Journal of Computer Science 5 (11): 890-897, 2009
ISSN 1549-3636


# Challenges of Hidden Data in the Unused Area Two within Executable Files

A.W. Naji, A.A. Zaidan and B.B. Zaidan
Department of Electrical and Computer Engineering, Faculty of Engineering,
International Islamic University Malaysia, 53100 Gombak, Kuala Lumpur, Malaysia

**Abstract: Problem statement:** The executable files are one of the most important files in operating systems and in most systems designed by developers (programmers/software engineers), and then hiding information in these file is the basic goal for this study, because most users of any system cannot alter or modify the content of these files. There are many challenges of hidden data in the unused area two within executable files, which is dependencies of the size of the cover file with the size of hidden information, differences of the size of file before and after the hiding process, availability of the cover file after the hiding process to perform normally and detection by antivirus software as a result of changes made to the file. **Approach:** The system designed to accommodate the release mechanism that consists of two functions; first is the hiding of the information in the unused area 2 of PE-file (exe.file), through the execution of four process (specify the cover file, specify the information file, encryption of the information, and hiding the information) and the second function is the extraction of the hiding information through three process (specify the steno file, extract the information, and decryption of the information). **Results:** The programs were coded in Java computer language and implemented on Pentium PC. The designed algorithms were intended to help in proposed system aim to hide and retract information (data file) with in unused area 2 of any execution file (exe.file). **Conclusion:** Features of the short-term responses were simulated that the size of the hidden data does depend on the size of the unused area2 within cover file which is equal 20% from the size of exe.file before hiding process, most antivirus systems do not allow direct write in executable file, so the approach of the proposed system is to prevent the hidden information to observation of these systems and the exe.file still function as usual after the hiding process

**Key words:** Cryptography Vs steganography, hidden data within executable file

## INTRODUCTION

The hurried development of multimedia and internet allows wide distribution of digital media data. It becomes much easier to edit, modify and duplicate digital information. In additional, digital document is also easy to copy and distribute, therefore it may face many threats. It became necessary to find an appropriate protection due to the significance, accuracy and sensitivity of the information. Nowadays, protection system can be classified into more specific as hiding information (Steganography) or encryption information (Cryptography) or a combination between them[1]. Cryptography is the practice of 'scrambling' messages so that even if detected, they are very difficult to decipher. The purpose of Steganography is to conceal the message such that the very existence of the hidden is 'camouflaged'[1].

However, the two techniques are not mutually exclusive. Steganography and cryptography are in fact complementary techniques. No matter how strong algorithm, if an encrypted message is discovered, it will be subject to cryptanalysis[1,4]. Likewise, no matter how well concealed a message is, it is always possible that it will be discovered. By combining Steganography with Cryptography we can conceal the existence of an encrypted message, in doing this; we make it far less likely that an encrypted message will be found[1,5].

Also, if a message concealed through Steganography is discovered, the discoverer is still faced with the formidable task of deciphering it. Also the strength of the combination between hiding and encryption science is due to the non-existence of standard algorithms to be used in (hiding and encryption) secret messages. Also there is randomness in hiding methods such as combining several media (covers) with different methods to pass a secret message. Furthermore, there is no formal method to be followed to discover a hidden data[1].

**Corresponding Author:** A.W. Naji, Department of Electrical and Computer Engineering, Faculty of Engineering, International Islamic University Malaysia, Post Code: 53100, 53100 Gombak, Kuala Lumpur, Malaysia

