International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010
1793-8201

A New System for Hiding Data within (Unused Area Two + Image Page) of Portable Executable File using Statistical Technique and Advance Encryption Standared

A.A.Zaidan, B.B.Zaidan, Hamid.A.Jalab

Abstract— **The internet and the World Wide Web have revolutionalized the way in which digital data is distributed. The widespread and easy access to multimedia content has motivated development of technologies for steganography or data hiding, so 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 encrypting secret messages. Also there are many ways 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. For this reason, the task of this paper becomes difficult. In this paper proposed a new system of information hiding using computation between cryptography and steganography is presented. The proposed system aim to hide information (data file) using computation between cryptography and steganography with in computation area which is unused area two and image page of any execution file (exe.file), to increase the degree of security and the amount of hidden data within exe file without change the size of cover file, to 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. The system includes two main functions; first is the hiding of the information in the with in computation area which is unused area two and image page 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). The proposed system is implemented by using Java.**

Index Terms—**Information Hiding, portable executable file, Steganography, Statistical Technique.**

Manuscript received November 1, 2009.

A. A. Zaidan – PhD candidate, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, phone: +60172452457, Postcode: 50603 and Email: aws.alaa@gmail.com or aws.alaa@yahoo.com.

B. B. Zaidan – PhD candidate, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, bilal@perdana.um.edu.my.

Dr. Hamid.A.Jalab- Senior Lecturer, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, Email:hamidjalab@um.edu.my

I. Introduction

Steganography is the idea of hiding private or sensitive data or information within something that appears to be nothing out of the normal. Steganography and cryptology are similar in the way that they both are used to protect important information [1]. The difference between the two is that Steganography involves hiding information so it appears that no information is hidden at all. Nowadays the term "Information Hiding" relates to both watermarking and steganography [2]. Watermarking is the technique use to hides information in a digital object (video, audio or image) so that information is robust to adjustments or alterations [1],[2]. By watermarking, the mark itself is invisible or unnoticeable for the human vision system. In addition, it should be impossible to remove a watermark without degrading the quality of the data of the digital object [3]. The important application of watermarking is to copyright protection systems, which are intended to prevent unauthorized copying of digital media (pirating). For example if the digital signal (audio, pictures or video) is copied, then the information is also carried in the copy [2], [3]. On the other hand, the main goal of steganography is to hide secret information in the other cover media (video, audio or image) so that other persons will not notice the presence of the information [2],[3]. This is a major distinction between this method and the other methods of covert exchange of information because, for example, in cryptography, the individuals notice the information by seeing the coded information but they will not be able to comprehend the information. However, in steganography, the existence of the information in the sources will not be noticed at all. Although steganography is separate and different from cryptography, but they are related in the way that they both are used to protect valuable information [3]. From here emerged the urgent need to find new techniques alternative organization to overcome these weaknesses, giving rise to conceal information technology (Information Hiding), which are based on a different principle to the idea of organization, where they are buried information (Information Embedding) within other media carrier, and making them aware (Imperceptible) by hackers and attackers, and so are the public domain of information to users of the network, while

the content monopoly "on the relevant agencies, which alone knows how to extract content .Nowadays, protection framework can be classified into more specific as hiding information (Steganography) or encryption information (Cryptography) or a combination between them[4].

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. Steganography and Cryptography are in fact complementary techniques [4]. 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 [4],[5]. 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 [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 [5].

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:

- Executable file types.
- Characteristics of executable files.
- 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 [6]:

EXE (DOS"MZ")

DOS-MZ was introduced with MS-DOS (not DOS v1 though) as a companion to the simplified DOS COM file format. DOS-MZ was designed to be run in real mode and having a relocation table of SEGMENT: OFFSET pairing. A very simple format that can be run at any offset, it does not distinguish between TEXT, DATA and BSS. The maximum file size of (code + data + bss) is one-mega bytes in size. Operating systems that use are: DOS, Win*, Linux DOS.

1) EXE (win 3.xx "NE"):

The WIN-NE executable formatted designed for windows 3.x is the "NE" new-executable. Again, a 16-bit format, it alleviates the maximum size restrictions that the DOZ-MZ has. Operating system that uses it is: windows 3.xx

2) EXE (OS/2 "LE"):

The "LE" linear executable format was designed for IBM's OS/2 operating system by Microsoft. Supporting both 16 and 32-bit segments Operating systems that are used in: OS/2, DOS.

