Detecting MP3Stego and Estimating the Hidden Size

Hua SONG, Tie HU, Yafeng HUANG, Miao GUO
Beijing Institute of Electronic Technology Application
P.O.B. 1950, 36A Xin Jian Gong Men Lu Street, Hai Dian District, Beijing 100091
China

Abstract: - This paper analyzes the properties of the audio information-hiding tool -- MP3Stego, and proposes a new detecting method. This method can detect mp3 files produced by MP3Stego correctly by calculating the statistics of part23 length and stuffingBits in mp3 files, and can estimate the size of embedded text accurately by calculating the variance sequence of the block length. The experiments prove that this method is effective.

Key-Words: - Information Hiding, MP3Stego detecting methods

1 Introduction

MP3Stego\(^1\) is a program to hide information in MP3 files, which was developed by the security laboratory of the Cambridge University. Its source code can be gotten on the Internet. Because MP3 is the most popular music file format on the Internet and MP3Stego is very easy to use, MP3Stego has become a welcomed tool to protect secret message. The author of MP3Stego said that MP3Stego was a secure information-hiding tool and could be used for secret communication and intellectual property protection.

There were some researches to realize attacking or detecting MP3Stego. Liang Jinghong et al\(^2\) said that MP3Stego could be attacked by changing the lowest bit of part23 length because the secret information was hidden in the odevy of part23 length and the change of the odevy could destruct the secret information. Andreas Westfeld\(^3\) said that MP3Stego could be detected by analyzing the statistics of part23 length. The inner iteration loop of the MP3 quantization and encoding process can be ended when the part23 length is less than the specified max_length in normal case. But in hiding case the loop will continue until the length’s odevy is equal to the hidden bit. The final part23 length becomes smaller and the next frame’s part23 length becomes larger. So the variance of block length becomes larger.

The variance of block length is different with or without information-hiding. But there are many encoders and each encoder has a different encoding feature. For example the variance distribution produced by the Lame encoder’s normal encoding process is similar to that after MP3Stego's with information-hiding. So MP3Stego can’t be detected from the variance of the block length easily.

This paper introduces and analyzes the MP3Stego’s information-hiding method first, then puts forward a new fast MP3Stego detecting algorithm, and gives a method to estimate the hiding capacity of MP3Stego. At last the experiments of the new method are given.

2 MP3Stego’s Information-hiding method

MP3\(^4\) was the research result of the Fraunhofer-IIS Institute. The MP3 audio encoding process is shown in Figure 1.

![Figure 1 The MP3 Encoder](image)

The important parts that carry out the MP3 encoding are the two loops. The inner loop is a quantization and encoding loop. If the bits produced by quantization and encoding exceed the available max_length bits, the quantization step is increased to get a smaller quantization value. This operation is repeated continuously, until the Huffman coding bits are short enough. The outer loop is a noise control loop. It controls the quantization noise according to the threshold value and adjusts the scale factor.

When the digital audio signal is quantized and encoded, quantization error appears. This error is an indetermination value, for example adopting different psycho-acoustic models results in
different error. And this error can be adjusted in the process of quantization and encoding. The MP3Stego uses the block length after encoding to hide the secret information by adjusting the quantization error. Odd length means 1 and even length means 0. Thus the secret information is concealed in the MP3 bit stream.

MP3Stego was carried out on the base of the 8hz-mp3 encoder. The hiding process could be explained by Figure 2.

```c
static int inner_loop( … )
{
    …
    do {
        quantizerStepSize += 1.0;
        bits=quantize( );
        switch (hiddenBit) {
            case 2:
                embedRule = 0; break;
            case 0: case 1:
                embedRule = ((bits + part2length) % 2) != hiddenBit;
                break;
            default:
                ERROR("inner_loop: unexpected hidden
                     bit.");}
        } while((bits>max_bits) | embedRule);
    return bits;
}
```

Figure 2 The simplified inner iteration loop of MP3Stego

In the inner iteration loop of quantization and encoding, the hiding rule is added to the quantization rules. If the part23 length after being quantified doesn’t obey the hiding rule (odd represents 1 and even represents 0) or quantization rules, the quantization error is adjusted and the encoding process is repeated.