It is general in hiding information to embed a limited amount of information in media such as image and music files, so the embedding methods could be under surveillance from system managers in an organization that requires the high level of security. This fact requires researches on new hiding techniques and cover objects which hidden information is embedded in? It is the result from the researches to embed information in executable files. It can be considered that Stilo and Hydan represent common techniques for the embedding information in executable files[2,3]. These techniques make original files modified, so code signing techniques that guarantee the integrity of the code can be used for the detecting hidden information[2,3]. But, it becomes general phenomenon making executable files as the level of using computer and computing environment has been raised. In addition, it has not been general to use code signing techniques[2,3]. Silo and Hydan modify program binaries that have been in optimization, so the performance of the program binaries may fall. In addition, the amount of information to be embedded in executable files by using Silo and Hydan is limited under than 15% from the total cover size because these tools determine the number of bytes to be hidden on the foundation of the size of the program binaries[2,3]. Aws and bilal modify program, they implement a system computation between cryptography and steganography which embeds information within unused area 1 of exe.file[4]. Flowing that Ahmed and aws modify a new technique of hidden data in the (Unused Area 2 within exe.file) using computation Between Cryptography and Steganography that files[5].

These research aims are find a secure solution of cover file without change the size of cover file[4,5]. But these research never mentions about the detection by anti-virus, the functionality of the exe.file is still functioning or not and the size of the information hiding still limited and not known percentage. These points consider main challenges for hidden data in the executable files. Therefore, it needs to carry out researches on new hiding techniques that consider the efficiency of program using computation between cryptography and steganography. To process all the challenges of the executable file when be used for cover. In this study, we examine the new methods that consider the efficiency, the amount of information to be hidden and make sure changes made to the exe.file will not be detected by anti-virus and the functionality of the exe.file is still functioning. Furthermore we discuss the analysis techniques which can be applied to detect and recover data hidden using each of these methods.

## MATERIALS AND METHODS

**System concept:** Concept of this system can be summarized as hiding the password or any information beyond the end of an executable file so there is no function or routine (open-file, read, write and close-file) in the operating system to extract it. This operation can be performed in two alternative methods.

Building the file handling procedure independently of the operating system file handling routines. In this case we need canceling the existing file handling routines and developing a new function which can perform our need, with the same names. This way needs the customer to install the system application manually as shown in Fig. 2.

Development the file handling functions depending on the existing file handling routines. This way can be performed remotely as shown in Fig. 3. The advantage of the first method is it doesn't need any additional functions, which can be identified by the analysts.

The disadvantage of this method is it needs to be installed (can not be operated remotely). The advantage of the second method is it can be executed remotely and suitable for networks and the internet applications. So we choose this concept to implementation in this study.

**System features:** This system has the following features:

- The hiding operation of (unused area 2 within exe. File) increases the degree of security of hiding technique which is used in the proposed system because within unused area 2 of exe.file, it have different size from one fie to another, So the attacker cannot be attack the information hidden
- The cover file can be executed normally after hiding operation. Because the hidden information already hide in the unused area 2 within exe.file and thus cannot be manipulated as the exe.file, therefore, the cover file still natural, working normally and not effected, such as if the cover is exe.file (WINDOWES XP SETUP) after hiding operation it'll continued working, In other words, the exe.file can be installed of windows
- It's very difficult to extract the hidden information it's difficult to find out the information hiding, that is because of three reasons:
  - The information hiding will be encrypted before hiding of the information by AES method; this method very strong, 128-bit key would be in theory being in range of a military budget within 30-40 years. An illustration of

the current status for AES is given by the following example, where we assume an attacker with the capability to build or purchase a system that tries keys at the rate of one billion keys per second. This is at least 1 000 times faster than the fastest personal computer in 2004. Under this assumption, the attacker will need about 10 000 000 000 000 000 000 000 years to try all possible keys for the weakest version

- The information hiding should be decrypted after retract of the information

- Virus detection programmers' can't detect such as files, the principle of antivirus check are checking from beginning to end. When checking the exe.files by antivirus, will checked it from beginning to end of it, since the principle of information hiding for that system within unused area 2 of exe.file. The information hiding will be encrypt after that it will be hidden, the antivirus discontinue checking in the unused area 2 of exe.file after hiding process because the unused area 2 still empty so didn't mention to anything inside the exe.file while doing scanning.

