Covert Communication with Authentication via Software Programs Using Invisible ASCII Codes --- A New Approach

I-Shi Lee
Dept. of Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30010
Dept. of Management Information, Technology and Science Institute of Northern Taiwan, Taipei, Taiwan 11202
Email: gis87809@gmail.com

Wen-Hsiang Tsai
Dept. of Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30010
Dept. of Computer Science and Information Engineering, Asia University, Taichung, Taiwan 41354
Email: whtsai@cis.nctu.edu.tw

***Abstract*—A new covert communication method by hiding messages in source programs is proposed. Each binary message, after being encoded by certain ASCII codes and inserted at specific C++ program locations, becomes invisible in the source code editors of Visual C++ and C++ Builder under Windows OS environments, creating an effect of steganography. A scheme for tamper-proof authentication of the embedded message has also been proposed. Experimental results show the feasibility of the proposed method.**

***Index Terms*—authentication, covert communication, data hiding, invisible ASCII codes, source program.**

I. Introduction

Information hiding [3] 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.

A survey about watermarking in programs can be found in Zhu, et al. [8]. Two approaches have been identified: static and dynamic. The former inserts and extracts watermarks in program codes without running the program while the latter does the same in the execution state of a software object. Two respective examples are Venkatesan, et al. [7] and Collberg and Thomborson [2]. There exist other methods with digital text, sentence syntax, text typos, e-mails [1, 3-6] as cover media.

However, embedding message in software programs *in source form* has not been studied yet. In this paper, a new covert communication method by embedding messages in *source programs* is proposed. A binary message, after being encoded into some ASCII control codes and embedded into certain C++ program locations, becomes *invisible* in the source code editors of Visual C++ and C++ Builder under some Windows OS environments, creating a steganographic effect. 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 rest of the paper, the source program into which information is to be embedded will be referred to as *cover program*, and the result of the embedding as *stego-program*.

A stego-program still can be compiled and executed to perform the function of the original source program, and each source program may be selected arbitrarily, thus enhancing the steganographic effect.

To improve the security protection effect further, we propose additionally a tamper-proof authentication scheme for the embedded message. The protection capability is carried out with the provision of a secret random key through the use of certain mathematical operations. Without a correct secret key it is impossible to pass such an authentication process with a modified stego-program.

In the sequel, we describe how invisible ASCII codes are found and used for data hiding, and propose the secret hiding, recovery, and authentication processes in Sections II and III, respectively. Experimental results are presented in Section IV, followed by a conclusion.

II. 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 C++ programs, become *invisible* in the source code editors of Visual C++ and C++ 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.

Such invisibility was found in fours environments formed by Microsoft Visual Studio (MVS) .NET 2003 and Borland C++ Builder (BCB), version 6, in Windows XP Service Pack 2 and its Chinese version, which will be called the English and Chinese OS, respectively, subsequently. The details are summarized in Table 1.

In type-1 environment with the MVS in the English OS, 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. Also, 09 appears to be eight spaces in BCB instead of four as in MVS.

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

A. 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:

$$\textit{if } b_1b_2....b_n = m, \textit{ replace 20 by } C_m$$

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.

B. 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 $C_1, ..., C_{2^n-1}$, by regarding 20 as C_0 we may embed n bits $b_1, b_2, ..., b_n$ as follows:

$$\textit{if } b_0b_1...b_n = m, \textit{ embed } C_m \textit{ before the line end}$$

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.

C. 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.

III. Secret Hiding, Recovery and Authentication

The proposed data hiding process essentially is to apply alternative, line-end, and null space coding in order. Since the three schemes are applied to *distinct* locations in a program, the data may be recovered without ambiguity. As an example, we describe in the following an algorithm for type-1 environment. To facilitate data recovery, we prefix to the beginning of the input binary string of the message a binary number specifying the length of the input, resulting in an extended bit string S.

