Embedding Secret Data in Html Web Page

Sandipan Dey[1], Hameed Al-Qaheri[2], and Sugata Sanyal[3]

[1] Cogniant Technology Solutions sandipan.dey@gmail.com
[2] Department of Quantitative Methods and Information Systems College of Business Administration Kuwait University alqaheri@cba.edu.kw
[3] School of Technology and Computer Science Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai - 400005, India sanyal@tifr.res.in

Summary. In this paper, we suggest a novel data hiding technique in an Html Web page. Html Tags are case insensitive and hence an alphabet in lowercase and one in uppercase present inside an html tag are interpreted in the same manner by the browser, i.e., change in case in an web page is imperceptible to the browser. We basically exploit this redundancy and use it to embed secret data inside an web page, with no changes visible to the user of the web page, so that he can not even suspect about the data hiding. The embedded data can be recovered by viewing the source of the html page. This technique can easily be extended to embed secret message inside any piece of source-code where the standard interpreter of that language is case-insensitive.

1 Introduction

Some techniques for hiding data in executables are already proposed (e.g., Shin et al [4]). In this paper we introduce a very simple technique to hide secret message bits inside source codes as well. We describe our steganographic technique by hiding inside html source as cover text, but this can be easily extended to any case-insensitive language source codes. Html Tags are basically directives to the browser and they carry information regarding how to structure and display the data on a web page. They are not case sensitive, so tags in either case (or mixed case) are interpreted by the browser in the same manner (e.g., "$< head >$" and "$< HEAD >$" refers to the same thing). Hence, there is a redundancy and we can exploit this redundancy. To embed secret message bits into html, if the cases of the tag alphabets in html cover text are accordingly manipulated, then this tampering of the cover text will be ignored by the browser and hence it will be imperceptible to the user, since there will not be any visible difference in the web page, hence there will not be any suspect for it as well. Also, when the web page is displayed in the browser, only the text contents are displayed, not the tags (those can only be seen when the user does 'view source'). Hence, the secret messages will be kind of hidden to user.

Both redundancy and imperceptibility conditions for data hiding are met, we use these to embed data in html text. If we do not tamper the html text data that is to be displayed by the browser as web page (this html cover text is analogical

to the cover image, when thought in terms of steganographic techniques in images [1, 2, 3]), the user will not even suspect about hidden data in text. We shall only change the case of every character within these Html tags (elements) in accordance with the secret message bits that we want to embed inside the html web page. If we think of the browser interpreter as a function, $f_B : \sum^* \rightarrow \sum^*$ we see that it is non-injective, i.e., not one to one, since $f_B(x) = f_B(y)$ whenever $x \in \{'A' \ldots' Z'\}$, $y \in \{'a' \ldots' z'\}$ and $Uppercase(y) = x$. The extraction process of the embedded message will also be very simple, one needs to just do 'view source' and observe the case-patterns of the text within tags and can readily extract the secret message (and see the unseen), while the others will not know anything.

The length (in bits) of the secret message to be embedded will be upperlimited by the sum of size of text inside html tags (here we don't consider attribute values for data embedding. In case we consider attribute values for data embedding, we need to be more careful, since for some tags we should think of case-sensitivity, e.g. <A HREF="link.html">, since link file name may be casesensitive on some systems, whereas, attributes such as <h2 align="center"> is safe). If less numbers of bits to be embedded, we can embed the information inside Header Tag specifying the length of embedded data (e.g. '<Header 25 >' if the length of secret data to be embedded is 25 bits) that will not be shown in the browser (optionally we can encrypt this integer value with some private key). In order to guarantee robustness of this very simple algorithm one may use some simple encryption on the data to be embedded.

2 The Algorithm for Embedding

The algorithm for embedding the secret message inside the html cover text is very simple and straight-forward. First, we need to separate out the characters from the cover text that will be candidates for embedding, these are the caseinsensitive text characters inside Html tags. Figure 2 shows a very simplified automata for this purpose.

We define the following functions before describing the algorithm:

- $l : \sum^* \rightarrow \sum^*$ is defined by,

$$l(c) = \left\{ \begin{array}{cc} ToLower(c) & c \in \{'A' \ldots' Z'\} \\ c & \text{otherwise} \end{array} \right\} \text{ were } ToLower(c) = c + 32$$

- Similarly, $u : \sum^* \rightarrow \sum^*$ is defined by,

$$l(c) = \left\{ \begin{array}{cc} ToUpper(c) & c \in \{'a' \ldots' z'\} \\ c & \text{otherwise} \end{array} \right\} \text{ were } ToUpper(c) = c - 32$$

Here the ascii value of 'A' is 65 and that of 'a' is 97, with a difference of 32.