**The proposed system structure:** To protect the hidden information from retraction the system encrypts the information by the built-in encryption algorithm provided by the Java. The algorithm for hiding operation procedure is shown in Fig. 1. The algorithm for retract operation procedure is shown in Fig. 2.

**Testing of the system:** There are two fundamental approaches to identifying test cases, these are known as functional and structure testing, each of these approaches has several distinct test case identification methods, more commonly called testing methods, functional testing is based on the view that any program can be considered to be a function that maps values from its input domain to values in its output range. (Function, domain and range) this notion is commonly used in engineering[6]. There are two distinct advantages to functional test cases, they are independent of how the software is implemented, so if the implementation changes, the test cases are still useful and test case development can occur in parallel with the implementation, thereby reducing overall research development interval, on other side, functional test cases frequently suffer from two problems: there can be significant redundancies among test cases and this is compounded by the possibility of gaps of untested software (Fig. 3)[6].

Procedure: Hide operation.
Input: Hidden file name, cover file name.
Output: Stego-File.

- Begin.
- Opens the cover file (EXE file).
- Assign a pointer to the end (MS-DOS 2.0 stub program & Relocation), which before the unused space 2 of the cover file.
- Write the hidden file name in the unused space 2 to the cover file
- Assign a pointer to (EXE file) after hidden file name.
- Encrypt the hidden file.
- Write the encrypt contact to the file cover (EXE file).
- End.

Fig. 1: Algorithm for hiding operation

Procedure: Retract operation.
Input: Stego-File.
Output: hidden information.

- Begin(1)
- Select the cover file.
- Get the end of (MS-DOS 2.0 stub program & Relocation) of EXE File.
- If end of (MS-DOS 2.0 stub program & Relocation) Pointer exists.
- Begin (2) read the name of hidden file.
- Read the hiding data.
- Decrypt the data using the file name as a key.
- Create a file using hiding file name.
- Write in to the create file the decrypt data.
- End (2).

  Else
- Display a message (no hiding file).
- End (1).

Fig. 2: Algorithm for retract operation

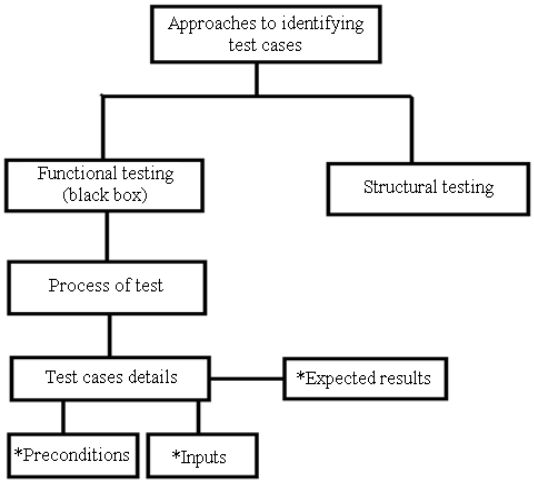


Fig. 3: Approaches to identifying test cases

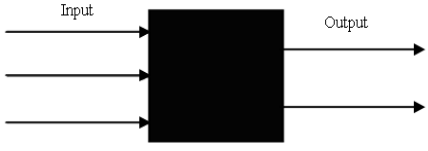


Fig. 4: Black box

When systems are considered to be "black boxes" test cases are generated and executed from the specification of the required functionality at defined interfaces, this leads to the function of the black box is understood completely in terms of its inputs and outputs, as shown in Fig. 4. Black-box testing has some important advantages[6]:

- It does not require that the code is seen, it is testing. Sometimes code will not be available in source code form, yet it can still construct useful test cases without it. The person writing the test cases does not need to understand the implementation
- The test cases do not depend on the implementation. They can be written in parallel with or before the implementation. Further, good black-box test cases do not need to be changed. Even if the implementation is completely rewritten
- Constructing black-box test cases causes the programmer to think carefully about the specification and its implications. Many specification errors are caught this way

The disadvantage of black box testing is that its coverage may not be as high as like, because it has to work without the implementation. But it is a good place to start when writing test cases, with the functional approach to test case identification; the only information that is used is the specification of the software[6].