1. At each between-word space coded by 20, remove the leading bit b from S, and replace 20 by A0 if b =1.
2. Find the maximum $L_{\max}$ of all program line lengths.
3. For each program line, repeat the operations of removing the leading bit b from S and inserting before the line end the code 09 if b = 1; or 20 if b = 0, until the length of the line, as it appears in the source code editor, reaches $L_{\max}$.
4. Count the number M of all between-character positions in the comments, as well as the number L of the remaining bits in S; compute the ceiling value $\lceil L/M \rceil$; add 1 to it to make it even if it is not; and denote the final value as q.
5. For each between-character position in the comments, take q leading bits from the remaining bits in S, and for every two bits b_0b_1 of them, insert C_m into the position if $b_0b_1 = m$, where C_m is one of C_0 through C_3 representing 1C, 1D, 1E, and 1F, respectively.

The proposed data recovery process, after extracting from the input string the leading bits which specify the length of the original message, performs essentially the reverse versions of the three coding schemes involved in the data hiding process. The details are omitted due to the page limit.

In the proposed authentication scheme, we use a L-bit key K and the input message string to generate an authentication signal A which is then embedded in the stego-program as well using null space coding. The signal is computed as the modulo-K value of the sum of the key value and the L bits of every two characters in the input message string. Then, in the data recovery process, the embedded authentication signal A is extracted to match with an authentication signal A' computed similarly from the extracted message content and the key. If the embedded message content has not been tampered with, then A and A' will match. If not, then the message must have been modified. In such a way, even when the data hiding algorithm is known to the public as is usually assumed, without the secret key it is impossible to pass such an authentication process with a modified stego-program.

IV. Experimental Results

One of the experiments we conducted for type-1 environment is reported here. A message "This is a new covert communication method" is embedded into a cover program, part of which is shown in Fig. 1(a). The binary form of the message is obtained from the ASCII characters representing the message. It is <u>00000001</u> <u>01100000</u> <u>01010100</u> <u>01101000</u>... in which the first 16 bits specify the length of the message string, and the remaining ones represents T, h, and so on. And the encoding result of it is 20 20 20 20 20 20 20 A0 20 A0 A0 20 20 20 20 20 ... The stego-program seen in the source code editor is shown in Fig. 1(b), which looks no difference from Fig. 1(a). And the real content of the program seen in the UltraEdit editor is shown in Fig. 1(c), in which the hidden invisible codes can be seen with those for the first 16 bits being enclosed by rectangles. The recovered message is shown in Fig. 1(d). As a demonstration of authentication, we show in Fig. 2(a) a modified version of the stego-program of Fig. 1(b) in the UltraEdit editor, in which the codes fro the 8th and

9th bits of the message have been modified. The authentication result is shown in Fig. 2(b) in which a warning message issued by the data recovery process is seen.

V. Conclusion

A new approach to covert communication via C++ source programs using invisible ASCII codes has proposed. A binary secret message is encoded by some special ASCII codes, which are then embedded in a cover program. Such codes are invisible in the source code editors of Visual C++ and C++ Builder under Windows OS environments, creating a good steganographic effect without changing the original function of the cover program. To enhance security, tamper-proof authentication of the stego-program content using a secret key has also been proposed. Without the key, false messages cannot pass the authentication process. Experimental results show the feasibility of the proposed approach. Future works may be directed to applying the proposed data hiding technique to other applications.

Acknowledgements

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-96-2752-E-009-006-PAE. And partially by the NSC project NSC96-2422-H-009-001.

References

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, “Techniques for data hiding,” *IBM System Journal*, vol. 35, nos. 3 & 4, 1996.

[2] C. Collberg & C. Thomborson, “Watermarking, tamper-proofing, and obfuscation — tools for software protection,” *IEEE Transactions on Software Engineering*, vol. 28, pp. 735-746, 2002.

