A Survey on Text Based Steganography

Hitesh Singh, Pradeep Kumar Singh, Kriti Saroha
School of Information Technology, Center for development of Advance Computing, Noida, India
hitesh.singh.85@gmail.com, pradeep_84cs@yahoo.com kritisaroha@cdacnoida.in

ABSTRACT
In a modern era of Information Technology, illicit copying and illegal distribution accompany the adoption of widespread electronic distribution of copyrighted material. This is the main reason why people think about how to protect their work and how to prevent such unlawful activities. For this purpose various methods including cryptography, steganography, coding and so on have been used. Steganography is the best-suited technique that allow user to hide a message in another message (cover media). Most of steganography research uses cover media as pictures, video clips and sounds. However, text steganography is not normally preferred due to the difficulty in finding redundant bits in text document. To embed information inside a document its characteristics should be altered. These characteristics can be either the text format or characteristics of the character. But the problem is that if slight change has been done to the document then it will become visible to the third party or attacker. The key to this problem is that to alter the document in such a way that it is simply not visible to the human eye yet it is possible to decode it with computer. For this purpose various methods of text-based steganography have been purposed like line shifting, word shifting, feature coding, white space manipulation etc. In this paper, we present an overview of the steganography, with a particular focus on text-based steganography in details.

Keywords: copyrighted material, cryptography, steganography, and text-based steganography.

1. INTRODUCTION
Though security is nothing new, the way that security has become a part of our daily lives today is unprecedented. From pass codes that we use to enter our own highly secure homes, to retina-scanning technology that identifies us as we enter our office buildings, to scanners in airports, we have made security technology as much a part of our daily lives as the telephone or automobile. We are also surrounded by a world of secret communication, where people of all types are transmitting information as innocent as an encrypted credit card number to an online store and as insidious as a terrorist plot to hijackers. The schemes that make secret communication possible are not new. Julius Caesar used cryptography to encode political directives. Steganography (commonly referred to as stego), the art of hidden writing, has also been used for generations. But the intersection of these schemes with the pervasive use of the Internet, high-speed computer and transmission technology, and our current world political climate makes this a unique moment in history for covert communication.

The word steganography comes from the Greek stegauw (covered writing) [3]. Steganography is an ancient art of embedding private messages in seemingly innocuous messages in such a way that prevents the detection of the secret messages by a third party. In other words, steganography means establishing covert channels. A covert channel is a secret communication channel used for transmitting information. As shown in Figure 1.1, two general directions can be distinguished within steganography: protection against detection and protection against removal. Protection against detection is achieved using schemes that do not modify in a visible way the original unmarked object; the modifications are not visible by the humans or by the computers. Protection against removal supposes that the scheme should be robust to common attacks; it is impossible to remove the hidden data without degrading the object’s quality and rendering it useless.

Figure 1.1. Directions within Steganography.

2. TEXT STEGANOGRAPHY
Text steganography [2,3,8], which is what this paper specifically deals with, uses text as the medium in which to hide information. It is the most difficult kind of steganography; this is due largely to the relative lack of redundant information in a text file as compared with a picture or a sound file. The structure of text documents is identical with what we observe, while in other types of documents such as in picture, the structure of document is different from what we observe. Therefore, in such documents, we can hide information by
introducing changes in the structure of the document without making a notable change in the concerned output. Contrary to other media such as pictures, sounds and video clips, using text documents has been common since very old times. Even after invention of printing machine, most of the books and documents have contained only texts. This has extended until today and still, using text is preferred over other media, because the texts occupy lesser memory, communicate more information and need less cost for printing as well as some other advantages.

Text steganography is broadly classified into the two categories; Linguistic Steganography[8], which is further divided into semantic and syntactic method. Another category is Format based steganography which is further divided into following categories, line-shift encoding, word-shift encoding, open-space encoding and feature encoding as described in the figure 2.1.