It's easy to see that if the domain $\sum^* = \{'a' \ldots' z'\} \cup \{'A' \ldots' Z'\}$, then $l : \{'A' \ldots' Z'\} \rightarrow \{'a' \ldots' z'\}$ and $u : \{'a' \ldots' z'\} \rightarrow \{'A' \ldots' Z'\}$, implies that $l(.) = \overline{u(.)} = \sum^* - l(.)$.

Now, we want to embed secret data bits $b_1, b_2 \ldots b_k$ inside the case-insensitive text inside the Html Tags. If $c_1 c_2 \ldots c_n$ denotes the sequence of characters inside the html tags in cover text (input html). A character c_i is a candidate for hiding a

secret message bit iff it is an alphabet. If we want to hide the j^{th} secret message bit b_j inside the cover text character c_i, the corresponding stego-text will be defined by the following function f_{stego}: $\forall c_i \in \{'a' \dots' z'\} \cup \{'A' \dots' Z'\}$, i.e. if $IsAlphabet(ci)$ is true, $f_{stego}(c_i) = \begin{cases} l(c_i) & b_j = 0 \\ u(c_i) & b_j = 1 \end{cases}$ Hence, we have the following:

$$c_i \in \{'a' \dots' z'\} \cup \{'A' \dots' Z'\} \Rightarrow f_{stego}(c_i) = l(c_i) \cdot \overline{b_j} + u(c_i) \cdot b_j, \forall i \qquad (1)$$

Number of bits (k) of the secret message embedded into the html cover text must also be embedded inside the html (e.g., in Header element). The figure 1 and the algorithm 1 together explain this embedding algorithm.

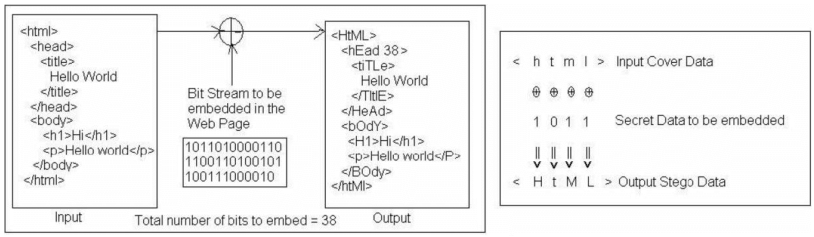

Fig. 1: Illustration of how the data hiding works

3 The Algorithm for Extraction

Algorithm 1 Embedding Algorithm

Search for all the html tags present in the html cover text and extract all the characters $c_1 c_2 \dots c_n$ from inside those tags using the DFA described in the Fig.2. Embed the secret message length k inside html header in the stego text.
$j \leftarrow 0$.
for $c_i \in HTMLTAGS$, $i = 1 \dots n$ **do**
  if $c_i \in \{'a' \dots' z'\} \cup \{'A' \dots' Z'\}$ **then**
    $f_{stego}(c_i) = l(c_i) \cdot \overline{b_j} + u(c_i) \cdot b_j$.
    $j \leftarrow j + 1$.
  else
    $f_{stego}(c_i) = c_i$.
  end if
  if $j == k$ **then**
    break.
  end if
end for

The algorithm for extraction of the secret message bits will be even more simple. Like embedding process, we must first seperate out the candidate text (text within tags) that were chosen for embedding secret message bits. Also, we must extract the number of bits (k) embedded into this page (e.g., from the header element). One has to use 'view source' to find out the stego-text.

Now, we have $d_i = f_{stego}(c_i)$, $\forall i \in \{1, 2, \ldots, n\}$. If $d_i \in \{'a' \ldots' z'\} \cup \{'A' \ldots' Z'\}$ i.e., an alphabet, then only it is a candidate for decoding and to extract b_i from d_i, we use the following logic: $b_i = \begin{cases} 0 \; d_i \in \{'a' \ldots' z'\} \\ 1 \; d_i \in \{'A' \ldots' Z'\} \end{cases}$ Repeat the above algorithm $\forall i < k$, to extract all the hidden bits. The figure 2 and the algorithm 2 together explain this embedding algorithm.

