

ISSN 1425-140X

**IMAGE
PROCESSING
&
COMMUNICATIONS**

Volume 14, Number 2-3, 2009.

An International Journal

Editor - in - Chief

Ryszard S. Choraś

Institute of Telecommunications

UT&LS Bydgoszcz

POLAND

**Published by the Institute of Telecommunications
Bydgoszcz, Poland**

IMAGE PROCESSING & COMMUNICATIONS

An International Journal

Aims and Scope

IMAGE PROCESSING and COMMUNICATIONS publishes original research articles and serves as a forum for stimulating innovative research ideas, state - of - the art methods and tools in all aspects of image processing and communications. Theoretical, experimental and survey articles are all appropriate to the journal.

Specific areas of interest include:

- Image processing: Coding, Analysis and Recognition
- Image manipulation
- Communication of visual data
- Network architecture for real-time video transport
- Video coding algorithms and technologies for ATM/packet network
- Protocols for packet video
- New visual services over ATM/packet network.

Papers are solicited in these topic areas.

Copyright

It is the condition of publication that manuscript submitted to this journal have not been published and will not be simultaneously submitted or published elsewhere. By submitted of manuscript, the authors agree that the copyright for their article is transferred to the publisher if and when the article is accepted for publication. The copyright covers the exclusive rights to reproduce and distribute the article, including reprints, photographic reproduction or any other reproductions of similar nature and translations.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form a by any means, electronic, photocopying or other wise, without permission in writing from the copying holder.

The copyright owner's consent does not extend to copying for general distribution, for promotion, for creating new works, or for resale.

Specific written permission must be obtained from the publisher for copying.

Subscription information

The subscription price for 2008 is 150 Euro / PLZ 350 including carriage charges. IMAGE PROCESSING and COMMUNICATIONS may be ordered directly at the Institute of Telecommunications, 85-791 Bydgoszcz, ul.S. Kaliskiego 7, POLAND.

IMAGE PROCESSING & COMMUNICATIONS

An International Journal

Volume 14, Number 2-3

OPTIMIZED ARCHITECTURES OF CABAC CODEC FOR IA-32-, DSP- AND FPGA-BASED PLATFORMS - D. Karwowski, M. Domański	5
TRACKING OF 3D OBJECTS IN A VISION BASED TRAVEL AID SYSTEM FOR THE BLIND - P. Pelczynski	13
OPTICAL FEATURE CLUSTERING ALGORITHM FOR OBJECT TRACKING IN IMAGE SEQUENCES - P. Pelczynski, J. A. Arriola	23
CONSTRAINED CONTOUR MATCHING IN HAND POSTURE RECOGNITION - A. Wilkowski, W. Kasprzak	31
SIGNATURE RECOGNITION METHOD BY MEANS OF THE WINDOWS TECHNIQUE - P. Porwik, K. Wrobel and R. Doroz	43
A NEURAL NETWORK METHOD OF IMAGE RECONSTRUCTION FROM PROJECTIONS USING GRID-"FRIENDLY" PROJECTION ANGLES - R. Cierniak	51
HIDING INSIDE HTML AND OTHER SOURCE CODES - H. Al-Qaheri, S. Dey, S. Sanyal	59
A METHOD FOR SEMI-AUTOMATED ASSESSMENT OF USER SATISFACTION WHEN USING WWW SERVICES WITH MOBILE TERMINALS - A. Flizikowski, M. Choraś, M. Wachowiak, W. Hołubowicz	69