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

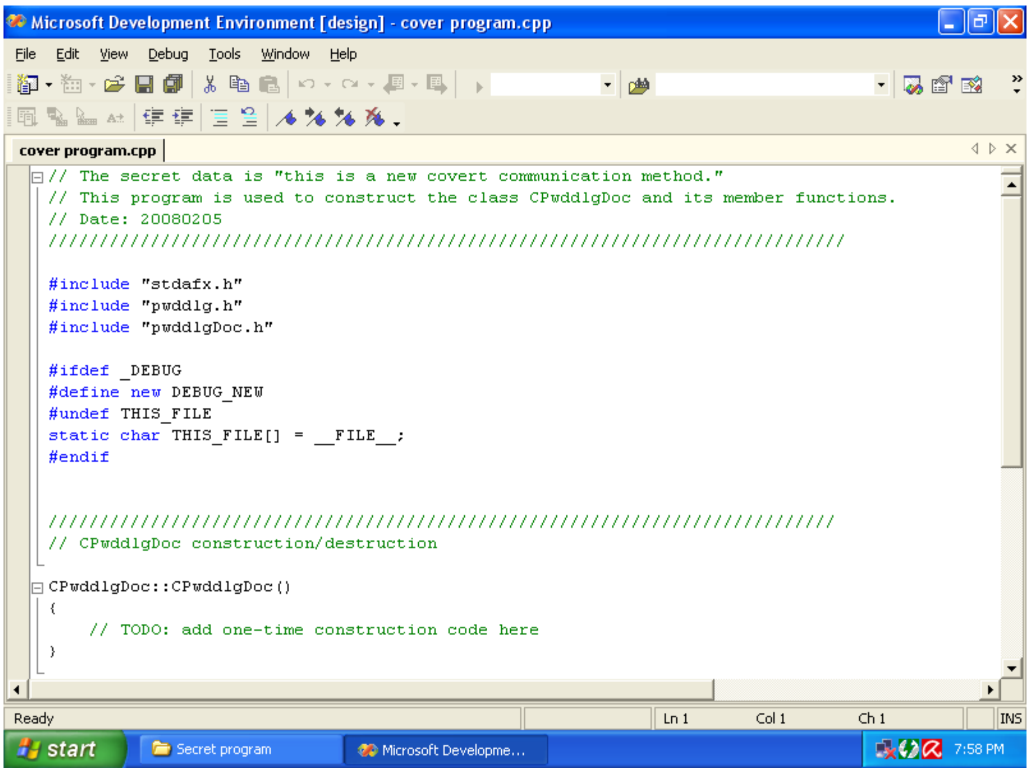
[4] I. S. Lee & W. H. Tsai, “Data hiding in emails and applications by unused ASCII control codes,” *Proceedings of 2007 National Computer Symposium*, Taichung, Taiwan, 2007.

[5] H. M. Meral, E. Sevinc, E. Unkar, B. Sankur, A. S. Ozsoy, and T. Gungor, “Syntactic tools for text watermarking,” *Proceedings of SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, USA, 2007.

[6] M. Topkara, U. Topkara, & M. J. Atallah, “Information hiding through errors: a confusing approach,” *Proceedings of SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, USA, 2007.

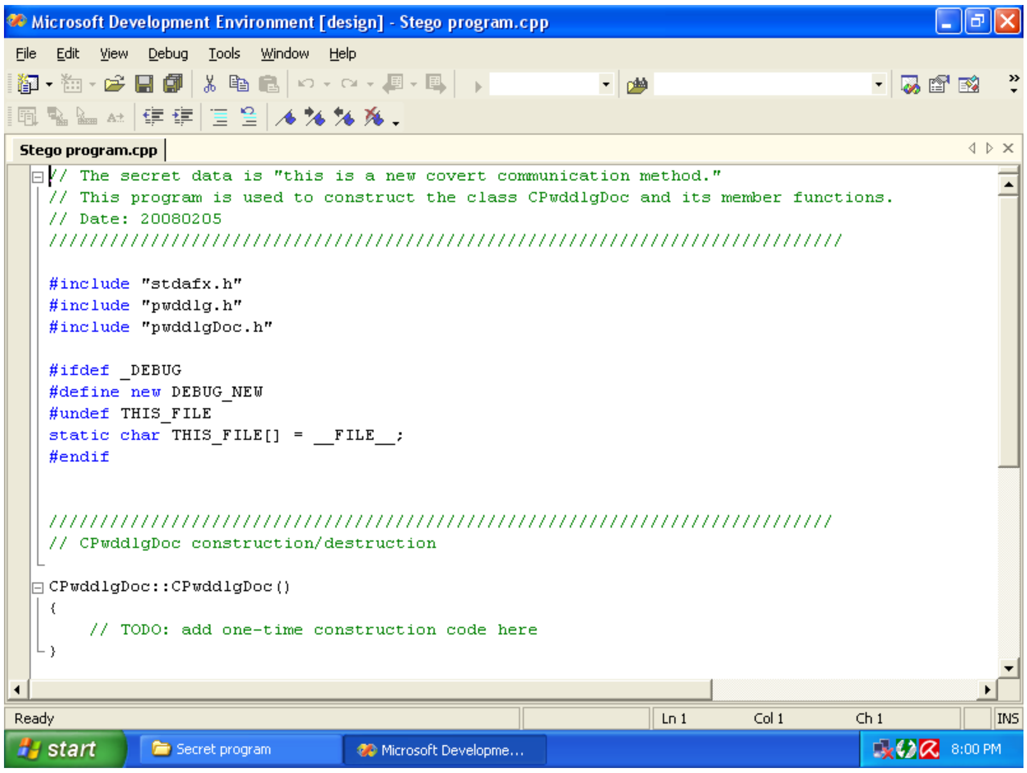
[7] R. Venkatesan, V. Vazirani, & S. Sinha, “A graph theoretic approach to software watermarking,” *Proceedings of 4th International Information Hiding Workshop*, Pittsburgh, Pennsylvania, USA, 2001.

