# Security Protection of Software Programs by Information Sharing and Authentication Techniques Using Invisible ASCII Control Codes

I-Shi Lee[1,3] and Wen-Hsiang Tsai[1,2]

*(Corresponding author: I-Shi Lee)*

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30010[1]

Department of Information Communication, Asia University, Taichung, Taiwan 41354[2]

Department of Management Information, Technology and Science Institute of Northern Taiwan, Taipei, Taiwan[3]

(Email: {gis87809, whtsai}@cis.nctu.edu.tw)

*(Received June 19, 2008; revised and accepted July 4, 2008)*

## Abstract

A new method for software program protection by information sharing and authentication techniques using invisible ASCII control codes is proposed. A scheme for sharing a secret source program written in Visual C++ among a group of participants, each holding a camouflage program to hide a share, is first proposed for safe keeping of the secret program. Only when all the shares hidden in the camouflage programs are collected can the secret program be recovered. The secret program, after being exclusive-ORed with all the camouflage programs, is divided into shares. Each share is encoded next into a sequence of special ASCII control codes which are invisible when the codes are inserted in the comment of the Visual C$^{++}$ program and viewed in the window of the Microsoft VC$^{++}$ editor. These invisible codes then are hidden in the camouflage program, resulting in a stego-program for a participant to keep. Each stego-program can still be compiled and executed to perform the original function of the camouflage program. A secret program recovery scheme is also proposed. To enhance security under the assumption that the sharing and recovery algorithms are known to the public, three security measures via the use of a secret random key are also proposed, which not only can prevent the secret program from being recovered illegally without providing the secret key, but also can authenticate the stego-program provided by each participant, during the recovery process, by checking whether the share or the camouflage program content in the stego-program have been tampered with incidentally or intentionally. Experimental results show the feasibility of the proposed method.

*Keywords: Authentication, camouflage program, information sharing, invisible ASCII control codes, program sharing, secret program, security protection, software program, source program, stego-program*

## 1 Introduction

Software programs written in various computer languages are important resources of intellectual properties. They need protection from being tampered with. One technique of information protection is *information sharing* [7]. When applied to software programs, this technique means that a secret program is, via a certain sharing scheme, transformed into several copies, called *shares*. Each share is individually different from the original secret program in appearance, content, and/or function. The secret program cannot be recovered unless the shares are collected and manipulated with a reverse sharing scheme. Such a technique of program sharing may be regarded as one way of *secret keeping*, which is necessary in many software-developing organizations.

The concept of secret sharing was proposed first by Shamir [8]. By a so-called $(k, n)$-threshold scheme, the idea is to encode a secret data item into $n$ shares for $n$ participants to keep, and any $k$ or more of the shares can be collected to recover the original secret, but any $(k-1)$ or fewer of them will gain no information about it. A similar scheme, called visual *cryptography*, was proposed by Naor and Shamir [6] for sharing an image. The scheme provides an easy and fast decryption process consisting of xeroxing the shares onto transparencies and stacking them to reveal the original image for visual inspection. This technique has been investigated further in [1, 2, 5], though it is suitable for binary images only. Verheul and van Tilborg [9] extended the visual cryptography technique for processing images with small numbers of gray levels or colors. Lin and Tsai [4] proposed a digital version of the visual cryptography scheme for color images with no limit on the number of colors. The $n$ shares obtained from a color image are hidden in $n$ camouflage images which may be selected to have well-known contents, like famous characters or paintings, to create addi-

Table 1: ASCII control codes and descriptions

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

tional steganographic effects for security protection of the shares.

Sharing of software programs *in source form* has not been studied yet. In this paper, we propose a method for this purpose, which is based on the use of some specific ASCII control codes *invisible* in certain software editors. Invisibility of such ASCII control codes is a finding of this study through a systematic investigation of the visibility of all the ASCII codes in the window of the Visual $C^{++}$ editor of Microsoft Visual Studio .NET 2003, Service Pack 1 (abbreviated as the $VC^{++}$ editor in the sequel). By the use of the logic operation of "exclusive-OR," each source program to be shared is transformed into a number of shares, say $N$ ones, which are then hidden respectively into $N$ pre-selected *camouflage source programs*, resulting in $N$ *stego-programs*. Each stego-program still can be compiled and executed to perform the function of the original camouflage program, and each camouflage program may be selected arbitrarily, thus enhancing the steganographic effect.

To improve the security protection effect further, we propose additionally an authentication scheme for verifying the correctness of the contents of the stego-programs brought by the participants to join the process of secret program recovery. This is advantageous to prevent any of the participants from accidental or intentional provision of a false or destructed stego-program. The verified contents include the share data and the camouflage program contained in each stego-program. Any "bad" share or camouflage program will be identified and picked out in the secret program recovery process. This double capability of authentication is based on the use of certain *authentication signals* embedded in the stego-programs. Each signal is generated from the contents of the share data and the camouflage program content. A third measure proposed to enhance security protection in this study is to prohibit recovery of the secret program with *illegally* collected stego-programs. All of these protection capabilities are carried out with the provision of a secret random key through the use of certain mathematical operations.

In the remainder of this paper, we describe in Section 2 the finding of the invisible ASCII codes and a scheme of binary data encoding into such codes for use in generating stego-programs. In Section 3, an algorithm describing the proposed source program sharing and authentication signal generation schemes is presented, and in Section 4, an algorithm for stego-program authentication and secret source program recovery is described. And finally in Section 5, some experimental results are presented, followed by a conclusion in Section 6.