IMAGE PROCESSING & COMMUNICATIONS

An International Journal

Editor - in Chief

Ryszard S. Choraś

Institute of Telecommunications

UT&LS Bydgoszcz

85-791 Bydgoszcz, ul. S. Kaliskiego 7, POLAND

e-mail: choras@utp.edu.pl

Editorial Board

Gunilla Borgefors

Centre for Image Analysis

Swedish University for Agricultural Sciences

Uppsala, Sweden

Marek Domański

Chair of Multimedia Telecommunications

and Microelectronics

Poznań University of Technology, Poland

Murat Kunt

Signal Processing Laboratory

Swiss Federal Institute of Technology

CH - 1015 Lausanne, Switzerland

D. Dutta Majumder

Electronics & Communication Science Unit

Indian Statistical Institute

Calcuta 700 035, India

Władysław Skarbek

Institute of Radioelectronics

Faculty of Electronics and Computer Technology

Warsaw, Poland

Leonid P. Yaroslavsky

Faculty of Engineering

Dept. of Interdisciplinary Studies

Tel Aviv University, Israel

Christophe Charrier

University of Caen Basse-Normandie

Vision and Image Analysis Group

Saint - Lo, France

Kalman Fazekas

Department of Microwave Telecommunications

Technical University of Budapest

Budapest, Hungary H-1111

Stefano Levialdi

Dipartimento di Scienze dell' Informazione

Universita degli Studi di Roma

00198 Roma, Italy

Ioanis Pitas

Department of Electrical Engineering

Aristotle University of Thessaloniki

54006 Thessaloniki, Greece

Ryszard Tadeusiewicz

Institute of Automation

University of Mining and Metallurgy

Krakow, Poland

Antoni Zabłudowski

Institute of Telecommunications

UTP Bydgoszcz, Poland

HIDING INSIDE HTML AND OTHER SOURCE CODES

HAMEED AL-QAHERI¹ SANDIPAN DEY² SUGATA SANYAL³

¹Department of Quantitative Methods and Information Systems,
College of Business Administration
Kuwait University

alqaaheri@cba.edu.kw

²Cogniant Technology Solutions

sandipan.dey@gmail.com

³School of Technology and Computer Science,
Tata Institute of Fundamental Research,
Homi Bhabha Road, Mumbai - 400005, India

sanyal@tifr.res.in

Abstract. Many steganographic techniques [1] [2] [3] [4] were proposed for hiding secret message inside images, the simplest of them being the LSB data hiding [6] [7] [8] [9] [10], [11]. In this paper, we suggest a novel data hiding technique in an Html Web page [12] and also propose some simple techniques to extend the embedding technique to source codes written in any programming language (both case insensitive like Html, PASCAL and case sensitive languages like C, C++, Java) - an extension to [12]. We basically try to exploit the case-redundancy in case-insensitive language, while we try hiding data with minimal changes in the source code (almost not raising suspicion). 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 first 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. For case-sensitive programming languages we do minimal changes in the source code (e.g., add an extra character in the token identified by the lexical analyzer) without violating the lexical and syntactic

notation for that language) and try to make the change almost imperceptible.

1 Introduction

Steganography is another name of hiding secret data in cover medium, thereby ensuring imperceptibility and exploiting redundancies in representation of the cover medium. For instance, in case of LSB data hiding the property that the cover image visual representation is least affected (almost unaffected) by the change of the LSB of any pixel, is used and this redundancy (or cover image oblivious to change in LSB) is exploited to embed secret data in LSB [11]. Also, some decomposition techniques were proposed to enhance the LSB data hiding technique by increasing the number of bitplanes [6] [7] [8] [9]. 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, as an extension of [12]. We describe our steganographic technique by hiding inside html source as cover text, but this can be extended to any case-insensitive language source codes like Basic, PASCAL or FORTRAN.

2 Hiding Data inside Html

2.1 Exploiting the Case-Insensitivity

As we know, 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 in terms of case-insensitivity and we shall 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, since they will have no effect on the page displayed by the browser. In other words, browser will help us hiding the data, by being indifferent to cases of the html tags, but we shall use those as key places for hiding data.

Since only the portion of cover text inside the html tags will be used (and possibly will undergo a case-conversion) for hiding secret message bits and we are not going to tamper the html text data (outside the tags) that are going 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 have any reason to 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 hide inside the html source.

As described in [12], if we think of the browser interpreter as a function, $f_B : \Sigma^* \rightarrow \Sigma^*$ we see that it is non-injective, i.e., not one to one, since $f_B(x) = f_B(y)$ whenever $x \in \{ 'A' \dots 'Z' \}$, $y \in \{ 'a' \dots 'z' \}$ and $Uppercase(y) = x$. The extraction process of the hidden 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. So, both the embedding and extraction of secret message bits be-