**Process of the test:**

**Test case one:** In this phase making comparison between the cover files size after and before hiding operation.

**Test case two:** In this case making test for the usage of exe.files after the hiding operation to be done.

Four pictures approve the cover (exe.files) usage after the hiding operation and these pictures divides to:

- First picture of text
- Second picture of image
- Third picture of video
- Fourth picture of audio

**Test case three:** Testing for Scanning Result (undetectable by antivirus software).

Four pictures approve the cover (exe files) undetectable from antivirus software after the hiding operation and this picture divides to:

- First picture of text
- Second picture of image
- Third picture of video
- Fourth picture of audio

**Test cases details:** are known preconditions, inputs and expected results, which is worked out before the test is executed (Table 1). The definition of software installation needed for test an (Preconditions) and the definition inputs should needed for test an (inputs) and the definition predictable results for outputs an (except results).

**Preconditions:**

- Installation (Microsoft windows XP for any version or vista)
- Installation (Jcreators and JDK or net beans editor)
- Installation (Microsoft office word document 2003 or 2007)
- Installation (Software antivirus)
- Installation (Real player programmed)
- Installation (Jet audio programmed)
- Installation (ACDSEE programmed)
- System application for this research

**Inputs:** The system has two types of inputs:

- Inputs for cover (exe.files)
- Inputs for information hidden

Table 1: Inputs for test cases

| Name of input | Type of input | Size of inputs/bytes |
|---|---|---|
| Cover 1 | VMware player setup | 4,027,802 |
| Cover 2 | SSH | 532,480 |
| Cover 3 | JCreator editor setup | 22,806,060 |
| Cover 4 | 1 JDK setup | 68,830,616 |
| | 2 JDK setup | 76,445,080 |
| | 3 JDK setup | 81,208,728 |
| Text 1 | Word document | 805560,4 |
| Text 2 | Word document | 106496 |
| Text 3 | Word document | 4561212 |
| Text 4 | Word document | 13766123,2 |
| Video 1 | Real player | 805560,4 |
| Video 2 | Real player | 106496 |
| Video 3 | Real player | 4561212 |
| Video 4 | Real player | 15289016 |
| Audio 1 | Jet audio | 805560,4 |
| Audio 2 | Jet audio | 106496 |
| Audio 3 | Jet audio | 456122 |
| Audio 4 | Jet audio | 16241745,6 |
| Image 1 | JPEG | 805560,4 |
| Image 2 | JPEG | 106496 |
| Image 3 | JPEG | 4561212 |
| Image 4 | JPEG | 16241745,6 |

## RESULTS

**Expected results:**

- Secure cover (exe.files)
- The hidden information can be of any type of multimedia files dependent of the size of unused area 2 within cover file which is equal 20% from the size of exe.file
- These covers (exe.files) usage after the hiding operation
- These covers (exe.files) undetectable from antivirus software after the hiding operation

**Test case one:** In this test case can be shown Table 2 for cover files and information hidden before and after hiding operation of all types of multimedia files (text, image, audio and video),which related with this system, approve these covers (exe.files) are secure and there are no limitations on the hidden files size.

In Table 2 in test case one can be concluding:

- In the hidden files size inside the cover files can be hide different size inside the exe.files dependent of size of unused area 2 within exe.file which is equal 20% from the size of exe.file before hiding process
- The attacker can not attack the information hiding, because can not guess the exe.files size. The exe.files size does not have constant size, where it can be different size of the same type of exe.files like cover file number 4 they have three sizes in same type of the cover file

**Test case two:** In this test case shows picture of the cover files after hiding operation of all types of multimedia files in Fig. 5-8 (text, image, audio and video), which related with this system, approve these cover (exe.files) usage after the hiding operation.