## 2 Invisible ASCII Control Codes for Binary Data Encoding

ASCII codes, usually expressed as hexadecimal numbers, are used very commonly to represent texts for information interchanges on computers. Some of the ASCII codes of 00 through 1F were used as *control codes* to control computer peripheral devices like printers, tape drivers, teletypes, etc. (see Table 1). But now they are rarely used for their original purposes because of the rapid development of new peripheral hardware technologies, except those codes for text display controls, such as 0A and 08 with the meanings of "line feed" and "backspace," respectively. It is found in this study that some of the ASCII control codes, when displayed by certain text editors under some OS environments, are *invisible*. Such ASCII codes may be utilized for various secret data hiding purposes [3].

Table 2: Invisible character coding table

| Bit pair | Corresponding invisible ASCII code |
|---|---|
| 00 | 1C |
| 01 | 1D |
| 10 | 1E |
| 11 | 1F |

The finding of such invisible codes resulted from a systematic test conducted in this study, in which all the ASCII control codes in the environment of the VC$^{++}$ editor of Microsoft Visual Studio .NET 2003, Service Pack 1 were inspected one by one. Four of such codes so found are 1C, 1D, 1E, and 1F, which are invisible in the *comments* or *character strings* of VC$^{++}$ programs (see Table 2). Such codes will simply be said *invisible* in subsequent discussions.

As an illustrative example, in Figure 1 we show a simple source program in Figure 1(a) with a short comment "test a file." In the comment, we inserted consecutively the four codes 1C, 1D, 1E, and 1F between the letters "s" and "t" in the word "test." Their existences can be checked with the text editor UltraEdit 32, as can be seen from Figure 1(b). But the four codes are invisible in the VC$^{++}$ editor, as can be seen from Figure 1(a). Such invisibility usually will arouse no suspicion from common program developers and so achieve a steganographic effect, since, unless necessary, a programmer will always use the VC$^{++}$ editor for program inspection and development. We utilize such an "invisibility phenomenon" for hiding both share data and authentication signals in source programs in this study, as described in the following.

For the purpose of program sharing among several participants, after a given secret source program is transformed into shares, each share is transformed further into a string of the above-mentioned invisible ASCII control codes, which is then embedded into a corresponding camouflage source program held by a participant. And for the purpose of security protection, authentication signals, after generated, are transformed as well into invisible ASCII control codes before embedded. These two data transformations are based on a binary-to-ASCII mapping proposed in this study, which is described as a table as shown in Table 2, called *invisible character coding table* by regarding each ASCII code as a character.

Specifically, after the share and the authentication signal data are transformed into binary strings, the bit pairs 00, 01, 10, and 11 in the strings are encoded into the hexadecimal ASCII control codes 1C, 1D, 1E, and 1F, respectively. To promote security, a secret random key is also used in generating the authentication signal and in protecting the generated shares. The details are described in the next section.

# 3 Proposed Program Sharing Scheme

In the sequel, by a program we always mean a source program. A sketch of the proposed process for sharing a secret program is described as follows, in which the used symbols are in Table 3:

- *Creating shares*: Apply exclusive-OR operations to the contents of the secret program, all the camouflage programs, and the secret key $Y$, and divide the resulting string into $N$ segments as shares, with the one for the $k$-th participant to keep being $E_k$.
- *Generating authentication signals*: For each camouflage program $P_k$, use the random key value $Y$ to compute two modulo-$Y$ values from the binary values of the contents of $P_k$ and $E_k$, respectively; and concatenate them as the authentication signal $A_k$ for $P_k$.
- *Encoding and hiding shares and authentication signals*: Encode $E_k$ and $A_k$ respectively into invisible ASCII control codes by the invisible character coding table (Table 2) and hide them evenly at the right sides of all the characters of the comments of camouflage program $P_k$, resulting in a stego-program for the $k$-th participant to keep.

A detailed algorithm for the above scheme is given in the following. Given two ASCII characters $C$ and $D$, each with 8 bits, denoted as $C = c_0c_1 \ldots c_7$ and $D = d_0d_1 \ldots d_7$, we define the result of "exclusive-ORing" the two characters as $E = C \oplus D = e_0e_1 \ldots e_7$ with $e_i = c_i \oplus di$ for $i = 0, 1, \ldots, 7$, where $\oplus$ denotes the bitwise exclusive-OR operation. Note that $E$ has eight bits, too. And given two equal-lengthed character strings $S$ and $T$, we define the result of exclusive-ORing them, $U = S \oplus T$, as that resulting from exclusive-ORing the corresponding characters in the two strings.