3) EXE (win *9x/NT "PE"):*

With windows 95/NT a new executable file type is required, thus was born the "PE" portable executable. Unlike its predecessors, the WIN-PE is a true 32-bit file format, supporting reloadable code. It does distinguish between TEXT, DATA, and BSS. It is in fact, a bastardized version of the common object file format (COFF) format. Operating systems that use it are: windows 95/98/NT/2000/ME/CE/XP.

4) ELF (Execu*table Linkable Format)*

The ELF was designed by SUN for use in their UNIX clone. A very versatile file format, it was later picked up by many other operating systems for use as both executable files and as shared library files. It does distinguish between TEXT, DATA and BSS.

TEXT: the actual executable code area.
DATA: "initialized" data, (Global Variables).
BSS : "un- initialized" data, (Local Variables).

B. Characteristics of Executable Files

The characteristics of the Executable file does not have a standard size, like other files, for example the image file (BMP) the size of this file is between (2-10 MB), Other example is the text file (TEXT) the size often is less than 2 MB.Through our study the characteristics of files have been used as a cover, it found that lacks sufficient size to serve as a cover for information to be hidden. For these features of the Executable file, it has unspecified size; it can be 650 MB like window setup File or 12 MB such as installation file of multi-media players. For taking advantage of this feature (disparity size) make it a suitable environment for concealing information without detect the file from attacker and discover hidden information in this file [5],[6].

C. 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 [7]. 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 [7]. 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][7].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 [7].

D. Techniques Related with PE

Before looking inside the PE file, we should know special techniques some of which are [8]:

5) 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[8].

6) 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 [7],[8].

7) 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 [8].

E. PE File Layout

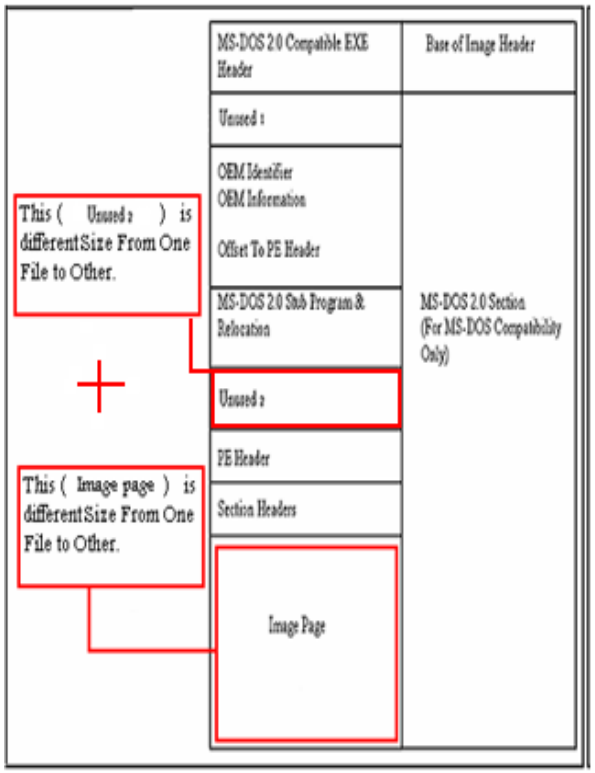

Figure 1.Typical 32-bit Portable .EXE File Layout

III. CRYPTOGRAPHY

A. Block Cipher

In cryptography, a block cipher is a symmetric key cipher which operates on fixed-length groups of bits, termed blocks, with an unvarying transformation. When encrypting, a block cipher might take a (for example) 128-bit block of plaintext as input, and outputs a corresponding 128-bit block of cipher text. The exact transformation is controlled using a second input — the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of cipher text together with the secret key, and yields the original 128-bit block of plaintext. To encrypt messages longer than the block size (128 bits in the above example), a mode of operation is used. Block ciphers can be contrasted with stream ciphers; a stream cipher operates on individual digits one at a time and the transformation varies during the encryption. The distinction between the two types is not always clear-cut: a block cipher, when used in certain modes of operation, acts effectively as a stream cipher as shown in Figure 2 [8].

