New Data Hiding Techniques via BBS Articles Using Special Big-5 Codes and Their Applications

Yi-An Wang
Institute of Multimedia Engineering
National Chiao Tung University, Taiwan
Email: yian.cs98g@nctu.edu.tw

Wen-Hsiang Tsai
Department of Computer Science
National Chiao Tung University, Taiwan
Email: whtsai@cs.nctu.edu.tw

***Abstract*—A new data hiding approach via the popular Internet forum BBS (bulletin board system) is proposed. Two data hiding techniques utilizing special Big-5 codes are proposed. These special codes usable for data hiding are obtained by continuous tests and observations on common BBS browsers and many different BBS sites. Due to users' article publishing habits and special BBS features, these codes, when hidden in BBS articles using proposed techniques, become imperceptible. Each of the proposed data hiding techniques may be used both for covert communication and for article authentication, creating four combinations of new data hiding techniques and applications. Good experimental results show the feasibility of the proposed techniques.**

***Index Terms*—information hiding, BBS, Big-5 codes, covert communication; authentication.**

I. INTRODUCTION

With the progress of computer and networking technologies, communication via the Internet has become more and more popular nowadays, and so security protection of communication messages on the Internet becomes a necessity. The BBS is a kind of Internet forum. Its screen view is presented simply by monochrome or chromatic text. In Taiwan, the number of users on the most popular BBS site, PTT, can reach 60 to 150 thousand at any time. Users can ask various questions, discuss any matter, interact with one another, and even send BBS mails mutually. Furthermore, journalists also write reports by adopting some conspicuous articles from the BBS. Because the data hiding technique is a good way for safe exchanges of information, it is desired to develop data hiding techniques via BBS articles in this study. To our knowledge of the literature about information hiding research, such work has not been attempted so far.

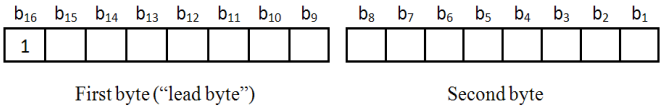
Articles on the BBS belong to soft-copy texts [1]. In recent years, some methods of data hiding via text documents have been proposed, like using file headers [2], file structures, and file features [3]. However, the BBS is not of the format of conventional media such as pictures, videos, and text files, so controllable items like data structures, special file syntax, and even file headers are not found in them. It can only accept text-typed words. Hence, one can implement data hiding techniques on them only by embedding special character codes.

In the past, some data hiding methods using character codes have been proposed. Wayner [4] proposed a method to use the context-free grammar to create secret text messages in cover files for covert communication. Bender et al. [1] proposed the use of infrequent additional spaces to form secret data and transmitted them in soft-copy texts, including inter-sentence spacing, end-of-line spacing, and inter-word spacing in texts.

Additionally, although many information hiding techniques have been proposed in recent decades, methods about hiding data in Internet applications directly are very few. Lee and Tsai [5] and Huang and Tsai [6] proposed some techniques for data hiding by embedding special codes in HTML files to substitute for the original white spaces in the files. In these cases, message data were hidden in HTML files so that these files became stego-media for secret communication or secret sharing when the HTML files are displayed on the Internet. However, these methods are indirect data hiding techniques for Internet applications. In another paper, Lee and Tsai [7] proposed a direct data hiding technique to embed secret data into email text line ends using special ASCII control codes.

These special ASCII control codes are invisible when displayed on the screen and so will not affect a user's reading of the resulting email.

Our goal in this study is to hide data secretly in the popular Internet application, BBS, by embedding some special character codes into BBS articles without changing the article appearances. The used character coding format is the Big-5. It is a double-byte character set for traditional Chinese characters coding and the main text coding format for the BBS. The structure of the standard Big-5 format is shown in Fig. 1.



First byte ("lead byte")	Second byte
0x81 to 0xfe	0x40 to 0x7e, 0xa1 to 0xfe

Fig. 1. Big-5 coding format.

The Big-5 format is compatible with the ASCII format [8]. If the highest bit in a code is "1," the code will be regarded as of the standard 2-byte Big-5 format; otherwise, if the highest bit is "0," then the code will be regarded as an ASCII code without adopting the first byte. Furthermore, the Big-5 format also can be transcoded to the Unicode format and the transcoding table is listed in Microsoft Windows Codepage 950 (CP950) [9].

In this study, two new data hiding techniques via BBS articles are proposed. One uses *invisible* Big-5 codes, and the other special Big-5 *space* codes. We call generally these two kinds of codes *special Big-5 codes*. Each of the two techniques may be used for covert communication and article authentication, creating four new combinations of new data hiding techniques and applications.

