Data Hiding in Emails and Applications by Unused ASCII Control Codes

利用未使用的 ASCII 控制碼將資料隱藏於電子郵件及其應用

[1,3]I-Shi Lee (李義溪) and [1,2]Wen-Hsiang Tsai (蔡文祥)

[1]Dept. of Computer Science, National Chiao Tung University
[2]Dept. of Computer Science and Information Engineering, Asia University
[3]Dept. of Information Management, Technology and Science Institute of Northern Taiwan

E-mails: gis87809@cis.nctu.edu.tw & whtsai@cis.nctu.edu.tw

Abstract

A new technique for data hiding in emails via Outlook Express and IE is proposed. Secret data are encoded by special ASCII control codes and embedded into cover emails by inserting the data into email text line ends. These special codes were found out by a systematic test of all the ASCII codes on various email server software systems and standards. They are invisible to the user when displayed, achieving the effect of steganography. The proposed data encoding technique is a combination of five coding rules found in this study. The inserted codes will not change the meanings of the sentences in the cover email, neither causing any noticeable difference to the reader. Furthermore, hidden data can be extracted from a stego-email completely to recover the original email text content. Also described are two applications of the proposed data hiding technique, namely, covert communication via emails and authentication of emails. In the former application, security is enhanced by the use of a secret key, and in the latter, an authentication signal is generated from the cover email for email fidelity checking. Good experimental results show the feasibility of the proposed method.

摘要

本文提出一種透過 Outlook Express 與 IE 將資料隱藏於電子郵件中的新技術。秘密資料係以特殊的 ASCII 控制碼加以編碼並藏入在電子郵件本文的每行行尾。這些特殊碼是在有系統的將所有 ASCII 碼在各種不同的電子郵件伺服器系統及標準上測試後所得到的。當顯示時使用者看不到他們，因此達到資料隱藏的效果。本論文的資料編碼技巧是組合 5 種本研究所找到的編碼規則。插入原始電子郵件的密碼不會改變郵件本文句子的意義，也不會造成讓讀者注意到的變化。此外隱藏的資料可以完全地抽取出來並回復原來內容。本論文也提出 2 種資料隱藏技巧的應用，即秘密傳輸及電子郵件鑑定。對前者可使用秘密金鑰加強安全，後者用的鑑定信號可自原始郵件中產生，用以檢查郵件的完整性。良好的實驗結果顯示本論文所提方法的可行性。

Key words: data hiding, ASCII control codes, email, covert communication, email authentication

關鍵詞：資料隱藏，ASCII 控制碼，電子郵件，秘密通信，電子郵件鑑定

1. Introduction

With the advance of the network technology, a great deal of information is transmitted conveniently on the Internet. But some users might use the Internet to collect, copy, or intercept other people's data illegally. Cryptography is generally adopted to protect important data from being tampered with. Another solution to this problem is to use the data hiding technique, which embeds crucial data into given carriers, making the embedded information imperceptible to viewers. The given carrier may be various kinds of multimedia like image, text, video, audio, and image [1-3]. In the sequel, we call the carrier into which information is to be embedded a *cover carrier*, and the result of embedding a *stego-carrier*. In this study, we deal with emails, so a *stego-*

email is the result of information embedding in a *cover email*.

In contrast with other multimedia, digital texts contain less redundant information for embedding data. Most data hiding methods for digital text documents try to encode information directly *into the text itself* or *into the text format*. One way of into-text hiding is to exploit the natural redundancy of languages, and one way of into-format hiding is to adjust inter-word or inter-line space [3]. On the other hand, from the steganographic point of view, digital text documents can be classified into two types: hard-copy and soft-copy [2]. A hard-copy text may be treated as a binary image resulting from scanning a text document, while a soft-copy text may be regarded as an American Standard Code for Information Interchange (ASCII) text that can be edited by a text editing software like Notepad.

For a hard-copy text, which is interpreted as a highly-structured image, information can be embedded into the layout or format of the image. Maxemchuk et al. [4-5] presented text-based steganographic methods which use the distances between consecutive lines of texts or between consecutive words to hide information. If the space between two lines is smaller than a threshold, a "0" is represented; otherwise, a "1."

In contrast with hard-copy texts and other digital media, soft-copy texts are more difficult to hide data due to the lack of redundant information. Even a slight modification, like rewriting a letter, may be noticed by a reader. However, huge amounts of text documents that people deal with daily on the Internet are essentially soft-copy texts in nature. Thus, the protection of digital rights of this type of text document becomes more and more important.

Bender et al. [2] proposed the use of infrequent additional spaces to form secret data and transmit them in soft-copy texts, including inter-sentence spacing, end-of-line spacing, and inter-word spacing in texts. For example, one space between words is taken to represent a "0" and two spaces a "1." Wayner [6] proposed a method to use the context-free grammar to create secret text messages in cover files for covert communication; the secret message is not embedded in the cover file directly. And a receiver extracts the hidden message by parsing. A constraint is that the cover text should be a meaningful message; otherwise, a reader will doubt it.

Cantrell and Dampier [7] proposed to embed data into unused spaces in file headers. These spaces are invisible to usual users because they are disregarded when the files are opened. The spaces can be seen when examined at the byte level, but few users would do so. Johnson et al. [8] proposed another way to embed information in unused spaces that are imperceptible to an observer, which is based on the fact that usually operating systems allocate more space than the need of a file and the result leaves some unused space to hide information. A third method is to create a hidden partition in a file system to embed information. The partition is not viewed normally. This concept has been expanded in a steganographic file system [9]. If a user knows the file name and the password, access to the file will be granted; otherwise, no evidence of the file will be revealed in the system of the hidden files.

Characteristics inherent in network protocols may also be taken advantage of to hide information [10]. For example, TCP/IP packets can be used to transmit secret messages across the Internet by embedding unused spaces in the packet header. Finally, Chang and Tsai [11] proposed a special space encoding to embed copyright information into the HTML text content. The bit "1" is encoded by inserting a so-called pseudo-space string " " before a real space, while the bit "0" is represented by a normal space between two words or sentences.

It is impossible for a single administrator to check all data on networks daily because of the huge amount of data. For this reason, email, a kind of digital text document, is very suitable for use as a cover carrier to hide secret information. Nowadays, most users transmit and receive emails either by Outlook Express of Microsoft or by a browser like the Internet Explorer (IE). The type of email handled by the latter way is called *web mail*.

In this paper, we propose a new technique for data hiding in emails via Outlook Express and IE under the operating system of the traditional Chinese version of Microsoft Windows XP, service pack 2, 2002. The idea is based on the use of unused ASCII codes. Secret data are encoded by special ASCII control codes and embedded into cover emails by inserting the data into the text line ends in the body of a given email. These ASCII control codes, when displayed both by Outlook Express and IE, are invisible to the user, achieving the effect of steganography. Such invisible ASCII control codes were found out in this study by a systematic test of all the ASCII codes on various email server software systems and standards. The proposed data encoding technique is a combination of five coding rules found in this study, which insert special ASCII control codes into different places in email texts. The inserted codes will not change the meanings of the sentences in the cover email, neither causing any noticeable difference to the reader. Furthermore, hidden data can be extracted from a stego-email completely to recover the original email text content. Also described in this paper are two applications of the proposed data hiding technique, namely, covert communication via emails and authentication of emails. In the former application, security is enhanced by the use of a secret key, and in the latter, an authentication signal is generated from the cover email for email fidelity checking.

In the remainder of this paper, some properties of ASCII control codes and some email systems useful for this study are described in Sections 2 and 3, respectively. The techniques we propose for encoding secret data by ASCII control codes and embedding them into emails are described in Section 4. In Section 5, we describe an algorithm which implements the proposed data hiding method for covert communication. In Section 6, we describe the proposed data recovery technique and a corresponding algorithm. In Section 7, we describe an algorithm for the application of email authentication. Some experimental results are given in Section 8, followed by conclusions and discussions in Section 9.