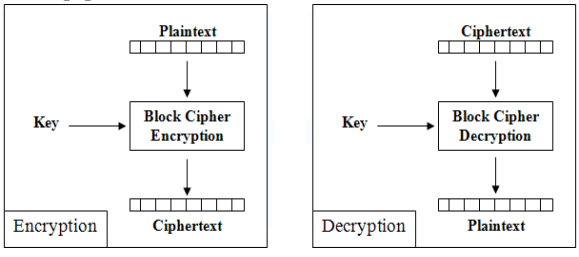

Figure 2. Encryption and Decryption

An early and highly influential block cipher design is the Data Encryption Standard (DES). The (DES) is a cipher (a method for encrypting information) selected as an official

Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally. The algorithm was initially controversial, with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny, and motivated the modern understanding of block ciphers and their cryptanalysis. DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; DES keys have been broken in less than 24 hours. There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks [1][8]. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES)[4],[5].

B. Advanced Encryption Standard

Advance Encryption Standard (AES) and Triple DES (TDES or 3DES) are commonly used block ciphers. Whether you choose AES or 3DES depend on your needs. In this section it would like to highlight their differences in terms of security and performance [3].Since 3DES is based on DES algorithm, it will talk about DES first. DES was developed in 1977 and it was carefully designed to work better in hardware than software. DES performs lots of bit manipulation in substitution and permutation boxes in each of 16 rounds. For example, switching bit 30 with 16 is much simpler in hardware than software. DES encrypts data in 64 bit block size and uses effectively a 56 bit key [4]. 56 bit key space amounts to approximately 72 quadrillion possibilities. Even though it seems large but according to today's computing power it is not sufficient and vulnerable to brute force attack. Therefore, DES could not keep up with advancement in technology and it is no longer appropriate for security. Because DES was widely used at that time, the quick solution was to introduce 3DES which is secure enough for most purposes today.3DES is a construction of applying DES three times in sequence. 3DES with three different keys (K1, K2 and K3) has effective key length is 168 bits (The use of three distinct key is recommended of 3DES.) [4]. Another variation is called two-key (K1 and K3 is same) 3DES reduces the effective key size to 112 bits which is less secure. Two-key 3DES is widely used in electronic payments industry [5],[6]. 3DES takes three times as much CPU power than compare with its predecessor which is significant performance hit. AES outperforms 3DES both in software and in hardware [8]. The Rijndael algorithm has been selected as the Advance Encryption Standard (AES) to replace 3DES. AES is modified version of Rijndael algorithm. Advance Encryption Standard evaluation criteria among others was:

- Security
- Software & Hardware performance
- Suitability in restricted-space environments
- Resistance to power analysis and other implementation attacks.

Rijndael was submitted by Joan Daemen and Vincent Rijmen. When considered together Rijndael combination of security, performance, efficiency, implement ability, and flexibility made it an appropriate selection for the AES.By design AES is faster in software and works efficiently in hardware. It works fast even on small devices such as smart phones; smart cards etc.AES provides more security due to larger block size and longer keys.AES uses 128 bit fixed block size and works with 128, 192 and 256 bit keys.Rigndael algorithm in general is flexible enough to work with key and block size of any multiple of 32 bit with minimum of128 bits and maximum of 256 bits.AES is replacement for 3DES according to NIST both ciphers will coexist until the year2030 allowing for gradual transition to AES.Even though AES has theoretical advantage over 3DES for speed and efficiency in some hardware implementation 3DES may be faster where support for 3DES is mature [1][2][5].

IV. STEGANOGRAPHY

A. General Steganography System

A general Steganography system is shown in Figure 3. It is assumed that the sender wishes to send via Steganography transmission, a message to a receiver. The sender starts with a cover message, which is an input to the stego-system, in which the embedded message will be hidden. The hidden message is called the embedded message. A Steganography algorithm combines the cover massage with the embedded message, which is something to be hidden in the cover The algorithm may, or may not, use a Steganography key (stego key), which is additional secret data that may be needed in the hidden process. The same key (or related one) is usually needed to extract the embedded massage again. The output of the Steganography algorithm is the stego message. The cover massage and stego message must be of the same data type, but the embedded message may be of another data type. The receiver reverses the embedding process to extract the embedded message [4].

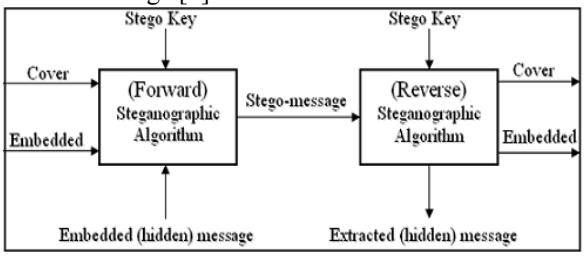

Figure 3: General Steganography System

B. Characterization of Steganography Systems

Steganography techniques embed a message inside a cover. Various features characterize the strength and weaknesses of the methods. The relative importance of each feature depends on the application [7].