In the remainder of this paper, the major idea of the proposed approach and the test procedures to obtain the special Big-5 codes used for data hiding in this study are described first in Section II. The proposed data hiding techniques for covert communication and BBS article authentication are described next in Sections III and IV, respectively. In these sections, experimental results are also included respectively to show the feasibility of the two proposed techniques. Finally, conclusions are made in the last section.

II. MAJOR IDEAS OF PROPOSED METHODS BY USE OF SPECIAL BIG-5 CODES

A. Data Hiding by Invisible Big-5 Codes

To achieve the goal of data hiding in BBS articles, it seems that the invisible ASCII codes proposed by Lee and Tsai [7] mentioned previously may be used because the ASCII codes are compatible with the Big-5 codes as the kernel set. However, this idea was found infeasible because these invisible ASCII control codes are utilized to implement some system functions on the BBS. Instead, we use some *invisible* Big-5 codes for data hiding which were found in this study.

In Taiwan, many BBS's like the PTT and the school BBS sites are built on the servers with the Big-5 coding format, so uses of them for data hiding is appropriate. On the other hand, nowadays most of the popular operating systems such as Windows XP and Windows 7 use the Unicode format as their text coding systems, because the Unicode is a universal and complete standard format. No matter what coding formats are used for text, they will be transformed into the appearance of the Unicode format by these operating systems when they are displayed on the screen. Taking Windows XP as an example. In this operating system which contains many different conversion tables for transcoding between various text coding formats and the Unicode, all the text with the Big-5 format on the BBS will be displayed on the screen with the Unicode format by referring to the *CodePage 950* which is a transcoding table between the Big-5 and the Unicode [9].

For this reason, we tried to find the mapping relationship between all the Big-5 codes and the Unicode codes, and discovered that some special Big-5 codes, which originally represent certain rarely-used Chinese characters or Japanese characters, are *invisible*, and look just like *white spaces* when these codes are transcoded into the Unicode format and displayed on the BBS. This phenomenon resulted from the fact that these corresponding Unicode codes are located in the Unicode *Private Use Area*, which ranges from code E000 to code E8FF and does not contain any

character assignment so that no character code chart is provided for this area.

However, on some popular BBS browsers such as PCMan and Pietty, to facilitate users to read and type some special characters, certain special Big-5 codes mentioned previously are presented in their original appearances through the simulated *Unicode compensation plan* implemented by the BBS browser software. So, by continuous tests and observations in our experiments on many different BBS sites through popular BBS browsers including PCMan, KKMan, Pietty, and the basic Windows XP telnet connection program, we found 185 special Big-5 codes useful for our study. In addition, we supplemented the 185 codes to a total of 256 symbols by padding a white space after each of the first 71 ones of them. The procedure of these tests and observations is shown in Fig. 2.

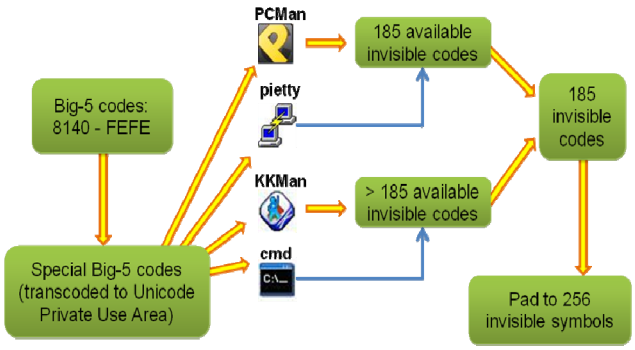
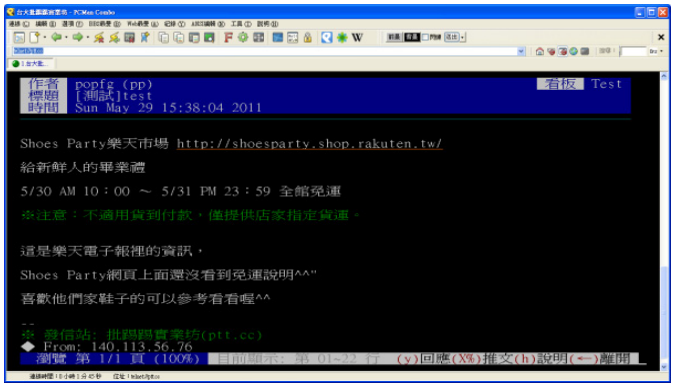


Fig. 2. Tests and observation procedure.