3. Format Based Steganography

Format-based methods[8] use the physical formatting of text as a space in which to hide information. Format-based methods generally modify existing text in order to hide the steganographic text. Insertion of spaces or non-displayed characters, deliberate misspellings distributed throughout the text, and resizing of fonts are some of the many format-based methods used in text steganography. Some of these methods, such as deliberate misspellings and spaceinsertion, might fool human readers who ignore occasional misspellings, but can often be easily detected by a computer. On the other hand, a computer might not recognize font resizing as a problem, particularly if it is only concentrating on text contents within a document for which text-resizing might be expected (reports with figures, etc.); however, a human might detect strange font sizes almost immediately. Additionally, if the original plaintext is available, comparing this plaintext with the suspected steganographic text would make manipulated parts of the text quite visible.

3.1 The Line-Shift Coding

This technique alters a document by vertically shifting the position of the locations of text lines and may be applied to both the page image and the file format [1,3]. The codeword reassigned for a certain document specifies the text lines that will be moved in that document. We may have a “0” for a line shifted up and a “1” for a line shifted down. But also we may have a “-1” for a line shifted up, a “0” for an unmoved line and a “+1” for a line shifted down. This technique uses the differential encoding technique for achieving performance and robustness. The length of each codeword that can be hidden is reduced, comparing to the technique that shifted every line, but the number can still be large. For example, having a page with 40 lines, that is $220 = 1,048,576$ distinct codewords per page. The encoder has to shift the lines up or down, according to the mark wanting to embed into the file. The decoder measures the distance between each pair of two neighboring lines. This can be done using two different techniques: either the decoder measure the distance between the baselines of adjacent lines or the decoder measures the distance between the centroids of two adjacent lines. A baseline is a logical line on which the characters of a line sit; a centroid is the center of mass of a certain text line. Supposing that the text lines $i$-$l$ and $i+l$ are not shifted and the line $i$ is shifted either up or down. In an unaltered document file the distance between the baselines of two adjacent lines is constant. Now let $si$-$l$ and $si$ be the distances between baselines $i$-$l$ and $i$ and between baselines $i$ and $i+l$, respectively. Unlike baseline spacing, centroid spacing may not be necessarily uniformly spaced. In methods that measure the distance between centroids the decision is based on the difference between centroid spacing in the original document and in the altered document. To calculate the position of centroids, we can use the following formula:

$$c_i = \frac{\sum_{j=1}^{N} f_i n(j)}{\sum_{j=1}^{N} n(j)}$$

where $i=1..N$, is the current line, $N$ is the number of lines on the page, $ti$ and $bi$ are top and bottom limits of the line $i$, $n$ is a function that counts how many pixels are ON $(f(k,j)=1, k=0..W$, see (1)) The next step is to calculate the distance between the centroids of the lines $i$-$l$ and $i$, and $l$ and $i$-$l$, let it be $si$-$l$ and $si$. Figure 3.1 shows a fragment of a document encoded using line shifting coding. In the second part, the second line was moved with 0.1 mm.

![Figure 2.1 Types of Text Steganography.](image)

![Figure 3.1 Line Shift encoding.](image)
3.2 The WordShift Coding

This scheme [1,5,12] alters the document by horizontally shifting the locations of words within text lines to embed the unique mark. The space between adjacent words must be different in order to apply this technique. Variable word spacing is commonly used to distribute white space when justifying a document (as is this paper). Because of the variable spacing, the decoder needs the original document or a specification about word spacing in the original document. The encoder first determines if a line has sufficient number of words to encode; short lines are no encoded. On each encodable text line found is applied the differential encoding technique for this scheme. The second, fourth, sixth, etc. word from the left margin is displaced. The first and the last word on each line are unshifted to maintain the column justification. After the process of shifting words is finished, the document is distributed. The decoder needs information about the original document. This is not a drawback, knowing the fact that in general the authors are tracing their documents and they own a copy of the original document. The information needed is the position of the beginning of each word or the position of centroids for each word. The centroids’ positions are calculated using a similar formula, calculated for line shift encoding. Suppose that the \( i \) th word was shifted and the original centroids’ positions are \( c_i, c_{i-1} \) and the modified centroids’ positions are \( c_i', c_{i-1}' \) then the algorithm calculates:

\[
\begin{align*}
   d_i &= c_i - c_{i-1} \\
   d_i' &= c_{i-1}' - c_i
\end{align*}
\]

and finally will decide if the word was shifted left or right accordingly to the next statement:

- If \( d_i' - d_i > d_{i-1}' - d_{i-1} \) then the word \( i \) was shifted left
- If \( d_i' - d_i < d_{i-1}' - d_{i-1} \) then the word \( i \) was shifted right.