**Test case three:** In this test case shows picture of cover files after hiding operation of all types of multimedia Files in Fig. 9-12 (text, image, audio and video), which related with this system, approve these covers (exe.files) undetectable from antivirus software after the hiding operation.

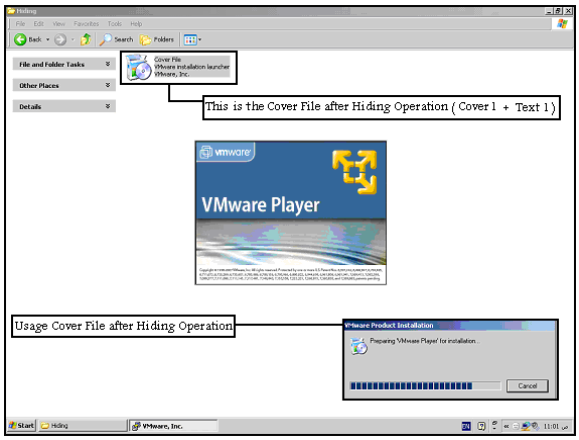


Fig. 5: Text: After hiding operation inside the (hiding folder), executable file (cover 1) still working

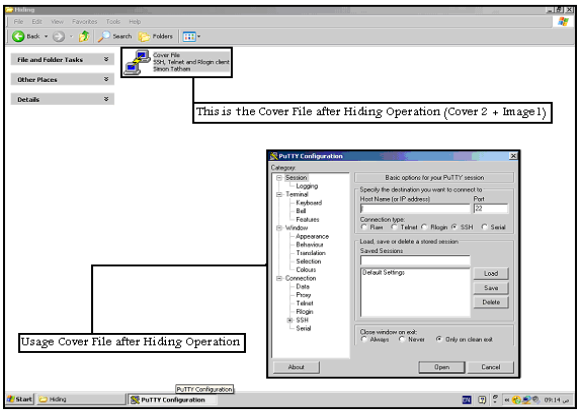


Fig. 6: Image: After hiding operation inside the (hiding folder), executable file (cover 2) still working

Table 2: Different size of the cover with different type of the exe.files and different size for the information of each type of multimedia files

| Before hide operation | | | | After hide operation | ((Size of cover after hide operation-size of cover befor hide operation)/size of cover before hide operation)*/100% |
|---|---|---|---|---|---|
| Information hidden | No. of cover | Size of IH/bytes | Size of cover/bytes | Size of cover/bytes | |
| Text 1 | 1 | 805560,4 | 4,024,802 | 7,027,807 | 0,00012 |
| Text 2 | 2 | 106496 | 532,480 | 532,486 | 0,00112 |
| Text 3 | 3 | 4561212 | 22,806,060 | 22,806,068 | 0,00003 |
| Text 4 | 4 | 13766123,2 | 68,830,616 | 68,830,619 | 0,000004 |
| Image 1 | 1 | 805560,4 | 4,024,802 | 4,027,980 | 0,002 |
| Image 2 | 2 | 106496 | 532,480 | 532,520 | 0,007 |
| Image 3 | 3 | 4561212 | 22,806,060 | 22,806,140 | 0,0003 |
| Image 4 | 4 | 16241745,6 | 76,445,080 | 76,445,150 | 0,00009 |
| Audio 1 | 1 | 805560,4 | 4,024,802 | 4,028,502 | 0,017 |
| Audio 2 | 2 | 106496 | 532,480 | 532,990 | 0,095 |
| Audio 3 | 3 | 456122 | 22,806,060 | 22,806,980 | 0,004 |
| Audio 4 | 4 | 16241745,6 | 81,208,728 | 81,208,959 | 0,0002 |
| Video 1 | 1 | 805560,4 | 4,024,802 | 4,037,802 | 0,248 |
| Video 2 | 2 | 106496 | 532,480 | 534,480 | 0,375 |
| Video 3 | 3 | 4561212 | 22,806,060 | 22,819,060 | 0,057 |
| Video 4 | 4 | 15289016 | 81,208,728 | 81,229,728 | 0,025 |

