Implementing Secure Communication on Short Text Messaging

Avinash Chandragiri, Peter A. Cooper, Yanxin Liu and Qingzhong Liu

Department of Computer Science, Sam Houston State University, Huntsville, TX 77341, USA
Emails: akc021@shsu.edu; cooper@shsu.edu; yanxin@shsu.edu; liu@shsu.edu

Abstract—On-line Short Text (OLST) in social networking tools such as micro blogs, instant messaging platforms, and short message service (SMS) via smart phones has become a routine in daily life. OLST is appealing for personal covert communication because it can hide information in a very short carrier text, and this concealment is hard to detect due to the diversity of normal traffics. However, designing appropriate schemes confronts several challenges: they need to be provably secure, and their performance needs to maintain high efficiency and handy usability due to the short length of OLST messages. In this paper, we implement a family of customized Cryptographic schemes known as HSym, HCod, HNum, and HPhs and Steganographic schemes HMea, HAbr, and HEmt for text hiding in OLST. These schemes are evaluated in terms of their security and their performance with regard to metric that address the particular characteristics of OLST: hiding rate. All implemented schemes are proved to be at least computationally secure, and their performance in terms of hiding rate justifies their applicability in social networking tools that utilize OLST.

Keywords— Steganography, secure communication, short text, text messaging.

I. INTRODUCTION

Steaganography is the art or practice of concealing information within another digital cover file, which is currently dominated by multimedia files such as image, video, and audio data [4, 5] and most literatures in steganography and the detection have been focused on multimedia-based steganography and steganalysis [15,16,17,18,19,20,21,22, 23]. Recently, text steganography, where information is hidden in carrier (or cover) texts, has attracted considerable interest from researchers [6]. Text cover files are very different from image, video and audio multimedia data files, since multimedia data files generally have good room of redundancy to accommodate the hidden data, and human eyes/ears cannot perceive the changes.

In text steganography, a cover text is usually of ordinary length, such as a Microsoft Word document [6, 7]. While text steganography is different from multimedia-based steganography, On-Line-Short-Text (OLST) [1] is even quite different from cover texts in terms of information hiding. It has a limited word or symbol count, and it is usually input by hand. The text may include some personal touches such as shorthand acronyms, special phrases, emotion symbols (“emoticons”), and preferred formats or styles. Currently, several special information hiding methods exist for specific languages [10], and some methods for text hiding have been proposed [11]. However, most existing schemes for text hiding have not yet been strictly justified in a formal framework for provable security. More importantly, because OLST has its own special characteristics, new methods that can address these particular features are needed. Compared to hiding methods for cover texts, those for OLST require more subtle justification. Moreover, the performance of text hiding for OLST is different from other types of text steganography. For example, the concealing rate needs to be very high because the text is short, and the ease of use for users must be maximized because OLST is usually input manually. OLST usually has a client tool that provides graphic user interface to facilitate text input. People using OLST platforms communicate with each other by typing characters and punctuation. Most client tools also support shortcut input of emotion symbols (“emoticons”) for “angry”, “sad”, “happy”, “frustrated”, etc. (there are dozens of such symbols), and the character size and font can also be set as a customized option by users. Usually, chatting occurs between two peers; however, chatting can also occur between a peer and a group, where all members of a group can receive one peer’s input. In this case, the intended recipient in the group is the information reveler, and the others are observers).

II. PROBLEM FORMULATION

Generally speaking, there are three entities in OLST:

(1) An information hider [1]: A person or a program that hides hidden information in OLST;

(2) A hidden information reveler [1]: A person or a program that recovers hidden information in OLST; and

(3) An observer [1]: A person or a program that fully accesses the OLST and may be aware of the existence of hidden information in the observed OLST.

The formal definitions used are as follows.

Definition 1 Information hider is a Polynomial Time Turing Machine (PTTM) that can hide hidden information in carrier information.

Definition 2 Hidden information reveler is a PTTM that can recover hidden information from carrier information.

Definition 3 Observer is a PTTM that observes carrier
information that hides hidden information.