![Figure 3.2.1 Word Shifting Example.](image)

3.3 Feature Coding

According to feature coding scheme, [2,3] the image is examined for chosen text features and those features are altered, depending on the mark inserted. Such features may be the vertical lines of the letters \( b, d, h, k, \) etc. The length of those lines may be modified in a way that is imperceptible to the ordinary readers. The character heights within a given font may also be changed. There are also techniques that change the words themselves substituting them with synonyms. Usually there are two pairs of synonyms and using one or the other synonym is equivalent with embedding a “0” or a “1”. The two parties involved must share the synonymous pairs. The difference between those two techniques is that the first one can be used for embedding copyright information, but the second one only hides information, being adequate in the prisoners’ problem. In the former all documents will have the same content, but some characters will be modified, in the latter two documents having different marks embedded will be different. This feature coding can further be explained with the help of few examples.

3.3.1 Evaluation of Steganography for Urdu/Arabic Text

This method proposed in [9] used for Arabic/Urdu watermarking. One of the characteristics of Arabic language is the use of Araabs i.e. (Fatah, Kasra, and Damma). Where Fatha is slash like symbol and is written over the character, whereas Kasra is also a slash like symbol but is used below the character and Damma is number nine like symbol which is also placed over the character. These araabs are applicable on every single character of the Arabic language. In this paper [9] they have used fatha in reversed order to represent secret characters in the text. Fatha in reverse order is known as reverse fatha. For applying this method on any Urdu text first of all make a secret message which is probably of one line hardly 5 words containing 10-15 characters. Now let us select an article of 4 to 5 paragraph or as the number of paragraphs is increased the security is also increased. Then we put araabs on the complete text or take an article having araabs already. After this let read the secret message character by character and match it in the article, and we have to put reverse fatha where the secret characters exist sequentially try to use the reverse fatha not on the same line but on different lines. Now the article to the reader is ready to be sent in the form of letter.

![Figure 3.3.1 Example of Reverse Fatha.](image)

When the receiver gets the letter the reader will have to read the article carefully and tries to find the reverse fatha in the letter and extract the characters exact below the reverse fatha and collect them. In the end when letter is completely read then concatenate the collected characters. Finally when the concatenated characters are read it represents the meaningful message.

3.3.2 A new approach to Persian/Arabic text steganography

This method [10] of text steganography in Persian and Arabic texts, one of the characteristics of Persian language is abundance of points in its letter. In Persian language 18 letters out of 32 alphabet letters have points. From these 18, 3 letters have 2 points each, 5 letters have 3 points each and the other 10 letters have 1 point each. In this method the concerned information is first of all compressed. Then, we look for the first pointed letter in the given text. By finding this character, we go to the compressed information and read the first bit of information which has one the values of zero or one. If the
value of the bit were zero, the concerned character remains unchanged. If the value of the bit were one, we shift the point on the concerned character a little upward.

Figure 3.3.2 Points in Persian Language.

This procedure is repeated for the next pointed characters in the text and the next bits of information. Thus, the entire information is hidden. In order to divert the attention of readers, after hiding all information, the points of the remaining characters are also changed randomly. Of course, before doing this, the size of hidden information is also hidden in the beginning of the text. For the characters with two or three points, all points shift, because shifting one point among the points of a character raises attentions. While extracting information from the text, the program starts identifying the quantity of hidden bit in the character based on the place of points on the character. By placing all the extracted bits side by side, the compressed information is obtained. Now, this compressed piece of information is uncompressed and the original data is recovered.

3.4 Open space methods

There are two reasons why the manipulation of white space [2] in particular yields useful results. First, changing the number of trailing spaces has little chance of changing the meaning of a phrase or sentence. Second, a casual reader is unlikely to take notice of slight modifications to white space. We describe three methods of using white space to encode data [2,3]. The methods exploit inter-sentence spacing, end-of-line spaces, and inter word spacing in justified text.