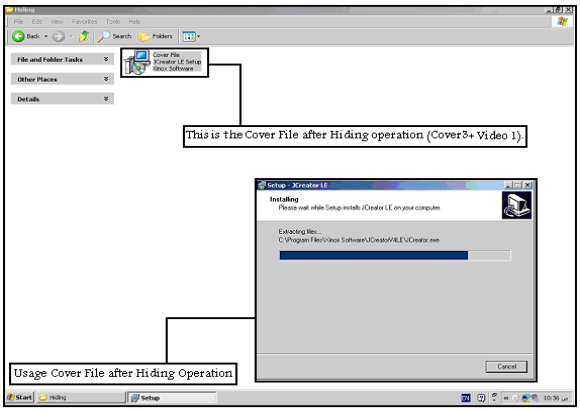


Fig. 7: Video: After hiding operation inside the (hiding folder), executable file (cover 3) still working

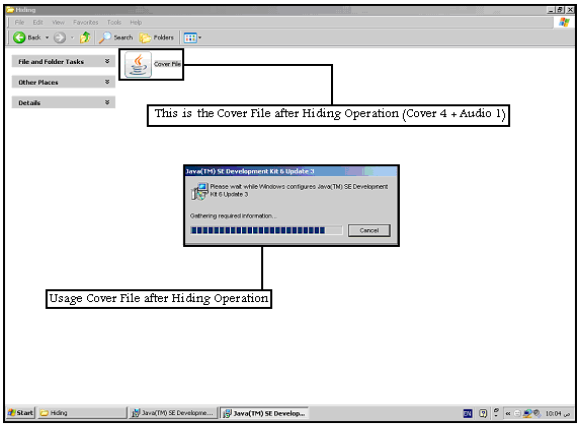


Fig. 8: Audio: After hiding operation inside the (hiding folder), executable file (cover 4) still working

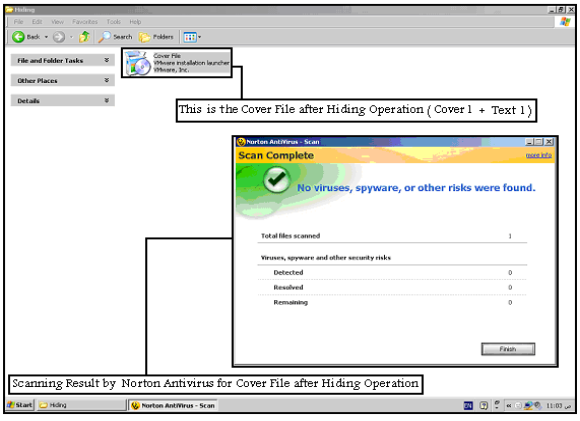


Fig. 9: Text: Shows that the executable file (cover 1) inside (hiding folder) immune to anti-virus program

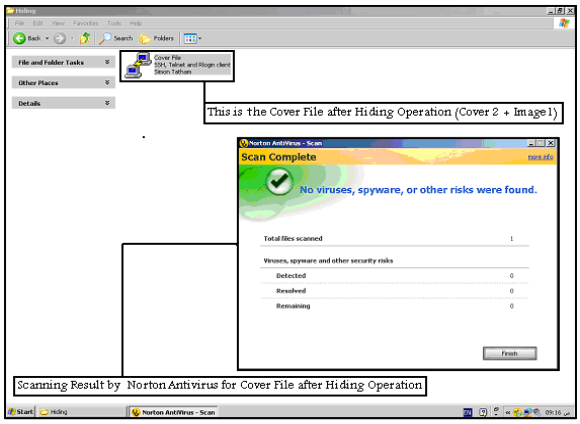


Fig. 10: Image: Shows that the executable file (cover 2) file inside (hiding folder) undetectable by anti-virus program