8) apacity

The notion of capacity in data hiding indicates the totCal number of bits hidden and successfully recovered by the Stego system.

9) Robustness

Robustness refers to the ability of the embedded data to remain intact if the stego-system undergoes transformation,

such as linear and non-linear filtering; addition of random noise; and scaling, rotation, and loose compression.

10) Undetectable

The embedded algorithm is undetectable if the image with the embedded message is consistent with a model of the source from which images are drawn. For example, if a Steganography method uses the noise component of digital images to embed a secret message, it should do so while not making statistical changes to the noise in the carrier.Undetectability is directly affected by the size of the secret message and the format of the content of the cover image.

11) Invisibility Perceptual Transparency

This concept is based on the properties of the human visual system or the human audio system. The embedded information is imperceptible if an average human subject is unable to distinguish between carriers that do contain hidden information and those that do not. (Ross, 2005) It is important that the embedding occurs without a significant degradation or loss of perceptual quality of the cover.

12) Security

It is said that the embedded algorithm is secure if the embedded information is not subject to removal after being discovered by the attacker and it depends on the total information about the embedded algorithm and secret key.

C. Statistical Steganography Techniques

Statistical steganography techniques utilize the existence of "1-bits" Steganography schemes, which embed one bit of information in a digital carrier. This is done by modifying the cover in such a way that some statistical characteristics change significantly if a "1" is transmitted. Otherwise, the cover is left UN changed. So the receiver must be able to distinguish unmodified covers from modified ones. A cover is divided into l (m) disjoint blocks B1...B l (m)[8]. A secret bit, mi is inserted into the ith block by placing "1" in to Bi if mi=1.Otherwise, the block is not changed in the embedding process. The detection of a specific bit is done via a test function which distinguishes modified block from unmodified block (1)[1],[8]:

$$f(B_i) = \begin{cases} 1 & \text{block } B_i \text{ was modified in the embedding process} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The function f can be interpreted as a hypothesis-test function and the test of null-hypothesis "block Bi was not modified "against the alternative hypothesis "block Bi was modified." Therefore, the whole class of such steganography systems statistical steganography .the receiver successively applies f to alaa cover-block Bi in order to restore every bit of the secret message. The main question which remains to be solved is how such a function f in (8) can be constructed. If they interpret f as a hypothesis-testing function , they can use the theory of hypothesis testing from mathematical statistics .Let us assume could find a formula h(Bi), which depends on some elements of the cover-block Bi, and knew the distribution of h(Bi), in the unmodified block (i.e,the Hypothesis holds in this case) could then use standard procedure to test if h(Bi), equals or exceeds a specific value. If managed to alter h(Bi) in the embedding process in a way that its expected value is 0 if the block Bi was not modified, and expected value is much greater otherwise , could test whether h(Bi) equals zero under the given distribution of h(Bi). Statistical steganography techniques are, however, difficult to apply in many cases. First, a good test statistic h(Bi) must be found which allows distinction between modified and unmodified cover-blocks. Additionally, the distribution of h (Bi) must be known for a "normal" cover; in most cases, this is quite a difficult task. In practical implementations many (quite questionable) assumptions are made in order to determine a closed formula for this distribution. As an example, wanted to construct a statistical steganography algorithm out of pitas' watermarking system, which is similar the patchwork approach of bender et al. Suppose every cover-block Bi is a rectangular set of pixels p(i)n,m .Furthermore, let S={s(i)n,m} be a rectangular pseudorandom binary pattern of equal size, where the number of one is S equals the number of zeros. Would assume that both the sender and receiver have access to S, which represents the stego-key in this application. The sender first splits the image block Bi into two sets, Ci and Di of equal size (i.e., putting all pixels with indices (n,m)into set C where the corresponding key bit n,m equals zero)[2],[8]:

$$\begin{aligned} C_i &= \{p_{n,m}^{(i)} \in B_i | s_{n,m} = 1\} \\ D_i &= \{p_{n,m}^{(i)} \in B_i | s_{n,m} = 0\} \end{aligned} \quad (3)$$