[8] W. Zhu, C. Thomborson, & F. Y. Wang, “A survey of software watermarking,” *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, Atlanta, Georgia, USA, 2005.


Microsoft Development Environment [design] - cover program.cpp
File Edit View Debug Tools Window Help
cover program.cpp
Ready
Ln 1
Col 1
Ch 1
INS
start
Secret program
Microsoft Developme...
7:58 PM

```
// The secret data is "this is a new covert communication method."
// This program is used to construct the class CPwddlgDoc and its member functions.
// Date: 20080205
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "pwddlg.h"
#include "pwddlgDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CPwddlgDoc construction/destruction

CPwddlgDoc::CPwddlgDoc()
{
    // TODO: add one-time construction code here
}
```

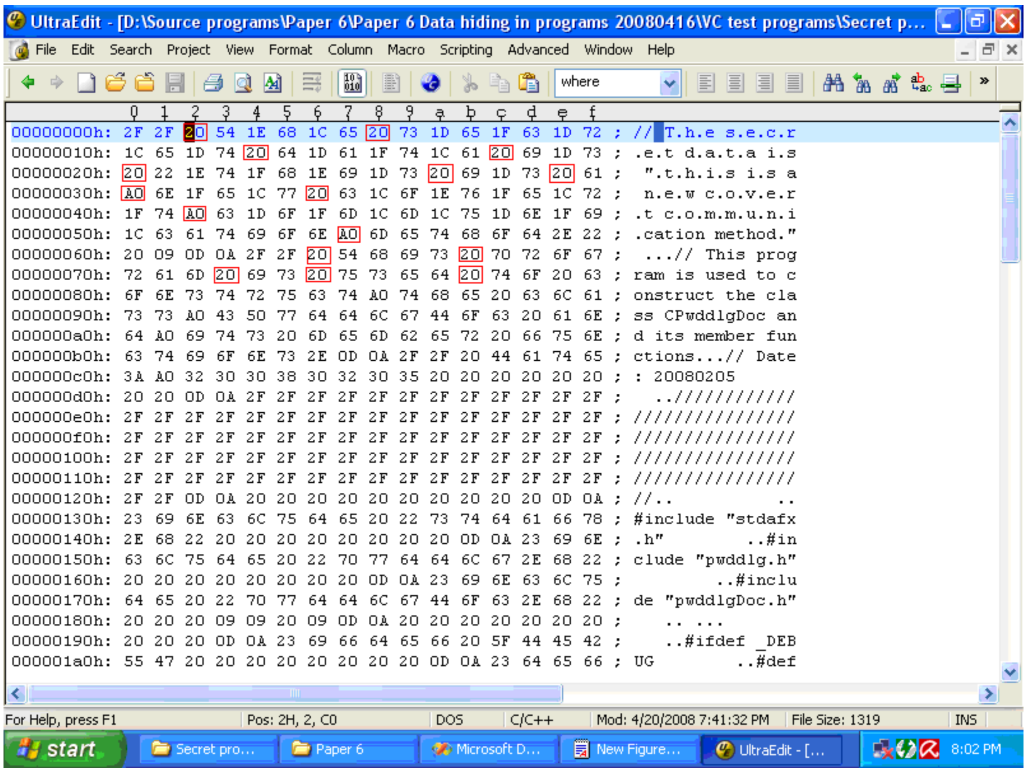
(a) Cover program seen in source code editor.