3 A new method to detect MP3Stego

When encoding a MP3 file, the frame length is in byte according to the bit stream format, but at the time of quantization and encoding, the Main Data is a bit stream. So when data is packed to frames, it may produce 1 to 7 padding bits to satisfy the request of integrity byte. Moreover, different frames have different compression rate. To satisfy the request of invariant frames rate, it may produce some padding bits to fill the frame. These additional bits are kept in the stuffingBits or ancillary positions, as shown in Figure 3.

```c
void ResvFrameEnd(L3_side_info_t *l3_side, int mean_bits )
{
    …
    if(stuffingBits) {  …
        #ifdef MP3STEGO //to satisfy the request of odevity
        if (stuffingBits % 2) {
            gi->part2_3_length += stuffingBits - 1;
            stuffingBits = 1;
            l3_side->resvDrain=stuffingBits;
        } else gi->part2_3_length += stuffingBits;
        #else
            gi->part2_3_length += stuffingBits;
        //in normal case this request is not needed
        #endif
    } …
}
```

Figure 4 The simplified bit reservoir codes of MP3Stego

When 8hz-mp3 encodes a MP3 file, 1 to 7 padding bits are appended to the part23’s stuffingBits positions so that the part23 length is adjusted to integrity byte, then other needed padding bits are appended to the ancillary positions. But in the case of MP3Stego encoding, the padding stuffingBits must be even to ensure the hiding rule that odd length of part23 represents 1 and even length of part23 represents 0. Because there is 50% probability that the padding bits are odd number, it will remain 1 bit, which will be appended to the ancillary position by MP3Stego as shown in Figure 4.

Although this change will not interrupt the MP3 decoding process, it doesn’t accord with the normal case. The differences of the normal MP3 encoding and the MP3Stego encoding can be distinguished by the statistics of the part23 length, the stuffingBits length and the ancillary length. Table 1 shows the statistics of the part23 length, the stuffingBits length and the ancillary length modulo 8 when a WAVE file is encoded to a MP3 file.

Table 1 The statistics of part23 length, stuffingBits and ancillary of MP3Stego, 8hz-mp3 and Lame
Obviously MP3Stego doesn’t adjust all part23 length to integrity byte but 50% to integrity byte and other 50% modulo 8 to 7. The distribution of StuffingBits modulo 8 at 1,3,5,7 is 0 and at 2,4,6 is nearly equal, and the ancillary’s distribution is 50% equal to 1 and 50% equal to 0. Mean-while, 8hz-mp3 adjusts the part23 length to integrity byte when coding, StuffingBits’ distribution at 1-7 is nearly equal, and ancillary’s is equal to 0. There is distinct difference between MP3Stego and 8hz-mp3.

There are many kinds of MP3 encoders. For comparing different characteristics of these coders, we need to calculate the statistics of other encoders. Table 1 gives the statistics of the classic MP3 encoder Lame. We can see that Lame encoder doesn’t adjust the part23 length, but it appends the padding bits to ancillary position. So all stuffingBits are equal to 0, and the sum of the part23 length and the ancillary in the same frame modulo 8 is equal to 0.

By comparing these statistics of different encoders, a statistic formula is given to detect MP3Stego.

\[ R = \frac{\text{abs}(\frac{1}{3}[L - \text{abs}(N - X)]) + \text{abs}(\text{abs}(N - X)) + \text{abs}(\text{abs}(N - X))}{\sum_{i=1}^{L} \text{abs}(\text{abs}(N - X))} \]

\( iL \) indicates the statistic number of part23 length modulo 8 equals to \( i \), \( iN \) indicates the statistic number of stuffingBits modulo 8 equals to \( i \). The threshold can be 0.5, when \( R \) is more than 0.5 it means MP3Stego is detected, or when \( R \) is less than 0.5 it means MP3Stego is not detected.