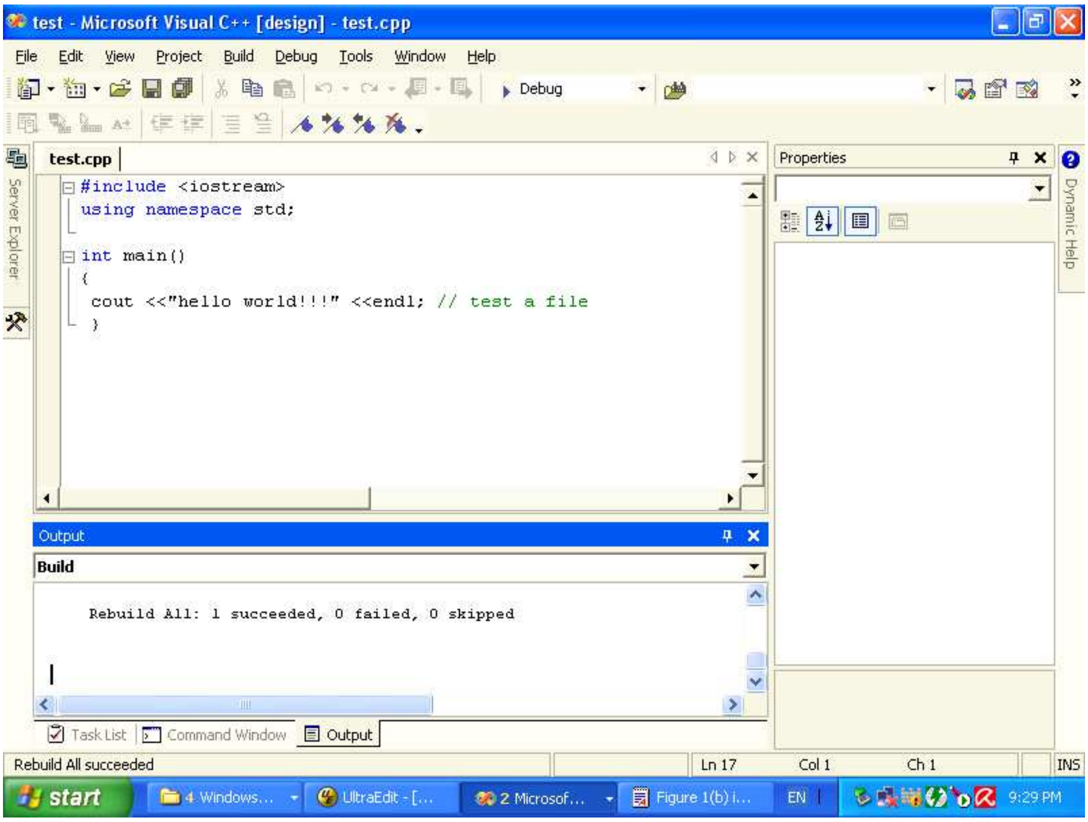
**Algorithm 1: Program sharing and authentication.**

**Input.** (1) a secret program $P_s$ of length $l_s$; (2) $N$ pre-selected camouflage programs $P_1, P_2, \ldots, P_N$ of lengths $l_1, l_2, \ldots, l_N$, respectively; and (3) a secret key $Y$ which is a random binary number with length $l_Y$ (in the unit of bit).

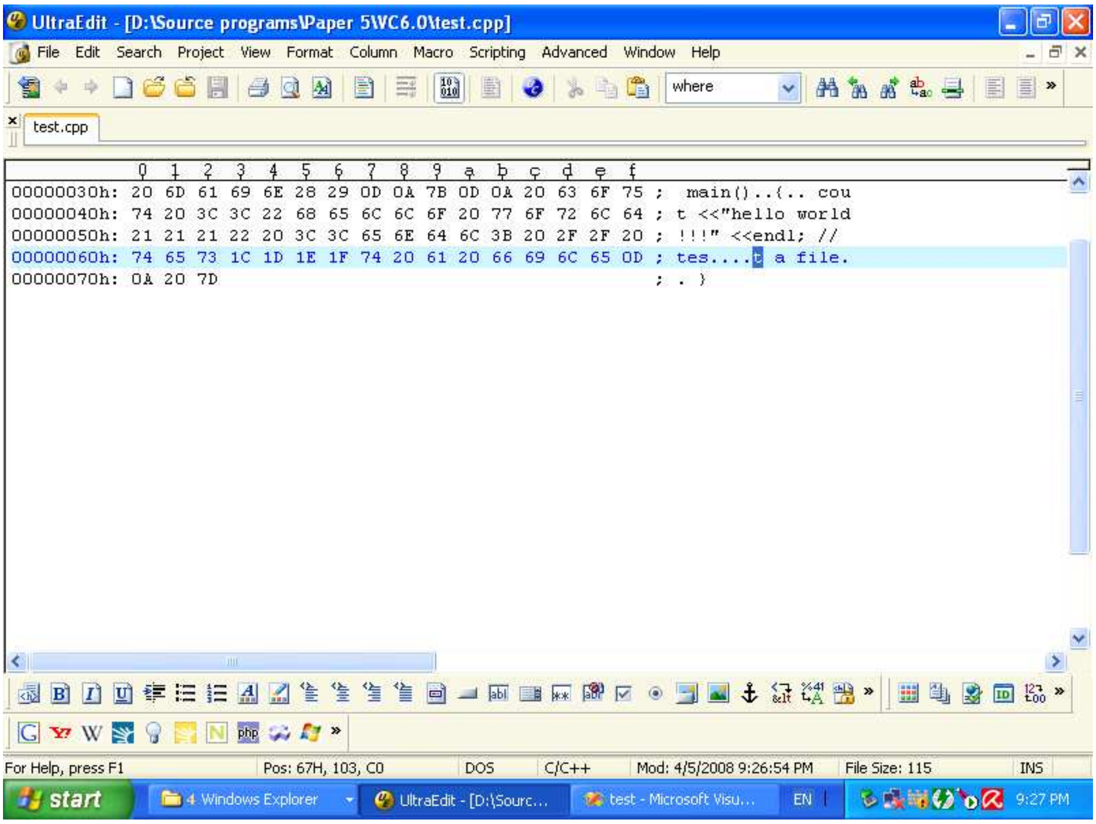
**Output.** $N$ stego-programs, $P'_1, P'_2, \ldots, P'_N$, in each of which a share and an authentication signal are hidden.

**Steps. Stage 1.** Creating shares from the secret program.

1) Create $N + 2$ character strings, all of the length $l_s$ of $P_s$, from the secret program and the camouflage programs in the following way.




(a) A source program with four invisible ASCII control codes inserted in the comment “test a file.”




(b) The program seen in the window of the text editor UltraEdit with the four ASCII control codes visible between the letters “s” and “t” of the word “test” in the comment.

