ONLINE WEB PAGES STEGANOGRAPHY

Petra Novandi Barus
1) Informatics Engineering, ITB
Jl. Ganesha 10, Bandung, Indonesia
email: me@van-odin.net, website: http://van-odin.net

ABSTRACT

World wide web is a system of interlinked documents running on the internet. This media could help organization to broadcast hidden message to its member within web documents. Hidden message broadcasts using internet media would reduce the suspicions of the adversaries as well as the amount of cost to broadcast messages to rightful persons.

This paper propose few methods to hide message into web documents such as the well known whitespace manipulation and attribute permutation. To enlarge the possible size of the hidden message, this paper also bring the idea to split hidden message into multiple documents by distribute the hidden message with certain rule.

From all the method discussed on this paper, the strength of the whitespace manipulation lies on message encryption while the attribute permutation's lies on the original attribute sequence list. The multiple documents strength lies on the distribution rule.

Keywords: Steganography, online web pages, web documents

1. INTRODUCTION

Since its creation in, the World Wide Web or WWW is becoming a popular media to transfers informations among people. World Wide Web itself is a system of interlinked hypertext documents running on the internet. This technology was invented at 1989 by Tom Berners-Lee, who initiate the invention by creating the first web site, the first web page, and the first web server.

A web page is a part of a website. It is the document sent by web server to client using the internet. The web page is then opened by the client using web browser. The web browser then show the information contained by the web page and also link to another web pages.

These documents do not directly contain the information. In fact, besides the information, the document also contain the information about the information itself, the structure of the information, and also the presentational aspect of the information. These informations contained by a web page are defined using HTML or Hypertext Markup Language. HTML itself, is defined and maintained by W3C, World Wide Web Consortium, a organization lead by Tom Berners-Lee himself, which maintain all standards on the WWW.

Web documents can be used not only to give the information that will be shown by web browser but also a hidden information that can't be detected and shown by the browser. This is where the steganography comes.

Steganography is the art of hiding messages into another message. The original message is called plaintext while the combined message is called stegotext. In this case plaintexts would be called plaindocuments and stegotexts would be called stegodocuments since the messages are hidden on web documents.

As mentioned before, the nature of a web document is that the documents are defined with HTML and then sent to the client. After documents arrive at the client, the browser then parse the HTML and synthesize the information so that it could show the informations. To hide a message, the browser should be tricked in order to ignore the hidden message when it synthesize and parse the documents.
2. METHODS

1 Single Page Message Hiding

1.1 Whitespace Manipulation

Whitespace manipulation is a method to hide messages by adding whitespaces into plaintext. At glance, people could not recognize the differences between plaintext and stegotext. This is because human eyes are not very sensitive to whitespaces differences within a document full of texts.

This method is mostly applied on ASCII documents such as text, source code, scripts, etc.

1.1.1 End Of Line Whitespace

This method is well known implemented in Snow, a program created by Matthew Kwan. Snow is a program and algorithm used to conceal messages in all kind of ASCII text Snow stands for Steganographics Nature Of Whitespace. Besides its message hiding function, this program also provide message compression and message encryption before it hides the message.

The idea of this method is simply to append whitespace, space and tabs, at the end of lines. To show the beginning of the hidden message, a tab will be appended on the first line of the plaintext where it will fit. Then the hidden message will be written each 3 bits, coded for 0 to 7 spaces. Tab character will be needed to separate between those 3 bits.

<table>
<thead>
<tr>
<th>Bits</th>
<th>#Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
</tbody>
</table>

The program in default will limit the number of spaces and tabs in 80 characters each line. Thus each line could contain exact 60 bits. This method however depends on how many lines the plaintext document has. If the hidden message exceed the plaintext document, the program will append more lines to the plaintext.

Further details of the method and extra features could be obtain from its site page.

1.1.2 Inline Tag Whitespace

Another whitespace manipulation which also will not affect the presentation of the web document is by manipulating whitespaces within a HTML tag. Consider a HTML tag

```
<table border = "1px" cellpadding = "10px"
cellspacing = "20px">
```

The HTML tag above consists of 3 attributes: `border`, `cellpadding`, and `cellspacing`. Between each two tag attribute lies a space to separate them. Between the attribute and its value lies a '=' character which could be preceded or succeed by space or not. These spaces could be manipulate in the way they could be appended with new spaces to represent the hidden message, just like Snow.