Algorithm 2 Extraction Algorithm

Search for all the html tags present in the html stego text and extract all the characters $d_1 d_2 \ldots d_n$ from inside those tags using the DFA described in the Fig.2.
Extract the secret message length k from inside html header in the stego text.
$j \leftarrow 0$.
for $d_i \in HTMLTAGS$, $i = 1 \ldots n$ **do**
  if $d_i \in \{'a' \ldots' z'\}$ **then**
    $b_j = 0$.
    $j \leftarrow j + 1$.
  else
    if $d_i \in \{'A' \ldots' Z'\}$ **then**
      $b_j = 1$.
      $j \leftarrow j + 1$.
    end if
  end if
  if $j == k$ **then**
    break.
  end if
end for

Figures 3, 4 and 5 show an example of how our method works, while Figure 6 shows the comparison of the histogram of the cover html and stego html in terms of the (ascii) character frequencies. Classical image hiding techniques like LSB data hiding technique always introduce some (visible) distortion [5, 10] in the stego image (that can be reduced using techniques [6, 7, 8, 9]), but our data hiding technique in html is novel in the sense that it introduces no visible distortion in stego text at all.

4 Conclusions

In this paper we presented an algorithm for hiding data in html text. This technique can be extended to any case-insensitive language and data can

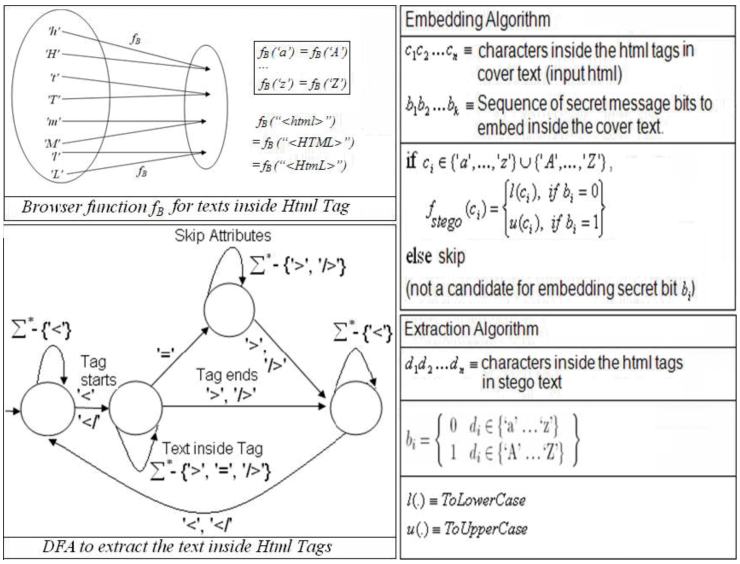

Fig. 2: Basic block-diagram for html data-hiding technique

be embedded in the similar manner, e.g., we can embed secret message bits even in source codes written in languages like basic or pascal or in the case-insensitive sections (e.g. comments) in C like case-sensitive languages. Data hiding methods in images results distorted stego-images, but the html data hiding technique does not create any sort of visible distortion in the stego html text.