Figure 1: Illustration of invisible ASCII control codes in a comment of a source program.

Table 3: Symbol notation

| | |
|---|---|
| $N$ | the number of participants in the secret program sharing activity; |
| $Y$ | the input secret random key; |
| $P_k$ | a camouflage program for the $k-th$ participant to keep where $k = 1, 2, \ldots, N$; |
| $E_k$ | a share which is embedded in $P_k$; |
| $P_s$ | a secret program; |
| $A_k$ | the generated authentication signal for $P_k$; |
| $P'_k$ | a stego-program which is the result of embedding $E_k$ in $P_k$; |
| $S_s$ | the character string of $P_s$; |
| $S_1, S_2, \ldots, S_N$ | the character string of $P_1, P_2, \ldots, P_N$ respectively; |
| $l_s$ | the length of $S_s$ (in the unit of ASCII character); |
| $l_1, l_2, \ldots, l_N$ | the length of $S_1, S_2, \ldots, S_N$ respectively (in the unit of ASCII character); |
| $l_Y$ | the length of $Y$ (in the unit of bit). |

a. Scan the characters (including letters, spaces, and ASCII codes) in the secret program $P_s$ line by line, and concatenate them into a character string $S_s$.

b. Do the same to each camouflage program $P_k$, $k = 1, 2, \ldots, N$, to create a character string $S_k$ of length $l_s$ (not $l_k$) either by discarding the extra characters in $P_k$ if $l_k > l_s$ or by repeating the characters of $P_k$ at the end of $S_k$ if $l_k < l_s$, when $l_k \neq l_s$.

c. Repeat the key $Y$ and concatenate them until the length of the expanded key $Y'$ in the unit of character (8 bits for a character) is equal to $l_s$, the length of $S_s$.

2) Compute the new string $E = S_s \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y'$.

3) Divide $E$ into $N$ equal length segments $E_1, E_2, \ldots, E_N$ as shares.

**Stage 2.** Generating authentication signals from the contents of the shares and the camouflage programs.

1) Generate an authentication signal $A_k$ for each camouflage program $P_k$, $k = 1, 2, \ldots, N$, using the data of $S_k$ and $E_k$ in the following way.

a. Regarding $S_k$ as a sequence of 8-bit integers with each character in $S_k$ being composed of 8 bits, compute the sum of the integers, take the modulo-$Y$ value of the sum as $A_{S_k}$, transform $A_{S_k}$ into a binary number, and adjust its length to be $l_Y$, the length of the key $Y$, by padding leading 0's if necessary.

b. Do the same to $E_k$ to obtain a binary number $A_{E_k}$ with length $l_Y$, too.

c. Concatenate $A_{S_k}$ and $A_{E_k}$ to form a new binary number $A_k$ with length $2l_Y$ as the authentication signal of $P_k$.

**Stage 3.** Encoding and hiding the share data and authentication signals.

1) For each camouflage program $P_k, k = 1, 2, \ldots, N$, perform the following tasks.

a. Concatenate the share $E_k$ and the authentication signal $A_k$ as a binary string $F_k$.

b. Encode every bit pair of $F_k$ into an invisible ASCII control code according to the invisible coding table (Table 2), resulting in a code string $F'_k$.

c. Count the number $m$ of characters in all the comments of $P_k$.

d. Divide $F'_k$ evenly into $m$ segments, and hide them in order into $P_k$, with each segment hidden to the right of a character in the comments of $P_k$.

2) Take the final camouflage programs $P'_1$, $P'_2$, $\cdots$, $P'_N$ as the output stego-programs.

In Step 3 of the above algorithm, we assume that the number of characters in the secret program is a multiple of N, the number of participants, for simplicity of algorithm description; if not, it can be made so by appending a sufficient number of blank spaces at the end of the original secret program. In Steps 1.a and 1.b of Stage 2, the purpose we compute the signals $A_{S_k}$ and $A_{E_k}$ from the contents of the camouflage program $P_k$ and the share $E_k$, respectively, for use in generating the authentication signal $A_k$ is to prevent any participant from intentionally or accidentally changing the contents of the original camouflage program or the hidden share; illegal tampering with them will be found out in the process of secret program recovery described in the next section. It is also noted that each stego-program yielded by the algorithm still can be compiled and executed to perform the function of the original camouflage program.

# 4 Secret Program Recovery Scheme

A sketch of the proposed process for recovering the secret source program is described as follows, for which it is assumed that the stego-program brought to the recovery activity by participant $k$ is denoted as $P'_k$. Also, the original key with value $Y$ used in Algorithm 1 is provided.

1) *Extracting hidden shares and authentication signals*: Scan the comments of each stego-program $P'_k$ to collect the invisible ASCII control codes hidden in them and concatenate the codes as a character string; decode the string into a binary one by the invisible character coding table (Table 2); and divide the string into two parts, the share data $E_k$ and the authentication signal $A_k$. Also, remove the hidden codes from $P'_k$ to get the original camouflage program $P_k$.

2) *Authenticating the shares and the camouflage programs*: Use the authentication signal $A_k$ as well as the key $Y$ to check the correctness of the contents of the extracted share data $E_k$ and the camouflage program $P_k$ by decomposing $A_k$ into two signals and matching them with the modulo-$Y$ values of the binary values of $P_k$ and $E_k$, respectively. Issue warning messages if either or both authentications fail.

3) *Recovering the secret program*: Apply exclusive-OR operations to the extracted share data $E_1$ through $E_N$, the same secret key $Y$ as that used in Algorithm 1, and the camouflage programs $P_1$ through $P_N$ to reconstruct the secret program $P_s$.