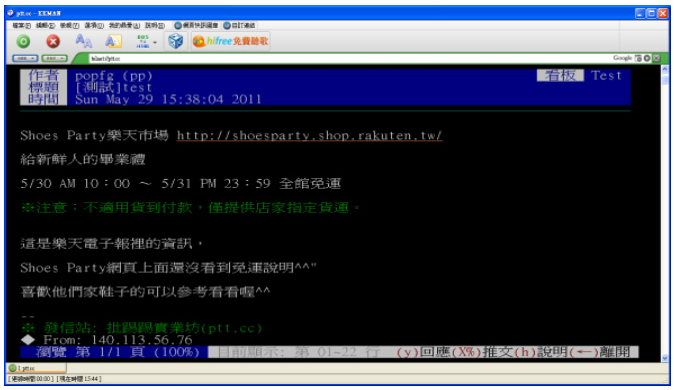
Two examples of the appearances of embedding some of these symbols in BBS articles on the aforementioned browsers are shown in Fig. 3. And some of the codes are listed in Table 1. We have also created an *end signal* which is composed of a special Big-5 code, FEAE, and the original white space. This signal is used in the data hiding process to identify the end of a text line in which data are embedded.

B. Data Hiding by Special Big-5 Space Codes

We have also proposed another data hiding technique for the BBS by the use of some special Big-5 *space* codes. In this technique, two kinds of Big-5 codes are used, one being the original white space code and the other a Big-5 space code. Because the two codes are both included in the Big-5 standard, and appear to be invisible when they are displayed on BBS browsers, we can use them to achieve the aim of data hiding in the BBS article under most general operating systems by assembling them in a proper order.



(a)



(b)

Fig. 3. Stego-articles with some embedded invisible Big-5 symbols on (a) PCMan and (b) KKMan.

Table 1. Encoding table of invisible Big-5 codes (partially shown).

0	8F 53	22	92 65	44	9D 78
1	90 F6	23	98 FB	45	9D 79
2	90 F7	24	98 FD	46	9D A1
3	90 F9	25	98 FE	47	9D A2
4	90 FA	26	99 FB	48	9D A3
5	91 C7	27	9B DE	49	9D A4
6	91 C8	28	9B DF	50	9D A5
7	91 CF	29	9B E3	51	9D A6
8	91 D0	30	9C D7	52	9D A7
9	91 D8	31	9C D9	53	9D A8
10	91 D9	32	9C E4	54	9D A9
11	91 DA	33	9C E7	55	9DAA
12	91 DF	34	9D 5F	56	9D AB
13	91 E1	35	9D 60	57	9D AC
14	91 E2	36	9D 61	58	9DAD
15	91 E3	37	9D 62	59	9D AE
	…		…		…

On the BBS, many users are accustomed to publishing articles with alternate blank lines and this habit facilitates us to hide secret messages after the *line feed* code of each line end. So, we tried to devise an appropriate scheme to efficiently utilize the two mentioned invisible codes for the largest utilization ratio of the blank spaces in each line.

More specifically, there are two kinds of character lengths on the BBS. One occupies a *unit* which is the length of a code displayed on BBS browsers, like the original white space; and the other is two-unit long, like the special Big-5 space code. For the *variability* and *efficiency* of using the Big-5 symbols for data hiding, we allow them to be arranged in three ways: (1) a special Big-5 code, (2) a combination of a special Big-5 code and a white space, or (3) a combination of a special Big-5 code and several white spaces, as shown in Table 2, which we mention as an *encoding table* for the used special Big-5 space codes. Here, *efficiency* is judged by the average required number of units for hiding one bit.

Table 2. Encoding table for used special Big-5 space codes.

Bit stream (binary)	00	01	10	11	End signal
Special Big-5 codes (embedded symbol)					
Occupied units	two	three	four	five	two

(Note: : special Big-5 space code. : original white space code. : line feed code.)

C. *The Proposed Coding Scheme and Coding Efficiency*

In our study, we only use four types (except the *end signal*) of symbols for coding of message bits, including:

type 1:  , type2:  , type3:  , and type4:  .

Specifically, the four types in order are used to encode the bits 00, 01, 10, 11, respectively. Though we can create more types of symbols following the same symbol-creating logic by padding more white space codes after the symbol  , yet we can prove that the *4-symbol codes* created in Table 2 have the *largest* efficiency of symbols, as discussed next.

