Novel Approach for Secure Cover File of Hidden Data in the Unused Area within EXE File Using Computation between Cryptography and Steganography

A.W. Naji*, A.A.Zaidan, B.B.Zaidan, Shihab A. Hameed and Othman O. Khalifa

Electrical and Computer Engineering Department, Faculty of Engineering,
International Islamic University Malaysia,
53100 Gombak, Kuala Lumpur,
Malaysia.

ABSTRACT

The strength of the information hiding science is due to the non-existence of standard algorithms to be used in hiding secret messages. Also there is randomness in hiding methods such as combining several media (covers) with different methods to pass a secret message. In addition, there are no formal methods to be followed to discover the hidden data. For this reason, the task of this paper becomes difficult. In this paper a new method is implementing to hide a file of unused area 2 within .EXE file and to detect the hidden file. The aim of this paper is implementation of system computation between Cryptography and Steganography which embeds information in unused area 2 within EXE files to find a secure solution to cover file without change the size of cover file. The system includes two main functions; first is the hiding of the information in 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) and The proposed system is implemented by using java.

Keyword

Cryptography, Steganography, portable Executable File, Advance Encryption Standard

I. STEGANOGRAPHY

Steganography is the art of concealing the presence of information within an innocuous container. Steganography has been used throughout history to protect important information from being discovered by enemies. A very early example of Steganography comes from the story of Demartus of Greece. He wished to inform Sparta that Xerces the King of Persia was planning to invade. In ancient Greece wax covered wooden tablets were used to record written text [1].In order to avoid detection by the Persians, Demartus scraped the wax from a tablet, etched the message into the underlying wood, then recovered the tabled with wax. This concealed the underlying message from the sentries who inspected the tablets as they left Persia by courier for Greece[1].Other historical examples of Steganography are the use of invisible inks. A common experiment conducted by young kids everywhere is to use a toothpick dipped in vinegar to write a message on a piece of paper. Once the vinegar dries, the presence of the message is not obvious to a casual inspector (aside from the smell). Upon slight heating of the paper, a chemical reaction occurs which darkens the vinegar and makes the message readable. Other, less smelly, invisible inks have been used throughout history similarly even up until World War II.A more recently developed Steganography technique was invented by the Germans in World War II, the use of microdots.Microdots were very small photographs, the size of a printed period, which contain very clear text when magnified. These microdots contained important information about German war plans and were placed in completely unrelated letters as periods. Although Steganography is related to Cryptography, the two are fundamentally different [1], [2].

A. Cryptography vs. Steganography

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’. However, the two techniques are not mutually exclusive [2]. 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. Likewise, no matter how well concealed a message is, it is always possible that it will be discovered [2]. 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. Also, if a message concealed through Steganography is discovered, the discoverer is still faced with the formidable task of deciphering it [2].

II. PORTABLE EXECUTABLE FILE (PE-FILE)

The proposed system uses a portable executable file as a cover to embed an executable program as an example for the proposed system. This section is divided into four parts[3],[4],[5]:

Manuscript received May 5, 2009
Manuscript revised May 20, 2009

- Executable file types.
- Concept related with PE-file.
- Techniques related with PE-file.
- PE-file Layout.

A. *Executable File Types*

The number of different executable file types is as many and varied as the number of different image and sound file formats. Every operating system seems to have several executable file types unique to it. These types are [5]:

- EXE (DOS"MZ")
- EXE (win 3.xx "NE")
- EXE (OS/2 "LE")
- EXE (win 9x/NT "PE")
- ELF