### 4 Analyzing the size of the hidden information

After confirming MP3Stego encoder, the size of the hidden information can be determined by calculating the changes of block length variance, because the variance changes obviously between the mp3 data with or without information-hiding. Let \( x \) denote the block length, and \( \sum x \) denote the summation of \( n \) blocks length, \( \sum x^2 \) denote the summation of square of \( n \) blocks length, the calculating formula of the block length variance is:

\[ V = \frac{1}{n} \sum x^2 - \left( \frac{1}{n} \sum x \right)^2 \]

Let \( n=8 \). After calculating the variance sequence of the block length of the whole MP3 file, as shown in Figure 5, we can distinctly find the secret information hidden in the MP3 file, where the variances have a big value. The x-axis in the figure is the index of the block length variance sequence, and the y-axis is the block length variance.

![Figure 5 The variance sequence of the block length](image)

To calculate the size of the hidden information, we set threshold=5000. Because there are some unexpected value in the sequence, we set “8 sequential variance less than 5000” as a judgment to figure out the end of hidden data and the capacity (bits) is the summary of the blocks. Because MP3Stego adopts the SHA hash function to hide information, only about 60% blocks are used for information-hiding, so

Real capacity of information-hiding=60% * summary of hiding blocks

### 5 Detection result

In order to examine the detection formula obtained in the last sector, 90 MP3 files were downloaded stochastically from the Internet, and 10 MP3 files were added, in which the secret information was hidden by the MP3Stego encoder. Then these 100 MP3 files were detected by our method, the result of which was shown as below.

<table>
<thead>
<tr>
<th>encoder</th>
<th>encoder</th>
<th>encoder</th>
<th>encoder</th>
<th>encoder</th>
<th>encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>%8</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>MP3-Stego</td>
<td>part23</td>
<td>396</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>stuffBits</td>
<td>ancillary</td>
<td>396</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8hz-mp3</td>
<td>part23</td>
<td>792</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>stuffBits</td>
<td>ancillary</td>
<td>792</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lame</td>
<td>part23</td>
<td>106</td>
<td>101</td>
<td>90</td>
<td>110</td>
</tr>
<tr>
<td>stuffBits</td>
<td>ancillary</td>
<td>1585</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 6 The result of the MP3Stego detection

It could be seen clearly that the statistics R of the MP3 files using MP3Stego to hide information were equal to about 1. The statistics of the most MP3 files from the Internet were bigger than 0 slightly, and some MP3 files’ statistics were between 0.1 to 0.3. Through this detecting formula, the MP3 files that use MP3Stego to hide information could be distinguished clearly.

According to the block length variance formula and the method to calculate the hidden information size, the hidden sizes of these 10 MP3 files were estimated. Table 2 shows the comparison between the estimated sizes and the real values.

Table 2 The estimated hided sizes and the real sizes of MP3Stego

<table>
<thead>
<tr>
<th>The size of txt file</th>
<th>1</th>
<th>66</th>
<th>307</th>
<th>560</th>
<th>985</th>
<th>1428</th>
<th>2097</th>
<th>3992</th>
<th>5192</th>
</tr>
</thead>
<tbody>
<tr>
<td>The size after compression</td>
<td>0</td>
<td>24</td>
<td>96</td>
<td>280</td>
<td>384</td>
<td>584</td>
<td>816</td>
<td>1176</td>
<td>2096</td>
</tr>
<tr>
<td>The estimated hiding size</td>
<td>1</td>
<td>29</td>
<td>102</td>
<td>283</td>
<td>389</td>
<td>590</td>
<td>820</td>
<td>1183</td>
<td>2101</td>
</tr>
</tbody>
</table>

From Table 2 you can see that after the MP3Stego encoder was detected, the hidden information size could be estimated accurately by computing the entire MP3 file’s block length variance sequence.

6 Conclusion

This paper introduced MP3Stego first. After the analysis of the properties of the mp3 bit stream data, a new method was found to detect the hidden information. This method could detect mp3 files produced by MP3Stego correctly by calculating the statistics of part23 and stuffingBits in mp3 files, and could estimate the size of the hidden information accurately by calculating the variance sequence of the block length.

References:

[1] [http://www.cl.cam.ac.uk/~fapp2/steganography/mp3stego/](http://www.cl.cam.ac.uk/~fapp2/steganography/mp3stego/)


[5] [http://www.8hz.com](http://www.8hz.com)