Microsoft Development Environment [design] - Stego program.cpp
File Edit View Debug Tools Window Help
Stego program.cpp
Ready
Ln 1
Col 1
Ch 1
INS
start
Secret program
Microsoft Developme...
8:00 PM

```
// The secret data is "this is a new covert communication method."
// This program is used to construct the class CPwddlgDoc and its member functions.
// Date: 20080205
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "pwddlg.h"
#include "pwddlgDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CPwddlgDoc construction/destruction

CPwddlgDoc::CPwddlgDoc()
{
    // TODO: add one-time construction code here
}
```

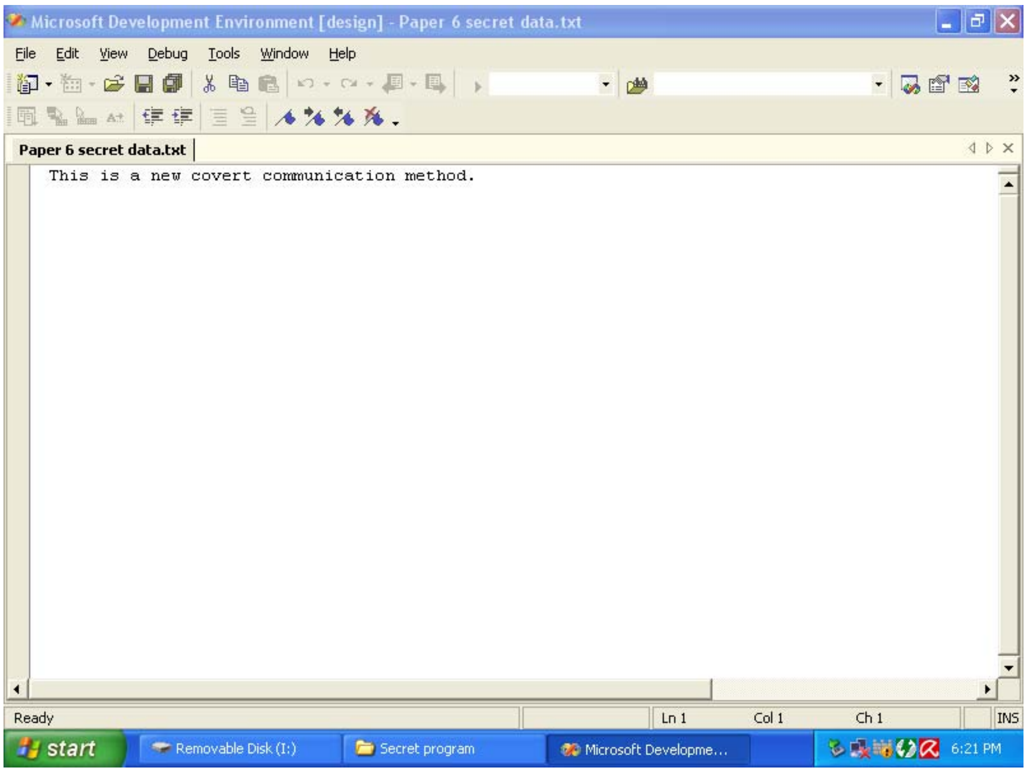
(b) Stego-program seen in source code editor.

Fig. 1 An experimental result.