Assume that the probability for each symbol to appear is identical, and that a symbol can represent n embedded bits. And let u_i and p_i specify the occupied units and the appearance probability of the *i-th* type of symbols like those defined in Table 2, respectively. Then, the *efficiency* E of the symbols is defined by the following equation in this study:

$$E = (\sum_{i=1}^{2^n} u_i \times p_i) / n . \quad (1)$$

Also, under the assumption that all the symbols have equal appearance probabilities, we may substitute u_i and p_i in (1) above with their real values to obtain

$$E = f(n) = [2 \times \frac{1}{2^n} + 3 \times \frac{1}{2^n} + \ldots + (2^n + 1) \times \frac{1}{2^n}] / n \quad (2)$$

which can be reduced easily to be

$$f(n) = (2^n + 3)/2n \quad (3)$$

and differentiated to get

$$\frac{df(n)}{dn} = g(n) = [(2^{n+1}) \times n \times \ell n(2) - 2^{n+1} - 6] / 4n^2 \quad (4)$$

where ℓn is the natural logarithm function. Setting $g(n) = 0$ in (4) above results in the following equation:

$$(2^{n+1}) \times n \times \ell n(2) - 2^{n+1} - 6 = 0 \quad (5)$$

which can be solved to get the extreme value for n. However, because the number of bits must be an integer and such an integer satisfying Eq. (5) is inexistent, we must take n to be 2 for $g(n)$ to be closest to zero. Alternatively, from Eq. (3) we have $f(n)$ = 5/2, 7/4, and 11/6 for n = 1, 2, and 3, respectively. Since 5/2 ≥ 7/4 and 7/4≤ 11/6, we see that n = 2 indeed is an optimal value to make $f(n)$ minimum, i.e., to yield the smallest average required number of units, 7/4, for hiding one bit. This completes the proof that the 4-symbol codes (except the end signal) listed in Table 2 are optimal, and yield the largest efficiency of coding. By the way, we created the end signal to be composed of a special Big-5 space code and a *line feed* code, rather than a white space code and a *line feed* code, since all white spaces between any other code and the *line feed* will be removed when an article is published on the BBS.

Because secret information embedded by the two proposed data hiding techniques is almost imperceptible on the BBS even when a user highlights BBS articles by a mouse, we can use the techniques to achieve the goals of covet communication and article authentication on the BBS.

III. COVERT COMMUNICATION VIA THE BBS

A. Proposed Algorithm for Data Embedding

When a *stego-article* is displayed on the BBS, it is desired that the article body can fit the width of the BBS article window to reduce the possibility of arousing a hacker's notice of the embedded message data. For this aim, we adopt the following strategy.

(1) First, we fold longer article lines into shorter ones, leaving at least eight characters at each line end as a *data embedding slot*;
(2) Next, we use a secret key as a seed to randomize the content of a secret message which we want to embed in a cover article for covert communication;
(3) Then, we map the randomized message data to corresponding invisible symbols according to the user's choice — if the first data hiding technique as mentioned previously is chosen, we conduct the mapping by referring to Table 1; otherwise, we replace each special Big-5 space code in the cover article with two original white space codes (for the process of data extraction described in the next section to be performed correctly), and conduct the mapping by referring to Table 2.
(4) Finally, we sequentially embed the symbols obtained from the mapping into the folded article to obtain a stego-article with the randomized secret message data hidden imperceptibly.

The algorithm for conducting this process is described in the following, in which *a line* in a BBS article means a number of characters in a row with an LF appended to the end of the line.

Algorithm 1: message data embedding for covert communication.

Input: a secret message S, a secret key K, and a cover BBS article B.

Output: a stego-article B'.

Steps.

1. Fold sequentially each text line l_i with a length larger than 70 units in BBS article B into a 70-unit line by inserting a line feed, denoted as LF and occupying *zero unit*, *after the original 70th character in* l_i to generate a *folded article*, denoted as F.
2. Compute the size L_i of the data embedding slot at the end of each text line l_i in F by:
   $$L_i = 78 - \textit{the length of } l_i,$$
   which means that the maximum number of characters that can be inserted at the end of l_i.
3. Use secret key K as a seed to generate a sequence Q of random numbers.
4. Randomize the input secret message S with Q to get a *randomized message* S'.
5. Choose a technique to hide S':
   (1) If technique 1 is chosen, then perform Step 6.
   (2) If technique 2 is chosen, then go to Step 9.
6. (*Technique 1*) Separate the bits of S' into 8-bit segments and map them to invisible symbols t_1, t_2, ..., t_k, respectively, according to Table 1.
7. Let $|l|$ be the total number of lines, $|T|$ be the total number of t_1 through t_k, and Ut_1, Ut_2, …, Ut_k be the numbers of units occupied by t_1 through t_k, respectively.
8. Embed the invisible symbols t_1 through t_k obtained in Step 6 sequentially into the folded article F from the first line (that is, take the index number i of l_i and the index number k of t_k both to be 1 initially), and then conduct the following steps.
   8.1 If $i \le |l|$, then perform one of the following three operations at the end of l_i; otherwise, perform Step 8.2.
      (1) If $k \le |T|$ and $L_i - Ut_k > 2$, then embed t_k in the data embedding slot of l_i, decrement L_i by Ut_k, increment k by 1, and repeat Step 8.1 again.
      (2) If $k \le |T|$ and $L_i - Ut_k \le 2$, then scan l_i to find the line feed LF, remove it, embed an end signal in the data embedding slot of l_i, increment i by 1, and repeat Step 8.1 again.