B. *Concepts Related With PE*

The addition of the Microsoft® windows NT™ operating system to the family of windows™ operating systems brought many changes to the development environment and more than a few changes to applications themselves. One of the more significant changes is the introduction of the Portable Executable (PE) file format. The name "Portable Executable" refers to the fact that the format is not architecture specific [6].In other words, the term "Portable Executable" was chosen because the intent was to have a common file format for all versions of Windows, on all supported CPUs [5].The PE files formats drawn primarily from the Common Object File Format (COFF) specification that is common to UNIX® operating systems. Yet, to remain compatible with previous versions of the MS-DOS® and windows operating systems, the PE file format also retains the old familiar MZ header from MS-DOS [6].The PE file format for Windows NT introduced a completely new structure to developers familiar with the windows and MS-DOS environments. Yet developers familiar with the UNIX environment will find that the PE file format is similar to, if not based on, the COFF specification [6].The entire format consists of an MS-DOS MZ header, followed by a real-mode stub program, the PE file signature, the PE file header, the PE optional header, all of the section headers, and finally, all of the section bodies [4].

C. *Techniques Related with PE*

Before looking inside the PE file, we should know special techniques some of which are [6]:

- General view of PE files sections

A PE file section represents code or data of some sort. While code is just code, there are multiple types of data. Besides read/write program data (such as global variables), other types of data in sections include application program interface (API) import and export tables, resources, and relocations.

Each section has its own set of in-memory attributes, including whether the section contains code, whether it's read-only or read/write, and whether the data in the section is shared between all processes using the executable file. Sections have two alignment values, one within the desk file and the other in memory.

The PE file header specifies both of these values, which can differ. Each section starts at an offset that's some multiple of the alignment value. For instance, in the PE file, a typical alignment would be 0x200. Thus, every section begins at a file offset that's a multiple of 0x200.Once mapped into memory, sections always start on at least a page boundary. That is, when a PE section is mapped into memory, the first byte of each section corresponds to a memory page. On x86 CPUs, pages are 4KB aligned, while on the Intel Architecture IA-64, they're 8KB aligned.

- Relative Virtual Addresses (RVA)

In an executable file, there are many places where an in-memory address needs to be specified. For instance, the address of a global variable is needed when referencing it. PE files can load just about anywhere in the process address space. While they do have a preferred load address, you can't rely on the executable file actually loading there. For this reason, it's important to have some way of specifying addresses that are independent of where the executable file loads. To avoid having hard coded memory addresses in PE files, RVAs are used. An RVA is simply an offset in memory, relative to where the PE file was loaded. For instance, consider an .EXE file loaded at address 0x400000, with its code section at address 0x401000. The RVA of the code section would be:

(Target address) 0x401000 – (load address) 0x400000 = (RAV) (1)

To convert an RVA to an actual address, simply reverse the process: add the RVA to the actual load address to find the actual memory address. Incidentally, the actual memory address is called a Virtual Address (VA) in PE parlance. Another way to think of a VA is that it's an RVA with the preferred load address added in.

- Importing Functions

When we use code or data from another DLL, we're importing it. When any PE files loads, one of the jobs of the windows loader is to locate all the imported functions and data and make those addressees available to the file being loaded.

D. PE File Layout

The PE file layout is shown in Figure 1. There are two unused spaces in PE file layout [7], and these unused spaces are suggested to hide a watermark.The size of the second unused space is different from one file to another [7].The most important reason behind the idea of this system is that the programmers always need to create a back door for all of their developed applications, as a solution to many problems such that forgetting the password. This idea leads the customers to feel that all programmers have the ability to hack their system any time. At the end of this discussion all customers always are used to employ trusted programmers to build their own application. Programmers want their application to be safe any where without the need to build ethic relations with their customers. In this system a solution is suggested for this problem [6],[8].

The solution is to hide the password in the executable file of the same system and then other application to be retracted by the customer himself. Steganography needs to know all files format to find a way for hiding information in those files. This technique is difficult because there are always large numbers of the file format and some of them have no way to hide information in them [8].