UltraEdit - [D:\Source programs\Paper 6\Paper 6 Data hiding in programs 20080416\VC test programs\Secret p...
File Edit Search Project View Format Column Macro Scripting Advanced Window Help
where

```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 2F 2F 20 54 1E 68 1C 65 20 73 1D 65 1F 63 1D 72 ; // T.h.e s.e.c.r
00000010h: 1C 65 1D 74 20 64 1D 61 1F 74 1C 61 20 69 1D 73 ; .e.t d.a.t.a i.s
00000020h: 20 22 1E 74 1F 68 1E 69 1D 73 20 69 1D 73 20 61 ;  ".t.h.i.s i.s a
00000030h: A0 6E 1F 65 1C 77 20 63 1C 6F 1E 76 1F 65 1C 72 ;  n.e.w c.o.v.e.r
00000040h: 1F 74 A0 63 1D 6F 1F 6D 1C 6D 1C 75 1D 6E 1F 69 ; .t c.o.m.m.u.n.i
00000050h: 1C 63 61 74 69 6F 6E A0 6D 65 74 68 6F 64 2E 22 ; .cation method."
00000060h: 20 09 0D 0A 2F 2F 20 54 68 69 73 20 70 72 6F 67 ;  ...// This prog
00000070h: 72 61 6D 20 69 73 20 75 73 65 64 20 74 6F 20 63 ; ram is used to c
00000080h: 6F 6E 73 74 72 75 63 74 A0 74 68 65 20 63 6C 61 ; onstruct the cla
00000090h: 73 73 A0 43 50 77 64 64 6C 67 44 6F 63 20 61 6E ; ss CPwddlgDoc an
000000a0h: 64 A0 69 74 73 20 6D 65 6D 62 65 72 20 66 75 6E ; d its member fun
000000b0h: 63 74 69 6F 6E 73 2E 0D 0A 2F 2F 20 44 61 74 65 ; ctions...// Date
000000c0h: 3A A0 32 30 30 38 30 32 30 35 20 20 20 20 20 20 ; : 20080205
000000d0h: 20 20 0D 0A 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ;   ..////////////
000000e0h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
000000f0h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000100h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000110h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000120h: 2F 2F 0D 0A 20 20 20 20 20 20 20 20 20 20 0D 0A ; //..          ..
00000130h: 23 69 6E 63 6C 75 64 65 20 22 73 74 64 61 66 78 ; #include "stdafx
00000140h: 2E 68 22 20 20 20 20 20 20 20 20 0D 0A 23 69 6E ; .h"        ..#in
00000150h: 63 6C 75 64 65 20 22 70 77 64 64 6C 67 2E 68 22 ; clude "pwddlg.h"
00000160h: 20 20 20 20 20 20 20 20 0D 0A 23 69 6E 63 6C 75 ;         ..#inclu
00000170h: 64 65 20 22 70 77 64 64 6C 67 44 6F 63 2E 68 22 ; de "pwddlgDoc.h"
00000180h: 20 20 20 09 09 20 09 0D 0A 20 20 20 20 20 20 20 ;    .. ...
00000190h: 20 20 20 0D 0A 23 69 66 64 65 66 20 5F 44 45 42 ;    ..#ifdef _DEB
000001a0h: 55 47 20 20 20 20 20 20 20 20 0D 0A 23 64 65 66 ; UG        ..#def
```

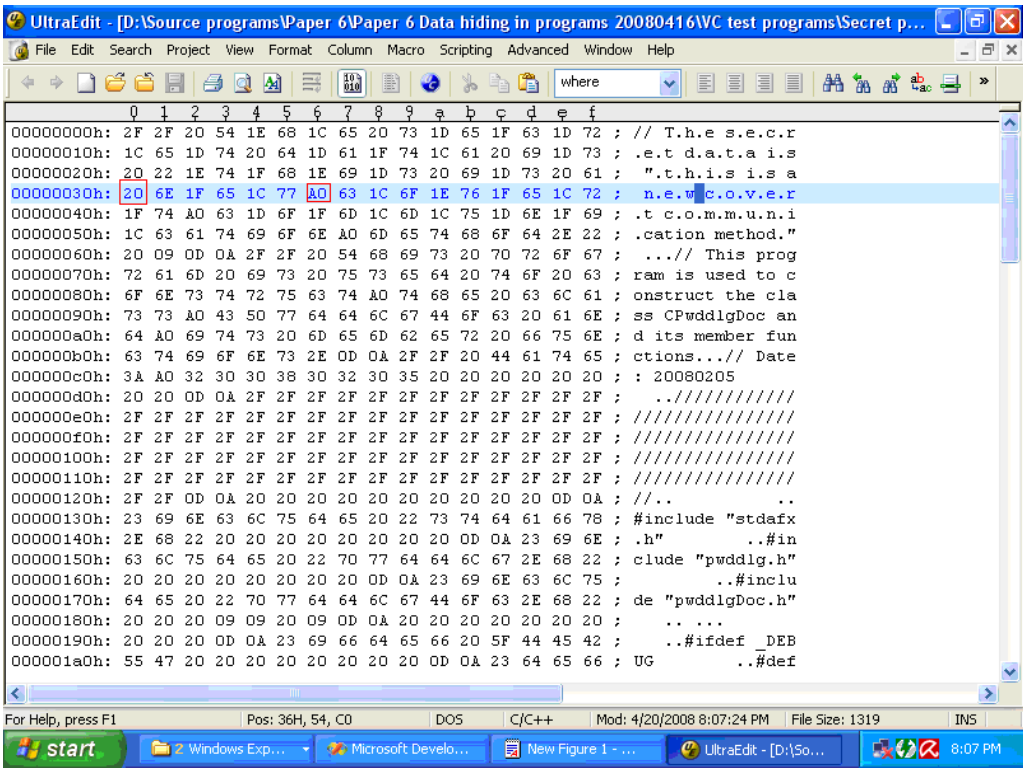
(c) Stego-program seen in UltraEdit.