2. Properties of ASCII Control Codes

ASCII codes, usually expressed as hexadecimal numbers, are used very commonly to represent text for information interchange on computers. Parts of the ASCII codes, namely, from 00 through 1F, are used as *control codes* which are listed in Table 1. They were originally designed to control computer peripheral devices like printers, tape drivers, teletypes, etc. But now they are rarely used for their original purpose because of the rapid development of new peripheral hardware technologies, except those codes for text display control, such as 0A with the meaning of *line feed* and 08 with the meaning of *backspace*. Besides, some of the control codes, when displayed by a text editing program or a browser on monitors, are invisible; and some others are shown as spaces under certain software environments, just like the function of the original ASCII space code 20. These two types of ASCII codes may be utilized to increase secret data encoding variability in the data hiding process. For convenience of reference, we say that the former type displays a *null space*, in contrast with the *white space* displayed by the latter type.

On the other hand, as computer technology spreads throughout the world, many coding standards have been developed to facilitate the expression of non-English alphabets. But these alphabet coding standards, such as the Unicode and the Big 5, all include the ASCII codes as the kernel set. For example, the popular Unicode standard, UTF-8, equates exactly to the ASCII codes for code values below 128. Therefore, the good property of the ASCII control codes for embedding secret data in text documents is still preserved in various coding standards.

In this study, it is desired to use the white-space and null-space codes to embed data in text documents of the Unicode UTF-8 format without causing noticeable artifacts under the popular software environments of Outlook Ex-

press, IE and the operating system of the traditional Chinese version of Microsoft Windows XP, service pack 2, 2002.

Table 1. ASCII control codes and description

Dec	Hex	Char	Description
0	0	NUL	null character
1	1	SOH	start of header
2	2	STX	start of text
3	3	ETX	end of text
4	4	EOT	end of transmission
5	5	ENQ	enquiry
6	6	ACK	acknowledge
7	7	BEL	bell (ring)
8	8	BS	backspace
9	9	HT	horizontal tab
10	A	LF	line feed
11	B	VT	vertical tab
12	C	FF	form feed
13	D	CR	carriage return
14	E	SO	shift out
15	F	SI	shift in
16	10	DLE	data link escape
17	11	DC1	device control 1
18	12	DC2	device control 2
19	13	DC3	device control 3
20	14	DC4	device control 4
21	15	NAK	negative acknowledge
22	16	SYN	synchronize
23	17	ETB	end transmission block
24	18	CAN	cancel
25	19	EM	end of medium
26	1A	SUB	substitute
27	1B	ESC	escape
28	1C	FS	file separator
29	1D	GS	group separator
30	1E	RS	record separator
31	1F	US	unit separator

3. Properties of Email Systems

In this study, it is assume that all emails are transmitted through the popular Simple Mail Transfer Protocol (SMTP) [12-14] and that users retrieve their emails from remote server systems of the Post Office Protocol version 3 (POP3) standard [15]. In addition, most emails nowadays are of the Multipurpose Internet Mail Extensions (MIME) format [16-18] which is compatible with the SMTP standard.

However, some mail server systems do not follow the SMTP standard precisely [18]. Therefore, before we make use of an email document for data embedding, we must find out servers which do not change the content of an email body, or must set up a new SMTP server. Otherwise, data embedded in the email might be destroyed before being read and retrieved on the server of the receiver end.

According to the SMTP standard [14], the body of an email consists simply of text lines of ASCII codes. The codes 0D for carriage return (CR) and 0A for line feed (LF) must appear together as 0D0A (denoted as CRLF in the sequel) for use at the end of each line. A text line, if folded, should be limited to be 78 characters in length, excluding CRLF. Here, by *folding* we mean to split a long text line into multiple shorter ones. A folding will occur when a CRLF is inserted in a line to replace a space, separating the line into two parts.

Outlook Express, after being opened, often has a smaller window for viewing the mail content. The window width is about 70 characters. In this study, we propose to hide secret data in an email by adding ASCII control codes at the end of each text line with the resulting line being *of this width*, such that when the resulting stego-email is opened by Outlook Express, the mail body can fit the window width, thus increasing the steganographic effect. For this aim, we fold the original email lines into shorter ones, each being 65 characters in length, leaving 5 characters at each line end as a data embedding *slot*.

Another popular protocol by which emails are accessed on a server is the Internet Message Access Protocol version 4 (IMAP4) [19]. The IMAP4 supports single web-mail servers and permits manipulations of mailboxes as remote message folders in a way that is functionally equivalent to local folders. Web mails enjoy its popularity because people can use the same client software to both surf the Internet and transmit/receive emails. And IE is probably the most popular browser for manipulating web mails. In this study, we assume that Outlook Express 6.0 and IE 6.0 are used as the client software to manipulate emails.

4. Embedding ASCII Control Codes into Emails

In this study, we identify five possible ways for secret data embedding in emails by use of ASCII control codes. They are listed as follows.

(1) *White-space coding* --- As mentioned previously, there are many different white-space codes, each of which, when displayed, appears to be a white space, yielding the same effect as the original ASCII space code 20. For example, under the environment of the Big 5 standard using Outlook Express, each of the three ASCII codes, 07, 09, and 0C, will be displayed as a white space, as found in this study. Therefore, we can use each of them to replace a white space in an email text in a data hiding process, with the resulting stego-email bringing no reader's notice.

(2) *Inserting multiple white-space codes at text line ends* --- We may place multiple white-space codes before the CRLF at the end of a text line. Since no character but *background* white spaces are shown after the CRLF, these additionally inserted white-space codes, though displayed as *visible* white spaces, will be connected to the background white spaces and thus bring no noticeable effect to the reader.

(3) *Null-space coding* --- As mentioned previously, there are many null-space codes, which are displayed as *nothing*. We can thus insert them *at any position* in a line *for any repetitions* in a data hiding process without causing the reader's notice. For example, under the environment of the UTF-8 standard using IE, the four null-space codes 1C, 1D, 1E, and 1F, as found in this study, are invisible.

(4) *Inserting multiple null-space codes at text line ends* --- We may place null-space codes repetitively at the end of a text line without causing noticeable effect because they are invisible when displayed, as in the case of (2) above.

(5) *Combining techniques of the above* --- We may combine the above techniques in arbitrary ways if both white-space and null-space coding are applicable in the environment.

In the above discussions, we see that the ASCII control codes usable for embedding secret data are *variant* for different kinds of servers, browsers, and character sets. In order to have a systematic investigation in this aspect, in this study we created an email file which includes all ASCII control codes shown in Table 1 to find out SMTP server software suitable for data embedding, as well as the corresponding appearances of the ASCII control codes after they are processed and displayed in the environment of such server software. The investigation results are described as follows.

First, we have found four SMTP email servers which do not change the text contents of emails, and so can be used as *standard* SMTP servers for the purpose of data embedding in this study. Their uniform resource locators (URLs) are http://cis.nctu.edu.tw, http://mis.tsint.edu.tw, http://tw.yahoo.com and http://www.hotmail.com. The first is located in the Department of Computer Science at National Chiao Tung University in Taiwan, with an SMTP software of Twig 2.7.7. The department has additionally another SMTP server system, Horde, for web mails. The second server is located at the Department of Management Information at Technology and Science Institute of Northern Taiwan. The SMTP software is SendMail 8.12. The third server is located in Taiwan and deals with web mails with the name Yahoo! Mail. The last server is Hotmail, a web mail server of Microsoft Corporation. After registering at any of these four servers, a user may read, transmit, or receive emails by Outlook Express or IE.

In this study, the email format we use is MIME 1.0, the content-type is text/plain, and the character set is UTF-8. These formats are very commonly used and so are adopted in this study for data hiding applications.