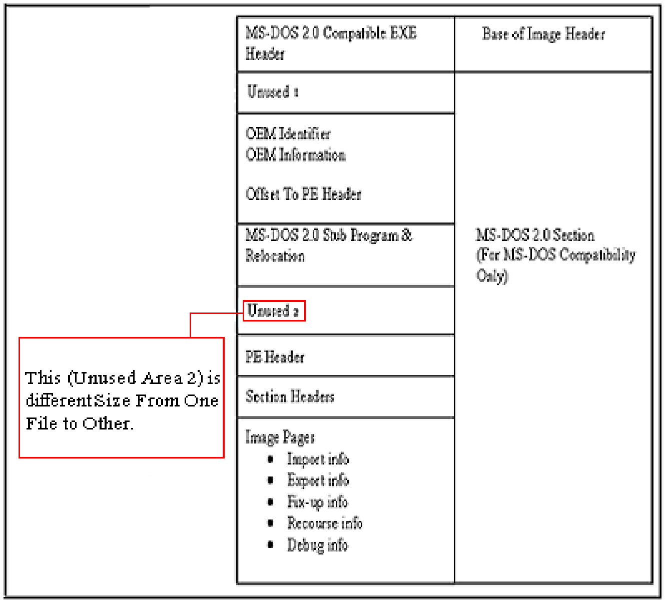

MS-DOS 2.0 Compatible EXE Header
Base of Image Header
Unused 1
OEM Identifier
OEM Information
Offset To PE Header
MS-DOS 2.0 Stub Program & Relocation
MS-DOS 2.0 Section (For MS-DOS Compatibility Only)
Unused 2
This (Unused Area 2) is different Size From One File to Other.
PE Header
Section Headers
Image Pages
Import info
Export info
Fix-up info
Recourse info
Debug info

Figure 1.Typical 32-bit Portable .EXE File Layout.

III. ADVANCE ENCRYPTION STANDARD

In 1997, the NIST called for submissions for a new standard to replace the aging DES. The contest terminated in November 2001with the selection of the Rijndael cryptosystem as the Advanced Encryption Standard (AES) [1],[2]. The Rijndael cryptosystem operates on 128-bit blocks, arranged as 4 × 4 matrices with 8-bit entries. The algorithm can use a variable block length and key length. The latest specification allowed any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits.
AES may, as all algorithms, be used in different ways to perform encryption. Different methods are suitable for different situations. It is vital that the correct method is applied in the correct manner to each and every situation, or the result may well be insecure even if AES as such is secure. It is very easy to implement a system using AES as its encryption algorithm, but much more skill and experience are required to do it in the right way for a given situation. To describe exactly how to apply AES for varying purposes is very much out of scope for this short introduction.

A. Strong keys

Encryption with AES is based on a secret key with 128, 192 or 256 bits. But if the key is easy to guess it doesn’t matter if AES is secure, so it is as critically vital to use good and strong keys as it is to apply AES properly. Creating good and strong keys is a surprisingly difficult problem and requires careful design when done with a computer. The challenge is that computers are notoriously deterministic, but what is required of a good and strong key is the opposite unpredictability and randomness. Keys derived into a fixed length suitable for the encryption algorithm from passwords or pass phrases typed by a human will seldom correspond to 128 bits much less 256. To even approach 128- bit equivalence in a pass phrase, at least 10 typical passwords of the kind frequently used in day-to-day work are needed. Weak keys can be somewhat strengthened by special techniques by adding computationally intensive steps which increase the amount of computation necessary to break it. The risks of incorrect usage, implementation and weak keys are in no way unique for AES; these are shared by all encryption algorithms. Provided that the implementation is correct, the security provided reduces to a relatively simple question about how many bits the chosen key, password or pass phrase really corresponds to. Unfortunately this estimate is somewhat difficult to calculate, when the key is not generated by a true random generator [3].

B. The Round Transformations

There are four transformations:

- Add Round Key

Add Round Key is an XOR between the state and the round key. This transformation is its own inverse [4].

- Sub Bytes

Sub Bytes is a substitution of each byte in the block independent of the position in the state. This is an S-box. It is bisection on all possible byte values and therefore invertible (the inverse S-box can easily be constructed from the S-box). This is the non-linear transformation. The S-box used is proved to be optimal with regards to non-linearity. The S-box is based on arithmetic in GF (2^8).