The sender then adds a value k > 0 to all pixels in the subset Ci but leaves all pixels in Di unchanged. In the last step, Ci and Di are merged to form the marked image block Bi. In order to extract the mark, the receiver reconstructs the sets Ci and Di. If the block contains a mark, all value in Ci will be larger than the corresponding values in the embedding step; thus testing the difference of the means of sets Ci and Di. If assumed that all pixels in both Ci and Di are independent identically distributed random variables with an arbitrary distribution, the test statistic [2],[8]:

$$q_i = \frac{\overline{C_i} - \overline{D_i}}{\hat{\sigma}_i}$$

with

$$\hat{\sigma}_i = \sqrt{\frac{\mathrm{Var}[C_i] + \mathrm{Var}[D_i]}{|S|/2}} \quad (4)$$

Where $\overline{Ci}$ denotes the mean over all pixels in the set Ci and Var [Ci] the estimated variance of the random variables in Ci , will follow a N(0,1) normal distribution asymptotically due to the central limit theorem . if a mark is embedded in the image block Bi , the expected value of q will be greater than zero. The receiver is thus able to reconstruct the ith secret message bit by testing whether the statistic qi of block Bi equals zero under the N (0, 1) distribution[2],[8].

V. METHODOLOGY

A. System Overview

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 anywhere without the need to build ethic relations with their customers. In this system a solution is suggested for this problem. 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**.**

B. System Concept

The 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.

C. System Features

This system has the following feature*:*

- The hiding operation within computation between unused area two and image page of EXE file, increases the degree of security for the information hiding which is used in the proposed system because the data which is embedded inside the EXE file is not embed directly of EXE file , it will be hiding within unused area two and then image page of EXE file. So the attacker can not be guessing the information hidden.
- The cover file can be executed normally after hiding operation. Because the hidden information already hide in the unused area two and image page 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:
  1) 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 10 000 000 000 000 000 000 000 000 years to try all possible keys for the weakest version.
  2) 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.
- 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 this system within unused area two and image page of EXE file , the EXE file after hiding process is same manufacture of EXE file before hiding process. That is why the EXE file undetectable by UV.

D. 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 system algorithm for hiding operation procedure as shown in Figure 4.
- The following system algorithm for Retract operation procedure as shown in Figure 5.

The following algorithm is the hiding operation procedure:

Procedure: Hide operation.

Input: Hidden file name, cover file name

Output: Stego-File.

- Begin (1)
- Opens the cover file (EXE file)
- Assign a pointer to the end (MS.DOS 2.0 stub program & Relocation), This is before the unused space two of the cover file.
- Write the hidden file name in the unused space two to the cover file.
- Assign a pointer to (EXE file) after hidden file name.
- Encrypt the hidden file.
- Write the encrypt contact to the unused are two of the cover file, This is equivalent 20% from the size of the cover file, after filling this unused space two, assign a pointer to the end (section header), which is before the image page of the cover file.
- Select the image page consider normal image, it consider the cover for data :
  - Begin (2)
  - From the last point of the encrypt data which is written of unused area two, start to write the encrypt contact to the image page of the cover file.
  - End (2)
- End (1)

Figure 4. Shows System Algorithm for Hiding Operation

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 the EXE file.
  - If (1) end of (MS.DOS 2.0 stub program & Relocation) pointer exists.
    - Begin (2) read the name of hidden file.
    - Read the hiding data from unused space two.
    - If (2) the size of data hidden > 20% from the size of cover file before hiding operation.
      - Begin (3)
      - Read the hiding data from the image page of the cover file respectively after the read from unused area two.
      - End (3)
    - Else (2)
    - 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 (1)
- Display a message (no hiding file).
- End (1).

Figure 5. Shows System Algorithm for Retract Operation.

VI. Conclusion

The .EXE 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 paper,

because most users of any system cannot alter or modify the content of these files. We get the following conclusions:

- 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, so the approach of the proposed system is to prevent the hidden information to observation of these systems.
- The encryption of the message increases the degree of security of hiding technique which is used in the proposed system.
- One of important conclusion 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.
- The cover file can be executed normally after hiding operation. Other word the cover file still natural, working normally and not affected.

ACKNOWLEDGEMENT

Thanks in advance for the entire worker in this project, and the people who support in any way, also I want to thank UM for the support which came from them.

REFERENCES

[1] A.A.Zaidan, B.B.Zaidan, Fazidah Othman, "New Technique of Hidden Data in PE-File with in Unused Area One", International Journal of Computer and Electrical Engineering (IJCEE), Vol.1, No.5, ISSN: 1793-8198, 2009, pp 669-678.