For example, in HTML tag above, for each space in the plaintext, we could write each 2 bits of the hidden message “0100 1100 1010 0110 11” coded to 4 whitespaces resulting stegotext below

```
<table border = "1px" cellpadding = "10px"
cellspacing = "20px">
```

Thus, if the hidden message written by 2 bits, for each attribute contained in a tag, there could be 6 bits each attribute. Moreover if there is a inline style tag like

```
<table border = "1px" cellpadding = "10px"
cellspacing = "20px" style = "text-align : center; font-size : large; color : red">`

the whitespace manipulation could be done in a very
similar way. However this method could only work on a tag which contains attribute.

### 1.2 Attribute Permutation

Attribute permutation is a way to represent a code with attributes sequences. While whitespace manipulation could make the original document size bigger, attribute permutation would preserve the original document size.

Suppose a tag within the document

```html
<table border = "3px" cellpadding = "10px"
cellspaceing = "20px" rules = "none">
```

Figure 6. HTML table tag

If the tag were rewritten to

```html
<table cellspacing = "20px" cellpadding = "10px" border = "1px" rules = "none">
```

Figure 7. Rewrite Figure 6

Both of the stegodocument presentation and the stegodocument size will remain the same as the plaindocument.

In this way, we could write a certain permutation in such way that the sequence could represent certain code.

For example, the tag above has 4 attributes: cellspacing, border, rules, cellpadding. Those tag will have 4! or 24 kind of sequences. Those sequences could represent at least \( \log_2(24) \) or 4 bits, leaving 8 possibilities out of 24 all possibilities. It is obvious that every integers larger than 2, the logarithm by 2 of the integer would have remainder.

Take tag table's list of attributes from the HTML 4.01 Document Type Definition:

```html
<!ATTTLIST TABLE
summary
width
border
frame
rules
cellspaceing
cellpadding
```

Figure 8. HTML DTD Table Attribute list

The original sequence of the tag is border, rules, cellspacing, cellpadding. An original sequence is a sequence that represents all zero bits. Bits of hidden message could be represented from its permutation of the sequence.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>b r cs cp</td>
</tr>
<tr>
<td>0001</td>
<td>b r cp cs</td>
</tr>
<tr>
<td>0010</td>
<td>b cs r cp</td>
</tr>
<tr>
<td>0011</td>
<td>b cs cp r</td>
</tr>
<tr>
<td>0100</td>
<td>b cp r cs</td>
</tr>
<tr>
<td>0101</td>
<td>b cp cs r</td>
</tr>
<tr>
<td>0110</td>
<td>r b cs cp</td>
</tr>
<tr>
<td>0111</td>
<td>r b cp cs</td>
</tr>
<tr>
<td>1000</td>
<td>r cs b cp</td>
</tr>
<tr>
<td>1001</td>
<td>r cs cp b</td>
</tr>
<tr>
<td>1010</td>
<td>r cp b cs</td>
</tr>
<tr>
<td>1100</td>
<td>r cp cs b</td>
</tr>
<tr>
<td>1101</td>
<td>cp b r cs</td>
</tr>
<tr>
<td>1110</td>
<td>cp b cs r</td>
</tr>
<tr>
<td>1111</td>
<td>cp r b cs</td>
</tr>
<tr>
<td>skip</td>
<td>cp r cs b</td>
</tr>
<tr>
<td>skip</td>
<td>cp cs b r</td>
</tr>
<tr>
<td>skip</td>
<td>cp cs r b</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>skip</td>
<td>cs cp b r</td>
</tr>
</tbody>
</table>

The table above represents all the possible bits representation from the attribute permutation, b for border attribute, r for rules, cs for cellspacing, and cp for cellpadding.

If the bits from the hidden message is “1101”, all that have to do is just rewrite the tag into