- Shift Rows

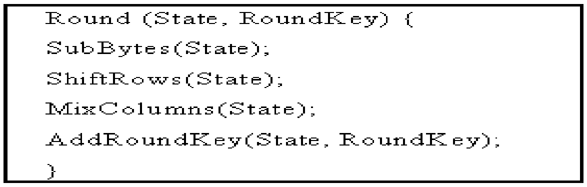
Shift Rows is a cyclic shift of the bytes in the rows in the state and is clearly invertible (by a shift in the opposite direction by the same amount) [5].

- Mix Columns

Each column in the state is considered a polynomial with the byte values as coefficients. The columns are transformed independently by multiplication with a special polynomial c(x). c(x) has an inverse d(x) that is used to reverse the multiplication by c (x) [1].

C. The Rounds

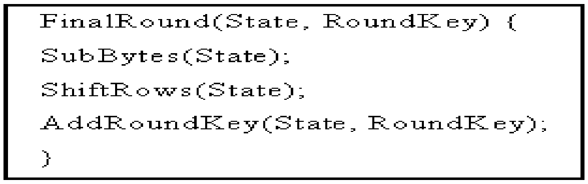
A round transformation is composed of four different transformations as shown in fig.2. The Round keys are made by expanding the encryption key into an array holding the Round Keys one after another. The expansion works on words of four bytes. Nk is a constant defined as the number of four bytes words in the key. The encryption key is filled into the first Nk words and the rest of the key material is defined recursively from preceding words. The word in position i, W[i], except the first word of a Round Key, is defined as the XOR between the preceding word, W[i-1], and W[i-Nk]. The first word of each Round Key, W[i] (where i mod Nk == 0), is defined as the XOR of a transformation on the preceding word, T (W [i - 1]) and W [i - Nk]. The transformation T on a word, w, is w rotated to the left by one byte, XOR'ed by a round constant and with each byte substituted by the S-box [5].



```
Round (State, RoundKey) {
SubBytes(State);
ShiftRows(State);
MixColumns(State);
AddRoundKey(State, RoundKey);
}
```

Figure 2. Four Different Transformations.

The final round is like a regular round, but without the mix columns transformation as shown in fig. 4:



```
FinalRound(State, RoundKey) {
SubBytes(State);
ShiftRows(State);
AddRoundKey(State, RoundKey);
}
```

Figure 3. Final Round.

IV. STEGANOGRAPHY TYPES

As it is known there is much communication between people and organizations through the use of the phone, the fax, computer communications, radio, and of course all of these communication should be secure. There are basically three Steganography types [8]:-

- Pure Steganography.
- Public key Steganography.
- Secret key Steganography.

a) Pure Steganography

Pure Steganography is the Steganography system that doesn't require prior exchange of some secret information before sending message; therefore, no information is required to start the communication process: the security of the system thus depends entirely on its secrecy. The pure Steganography can be defined as the quadruple(C, M, D, and E) where:

C: the set of possible covers.
M: the set of secret massage with |C| ≥ |M|.
E: C×M→C the embedding function.
D: C→M of the extraction function with the property that
D: (E(c, m)) =m for all m Є M and c Є C.

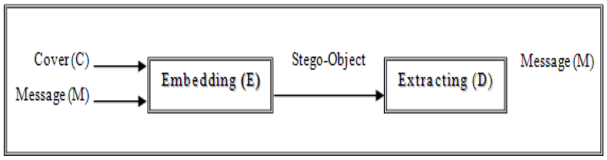


Figure 4. Pure Steganography

In most applications, pure Steganography is preferred, since no stego-key must be shared between the communication partners, although a pure Steganography protocols don't provide any security if an attacker knows the embedding method.