**Definition 4** Hidden information [1] is hidden by information hider. Usually, Hidden information belongs to \{a, b, c……z, A, B, C……Z, Symbols\}.

**Definition 5** Carrier information [1] is transferred from information hider to hidden information revealer and hides hidden information.

### III. EXISTING SYSTEMS

#### A. Word Shifting

Word Shifting [12] method is a method of altering a document by horizontally shifting the locations of words within text lines to encode the document uniquely. This method is identified less, because change of distance between words to fill a line is quite common. But if somebody was aware of the algorithm of distances, they can compare the present text with the algorithm and extract the hidden information by using the difference. Although this method is very time consuming, there is a high probability of finding information hidden in the text.

#### B. Hiding using whitespaces

This concept is very straightforward. A message to hide is first converted into a binary format. Then, every bit whose value is 1 is represented by an extra whitespace between a particular set of two words in the carrier text; whereas, every bit whose value is 0 leaves the original single whitespace between the next particular set of two words. For example, “the boy went to school today” can be deciphered as “101001”. In fact, two spaces exist between “the” and “boy”, between “went” and “to”, and between “today” and the end of the sentence. This results in a bit of value 1 in positions 0, 2, and 5 respectively. In contrast, only a single space exists between “boy” and “went”, between “to” and “school”, and between “school” and “today”. This results in a bit of value 0 in positions 1, 3, and 4 respectively. Basically, the whitespace technique is very suspicious as a normal reader would right away notice the existence of some extra whitespaces in the text. Additionally, this method cannot encode too much information especially in small text.

### IV. PROPOSED METHODS

In this section we implement new text hiding methods for Online Short Text (OLST).

#### 1. Cryptographic Methods

##### A. Hiding by Symbols: HSym

In this method the text is hidden by mixing some character to original text. The resultant hiding text will be send to others.

**Input:** Text (what)  
**Output:** Hidden Text (dfefe\{dd)

**Algorithm:**
1: Read input Text T  
2: Get each Character C
3: for each Character in C  
   a. Get ASCII Value of Character C  
   b. Add Random integer Value  
   c. Convert to equivalent character.  
   d. Add to array  
4: Do steps a, b, c, d until you reach end of the input.

##### B. Hiding by Coding: HCod

In this method the text is hidden by some code. Here we are using Huffman code algorithm for hiding the text in the form binary form.

Huffman coding [1] is a form of statistical coding, not all characters occur with the same frequency, yet all characters are allocated the same amount of space. Code word lengths are no longer fixed like ASCII. Code word lengths vary and will be shorter for the more frequently used characters.

**Input:** Text (what)  
**Output:** 00011010111001111011

**Algorithm:**
1. Scan text to be compressed and tally occurrence of all characters.  
2. Sort or prioritize characters based on number of occurrences in text.  
3. Build Huffman code tree based on prioritized list.  
   a. Count up the occurrences of all characters in the text.  
   b. What characters are present?  
   c. What is the frequency of each character in the text?  
   d. Create binary tree nodes with character and frequency of each character  
   e. Place nodes in a priority queue– The lower the occurrence, the higher the priority in the queue.  
   f. While priority queue contains two or more nodes
      - Create new node  
      - Dequeue node and make it left sub tree  
      - Dequeue next node and make it right sub tree  
      - Frequency of new node frequency of left and equals sum of right children  
      - Enqueue new node back into queue  
      - Dequeue the single node left in the queue.  
      - This tree contains the new code words for each character.  
      - Frequency of root node should equal number of characters in text.  
4. Perform a traversal of tree to determine all code words.  
5. Scan text again and create new file using the Huffman codes.

##### C. Hiding by Number: HNum

In this method the text is hidden by mixing numeric values to original text. The resultant text will be in the form of number.

**Input:** Text (what)  
**Output:** Number (436)

**Algorithm:**
1: Read input Text T
2. Split the text into words where space occurs and store them in an array.
3. Get the element in the array and store it as a string.
4. Until you reach the end of the string do the following:
   A. Get each character in the string.
   B. convert the character into integer and add random number to it.
5. Add the values of each character in the string.
6. Add it to a string.
7. Do steps 3, 4, 5 until you reach end of the text.