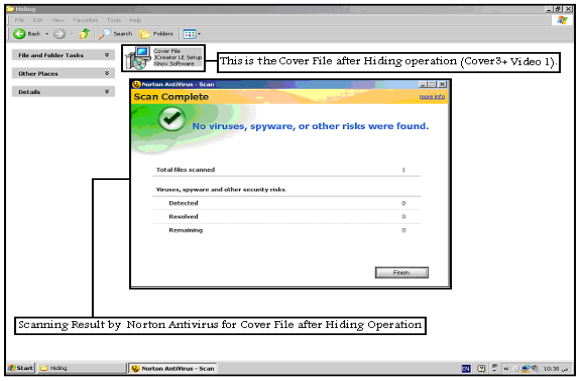


Fig. 11: Video: Shows that the executable file (cover 3) inside (hiding folder) immune to anti-virus program

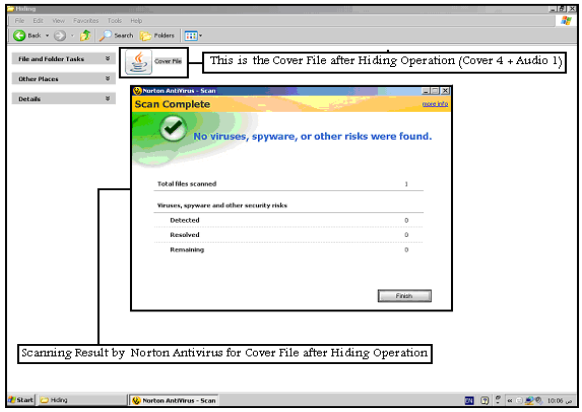


Fig. 12: Audio: Shows that the executable file (cover 4) file inside (hiding folder) immune to anti-virus program

**Evaluation of the system:**

- The size of the hidden message dependent of the size of unused area 2 within cover files which is equal 20% from the size of exe.file before hiding process
- The executable files still working after its use as cover for embedding data
- The executable file undetectable from Norton antivirus software after the hiding operation
- The hiding method makes the relation between the cover and the message dependent the size of unused area 2 with in exe.file. So when the size of the cover exe.files upgrade, the secure is very height because in this case the exe.file have high size of unused area 2 and when the information hidden less inside the cover, the cover files in this case has been more secure.
- From the information which is shown in Table 3 and 4 concludes that:

The proportion of potential discovery of embedded data in:

Text→[((((text1+cover1)-cover1)/cover1) *100%)+((((text2+cover2)-cover2)/cover2) *100%)+((((text3+cover3)-cover3)/cover3) *100%)+((((text4+cover4)-cover4)/cover4)*100%)]/4

[(0.00012%)+(0.00112%)+(0.00003%)+(0.000004%)]/ 4 = 0.0003185%

Image →[((((image1+cover1)-cover1)/cover1) *100%)+((((image2+cover2)-cover2)/cover2) *100%)+((((image3+cover3)-cover3)/cover3) *100%)+((((image4+cover4)-cover4)/cover4) *100%)]/4

[(0.002%)+(0.007%)+(0.0003%)+(0.00009%)]/4 = 0.0023475%

Audio →[((((audio1+cover1)-cover1)/cover1) *100%)+((((audio2+cover2)-cover2)/cover2) *100%)+((((audio3+cover3)-cover3)/cover3) *100%)+((((audio4+cover4)-cover4)/cover4)*100%)]/4

[(0.017%)+(0.095%)+(0.004%)+(0.0002%)]/4 = 0.02905%

Video →[((((video1+cover1)-cover1)/cover1) *100%)+((((video2+cover2)-cover2)/cover2) *100%)+((((video3+cover3)-cover3)/cover3) *100%)+((((video4+cover4)-cover4)/cover4)*100%)]/4

[(0.248%)+(0.375%)+(0.057%)+(0.025%)]/4= 0.17625%

Table 3: Inputs and outputs for test case two

| Before hiding operation | | After hiding operation |
|---|---|---|
| No. of cover | Information hidden | Usage the EXE covers |
| 1 | Text 1 | Fig. 5 |
| 2 | Image 1 | Fig. 6 |
| 3 | Video 1 | Fig. 7 |
| 4 | Audio 1 | Fig. 8 |