b) Public key Steganography

Public key Steganography does not depend on the exchange of a secret key. It requires two keys, one of them private (secret) and the other public: the public key is stored in a public database, whereas the public key is used in the embedding process. The secret key is used to reconstruct the secret message. One way to build a public key Steganography system is to use a public key crypto system. The sender and the receiver can exchange public keys of some public key cryptography algorithm before imprisonment. Public key Steganography utilizes the fact that the decoding function in a Steganography system can be applied to any cover, whether or not it already contains a secret message. The public key Steganography relies on the fact that encrypted information is random enough to hide in plain sight. The sender encrypts the information with the receiver's public key to obtain a random-looking massage and embeds it in a channel known to the receiver, thereby replacing some of the natural randomness with which every communication process is accompanied. Assume that both the cryptographic algorithms and the embedding functions are publicly known. The receiver who cannot decide a priori if secret information is transmitted in a specific cover will suspect the arrival of message and will simply try to extract and decrypt it using his private key. If the cover actually contained information, the decryption information is the sender's message.

c) Secret key Steganography

A secret key Steganography system is similar to a symmetric cipher, where the sender chooses a cover and embeds the secret message into the cover using a secret key. If the secret key used in the embedding process is known to the receiver, he can reverse the process and extract the secret message. Anyone who doesn't know the secret key should not be able to obtain evidence of the encoded information.

The secret key Steganography can be defined as the quintuple (C, M, K, DK, and EK) where:

C: the set of possible covers.

M: the set of secret message.

K: the set of secret keys.

$Ek: C\times M\times K\rightarrow C$

With the property that $DK(EK(c,m,k),k)=m$ for all $m \in M$, $c \in C$ and $k \in K$

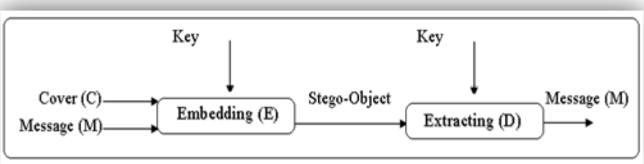


Figure 5. Secret Key Steganography

The final scenario for the algorithm as fallowed below in the figure 6, where Kenny hide and encrypt a secret data and send this data to cartman, cartman will decrypt the data and extract the data again

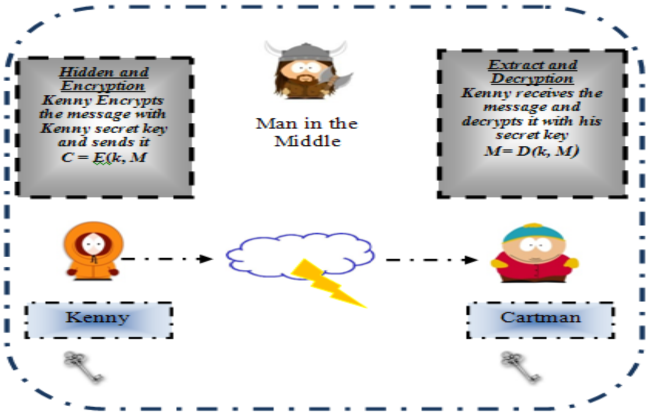

Hidden and Encryption
Kenny Encrypts the message with Kenny secret key and sends it
C = E(k, M
Man in the Middle
Extract and Decryption
Kenny receives the message and decrypts it with his secret key
M= D(k, M)
Kenny
Cartman

Figure 6. send and receive massages through the two ends, its shows how Kenny encrypt the massage and send it to Cartman , the figure above shows also the man in the middle waiting to have any information may help to decrypts the massages.

V. METHODOLOGY

A. 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. The advantage of these methods is it doesn't need any additional functions, which can be identified by the analysts. And it can be executed remotely and suitable for networks and the internet applications .The disadvantage of these methods is it needs to be installed (can not be operated remotely). So we choose this concept to implementation in this paper.