D. Hiding by Phrase: HPPhs
As many phrases are involved in chatting, we therefore propose HPPhs, a method that uses phases to represent hidden information. HPPhs can easily generate carrier text actually, it can hide any hidden information independent from carrier information. The content of carrier information has no relation to hidden information, so perfectly secure hiding is guaranteed. We select the most frequently used phrases [1] and create a file that includes interjections (“ah”, “well”, “haha”, “oh”, “hey”, and “yeah”), remarks (“wait”, “OMG”, and “gosh”), modal particles (“hmm” and “ok”), emotion symbols (“:-(,” or “:-),”), punctuation (“?”, “!”), and phrases (“BTW” (by the way), “FYI” (for your information), “IC” (I see), “TY” (thank you), “great”, “neat”, and “cool”).

Algorithm:
1. Store list of phrase words
2. Get each Sentence S
3. for each Sentence in Text
   A. select phrase
   B. hide the original S with selected Phrase
4. Display hide info

2. Steganographic Methods

A. Hiding by Meaning: HMea
In this method the information hider and the revealer share the table which contains the information about the words in American English which mean the same as in British English, but spelled differently.

Check list
0 $\rightarrow$ British English
1 $\rightarrow$ American English

Example:
Embedding: movie in my flat (“secret 10”) 
movie in my flat
Extracting: movie in my flat 
“secret 10”

B. Hiding by Abbreviations: HAbr
In this method we make two lists of abbreviated forms one with usual abbreviations and other with chat abbreviations.

Information hider and revealer share the usual abbreviation list, chat abbreviation list and check list.

Check list
0 $\rightarrow$ full form, 1 $\rightarrow$ abbreviated form

Information hider and revealer share the usual abbreviation list, chat abbreviation list and check list.

Example:
Embedding :
“I am dng mastersofscience at SHSU.”
secret “101”

“I am dng mastersofscience at SHSU.”

Extracting :
“I am dng mastersofscience at SHSU.”
secret "101"

C. Hiding by Emoticons: HEmt
In this method we hide the information in Emoticons symbols that are being used most frequently in chat. The information revealer and hider share certain set of emoticons list. Based on the position and appearance of the emoticons its meaning changes.

Check list:
Emoticon Sentence=0
Sentence Emoticon=1
-Emoticon =10
Emoticon+=11
Emoticon’= Equivalent number assigned to emoticon
(Emoticon)$^2$ = First letter of name of emoticon
Other emoticon symbols = Do nothing.

V. ANALYSIS

A. Security analysis
In all our methods we have used random generator which has high security properties:

Pseudo randomness: The generator’s output looks random to an outside observer.

Forward security: The third party which learns the internal state of the generator at a specific time cannot learn anything about previous outputs of the generator.

Break-in recovery / backward security. The third party which learns the state of the generator at a specific time does not learn anything about future outputs of the generator, provided that sufficient entropy is used to refresh the generator’s state.

Even if the third part know the algorithm and the reverse engineering is performed in knowing the original text it will consume a lot of time (days/years).
B. Performance analysis

Using cryptographic methods we hide all the symbols as we are demolishing the original text into a disguised text which can provide almost an unbreakable security which proves that its hiding rate is 100%, but it gives a suspicious look. Using steganographic methods the number of symbols hiding will be less but it will be high secured as no one knows the meaning. The message will not be in a suspicious form. For HMea let there be two sentences’ in which we are hiding 3 symbols for every sentence and every sentence contains 5 words and each word is of 5 characters. Therefore hiding rate is 6/25. For HAbr and HEmt, use of abbreviations and emoticons in chat or SMS is high, usually more than half of the text will be either in emoticons or abbreviations, therefore the hiding rate will be more than 50%.

VI. Conclusion

The implemented family of text hiding schemes in OLST known as HSym, HCod, HNum, and HPhs, HMea, HAbr, and HEmt. Their security and their performance in terms of hiding rate were examined. Analysis shows that all the implemented schemes have at least computationally secure hiding. Moreover, the schemes hiding rate guarantees their applicability in social networking tools that utilize OLST.