Information Security Using Blend of Steganography and Cryptography

Rajkumar Yadav

U.I.E.T, Maharshi Dayanand University, Rohtak-124001, Haryana, India
rajyadav76@rediffmail.com

Abstract. In this paper, information security is achieved using the combination of both cryptography and steganography techniques. Firstly preprocessing of data is done by using data dictionary module and then encryption and compression of data before sending it to receiver using gray level modification (GLM) technique [1]. Matrix and coding technique has been used for encryption of the data. The work in this paper shows an improvement in the embedding capacity of the existing steganography techniques.

Keywords: Cryptography, Steganography, GLM, DM

1. Introduction

Those who seek the ultimate in private communication can combine encryption and steganography. Encrypted data is more difficult to differentiate from naturally occurring phenomena than plain text is in the carrier medium. There are several tools by which we can encrypt data before hiding it in the chosen medium [3]. In some situations, sending an encrypted message will across suspicion while an invisible message will not do so. Both methods can be combined to produce better protection of the message. In case, when the steganography fails and the message can be detected, it is still of no use as it is encrypted using cryptography techniques.

In this paper, a combination of cryptography and steganography has been used to enhance embedding capacity of a steganographic channel by preprocessing the secret data and applying encryption technique over it to make it more robust against Steganalysis. Steganography [4] is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. On the other hand cryptography [6] is an effective way of protecting sensitive information. Although the ultimate goal of cryptography, and the mechanisms that make it up, is to encrypt information from unauthorized individuals. So, cryptography encrypts the message and converts it into cyphertext while Steganography on the other hand, hide the message so there is no knowledge of the existence of the message. The advantage of steganography over cryptography is that stego image does not attract attention while plainly visible encrypted messages no matter how unbreakable will arouse suspicion. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties. [7]

The proposed work technique is limited to textual data only. Here the concept of Scrambled Letters, Dictionary Module [2] has been used. All these concepts when applied together give a phenomenal embedding capacity. Typically, the message is embedded within another object known as a cover work, by tweaking its properties [5].

Though it can reduce the size of data very well but it is not secure to staganalysis. If intruder is somehow able to get this text he can have idea of text and can retrieve it using some searching algorithms. So what this paper discussed is by improving this technique by applying encryption technique and then applying one compression technique so that an improved technique can send more text as compared to other traditional techniques and which is more robust and more secure against staganalysis.

Theoretical results show that it can achieve at least 15-25% increase in embedding capacity with increased data security. This technique is generic and can be applied to any form of textual data without any form of graphics like images or graphs. Once the data is preprocessed by using this technique it can be easily embedded in any steganographic cover medium by using any steganographic algorithm.

In this 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. Potdar et al preprocess
the secret data before even compressing it. He introduce this step to reduce the size of data set which would be compressed and finally embedded. To preprocess the data he introduces two techniques. One of his techniques is generic and can be applied to any form of textual data.

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 it mean processing each word, it 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 following example is taken. Suppose we have a word 'Engineering'.

We keep the first and the last letter intact, while we delete the alternate letters within, i.e. instead of 'Engineering' we can have 'E g n e i g'. As a result of this step we reduce the size of the data. We would only transmit 'E g n e i g' instead of the complete word 'Engineering'.

![Dictionary Module](image)

words to be transmit.

The words to be transmit according to DM.

We take another example suppose we have a word 'Information', instead of transmitting whole world 'Information' we would transmit 'I f r a i n'.

![Dictionary Module](image)

The original words to be transmit.

The words to be transmit according to DM.

This preprocessed data can easily be transmitted to other end but the real problem arises at receiver’s end when we will get this preprocessed text 'E g n e i g' or 'I f r a i n'. How we will understand it. To recover the word properly we need some black box, which takes in the processed word [E g n e i g] and returns the original word [Engineering]. This black box we term the 'Dictionary Module' (DM).

The rest of the paper is organized as follows:
Section 2 describes the concept of Dictionary Module. Section 3 gives the algorithms steps. Section 4 and section 5 shows the flowchart process of sending the data and receiving of data using our work. In Section 6, some experimental results with the help of example is shown. Section 7 provides the conclusion of our work and also gives some attention towards future work.

2. Dictionary Module

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:
1. Alphabetical Order: Firstly all the words are stored in an alphabetical order.
2. Ending Letter: Secondly the words are sorted according to the letter with which they end but maintaining the alphabetical order. E.g. [Steganographic, Steganography]
3. 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 'Engineering' from 'E g n e i g' using this module.

A graphical representation of retrieving original word from Dictionary Module is shown in Error! Reference source not found.

Now a technical detail of the DM has been discussed. Basically DM is a search algorithm which takes four parameters:
1. Length of the original word: This can be deduced form the processed word.
2. The Starting and the Ending letter of the processed word: This is the same as in the original word, e.g. E and G.
3. List of letters from the processed word: E.g. In case of 'Ifrain', the list of letters would be [f,r,a,i]
4. Location of these letters: E.g. ‘f’ is 3rd, ‘r’ is 5th is 7th, ‘i’ is 9th 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.