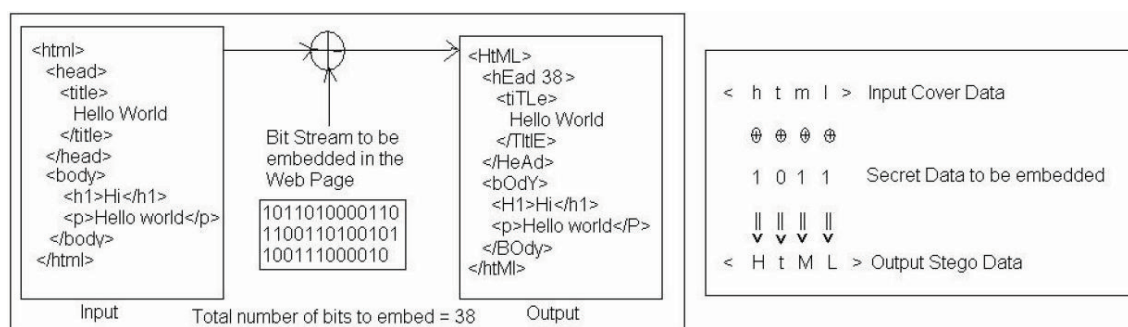


Fig. 1: Illustration of how the Html data hiding works

come very simple.

As it can be guessed, we can not hide arbitrary long data inside a given cover text. More precisely, the length (in bits) of the secret message to be hidden inside the html cover-text will be upper-limited 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. ``, since link file name may be case-sensitive 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.

2.2 The Algorithm for Hiding Data

As described in [12], 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 case-insensitive text characters inside Html tags. Fig. 2 shows a very simplified automata for this purpose.

Also, let us define the following functions before describing the algorithm:

- $l : \Sigma^* \rightarrow \Sigma^*$ as:

$$l(c) = \begin{cases} ToLower(c) & c \in \{ 'A' .. 'Z' \} \\ c & otherwise \end{cases}$$

where $ToLower(c) = c + d$

- $u : \Sigma^* \rightarrow \Sigma^*$ as:

$$u(c) = \begin{cases} ToUpper(c) & c \in \{ 'a' .. 'z' \} \\ c & otherwise \end{cases}$$

where $ToUpper(c) = c - d$

- Here $d = 'A' - 'a'$.

The ascii value of 'A' = 65 and the ascii value of 'a' = 97, with a difference $d = 32$.

It's easy to see that if the domain $\Sigma^* = \{ 'a' .. 'z' \} \cup \{ 'A' .. 'Z' \}$, then

$l : \{ 'A' .. 'Z' \} \rightarrow \{ 'a' .. 'z' \}$ and $u : \{ 'a' .. 'z' \} \rightarrow \{ 'A' .. 'Z' \}$, implies that $l(\cdot) = \overline{u(\cdot)} = \Sigma^* - l(\cdot)$.

Now, proceeding as in [12], we want to embed secret data bits $b_1 b_2 .. b_k$ inside the case-insensitive text inside the Html Tags. If $c_1 c_2 .. c_n$ 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 if 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

$$f_{stego}: \forall c_i \in \{ 'a' .. 'z' \} \cup \{ 'A' .. 'Z' \}, \text{ i.e. if } \text{IsAlphabet}(c_i) \text{ is true,}$$

$$f_{stego}(c_i) = \left\{ \begin{array}{ll} l(c_i) & b_j = 0 \\ u(c_i) & b_j = 1 \end{array} \right\},$$

Hence, we have the following:

$$c_i \in \{ 'a' .. 'z' \} \cup \{ 'A' .. 'Z' \} \Rightarrow f_{stego}(c_i) = l(c_i).\bar{b}_j + u(c_i).b_j, \forall i \quad (1)$$

The 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 Fig. 1, 2 and the algorithm 1 together explain this data hiding algorithm.

Algorithm 1 Algorithm to Hide Data inside Html

- 1: 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.
 - 2: Embed the secret message length k inside html header in the stego-text.
 - 3: $j \leftarrow 0$.
 - 4: **for** $c_i \in HTMLTAGS, i = 1 \dots n$ **do**
 - 5: **if** $c_i \in \{ 'a' \dots 'z' \} \cup \{ 'A' \dots 'Z' \}$ **then**
 - 6: $f_{stego}(c_i) = l(c_i).\bar{b}_j + u(c_i).b_j$.
 - 7: $j \leftarrow j + 1$.
 - 8: **else**
 - 9: $f_{stego}(c_i) = c_i$.
 - 10: **end if**
 - 11: **if** $j == k$ **then**
 - 12: **break**.
 - 13: **end if**
 - 14: **end for**
-

2.3 The Algorithm for Hidden Data Extraction

Again, proceeding as in [12], the algorithm for extraction of the secret message bits will be even simpler. As in the embedding process, we

must first separate out the candidate text (exactly the text within the Html tags) that were chosen in the earlier step for embedding secret message bits. Also, we must extract the number of bits (k) embedded into this page (e.g., from the header element). In order to find out the stego-text, one has to use 'view source'.