After a systematic test of the ASCII character set on the above-mentioned four servers, we found that the hexadecimal ASCII control codes appropriate for data embedding *under*

both the Outlook Express and the IE environments are 1C, 1D, 1E, and 1F. These four codes *all* appear to be invisible on the IE browser, and *all* are shown as white spaces in the Outlook Express window. They can so be used for data embedding *respectively* according to the techniques of (2) and (4) mentioned above. However, our goal is to take into account *simultaneously*, instead of respectively, the techniques of (2) and (4), resulting in a method of repeatedly placing these four ASCII control codes *at the ends of email text lines*. The displayed result of the stego-email will be of no difference from the appearance of the original cover email, thus achieving the steganographic effect.

More specifically, we use the following encoding rules to embed secret data into the text line ends of a cover email.

1. Encode 2-bit binary secret data "00," "01," "10," and "11" with the four ASCII codes 1C, 1D, 1E, and 1F, respectively.
2. Put the unique combined ASCII codes 201E in front of a sequence of secret data as its *start signal*, and append another copy of it at the sequence tail as the *end signal*.
3. Use the unique combined ASCII codes 201C to encode the 1-bit data '0,' and the combined codes 201D to encode '1.'
4. Use the unique combined ASCII codes 201F as a *separator* to stop the underline display that starts from a special lexical token of the network protocol, like http, ftp, email, …, etc.

Rule 4 above is necessary because otherwise the extra white-space codes we insert at the end of a text line, when happening to be connected to the end of a network protocol text line, will appear to be *underlined* white spaces, like in

http://cis.nctu.edu.tw ,

which obviously are against the purpose of steganography.

Based on the above rules, we describe the proposed data hiding algorithm for the purpose of covert communication and authentication in the next section.

5. Proposed Data Hiding Method for Emails

We first describe the technique we propose to embed secret data into an email as Algorithm 1 below, and then describe how to transmit the stego-email by Outlook Express or IE. In the following, when we refer to an email, we mean its text body, excluding the header.

Algorithm 1. ***Data embedding in an email.***
Input: a secret data file S and a cover email E long enough to hide S.
Output: a stego-email E'.
Steps:

1. Set the format of the cover email E to MIME 1.0, the content-type to text/plain, and the character set to UTF-8.
2. Fold sequentially each long text line in E with over 65 characters into a 65-character line by inserting a CRLF to replace the first space code 20 *found backward from the 65th character breakpoint* in the line.
3. Check every line in the resulting E to see if there exists in it any special lexical token of the network protocol right before the CRLF; if so, insert a separator code 201F before the CRLF so that we can insert secret data in between the separator code and the CRLF, as described next.
4. Get a text line from E, starting from the first, and perform the following operations.

4.1 Insert the start signal 201E before the CRLF which appears at the line end.

4.2 Compute the *embedding capacity EC* between the start signal and the CRLF in the following way:

$EC = 70 -$ *position of CRLF in text line*,

which means the number of secret data bits we can insert before the CRLF until the line becomes 70 characters long and should not be made longer, as discussed previously.

4.3 Perform one of the following three cases (assuming that $|S|$ means the length of S):

(1) if $EC \neq 0$ and $|S| > 1$, then get a pair of bits from the prefix of S, encode it with the corresponding code (one of 1C, 1D, 1E, 1F), insert the result before the

CRLF, decrement *EC* by 1, decrement |*S*| by 2, and perform Step 4.3 again;

(2) if *EC* = 0 and |*S*| > 1, then get the next text line in *E* and perform Step 4.2;

(3) if |*S*| ≤ 1, then continue.

5. Check *S* to see if there still remains a single bit *B* in *S*. If so, then:
   (1) if *EC* ≠ 0, insert the code 201C before the CRLF if *B* is '0' or the code 201D if *B* is '1';
   (2) if *EC* = 0, then get a text line in *E* with nonzero embedding capacity *EC* and conduct the insertion as in Step 5(1) above.
6. Append the end signal 201E at the end of all the codes inserted in the previous steps.
7. Output the result as the desired stego-email *E'*.

After a stego-mail *E'* is obtained, we want to send it to the receiver site through Outlook Express or IE as a traditional email or a web mail, respectively. For the former way using Outlook Express, we open a new email, denoted as *En*, set the character set of *En* to UTF-8, expand the window size of *En* to the maximum, copy the text body of *E'* into *En*, and finally send the result to the receiver without encrypting it. For the latter way using IE, we use IE to log in the selected web mail server, and do all the same to complete the mail transmission.

6. Proposed Data Recovery Process for Emails

At the receiver end, after a stego-mail is received by the use of Outlook Express or IE, its content of ASCII codes is checked for secret data extraction. The algorithm for this purpose is described as follows.

Algorithm 2. ***Data recovery from a stego-email text body.***

Input: a stego-email text *E'*, presumably including a secret data file *S*.

Output: the file *S*.

Steps:

1. Scan separator signals 201F in *E'* and remove all of them, if there exists any.
2. Scan the resulting *E'* to find the start signal 201E in *E'* and remove it
3. Perform the following steps.
   3.1 Get a pair of ASCII codes in order from *E*.
   3.2 If the code pair *P* is the end signal of 201E, then perform Step 4; otherwise:
      (1) if *P* is either 201C or 201D, then decode *P* to be the bit 0 or 1, respectively;
      (2) if *P* is neither 201C nor 201D, then check each code *Q* in *P* and if *Q* is one of 1C, 1D, 1E, and 1F, then decode *Q* to get the corresponding secret bit pair (one of 00, 01, 10, and 11) and remove *Q*.
   3.3 Go to Step 3.1.
4. Remove the end signal.
5. Concatenate all the decoded secret data bits extracted in the previous steps into a sequence as the desired secret data file *S* and exit.

7. Proposed Authentication Process for Email Documents

The data embedding and extraction techniques proposed previously, in addition to being useful for the purpose of covert communication, may be used for the purpose of email authentication. More specifically, by embedding appropriately-designed codes as an *authentication signal*, the signal, when extracted, can be used to check the *fidelity* of a received email, proving that it was transmitted by a specified server and not tampered with before received. In this study, we achieve this goal by embedding an authentication signal into an email by Algorithm 1 to generate an *authenticable* stego-email. The signal is generated by the use of the content of an email by a division operation. The fidelity verification work is accomplished by matching the authentication signal extracted from a given authenticable stego-email with that computed directly from the original text content of the email. The details are described as two algorithms below.

Algorithm 3. ***Generation of an authenticable email***.

Input: a cover email *E* and a secret key *K*.

Output: an authenticable email *E′*.
Steps:

1. Fold each long text line in *E* with over 65 characters into a 65-character line by inserting a CRLF code to replace the first space code found backward from the 65th character breakpoint.
2. Compute a value *M* by summing up all the ASCII code values in the resulting *E* after excluding all the special codes of 1C, 1D, 1E, 1F, 201C, 201D, 201E, and 201F.
3. Compute an authentication signal *A* as the remainder of dividing *M* by the secret key *K*.
4. Use Algorithm 1 to embed *A* into *E* to obtain an authenticable email as the desired output *E′*.

In Step 2 above, the reason of excluding the special codes is that these codes are to be used for embedding the authentication signal *A* in Step 4.

Algorithm 4. ***Authentication of an email***.
Input: a stego-email *E′*, presumably including an authentication signal; and a secret key *K*.
Output: an authentication message about the fidelity of the displayed text of *E′*.
Steps:

1. Compute a value *M* by summing up all the ASCII code values in *E′* after excluding all the special codes of 1C, 1D, 1E, 1F, 201C, 201D, 201E, and 201F.
2. Compute an authentication signal *A* as the remainder of dividing *M* by the secret key *K*.
3. Extract the hidden authentication signal *A′* from *E′* by Algorithm 2.
4. Compare *A′* with *A*, and if they are identical, then output the authentication message "pass," meaning the displayed text content of *E′* is genuine; else, the message "fail," meaning the reverse.