B. 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 the size of cover file doesn't change and the unused area 2 they have different size from one fie to another, So the attacker can not 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.Files (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 was 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

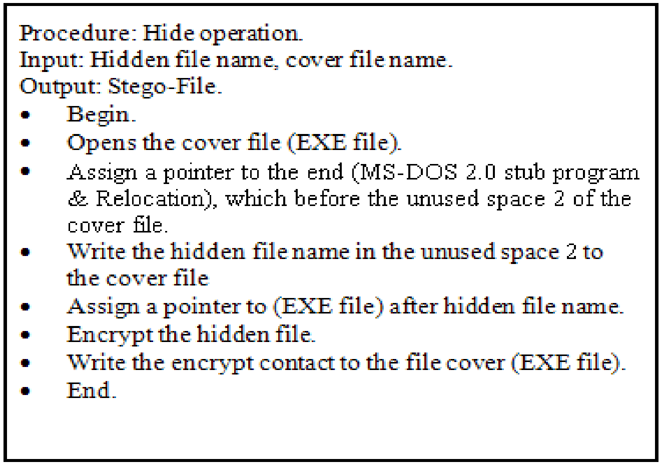
at the rate of one billion keys per second. This is at least 1 000 times faster than the fastest personal computer in 2004. Under this assumption, the attacker will need about 10 000 000 000 000 000 000 000 years to try all possible keys for the weakest version.

- The attacker impossible guessing the information hiding inside the EXE file because of couldn't guessing the real size of (EXE file and information hiding).
- The information hiding should be decrypted after retract of the information.

C. *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 following algorithm for hiding operation procedure as shown in Figure 7. The following algorithm for Retract operation procedure as shown in Figure 8.

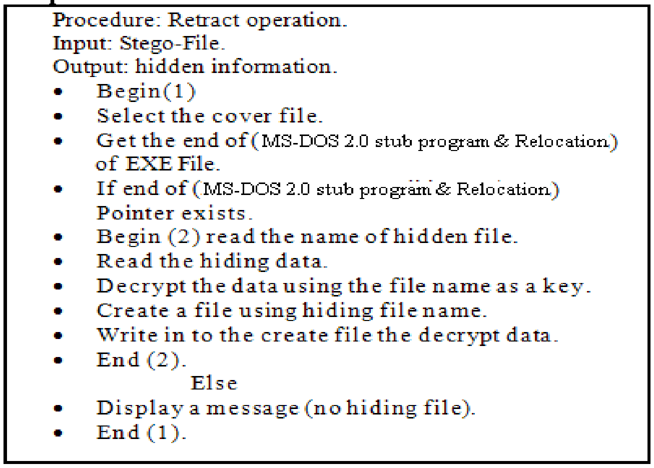
1. The following algorithm is the hiding operation procedure:



```
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.
```

Figure 7. Shows Algorithm for Hiding Operation.

2. The following algorithm is retraction operation procedure:



```
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).
```

Figure 8. Shows Algorithm for Retract Operation.

VI. CONCLUSION

One of the important conclusions in implementation of the proposed system is the solving of the problems that are related to the change size of cover file, so the hiding method makes the relation between the cover and the message dependent without change of cover file and The encryption of the message increases the degree of security of hiding technique which is used in the proposed system and 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 , finally The system has achieved the main goal, makes the relation of the size of the cover file and the size of information dependent without change the size of cover file , so There is no change on the cover file size where you can hide file of Unused area 2 within portable executable file by Structure on the property of the EXE file and The proposed system is implemented by using Java.

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] A.A.Zaidan, B.B.Zaidan, M.M.Abdulrazzaq, R.Z.Raji and S.M.Mohammed, “Implementation Stage for High Securing Cover-File of Hidden Data Using Computation between Cryptography and Steganography”. International Association of Computer Science and Information Technology (IACSIT), indexing by Nielsen, Thomson ISI (ISTP), IACSIT Database, British Library and EI Compendex, Volume 20, 2009, Manila, Philippines.

[2] B.B Zaidan, A.A Zaidan, Fazidah Othman, R.Z.Raji, S.M.Mohammed, M.M.Abdulrazzaq, “Quality of Image vs. Quantity of Data Hidden in the Image”, International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'09), 2009, Las Vigas, USA.