Algorithm 2 Hidden Data Extraction Algorithm

- 1: Search for all the html tags present in the html stego-text and extract all the characters $d_1 d_2 \dots d_n$ from inside those tags using the DFA described in the Fig. 2.
 - 2: Extract the secret message length k from inside html header in the stego-text.
 - 3: $j \leftarrow 0$.
 - 4: **for** $d_i \in HTMLTAGS, i = 1 \dots n$ **do**
 - 5: **if** $d_i \in \{ 'a' \dots 'z' \}$ **then**
 - 6: $b_j = 0$.
 - 7: $j \leftarrow j + 1$.
 - 8: **else if** $d_i \in \{ 'A' \dots 'Z' \}$ **then**
 - 9: $b_j = 1$.
 - 10: $j \leftarrow j + 1$.
 - 11: **end if**
 - 12: **if** $j == k$ **then**
 - 13: **break**.
 - 14: **end if**
 - 15: **end for**
-

Now, we have $d_i = f_{stego}(c_i), \forall i \in \{1, 2, \dots, n\}$. If $d_i \in \{ 'a' .. 'z' \} \cup \{ 'A' .. 'Z' \}$ 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:

$$b_i = \left\{ \begin{array}{ll} 0 & d_i \in \{ 'a' \dots 'z' \} \\ 1 & d_i \in \{ 'A' \dots 'Z' \} \end{array} \right\}$$

Repeat the above algorithm $\forall i < k$, to extract all the hidden bits.

2.4 Experimental Results

As in [12], we obtained the following results: The Fig. 3, 4 and 5 (as in [12]) show an example of how our method works, while Fig. 6

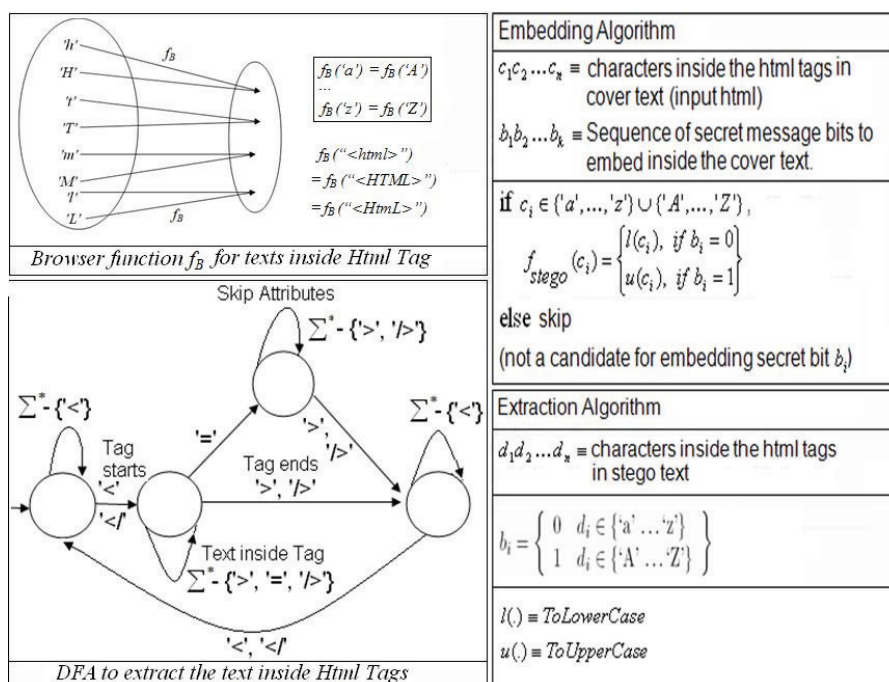


Fig. 2: Basic block-diagram for the Html data-hiding technique

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.

3 Hiding Data inside Other Source Codes

3.1 Case-insensitive Programming Language Sources

In order to hide data in the source codes in languages for which the letter-case is ignored (e.g. PASCAL), we can follow similar approaches as before. But we must ensure the following things:

- While embedding secret messages we must

not change anything in the source code, so that the output of the program (when run) changes.