[2] 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 Conference on Computer Engineering and Applications (ICCEA09), Telecom Technology and Applications (TTA), Vol.19, Session 6, p.p 482-489, ISBN: 978-1-84626-017-9,June 6 (2009), Manila, Philippines

[3] A.W. Naji, Shihab A. Hameed, B.B.Zaidan, Wajdi F. Al-Khateeb, Othman O. Khalifa, A.A.Zaidan and Teddy S. Gunawan, " Novel Framework for Hidden Data in the Image Page within Executable File Using Computation between Advance Encryption Standared and Distortion Techniques", International Journal of Computer Science and Information Security (IJCSIS), Vol. 3,No 1 ISSN: 1947-5500, 2009, P.P 73-78.

[4] Alaa Taqa, A.A Zaidan, B.B Zaidan ,"New Framework for High Secure Data Hidden in the MPEG Using AES Encryption Algorithm", International Journal of Computer and Electrical Engineering (IJCEE),Vol.1 ,No.5, ISSN: 1793-8198,2009, pp.589-595 .

[5] A.W.Naji, A.A.Zaidan, B.B.Zaidan, Shihab A, Othman O. Khalifa, " Novel Approach of Hidden Data in the (Unused Area 2 within EXE File) Using Computation Between Cryptography and Steganography ", International Journal of Computer Science and Network Security (IJCSNS) , Vol.9, No.5 , ISSN : 1738-7906,2009, pp. 294-300.

[6] A.W. Naji, A.A.Zaidan, B.B.Zaidan, Ibrahim A.S.Muhamadi, "Novel Approach for Cover File of Hidden Data in the Unused Area Two within EXE File Using Distortion Techniques and Advance Encryption Standard.", Academic and Scientific Research Organizations (WASET), International Conference on Computer, Electrical, and Systems Science, and Engineering (CCESSE09), ISSN:2070-3724,2009.

[7] A.W. Naji, A.A.Zaidan, B.B.Zaidan, Ibrahim A.S.Muhamadi, "New Approach of Hidden Data in the portable Executable File without Change the Size of Carrier File Using Distortion Techniques", Academic and Scientific Research Organizations (WASET), International Conference on Computer, Electrical, and Systems Science, and Engineering(CCESSE09), , ISSN:2070-3724,2009.

[8] B.B.Zaidan, A.A.Zaidan, Fazidah. Othman, Ali Rahem," Novel Approach of Hidden Data in the (Unused Area 1 within EXE File) Using Computation Between Cryptography and Steganography ", Academic and Scientific Research Organizations (WASET), International Conference on Cryptography, Coding and Information Security (ICCCIS09), Vol.41, Session 24, ISSN: 2070-3740,2009.

Aos Alaa Zaidan - He obtained his 1st Class Bachelor degree in Computer Engineering from university of Technology / Baghdad followed by master in data communication and computer network from University of Malaya. He led or member for many funded research projects and He has published more than 45 papers at various international and national conferences and journals, he has done many projects on Steganography for data hidden through different multimedia carriers image, video, audio, text, and non multimedia carrier unused area within exe.file, Quantum Cryptography and Stego-Analysis systems, currently he is working on the multi module for Steganography. He is PhD candidate on the Department of Computer System & Technology / Faculty of Computer Science and Information Technology/University of Malaya /Kuala Lumpur/Malaysia. He is members IAENG, CSTA, WASET, and IACSIT. He is reviewer in the (IJSIS, IJCSNS, IJCSN and IJCSE).

Bilal Bahaa Zaidan - he obtained his bachelor degree in Mathematics and Computer Application from Saddam University/Baghdad followed by master from Department of Computer System & Technology Department Faculty of Computer Science and Information Technology/University of Malaya /Kuala Lumpur/Malaysia, He led or member for many funded research projects and He has published more than 45 papers at various international and national conferences and journals. 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 and multi module for Steganography, he is PhD candidate on the Department of Computer System & Technology / Faculty of Computer Science and Information Technology/University of Malaya /Kuala Lumpur/Malaysia. He is members IAENG, CSTA, WASET, and IACSIT. He is reviewer in the (IJSIS, IJCSNS, IJCSN and IJCSE).

Dr.Hamid.A.Jalab Received his B.Sc degree from University of Technology, Baghdad, Iraq. MSc & Ph.D degrees from the State University of Technology, Odessa, Ukraine 1987 and 1991, respectively. Presently, Visiting Senior Lecturer of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His areas of interest include neural networks and cryptography.