The concept of DM was inspired by the article posted on the Internet 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.

Using this DM all the words of textual data can be processed. Then these processed words to be transmitted to sender instead of original words. At the receiving end original words can be retrieved from these processed words.

<table>
<thead>
<tr>
<th>Original Word</th>
<th>Processed Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Ifrain</td>
</tr>
<tr>
<td>Technology</td>
<td>Tcnlgy</td>
</tr>
<tr>
<td>Engineering</td>
<td>Egneig</td>
</tr>
<tr>
<td>Steganography</td>
<td>Saeorpy</td>
</tr>
<tr>
<td>Doctor</td>
<td>Dcor</td>
</tr>
<tr>
<td>Teacher</td>
<td>Tahr</td>
</tr>
<tr>
<td>Student</td>
<td>Suet</td>
</tr>
<tr>
<td>The</td>
<td>Te</td>
</tr>
</tbody>
</table>
3. **Algorithms Steps**

3.1 **DATA PREPROCESSING 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.

3.2 **DATA ENCRYPTION STEP**

1. Input :Processed Data, **Output:** Encrypted Data
2. We will use the Matrix Encoding Technique here
3. In this first we will assign values of to alphabets as a=1, b=2 ,c=3 ……… , x=24 , y=25 and z=26.
4. Write the data using 2*2 matrix.
5. Then we will replace each alphabet of textual data by its corresponding assigned numbers.
6. Now matrix is encoded by multiplying it to the coding matrix ( key ) which is known only be the sender and the receiver.

3.3 **DATA COMPRESSION STEP**

**Input:** Encrypted Data, **Output:** Compressed Data

1. We will apply Huffman Coding algorithm.
2. First, list all the letters/symbols used, including the "space" character, along with the frequency with which they occur in the message.
3. Consider each of these character/frequency pairs to be nodes, which are actually leaf nodes.
4. Pick the two nodes with the lowest frequency, and if there is a tie, pick randomly amongst those with equal frequencies.
5. Make a new node out of these two, and make the two nodes its children. This new node is assigned the sum of the frequencies of its children.
6. Continue the process of combining the two nodes of lowest frequency until only one node, the root, remains.
7. To create the codes, start at the root and proceed by assigning 0 to the left branch and 1 to the right branch. Repeat this for all branches in the tree.
3.4 DATA HIDING USING GLM

1. Select pixels according to an arbitrary function g (x, y).

2. Modify the gray level values of the selected pixels to make them even by adding one. These even gray levels will represent 0 in a bit stream.

3. To represent 1, modify the appropriate pixel by decrementing its gray level value by one.

4. Thus, we can represent both 1s and 0s using pixels, which satisfy the condition of being odd or even.

3.5 DATA RETRIEVAL USING GLM

1. Now there is a modified image and the secret function (which is used to find pixel location). Using this function, those pixels that have been used to embed data are identified.

2. The receiver knows the algorithm logic that an even value of gray level represents 0 while an odd value represents 1. Knowing this, the receiver can acquire the hidden data.

3. The receiver first picks up the selected pixels and map them to the respective binary data.

4. Odd number is mapped to one while even number is mapped to zero

3.6 DATA ENCODING USING HUFFMAN CODING

1. The recipient of the message simply starts at the root of the tree and uses the numbers to arrive at a leaf node.

2. At that point the letter in the leaf node is recorded, and the receiver starts again at the root.

3. Decode all the codes is similar way using Huffman Tree.

3.7 DECODING OF MESSAGE USING MATRIX DECRYPTION

1. First write the coded message in 2*2 matrix.

2. The decoding matrix (key) is the inverse of the coding matrix.

3. Multiply the inverse key matrix by coded message matrix.

3.8 DATA RETRIEVAL STEP USING DM

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.

4. FLOWPROCESS SHOWING SENDING DATA

Flow process of sending data is shown in **Error! Reference source not found.**

![Flow process of sending the data](image-url)

**Fig. 4.2 Flow process of sending the data**
5. Flow Process of Retrieving the Data

Fig. Error! No text of specified style in document..1 Flow process of Data Retrieval
6. Results & Analysis With the Example

Preprocessing of data

Suppose we have to send

“Steganographic techniques are very important for data transmission”

According to ASCII coding we have to send it in standard 8 bit codes. There are total 66 alphabets in this sentence including space therefore it will use 66*8=528 Bits. Now we will calculate how many bits will it consume in our proposed technique. When we apply preprocessing of bits according to Potdar et al the given sentence will become

“S e a o r p i c t c n q e s a e v r y i p r a t f r d t a t a s i s o n”

Now we will send this preprocessed data for Encryption.

Data Encryption using Matrix Encoding.