- Should not embed inside variable values, for instance string literals.
- Should store the total number of bits embedded inside the source code somewhere, with or without encryption.

There can be a few very simple ways of embedding and depending upon where to embed secret data bits there can be a few variants accordingly:

- Embed aggressively at every possible places (except possibly inside variable values, constants or string literals), in every keyword and identifier.
- do not use all the characters of a candidate word for embedding, instead only use the first character (change the case depending upon the next secret bit to be embedded, keeping all other characters unchanged).


```

<html>
<head 200>
<title> Test Html (Steganography - Seeing the Unseen)</title>
</head>
<body>

<table border="1" cellpadding="5" cellspacing="10">
<tr>
<td>&nbsp; </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 DCI 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"> 
</td>
</tr>
<tr>
<td align="right" valign="bottom"> 
</td>
<td colspan="2"> <p>Lena is an image that is used to test many of the classical image processing techniques and algorithms including histogram equalization, optimal thresholding<br>
<b>LSB data hiding</b> (LSB data hiding technique changes the LSB s of the image data to hide secret message);<br>
<b>Steganography</b> (Steganography is seeing the unseen)<br>
<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<br>
<b>cover text</b> to get<br>
<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<br>
<b>easy</b> (just you need to do)<br>
<b>view source</b> (and extract the message). Just checking the cases of the html tags reveals the embedded 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)		
<pre> 0100001101101111011100000111100101110010011010010110011101101000011101000010000001000000101 0011011000010110111001100100011010010111000011000010110111000100000100010001100101011100100000000 </pre>		

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

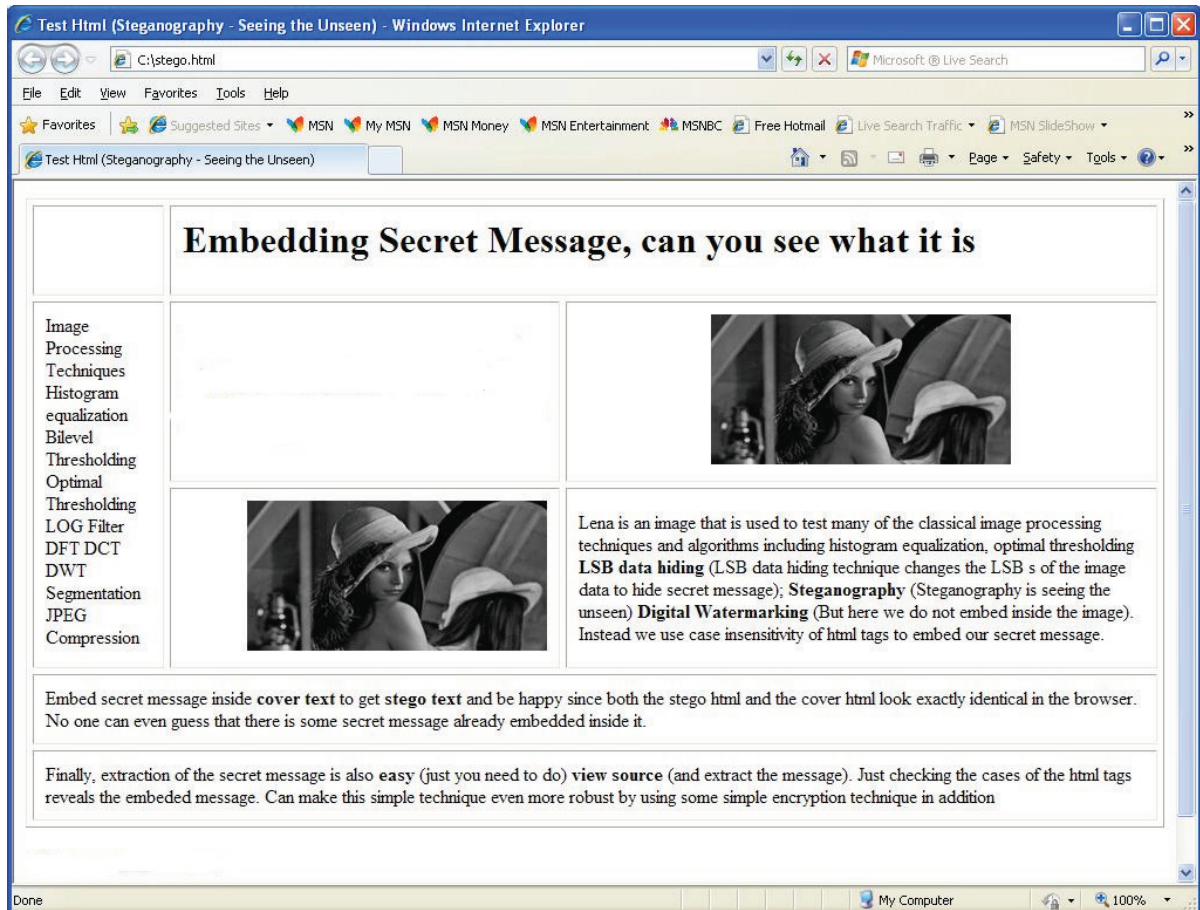


Fig. 5: Cover & Stego Html

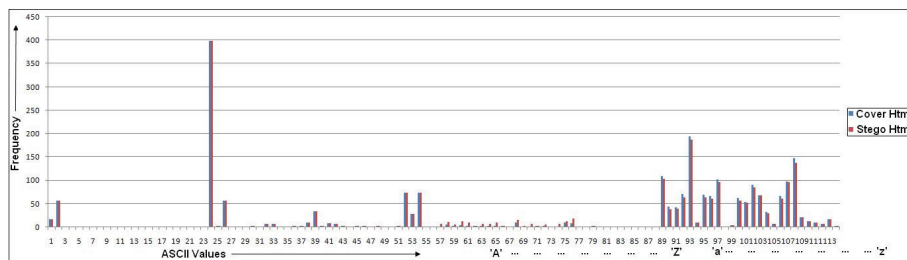
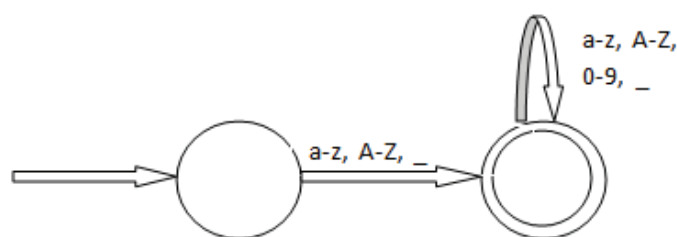


Fig. 6: Cover vs Stego Html Histogram



DFA for an IDENTIFIER

Regular Expression: $[a-zA-Z_][a-zA-Z0-9_]+$

Fig. 7: DFA for accepting IDENTIFIER

- Only embed inside the keywords.
- Only embed inside the identifiers.

3.2 Case-sensitive Programming Language Sources

Languages like C are case-sensitive, we can not use the above mentioned techniques directly. Off course we can hide data inside comments, but what if a source does not have a comment at all? Creating some arbitrary artificial comments and embedding data inside them may not be a good idea. Instead we can use the following simple general technique:

- Use lexical analyzer (some simple scanner for the language) to find IDENTIFIER tokens ($IDENTIFIER: [a-zA-Z_][a-zA-Z0-9_]+$, as shown in Fig. 7).