[3] B.B.Zaidan, A.A.Zaidan, Fazidah Othman “Enhancement of the Amount of Hidden Data and the Quality of Image", Malaysia Education Security (MyEduSec08), Grand Continental Hotel, 2008, Kuala Trengano, Malaysia

[4] Avedissian, L.Z," Image in Image Steganography System”, Ph.D.Thesis, Informatics Institute for Postgraduate Studies (IIIPS), University of Technology, Baghdad, Iraq, 2008.

[5] C. J. S. B,” Modulation and Information Hiding in Images”, of Lecture Notes in Computer Science, University of Technology, Malaya, Vol. 1174, pp.207-226, 2007.

[6] Clelland, C.T.R, V.P & Bancroft, “ Hiding Messages in DNAMicroDots ” , International Symposium on Industrial Electronics (ISIE) , University of Indonesia , Indonesia, Vol. 1, pp.315-327, 2007.

[7] Davern, P.S, M.G, “Steganography It History and Its Application to Computer Based Data Files”, School of Computer Application (SCA), Dublin City University. Working Paper. Studies (WPS), Baghdad, Iraq, 2007.

[8] Dorothy, E.R, D.K, "Cryptography and Data Security", IEEE International Symposium on Canada Electronics (ISKE), University of Canada, Canada, Vol.6, pp.119-122, 2006,

AUTHORS INFORMATION

Dr. Ahmed Wathik Naji - He obtained his 1st Class Master degree in Computer Engineering from University Putra Malaysia followed by PhD in Communication Engineering also from University Putra Malaysia. He supervised many postgraduate students and led many funded research projects with more than 50 international papers. He has more than 10 years of industrial and educational experience. He is currently Senior Assistant Professor, Department of Electrical and Computer Engineering, International Islamic University Malaya, Kuala Lumpur, Malaysia.

Aos Alaa Zaidan - has received his master from Department of Computer System & Technology Department Faculty of Computer Science and Information Technology/University of Malaya /Kuala Lumpur/Malaysia, his research interest on Steganography & Quantum Cryptography with his group he has published many papers on data hidden through different multimedia carriers such as image, video, audio, text, and non multimedia careers such as unused area within EXE file, he has done projects on Stego-Analysis systems, currently he is working on Quantum Key Distribution QKD, his PhD on the Tracking System.

Bilal Bahaa Zaidan - has received his master from Department of Computer System & Technology Department Faculty of Computer Science and Information Technology/University of Malaya /Kuala Lumpur/Malaysia, his research interest on Steganography & Cryptography with his group he has published many papers on data hidden through different multimedia carriers such as image, video, audio, text, and non multimedia careers such as unused area within EXE file, he has done projects on Stego-Analysis systems, currently he is working on Quantum Key Distribution QKD, his PhD work on develop a new cryptography standard.

Dr. Shihab A Hameed - He obtained his PhD in software Engineering from UKM. He has three decades of industrial and educational experience. His research interest is mainly in the software engineering, software quality, surveillance and monitoring systems, health care and medication. He supervised numerous funded projects and has published more than 60 papers at various international and national conferences and journals. He is currently Senior Assistant Professor, Department of Electrical and Computer Engineering, International Islamic University Malaysia. Malaya, Kuala Lumpur.

Othman O. Khalifa received his Bachelor's degree in Electronic Engineering from the Garyounis University, Libya in 1986. He obtained his Master degree in Electronics Science Engineering and PhD in Digital Image Processing from Newcastle University, UK in 1996 and 2000 respectively. He worked in industrial for eight years and he is currently an Associate Professor and Head of the department of Electrical and Computer Engineering, International Islamic University Malaysia. His area of research interest is Communication Systems, Information theory and Coding, Digital image / video processing, coding and Compression, Wavelets, Fractal and Pattern Recognition. He published more than 130 papers in international journals and Conferences. He is SIEEE member, IEEE computer, Image processing and Communication Society member.