8. Experimental Results

Figures 1 through 4 illustrate some experimental results of applying Algorithms 1 and 2 for covert communication using Outlook Express. Figure 1 shows part of the content of a 9.3KB cover email. Figure 2 shows part of the content of the stego-email (12.7KB) obtained by applying Algorithm 1 with the cover email as the input. This content was displayed with Outlook Express by a receiver with email address tmp168@mis.tsint.edu.tw, to whom the stego-email was sent. From Figure 2, we see that no difference can be seen in the stego-email, when it is compared with the cover email. Figure 3 shows the content of the 1.07KB secret data file embedded in the stego-email. And Figure 4 shows the content of the 1.07KB secret data file extracted from the stego-email shown in Figure 2 by applying Algorithm 2. The two file contents can be seen to be the same. These results show that the proposed method of data hiding and recovery is feasible.

Figures 5 through 9 illustrate some additional experimental results of applying the proposed algorithms using IE. All password portions in the emails in these figures were blackened for protecting the privacy of the mail owners. Figure 5 shows the content of a 2.42KB cover email. Figure 6 shows the content of the corresponding stego-email (2.54KB) generated by Algorithm 1. Figure 7 shows part of the content of the stego-email seen as a web mail in IE at a receiver site with address gis87809@cis.nctu.edu.tw. Figure 8 shows the content of the original secret data file with 27 bytes. Figure 9 shows the content of the secret data file that was extracted from the stego-email shown in Figure 7. Again, the original and the extracted secret data are seen identical.

The experiments presented above were conducted under the condition that the transmitter's and the receiver's operations were performed on the same server. Actually, we also conducted experiments in which the transmitter's and receiver's operations were performed on difference servers. For example, one server we used was the mail server at Yahoo! in Taiwan, and the other a mail server in the Department of Computer Science at National Chiao Tung University in Taiwan. The results remained unchanged.

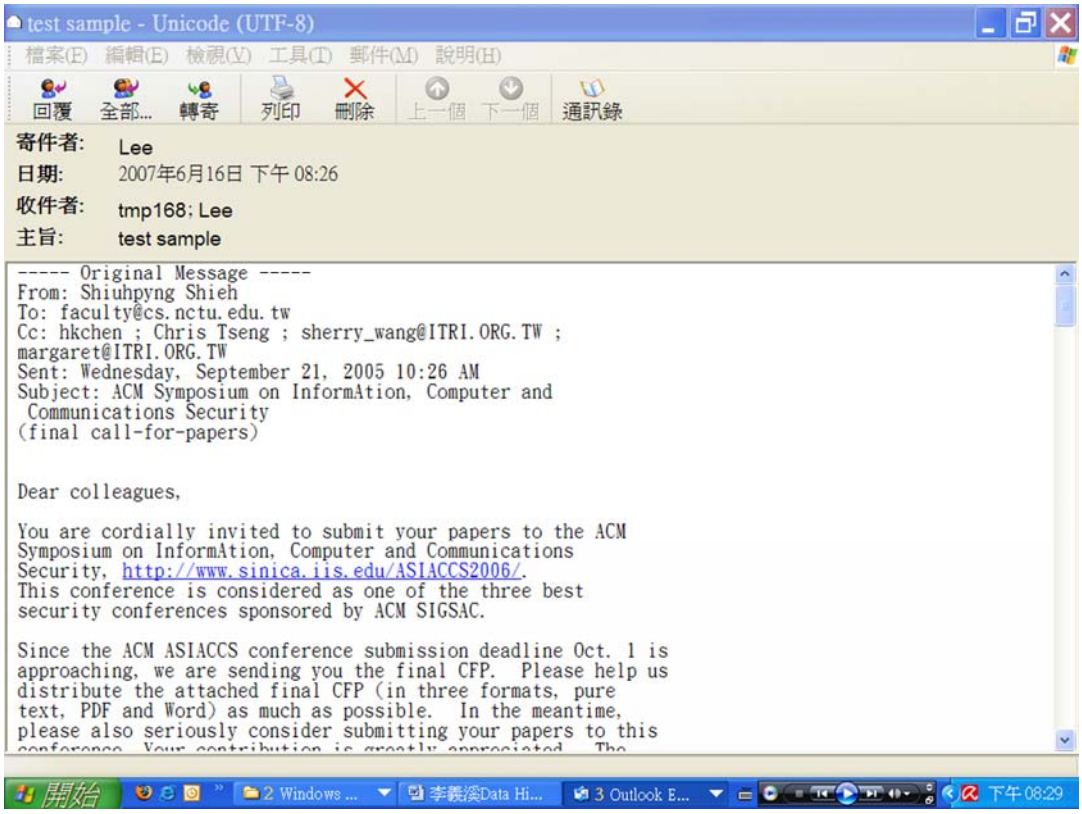

Figure 1. Part content of a cover email.

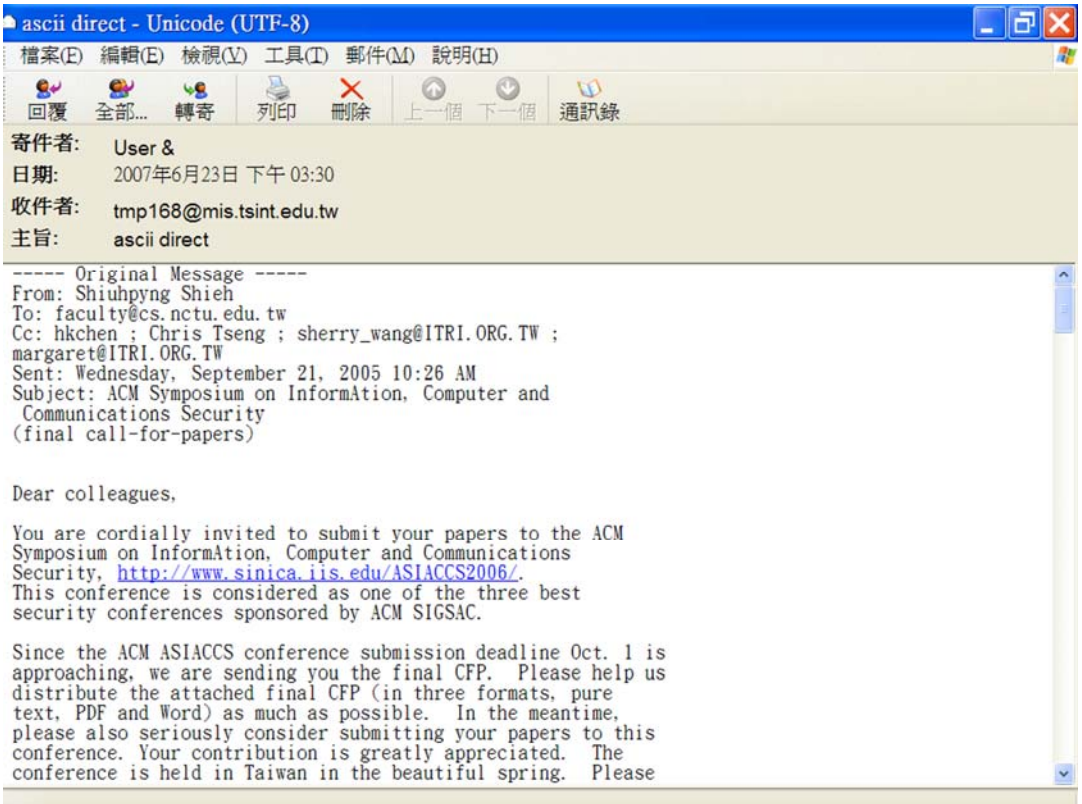

Figure 2. Part content of the stego-email generated from Figure 1.