References

1. Neil F. Johnson and Sushil Jajodia. Steganography: Seeing the Unseen IEEE Computer, February 1998, pp. 26-34.
2. W. Bender et al., Techniques for Data Hiding, Systems J., Vol. 35, Nos. 3 and 4, 1996, pp. 313-336.
3. B. Pfitzmann, Information Hiding Terminology, Proc. First International Workshop Information Hiding, Lecture Notes in Computer Science No. 1,174, Springer-Verlag, Berlin, 1996, pp. 347-356.
4. DaeMin Shin, Yeog Kim, KeunDuck Byun, SangJin Lee, Data Hiding in Windows Executable Files, Center for Information Security Technologies (CIST), Korea University, Seoul, Republic of Korea
5. C. Kurak, J. McHugh, A Cautionary Note On Image Downgrading, Proc. IEEE Eighth Annual Computer Security Applications Conf., IEEE Press, Piscataway, N.J., 1992, pp. 153-159.
6. F. Battisti, M. Carli, A. Neri, K. Egiaziarian, A Generalized Fibonacci LSB Data Hiding Technique, 3rd International Conference on Computers and De-

```
<html>
<head>
<title> Test Html (Steganography - Seeing the Unseen)</title>
</head>
<body>

<table border="1" cellspacing="5" cellpadding="10">
  <tr>
    <td>  </td>
    <td colspan="2"> <h1> Embedding Secret Message, can you see what it is</h1></td>
  </tr>
  <tr>
    <td rowspan="2" align="left" valign="top">
        <p> Image Processing Techniques<br>
              Histogram equalization<br>
        Bilevel Thresholding<br>
        Optimal Thresholding<br>
        LOG Filter<br>
        DFT  DCT DWT<br>
        Segmentation<br>
        JPEG Compression </p></td>
    <td><h1>The Lena Image - we don't embed secret message inside Lena!!!</h1></td>
    <td align="center" valign="middle"> <img src="images/lena.bmp" alt="Lena" width="256" height="128" border="0">
    </td>
  </tr>
  <tr>
    <td align="right" valign="bottom"> <img src="images/lena.bmp" alt="Lena" width="256" height="128" border="0">
    </td>
    <td> <p>Lena is an image that is used to test many of the classical image processing
              techniques and algorithms including histogram equalization, optimal thresholding
        <b>LSB data hiding</b>
              (LSB data hiding technique changes the LSB s of the image data to hide secret message);
        <b>Steganography</b>
        (Steganography is seeing the unseen)
        <b>Digital Watermarking</b>
            (But here we do not embed inside the image).
                Instead we use case insensitivity of html tags to embed our secret message.</p></td>
  </tr>
  <tr>
    <td colspan="3" align="left" valign="top">Embed secret message inside
        <b>cover text</b> to get
    <b>stego text</b> and be happy since both the stego html and the cover html look
          exactly identical in the browser. No one can even guess that there is some
          secret message already embedded inside it.</td>
  </tr>
  <tr>
    <td colspan="3" align="left" valign="top">Finally, extraction of the secret message is also
      <b>easy</b> (just you need to do)
      <b>view source</b> (and extract the message). Just checking the cases of the html tags reveals
          the embeded message.  Can make this simple technique even more robust by using some
          simple encryption technique in addition</td>
  </tr>
</table>

<p>&gt; <a href="index.htm">Steganography examples</a></p>
</body>
</html>
```

Cover Html Source

Fig. 3: Cover Html source before embedding the secret message

vices for Communication (CODEC- 06) TEA, Institute of Radio Physics and Electronics, University of Calcutta, December 18-20, 2006.

7. Sandipan Dey, Ajith Abraham, Bijoy Bandyopadhyay and Sugata Sanyal, Data Hiding Techniques Using Prime and Natural Numbers, Journal of Digital Information Management, Volume 6, 2008.
8. Sandipan Dey, Ajith Abraham and Sugata Sanyal An LSB Data Hiding Technique Using Natural Numbers, IEEE Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007, Nov 26-28, 2007, Kaohsiung City, Taiwan, IEEE Computer Society press, USA, ISBN 0-7695-2994-1, pp. 473-476, 2007.
9. Sandipan Dey, Ajith Abraham and Sugata Sanyal An LSB Data Hiding Technique Using Prime Numbers, Third International Symposium on Information Assurance and Security, August 29-31, 2007, Manchester, United Kingdom, IEEE Computer Society press, USA, ISBN 0-7695-2876-7, pp. 101-106, 2007.
10. R. Z. Wang, C. F. Lin and I. C. Lin, Image Hiding by LSB substitution and genetic algorithm, Pattern Recognition, Vol. 34, No. 3, pp. 671-683, 2001.

```
<hTml>
<heAD 200>
<tITlE> Test Html (Steganography - Seeing the Unseen)</TITlE>
</HEad>
<bodY>

<TABle BoRDEr="1" cEllSPaCinG="5" cELlpADDiNG="10">
  <tR>
    <td>  </td>
    <TD CoLspan="2"> <H1> Embedding Secret Message, can you see what it is</h1></td>
  </tr>
  <tR>
    <td rowspaN="2" align="left" vAlIgn="top">
        <P> Image Processing Techniques<Br>
              Histogram equalization<BR>
        Bilevel Thresholding<br>
        Optimal Thresholding<br>
        LOG Filter<Br>
        DFT  DCT DWT<BR>
        Segmentation<bR>
        JPEG Compression </P></Td>
    <tD><H1>The Lena Image - we don't embed secret message inside Lena!!!</h1></tD>
    <td aLIgN="center" vaLiGN="middle"> <Img src="images/lena.bmp" ALt="Lena" widTh="256" HEiGHT="128" borDer="0">
    </td>
  </tr>
  <Tr>
    <td AligN="right" ValIgN="bottom"> <iMG SRc="images/lena.bmp" aLt="Lena" width="256" height="128" border="0">
    </td>
    <td> <p>Lena is an image that is used to test many of the classical image processing
              techniques and algorithms including histogram equalization, optimal thresholding
        <b>LSB data hiding</b>
              (LSB data hiding technique changes the LSB s of the image data to hide secret message);
        <b>Steganography</b>
        (Steganography is seeing the unseen)
        <b>Digital Watermarking</b>
            (But here we do not embed inside the image).
              Instead we use case insensitivity of html tags to embed our secret message.</p></td>
  </tr>
  <tr>
    <td colspan="3" align="left" valign="top">Embed secret message inside
        <b>cover text</b> to get
    <b>stego text</b> and be happy since both the stego html and the cover html look
          exactly identical in the browser. No one can even guess that there is some
          secret message already embedded inside it.</td>
  </tr>
  <tr>
    <td colspan="3" align="left" valign="top">Finally, extraction of the secret message is also
      <b>easy</b> (just you need to do)
      <b>view source</b> (and extract the message). Just checking the cases of the html tags reveals
          the embeded message.  Can make this simple technique even more robust by using some
          simple encryption technique in addition</td>
  </tr>
</table>

<p>&gt; <a href="index.htm">Steganography examples</a></p>
</body>
</html>
```