(3) If $k > |T|$, then embed an end signal in the data embedding slot of l_i, and go to Step 13.

8.2 Embed the remaining symbol/symbols below F as one or more blank lines with an end signal appended at each line end, and go to Step 13.

9. (*Technique 2*) Replace each special Big-5 code in the folded article F with two white space codes.
10. Separate the bits of S' into 2-bit segments and map them to invisible symbols p_1, p_2, ..., p_k, respectively, according to Table 2.
11. Let $|l|$ be the total number of lines, $|P|$ be the total number of p_1 through p_k, and Up_1, Up_2, …, Up_k be the numbers of units occupied by p_1 through p_k, respectively.
12. Embed the invisible symbols p_1 through p_k sequentially into the folded article F from the first line (that is, take the index number i of l_i and the index number k of p_k both to be 1 initially), and then conduct the following steps.

12.1 If $i \le |l|$, then perform one of the following three operations at the end of l_i; otherwise perform Step 12.2.

(1) If $k \le |P|$ and $L_i - Up_k > 2$, then embed p_k in the data embedding slot of l_i, decrement L_i by Up_k, increment k by 1, and repeat Step 12.1 again.

(2) If $k \le |P|$ and $L_i - Up_k \le 2$, then scan l_i to find the line feed LF, remove it, embed an end signal in the data embedding slot of l_i, increment i by 1, and repeat Step 12.1 again.

(3) If $k > |P|$, then embed an end signal in the data embedding slot of l_i, and perform Step 13.

12.2 Embed the remaining symbols below F as one or more blank lines with an end signal at each line end, and continue.

13. Take the final version of F as the desired stego-BBS article B'.

B. Proposed Algorithm for Data Extraction

In the proposed message data extraction process, first we extract the invisible symbols embedded in a stego-article. Next, according to the adopted different techniques (technique 1 or 2 mentioned previously) for embedding the invisible symbols, we conduct two different data extraction processes. If technique 1 is used, we map the symbols into 8-bit segments by referring to Table 1; otherwise, we map the symbols into 2-bit segments by referring to Table 2. Then, we concatenate the segments into a random message. Finally, by using the same secret key which was used for embedding the message, we can recover the correct secret message. The details are described as an algorithm below.

Algorithm 2: Data extraction for covert communication.

Input: a stego-BBS article B' and the secret key K used in Algorithm 3.1 to produce B'.

Output: a secret message S.

Steps.

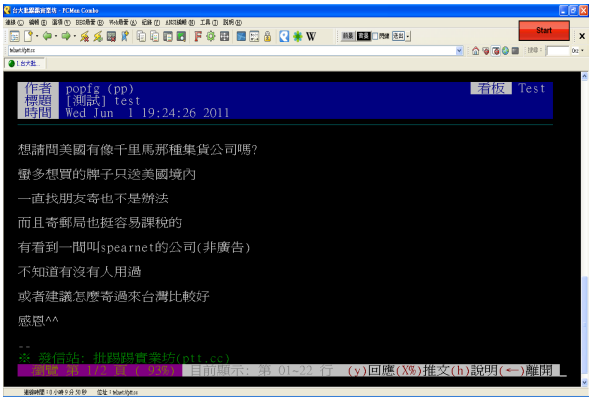
1. Check each line l_i in the stego-BBS article B' sequentially, starting from the first line; and extract the invisible symbols embedded in front of the end signal in l_i.
2. Transform the extracted symbols according to the respective data hiding technique used for embedding the secret message to be extracted.
   (1) If technique 1 is used, map them into 8-bit segments t_1, t_2, ..., t_k, respectively, by referring to Table 1.
   (2) If technique 2 is used, map them into 2-bit segments p_1, p_2, ..., p_k,, respectively, by referring to Table 2.
3. Concatenate the extracted segments into a random message Q.
4. Use the secret key K to reorder Q to obtain a result as the desired secret message S.

