Secured and Authenticated Sharing of Software Program

MADHAVI GANGURDE
Department of Information Technology, Vidyavardhini's College of Engineering & Technology, Mumbai University
Mumbai India

UMASHANKAR PRASAD
Department of Information Technology, Vidyavardhini's College of Engineering & Technology, Mumbai University
Mumbai India

PRASHANT PRAJAPATI
Department of Information Technology, Vidyavardhini's College of Engineering & Technology, Mumbai University
Mumbai India

RANVIR KUMAR
Department of Information Technology, Vidyavardhini's College of Engineering & Technology, Mumbai University
Mumbai India

ABSTRACT

A new method for software program protection by information sharing & authentication technique is proposed. In this scheme we will share a secret source program among a group of participants. Each of them holds a camouflage programs to hide a share. Each camouflage program will hold a secret source program, thus resulting in stego-program. Each stego-program will still be compiled and executed to perform its original functionality. The security will be further enhanced by encrypting source program with secret key which not only can prevent the secret program from being recovered illegally but also can authenticate the stego -program provided by each participant. The secret program will only be recovered when all the participant supplies correct stego-program and correct key. During the recovery process we can check the stego-program have been tempered or not incidentally/ intentionally.

This scheme can be applied to software programs for copyright protection, secret hiding in software program for covert channel, etc.

General Terms

Secured And Authenticated Sharing .

Keywords

Authentication, information sharing, invisible ASCII control codes, program sharing, secret program, security protection, software program.

1. INTRODUCTION

Information hiding is a promising approach to covert communication because it yields a steganographic effect which enhances communication security. So far, data hiding in computer programs is mainly for copyright protection. Also, the source program is seldom used as the cover media. However, embedding message in software programs in source form has not been studied yet. In this project, a new covert communication method by embedding messages in source programs is proposed.

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. 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.

To provide perfect security to share the software program is the main aim of our project.

Security will be provided in two ways that is why the name is "Secured and authenticated Sharing of software program". This is a new method for software program protection by information sharing and authentication techniques using invisible ASCII control codes.

A scheme for sharing a secret source program written in any language among a group of participants, each holding a camouflage program to hide a share, is first proposed for safe keeping of the secret program. 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. 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.

2. DETAIL PROBLEM EXPLANATION

2.1 Problem Definition

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 being embedded. These two data transformations are based on a binary-to-ASCII mapping proposed in this study, which is described as a table 1 as shown.

TABLE 1. Invesible Ascii Code Mapping

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

Specially, 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.

2.2 Problem Objectives

As our project name is "secured and authenticated sharing of software program", we will provide two way security for sharing software program. 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 has two features

1) Security: The secret program can't be recovered illegally without providing the secret key.
2) Authentication: It can also 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.

2.3 Scope Of Paper

The idea behind creating this application to provide the safety for sharing of intellectual properties ie. software programs. Traditional methods for encryption are ill-suited for simultaneously achieving high levels of confidentiality and reliability. This is because when storing the secret key, one must choose between keeping a single copy of the key in one location for maximum secrecy, or keeping multiple copies of the key in different locations for greater reliability. Increasing reliability of the key by storing multiple copies lowers confidentiality by creating additional attack vectors, there are more opportunities for a copy to fall into the wrong hands. Secret sharing schemes address this problem, and allow arbitrarily high levels of confidentiality and reliability to be achieved.

TABLE 2

Dec	Hex	Char	Description
28	1C	FS	file separator
29	1D	GS	group separator
30	1E	RS	record separator
31	1F	US	unit separator

3. DATA HIDING BY INVISIBLE ASCII CODES

ASCII codes, expressed as hexadecimal numbers, were designed to represent 8-bit characters for information interchange. It is found in this study that some ASCII codes, when embedded in certain locations in programs, become invisible in the source code editors and Builder under certain Windows OS environments. This phenomenon may be utilized for data hiding.

Two types of invisible codes are identified, one appearing as nothing like being non-existing, and the other as spaces just like the ASCII space code 20. We call the former null code and the latter spacing code. Inserting invisible codes into a program do not change its function. In type-1 environment four null codes, 1C, 1D, 1E, 1F, were found, which are invisible when inserted between two characters in a comment in a program. One spacing code A0, has been found, which appears as a space when inserted between two words in a comment. Also found as a spacing code is the tab-control code 09, which in default appears as four spaces when inserted before the end of a program line, i.e., before the code pair, 0D0A, for carriage return and line feed. The codes, A0 and 09, will be called between-word and line-end spacing codes, respectively. For the other three environment types, invisible codes also exist and are listed in Table 1 except that type-2 environment has no null code.

We can conduct data hiding using invisible codes in three ways as follows.