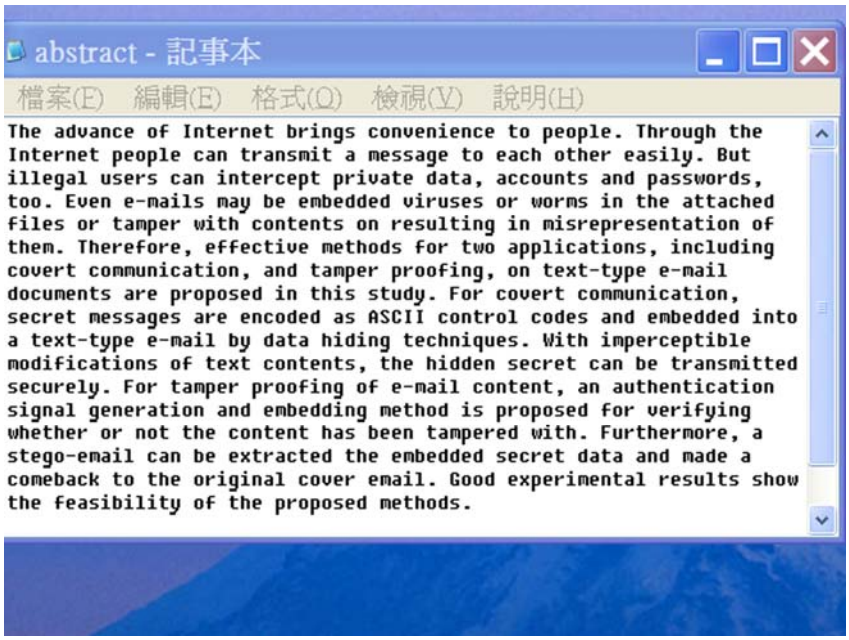

Figure 3. Content of an embedded secret data file.

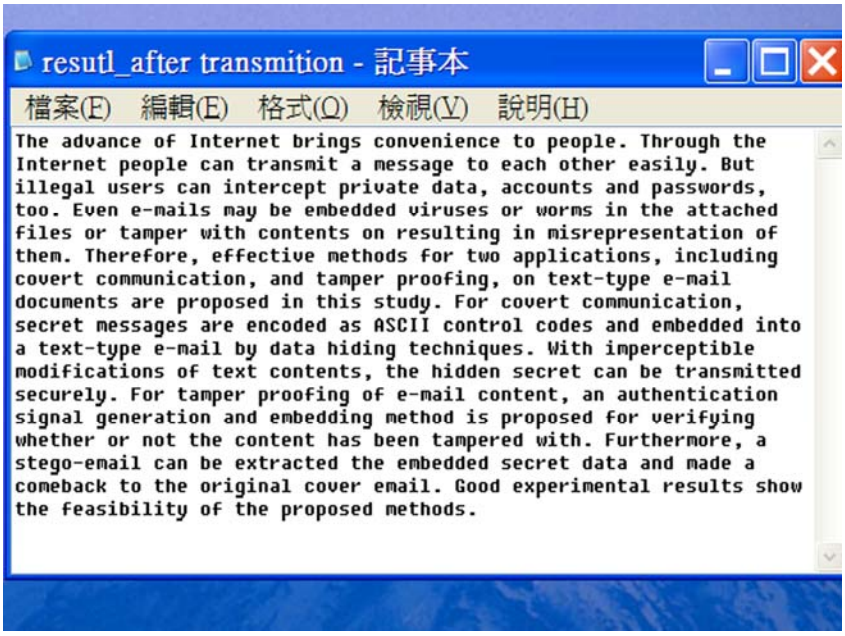

Figure 4. Content of the extracted secret data file.

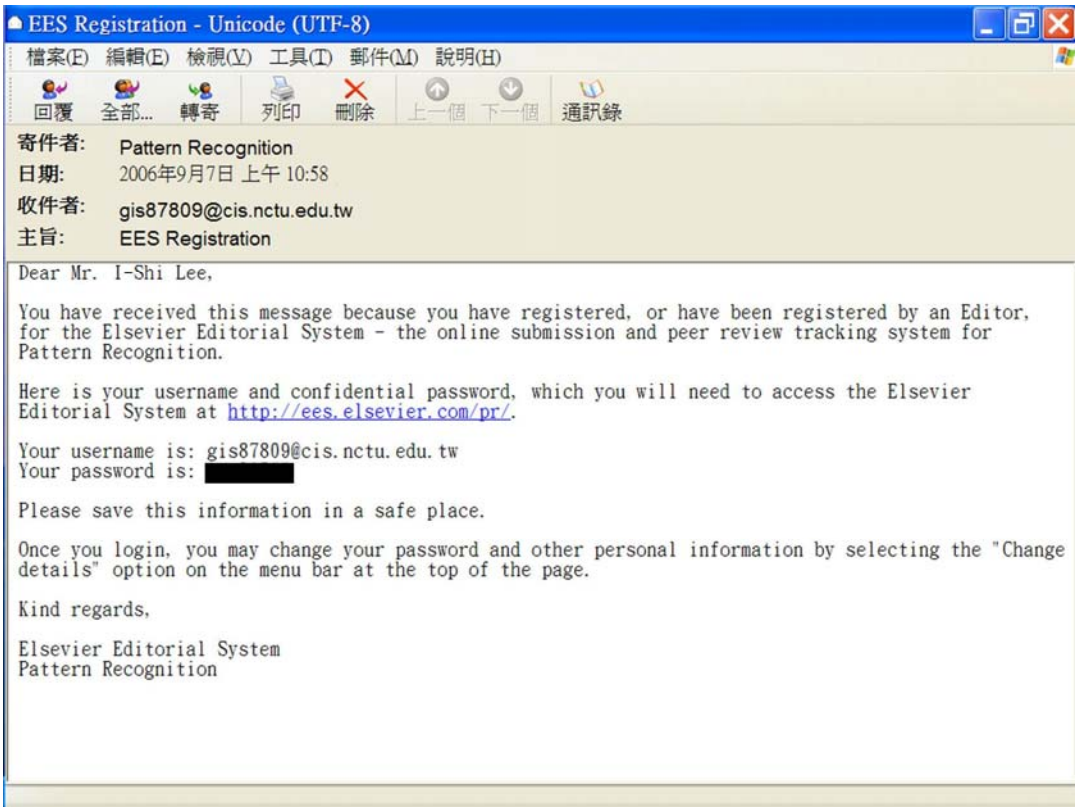

Figure 5. Content of a cover email.

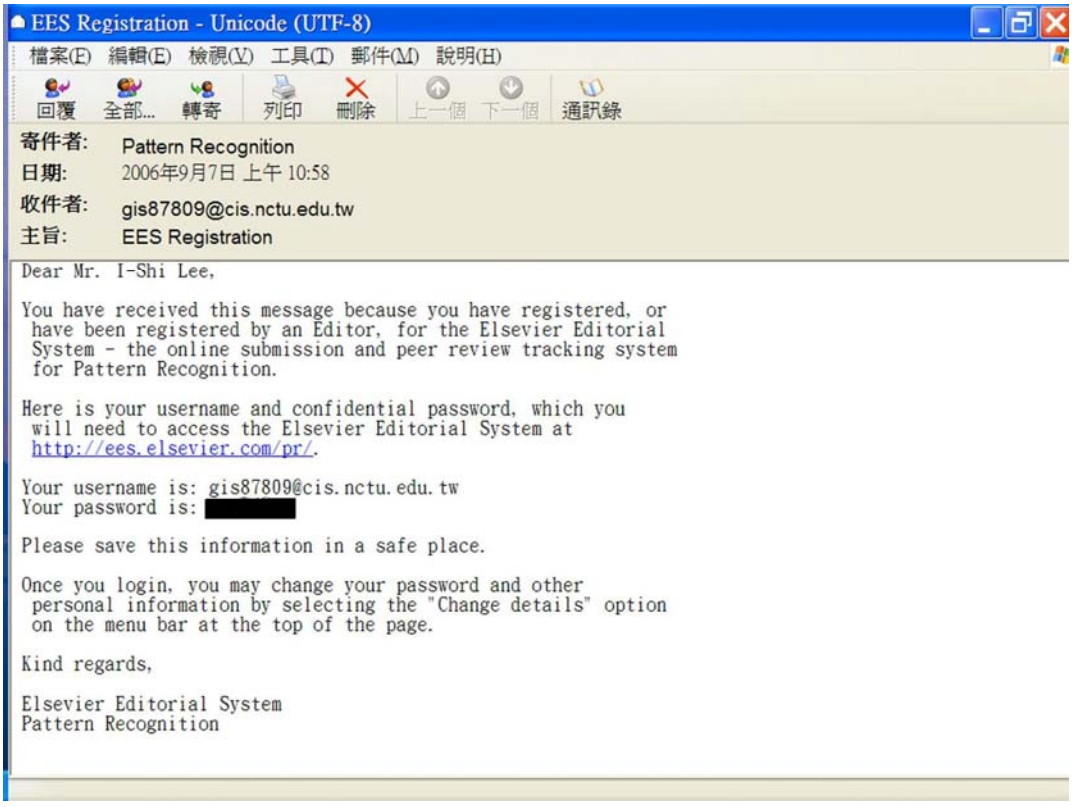

Figure 6. Content of the stego-email generated from Figure 5 before being transmitted.