A more detailed secret program recovery process is described as an algorithm in the following.

**Algorithm 2. Authentication of the stego-programs and recovery of the secret program.**

**Input.** $N$ stego-programs $P'_1$, $P'_2$, $\cdots$, $P'_N$ provided by the $N$ participants and the secret key $Y$ with length $l_Y$ used in secret program sharing (Algorithm 1).

**Output.** the secret program $P_s$ hidden in the $N$ stego-programs if the shares and the camouflage programs in the stego-programs are authenticated to be correct.

**Steps. Stage 1.** Extracting hidden shares and authentication signals.

1) For each stego-program $P'_k$, $k = 1, 2, \ldots, N$, perform the following tasks to get the contents of the camouflage programs and the authentication signals.
   a. Scan the comments in $P'_k$ line by line, and collect the invisible ASCII codes located to the right of the comment characters as a character string $F'_k$.
   b. Remove all the collected characters of $F'_k$ from $P'_k$, resulting in a program $P_k$ with length $l_k$, which presumably is the original camouflage program.
   c. Decode the characters in $F'_k$ using the invisible character coding table (Table 2) into a sequence of bit pairs, denoted as $F_k$.
   d. Regarding $F_k$ as a binary string, divide it into two segments $E_k$ and $A_k$ with the length of the latter being fixed to be $2l_Y$, which presumably are the hidden share and the authentication signal, respectively.
   e. Divide $A_k$ into two equal-lengthed binary numbers $A_{S_k}$ and $A_{E_k}$.

**Stage 2.** Authenticating share data and camouflage programs.

1) Concatenate all $E_k$, $k$ =1, 2, . . . , $N$, in order, resulting in a string $E$ with length $l_E$ which presumably equals $l_s$, the length of the secret program to be recovered.
2) For each $k$ =1, 2, . . . , $N$, perform the following authentication operations.
   a. Create a character string $S_k$ of length $l_E$ from the characters in $P_k$ either by discarding extra characters in $Pk$ if $l_k > l_E$ or by repeating the characters of $P_k$ at the end of $S_k$ if $l_k < l_E$, when $l_k \neq l_E$.
   b. Regarding $S_k$ as a sequence of 8-bit integers with each character in $S_k$ composed of 8 bits, compute the sum of the integers, take the modulo-Y value of the sum as $A'_{S_k}$, transform $A'_{S_k}$ into a binary number, and adjust its length to be $l_Y$, the length of the key $Y$, by padding leading 0's if necessary.
   c. Do the same to $E_k$, resulting in a binary number $A'_{E_k}$.
   d. Compare $A'_{S_k}$ with the previously extracted $A_{S_k}$; if mismatching, issue the message "the camouflage program is not genuine," and stop the algorithm.
   e. Compare $A'_{E_k}$ with the previously extracted $A_{E_k}$; if mismatching, issue the message "the share data have been changed," and stop the algorithm.

**Stage 3.** Recovering the secret program.

1) Repeat the key $Y$ and concatenate them until the length of the expanded key $Y'$ in the unit of character is equal to $l_s$, the length of $S_s$,
2) Compute $S_s = E \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y'$, and regard it as a character string.

3) Use the ASCII codes 0D and 0A ("carriage return" and "line feed") in $S_s$ as separators, break $S_s$ into program lines to reconstruct the original secret program $P_s$ as output.

Note that in Step 2 of Stage 3 above, we conduct the exclusive-OR operations of $E \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y'$. This will indeed result in the desired $S_s$ because E was computed as $E = S_s \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y'$ in Step 2 of Algorithm 1, and

$$\begin{aligned}
& E \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y' \\
= \; & (S_s \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N \oplus Y') \oplus S_1 \oplus S_2 \oplus \\
& \quad \cdots \oplus S_N \oplus Y' \\
= \; & S_s \oplus (S_1 \oplus S_1) \cdots (S_N \oplus S_N)(Y' \oplus Y') \\
= \; & S_s \oplus \mathbf{0} \oplus \mathbf{0} \oplus \ldots \oplus \mathbf{0} \\
= \; & S_s,
\end{aligned}$$

by the commutative and associative laws of the exclusive-OR operation and the facts that $X \oplus X = 0$ and $X \oplus \mathbf{0} = X$ for any bit $X$, where the bold character $\mathbf{0}$ is used to represent 8 consecutive bits of zero, i.e., $\mathbf{0} = 00000000$. In the previous discussions, we assume that the proposed algorithms of secret sharing and recovery (Algorithms 1 and 2) are known to the public, and that the key $Y$ is held by a supervisor other than any of the $N$ participants. The key is provided by the supervisor as an input to the secret program sharing and recovery processes described by Algorithms 1 and 2; it is not available to any participant. Under these assumptions and by Algorithm 2 above, if any participant changes the content of the camouflage program or that of the share contained in the stego-program which he/she holds before the secret program recovery process, such illegal tampering will be found out and warnings issued during the recovery process.

# 5 Experimental Results

In one of our experiments, we applied the proposed schemes described previously to share a secret program among three participants. The main part of the secret program seen in the window of the Microsoft VC$^{++}$ editor is shown in Figure 2(a), which has the function of generating a secret key from an input seed. And part of one of the three camouflage programs is shown in Figure 2(b). After hiding the shares and the authentication signals in the comments of each camouflage program, the stego-program resulting from Figure 2(b) appears to be the upper part of Figure 2(c) which is not different from that of Figure 2(b). The real content of the stego-program seen in the window of the UltraEdit 32 editor is shown in the lower part of Figure 2(c) which includes the ASCII codes representing the program on the left and the appearance of the codes as characters on the right. The recovered secret program is shown in Figure 2(d), which is identical to that shown in Figure 2(a).