3.4.1 Inter-sentence spacing

The first method encodes a binary message into a text by placing either one or two spaces after each terminating character, e.g., a period for English prose, a semicolon for C-code, etc. A single space encodes a “0,” while two spaces encode a “1.” This method has a number of inherent problems. It is inefficient, requiring a great deal of text to encode a very few bits. (One bit per sentence equates to a data rate of approximately one bit per 160 bytes assuming sentences are on average two 80-character lines of text.) Its ability to encode depends on the structure of the text. (Some text, such as free-verse poetry, lacks consistent or well-defined termination characters.) Many word processors automatically set the number of spaces after periods to one or two characters. Finally, inconsistent use of white space is not transparent.

3.4.2 End of line spacing

A second method of exploiting white space to encode data is to insert spaces at the end of lines. The data are encoded allowing for a predetermined number of spaces at the end of each line. Two spaces encode one bit per line, four encode two, eight encode three, etc., dramatically increasing the amount of information we can encode over the previous method. In Figure 3.4.2.1, the text has been selectively justified, and has then had spaces added to the end of lines to encode more data. Rules have been added to reveal the white space at the end of lines. Additional advantages of this method are that it can be done with any text, and it will go unnoticed by readers, since this additional white space is peripheral to the text. As with the previous method, some programs, e.g., “sendmail,” may inadvertently remove the extra space characters. A problem unique to this method is that the hidden data cannot be retrieved from hard copy.

3.4.3 Inter word spacing

A third method of using white space to encode data involves right-justification of text. Data are encoded by controlling where the extra spaces are placed. One space between words is interpreted as a “0.” Two spaces are interpreted as a “1.” This method results in several bits encoded on each line (see Figure 3.4.2.2). Because of constraints upon justification, not every inter-word space can be used as data. In order to determine which of the inter-word spaces represent hidden data bits and which are part of the original text, we have employed a Manchester-like encoding method. Manchester encoding groups bits in sets of two, interpreting “01” as a “1” and “10” as a “0.” The bit strings “00” and “11” are null. For example, the encoded message “1001011011” is reduced to “001,” while “110011” is a null string.

4. Linguistic Steganography

Computational power is increasingly able to analyze more and more complex linguistic structure [8]. Linguistic steganography specifically considers the linguistic properties of generated and modified text, and in many cases, uses linguistic structure as the space in which messages are hidden. This section describes
existing methods of linguistic steganography which is further divided into syntactic method and semantic method [14,15].

4.1 Syntactic methods
That white space is considered arbitrary is both its strength and its weakness where data hiding is concerned. While the reader may not notice its manipulation, a word processor may inadvertently change the number of spaces, destroying the hidden data [8]. Robustness, in light of document reformatting.

<table>
<thead>
<tr>
<th>American Spelling</th>
<th>British Spelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Favorite</td>
<td>Favourite</td>
</tr>
<tr>
<td>Criticize</td>
<td>Criticise</td>
</tr>
<tr>
<td>Fulfill</td>
<td>Fulfil</td>
</tr>
<tr>
<td>Center</td>
<td>Centre</td>
</tr>
<tr>
<td>Dialog</td>
<td>Dialogue</td>
</tr>
<tr>
<td>Medieval</td>
<td>Mediaeval</td>
</tr>
<tr>
<td>Check</td>
<td>Cheque</td>
</tr>
<tr>
<td>Defense</td>
<td>Defence</td>
</tr>
<tr>
<td>Tire</td>
<td>Tyre</td>
</tr>
</tbody>
</table>

is one reason to look for other methods of data hiding in text. In addition, the use of syntactic and semantic methods generally does not interfere with the open space methods. These methods can be applied in parallel. There are many circumstances where punctuation is ambiguous or when mispunctuation has low impact on the meaning of the text. For example, the phrases “bread, butter, and milk” and “bread, butter and milk” are both considered correct usage of commas in a list. We can exploit the fact that the choice of form is arbitrary. Alternation between forms can represent binary data, e.g., anytime the first phrase structure (characterized by a comma appearing before the “and”) occurs, a “1” is inferred, and anytime the second phrase structure is found, a “0” is inferred. Other examples include the controlled use of contractions and abbreviations. While written English affords numerous cases for the application of syntactic data hiding, these situations occur infrequently in typical prose. The expected data rate of these methods is on the order of only several bits per kilobyte of text. Although many of the rules of punctuation are ambiguous or redundant, inconsistent use of punctuation is noticeable to even casual readers. Finally, there are cases where changing the punctuation will impact the clarity, or even meaning, of the text considerably. This method should be used with caution. Syntactic methods include changing the diction and structure of text without significantly altering meaning or tone. For example, the sentence “Before the night is over, I will have finished” could be stated “I will have finished before the night is over.” These methods are more transparent than the punctuation methods, but the opportunity to exploit them is limited.