C. Experimental Results

We conducted experiments of covert communication via BBS articles using our program designed as a PCMan plug-in. Fig. 5 shows an example of the experimental results where a message “I want to tell you a secret” was embedding by Algorithm 1 into a BBS article shown in Fig. 5(a), resulting in the stego-BBS article of Fig. 5(b). Fig. 6 shows message data extraction results using Algorithm 2 with correct and wrong keys. The experimental results show

that the proposed method is effective for covert communication applications.

(a)



(b)

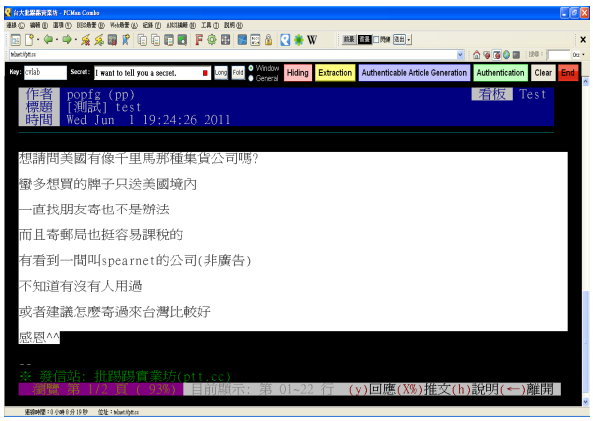
Fig. 5. An example of experimental results of message data embedding. (a) Cover article publishing and secret embedding process. (b) Stego-article with secret message embedded on the PCMan.

IV. BBS ARTICLE AUTHENTICATION

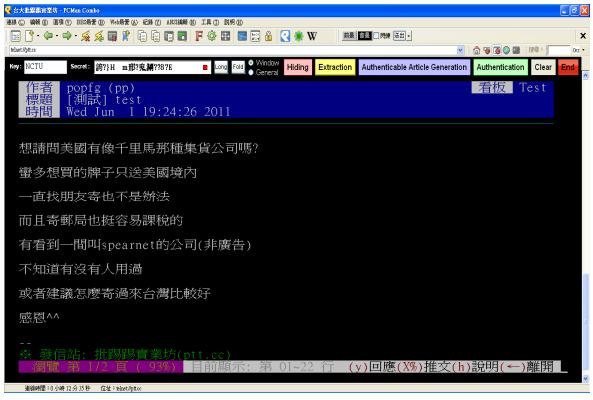
The basic concept we follow to design BBS article authentication techniques is to generate authentication signals from a BBS article to be protected and embed them into the article. Later, to authenticate the article, the authentication signals, presumably embedded in the article, are extracted and matched with those computed from the current content of the article; if matching fails, then it is decided that the article has been tampered with. More details are described in the following.

A. Proposed Authentication Signal Generation and Embedding Process

In the phase of authentication signal generation, first we fold the longer text lines in a BBS article, which we want to protect, into shorter ones, leaving at least eight characters at the end of each line as a data embedding slot. Then, assuming that technique 2 mentioned previously, which uses special Big-5 space codes, is selected to hide authentication signal data, we replace each Big-5 space code in the cover article with two white space codes. Next, we remove from the folded BBS article all the line feed signals so that the verification process described in the next section will not be interfered by redundant line feed signals. The modified BBS article and a secret key then are used to generate an authentication signal using a hash function and the exclusive-OR operation. Finally, we regard the signal as a secret message and hide it in the folded article using the proposed data embedding process described in Algorithm 1 with the same secret key to obtain a protected BBS article. By the way, technique 2 may also be used in the above process.



(a)



(b)

Fig. 6. Data extraction results from a stego-BBS article. (a) Extraction result with a correct message using a correct key. (b) Extraction result with a noise message using a wrong key.

Algorithm 3: Authentication signal generation and embedding.

Input: a secret key K, a hash function f (such as MD5), and a cover BBS article B.

Output: a protected BBS article B'.

Steps.

