using 5 bit UDC. However, this has some constraints. For example, numbers are not represented in numeric format; rather they would be represented in textual format (i.e. 1 = one and 20 = twenty). But having such a constraint can save a lot of space. We can make an immediate savings of 37.5% because we represent data using 5 bit UDC rather than the standard 8 bit ASCII.

From a practical point of view we can use 6 bit UDC, which would include characters like [a-z], [0-9] and some other outlined in Table 3. We can even manage by using either small (a-z) or capital letters (A-Z) rather than using both.

{	[	!	@	#	$	%	?	&	*
(	)	]	-	+	=	}	/	“	‘
:	;	<	,	>	.	Space			

Table 3: Special Characters included in 6 bit UDC

If we use 6 bits to represent one character, it reduces the data by 25%. As pointed out by Shannon, data is used to represent information and we can use ***Codes*** to represent the data that we want to transmit. We try to keep the codes as small as possible to increase the capacity of the transmission channel. We understand that our aim here is to transmit information (in as few bits as possible), and the data used to represent information can be defined according to the situation. Our aim here is to transmit information which is in the form of alphabets, words or numbers. Hence we design our own code instead of using the default ASCII code which used 8 bits to represent one character. These codes can be defined by the user to suit one's own needs. The concept of UDC has been introduced to achieve higher embedding capacity and reduce stego distortion. If we only use 63 commonly used characters to write text we can represent them

[1] www.jfcom.mil/about/glossary.htm

by only 6 bits instead of using 8 bits, which is the normal case.

V. Algorithm Description: Implementation Details

A. Data Preprocessing Algorithm

Data Processing Step

Input: Raw Textual Data, Output: Processed Data

1. Begin Data Preprocessing
2. Open the text data.
3. Read the first (or next) word.
4. Keep the first and the last letter intact.
5. Delete the alternate letters from the remaining word.
6. Any more words left?
7. If yes then go to Step 3
8. Else End Data Preprocessing.

After the Step 8 is completed, we can conveniently compress the processed data and embed it in any steganographic cover medium using any steganographic algorithm.

Data Retrieval Step

Input: Processed Data, Output: Raw Textual Data

1. Begin Data Retrieval.
2. Open the processed data.
3. Read the first (or next) word.
4. Keep the first and the last letter intact.
5. Calculate the number of letters in the word
6. Perform the search in the DM.
7. Identify the word.
8. Any more words left?
9. If yes then go to Step 3
10. Else End Data Retrieval.

Next we describe data preprocessing using UDC approach. Here the user has the freedom to decide on the UDC table. Based on the situation under consideration one can either choose a 5 bit or 6 bit UDC.

Data Processing Step

Input: Raw Textual Data in 8 bit ASCII

Output: Binary String of the Raw Textual Data based on UDC

1. Begin Data Preprocessing
2. Open the text data.
3. Define an empty binary string.
4. Read the first (or next) word.
5. Convert it into binary based on the UDC.
6. Concatenate it to the main binary string in Step 3.
7. Any more words left?
8. If yes then go to Step 4
9. Else End Data Preprocessing.

Data Retrieval Step

Input: Binary String of the Raw Textual Data based on UDC

Output: Raw Textual Data in 8 bit ASCII

1. Begin Data Retrieval.
2. Access the binary data string.
3. Define a new String.
4. Read the first 5 bits (or 6 bits) depending on UDC.
5. Map it to the equivalent character.
6. Concatenate it to the main String in Step 3.
7. Any more bits left?
8. If yes then go to Step 4
9. Else End Data Retrieval.

The detail description of the embedding and the extraction algorithm is outside the scope of this paper. But any steganographic algorithm could be used to embed the processed data.

VI. Conclusion

In this paper we introduced the concept of Dictionary Module and UDC. We showed how we can generate a word exactly provided we have the first and the last letter in the correct place and the word satisfies certain criteria. As a result of this, our technique uses less data to represent more information.

VII. References

[1] Article posted on Slashdot.org Website by an Anonymous Author. Mon Sep 15, 2003. Available on the internet at http://science.slashdot.org/science/03/09/15/2227256.shtml?tid=133&tid=134&tid=186

[2] Khan, M., Potdar V, Chang E., 'A prototype implementation of Grey Level Modification Steganography', in *Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society,* IECON 04, Korea.

[3] Lee, Y. K. & Chen, L. H., 2000. High Capacity Image Steganographic Model. *In IEE Proceedings Vision, Image and Signal Processing*, vol. 147 no. 3, pp. 288-294.

[4] Newman, R. E., Moskowitz, I. S., Chang, L., Brahmadesam M. M., 2002 " A Steganographic Embedding Undetectable by JPEG Compatibility Steganalysis". In: *Petitcolas (Ed.): Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands, October 7-9, 2002 LNCS Springer Verlag 258-277.*

[5] Potdar V, Chang E. 'Grey Level Modification Steganography for Secret Communication', *2nd IEEE International Conference on Industrial Informatics (INDIN2004)*, Berlin, Germany, June 24-26, 2004

[6] Provos, N., "Defending Against Statistical Steganalysis" 2001. In *Proceedings of the 10th USENIX Security Symposium*, pages 323-335, August 2001

[7] Sallee, P., "Model-Based Steganography" 2003. In International Workshop on Digital Watermarking, Seoul, 2003,

[8] Simmons, G. J., 1984. " The prisoner's problem and the subliminal channel" In *Advances in Cryptology -- CRYPTO '83"*, D. Chaum, ed., Plenum Press, 1984, 51-67.

[9] Soo-Chang, P, Jing-Ming, G., 2003. Hybrid pixel-based data hiding and block-based watermarking for error-diffused halftone images. *In IEEE Transactions on Circuits and Systems for Video Technology*. 13(8), 867- 884.

[10] Westfeld, A., 2001. "High Capacity Despite Better Steganalysis (F5-A Steganographic Algorithm)", In: *Moskowitz, I.S. (eds.): 4th International Workshop on Information Hiding*, LNCS, Vol. 2137. Springer-Verlag, New York, pp. 289--302, 2001.