4.2 Semantic methods.
A final category of data hiding in text involves changing the words themselves. Semantic methods [6,8,14,15] are similar to the syntactic method. Rather than encoding binary data by exploiting ambiguity of form, these methods assign two synonyms primary or secondary value. For example, the word “big” could be considered primary and “large” secondary. Whether a word has primary or secondary value bears no relevance to how often it will be used, but, when decoding, primary words will be read as ones, secondary words as zeros. Word webs such as WordNet can be used to automatically generate synonym tables. Where there are many synonyms, more than one bit can be encoded per substitution. (The choice between “propensity,” “predilection,” “pentrance,” and “prclivity” represents two bits of data.) Problems occur when the nuances of meaning interfere with the desire to encode data. For example, there is a problem with choice of the synonym pair “cool” and “chilly.” Calling someone “cool” has very different connotations than calling them “chilly.” The sentence “The students in line for registration are spaced-out” is also ambiguous. One of the examples of semantic methods is text steganography by changing word spellings [14].

4.2.1 Text Steganography by Changing Words Spelling
Recent publication by author paper [14] presents a new text Steganography method for hiding data in English texts. This method is based on substituting US and UK spellings of words. In English some words have different spelling in UK and US. For example “dialog” has different terms in UK (dialogue) and US (dialog). So we can hide data in the text by substituting these words. Table I shows a number of such words which have different spelling in UK and US. This method is composed of two parts one is hiding program which is responsible for hiding data in text. Another is extractor program which extracts data from the text containing hidden data. At first a list is prepared containing the words which have different UK and US spelling. The hiding method looks for existing words in the list in the text. Furthermore this method converts the concerned data to an arrangement of 0 and 1 bits. The method will place US word in sentence for hiding of the bit 0 and will place the UK word in the sentence in order to hide the bit 1. This way the data will be hidden in the concerned text. Of course the size of data is hidden in the text in order that the extractor method can work correctly. The extractor method will extract the data from the text. This method identifies the type of words in text by using the list of words having different UK and US spelling and saves the quantity of 0 or 1 in an arrangement according to the fact whether it is a US word or UK word. Now the hidden data will be extracted through conversion of this arrangement from the bits 0 and 1 to its original format. At the end the extracted data will be saved on the user's computer. This method has little capacity to hide data in the text. However this is related to the body of text and its size, but in overall its capacity is very low.

5. CONCLUSION
As steganography becomes more widely used in computing there are issues that need to be resolved. A wide variety of different techniques are discussed in present paper with their advantages and disadvantages. Many of currently used techniques are not robust enough to prevent detection and
removal of embedded data. The use of benchmarking to evaluate techniques should become more common and more standard definition of robustness is required to help overcome this problem.

6. FUTURE SCOPE

Data hidden in text has a variety of applications, including copyright verification, authentication, and notation. Making copyright information inseparable from the text is one way for publishers to protect their products in an era of increasing electronic distribution. Annotation can be used for tamper protection. For example, if a cryptographic hash of the paper is encoded into the paper, it is a simple matter to determine whether or not the file has been changed. Verification is among the tasks that could easily be performed by a server, which in this case would return the judgment "authentic" or "unauthentic" as appropriate. One of the possible uses of text-based steganography is reconstruction of printed document. This is done because if somehow the document is torn out then the important information will be lost. Extracting the information from the torn part of the document and recreate the document can help in regaining the lost information. Other uses of data hiding in text involve embedding instructions for an autonomous program in a text. For example, a mail server can be programmed to check for hidden messages when transmitting an electronic message. The message is rejected or approved depending on whether or not any hidden data are found. In this way a company running its own mail server can keep confidential documents from being inadvertently exported.

REFERENCES

[12] Ding Huang and Hong Yan, “Interword Distance Changes Represented by Sine Waves for Watermarking Text Images”, School of Electrical and Information Engineering, University of Sydney, NSW 2006.