We also tested the case of recovery with one of the stego-images (the second one) being damaged, as shown in Figure 3(a). The proposed scheme issued a warning message, as shown in Figure 3(b).

# 6 Conclusion

For the purpose of protecting software programs, new techniques for sharing secret source programs and authentication of resulting stego-programs using four special ASCII control codes invisible in the window of the Microsoft VC$^{++}$ editor have been proposed. The proposed sharing scheme divides the result of exclusive-ORing the contents of the secret program and a group of camouflage programs into shares, each of which is then encoded into a sequence of invisible ASCII control codes before being embedded into the comments of the corresponding camouflage program. The resulting stego-programs are kept by the participants of the sharing process. The original function of each camouflage program is not destroyed in the corresponding stego-program. The sharing of the secret program and the invisibility of the special ASCII codes as share data provides two-fold security protection of the secret program.

In the secret program recovery process, the reversibility property of the exclusive-OR operation is adopted to recover the secret program using the share data extracted from the stego-programs. To enhance security of keeping the camouflage programs, a secret random key is adopted to verify, during the recovery process, possible incidental or intentional tampering with the hidden share and the camouflage program content in each stego-program. The key is also utilized to prevent unauthorized recovery of the secret program by illegal collection of all the stego-programs and unauthorized execution of the proposed algorithms.

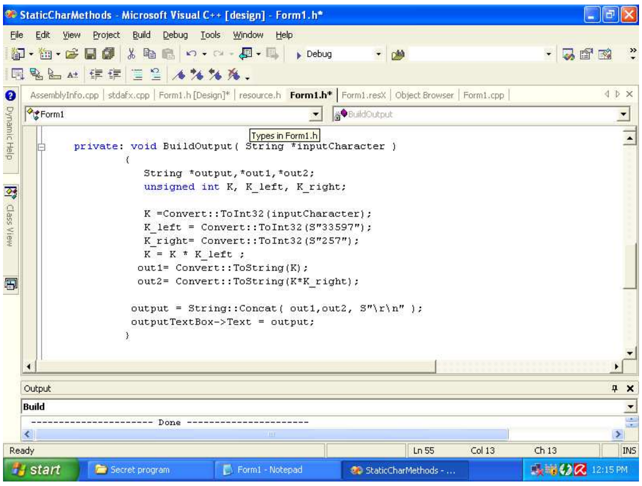
Experimental results have shown the feasibility of the proposed method. Future research may be directed to applying the invisible ASCII control codes to other applications, such as watermarking of software programs for copyright protection, secret hiding in software programs for covert communication, authentication of software program correctness, and so on.

# Acknowledgements

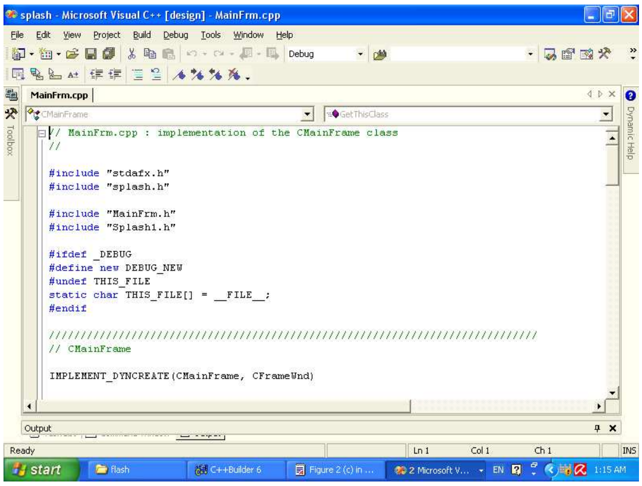
This work was supported partially by the NSC project Advanced Technologies and Applications for Next Generation Information Networks (II) - Subproject 5: Network Security, No. 96-2752-E-009-006-PAE and partially by the NSC project No. 96-2422-H-009-001.

# References

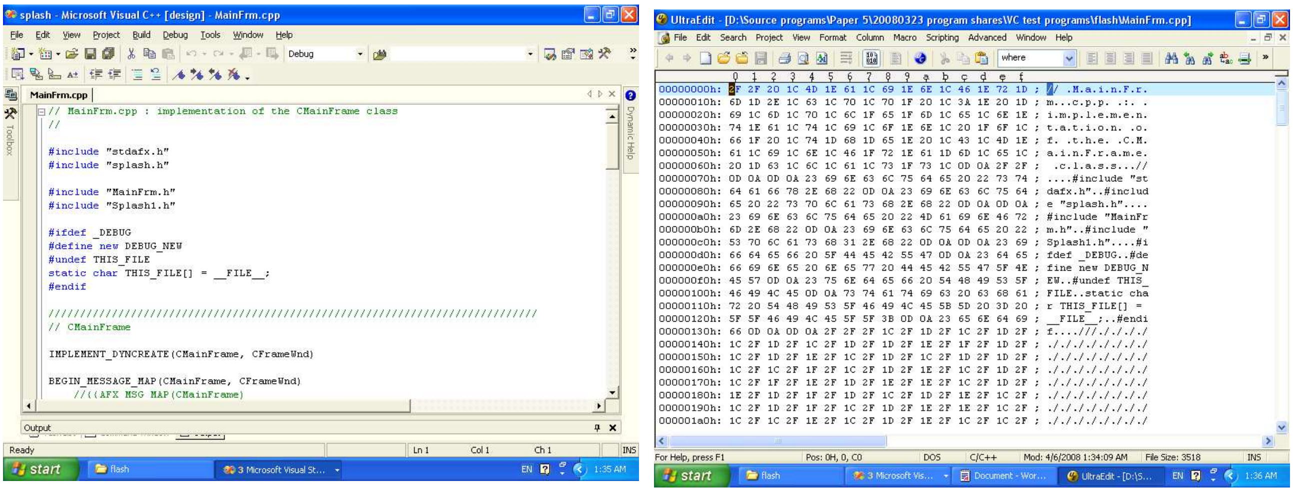
[1] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, "Visual cryptography for general access struc-