First assign values to alphabets

<table>
<thead>
<tr>
<th>a=1</th>
<th>b=2</th>
<th>c=3</th>
<th>d=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>e=5</td>
<td>f=6</td>
<td>g=7</td>
<td>h=8</td>
</tr>
<tr>
<td>i=9</td>
<td>j=10</td>
<td>k=11</td>
<td>l=12</td>
</tr>
<tr>
<td>m=13</td>
<td>n=14</td>
<td>o=15</td>
<td>p=16</td>
</tr>
<tr>
<td>q=17</td>
<td>r=18</td>
<td>s=19</td>
<td>t=20</td>
</tr>
<tr>
<td>u=21</td>
<td>v=22</td>
<td>w=23</td>
<td>x=24</td>
</tr>
<tr>
<td>y=25</td>
<td>z=26</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now arrange the preprocessed text

“S e a o r p i c t c n q e s a e v r y i p r a t f r d t a t a s i s o n”

In 2*2 matrix
Now replace each alphabet with its corresponding value assigned.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Now we will multiply each matrix with the coding matrix (Key) which is known only to the sender and receiver.

Here coding matrix is

Now multiply this coding matrix with all the above 2*2 matrices obtained above.
\[
\begin{align*}
\begin{pmatrix} 20 & 3 \\ 14 & 17 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 29 & 52 \\ 65 & 96 \end{pmatrix} \\
\begin{pmatrix} 5 & 19 \\ 1 & 5 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 62 & 86 \\ 16 & 22 \end{pmatrix} \\
\begin{pmatrix} 22 & 18 \\ 25 & 9 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 76 & 116 \\ 52 & 86 \end{pmatrix} \\
\begin{pmatrix} 16 & 18 \\ 1 & 20 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 70 & 104 \\ 61 & 82 \end{pmatrix} \\
\begin{pmatrix} 6 & 18 \\ 4 & 20 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 60 & 84 \\ 64 & 88 \end{pmatrix} \\
\begin{pmatrix} 1 & 20 \\ 1 & 19 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 61 & 82 \\ 58 & 78 \end{pmatrix} \\
\begin{pmatrix} 9 & 19 \\ 15 & 14 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} &= \begin{pmatrix} 66 & 94 \\ 57 & 86 \end{pmatrix}
\end{align*}
\]
Now data transmitted will be in form

34 58 46 62 66 100 18 30 29 52 65 96 62 86 16 22 76 116 52 86 70 104 61 82 60 84 64 88 61 82 58 78 66
94 57 86

We will apply Huffman coding on this data to construct a Huffman Tree

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Space</td>
<td>36</td>
</tr>
</tbody>
</table>

The Huffman tree generated for the above data is shown by figure 4.4.
Now data obtained after matrix encoding is converted into Huffman code.

<table>
<thead>
<tr>
<th>0</th>
<th>0110</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
</tr>
<tr>
<td>2</td>
<td>1010</td>
</tr>
<tr>
<td>3</td>
<td>10110</td>
</tr>
<tr>
<td>4</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>10110</td>
</tr>
<tr>
<td>8</td>
<td>010</td>
</tr>
<tr>
<td>9</td>
<td>10111</td>
</tr>
<tr>
<td>Space</td>
<td>11</td>
</tr>
</tbody>
</table>

This data to be send according to Huffman code

```
34 58 46 62 66 100 18 30 29 52 65 96 62 86 16 22 76 116 52 68 61 82 58 78 66 94 57 86
```

Here we use 205 bits to send using our improved technique.

So we got (192/528)*100=36.36 % less bits
“We are using Steganographic Technique here”

According to ASCII code we will use 336 bits to send this data.

According to our technique we will have to send 205 bit to send this same amount of data.

![Graph showing comparison between ASCII, POTDAR, and IMPROVED techniques]

Similarly we apply this technique on text:

“Programming Language”

According to ASCII code we will use 160 bits to send this data.

According to our technique we will have to send only 113 bit to send this same amount of data.

![Graph showing comparison between ASCII, POTDAR, and IMPROVED techniques]

Through our above technique the embedding capacity of transmission medium of Steganographic technique has improved. This technique has twin benefits of Encryption and Steganography. Theoretical results show that it has improved the embedding capacity of transmission medium to 15-25% (average).
7. Conclusion and Future Work

In this paper, Steganography and Cryptography has been discussed. It proposed a methodology to improve the embedding capacity of transmission medium of Steganography with the use of combining features of Steganography and Cryptography both. It shows how a data can be send more securely by applying some of Encryption techniques over it before sending it to the receiver using any of Steganographic Technique. There has been improvement in the method proposed by Potdar et al. In preprocessing of data has been applied using Potdar et al and then Matrix encoding has been applied to secure the data. After that Huffman Encoding is has been applied to compress the size of data to be sent. By this technique not only its able to reduce the number of bits to be used to send the data but also make it more robust against steganalysis. Theoretical results show that it has can improved the capacity of transmission medium to 15-25% (average). But the efficiency of this proposed method may vary with the length and content of textual data. Steganography, in its multitude of forms, has been in use literally for thousands of years. It appears to have been utilized primarily and most effectively in time of war or civil strife. With the advancement in technology improvements are done on Steganographic techniques as well as Steganalysis. So it will focus to develop more secure and robust Steganographic techniques which are easy to implement and which are more robust from security point of view.

References

4. www.wikipedia.com