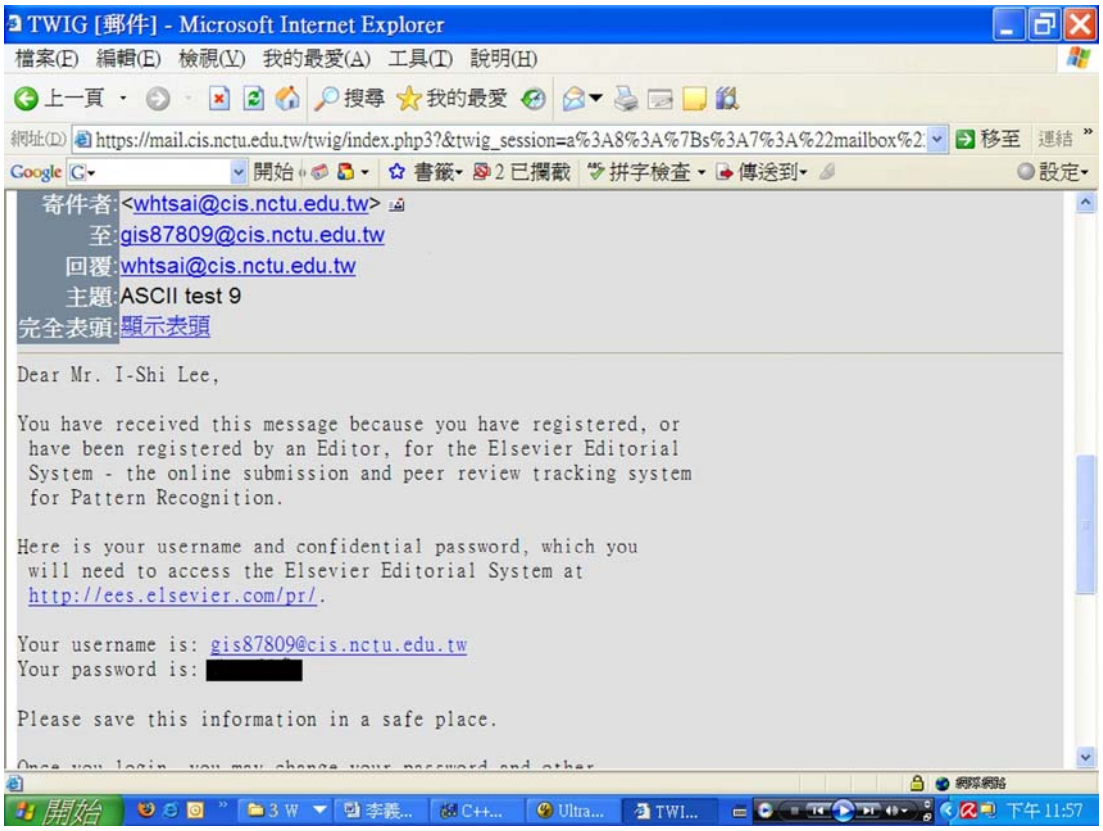

Figure 7. Part content of the stego-email received and displayed in IE.

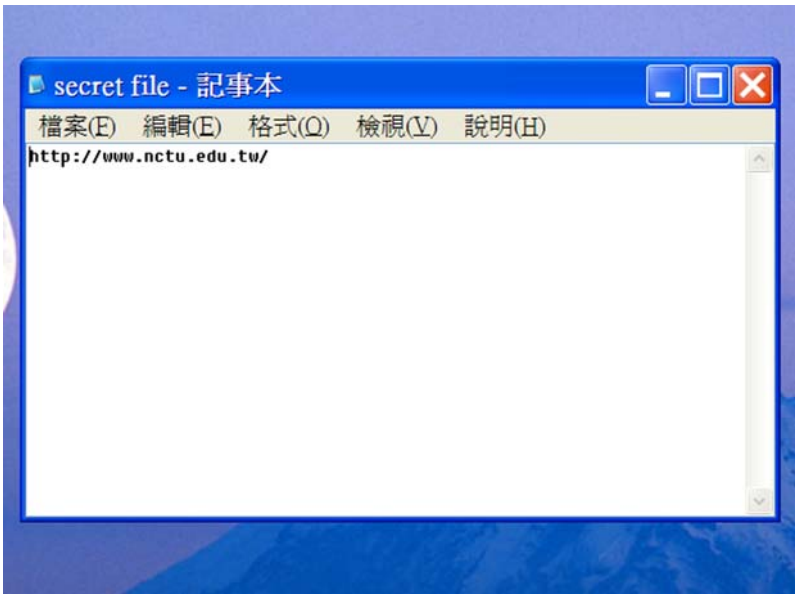

Figure 8. Content of the original secret file.

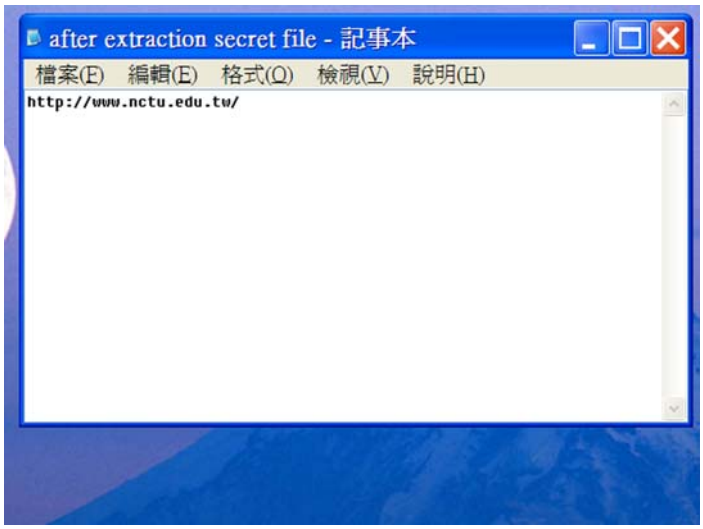

Figure 9. Content of the extracted secret file.

Figures 10 to 13 illustrate some experimental results of applying the proposed email authentication method. Figure 10 shows the content of a stego-email which was generated by Algorithm 3. The password portion in the stego-email was also blackened for protecting privacy. The embedded secret data are invisible to a casual reader. Figure 11 shows part of the content of the stego-email file after being received by Outlook Express, and the authentication result of "pass". Figure 12 shows part of the content of the stego-email after being received by IE, and the authentication result of "pass", too. Figure 13 shows part of the content of the stego-email file after being received by IE, and the authentication result of "fail," since the content has been tampered with (the word "Lee" has been changed to "lee"). These results show that the proposed email authentication method is effective.