Microsoft Development Environment [design] - Paper 6 secret data.txt
File Edit View Debug Tools Window Help
Paper 6 secret data.txt

```
This is a new covert communication method.
```

(d) Recovered message.

Fig. 1 An experimental result (cont'd).

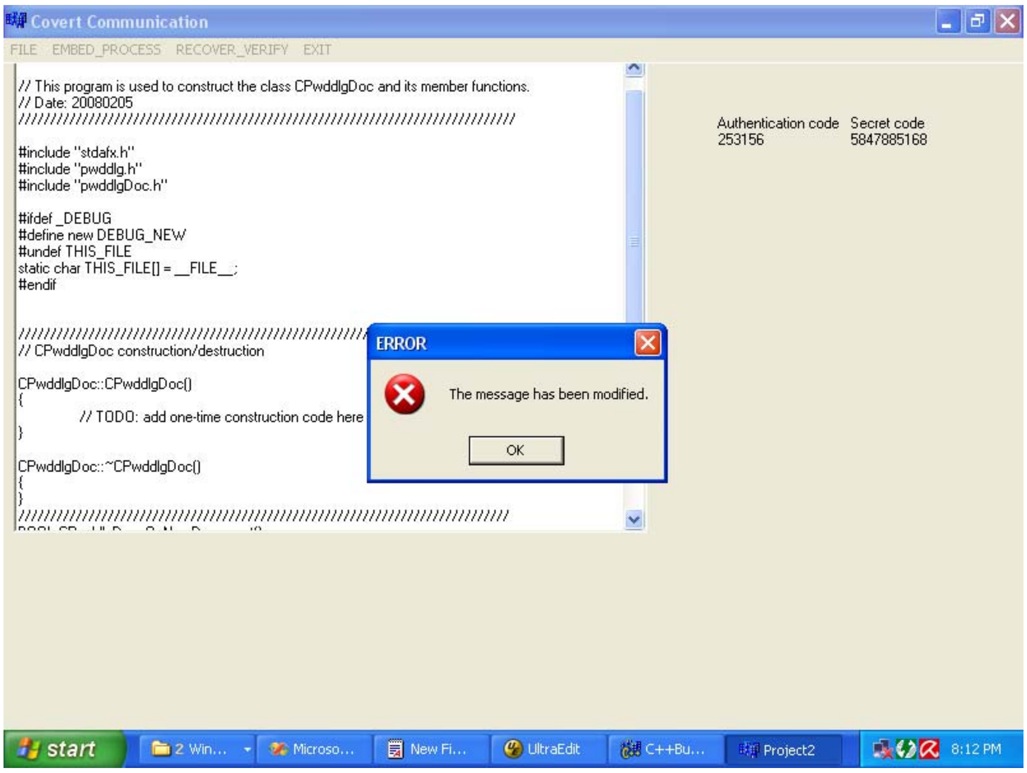


UltraEdit - [D:\Source programs\Paper 6\Paper 6 Data hiding in programs 20080416\VC test programs\Secret p...

```
              0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 2F 2F 20 54 1E 68 1C 65 20 73 1D 65 1F 63 1D 72 ; // T.h.e s.e.c.r
00000010h: 1C 65 1D 74 20 64 1D 61 1F 74 1C 61 20 69 1D 73 ; .e.t d.a.t.a i.s
00000020h: 20 22 1E 74 1F 68 1E 69 1D 73 20 69 1D 73 20 61 ;  ".t.h.i.s i.s a
00000030h: 20 6E 1F 65 1C 77 A0 63 1C 6F 1E 76 1F 65 1C 72 ;  n.e.w c.o.v.e.r
00000040h: 1F 74 A0 63 1D 6F 1F 6D 1C 6D 1C 75 1D 6E 1F 69 ; .t c.o.m.m.u.n.i
00000050h: 1C 63 61 74 69 6F 6E A0 6D 65 74 68 6F 64 2E 22 ; .cation method."
00000060h: 20 09 0D 0A 2F 2F 20 54 68 69 73 20 70 72 6F 67 ;  ...// This prog
00000070h: 72 61 6D 20 69 73 20 75 73 65 64 20 74 6F 20 63 ; ram is used to c
00000080h: 6F 6E 73 74 72 75 63 74 A0 74 68 65 20 63 6C 61 ; onstruct the cla
00000090h: 73 73 A0 43 50 77 64 64 6C 67 44 6F 63 20 61 6E ; ss CPwddlgDoc an
000000a0h: 64 A0 69 74 73 20 6D 65 6D 62 65 72 20 66 75 6E ; d its member fun
000000b0h: 63 74 69 6F 6E 73 2E 0D 0A 2F 2F 20 44 61 74 65 ; ctions...// Date
000000c0h: 3A A0 32 30 30 38 30 32 30 35 20 20 20 20 20 20 ; : 20080205
000000d0h: 20 20 0D 0A 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ;   ..////////////
000000e0h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
000000f0h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000100h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000110h: 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F ; ////////////////