- Only use variable names to embed secret data, no function name. Also, stick to local/static variables and do not use extern variable names to embed), use some simple parser to achieve this.
- If next message bit to embed is 1, change the identifier name to append (or prepend) by an underscore(_), otherwise leave it as it is (e.g., if the variable name is *var*, change it to *var_*, if the next bit to embed is

1, otherwise keep it as it is, while extraction interpret in the same manner).

- Use some kind of symbol table (hash map) to keep track of every change in identifier name and accordingly reflect the change to all places where the identifier is used.
- Skip compiler directives / Macros/ keywords.
- Keep track of total number of bits embedded (e.g., store it inside a beginning comment, with / without simple encryption).

4 Conclusions

In this paper we presented simple algorithms and techniques for hiding data in html text and other source codes. This technique can be extended to any case-insensitive language and data can 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. Even for C-like case sensitive languages we can embed with minimal distortion by twinking for instance the identifier name a little bit. Data hiding methods in images results distorted stego-images, but data hiding technique proposed for

html 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. available for download from <http://igneous.scis.ecu.edu.au/proceedings/\2008/forensics/Shin\%20Data\%20Hiding\\\%20in\%20Windows\%20Executable\\\%20Files.pdf>
- [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. Egiazarian, "A Generalized Fibonacci LSB Data Hiding Technique," 3rd International Conference on Computers and Devices 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*, ISSN 0972-7272, Volume 6, No 3, 2008, pp. 463-485.
- [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, IHHMSP 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.
- [11] Chi-Kwong Chan, L.M. Cbeng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, Vol. 37, pp. 469-474, 2004.
- [12] Sandipan Dey, Hameed Al-Qaheri, Sugata Sanyal, "Embedding Secret Data in Html Web Page," Published in Book: *Image Processing & Communications Challenges*, Eds. Ryszard S. Choras and Antoni Zabłudowski, Academy Publishing House EXIT, Warsaw 2009, pp. 474-481.

IMAGE PROCESSING & COMMUNICATIONS An International Journal

Instructions for Authors

Contributions to IMAGE PROCESSING & COMMUNICATIONS will be welcomed from throughout the world. Manuscript should be submitted to the Editor - in - Chief.

Manuscripts are reviewed and refereed.

Manuscripts.

Send one original and two photocopies of the manuscript and one original and two photocopies of each illustration. Upon acceptance by the Editor, we will request that you send, directly to the Editor, a soft copy of your paper in either MS Word or LaTeX format. All soft copies of accepted papers may be sent as e-mail attachments directly to Editor at: choras@utp.edu.pl.