```html
<table cellspacing = "20px" border = "1px"
rules = "none" cellpadding = "10px">
```

Figure 9 Permutation example from Figure 6

In conclusion, tag with \( n \) number of attribute could represent \( \log_2(n!) \) number of bits.

<table>
<thead>
<tr>
<th>N attributes</th>
<th>N bits</th>
<th>N attributes</th>
<th>N bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>11</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2. Permutation list

Table 3. Message capacity within \( n \) attributes tag
If the tag have inline style sheet, this permutation method could be applied to the style sheet rules which could enlarge the tag bit capacity.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>14</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>15</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>16</td>
<td>44</td>
</tr>
</tbody>
</table>

If the tag have inline style sheet, this permutation method could be applied to the style sheet rules which could enlarge the tag bit capacity.

Besides its document size preservation, this method could maintain its security by altering the original permutation. In this way, steganalyst will have difficulties extracting the hidden message because the original sequence probabilities are very large. If a tag contains \( n \) attributes, the number of possible original sequence will be \( n! \). For example, the tag in figure 6 could contain different bits if the original sequence were taken from the list from figure 10.

In addition, as mentioned before, not all the possibilities of the permutation could represent bits. Those possibilities are called remainder sequences. This method could confuse steganalysts by showing the remainder possibility to mark “skip”. The message receiver could recognize those remainder sequences by its original sequences list and then skip the those sequences.

The strength of this method lies on the original sequences list. This list, obviously, should be distributed from the message donor to message recipients before sending messages.

2 Multi Pages Message Hiding

To enlarge the size of hidden message that could be sent, there would be a need to enlarge the media capacity sent to the message recipients. However this need could be avoided by splitting the hidden message into several media linked each others.

Web is a system of interlinked documents. These documents are linked by hyperlinks. This hyperlinks navigate user to open documents within a website or from external website. In fact, every time one open a website, it is most likely that she/he would open many pages from the site. This nature of website could be used to split the hidden message into several web pages.

Pages usually link each other and forms a circular graph. To distribute the hidden message into pages, there would be a need to set the documents sequence or the distribution rule. This rule should determine which page will be the first page and then the pages that follow that first page. For example, we could break down the pages graph into a sequence with BFS (breadth-first search) distribution rule starts from a certain page then traverse hyperlinks by their appearance time on the page, shown in figure 12.

After that, the message then distributed to each pages queued depends on each pages' capacity. In other words, the split size would equal each other. The rule can be set from the time the hidden message was set and the message then would remain static in certain
documents. Therefore, the message recipients are ought to know the distribution rule.

Another mechanism is by dynamically put into the document depends on network session. This means when a device or a user connect to the web and request random first page, the page would be rewritten with the first split of the hidden message on it, so on and so forth. This mechanism would give web server extra hard work to record the user session and its remaining message.

Using static message distribution like stated first but set the distribution rule in a way that could confuse steganalysts is more preferable. In addition, user could access any page randomly and put them back together according to the distribution rule, shown by figure 15, and then extract the hidden message. By doing this way, message recipients wouldn't raise any suspicion from eavesdroppers.

### 3. IMPLEMENTATION

The single page message hiding is easy enough to be implemented. To implement both of whitespace manipulation and attributes permutation method, an external program could be used separately from the web server.

```
url rootpage = "http://www.example.com/page1.html";
//set the rootpage
queue<bit> hiddenmessage = open hidden message();
//open the hidden message, it could be either compressed or encrypted
queue<url> pagequeue = create_distribution_queue(rootpage);
//create queue from distribution rule
foreach(url page in pagequeue) { encode(page, &hiddenmessage);
    //encode the hiddenmessage into pages
}
```

The separate application works by storing the hidden message, compressing it or even encrypting it, and then crawling the documents relatively from their file path, not from their URL path. The documents’ file then permanently altered physically by the application. However this only work if the stegodocument pages are fully static, in other words, plain static HTML documents. Nowadays, almost all page scattered on the internet are made from dynamic server side scripting.
language such as PHP, ASP, JSP, or CGI scripts. This dynamic page means a page couldn't remain the same from time to time. This may cause the pages' capacity will differ unpredictable.

But there is a way to overcome this problem. Let say there is a distribution rule that isn't affected by the dynamic page changes. Then we could use similar way like the dynamic distribution mentioned before. An application to implement this should be working on the Common Gateway Interface or as a web server module.

When the message was input for the first time, the application would generate the distribution list and record the message. When a device or user starts a new session, the application create a unique record for the user along the network session identifier and then copies the distribution list and the hidden message. After that, whenever the device or user with the same session requests for a page, the application would check whether the device or user has requested the page before. If the page from the distribution list hasn't been requested before, then the page will be loaded, processed, and then sent back to the user or the device. In other hand, if the page has been requested then it could be either processed again with the same message written in the same page before or sent back directly without alteration. The among two or more same page, page with valid hidden message will be the first page requested.

4. CONCLUSION

Because world wide web could be accessed easily by many people at anytime, web pages could be a good media to send hidden message to many persons.

To hide message in a HTML document, it is a need to manipulate its source without change its presentation within a web browser. Of course, to extract the hidden message there would be necessary to provide an application which requests pages, collects the page contents and finally extracts the hidden message with either decompress or decrypts the message.

The strength of the whitespace manipulation lies on message encryption because whitespaces manipulation are to obvious to sophisticated adversaries. While the attribute permutation's lies on the original attribute sequence list. The multiple documents strength lies on the distribution rule.

It is best to use static distribution rule in multiple documents message hiding with all the documents are plain static HTML documents. The documents would be permanently transformed to stegodocument from the time the message was set to be hidden so that the web server would not spend its resource and time to alter the plaindocuments each time the document is being requested.

If the documents are written in dynamic server side scripting, which could change nondeterministically, there are no other choice rather than dynamically put the message on the run by recording the network

![Figure 17. Hiding message mechanism](image)
session, the requested page history, and the remaining message sent.

REFERENCES


[3] Internet Society (ISOC). *A Brief History of the Internet*  
   http://www.isoc.org/internet/history/brief.shtml

   http://www.w3.org/TR/html4/

[5] W3C *HTML 4.01 Document Type Definition*  
   http://www.w3.org/TR/html401/sgml/dtd.html

[6] The *SNOW Homepage*  

[7] The *Common Gateway Interface*  
   http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

[8] Apache *API notes*  
   http://httpd.apache.org/docs/1.3/misc/API.html