00000120h: 2F 2F 0D 0A 20 20 20 20 20 20 20 20 20 20 0D 0A ; //..          ..
00000130h: 23 69 6E 63 6C 75 64 65 20 22 73 74 64 61 66 78 ; #include "stdafx
00000140h: 2E 68 22 20 20 20 20 20 20 20 20 0D 0A 23 69 6E ; .h"        ..#in
00000150h: 63 6C 75 64 65 20 22 70 77 64 64 6C 67 2E 68 22 ; clude "pwddlg.h"
00000160h: 20 20 20 20 20 20 20 20 0D 0A 23 69 6E 63 6C 75 ;         ..#inclu
00000170h: 64 65 20 22 70 77 64 64 6C 67 44 6F 63 2E 68 22 ; de "pwddlgDoc.h"
00000180h: 20 20 20 09 09 20 09 0D 0A 20 20 20 20 20 20 20 ;    .. ...
00000190h: 20 20 20 0D 0A 23 69 66 64 65 66 20 5F 44 45 42 ;    ..#ifdef _DEB
000001a0h: 55 47 20 20 20 20 20 20 20 20 0D 0A 23 64 65 66 ; UG        ..#def
```

For Help, press F1 | Pos: 36H, 54, C0 | DOS | C/C++ | Mod: 4/20/2008 8:07:24 PM | File Size: 1319 | INS

(a) A modified stego-program of Fig. 1(b).



Covert Communication

```
// This program is used to construct the class CPwddlgDoc and its member functions.
// Date: 20080205
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "pwddlg.h"
#include "pwddlgDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////
// CPwddlgDoc construction/destruction

CPwddlgDoc::CPwddlgDoc()
{
        // TODO: add one-time construction code here
}

CPwddlgDoc::~CPwddlgDoc()
{
}
/////////////////////////////////////////////////////////////////////////////
```

Authentication code 253156
Secret code 5847885168

ERROR

The message has been modified.

(b) A warning message issued by authentication process.

Fig. 2 An example of authentication results.

Table 1. Invisible codes under various environments.

Environment	**Null codes**	**Between-word spacing codes**	**Line-end spacing codes**
Type 1: MVS under English OS	1C-1F	A0	09
Type 2: BCB under English OS	None	A0	09
Type 3: MVS under Chinese OS	1C-1F	01-08, 0B-0F, 80	09, 0B, 0C
Type 4: BCB under Chinese OS	1C-1F, 80	01-08, 0B-19, 1B	09, 0B, 0C