9. Conclusions and Discussions

In this study, we propose a method to embed secret data into emails via the use of the ASCII codes under the operating system of the traditional Chinese version of Microsoft Windows XP, service pack 2, 2002. After a systematic test of all the ASCII codes on various email server software systems and standards, four special ASCII control codes 1C, 1D, 1E, and 1F have been found to be invisible at the line ends of email texts on the SMTP email server in the environment of Outlook Express or IE. A technique has been proposed to utilize these special codes to encode secret data, which is a combination of five coding rules found in this study. Each stego-email can be transmitted to a receiver, and read as a normal email. Extra long lines of emails are folded to be of a proper length for normal displays on email servers to increase steganographic effects. The experiment results prove the feasibility of the proposed method.

In this study, 2-bit secret data are embedded into a white space of a text email. Comparing to other methods proposed by Bender et al. [2] and Chang and Tsai [11] in which on average each secret bit needs 1.5 white spaces to encode (one white space representing a "0," and two white spaces representing a "1," leading to the average of $1\times0.5+2\times0.5 = 1.5$ spaces for a secret bit), the proposed method needs only 0.5 white space for each secret bit (one ASCII code representing 2 secret bits), which is an increase of the embedding capacity for three times.

The proposed methods may put into practice in the four servers as listed previously. However, not all mail servers fully follow the SMTP standard. Instead, some mail servers have their own ways of management, like Gmail and Yahoo! Mail, which delete redundant spaces and undefined characters. So, the

proposed method is inapplicable to these two servers. Other applicable techniques should be investigated, and are left for further study. Another topic worth future investigation is to apply the proposed data hiding technique to check the integrity of an email, in addition to the fidelity check scheme proposed in this study. Finally, we may extend both the convert communication and authentication works of this study to dealing with web pages.

Acknowledgement

This work was supported partially by the NSC Advanced Technologies and Applications for Next Generation Information Networks (II) – Sub-project 5: Network Security, Project No. NSC-94-2752-E-007-001-PAE. And partially by the NSC project NSC94-2422-H-468-001.

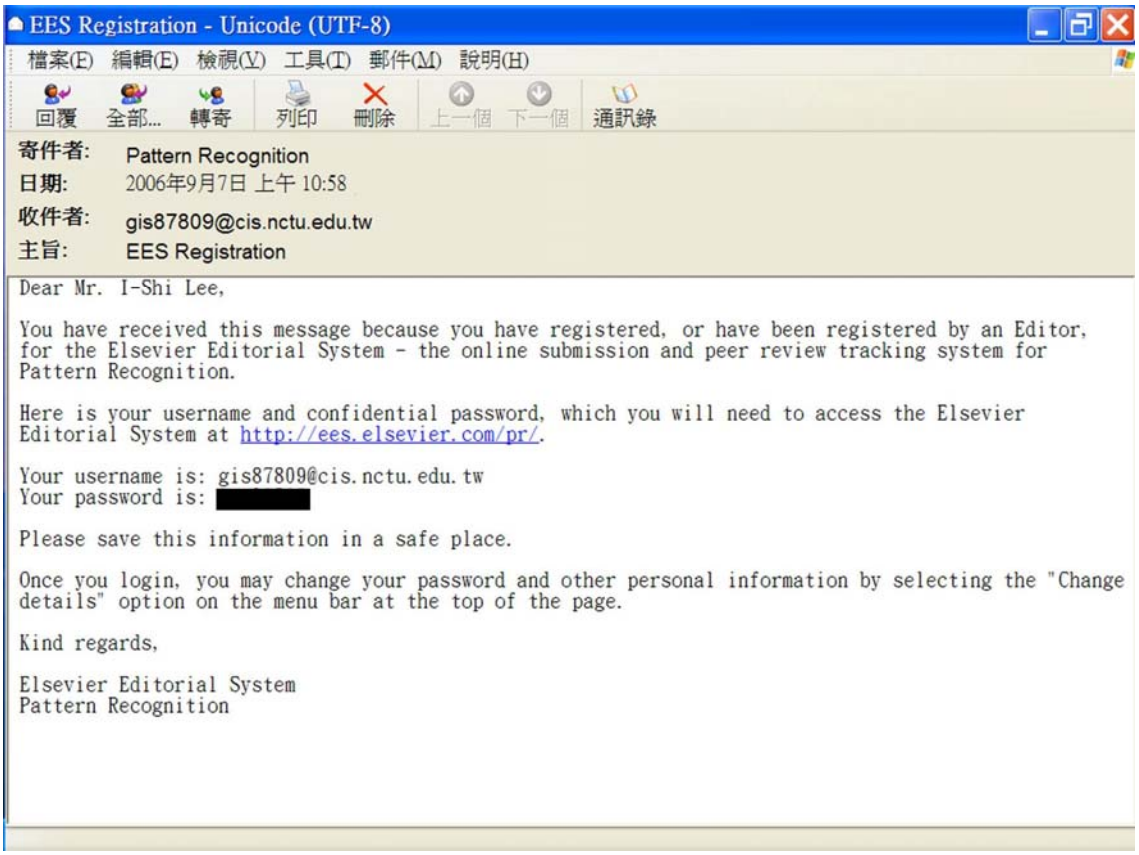

Figure 10. Content of a stego-email for authentication before transmission.

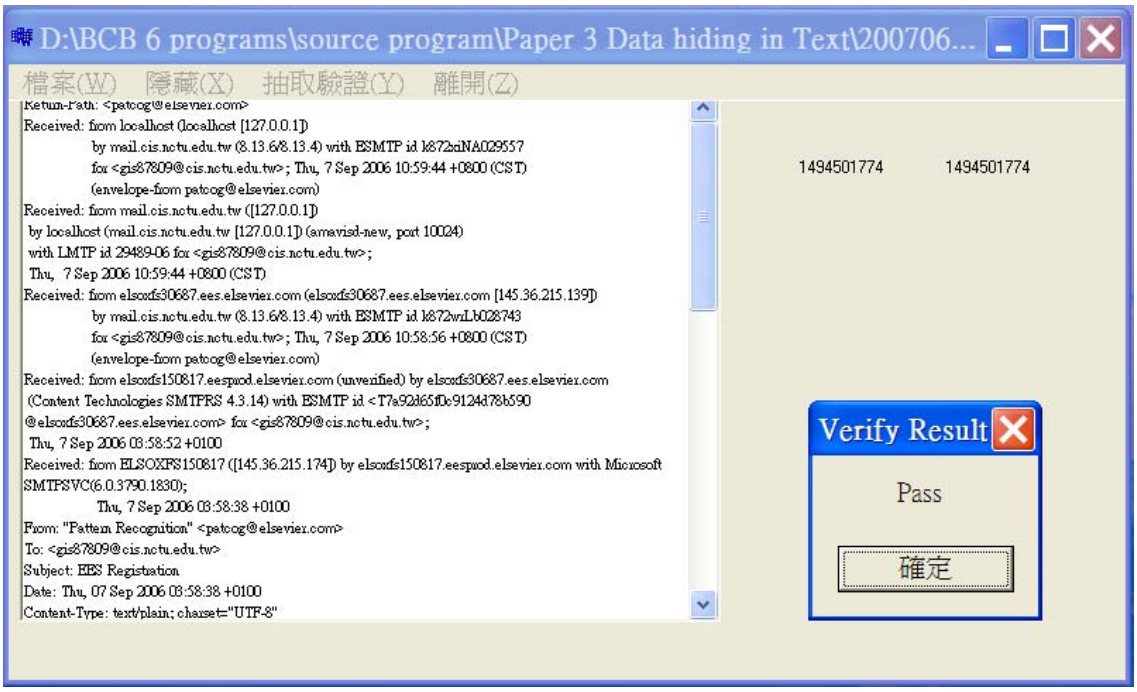

Figure 11. Authentication result of “pass” after receiving a stego-email by Outlook Express.