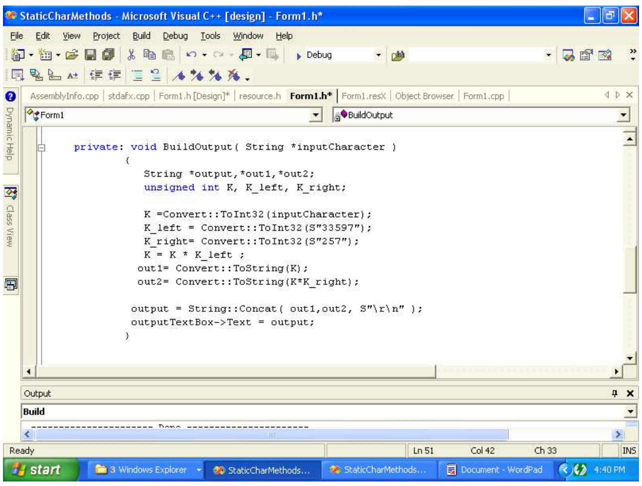

(a) Main part of the secret source program seen in the window of the Microsoft VC++ editor.




(b) Part of one camouflage program seen in the window of Microsoft Visual C++ editor.

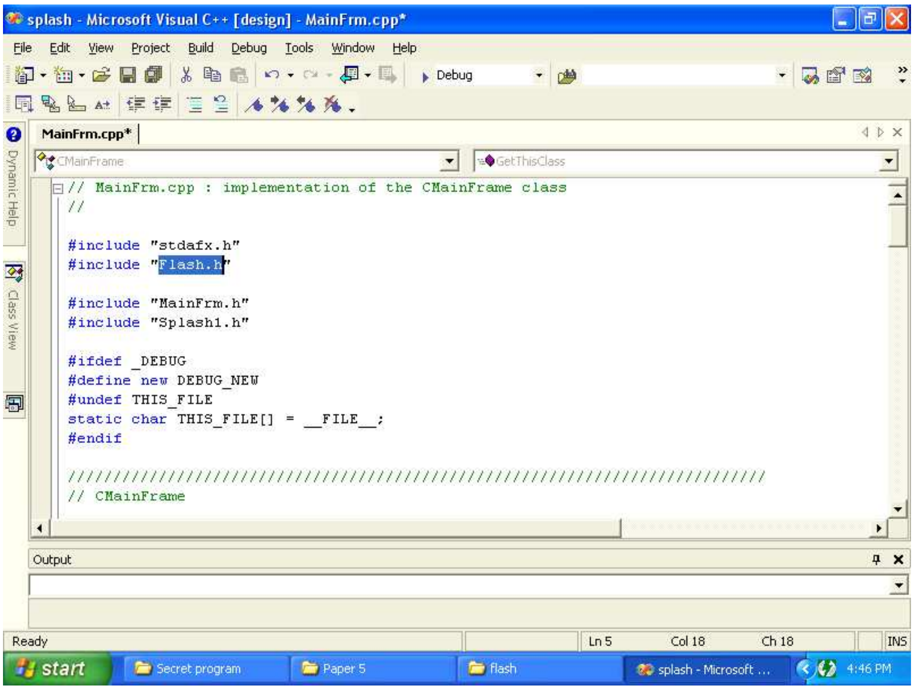



(c) The stego-program resulting from (b) seen in the window of Microsoft Visual C++ editor (left part) and UltraEditor 32 editor (right part).

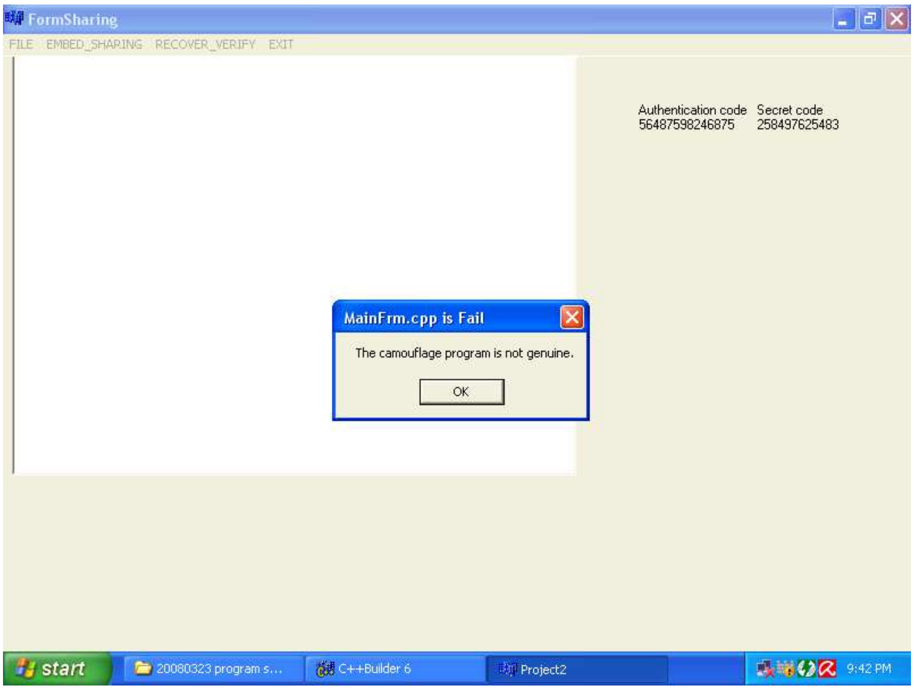



(d) Recovered secret program seen in the window of Microsoft Visual C++ editor.