Table 4: Inputs and outputs for test case three

| Before hiding operation | | After hiding operation |
|---|---|---|
| No. of cover | Information hidden | Usage the EXE covers |
| 1 | Text 1 | Fig. 9 |
| 2 | Image 1 | Fig. 10 |
| 3 | Video 1 | Fig. 11 |
| 4 | Audio 1 | Fig. 12 |

The percentage of success achieved by the innovative system:

Text 100%-0.0003185% = 99.9996815%
Image 100%-0.0023475% = 99.9976525%
Audio 100%-0.02905% = 99.97095%
Video 100%-0.17625% = 99.82375%

## CONCLUSION

The hiding information in exe file is the basic goal for this study, because most users of any system cannot alter or modify the content of these files. We get the following discussions:

- PE files structure is very complex because they depend on multi headers and addressing and then insertion of data to PE files without full understanding of their structure may damage them, so the choice is to hide the information beyond the structure of these files
- Most antivirus systems do not allow direct write in executable file, so the approach of the proposed system is to prevent the hidden information to observation of these systems
- One of the important discussion point in implementation of the proposed system is the solving of the problems that are related to the size of cover file, so the hiding method makes the relation between the cover and the message dependent of the size of unused area 2 within cover file files which is equal 20% from the size of exe.file before hiding process
- The encryption of the message increases the degree of security of hiding technique which is used in the proposed system
- The proposed hiding technique is flexible and very useful in hiding any type of data for files message (text, image, sound or video)

**ACKNOWLEDGEMENT**

Our sincere thanks to all researchers who have contribute to this project. Also we would like to acknowledge and thanks the researchers in UM for their support.

**REFERENCES**

1. Zaidan, A.A., B.B. Zaidan, M.M. Abdulrazzaq, R.Z. Raji and S.M. Mohammed, 2009. Implementation stage for high securing cover-file of hidden data using computation between cryptography and steganography. Int. Assoc. Comput. Sci. Inform. Technol., 20, Session 6, p.p 482-489.. http://WWW.IACSIT.ORG and Www.WordAcademicPress.com
2. El-Khalil, R. and A.D. Keromytis, 2004. Hydan: Hiding information in program binaries. Proceedings of the 6th International Conference on Information and Communications Security, Oct. 27-29, Springer Berlin, Heidelberg, pp: 187-199. http://cat.inist.fr/?aModele=afficheN&cpsidt=16334236
3. Anckaert, B.B. De Sutter, D. Chanet and K. De Bosschere, 2005. Steganography for executable and code transformation signatures. Proceedings of the 7th Information Security and Cryptology, May 24, Springer Berlin, Heidelberg, pp: 425-439. http://www.springerlink.com/content/vbxjdapj9g25agel/
4. Zaidan, B.B., Zaidan.A.A, F. Othman and A. Rahem, 2009. Novel approach of hidden data in the unused area 1 within exe files using computation between cryptography and steganography. Proceeding of the International Conference on Cryptography, Coding and Information Security, June 24-26, Academic and Scientific Research Organizations, Paris, France, pp: 1-22. http://www.waset.org/programs/Paris09.pdf
5. Naji, A.W., Zaidan.A.A. Zaidan.B.B. A. Shihab and O.O. Khalifa, 2009. Novel approach of hidden data in the unused area 2 within exe file using computation between cryptography and steganography. Int. J. Comput. Sci. Network Secur., 9: 294-300. http://paper.ijcsns.org/07_book/200905/20090539.pdf
6. Muhamadi, I.A.S., Zaidan .M.A., Zaidan A.A and Zaidan.B.B, 2009. Student record retrieval system using knowledge sharing. Int. J. Comput. Sci. Network Secur., 9: 97-106. http://paper.ijcsns.org/07_book/200906/20090614.pdf