1. Fold sequentially each text line l_i with a length larger than 70 units in BBS article B into a 70-unit line by inserting a line feed, denoted as LF and occupying zero unit, after the original 70th character in l_i to generate a *folded article*, denoted as F.
2. Choose one of the two previously-described data hiding techniques to embed secret data in the folded article by one of the following ways:
   (1) if technique 1 is selected, then perform Step 4;
   (2) if technique 2 is selected, then perform Step 3.
3. Replace each special Big-5 code in the folded article F with two white space codes.
4. Remove all the line feed signals in F, use the result and the secret key K as inputs to the hash function f to generate two 128-bit digests F' and K', respectively, and return all the removed LF signals back into their original positions in F.
5. Compute the exclusive-OR value $F' \oplus K'$ to obtain a 128-bit authentication signal S.
6. Regard S as a secret massage and embed it in the cover article using the proposed data hiding process in Algorithm 1 to obtain a protected BBS article as output.

B. Proposed Authentication Signal Extraction and Verification Process

To authenticate a protected BBS article, first we take it as input to the proposed data extraction process in Algorithm 2 with the secret key used in Algorithm 3 to obtain a secret massage, and regard it as the authentication signal S of the article. Next, we use the same key and the same hash function to transform the article, after all the secret symbols and the line feed signals in it are removed, into a verification signal T. Finally, we decide whether the protected article has been modified or not by comparing the two signals S and T. The details are described as an algorithm in the following.

Algorithm 4: Authentication signal extraction and BBS article verification.

Input: a secret key K and a hash function f both being the same as those used in Algorithm 5.1; and a protected BBS article B'.

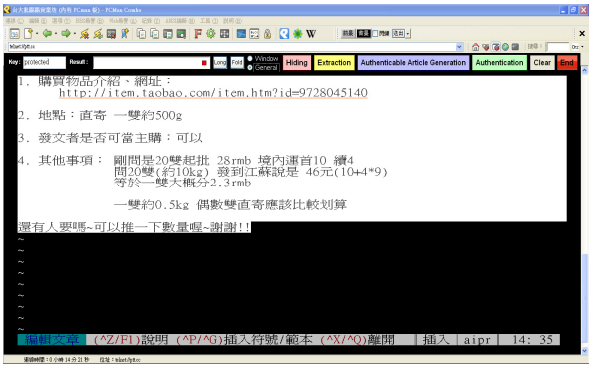
Output: an authentication report R.

Steps.

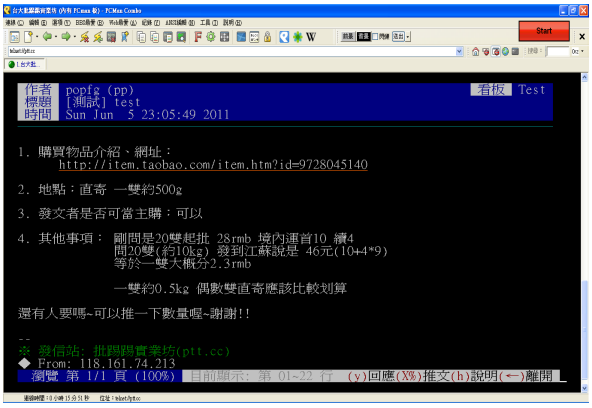
1. Use the protected BBS article B' as input to the proposed data extraction process in Algorithm 2 with the secret key K to obtain a secret massage, and regard it as the authentication signal S of the article.
2. Remove all the secret symbols and line feed signals from the BBS article, and use the result, as an input to the hash function f to generate a 128-bit digest B''.
3. Use the secret key K as an input to the hash function f to generate a 128-bit digest K'.
4. Compute the exclusive-OR value $B'' \oplus K'$ to get a 128-bit verification signal T.
5. Compare the authentication signal S and T, resulting in the following two cases.
   (1) If $S = T$, then regard the input B' as *unmodified* and mark it so in the authentication report R.
   (2) If $S \neq T$, then regard B' as *modified* and mark it so in R.
6. Output the authentication report R.

C. Experimental Results

Some experimental results of BBS article authentication are shown here. In Fig. 7(a), we show par of the process of generating a protected BBS article using a secret key as input to Algorithm 3, and the appearance of the generated protected article displayed on the PCMan is shown in Fig. 7(b). A successful verification result with Fig. 7(b) and a right secret key as inputs to Algorithm 4 is shown in Fig. 8(a), in which an authentication report “authentication is successful” is seen. After we tampered with the protected article by replacing a number “500” of the content with another “900,” and conducted the verification process using Algorithm 4 with the tampered article and the right key as inputs, we obtained an authentication result with a report of failure, as shown in Fig. 8(b).