Figure 2: Experimental results of sharing a secret program




(a) Destructed stego-program of Figure 2(b) seen in the window of Microsoft Visual C++ editor (the changed characters are highlighted).




(b) A message showing the content of the original camouflage program has been changed.

Figure 3: An experimental result of authenticating a destructed stego-program.

tures," *Information and Computation*, vol. 129, pp. 86-106, 1996.

[2] C. Blundo, and A. De Santis, "Visual cryptography schemes with perfect reconstruction of black pixels," *Computers & Graphics*, vol. 22, no. 4, pp. 449-455, 1998.

[3] I. S. Lee, and W. H. Tsai, "Data hiding in emails and applications by unused ASCII control codes," *Proceedings of 2007 National Computer Symposium*, vol. 4, pp. 414-422, Taichung, Taiwan, Dec. 2007.

[4] C. C. Lin, and W. H. Tsai, "Secret image sharing with steganography and authentication," *Journal of Systems & Software*, vol. 73, no. 3, pp. 405-414, 2004.

[5] M. Naor, and B. Pinkas, "Visual authentication and identification," *Advances in Cryptology, Crypto' 97*, LNCS 1294, pp. 322-336, 1997.

[6] M. Naor, and A. Shamir, "Visual cryptography," *Advances in Cryptology, Eurocrypt' 94*, LNCS 950, pp. 1-12, 1995.

[7] J. Pieprzyk and X. M. Zhang, "Ideal secret sharing schemes from permutations," *International Journal of Network Security*, vol. 2, no. 3, pp. 238-244, 2006.

[8] A. Shamir, "How to share a secret," *Communications of the Association for Computing Machinery*, vol. 22, no. 11, pp. 612-613, 1979.

[9] E. R. Verheul, and H. C. A. van Tilborg, "Construction and properties of k out of n visual secret sharing schemes," *Designs, Codes, and Cryptography*, vol. 11, pp. 179-196, 1997.

**I-Shi Lee** was born in Taipei, Taiwan, R.O.C., in 1961. He received the B. S. degree in electronic engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, Republic of China in 1987, the M. S. degree in the Department of Computer Science and Information Science at National Chiao Tung University in 1989, and the Ph.D. degree in the Institute of Computer Science and Engineering, College of Computer Science from National Chiao Tung University in 2008.

In 1992, he joined the Department of Management Information at Northern Taiwan Institute of Science and Technology and acted as a lecturer from 1992 to now. His recent research interests include pattern recognition, watermarking, and image hiding.

**Wen-Hsiang Tsai** was born in Tainan, Taiwan on May 10, 1951. He received the B. S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan in 1973, the M. S. degree in electrical engineering (with major in computer science) from Brown University, Providence, Rhode Island, U. S. A. in 1977, and the Ph. D. degree in electrical engineering (with major in computer engineering) from Purdue university, West Lafayette, Indiana, U. S. A. in 1979.

Dr. Tsai joined the faculty of National Chiao Tung University (NCTU), Hsinchu, Taiwan in November 1979. He is currently an NCTU Chair Professor in the Department of Computer Science and Information Engineering. From August 2004, he is also the President of Asia University, Taichung, Taiwan. Professor Tsai has been an Associate Professor of the Department of Computer Engineering (now Department of Computer Science) and the Acting Director of the Institute of Computer Engineering. In 1984, he joined the Department of Computer and Information Science (now also Department of Computer Science), and acted as the Department Head from 1984 through 1988. He has also been the Associate Director of the Microelectronics and Information System Research Center from 1984 through 1987, the Dean of General Affairs from 1995 to 1996, the Dean of Academic Affairs from 1999 to 2001, the Acting Dean of the College of Humanities and Social Science in 1999, and the Vice President of the University from 2001 to 2004.

Outside the campus, Professor Tsai has served as a Consultant to many major research institutions in Taiwan, including the Chun-Shan Institute of Science and Technology, the Industrial Technology Research Institute, and the Information Industry Institute. He has acted as the Coordinator of Computer Science in the Engineering Division of the National Science Council, and a member of the Counselor Committee of the Institute of Information Science of Academia Sinica in Taipei. He has been the Editor of several academic journals, including Computer Quarterly (now Journal of Computers), the Proceedings of National Science Council, the Journal of the Chinese Engineers, the International Journal of Pattern Recognition and Artificial Intelligence, the Journal of Information Science and Engineering, and Pattern Recognition. He was the Editor-in-Chief of the Journal of Information Science and Engineering from 1998 through 2000.

Professor Tsai's major research interests include image processing, computer vision, virtual reality, and information copyright and security protection. So far he has published 320 academic papers, including 121 journal papers and 199 conference papers. He is also granted six R. O. C. and U. S. A. patents. Dr. Tsai has supervised the thesis studies of 30 Ph. D. students and 138 master students. Dr. Tsai is a senior member of the IEEE, a member of the Chinese Image Processing and Pattern Recognition Society, and the International Chinese Computer Society. He served as the Chairman of the Chinese Image Processing and Pattern Recognition Society at Taiwan from 1999 to 2000. He is now the Chair of the Computer Society of IEEE Taipei Section.