3.1 Alternative Space Coding

Whenever a space represented by 20 appears between two words in a comment, it may be replaced by a between-word spacing code, like A0 for type-1 environment, without causing visual difference in a source code editor. When there are 2^n-1 between-word spacing codes C_1, C_2, ..., C_{2n-1}, by regarding 20 as C_0 we may embed n bits $b_1, b_2, ..., b_n$ as follows.

if $b_1b_2....b_n = m$, replace 20 by C_m

Which we call alternative space coding.

For the first two environments in Table 1, 1-bit alternative space coding is applicable. And for the latter two, there are 14 and 23 spacing codes, respectively and so 3-bit and 4-bit alternative space coding are applicable, respectively.

3.2 Line-End Space Coding

We may place multiple line-end spacing codes before each program line end without causing visual difference in a source code editor because such codes appear just like background spaces in the window of the editor. Since the code 20 may be used as well to create spaces, when there are 2^n-1 line-end spacing codes C_1, ... , C_{2n-1}, by regarding 20 as C_0 we may embed n bits $b_1, b_2, ..., b_n$ as follows.

if $b_0b_1...b_n = m$, embed C_m before the line end

Which we will call line-end space coding.

For the first two environments, there is only one line-end spacing code 09, so 1-bit line-end coding is applicable. For the latter two, since there are three such codes 09, 0B, and 0C, 2-bit coding can be implemented.

Line-end space coding may be repeated unlimited times before the each line end to increase the data hiding rate. But to avoid creating long lines which reduce the steganographic effect, we require that each processed program line should not appear to be longer than the longest original program line.

3.3 Null Space Coding

Except for type-2 environment, there are four null codes, 1C, 1D, 1E, 1F. Let them be represented by C_0 through C_3, respectively. We can embed a bit pair b_0b_1 as follows.

if $b_0b_1 = m$, insert C_m between two characters in a comment

Which we call null space coding.

Null space coding may be applied repetitively unlimited times as well. In practice, we embed message bits evenly into all between-character spaces among the comments so that the times will be limited.

4. PROGRAM SHARING SCHEME

A Overview of algorithm is proposed for sharing of 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 1) 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.

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 Ps, from the secret program and the camouflage programs in the following way
   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, . . . ,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 $E = S_s \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_N$ 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…..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_{Sk} , transform A_{Sk} 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_{Ek} with length l_Y , too.
   c) Concatenate A_{Sk} and A_{Ek} 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, . . . ,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 in-visible ASCII control code according to the invisible coding table (Table 1), 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 $P`_1, P`_2$ …. , $P`_N$ as the output stego-programs.

Table 3: Symbolic 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, . . . ,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).

5. PROGRAM RECOVERY SCHEME

A overview 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.

- ***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 1), 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.
- ***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.
- ***Checking correctness of shares and camouflage programs.*** Checking whether the content of Actual Share is tempered intentionally or accidentally by Matching the value of A_{ek}' and A_{ek}. Also Checking for camouflage program is tempered by Matching the value of A_{pk}' and A_{pk}. If both check are successful then we proceed for Recovery.
- ***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.

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

Input.
N stego-programs $P`_1, P`_2, \ldots, P`_N$ provided by the N participants and the secret key Y with length l_Y used in secret program sharing .

Output.
The secret program P_s hidden is 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, . . . ,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 1) 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_{Sk} and A_{Ek} .

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 P_k 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 $A`_{Sk}$, transform $A`_{Sk}$ 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 $A`_{Ek}$

d) Compare $A`_{Sk}$ with the previously extracted A_{Sk} , if mismatching, issue the message “The camouflage program is not genuine," and stop the algorithm.

e) Compare $A`_{Ek}$ with the previously extracted A_{Ek} , 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 $S_s = E \oplus S_1 \oplus S_2 \oplus . . \oplus S_N \oplus Y`$,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.

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 all Editor of Microsoft windows and Some linux 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.

Below is Figure of Example .

7. ACKNOWLEDGMENTS

This work was supported by our college Prof. Madhavi Gangurde and all staff, Department Of Information Technology, Vidyavardhini’s College of Engineering & Technology, Vasai (West) 401202, University of Mumbai.