(a)



(b)

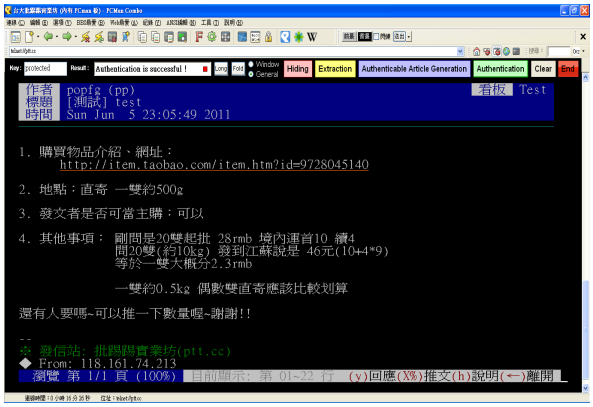
Fig. 7. Experimental results of authentication signal generation and embedding. (a) Protected article generation and publishing process. (b) Stego-article with an embedded secret message displayed on PCMan.

V. CONCLUSIONS

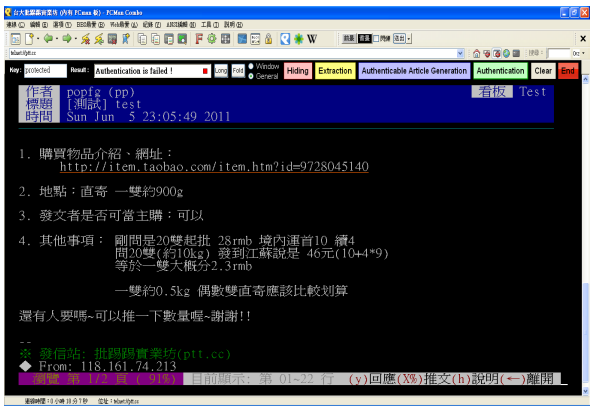
A. Process for Private Region Concealment

In this study, we investigate the new problem of data hiding via BBS articles and two techniques have been proposed. One is based on the use of invisible Big-5 codes, and the other on the use of special Big-5 space codes. We use the two proposed techniques to encode a secret message and embed the resulting secret symbols into a folded article to achieve covert communication via BBS articles. According to the experimental results, the secret message hidden in a BBS article is not observable from the appearance of the resulting stego-article. It has also been proposed to enhance the security of the proposed method by adding a user-defined secret key to randomize the content of the secret message, so that a malicious user cannot easily extract the secret even when he/she knows the proposed algorithm.

Furthermore, we have also used the two proposed techniques to accomplish BBS article authentication by generating and matching authentication signals. The experimental results proved the feasibility of the proposed data hiding techniques for real applications. Future studies may be directed to investigating more applications by use of the proposed techniques. For example, in addition to using the proposed techniques to authenticate protected BBS articles, it may also be tried to capture the screen displaying a BBS article by the "print screen" command, transform it to a grayscale image, and hide it into the BBS article to enhance protection of the BBS content.



(a)



(b)

Fig. 8. Experimental results of authentication signal extraction and BBS article verification. (a) An authentication result with a protected article untampered. (b) An authentication result of a protected article tampered by replacing a word.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM System Journal*, Vol. 35, Nos. 3 & 4, Feb. 1996.

[2] G. Cantrell and D. D. Dampier, “Experiments in hiding data inside the file structure of common office documents: a steganography application,” *Proceedings of 2004 International Symposium on Information and Communication Technologies,* pp. 146-151, Las Vegas, Nevada, U. S. A., 2004.

[3] T. Y. Liu and W. H. Tsai. "Quotation authentication: a new approach and efficient solutions by data hiding and cascaded hashing techniques," *IEEE Transactions on Information Forensics and Security*, Vol. 5, No. 4, pp. 945-954, Dec. 2010.

[4] P. Wayner, “Strong theoretical steganography,” *Cryptologia*, Vol. XIX/3, pp. 285-299, 1995.

[5] I. S. Lee and W. H. Tsai, “Secret communication through web pages using special space codes in HTML files,” *International Journal of Applied Science and Engineering*, Vol. 6, No. 2, pp. 141-149, Nov. 2008.

[6] K. L. Huang and *W. H. Tsai*. “Secret sharing with steganographic effects for HTML documents,” *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing*, Hualien, Taiwan, Aug. 2004.

[7] I. S. Lee and W. H. Tsai, “Data hiding in emails and applications by unused ASCII control codes,” *Journal of Information Technology and Applications*, Vol. 3, No. 1, pp. 13-24, Sept. 2008.

[8] American National Standard for Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986, American National Standards Institute, Inc., March 26, 1986.

[9] CP950 to Unicode table: http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WINDOWS/CP950.TXT, Jul. 2000