Stego Html Source

	Secret Message embedded	
	"Copyright @ Sandipan Dey"	

Embedded BitStream (length 200)

0100001101101110110000011110010111001001101001011001110110100001110100001000000100000001000000101
0011011000010110111001100100001101001011100000 0000101101110001000000100010001100101011110010000000

Fig. 4: Stego Html source after embedding the secret message

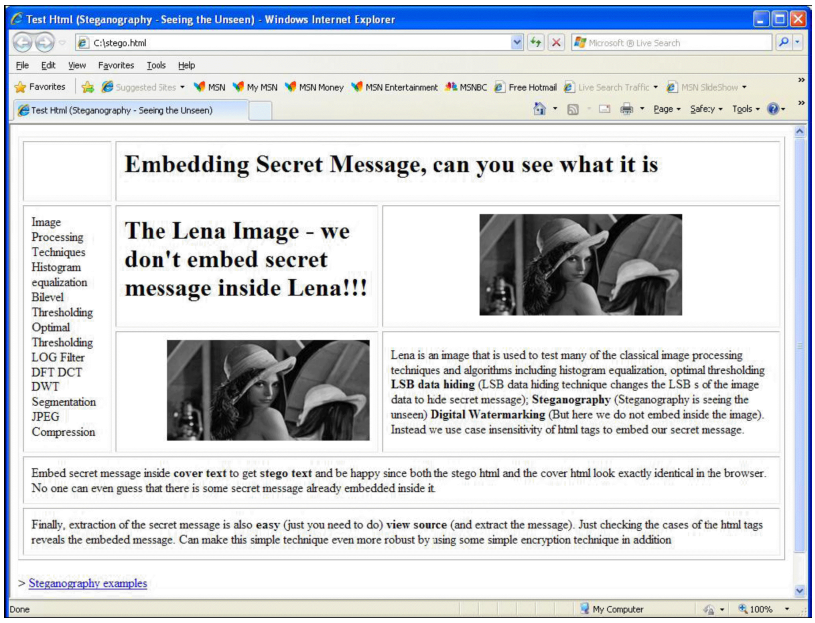

Fig. 5: Cover & Stego Html

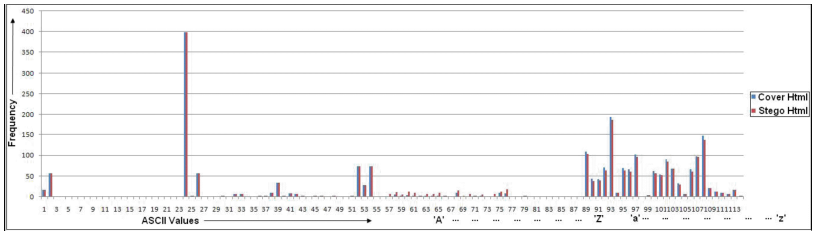

Fig. 6: Cover vs Stego Html Histogram