8. REFERENCES

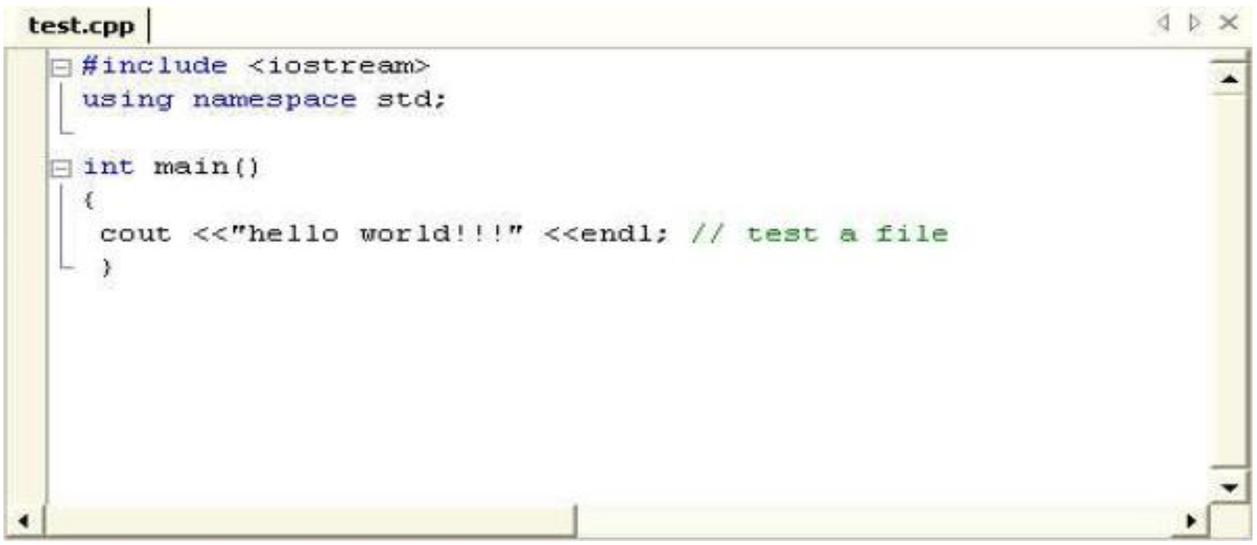
[1] S. Lee, and W. H. Tsai, “Data hiding in emails and applications by unused ASCII control codes," Proceedings of 2007 National Computer Symposium, vol. 4, pp. 414-422, Taichung, Taiwan, Dec. 2007.

[2] S. Lee, and W. H. Tsai, “Covert Communication with Authentication via Software Programs Using Invisible ASCII Codes"

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

[4] S. Lee, and W. H. Tsai, “Security Protection of Software Programs by Information Sharing and Authentication Techniques Using Invisible ASCII Control Codes," International Journal of Network Security, Vol.10, No.1, PP.1–10, Jan. 2010.

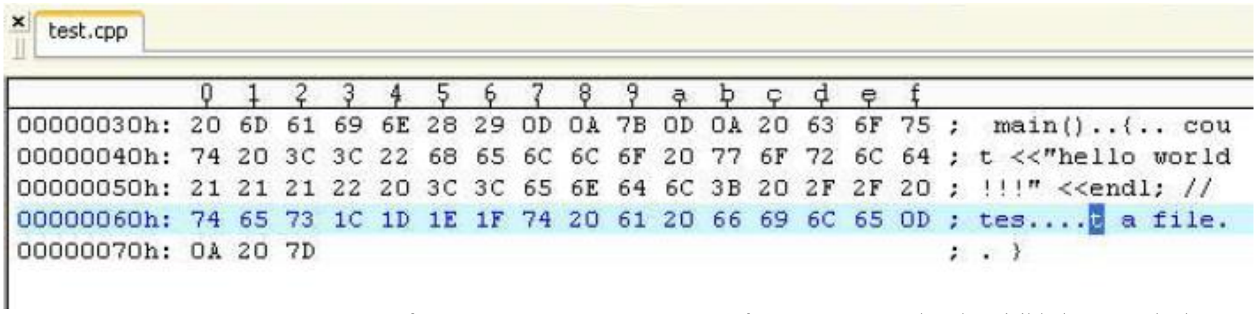
[5] The IEEE website. [Online]. Available: http://www.ieee.org/


test.cpp

```
#include <iostream>
using namespace std;

int main()
{
 cout <<"hello world!!!" <<endl; // test a file
 }
```

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



```
             0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000030h: 20 6D 61 69 6E 28 29 0D 0A 7B 0D 0A 20 63 6F 75 ;  main()..{.. cou
00000040h: 74 20 3C 3C 22 68 65 6C 6C 6F 20 77 6F 72 6C 64 ; t <<"hello world
00000050h: 21 21 21 22 20 3C 3C 65 6E 64 6C 3B 20 2F 2F 20 ; !!!" <<endl; //
00000060h: 74 65 73 1C 1D 1E 1F 74 20 61 20 66 69 6C 65 0D ; tes....t a file.
00000070h: 0A 20 7D                                        ; . }
```

(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.