The basic publishing fee (without reprints) is 20 Euro/page for the first 8 pages and 60 Euro/page for the extra pages (the minimum charge for the basic fee of a paper is 160 Euros). The hard (printed) issue with the soft copy (CD) plus post service is 85 Euros.

The manuscript must contain the following:

Title page listing the paper title, the name(s) of the author(s), and the affiliation and complete mailing address of each author.

Abstract, adequate as a summary (rather than an introduction), 200 words maximum. It should be possible from the abstract for readers to determine whether or not they have an interest in the paper.

Subject terms or keywords are needed, maximum of eight.

Text must be typed double-spaced. The paper must be A4-Sized (21 x 29,7cm, white) with double columns. The text must be written with 1" margins, 1" top and bottom margins and 1/5" space between columns. The texts do not contain footer and header. The tables, pictures, etc. included in the paper should fit this page size and should also be provided on diskette, the preferable format for graphics is TIFF, BMP, etc..

Equations must be typewritten, not handwritten. Number displayed equations sequentially without letters. Define characters that might be confused (such as el/one, or omega/w) at first usage.

References to published literature must be numbered consecutively in the order alphabetical under the first author's last name. Use the following formats for books or journal articles:

Books: names and initials of all authors, full title, edition, volume number, name of publisher, places of publication, year of publication.

Journals: names and initials of all authors, title of article, journal name spelled out in full, volume number, issue number, first and last page numbers, year of publication.

Author's biographies are printed. Authors should supply a brief biographical summary not to exceed 150 words.

IMAGE PROCESSING & COMMUNICATIONS

An International Journal

Volume 14, Number 2-3

OPTIMIZED ARCHITECTURES OF CABAC CODEC FOR IA-32-, DSP- AND FPGA-BASED PLATFORMS - D. Karwowski, M. Domański	5
TRACKING OF 3D OBJECTS IN A VISION BASED TRAVEL AID SYSTEM FOR THE BLIND - P. Pelczynski	13
OPTICAL FEATURE CLUSTERING ALGORITHM FOR OBJECT TRACKING IN IMAGE SEQUENCES - P. Pelczynski, J. A. Arriola	23
CONSTRAINED CONTOUR MATCHING IN HAND POSTURE RECOGNITION - A. Wilkowski, W. Kasprzak	31
SIGNATURE RECOGNITION METHOD BY MEANS OF THE WINDOWS TECHNIQUE - P. Porwik, K. Wrobel and R. Doroz	43
A NEURAL NETWORK METHOD OF IMAGE RECONSTRUCTION FROM PROJECTIONS USING GRID-"FRIENDLY" PROJECTION ANGLES - R. Cierniak	51
HIDING INSIDE HTML AND OTHER SOURCE CODES - H. Al-Qaheri, S. Dey, S. Sanyal	59
A METHOD FOR SEMI-AUTOMATED ASSESSMENT OF USER SATISFACTION WHEN USING WWW SERVICES WITH MOBILE TERMINALS - A. Flizikowski, M. Choraś, M. Wachowiak, W. Hołubowicz	69