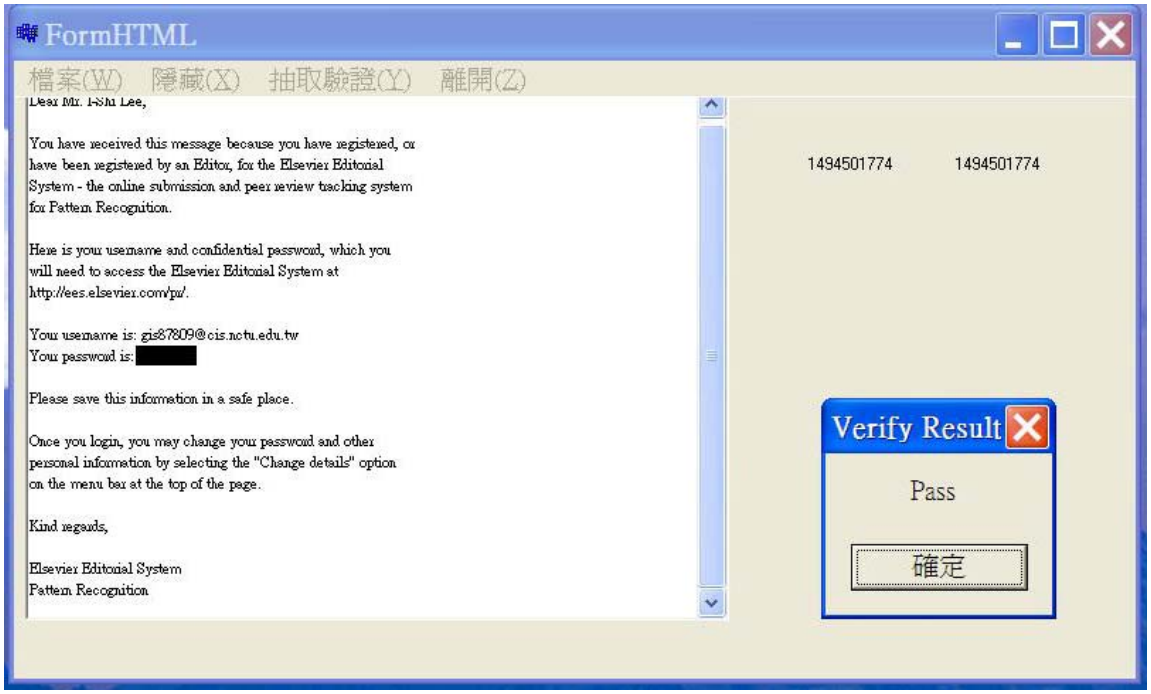

Figure 12. Authentication result of “pass” after receiving a stego-email by IE.

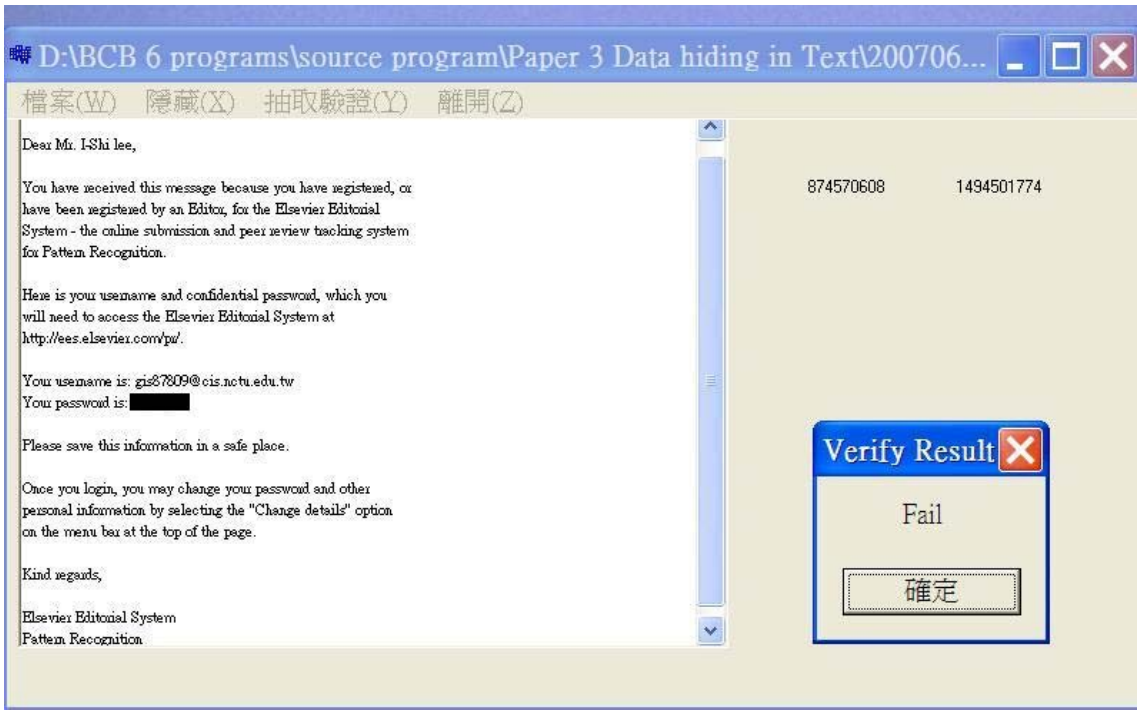

Figure 13. Authentication result of "fail" after receiving the stego-email by IE. The word "Lee" in the content has been modified to be "lee"

References

[1] D. C. Lou and J. L. Liu, "Steganographic Method for Secure Communications," *Computers and Security*, Vol. 21, No. 5, pp. 449-460, Oct. 2002.

[2] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM System Journal*, Vol. 35, No. 3 & 4, Feb. 1996.

[3] S. Katzenbeisser and F. A. P. Petitolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, MA, USA, 2000.

[4] Low, S. H., N. F. Maxemchuk, and A. M. Lapone, "Document Identification for Copyright Protection Using Centroid Detection," *IEEE Transactions on Communication*, Vol. 46, No. 3, pp. 372-383, 1998.

[5] J., Brassil, N. F. Maxemchuk, and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying," *Proceedings of 13th Annual IEEE Conference on Computer Communications(INFOCOM'94),* pp.1278-1287, 1994.

[6] P. Wayner, "Strong Theoretical Steganography," *Cryptologia*, Vol. XIX/3, pp. 285-299, 1995.

[7] G. Cantrell and D. D. Dampier, "Experiments in Hiding Data Inside the File Structure of Common Office Documents: A Stegonography Application," *Proceedings of 2004 International Symposium on Information and Communication Technologies*, Las Vegas, Nevada, USA, pp. 146-151, 2004.

[8] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking-Attacks and Countermeasures*, Kluwer Academic Publishers, Boston, MA, USA, 2001.

[9] R., Anderson, R. Needham, and A. Shamir, "The Steganographic file System," *Information Hiding: Second International Workshop*, Portland, Oregon, USA, 1998.

[10] T. G., Handel, and M. T. Stanford, "Hiding Data in the OSI Network Model," *Proceedings of First International Work-*

shop on Information Hiding, Cambridge, UK, pp. 23-38, 1996.
[11] Y. H. Chang and W. H. Tsai, “A steganographic method for copyright protection of HTML documents,” *Proceedings of 2003 National Computer Symposium*, Taichung, Taiwan, Dec. 2003.
[12] J. B. Postel, "Simple Mail Transfer Protocol," *STD 10, RFC 821, IETF*, August 1982.
[13] D. Crocker, "Standard for the Format of ARPA Internet Text Messages," *STD 11, RFC 822, IETF*, August 1982.
[14] P. Resnick, "Internet Message Format," *RFC 2822, IETF*, April 2001.
[15] J. G. Myers and M. T. Rose, “Standard Post Office Protocol - Version 3,” *STD 53, RFC 1939, IETF*, May 1996.
[16] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," *RFC 2045, IETF*, Nov. 1996.
[17] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," *RFC 2046, IETF*, Nov. 1996.
[18] N, Freed. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part 5: Conformance Criteria and Examples," *RFC 2049, IETF*, Nov. 1996.
[19] M. Crispin, " Internet Message Access Protocol - Version 4rev1," *RFC 3501, IETF*, March 2003